

### 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. Programming 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

© E+E Elektronik GmbH, Austria

Distribution and reproduction of this document, exploitation and communication of its contents are prohibited unless expressly permitted.

Non-compliance obligate to compensation. All rights for the case of patent, utility model or design patent registration reserved.

### 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<sub>2</sub> probe
- EE741 inline flow meter



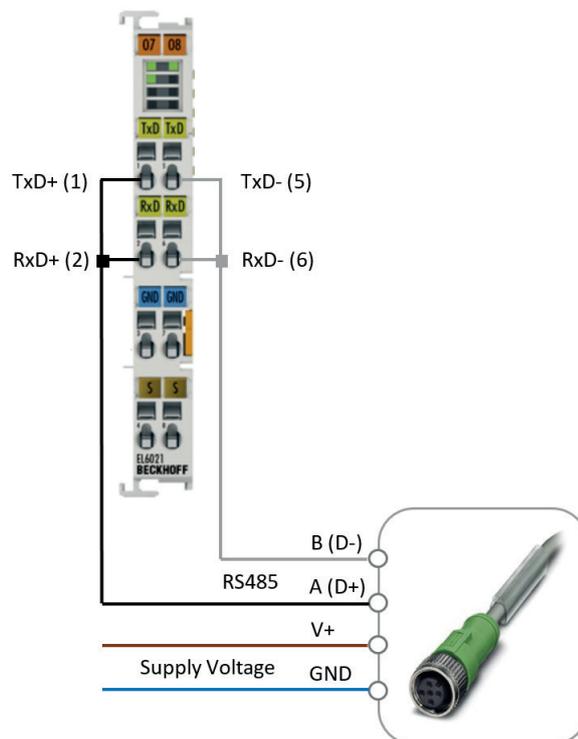
## Software

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

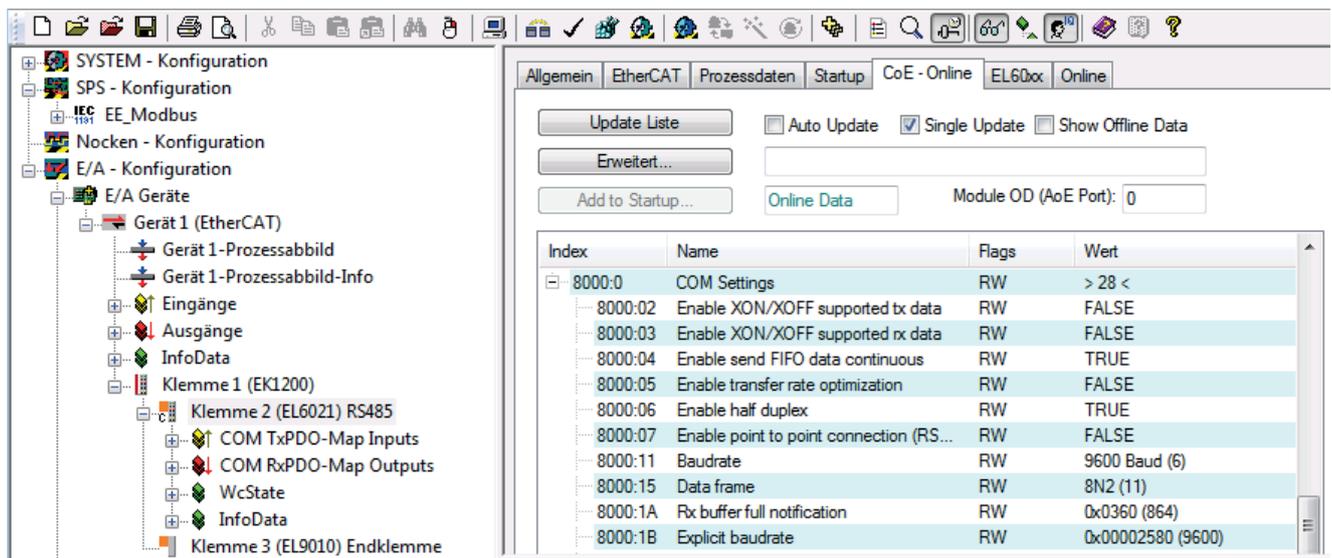
The E+E probes used are connected to the supply voltage and to the RS485 terminal (EL6021) by means of a four-pin and a five-pin M12 connector, respectively. The correct pin assignment and the permitted supply voltage can be found in the respective data sheet.



