warmafloor

we know a touch more

Sentio Modbus

Author: Jiří Šindelář
Document: TM420
Version: 07

1 Overview

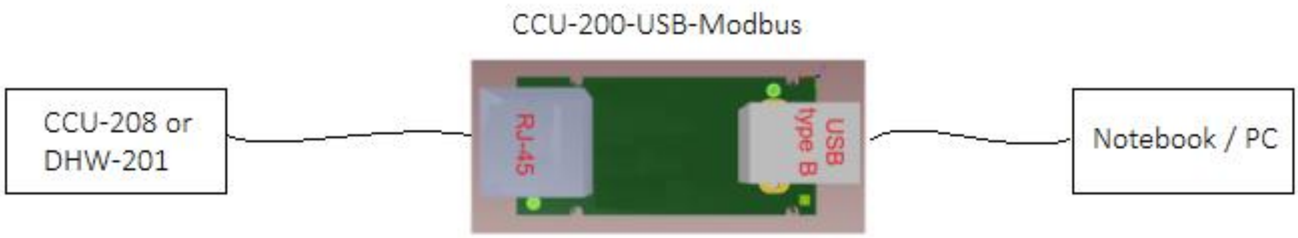
1.1 Document versions

Version	Change description	Chapter
07	Added detailed description for reading and writing text value	3.6.x
06	Added description how to work with strings	3.6
	Added quick start chapter	2.x
	Updated Modbus values description	4.x
05	Added description of connection with device	2.2

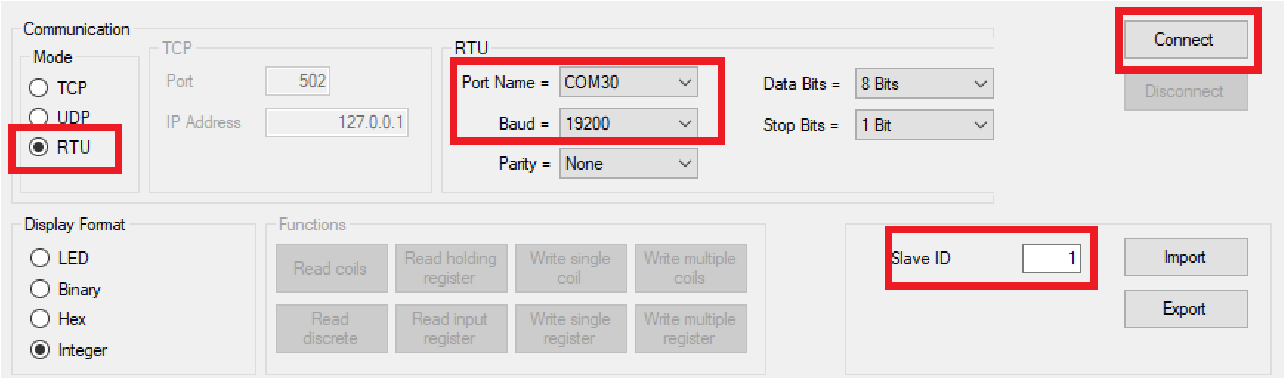
1.2 Standards

Modbus is implemented according to following specifications:
http://modbus.org/docs/Modbus_Application_Protocol_V1_1b3.pdf
http://modbus.org/docs/Modbus_over_serial_line_V1_02.pdf

2 Quick Start



1. Connect CCU-200-USB convertor to your PC
    a. You may see a “New device” notification and automatic driver installation.
    b. Open Windows Device Manager and check a COM number associated with the convertor. You will need it later for configuring Modbus Master application.
2. Configure device interface for Modbus
    a. See chapter Interface setup & wires
3. Connect CCU-200-USB convertor to the device
    a. See chapter Interface setup & wires
4. Install and run Modbus Master application
    a. Choose RTU communication
    b. Choose COM port. (You will find the right number in the Device Manager – see point 1b)
    c. Configure Baud to 19200 (Default value)
    d. Set Slave ID (Modbus address) to 1 (Default value)
    e. Click the **Connect** button



