Introduction

This document describes how E+E probes with Modbus RTU interface are connected to Beckhoff hardware and how they communicate by means of TwinCAT. Only the properties specific to Modbus communication are described. Programmming knowledge in IEC 61131-3 under TwinCAT is required.

This description is addressed to trained personnel of the control and automation technology that is familiar with the applicable national standards. For installation and commissioning of the components, the observance of the documentation and the following notes and explanations is absolutely necessary. For each installation and commissioning, the qualified personnel is obliged to use the documentation published at that time for each installation and commissioning. The qualified personnel must ensure that the application or use of the described products meets all safety requirements, including all applicable laws, regulations, provisions and standards.

Warranty and Liability

This application example is non-binding and does not claim to be complete with regard to configuration and equipment as well as all eventualities. The application example does not represent customer-specific solutions, but is only intended to provide assistance with typical tasks. You yourself are responsible for the proper operation of the products described. This application example does not release you from the obligation to handle the product safely during application, installation, operation and maintenance. By using this application example, you acknowledge that we cannot be held liable for any damage beyond the liability regulations described. We reserve the right to make changes to this application example at any time without notice. In case of discrepancies between the suggestions in this application example and other E+E publications, such as catalogues, the content of the other documentation takes precedence.
We assume no liability for the information contained in this document.

Trademarks

Beckhoff®, TwinCAT® and EtherCAT® are registered and licensed trademarks of Beckhoff Automation Ltd.

Copyright

Modbus RTU Interface

The Modbus RTU is used to read data from or write data to permanently defined data areas of a device. The information about which data are in which data area varies from device to device. To be able to address the Modbus RTU, the Modbus settings must first be defined (baud rate, parity and stop bits).
Communication is based on the master-slave principle. The communication always starts from the master by a request. Each slave has an address which must be assigned once. If a slave recognizes that it has been addressed by the master, it sends a reply. The slaves cannot communicate with each other. Nor can they start communication with the master.

Hardware

In this demo application the following control components from the control manufacturer Beckhoff® were used:

- CX9020 Compact CPU (CX9020-0111/1GB)
- EL6021 RS422/RS485 interface terminal
- EL9010 EtherCat end terminal

Connected E+E probes with Modbus RTU interface:

- EE072 humidity and temperature probe

- EE872 CO_2 probe

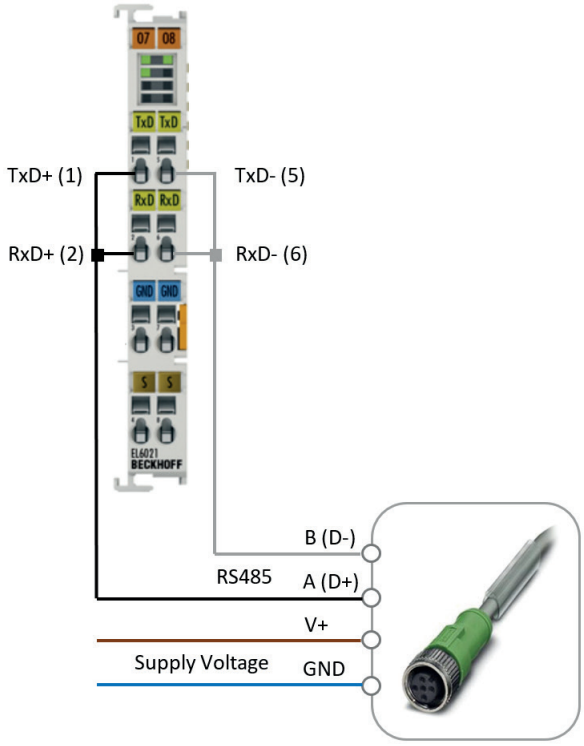
- EE741 inline flow meter

Software

The TwinCAT project uses the following system software:
- TwinCAT 2 PLC Control Version: v2.11.0 (Build 2618)
- TS6255 TwinCat PLC Modbus RTU (software library)