# 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)



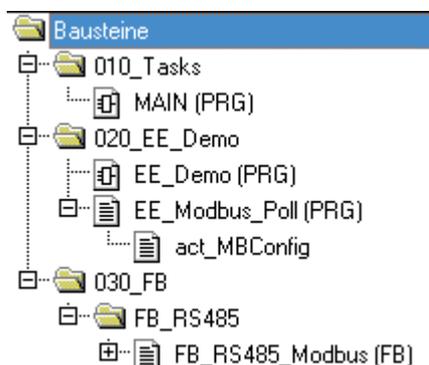
The controller should be restarted after the parameters have been set.

## 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.

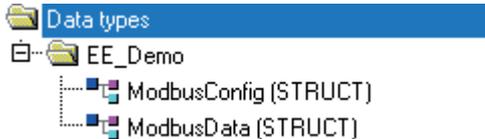
### Building Blocks



- 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

```

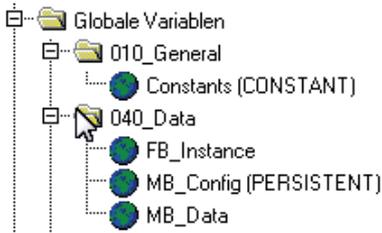
- ModbusData(STRUCT)  
Process Data: Read process data and status information

```

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

```

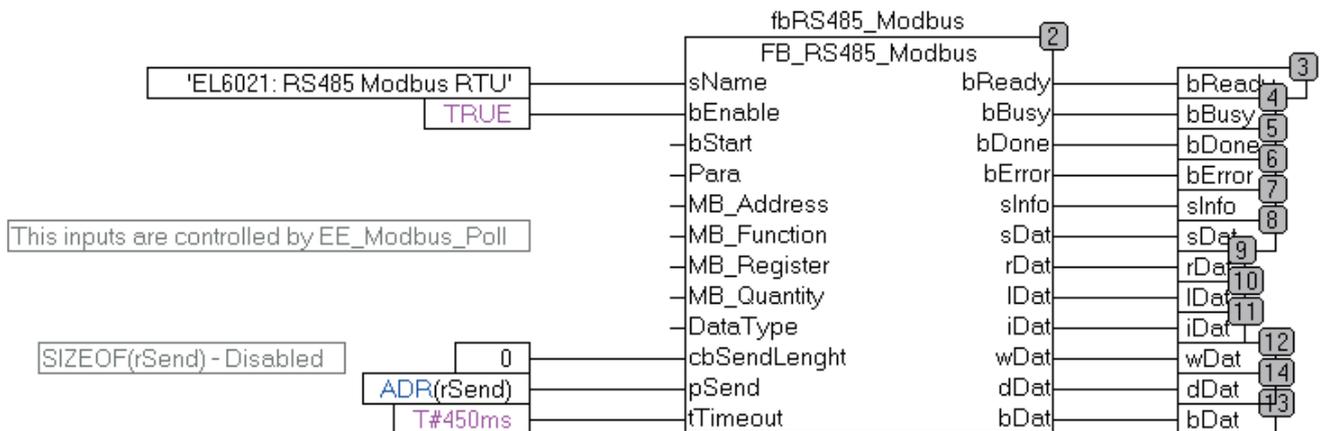
## Resources / Global Variables



- 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 *)

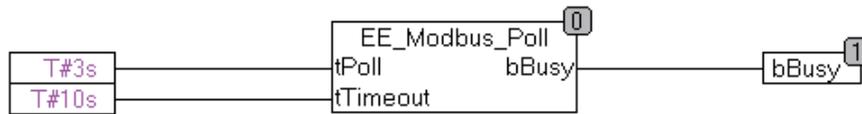
(* Send-Data *)
cbSendLenght : UINT; (* Send-Data-Lenght -> SIZEOF(SendData) *)
pSend : DWORD; (* Send-Data-ADR -> ADR(SendData) *)

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 "MaxMblD".

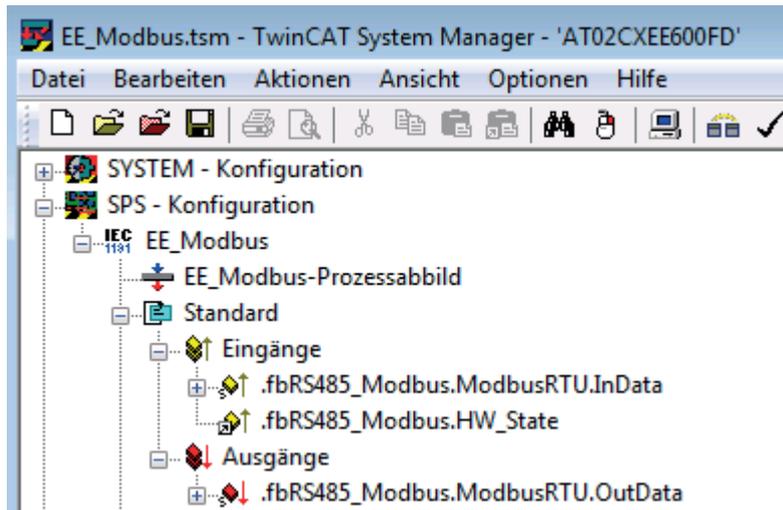
```