Communication

Mode: ○ TCP ○ UDP ● RTU

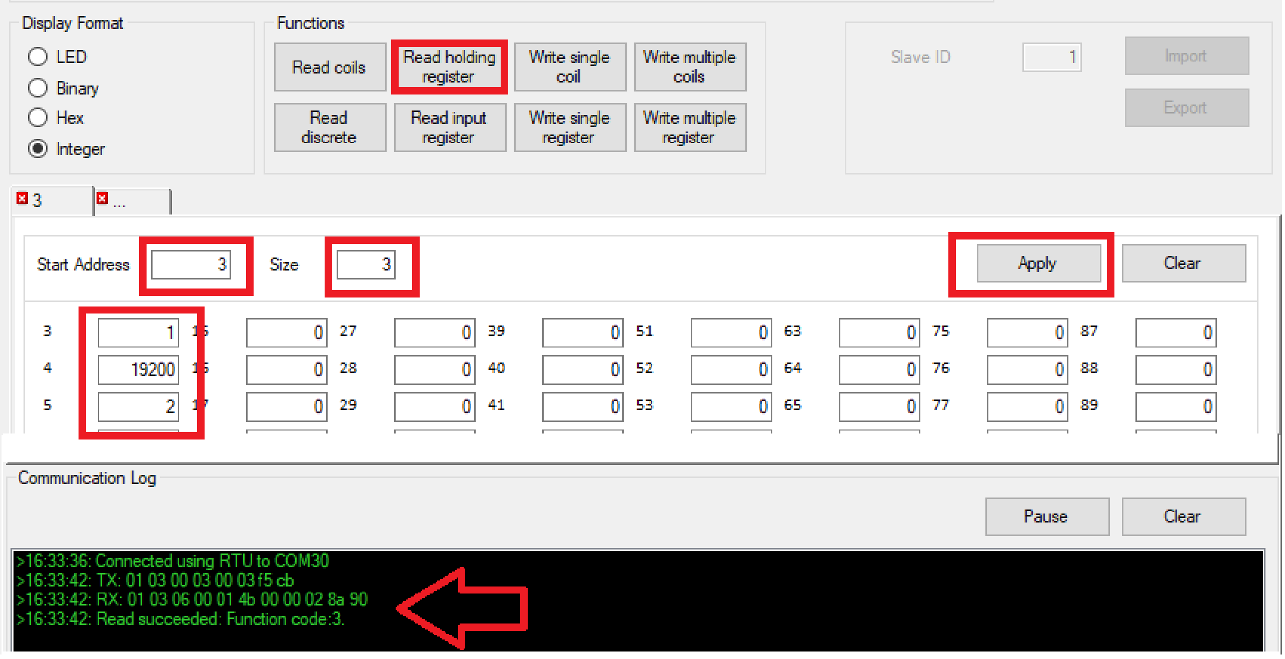
TCP: Port 502, IP Address 127.0.0.1

RTU: Port Name = COM30, Baud = 19200, Parity = None, Data Bits = 8 Bits, Stop Bits = 1 Bit

Display Format: ○ LED ○ Binary ○ Hex ● Integer

Slave ID 1

5. Try to read input registers
    a. Set Start Address to 3
    b. Set Size to 3
    c. Click **Apply** button! Please do not forget to click the Apply button every time you change Start Address or Size.
    d. Click **Read holding register**
    e. You should see Read succeeded message in the Communication Log (bottom part of the application).
    f. If you see communication timeout, then check connection and/or load default configuration (CCU-208, DHW-201)
    g. Registers read by Modbus command are shown in table – see position 0 – 4 where
        - 3 .. Modbus slave address
        - 4 .. Modbus baud rate
        - 5 .. Modbus mode



Display Format: ○ LED ○ Binary ○ Hex ● Integer

Slave ID: 1

Tabs: 3 | ...

Start Address: 3   Size: 3

3	1	15	0	27	0	39	0	51	0	63	0	75	0	87	0
4	19200	16	0	28	0	40	0	52	0	64	0	76	0	88	0
5	2	17	0	29	0	41	0	53	0	65	0	77	0	89	0

Communication Log

```
>16:33:36: Connected using RTU to COM30
>16:33:42: TX: 01 03 00 03 00 03 f5 cb
>16:33:42: RX: 01 03 06 00 01 4b 00 00 02 8a 90
>16:33:42: Read succeeded: Function code:3.
```

6. If you need to change the Modbus configuration
    a. Edit address (Slave ID) / mode / baud rate
    b. Click **Write multiple register**
    c. You should see Read succeeded message in the Communication Log
    d. Then the device will be unreachable – you have to update Modbus connection parameters in the Modbus Master application
        i. Click Disconnect
        ii. Update Slave ID (address) / Baud
        iii. Click Connect

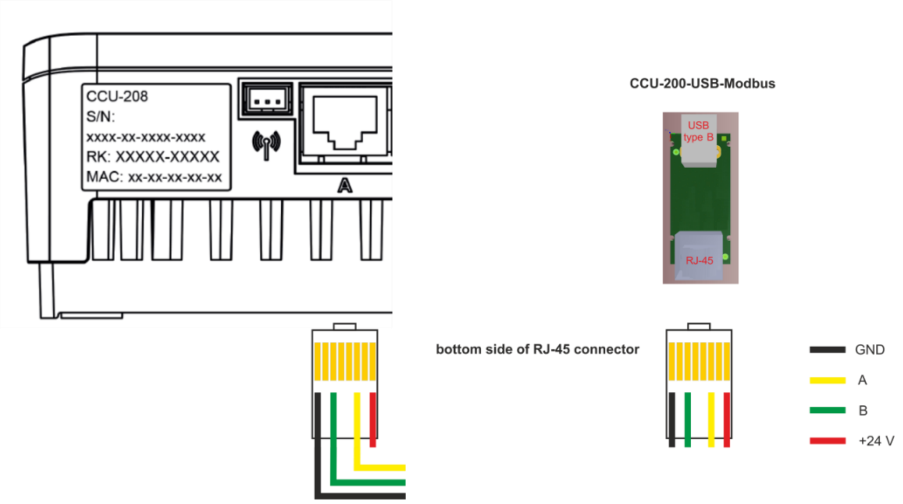
2.1 Bus parameters

Supported baud rates: 9600, 19200(default), 38400, 57600.
Default address: 1
Possible Modes: Disabled, Read Only, Read/Write (Default)