Electrical connection of the sensors

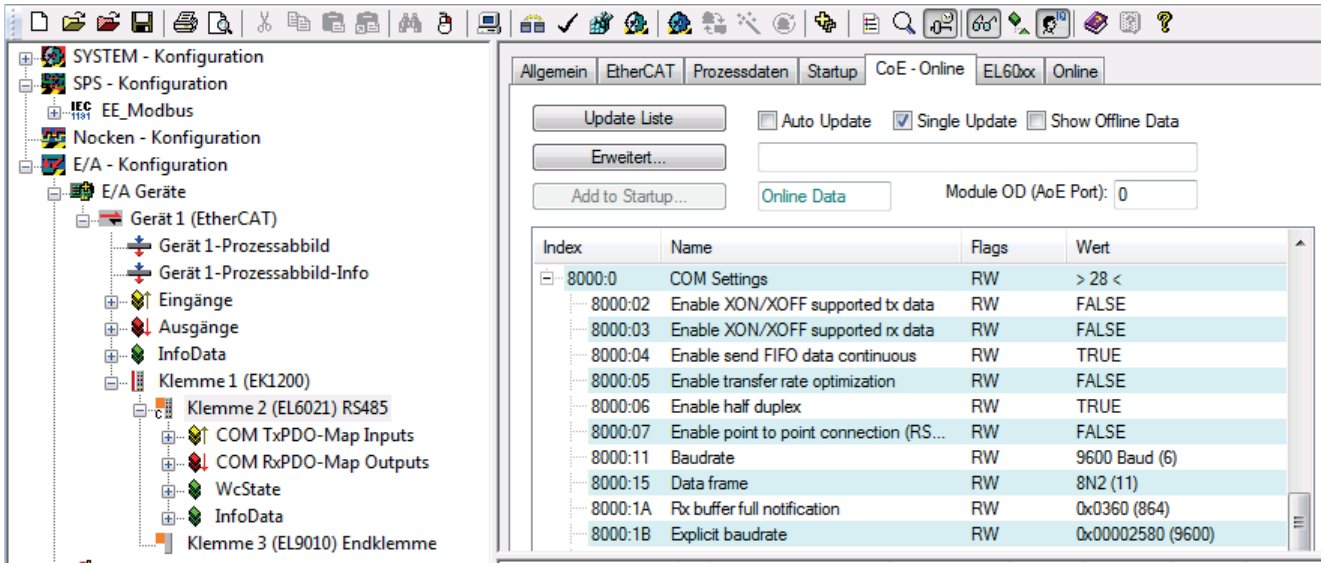
The E+E probes used are connected to the supply voltage and to the RS485 terminal (EL6021) by means of a four-pin and a five-pin M12 connector, respectively. The correct pin assignment and the permitted supply voltage can be found in the respective data sheet.



System Manager

The RS485 interface parameters need to be set in the System Manager. This is done at the EL6021 terminal using CoE online parameters. In this example, the following parameters are used:

➔	8000:04	Enable send FIFO data continuous	TRUE
➔	8000:06	Enable half duplex	TRUE
➔	8000:11	Baud rate	9600 Baud
➔	8000:15	Data frame	8N2 (8 data bits, no parity, 2 stop bits)



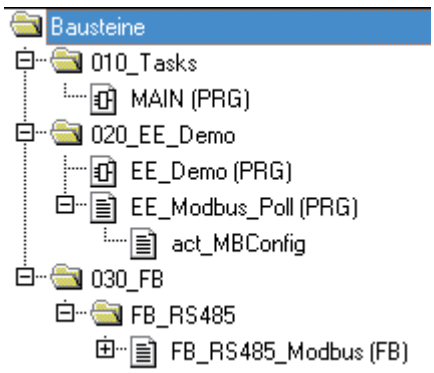
The controller should be restarted after the parameters have been set.

Programming

Integration in TwinCAT

This example describes how a simple PLC programme for E+E sensors can be written in TwinCAT and how it is linked to the hardware. It shall be possible to read out and visualise process data from the sensors by means of simple configuration.

Building Blocks



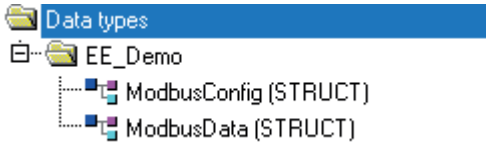
- MAIN (PRG)
  - Programme call around standard task (10 ms)

- EE_Demo (PRG)
  - Is called from MAIN (= E+E Demo Programme)

- EE_Modbus_Poll (PRG)
  - Modbus configuration list ”act_MBConfig
  - Cyclic readout of the configured process data

- FB_RS485_Modbus (FB)
  - Modbus master: Communication to the sensors via the RS485 terminal (EL6021)

Data Types



- ModbusConfig(STRUCT)