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) *)
  
```

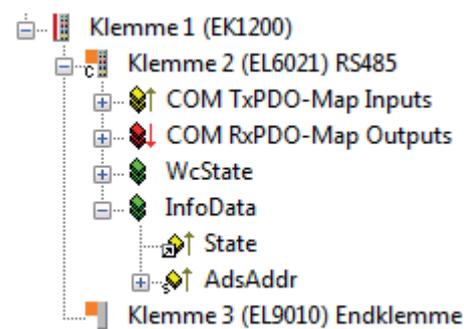
2. The programme uses the configuration list "MBConfig" and tries to request the process data from the sensors. The result of the communication is entered in the "MBData" list. The structure also contains information about the validity of the data (bOK, bError), as well as status information for easy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

The image shows two side-by-side windows from a software interface. The left window, titled 'MB\_Config', displays a tree view of configuration parameters. The right window, titled 'MB\_Data', displays a tree view of read-out data.

**MB\_Config**

- 0001 sBaudrate = '9600'
- 0002 sDataFrame = '8E1'
- 0003 MBConfig
  - 0004 MBConfig[1]
  - 0005 MBConfig[2]
    - 0006 MB\_Address = 234
    - 0007 MB\_Function = 3
    - 0008 MB\_Register = 1002
    - 0009 MB\_Quantity = 2
    - 0010 DataType = 3
    - 0011 Para = 0
    - 0012 sDescription = 'EE072 Temperature (float)'
    - 0013 sUnit = 'C'
  - 0014 MBConfig[3]
    - 0015 MB\_Address = 234
    - 0016 MB\_Function = 3
    - 0017 MB\_Register = 1020
    - 0018 MB\_Quantity = 2
    - 0019 DataType = 3
    - 0020 Para = 0
    - 0021 sDescription = 'EE072 Rel Humidity (float)'
    - 0022 sUnit = '%RH'
  - 0023 MBConfig[4]
  - 0024 MBConfig[5]
  - 0025 MBConfig[6]
  - 0026 MBConfig[7]
  - 0027 MBConfig[8]
  - 0028 MBConfig[9]
  - 0029 MBConfig[10]
  - 0030 MBConfig[11]

**MB\_Data**

- 0001 MBDData
  - 0002 MBDData[1]
  - 0003 MBDData[2]
    - 0004 sInfo = 'OK, T#90ms'
    - 0005 bOk = **TRUE**
    - 0006 bError = **FALSE**
    - 0007 sDat = '26.879'
    - 0008 rDat
    - 0009 IDat
    - 0010 iDat
    - 0011 wDat
    - 0012 dDat
    - 0013 bDat
  - 0014 MBDData[3]
    - 0015 sInfo = 'OK, T#90ms'
    - 0016 bOk = **TRUE**
    - 0017 bError = **FALSE**
    - 0018 sDat = '14.024'
    - 0019 rDat
    - 0020 IDat
    - 0021 iDat
    - 0022 wDat
    - 0023 dDat
    - 0024 bDat
  - 0025 MBDData[4]
  - 0026 MBDData[5]
  - 0027 MBDData[6]
  - 0028 MBDData[7]
  - 0029 MBDData[8]
  - 0030 MBDData[9]

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.



### Modbus Demo

Modbus Config								Received Data	
Discription	Adr	Fun	Reg	Quan	Type	Para	Unit	Value	Diag Info
EE072 Serial Number	234	3	0	8	4	1		19411600031748	OK, T#260ms
EE072 Temperature (float)	234	3	1002	2	3	0	°C	26.879	OK, T#90ms
EE072 Rel Humidity (float)	234	3	1020	2	3	0	%RH	14.024	OK, T#90ms
EE072 Dew Point (float)	234	3	1104	2	3	0	°C	-2.867	OK, T#90ms
EE072 Air Pressure (float)	234	3	5000	2	3	0	mbar	1013.25	OK, T#90ms
EE072 read register (float)	234	3	1002	4	3	0		26.879, 80.4	OK, T#110ms
