Communication between Siemens S7-1200 and Vaisala HMP7 probe via Modbus RTU

This guide is meant to be used to help you communicate with an HMP7 probe with a Siemens S7-1200 PLC using Simatic Step 7 v16 software.

In this guide we show you how to wire an HMP7 probe to the communication board (CB 1241 RS485) on a Siemens S7-1200 PLC, communicate between the HMP7 and the S7-1200 via Modbus RTU and convert the holding registers into human readable floating-point values. We use the Siemens S7-1200 as a Modbus master to read the relative humidity and temperature readings from the HMP7 as the Modbus slave. Some tables and diagrams are provided to use as a guide, but it is highly recommended to reference your own instruments' manual. It is assumed that you have at least some basic experience with ladder logic, the Simatic Step 7 software and connecting the Siemens S7-1200 to a PC.

1 Wiring

In this section we connect a single HMP7 probe to the communication board (CB 1241 RS485) on a Siemens S7-1200 PLC. We use a cable from Vaisala for the physical connection. Make sure your PLC is detached from a power source before you attempt any wiring.

1. If you are using a cable from Vaisala, the following wire colors signify the function:
   - Brown: Power supply
   - White: RS-485 –
   - Blue: Power GND and RS-485 common
   - Black: RS-485 +
2. If not, reference the diagram below.

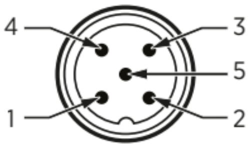
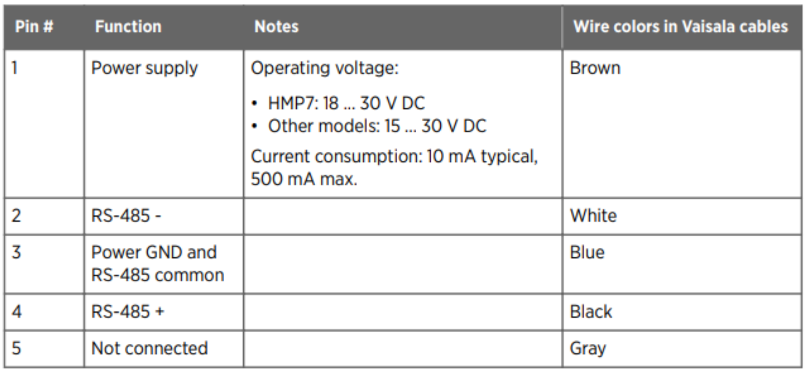


Figure 4 M12 5-pin A-coded male connector pinout



Pin #	Function	Notes	Wire colors in Vaisala cables
1	Power supply	Operating voltage: • HMP7: 18 ... 30 V DC • Other models: 15 ... 30 V DC Current consumption: 10 mA typical, 500 mA max.	Brown
2	RS-485 -		White
3	Power GND and RS-485 common		Blue
4	RS-485 +		Black
5	Not connected		Gray

(HMPx Modbus wiring: HMP Series User Guide p.22)

3. See the diagram below for the wiring.

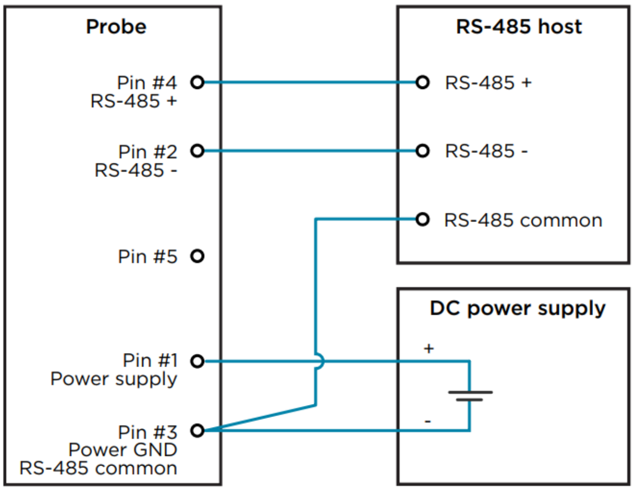
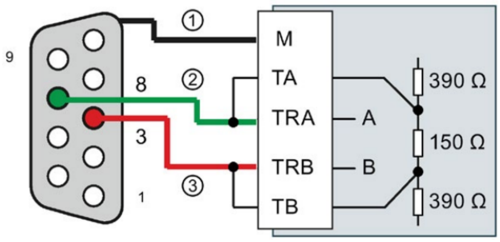
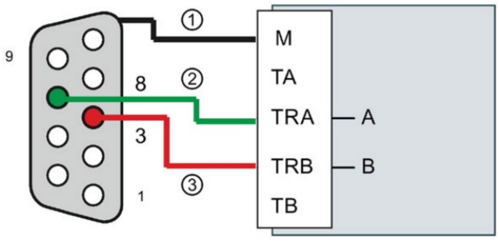


Figure 5 RS-485 wiring

Recommended maximum length of the RS-485 line is 30 m (98 ft).

(HMPx Modbus wiring: HMP Series User Guide p.23)

Terminating device (bias ON)	Non-terminating device (bias OFF)
	

① Connect M to the cable shield

② A = TxD/RxD - (Green wire / Pin 8)

③ B = TxD/RxD + (Red wire / Pin 3)

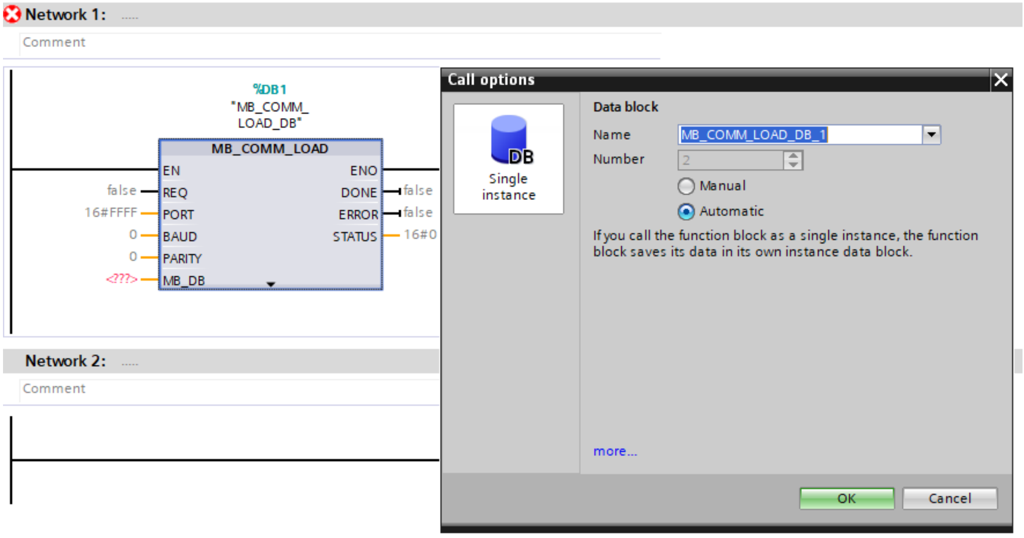
4. Attach T/RA to the white cable (pin 2) and T/RB to the black cable (pin 4) in the Vaisala cable. Connect the Brown cable (pin 1) to DC power +, and the blue cable (pin 3) to ground (M) and DC power -

2 Configuring the port on the communication module for Modbus RTU

In this section we will configure the communication module for Modbus RTU using the function block MB_COMM_LOAD in TIA16 with the Simatic Step 7 Basic software.

1. Drag an empty box to an empty network and type in MB_COMM_LOAD or select it from the instruction window on the right-hand side of the screen.

2. The Call options window opens where you can change the name of the block and confirm it.



Data block

Name: MB_COMM_LOAD_DB_1

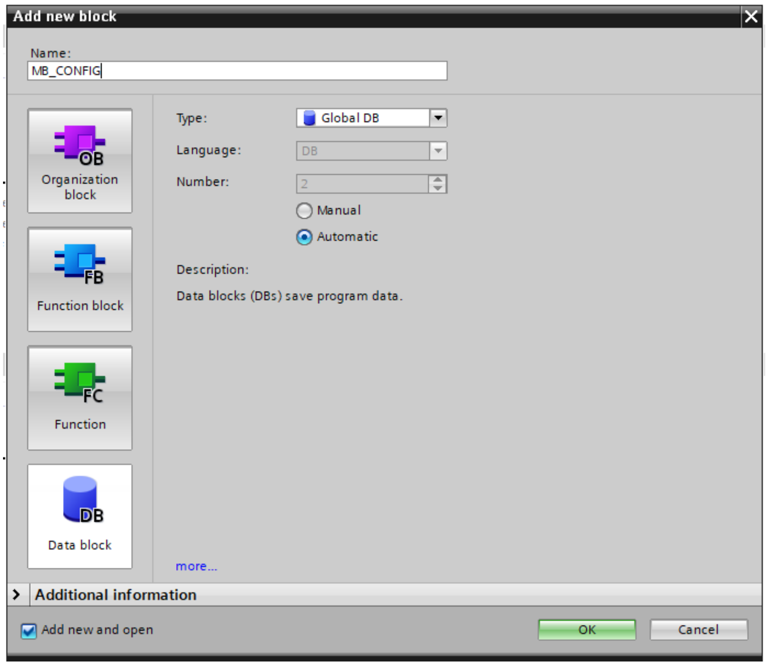
Number: 2

○ Manual

● Automatic

If you call the function block as a single instance, the function block saves its data in its own instance data block.

