Guideline on controlling the Logamatic 5000/Control 8000 controller series via Modbus TCP/IP

This document helps you to realize the connection and control of the Logamatic 5000/Control 8000 controller series via Modbus TCP/IP with your building control system.

Contents

1. General information about the control of the Logamatic 5000/Control 8000 controller series via Modbus TCP/IP

If you want to control the controller from a building management system (BMS) via Modbus TCP/IP, connect the network interface 1 (LAN 1) of the controller (see Figure 1-1: Module BCT 531 - Representation of the available interfaces) to your machine network, in which the BMS is also located, using a network cable. You can then start up the controller properly (see the documentation "Service manual for the specialist - Logamatic 5311/Control 8311" or "Service manual for the specialist - Logamatic 5313/Control 8313").

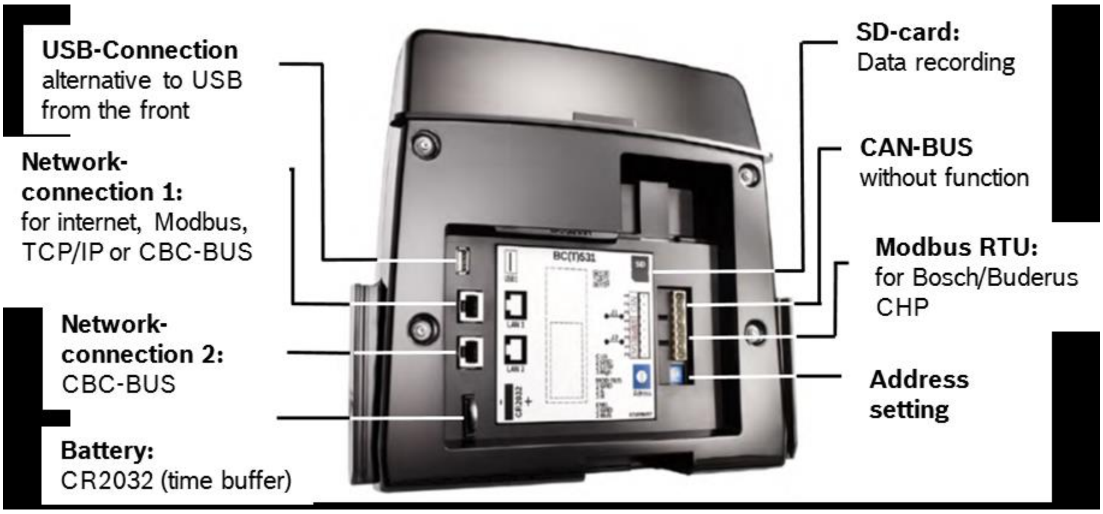


Figure 1-1: Modul BCT 531/BCT 831 – Representation of the available interfaces

2. Activating the controller via Modbus TCP/IP

In the following section you will find an excerpt of the cover sheet of the data point list of the controller, which is relevant for the setup and use of the Modbus TCP/IP communication. The complete data point list can be downloaded from the Buderus website.

The data of the Modbus addresses are listed in decimal format. To convert the addresses to the Modicon scheme, please refer to the chapter "General information about the conversion of the Modbus address scheme". When using the data points, please make sure that you select the correct register types and carry out any necessary conversions of the received values. The conversions are necessary to transfer decimal places.

The Modbus TCP/IP server of the controller supports the following nine Modbus function codes and data types:

Function Name	Function Code
Read Coils	01
Read Discrete Inputs	02
Read Multiple Holding Register	03
Read Input Registers	04
Write Single Coil	05
Write Single Holding Register	06
Write Multiple Coils	15
Write Multiple Holding Registers	16
Read/Write Multiple Registers	23
Data Type	**Description**
Coils	Read- and Writeable, 1 Bit
Discrete Inputs	Readable, 1 Bit
Input Registers	Readable, 16 Bit
Holding Registers	Read- and Writeable, 16 Bit

Note:

- Currently the data types Coils and Discrete Inputs are not used by the controller. Therefore the function codes 01, 02, 05 and 15 have no relevance for the control of the controller.