EE072 Temperature (integer)	234	3	4001	1	2	0	°C * 100	2687	OK, T#280ms
EE072 Rel Humidity (integer)	234	3	4010	1	2	0	%RH * 100	1402	OK, T#90ms
EE872 CO2 (float)	237	3	1062	2	3	0	ppm	529.02	OK, T#120ms
EE872 CO2 mw (float)	237	3	1060	2	3	0	ppm	508.829	OK, T#120ms
EE741 Standard flow (float)	240	3	516	2	3	0	Nm³/h	0.0	OK, T#90ms
EE741 Consumption (64bit)	240	3	528	4	5	0	m³	32.02308373096257	OK, T#120ms
Keller PA-33X Pressure abs	12	3	256	2	3	3	bar	9.8e-1	OK, T#120ms
Keller PA-33X Temperature	12	3	258	2	3	3	°C	21.0	Read Holding Register...
Keller PA-33X SNr	12	3	514	2	6	3		149780	OK, T#120ms
	0	0	0	0	0	0			
	0	0	0	0	0	0			
	0	0	0	0	0	0			
	0	0	0	0	0	0			
	0	0	0	0	0	0			

9600 8 E 1

Save Config

9600

Baudrate

8E1

Data frame

EL6021: RS485 Modbus RTU

Read Holding Registers...12\_3\_258\_2\_3

Example 1: All data OK



### Modbus Demo

Modbus Config								Received Data	
Discription	Adr	Fun	Reg	Quan	Type	Para	Unit	Value	Diag Info
EE072 Serial Number	234	3	0	7	4	1			MODBUSERROR_ILLEGAL_DATA
EE072 Temperature (float)	234	3	1002	2	3	0	°C	27.127	OK, T#90ms
EE072 Rel Humidity (float)	234	3	1020	2	3	0	%RH	15.157	OK, T#90ms
EE072 Dew Point (float)	234	3	1104	2	3	0	°C	-1.617	OK, T#90ms
EE072 Air Pressure (float)	234	3	5000	2	3	0	mbar	1013.25	OK, T#90ms
EE072 read register (float)	234	3	1002	8	3	0		27.127, 80.8, 0.0, 300.3	Value is not valid Nr. 3, T#110ms
EE072 Temperature (integer)	234	3	9999	1	2	0	°C * 100		MODBUSERROR_ILLEGAL_DATA
EE072 Rel Humidity (integer)	234	3	4010	1	2	0	%RH * 100	1516	OK, T#90ms
EE872 CO2 (float)	237	3	1062	2	3	0	ppm	608.396	OK, T#120ms
EE872 CO2 mw (float)	237	3	1060	2	3	0	ppm	612.027	OK, T#120ms
EE741 Standard flow (float)	240	3	516	2	3	0	Nm³/h		MODBUSERROR_NO_RESPONSE
EE741 Consumption (64bit)	240	3	528	4	5	0	m³		MODBUSERROR_NO_RESPONSE
Keller PA-33X Pressure abs	12	3	256	2	3	3	bar		Read Holding Registers...12_3_2
Keller PA-33X Temperature	12	3	258	2	3	3	°C		MODBUSERROR_NO_RESPONSE
Keller PA-33X SNr	12	3	514	2	6	3			MODBUSERROR_NO_RESPONSE
	0	0	0	0	0	0			
	0	0	0	0	0	0			
	0	0	0	0	0	0			
	0	0	0	0	0	0			
	0	0	0	0	0	0			

9600 8 E 1

Save Config

9600

Baudrate

8E1

Data frame

EL6021: RS485 Modbus RTU

Read Holding Registers...12\_3\_256\_2\_3

Example 2: Faulty configuration

# Appendix

## E+E Elektronik Product Literature

- EE072  
Datasheet: [http://downloads.epluse.com/fileadmin/data/product/ee072/datasheet\\_EE072.pdf](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](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](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](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](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](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/>