3. Create a new Data block to store the variables for MB_COMM_LOAD



Name: MB_CONFIG

Type: Global DB

Language: DB

Number: 2

○ Manual

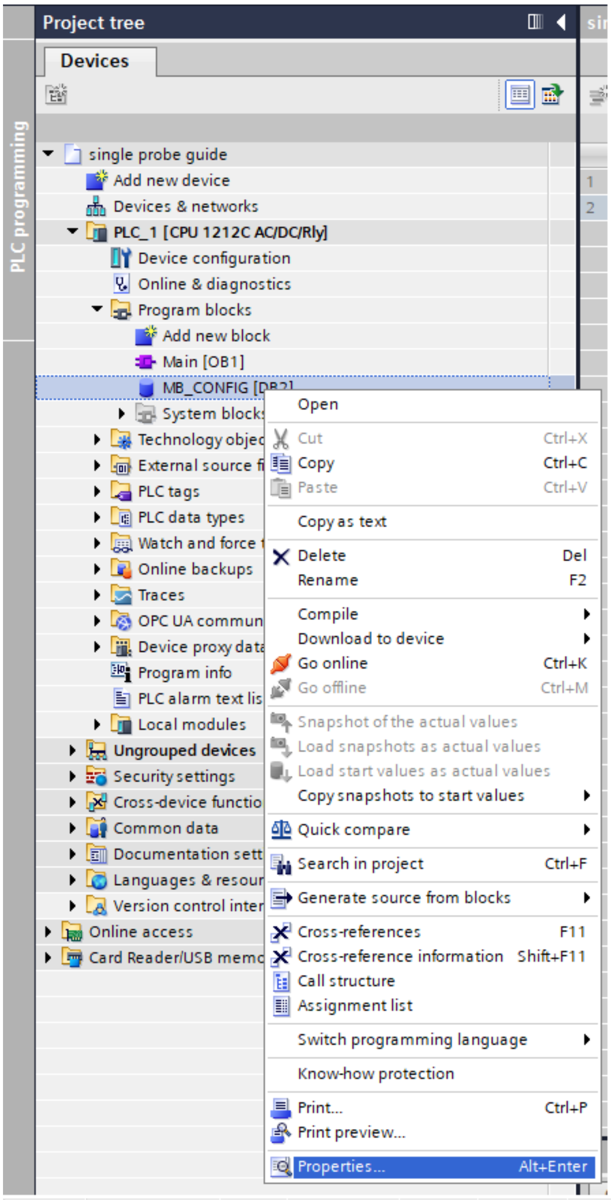
● Automatic

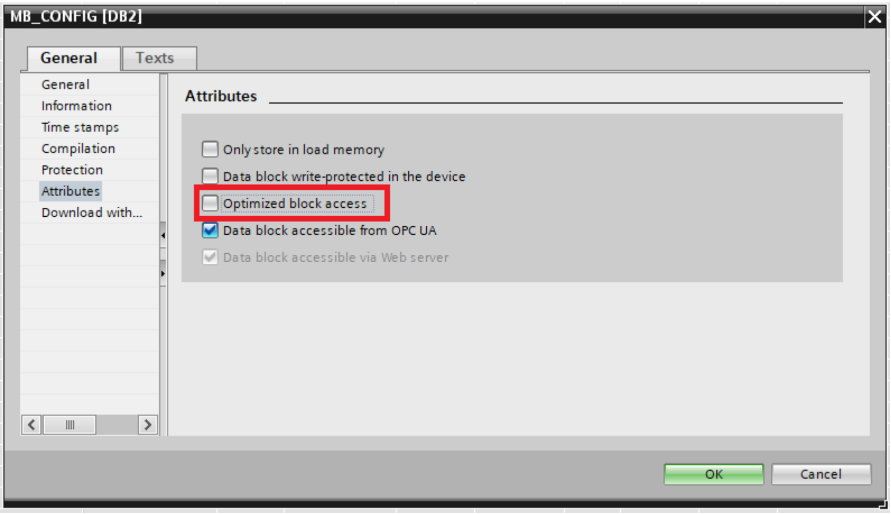
Description:

Data blocks (DBs) save program data.

☑ Add new and open

4. Uncheck the “Optimized block access” box under attributes in properties





Attributes

☐ Only store in load memory

☐ Data block write-protected in the device

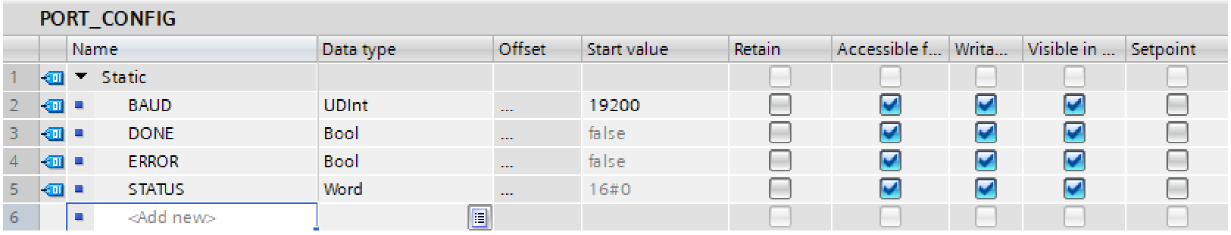
☐ Optimized block access

☑ Data block accessible from OPC UA

☑ Data block accessible via Web server

5. Add the following variables with the data types in the data block for the MB_COMM_LOAD function block:

    - BAUD, UDInt
    - PARITY, UInt
    - DONE, Bool
    - ERROR, Bool
    - STATUS, Word

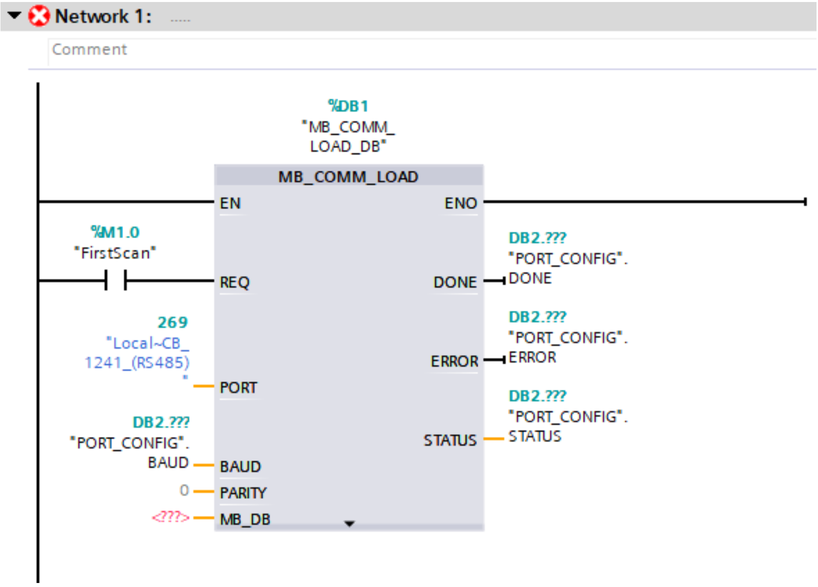


PORT_CONFIG

	Name	Data type	Offset	Start value	Retain	Accessible f...	Writa...	Visible in ...	Setpoint
1	▼ Static				□	□	□	□	□
2	BAUD	UDInt	...	19200	□	☑	☑	☑	□
3	DONE	Bool	...	false	□	☑	☑	☑	□
4	ERROR	Bool	...	false	□	☑	☑	☑	□
5	STATUS	Word	...	16#0	□	☑	☑	☑	□
6	<Add new>				□	□	□	□	□

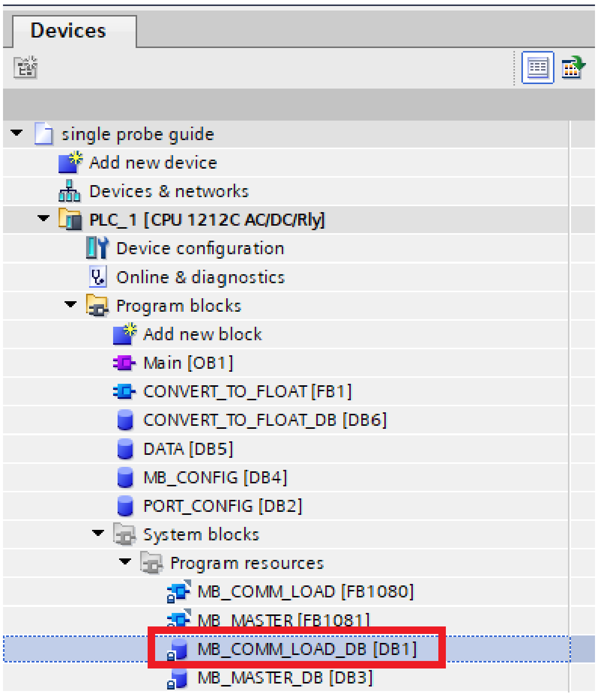
6. Set the baud rate and parity according to your probe's specifications. In our case 19200 for the baud rate and 0 for the parity.