2.2 Interface setup & wires

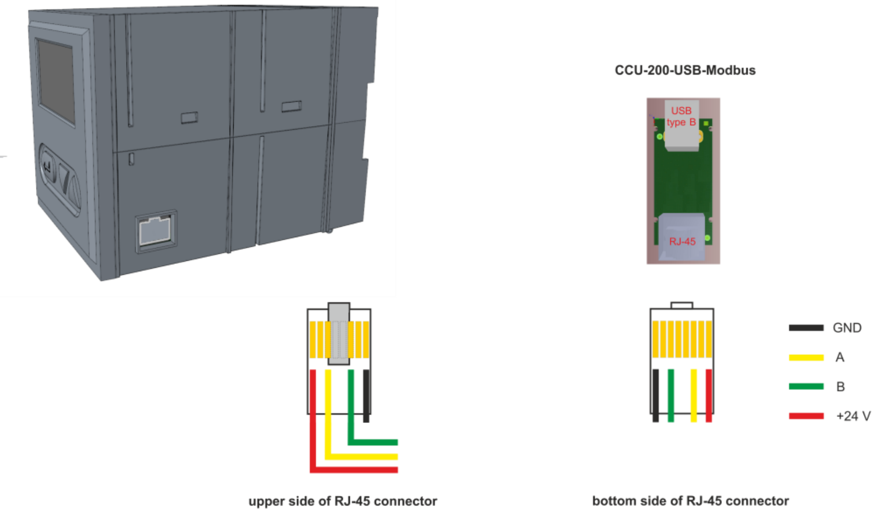
2.2.1 CCU-208

1. Interface has to be configured to Modbus mode using LCD-200
    - System | Installer settings | Modbus configuration
2. Connection is made via CCU-200-USB (**marked modification Modbus**).
3. Port A (RJ-45) has to be used.



2.2.2 DHW-201

1. Interface has to be configured to Modbus mode using DHW-201 user interface
    - Advanced | Network | RJ45 mode
2. Connection is made via CCU-200-USB (**marked modification Modbus**).



3 Modbus details

3.1 Address space

Register address in Modbus protocol can be in range 0 - 65535.
In Sentio system, this address range is divided to two parts AAABB.

- AAA - object address (e.g. Location, Room, …)
- BB - value address (measured temperature, output state, ...)

Sentio uses following Modbus model, where different Modbus commands uses different address space (areas):

Area Name	Access Width	Access Type	Usage
Discrete Inputs	1-bit	Read Only	Read system alarms and warnings
Input Registers	16-bit registers	Read Only	Read state values
Holding Registers	16-bit registers	Read / Write	Read/write configuration

3.2 Address space version

The location object contains major and minor version of address space. Modbus client may check these values to verify compatibility with CCU software version.

- major version
  - incompatible change in address space
  - e.g. changed objects addresses
- minor version
  - added new object
  - added new values to object

NOTE: When a value will be removed from the system in the future, the value will be still accessible to ensure backward compatibility until major version will be changed.

3.3 Modbus Commands

The areas described in the previous chapter can be accessed using following commands:

Code	Command	Area
0x02	Read Discrete Inputs	Discrete Inputs
0x04	Read Input Registers	Input Registers
0x03	Read Holding Registers	Holding Registers
0x06	Write Single Register	Holding Registers
0x10	Write Multiple Registers	Holding Registers

3.4 Error handling and return codes

3.4.1 Device booting

Device returns exception code SERVER DEVICE BUSY (06) during startup, because data integrity cannot be guaranteed during startup.

3.4.2 Invalid value

If a measured value is not initialized - e.g. due to failure or long response time from wireless peripherals - then INVALID_VALUE is returned as a response to read command. INVALID_VALUE is described in chapter Data Types below.

3.4.3 Data validation

When a configuration data is set, then it is validated and can be modified by system to meet the system requirements or it can be rejected.

- If value is lower than minimum, then it is set to minimum
- If value is higher than maximum, then it is set to maximum
- If value is not aligned to step, it is aligned (e.g. temperature 15.2 is aligned to 15.0)
- If a string value (val_utf8) is longer than the device is able to store, the string is shortened
- If it is not possible to set the value (e.g. because it is not supported), then exception code ILLEGAL_DATA_VALUE (03) is returned.

3.4.4 Removed values

When a value will be removed in a future firmware, it will stay in the address space so reading this value will never cause read error.