Configuration: Modbus setup data

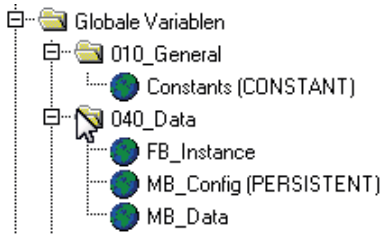
```
TYPE ModbusConfig :
STRUCT
    (* ModbusConfig *)
    MB_Address      :   BYTE := 0              ;   (* Modbus Address of  Slave (1-247) *)
    MB_Function     :   BYTE := 16#00          ;   (* Modbus Function Code:
                                                       16#01 => Read Coil Register,
                                                       16#03 => Read Holding Registers
                                                       16#04 => Read Input Register
                                                       16#05 => Write Coil Register
                                                       16#06 => Write Single Register
                                                       16#10 => Write Multiple Registers *)
    MB_Register     :   WORD := 16#0           ;   (* Data-Address from Slave *)
    MB_Quantity     :   WORD := 16#0           ;   (* Quantity of Registers (WORDS OR BITS) *)
    DataType        :   BYTE := 0              ;   (* 0=BYTE, 1=WORD, 2=INT, 3=REAL, 4=CHAR, 5=LREAL *)
    Para            :   BYTE := 2#00000000     ;   (* Parameter Options (byte order correction):
                                                       Para.0 => TRUE=WORD rotation
                                                       Para.1 => TRUE=DWORD rotation
                                                       *)
    (* Information Text *)
    sDescription    :   STRING                 ;   (* Data Description (only for information) *)
    sUnit           :   STRING                 ;   (* Data Unit  (only for information) *)
END_STRUCT
END_TYPE
```

- ModbusData(STRUCT)

Process Data: Read process data and status information

```
TYPE ModbusData :
STRUCT
    (* ModbusData from Slave*)
    sInfo           : STRING                   ;   (* Information/Debug Data *)
    bOk             : BOOL                     ;   (* Values are valid *)
    bError          : BOOL                     ;   (* Values error *)
    sDat            : STRING[255]              ;   (* STRING data from transmitter *)
    rDat            : ARRAY[1..10] OF REAL     ;   (* REAL data from transmitter *)
    lDat            : ARRAY[1..5] OF LREAL     ;   (* LREAL data from transmitter *)
    iDat            : ARRAY[1..20] OF INT      ;   (* INT data from transmitter *)
    wDat            : ARRAY[1..20] OF WORD     ;   (* WORD data from transmitter *)
    dDat            : ARRAY[1..10] OF DWORD;   (* DWORD data from transmitter *)
    bDat            : ARRAY[1..40] OF BYTE     ;   (* BYTE data from transmitter *)
END_STRUCT
END_TYPE
```

Resources / Global Variables



- Constants (CONSTANT)

```
VAR_GLOBAL CONSTANT
    MaxMbID     :    INT := 20;              (* Last polling Index = ARRAY-SIZE *)
END_VAR
```

- FB_Instance

```
VAR_GLOBAL
    fbRS485_Modbus :    FB_RS485_Modbus;    (* FB Instance *)
END_VAR
```

- MB_Config (PERSISTENT)

  Configuration: Modbus setup data

```
VAR_GLOBAL PERSISTENT
    MBConfig    :    ARRAY[1..MaxMbID] OF ModbusConfig    ;    (* Modbus Config *)
END_VAR
```

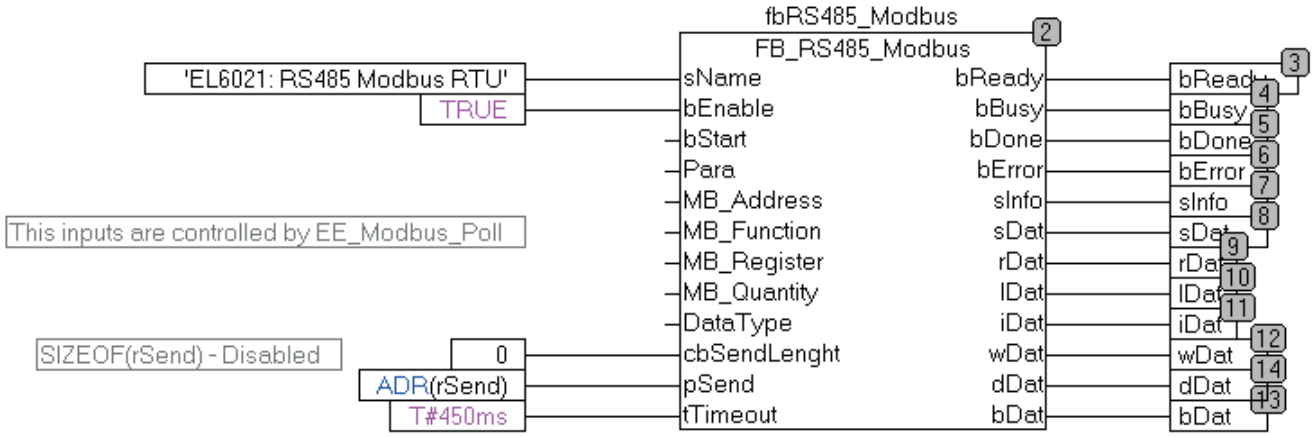
- MB_Data

  Process data: Readout process data and status information

```
VAR_GLOBAL
    MBData :    ARRAY[1..MaxMbID] OF ModbusData    ;    (* Modbus Data *)
END_VAR
```