7. Add the variables from the Data block to MB_COMM_LOAD_DB



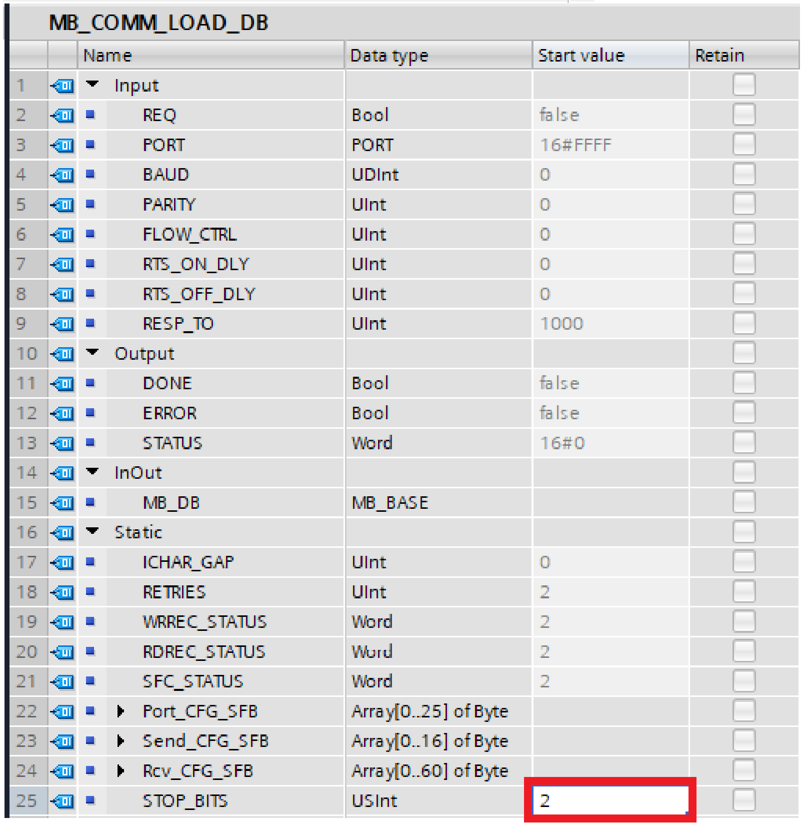
8. Set the PORT variable to the communication module that is installed on your PLC

9. To change the stop bits, under the Program blocks folder, open the System blocks folder then the Program resources folder, then open MB_COMM_LOAD_DB. At the bottom of MB_COMM_LOAD_DB you will find the STOP_BITS where you can set the appropriate value for your probe. We set the value as 2.



Devices

- single probe guide
  - Add new device
  - Devices & networks
  - PLC_1 [CPU 1212C AC/DC/Rly]
    - Device configuration
    - Online & diagnostics
    - Program blocks
      - Add new block
      - Main [OB1]
      - CONVERT_TO_FLOAT [FB1]
      - CONVERT_TO_FLOAT_DB [DB6]
      - DATA [DB5]
      - MB_CONFIG [DB4]
      - PORT_CONFIG [DB2]
      - System blocks
        - Program resources
          - MB_COMM_LOAD [FB1080]
          - MB_MASTER [FB1081]
          - MB_COMM_LOAD_DB [DB1]
          - MB_MASTER_DB [DB3]



MB_COMM_LOAD_DB

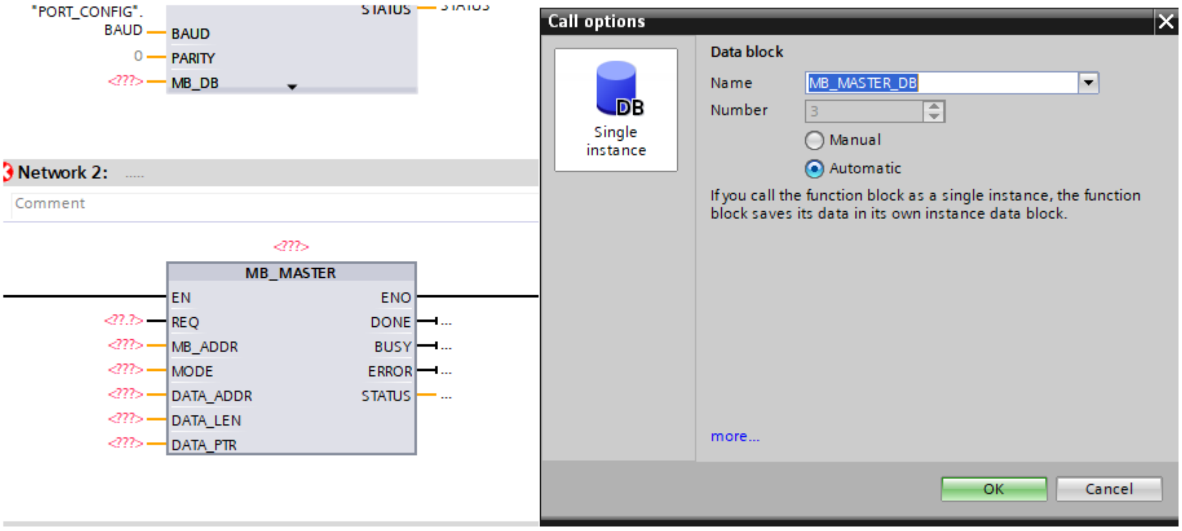
	Name	Data type	Start value	Retain
1	▼ Input			☐
2	REQ	Bool	false	☐
3	PORT	PORT	16#FFFF	☐
4	BAUD	UDInt	0	☐
5	PARITY	UInt	0	☐
6	FLOW_CTRL	UInt	0	☐
7	RTS_ON_DLY	UInt	0	☐
8	RTS_OFF_DLY	UInt	0	☐
9	RESP_TO	UInt	1000	☐
10	▼ Output			☐
11	DONE	Bool	false	☐
12	ERROR	Bool	false	☐
13	STATUS	Word	16#0	☐
14	▼ InOut			☐
15	MB_DB	MB_BASE		☐
16	▼ Static			☐
17	ICHAR_GAP	UInt	0	☐
18	RETRIES	UInt	2	☐
19	WRREC_STATUS	Word	2	☐
20	RDREC_STATUS	Word	2	☐
21	SFC_STATUS	Word	2	☐
22	▶ Port_CFG_SFB	Array[0..25] of Byte		☐
23	▶ Send_CFG_SFB	Array[0..16] of Byte		☐
24	▶ Rcv_CFG_SFB	Array[0..60] of Byte		☐
25	STOP_BITS	USInt	2	☐

3 Communicating with the HMP7 Probe

In this section we will show you how to use MB_MASTER function block to communicate with the HMP7 probe, which values to choose for the variables and how to store the holding registers.

1. Drag an empty box to an empty network and type MB_MASTER into it or select the block from the instruction window on the right-hand side of the screen.

2. The Call options window opens where you can change the name of the block and confirm it.



Data block

Name: MB_MASTER_DB

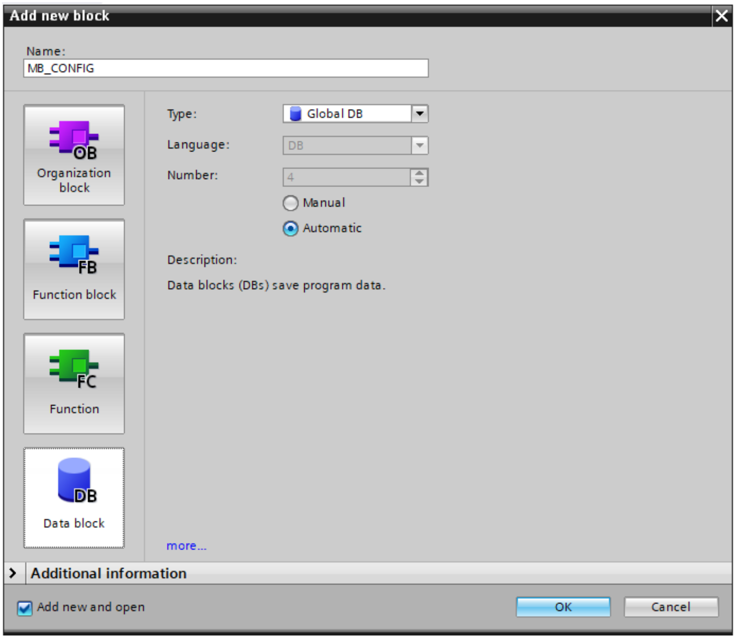
Number: 3

○ Manual

● Automatic

If you call the function block as a single instance, the function block saves its data in its own instance data block.