3.5 Data types

Type	Length	Range	INVALID_VALUE
val_enum	1B	0 .. 255	0xFF
val_u	1B	0 .. 255	0xFF
val_u2	2B	0 .. 65535	0xFFFF
val_u4	4B	0 .. 4294967295	0xFFFFFFFF
val_utf8	2B LEN + UTF8	UTF8 .. max 256B	LEN = 0xFFFF, no data
val_d2_fp100	2B	fixed-point (-327,68 .. 327,67)	0x7FFF

3.6 Reading and writing text values (datatype val_utf8)

Reading and writing of text values requires a special approach. We need to store some text with variable length into Modbus fixed-length registers.

This is solved using these tools

- Address space definition

  For each text value, the manufacturer reserves a specific address space represented by a group of consecutive registers. The number of these registers defines the maximum length of the stored text. For example, 16 registers are reserved for the location name, what means 32 bytes. See Chapter 4 for description of reserved address spaces.

- Marking the end of text

  If the text is shorter than the reserved address space, the rest of the space is filled with zeros. As soon as we read a byte with zero value, we know that we reached the end of the text.

- Saving text length

  The system automatically calculates and saves the length of the written text into the first register of reserved address space. So the text itself is written or read from the second register. The length of the text is the number of bytes of its data representation, not the number of text characters. One character at UTF8 may be represented by several bytes.

Before writing, we must first convert the text into its data (numeric) representation. The system uses UTF8 encoding. This creates a string of values, each occupying 1 byte. We divide this string into groups of 2 bytes and write these groups one by one into the registers of the specified address space. Finally, we will write a zero-valued byte.

3.6.1 Summary

- Reading
  - The first Modbus register contains the length of text converted to UTF8 encoding.
  - Each following register contains two bytes of text converted to UTF8 encoding.
  - Unused registers contain a zero value.
- Writing
  - Do not write to the first register. Its value is calculated automatically by device.
  - Write bytes of text converted to UTF8 encoding to registers (2 bytes per register).
  - Write zero value to the first unused byte.
  - All values written after the first zero are also automatically set to zero.

3.6.2 Example

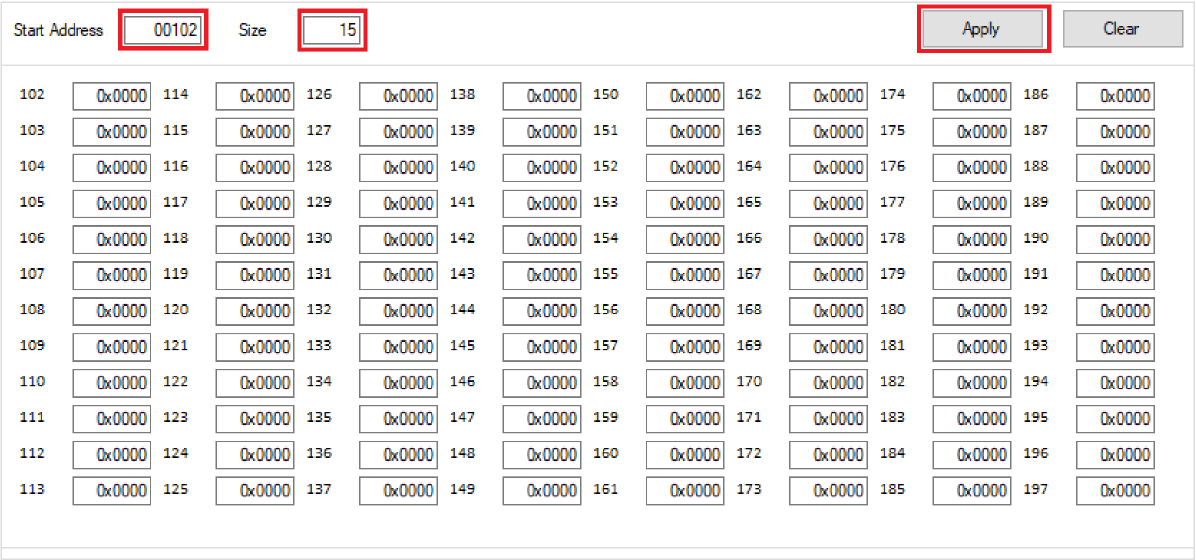
The following procedure will change the name of room 1 to "Blå Værelse". Note: The room must exist in the device.

1. Convert text to UTF8 encoding. Note that national characters are encoded into multiple bytes. We see that 13 bytes will be needed to save the entire text.

Character	Value
B	0x42
l	0x6c
å	0xc3 0xa5
space	0x20
V	0x56
æ	0xc3 0xa6
r	0x72
e	0x65
l	0x6c
s	0x73
e	0x65

2. In Chapter 4.3.3 we can find the registers reserved for the name of room 1.
   a. These are registers 00101 - 00116.
   b. Register 00101 is read only, used by system for text length storing.
   c. So we have 15 registers 00102 - 00116, which corresponds to 30 bytes.