FB_RS485_Modbus

This function block is used for communication with the RS485 terminal (EL6021). The ”ModbusRtuMaster_ KL6x22B” block from the Beckhoff library ”ModbusRTU.lib” is used internally. It also provides the communication variables to the system manager.



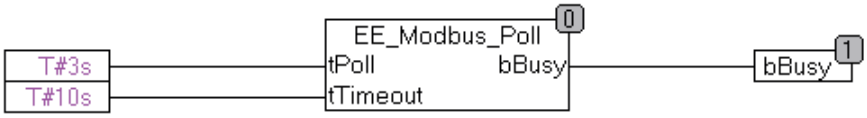
VAR_INPUT

```
sName    :   STRING := 'FB_RS485_Modbus' ;   (* Name of Functionblock *)
bEnable  :   BOOL := TRUE                ;   (* External enable *)

bStart   :   BOOL := FALSE               ;   (* Start Command *)
Para     :   BYTE := 2#00000000          ;   (* Parameter Options (byte order correction):
                                                Para.0 => TRUE=WORD rotation
                                                Para.1 => TRUE=DWORD rotation
                                                Para.2 => nc
                                                Para.3 => nc
                                                ...
                                                *)
(* ModbusConfig *)
MB_Address       :   BYTE := 247         ;   (* Modbus Address of  Slave (1-247) *)
MB_Function      :   BYTE := 16#03       ;   (* Modbus Function Code:
                                                16#01 => Read Coil Register,
                                                16#03 => Read Holding Registers
                                                16#04 => Read Input Register
                                                16#05 => Write Coil Register
                                                16#06 => Write Single Register
                                                16#10 => Write Multiple Registers *)
MB_Register      :   WORD := 16#19       ;   (* Data-Address from Slave *)
MB_Quantity      :   WORD := 16#02       ;   (* Quantity of Registers (WORDS OR BITS) *)
DataType         :   BYTE := 3           ;   (* Data Type:
                                                0=BYTE
                                                1=WORD
                                                2=INT
                                                3=REAL
                                                4=CHAR
                                                5=LREAL (64bit double)
                                                6=DWORD *)
(* Send-Data *)
cbSendLenght     :   UINT                ;   (* Send-Data-Lenght -> SIZEOF(SendData) *)
pSend            :   DWORD               ;   (* Send-Data-ADR -> ADR(SendData) *)

tTimeout :   TIME := T#400MS             ;   (* Communication timeout *)
```

VAR_OUTPUT

```
bReady      : BOOL                  ;   (* Communication ready -> Ready for bStart *)
bBusy       : BOOL                  ;   (* Communication busy *)
bDone       : BOOL                  ;   (* Communication done (no error) *)
bError      : BOOL                  ;   (* Communication error *)
sInfo       : STRING                ;   (* Information/Debug Data *)
sDat        : STRING[255]           ;   (* STRING data from transmitter *)
rDat        : ARRAY[1..10] OF REAL  ;   (* REAL data from transmitter *)
lDat        : ARRAY[1..5] OF LREAL  ;   (* LREAL data from transmitter *)
iDat        : ARRAY[1..20] OF INT   ;   (* INT data from transmitter *)
wDat        : ARRAY[1..20] OF WORD  ;   (* WORD data from transmitter *)
dDat        : ARRAY[1..10] OF DWORD;    (* DWORD data from transmitter *)
bDat        : ARRAY[1..40] OF BYTE  ;   (* BYTE data from transmitter *)
```

EE_Modbus_Poll



VAR_INPUT, VAR_OUTPUT

```
VAR_INPUT
    tPoll          :   TIME:= T#2s  ;   (* Polling timer *)
    tTimeout       :   TIME:= T#10s;    (* Timeout: Values not valid *)
END_VAR
VAR_OUTPUT
    bBusy          :   BOOL;            (* Program busy *)
END_VAR
```

This programme has two main tasks:

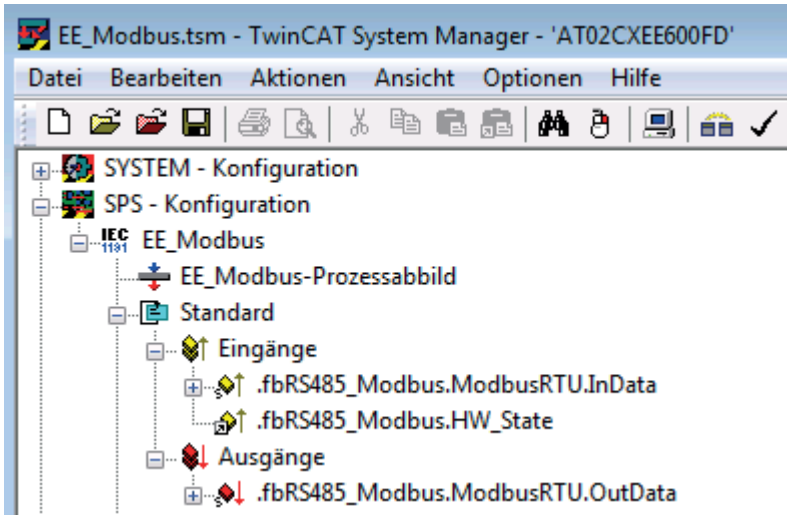
1. Definition of the Modbus communication in the action ”act_MBConfig”. Here the desired process data of the probes are defined and entered in the list ”MBConfig”. When the CPU is restarted, the data is loaded once. ATTENTION: The maximum length of the list can be changed with the global constant ”MaxMbID”.

```
0006  (* ModbusConfig Example*)
0007  i:= 1;
0008  MBConfig[i].sDescription   := 'EE072 Serial Number';   (* Data Description (only for information) *)
0009  MBConfig[i].MB_Address     := 234;                     (* 0=OFF *)    (* Modbus Address of  Slave (1-247) *)
0010  MBConfig[i].MB_Function    := 3;                       (* Modbus Function Code *)
0011  MBConfig[i].MB_Register    := 16#0;                    (* Data-Address from Slave *)
0012  MBConfig[i].MB_Quantity    := 8;                       (* Quantity of Registers (WORDS OR BITS) *)
0013  MBConfig[i].DataType       := 4;                       (* 0=BYTE, 1=WORD, 2=INT, 3=REAL, 4=CHAR, 5=LREAL, 6=DWORD *)
0014  MBConfig[i].sUnit :=       '';                         (* Data Unit  (only for information) *)
0015  MBConfig[i].Para :=        2#00000001;                 (* Para.0 => TRUE=WORD rotation, Para.1 => TRUE=DWORD rotation *)
0016  (*_____________________________________________________________________________________*)
0017  i:= 2;
0018  MBConfig[i].sDescription   := 'EE072 Temperature (float)';   (* Data Description (only for information) *)
0019  MBConfig[i].MB_Address     := 234;                     (* 0=OFF *)    (* Modbus Address of  Slave (1-247) *)
0020  MBConfig[i].MB_Function    := 3;                       (* Modbus Function Code *)
0021  MBConfig[i].MB_Register    := 16#3EA;                  (* Data-Address from Slave *)
0022  MBConfig[i].MB_Quantity    := 2;                       (* Quantity of Registers (WORDS OR BITS) *)
0023  MBConfig[i].DataType       := 3;                       (* 0=BYTE, 1=WORD, 2=INT, 3=REAL, 4=CHAR, 5=LREAL, 6=DWORD *)
0024  MBConfig[i].sUnit :=       '°C';                       (* Data Unit  (only for information) *)
0025  MBConfig[i].Para :=        2#00000000;                 (* Para.0 => TRUE=WORD rotation, Para.1 => TRUE=DWORD rotation *)
0026  (*_____________________________________________________________________________________*)
0027  i:= 3;
0028  MBConfig[i].sDescription   := 'EE072 Rel Humidity (float)';   (* Data Description (only for information) *)
0029  MBConfig[i].MB_Address     := 234;                     (* 0=OFF *)    (* Modbus Address of  Slave (1-247) *)
```

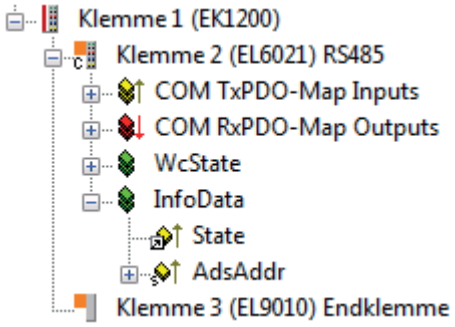
2. The programme uses the configuration list ”MBConfig” and tries to request the process data from the sensors. The result of the communication is entered in the ”MBData” list. The structure also contains information about the validity of the data (bOK, bError), as well as status information for easy diagnosis (sInfo).