3. Create a new Data block to store the variables for MB_MASTER



Name: MB_CONFIG

Type: Global DB

Language: DB

Number: 4

○ Manual

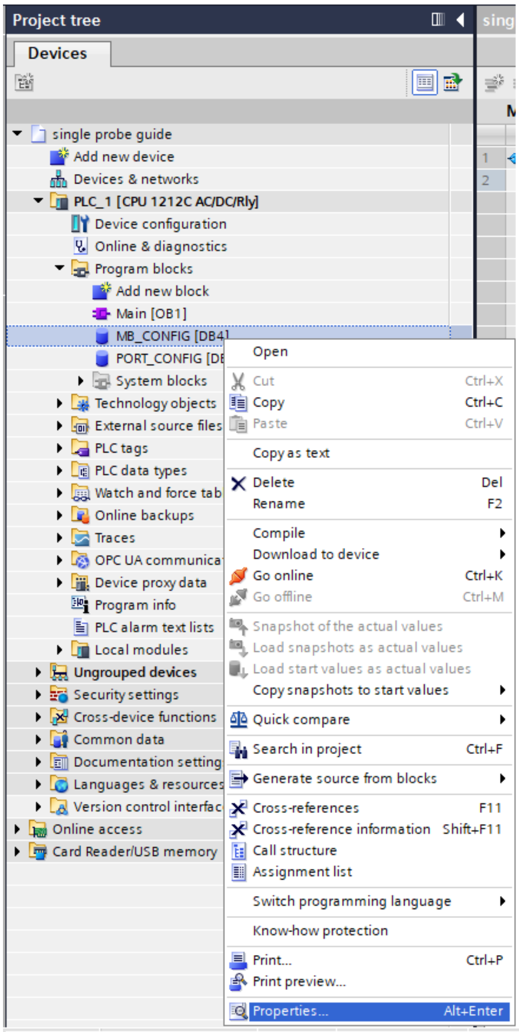
● Automatic

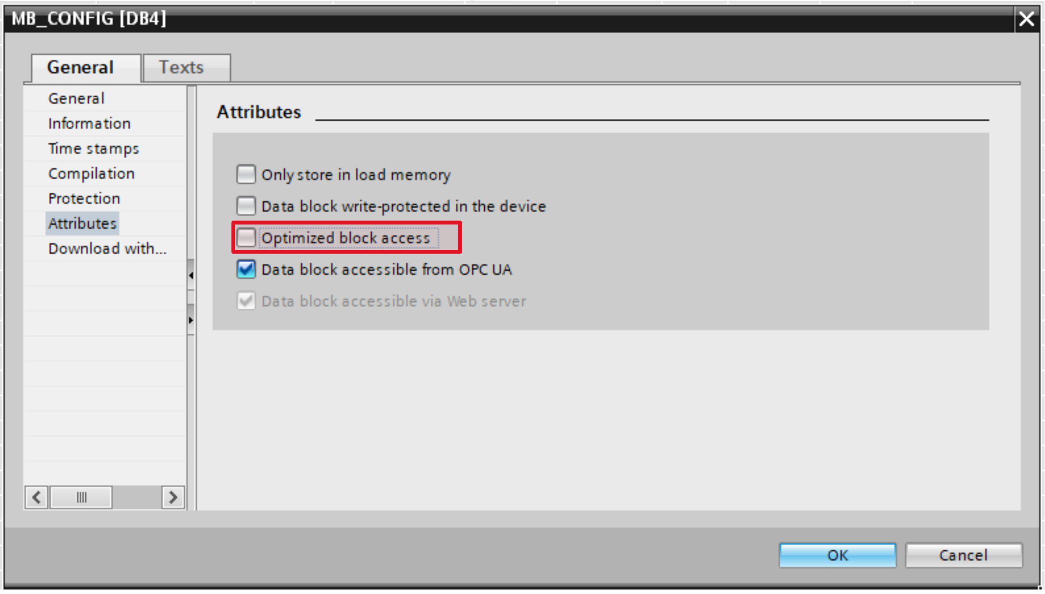
Description:

Data blocks (DBs) save program data.

☑ Add new and open

4. Uncheck the “Optimized data access” box under attributes in properties.





Attributes

□ Only store in load memory

□ Data block write-protected in the device

□ Optimized block access

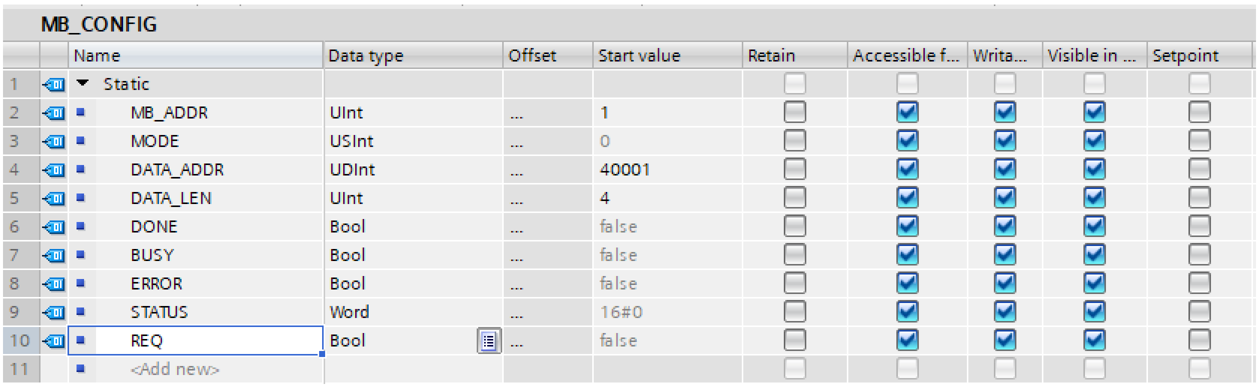
☑ Data block accessible from OPC UA

☑ Data block accessible via Web server

5. Add the following variables with the data types to the new data block:

- MB_ADDR, UInt
- MODE, USInt
- DATA_ADDR, UDInt

- DATA_LEN, UInt
- DONE, Bool
- BUSY, Bool
- ERROR, Bool
- STATUS, Word
- REQ, Bool



MB_CONFIG

	Name	Data type	Offset	Start value	Retain	Accessible f...	Writa...	Visible in ...	Setpoint
1	▼ Static				☐	☐	☐	☐	☐
2	MB_ADDR	UInt	...	1	☐	☑	☑	☑	☐
3	MODE	USInt	...	0	☐	☑	☑	☑	☐
4	DATA_ADDR	UDInt	...	40001	☐	☑	☑	☑	☐
5	DATA_LEN	UInt	...	4	☐	☑	☑	☑	☐
6	DONE	Bool	...	false	☐	☑	☑	☑	☐
7	BUSY	Bool	...	false	☐	☑	☑	☑	☐
8	ERROR	Bool	...	false	☐	☑	☑	☑	☐
9	STATUS	Word	...	16#0	☐	☑	☑	☑	☐
10	REQ	Bool	...	false	☐	☑	☑	☑	☐
11	<Add new>				☐	☐	☐	☐	☐

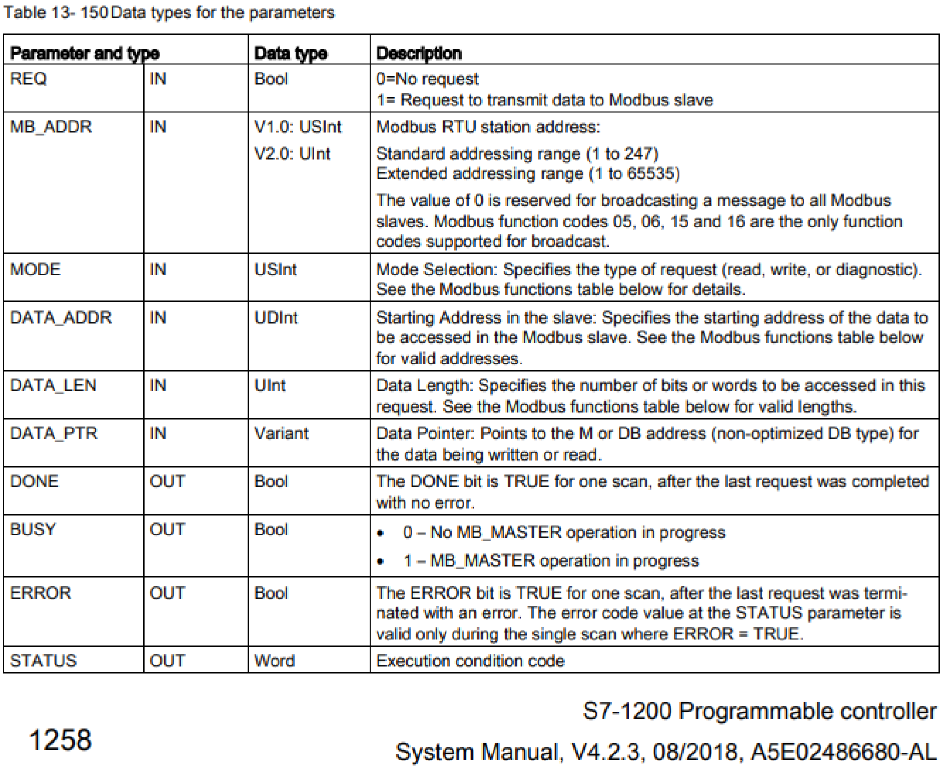


Table 13- 150 Data types for the parameters

Parameter and type		Data type	Description
REQ	IN	Bool	0=No request 1= Request to transmit data to Modbus slave
MB_ADDR	IN	V1.0: USInt V2.0: UInt	Modbus RTU station address: Standard addressing range (1 to 247) Extended addressing range (1 to 65535) The value of 0 is reserved for broadcasting a message to all Modbus slaves. Modbus function codes 05, 06, 15 and 16 are the only function codes supported for broadcast.
MODE	IN	USInt	Mode Selection: Specifies the type of request (read, write, or diagnostic). See the Modbus functions table below for details.
DATA_ADDR	IN	UDInt	Starting Address in the slave: Specifies the starting address of the data to be accessed in the Modbus slave. See the Modbus functions table below for valid addresses.
DATA_LEN	IN	UInt	Data Length: Specifies the number of bits or words to be accessed in this request. See the Modbus functions table below for valid lengths.
DATA_PTR	IN	Variant	Data Pointer: Points to the M or DB address (non-optimized DB type) for the data being written or read.
DONE	OUT	Bool	The DONE bit is TRUE for one scan, after the last request was completed with no error.
BUSY	OUT	Bool	• 0 – No MB_MASTER operation in progress • 1 – MB_MASTER operation in progress
ERROR	OUT	Bool	The ERROR bit is TRUE for one scan, after the last request was terminated with an error. The error code value at the STATUS parameter is valid only during the single scan where ERROR = TRUE.
STATUS	OUT	Word	Execution condition code

1258 S7-1200 Programmable controller

System Manual, V4.2.3, 08/2018, A5E02486680-AL

6. We want to read the holding registers, to do so we must choose Mode 0, and the Modbus Address starting from 40001. Address 40001 corresponds to register 1.