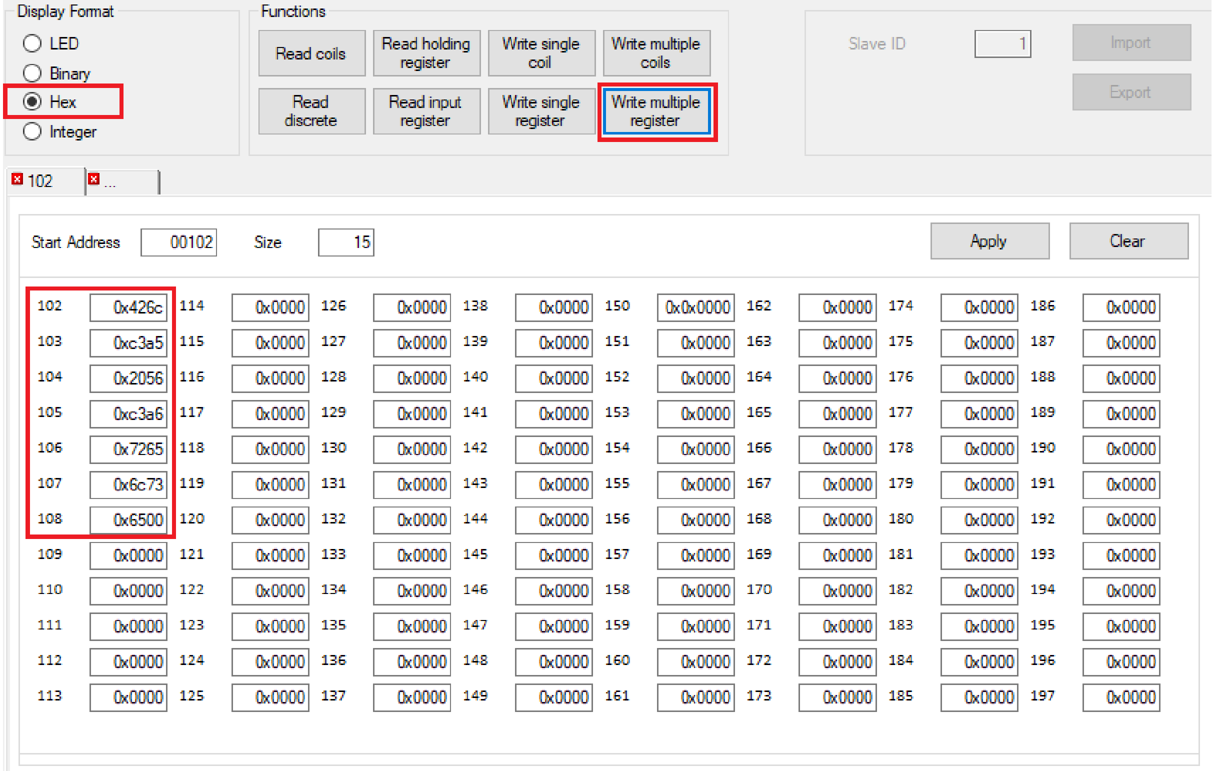
3. Choose these registers in the *Modbus Master* application.
   a. Enter the address of the first written register 00102 in the *Start Address* field. Note: It is possible to write without the leading zeros only as 102.
   b. Enter value 15 in the *Size* field. This is the maximum number of writable registers in reserved address space.
   c. Confirm with the *Apply* button.



Start Address 00102 Size 15 Apply Clear

102	0x0000	114	0x0000	126	0x0000	138	0x0000	150	0x0000	162	0x0000	174	0x0000	186	0x0000
103	0x0000	115	0x0000	127	0x0000	139	0x0000	151	0x0000	163	0x0000	175	0x0000	187	0x0000
104	0x0000	116	0x0000	128	0x0000	140	0x0000	152	0x0000	164	0x0000	176	0x0000	188	0x0000
105	0x0000	117	0x0000	129	0x0000	141	0x0000	153	0x0000	165	0x0000	177	0x0000	189	0x0000
106	0x0000	118	0x0000	130	0x0000	142	0x0000	154	0x0000	166	0x0000	178	0x0000	190	0x0000
107	0x0000	119	0x0000	131	0x0000	143	0x0000	155	0x0000	167	0x0000	179	0x0000	191	0x0000
108	0x0000	120	0x0000	132	0x0000	144	0x0000	156	0x0000	168	0x0000	180	0x0000	192	0x0000
109	0x0000	121	0x0000	133	0x0000	145	0x0000	157	0x0000	169	0x0000	181	0x0000	193	0x0000
110	0x0000	122	0x0000	134	0x0000	146	0x0000	158	0x0000	170	0x0000	182	0x0000	194	0x0000
111	0x0000	123	0x0000	135	0x0000	147	0x0000	159	0x0000	171	0x0000	183	0x0000	195	0x0000
112	0x0000	124	0x0000	136	0x0000	148	0x0000	160	0x0000	172	0x0000	184	0x0000	196	0x0000
113	0x0000	125	0x0000	137	0x0000	149	0x0000	161	0x0000	173	0x0000	185	0x0000	197	0x0000

4. Write the bytes of text converted to UTF8 into the fields of the reserved registers.
    a. Enter 2 bytes in each register field. If *Display Format* is set to *Hex*, we can simply write 2 numeric values next to each other into each field.
    b. Write 0x00 value after 0x65 in the register 108 field. This will mark the end of the text, and the values eventually written to registers 109 to 116 will be ignored.