Connection to the System Manager

Die TwinCat variables are read in the System Manager …



… and linked to the EL6021:

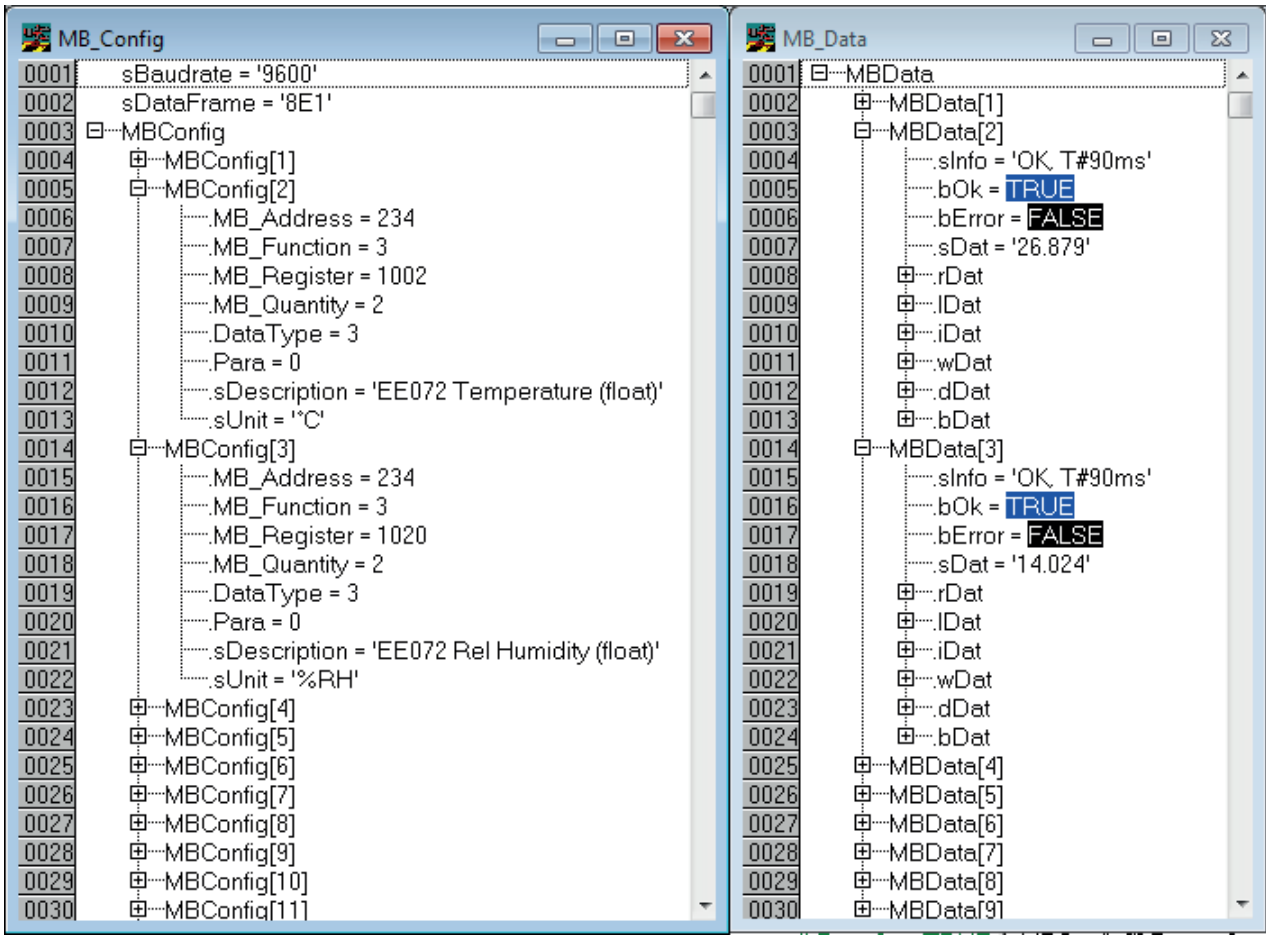


- InData = COM TxPDO-Map Inputs
- HW_State = State
- OutData = COM RxPDO-Map Outputs

After linking correctly, the configuration needs to be activated again.