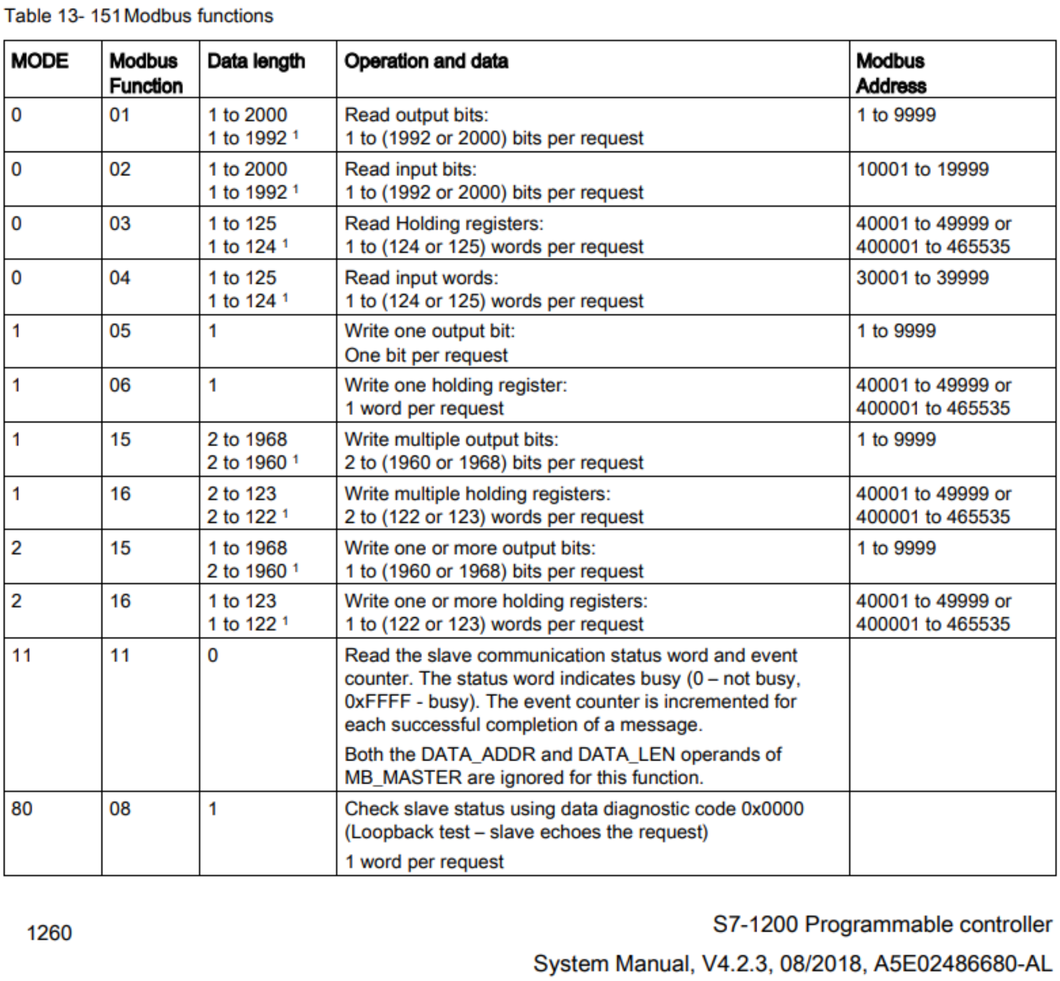


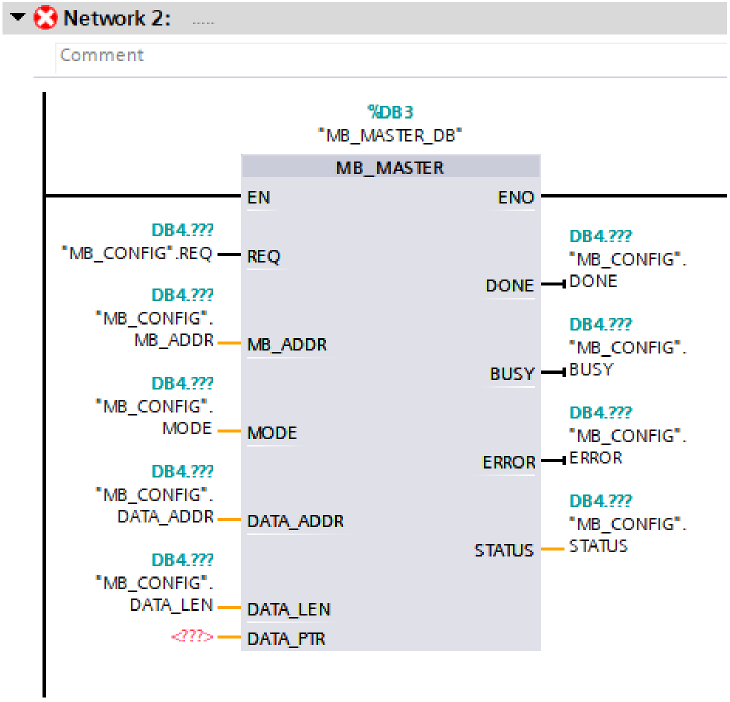
Table 13- 151 Modbus functions

MODE	Modbus Function	Data length	Operation and data	Modbus Address
0	01	1 to 2000 1 to 1992 [1]	Read output bits: 1 to (1992 or 2000) bits per request	1 to 9999
0	02	1 to 2000 1 to 1992 [1]	Read input bits: 1 to (1992 or 2000) bits per request	10001 to 19999
0	03	1 to 125 1 to 124 [1]	Read Holding registers: 1 to (124 or 125) words per request	40001 to 49999 or 400001 to 465535
0	04	1 to 125 1 to 124 [1]	Read input words: 1 to (124 or 125) words per request	30001 to 39999
1	05	1	Write one output bit: One bit per request	1 to 9999
1	06	1	Write one holding register: 1 word per request	40001 to 49999 or 400001 to 465535
1	15	2 to 1968 2 to 1960 [1]	Write multiple output bits: 2 to (1960 or 1968) bits per request	1 to 9999
1	16	2 to 123 2 to 122 [1]	Write multiple holding registers: 2 to (122 or 123) words per request	40001 to 49999 or 400001 to 465535
2	15	1 to 1968 2 to 1960 [1]	Write one or more output bits: 1 to (1960 or 1968) bits per request	1 to 9999
2	16	1 to 123 1 to 122 [1]	Write one or more holding registers: 1 to (122 or 123) words per request	40001 to 49999 or 400001 to 465535
11	11	0	Read the slave communication status word and event counter. The status word indicates busy (0 – not busy, 0xFFFF - busy). The event counter is incremented for each successful completion of a message. Both the DATA_ADDR and DATA_LEN operands of MB_MASTER are ignored for this function.	
80	08	1	Check slave status using data diagnostic code 0x0000 (Loopback test – slave echoes the request) 1 word per request	