Display Format

○ LED
○ Binary
● Hex
○ Integer

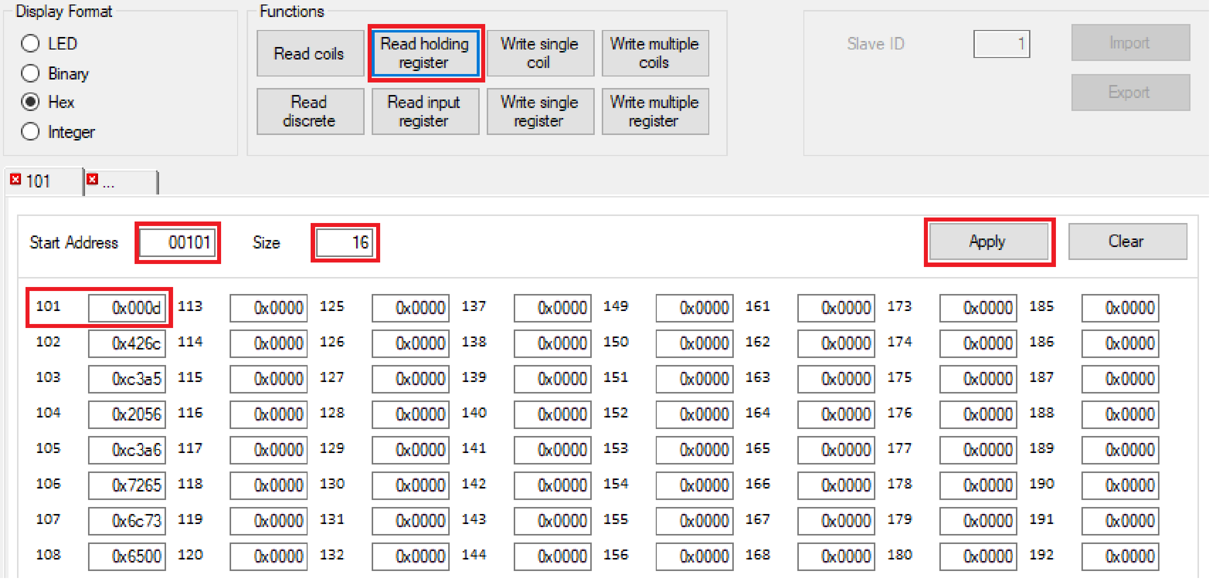
Slave ID 1

Start Address 00102 Size 15 Apply Clear

102	0x426c	114	0x0000	126	0x0000	138	0x0000	150	0x0x0000	162	0x0000	174	0x0000	186	0x0000
103	0xc3a5	115	0x0000	127	0x0000	139	0x0000	151	0x0000	163	0x0000	175	0x0000	187	0x0000
104	0x2056	116	0x0000	128	0x0000	140	0x0000	152	0x0000	164	0x0000	176	0x0000	188	0x0000
105	0xc3a6	117	0x0000	129	0x0000	141	0x0000	153	0x0000	165	0x0000	177	0x0000	189	0x0000
106	0x7265	118	0x0000	130	0x0000	142	0x0000	154	0x0000	166	0x0000	178	0x0000	190	0x0000
107	0x6c73	119	0x0000	131	0x0000	143	0x0000	155	0x0000	167	0x0000	179	0x0000	191	0x0000
108	0x6500	120	0x0000	132	0x0000	144	0x0000	156	0x0000	168	0x0000	180	0x0000	192	0x0000
109	0x0000	121	0x0000	133	0x0000	145	0x0000	157	0x0000	169	0x0000	181	0x0000	193	0x0000
110	0x0000	122	0x0000	134	0x0000	146	0x0000	158	0x0000	170	0x0000	182	0x0000	194	0x0000
111	0x0000	123	0x0000	135	0x0000	147	0x0000	159	0x0000	171	0x0000	183	0x0000	195	0x0000
112	0x0000	124	0x0000	136	0x0000	148	0x0000	160	0x0000	172	0x0000	184	0x0000	196	0x0000
113	0x0000	125	0x0000	137	0x0000	149	0x0000	161	0x0000	173	0x0000	185	0x0000	197	0x0000

5. Press the *Write multiple register* button to write the text value. Now we can verify the result by displaying the room name on the LCD.

6. Now we can try to read this text.
   a. Change the entry in the *Source Address* field to 00101.
   b. Enter 16 in the *Size* field.
   c. Confirm with the *Apply* button.
   d. Press the *Read holding register* button.
   e. In the register 101 we can now see 0x000d, which corresponds to the number of 13 bytes entered by us.