Online data

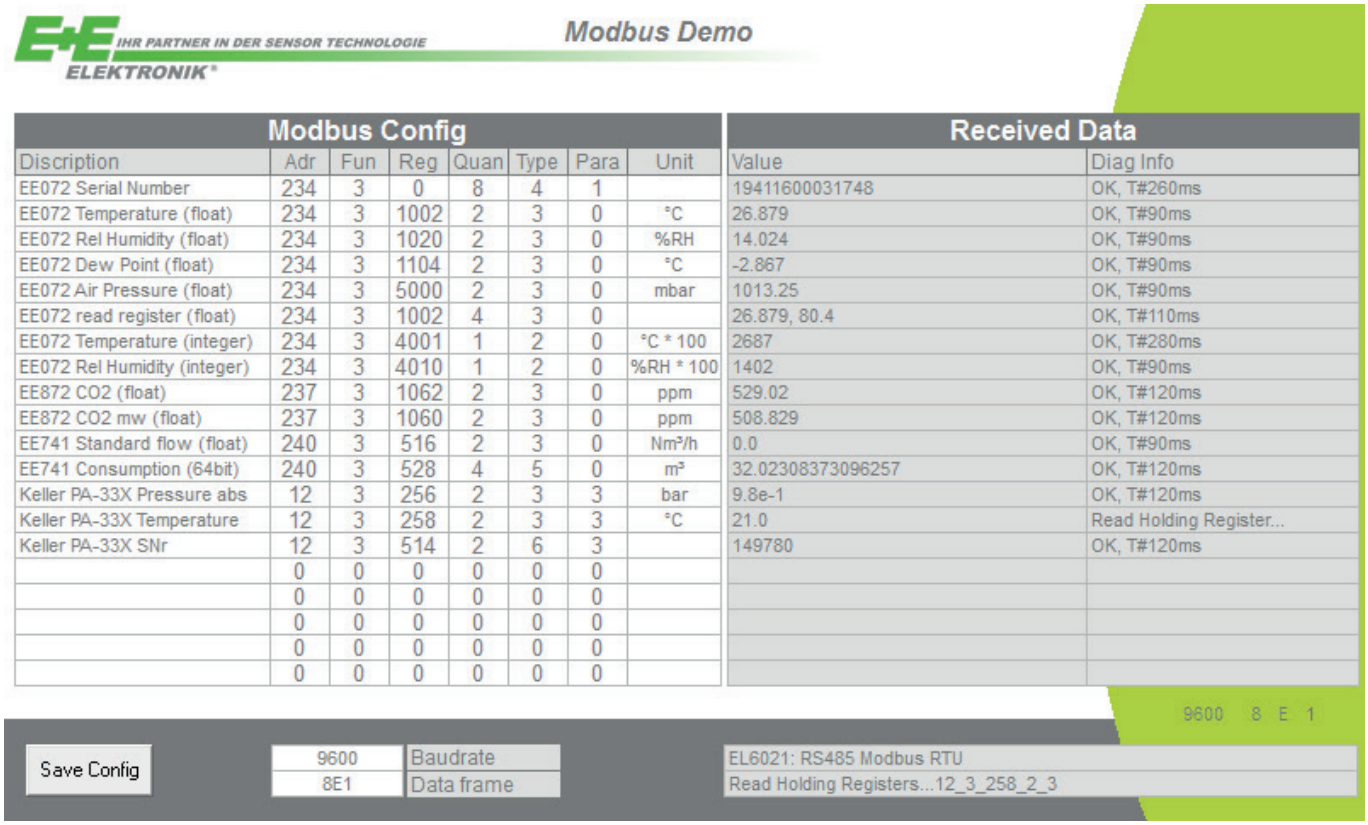
After successful commissioning, the read-out process data can be displayed.
Example index 2: Reading the temperature [°C] of an EE072