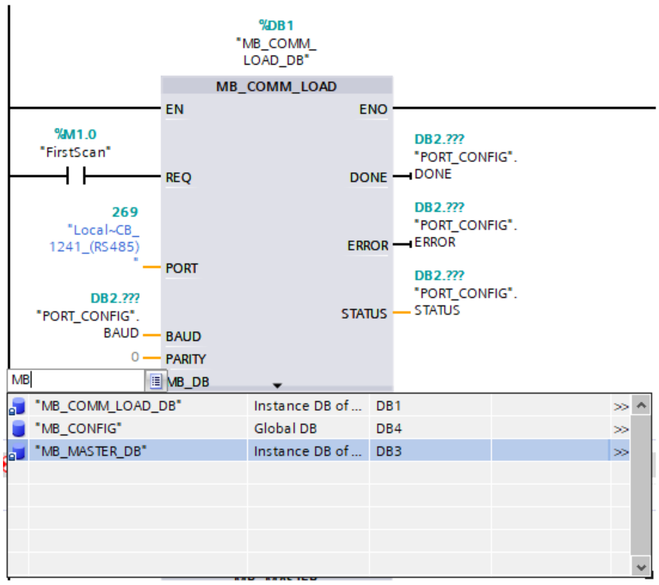
1260 S7-1200 Programmable controller

System Manual, V4.2.3, 08/2018, A5E02486680-AL

7. Set the variables to their appropriate places on MB_MASTER_DB



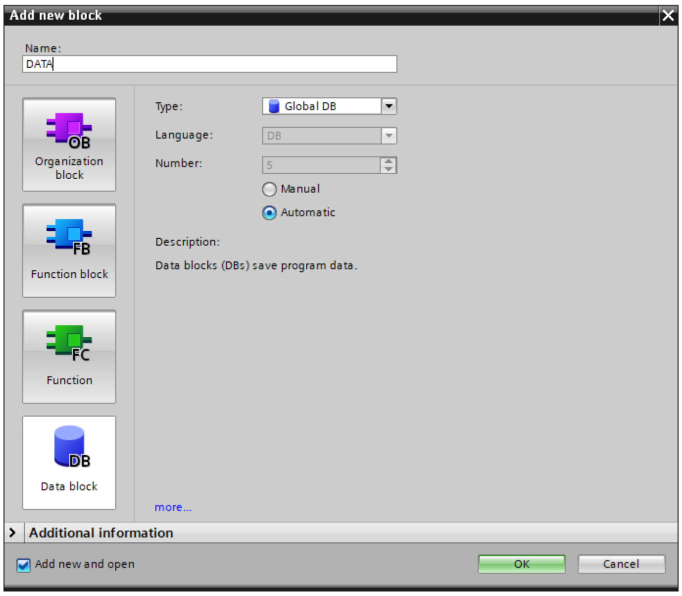
8. Once you have created MB_MASTER_DB it is time to pass the instance of it to the MB_DB argument of MB_COMM_LOAD_DB



"MB_COMM_LOAD_DB"	Instance DB of ...	DB1	>>
"MB_CONFIG"	Global DB	DB4	>>
"MB_MASTER_DB"	Instance DB of ...	DB3	>>

Now it is time to create a new Data block to store the values from the holding registers

1. Create a new data block to store the holding registers by clicking on the “Add new block” under Program blocks.



Name: DATA

Type: Global DB

Language: DB

Number: 5

○ Manual

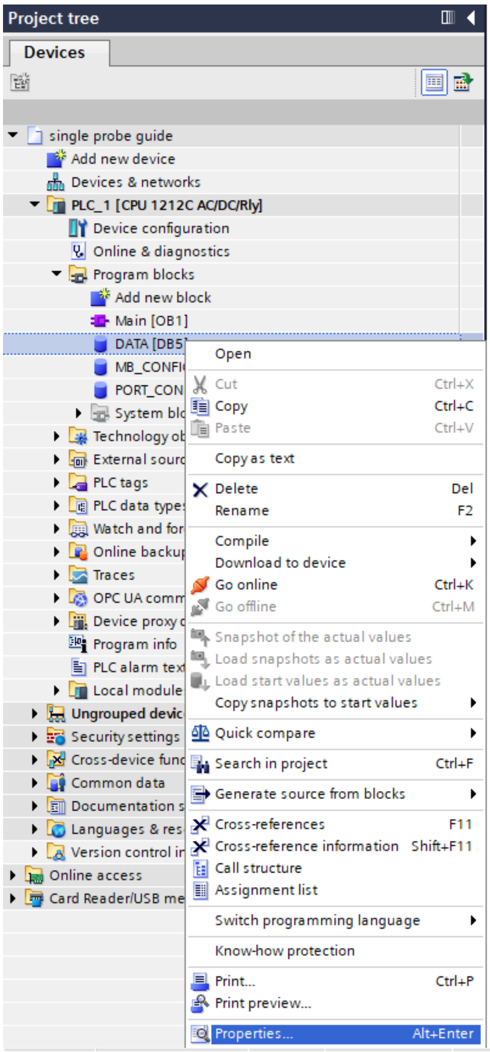
● Automatic

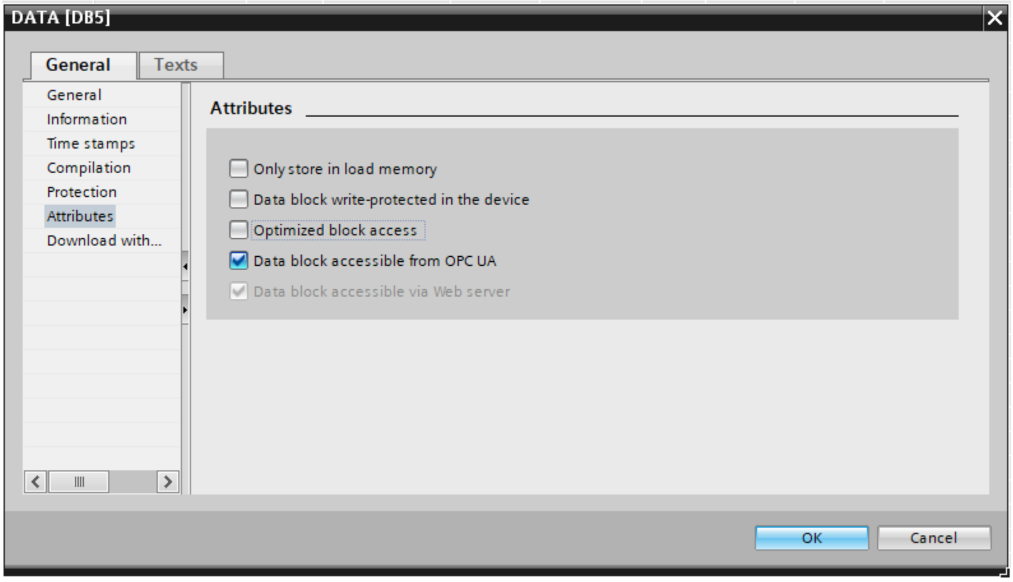
Description:

Data blocks (DBs) save program data.

☑ Add new and open

2. Uncheck “Optimized block access” in Attributes under properties.




DATA [DB5]

Attributes

☐ Only store in load memory

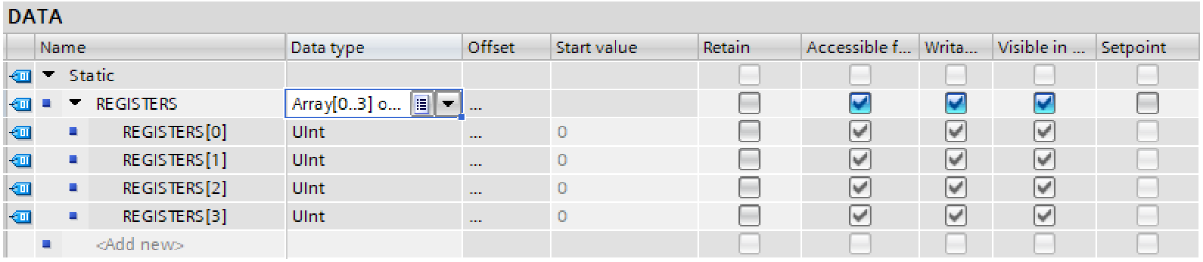
☐ Data block write-protected in the device

☐ Optimized block access

☑ Data block accessible from OPC UA

☑ Data block accessible via Web server

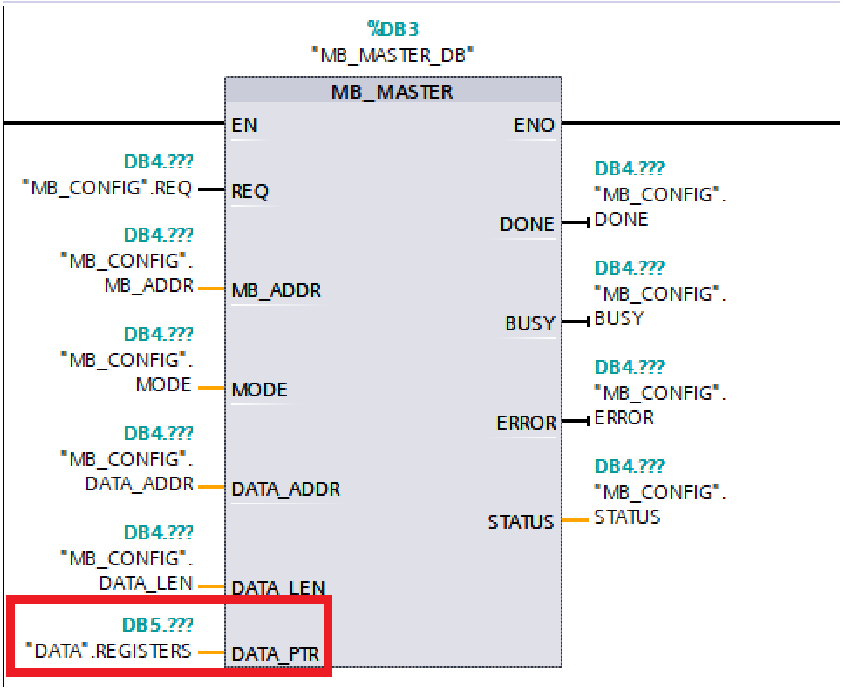
3. Create an array of UInt with the same number of elements as the value for DATA_LEN you selected.