Display Format: ○ LED ○ Binary ● Hex ○ Integer

Slave ID 1

Start Address 00101 Size 16

101	0x000d	113	0x0000	125	0x0000	137	0x0000	149	0x0000	161	0x0000	173	0x0000	185	0x0000
102	0x426c	114	0x0000	126	0x0000	138	0x0000	150	0x0000	162	0x0000	174	0x0000	186	0x0000
103	0xc3a5	115	0x0000	127	0x0000	139	0x0000	151	0x0000	163	0x0000	175	0x0000	187	0x0000
104	0x2056	116	0x0000	128	0x0000	140	0x0000	152	0x0000	164	0x0000	176	0x0000	188	0x0000
105	0xc3a6	117	0x0000	129	0x0000	141	0x0000	153	0x0000	165	0x0000	177	0x0000	189	0x0000
106	0x7265	118	0x0000	130	0x0000	142	0x0000	154	0x0000	166	0x0000	178	0x0000	190	0x0000
107	0x6c73	119	0x0000	131	0x0000	143	0x0000	155	0x0000	167	0x0000	179	0x0000	191	0x0000
108	0x6500	120	0x0000	132	0x0000	144	0x0000	156	0x0000	168	0x0000	180	0x0000	192	0x0000

4 Modbus registers

4.1 Objects overview

See Address space for more details how the address space is divided to objects.

Address	Max Count	Name	Description
000xx	1	Location	Location can be a house, cottage or a floor (section) in a building which is controlled by CCU-208. This object contains values shared across all objects in the location.
001xx ... 064xx	64	Room	Room in Location.
065xx	1	DHW Calefa	Domestic hot water controller.

4.2 Object Location (000)

4.2.1 Discrete inputs (R)

Address	Name	Description	TR416 - BigTable
00001	Aggregated warning	A problem is pending in whole system (Location)	-
00002	Aggregated error	A critical problem is pending in whole system (Location)	-

4.2.2 Input register (R)

Address	Data Type	Name	Description	TR416 - BigTable
00001	val_u1	Address space major version	= 1 (Incremented on incompatible change)	-
00002	val_u1	Address space minor version	= 0 (Incremented on compatible change)	-

00003-00009	-	-	reserved for modbus related things	-
00010	val_enum	Dev type	1 CCU-208 2 DHW-201 (Calefa)	-
00011	val_u1	Dev hw version		device.hw_version
00012	val_u1	Dev sw version		device.sw_version
00013	val_u1	Dev sw version minor		device.sw_version_minor
00014	val_u2	Dev serial number prefix	= 1530	device.serial_prefix
00015-00016	val_u4	Dev serial number		device.serial
00017-00019	-	-	reserved for additional device descriptors	-

4.2.3 Holding register (R/W)

Address	Data Type	Name	Description	TR416 - BigTable
00001	val_u1	Address space major version	= 1 (Incremented on incompatible change)	-
00002	val_u1	Address space minor version	= 0 (Incremented on compatible change)	-
00003	val_u1	Modbus slave address	Allowed values: 1 to 247 Default: 1	location.modbus_addr
00004	val_u2	Modbus baudrate	Allowed values: 9600, 19200, 38400, 57600 Default:19200	location.modbus_baudrate
00005	val_u1	Modbus mode	0 DISABLED 1 READ_ONLY 2 READ_WRITE Default: 2	location.modbus_mode
00006 - 00009	-	-	Reserved for modbus related things	-
00010 - 00025	val_utf8	Location name	Placeholder for 32 bytes of location description. See “working with strings” chapter for more info.	location.name
00026	val_u1	Standby	0 OFF 1 ON	location.standby
00027	val_u1	Vacation	0 OFF 1 ON	location.vacation

4.3 Object Room (001 - 064)

xxx = room number (001 – 064)

4.3.1 Discrete inputs (R)

Address	Name	Description	TR416 - BigTable
xxx01	Aggregated warning	A problem is pending in Room	-
xxx02	Aggregated error	A critical problem is pending in Room	-
xxx03	Warning - low battery	There are one or more peripherals in the room with low battery.	room.warn_periph_low_batery
xxx04	Error - peripheral lost	There are one or more peripherals in the room which are not responding.	room.alarm_periph_unreachable