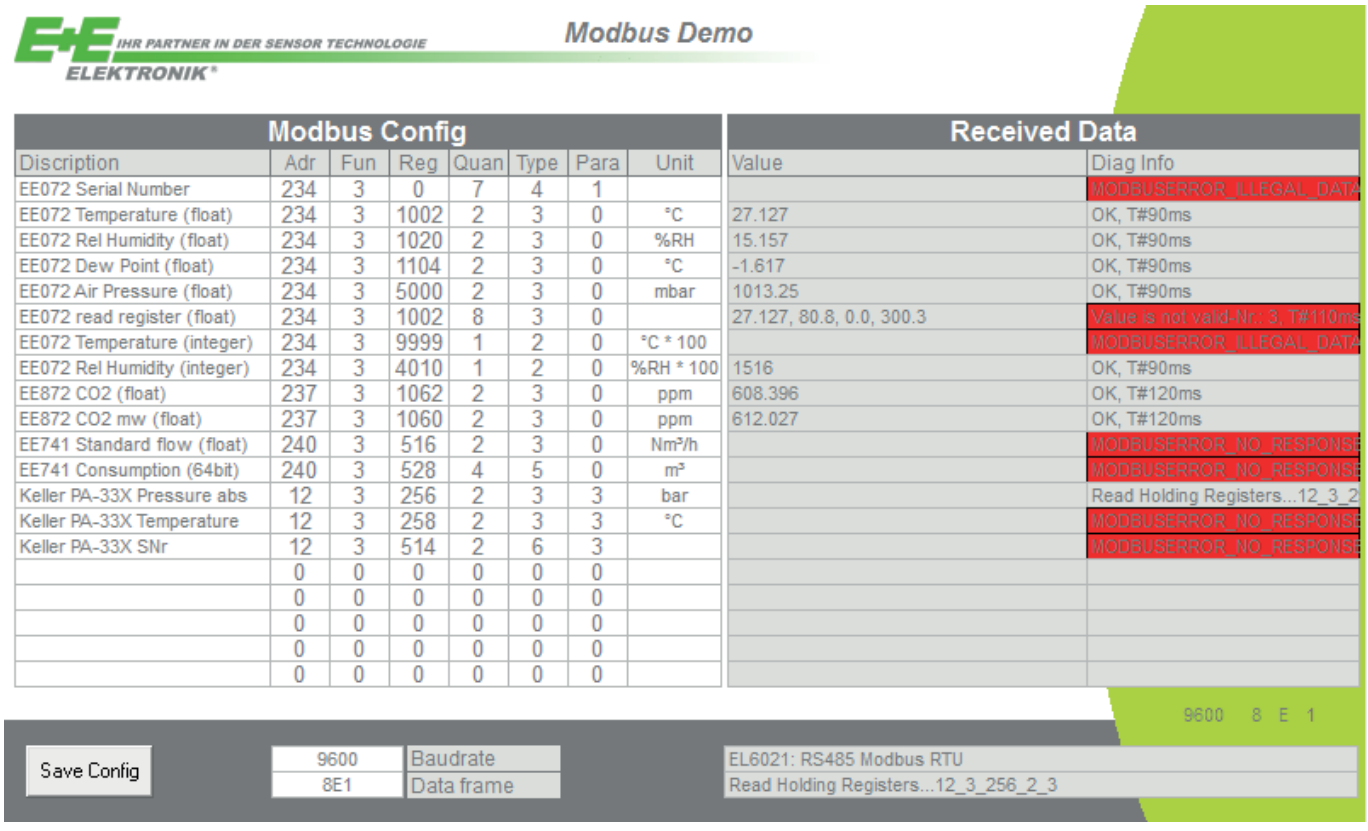
MBConfig[2] MBData[2]

Visualisation

The demo project contains the ”PLC_VISU” visualisation. It visualises the configuration and process data of the E+E probes. Moreover, it offers the possibility for Modbus configuration without the need for programming knowledge. Please find more information on the products’ Modbus register map in the according Quick Guide or User Manual, respectively.



Example 1: All data OK



Example 2: Faulty configuration

Appendix

E+E Elektronik Product Literature

- EE072
  Datasheet: http://downloads.epluse.com/fileadmin/data/product/ee072/datasheet_EE072.pdf
  Quick Guide: http://downloads.epluse.com/fileadmin/data/product/ee072/BA_EE072_short_v1_1.pdf
- EE872
  Datasheet: http://downloads.epluse.com/fileadmin/data/product/ee872/datasheet_EE872.pdf
  Quick Guide: http://downloads.epluse.com/fileadmin/data/product/ee872/BA_EE872_short.pdf
- EE741
  Datasheet: http://downloads.epluse.com/fileadmin/data/product/ee741/datasheet_EE741.pdf
  User Manual: http://downloads.epluse.com/fileadmin/data/product/ee741/BA_EE741_e.pdf

E+E Elektronik's Modbus Application Note

http://downloads.epluse.com/fileadmin/data/product/ee071/AN0103.pdf

Beckhoff

- EL6021
  http://beckhoff.de/el6021/
- TS6255
  http://beckhoff.de/TS6255/
- Beckhoff Information System
  https://infosys.beckhoff.de/