DATA

Name	Data type	Offset	Start value	Retain	Accessible f...	Writa...	Visible in ...	Setpoint
▼ Static				□	□	□	□	□
▼ REGISTERS	Array[0..3] o...	...		□	☑	☑	☑	□
REGISTERS[0]	UInt	...	0	□	☑	☑	☑	□
REGISTERS[1]	UInt	...	0	□	☑	☑	☑	□
REGISTERS[2]	UInt	...	0	□	☑	☑	☑	□
REGISTERS[3]	UInt	...	0	□	☑	☑	☑	□
<Add new>				□	□	□	□	□

4. Add the array to the DATA_PTR variable of MB_MASTER

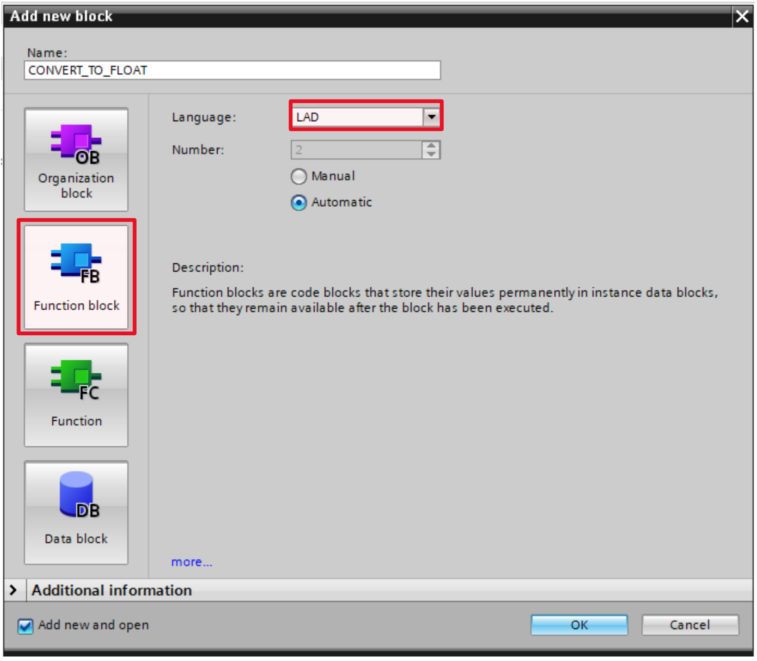


4 Converting word addresses to human readable 32-bit floating point values:

The data sent via Modbus is split into two 16-bit words, we need to combine the two words into a 32-bit floating point value to get a readable value. We will create a new Function block that moves and converts the bits into a 32-bit float.

1. Create a new Function block to convert the values by clicking on the “Add new block” under Program blocks and selecting function block. Make sure that the language selected is LAD.

2. Uncheck “Optimized block access” under attributes in properties



Add new block

Name: CONVERT_TO_FLOAT

Language: LAD

Number: 2

○ Manual

● Automatic

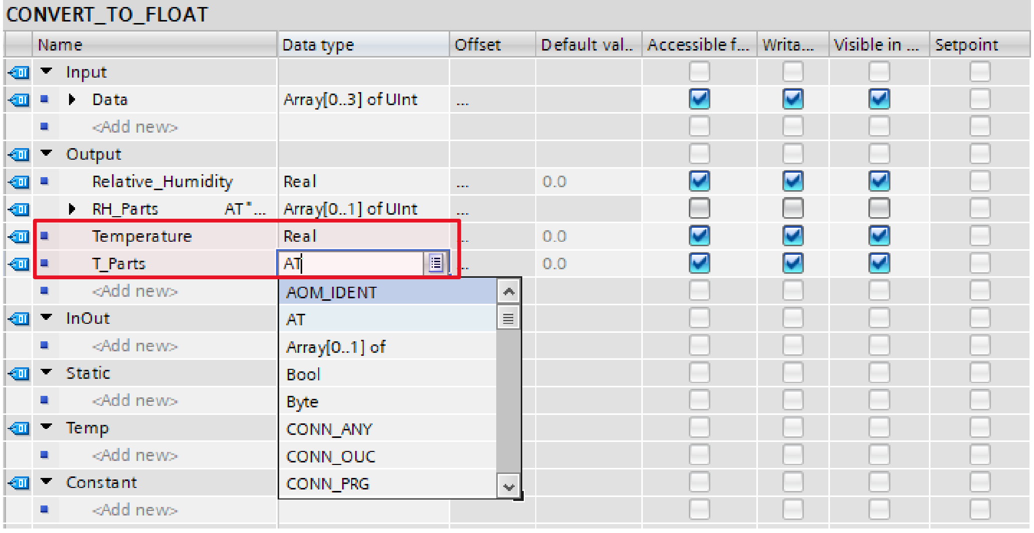
Description:

Function blocks are code blocks that store their values permanently in instance data blocks, so that they remain available after the block has been executed.

☑ Add new and open

3. We will take advantage of the AT overlay to combine the two words into a 32-bit floating point value. First create the Real variable into which the two words are combined in the Output section of the function block variables.

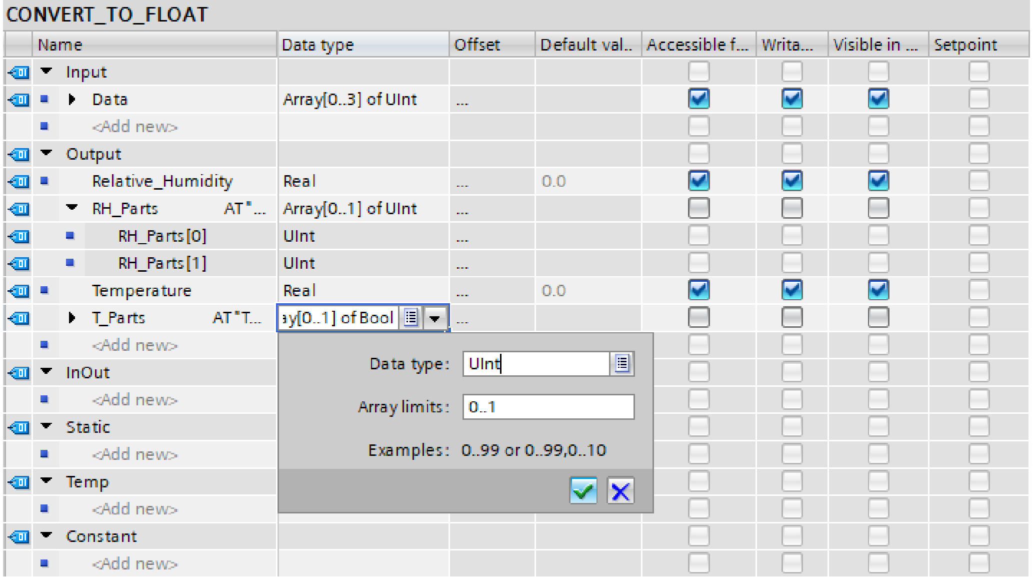
4. Directly under the Real variable, create another variable with the Data type as “AT”



CONVERT_TO_FLOAT

Name	Data type	Offset	Default val..	Accessible f...	Writa...	Visible in ...	Setpoint
Input				☐	☐	☐	☐
Data	Array[0..3] of UInt	...		☑	☑	☑	☐
<Add new>				☐	☐	☐	☐
Output				☐	☐	☐	☐
Relative_Humidity	Real	...	0.0	☑	☑	☑	☐
RH_Parts AT "...	Array[0..1] of UInt	...		☐	☐	☐	☐
Temperature	Real	...	0.0	☑	☑	☑	☐
T_Parts	AT	...	0.0	☑	☑	☑	☐
<Add new>				☐	☐	☐	☐
InOut				☐	☐	☐	☐
<Add new>				☐	☐	☐	☐
Static				☐	☐	☐	☐
<Add new>				☐	☐	☐	☐
Temp				☐	☐	☐	☐
<Add new>				☐	☐	☐	☐
Constant				☐	☐	☐	☐
<Add new>				☐	☐	☐	☐

5. The AT variable Data type should then change to that of an array, change the array Data type to UInt and the Array limits to 0..1