4.3.2 Input register (R)

Address	Data Type	Name	Description	TR416 - BigTable
xxx01	val_d2_fp100	Desired temp	Shows the desired temperature in the room. When "Room mode override" > NONE, then this temperature is defined by override mode. When "Room mode" == SCHEDULE, then this temperature is defined by room schedule. When ("Room mode" == MANUAL) AND ("Room mode override" == NONE) then this temperature is defined by "Room temperature setpoint"	room.tmp_ctrl.temp_desired
xxx02	val_enum	Heating/Cooling state	1 IDLE 2 HEATING 3 COOLING Shows, whether the system wants to heat or wants to cool in the room. However, this request can be blocked – See "Heating/Cooling blocked" value. You have to combine this value with "Heating/Cooling blocked" value to get the real state.	room.tmp_ctrl.state
xxx03	val_enum	Heating/Cooling blocked	0 NONE 1 UNKNOWN 2 CONTACT 3 FLOOR 6 DEW_POINT If the value is higher than 0 (NONE) then the heating / cooling is blocked.	room.tmp_ctrl.blocking
xxx04	val_d2_fp100	Air temperature	Shows air temperature measured in the room.	room.air_temp
xxx05	val_d2_fp100	Floor temperature	Shows floor temperature measured in the room.	room.floor_temp

xxx06	val_d2_fp100	Relative humidity	Shows humidity measured in the room.	room.humidity

4.3.3 Holding register (R/W)

Address	Data Type	Name	Description	TR416 - BigTable
xxx01 - xxx16	val_utf8	Room name	Placeholder for 32 bytes of room description. See “working with strings” chapter for more info.	room.name
xxx17	val_enum	Room mode	0 SCHEDULE 1 MANUAL In SCHEDULE mode, the “Room temperature setpoint” is not used and the room temperature is controlled by scheduler.	room.mode
xxx18	val_enum	Room mode override	0 NONE 1 TEMPORARY 2 VACATION_AWAY 3 ADJUST In override mode (> NONE), the “Room temperature setpoint” is not used. The requested temperature is corrected by user via room thermostat or mobile application. You can disable the override mode by setting this value to 0 (NONE)	room.mode_override
xxx19	val_d2_fp100	Room temperature setpoint	Temperature requested by user. This value may be not used – see “Room mode” and “Room mode override” values description.	tmp_ctrl.heat.temp_set_manual tmp_ctrl.cool.temp_set_manual

4.4 Object DHW Calefa (065)

4.4.1 Discrete inputs (R)

Address	Name	Description	TR416 - BigTable
06501	Aggregated warning	A problem is pending in DHW	-
06502	Aggregated error	A critical problem is pending in DHW	-

06503	Warning - Retentive Low Energy		dhw.calefa_ctrl.warn_low_energy
06504	Error - DHW Temp High		dhw.calefa_ctrl.alarm_high_temp
06505	Error - Motor failure		dhw.calefa_ctrl. alarm_motor_fail
06506	Error - DHI sensor failure		dhw.calefa_ctrl. alarm_dhi_sensor_fail
06507	Error - DHO sensor failure		dhw.calefa_ctrl. alarm_dho_sensor_fail
06508	Error - DHW sensor failure		dhw.calefa_ctrl. alarm_dhw_sensor_fail
06509	Error - DCW sensor failure		dhw.calefa_ctrl. alarm_dcw_sensor_fail

4.4.2 Input register (R)

Address	Data Type	Name	Description	TR416 - BigTable
06501	val_d2_fp100	Desired DHW temp	Shows the desired temperature of the domestic hot water.	dhw.calefa_ctrl.temp_desired
06502	val_enum	State	1 IDLE 2 HEATING 3 BYPASS Shows, whether the system wants to heat or to have bypass activated. However, this request can be blocked – See “Blocked” value.	dhw.calefa_ctrl.state
06503	val_enum	Blocked	0 NONE 1 UNKNOWN 2 BMS If the value is higher than 0 (NONE) then the heating / bypass is blocked.	dhw.calefa_ctrl.blocking
06504	val_d2_fp100	Circulation state	0 IDLE 1 ON	dhw.calefa_ctrl. circ_state

4.4.3 Holding register (R/W)

Address	Data Type	Name	Description	TR416 - BigTable
06501 - 06516	val_utf8	DHW name	Placeholder for 32 bytes of dhw description. See “working with strings” chapter for more info.	dhw.name
06517	val_enum	DHW mode	0 SCHEDULE 1 SCHEDULE_ADAPTIVE 2 ECO 3 COMFORT Eco = circulation and bypass are disabled Comfort = circulation and bypass are enabled	dhw.mode
06518	val_enum	Display access level	< 40 USER (user menu) >= 40 INSTALLER (inst. menu)	dhw.access_level

06519	val_u1	Block request	0 NONE 1 BLOCK_REQUEST When BLOCK_REQUEST is set, then the system blocks heating and bypass to eliminate consumption from heat supplier.	dhw. calefa_ctrl.block_by_bms
06520	val_u2	Power consumption limit		dhw. calefa_ctrl.pwr_consumption_limit
06521	val_d2_fp100	DHW temperature	Requested temperature of domestic hot water.	dhw. calefa_ctrl.temp_set
06522	val_d2_fp100	DHW bypass temperature		dhw. calefa_ctrl.bypass_temp_set
06523	val_enum	Circulation mode	0 SCHEDULER 1 PERMANENT	dhw.circ_ctrl.circ_mode
06524	val_d2_fp100	Circulation temperature		dhw.circ_ctrl. circ_temp