CONVERT_TO_FLOAT

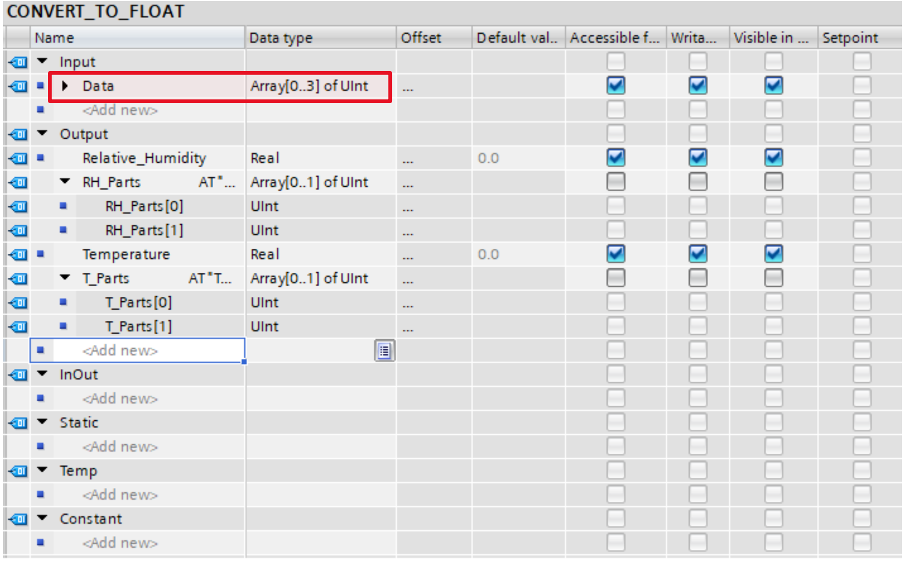
Name	Data type	Offset	Default val..	Accessible f...	Writa...	Visible in ...	Setpoint
Input				□	□	□	□
Data	Array[0..3] of UInt	...		☑	☑	☑	□
<Add new>				□	□	□	□
Output				□	□	□	□
Relative_Humidity	Real	...	0.0	☑	☑	☑	□
RH_Parts AT"...	Array[0..1] of UInt	...		□	□	□	□
RH_Parts[0]	UInt	...		□	□	□	□
RH_Parts[1]	UInt	...		□	□	□	□
Temperature	Real	...	0.0	☑	☑	☑	□
T_Parts AT"T...	ay[0..1] of Bool	...		□	□	□	□
<Add new>				□	□	□	□
InOut				□	□	□	□
<Add new>				□	□	□	□
Static				□	□	□	□
<Add new>				□	□	□	□
Temp				□	□	□	□
<Add new>				□	□	□	□
Constant				□	□	□	□
<Add new>				□	□	□	□

Data type: UInt

Array limits: 0..1

Examples: 0..99 or 0..99,0..10

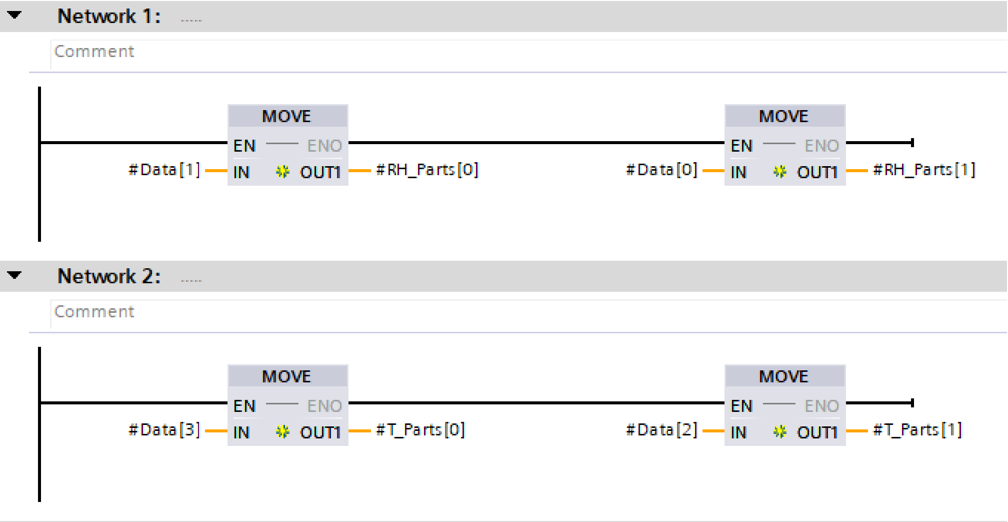
6. Add into the input section, an array of UInt with the same number of elements as the Holding register array.



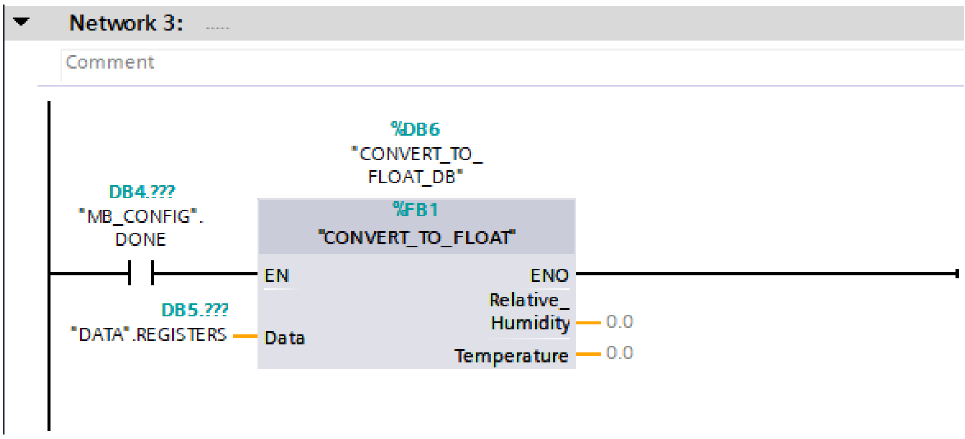
CONVERT_TO_FLOAT

Name	Data type	Offset	Default val..	Accessible f...	Writa...	Visible in ...	Setpoint
Input				□	□	□	□
Data	Array[0..3] of UInt	...		☑	☑	☑	□
<Add new>				□	□	□	□
Output				□	□	□	□
Relative_Humidity	Real	...	0.0	☑	☑	☑	□
RH_Parts AT"...	Array[0..1] of UInt	...		□	□	□	□
RH_Parts[0]	UInt	...		□	□	□	□
RH_Parts[1]	UInt	...		□	□	□	□
Temperature	Real	...	0.0	☑	☑	☑	□
T_Parts AT"T...	Array[0..1] of UInt	...		□	□	□	□
T_Parts[0]	UInt	...		□	□	□	□
T_Parts[1]	UInt	...		□	□	□	□
<Add new>				□	□	□	□
InOut				□	□	□	□
<Add new>				□	□	□	□
Static				□	□	□	□
<Add new>				□	□	□	□
Temp				□	□	□	□
<Add new>				□	□	□	□
Constant				□	□	□	□
<Add new>				□	□	□	□

7. Now it is time to move the holding register data into the overlaying array. We get the data from the probe in little endian format, we must then reverse the order of the words when we move them to the overlaying array. In other words, the second element of the data array goes to the first element of the target array and the first element of the data array goes to the second element of the target array.

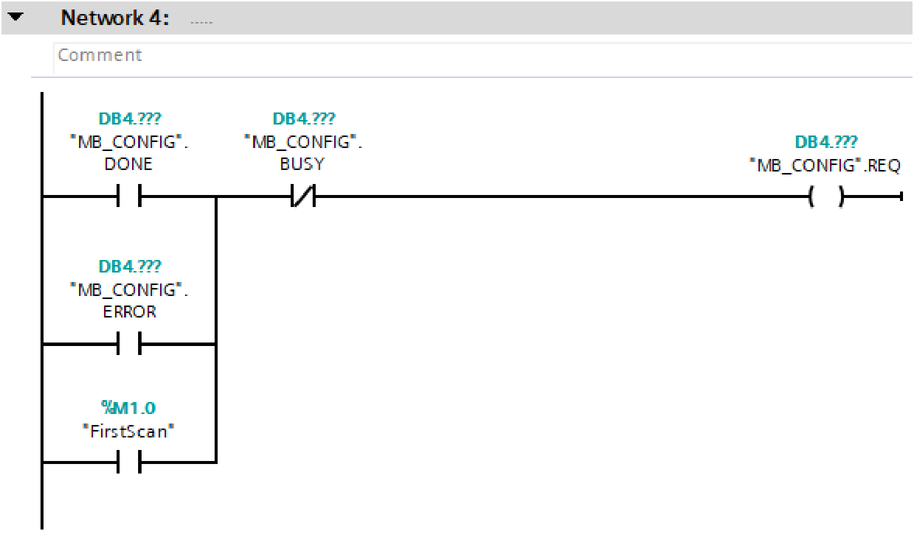

Network 1: .....
Comment
MOVE
EN ENO
#Data[1] IN OUT1 #RH_Parts[0]
MOVE
EN ENO
#Data[0] IN OUT1 #RH_Parts[1]
Network 2: .....
Comment
MOVE
EN ENO
#Data[3] IN OUT1 #T_Parts[0]
MOVE
EN ENO
#Data[2] IN OUT1 #T_Parts[1]

8. Add the function block to a new network in the main program block and have the done bit from MB_MASTER trigger the function block.
9. Set the data array as the input and save the outputs in another variable.



5 Triggering a MB_MASTER cycle

In this section we will show the ladder logic that continuously triggers a new MB_MASTER cycle after each completed cycle. We want the new cycle to be triggered after the probe has responded or it has ended in an error. We added a closed BUSY contact to make sure that a new request does not attempt to trigger a new cycle while MB_MASTER is already busy with the previous request. We added the “FirstScan” open contact to trigger the first cycle before the done or the error bit is activated.


Network 4:
.....
Comment
DB4.???
"MB_CONFIG".
DONE
DB4.???
"MB_CONFIG".
BUSY
DB4.???
"MB_CONFIG".REQ
DB4.???
"MB_CONFIG".
ERROR
%M1.0
"FirstScan"