- Starting with firmware version 1.3.X, the control, parameterization and some Modbus addresses of the controller via Modbus TCP/IP have changed. Therefore, check during commissioning which firmware your controller has (see chapter "Checking the current firmware of the controller") and, if necessary, perform a firmware update first. If you already control a controller with a version below 1.3.X via Modbus TCP/IP, you can activate the use of old Modbus addresses in your controller (see chapter "Necessary settings on the controller"). If you have activated the Modbus compatibility mode, you do not have to make any changes to your current solution for control via Modbus TCP/IP (see chapter "Outdated software versions and their peculiarities"). Updating the firmware to the latest version is recommended.
- The decision whether to use the decimal or modicon scheme depends on the software used and the configuration of the BMS. Please contact the manufacturer of your BMS for further information on the correct scheme.

2.1 Determining relevant Modbus TCP/IP data points

The data points provided by the controller or specified in the data point list vary depending on the hardware modules installed and their configuration. Since the controller provides all data points in the list, not all data points are relevant for you.

If, for example, you have not connected a boiler to the controller and want to read out the corresponding data points of the boiler, you will only receive invalid values (e.g. flow temperature of boiler = 0 °C). When writing data points that are not available in the control system, you can set the values, but they have no influence on the behavior of the controller.

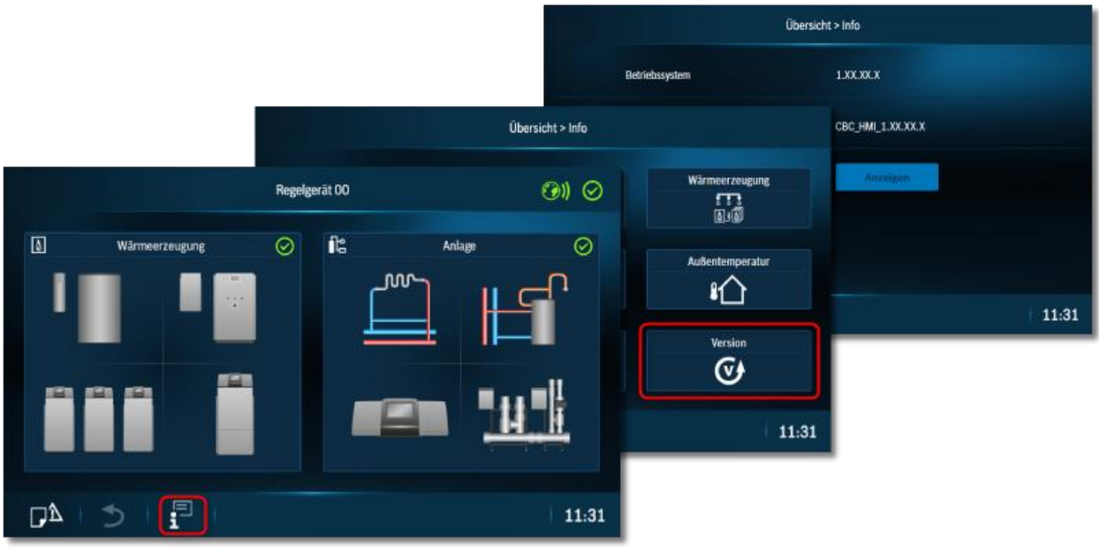
The data point list is structured according to components. If, for example, you do not use a boiler, you can ignore any data points that refer to the boiler component. Therefore, before using the data point list, check which data points are suitable for your application. Your Buderus representative can assist you in selecting the appropriate data points for your system.

If you use heating circuit modules, you must also pay attention to the slot. The slot determines the number of the heating circuit. You should also make sure that you have configured (e.g. activated) the components (heating circuits, boiler, etc.) in the controller accordingly.

2.2 Checking the current firmware of the controller

You can check the firmware used in the controller as follows:

First switch to the information menu. Click on the "Version" menu item here. You will find the current firmware in the "Operating system" parameter. If the displayed firmware is smaller than 1.3.X or if you have activated the mode "Modbus compatibility for firmware smaller than 1.3.6", you have to use the old Modbus data point list and control (see chapter "Outdated software versions and their peculiarities").



Betriebssystem 1.XX.XX.X

CBC_HMI_1.XX.XX.X

Figure 2-1: Information Menu

If, on the other hand, a firmware greater than or equal to 1.3.X is displayed, please use the Modbus data point list with the new control that matches the firmware currently in use (see chapter "Control of the controller via Modbus TCP/IP from and including version 1.3.X").

2.3 Necessary settings on the controller

First switch on the control and wait until it is completely ready for operation. On the user interface, you can then open the service menu with a long click on the "Fault message overview" icon at the bottom left.

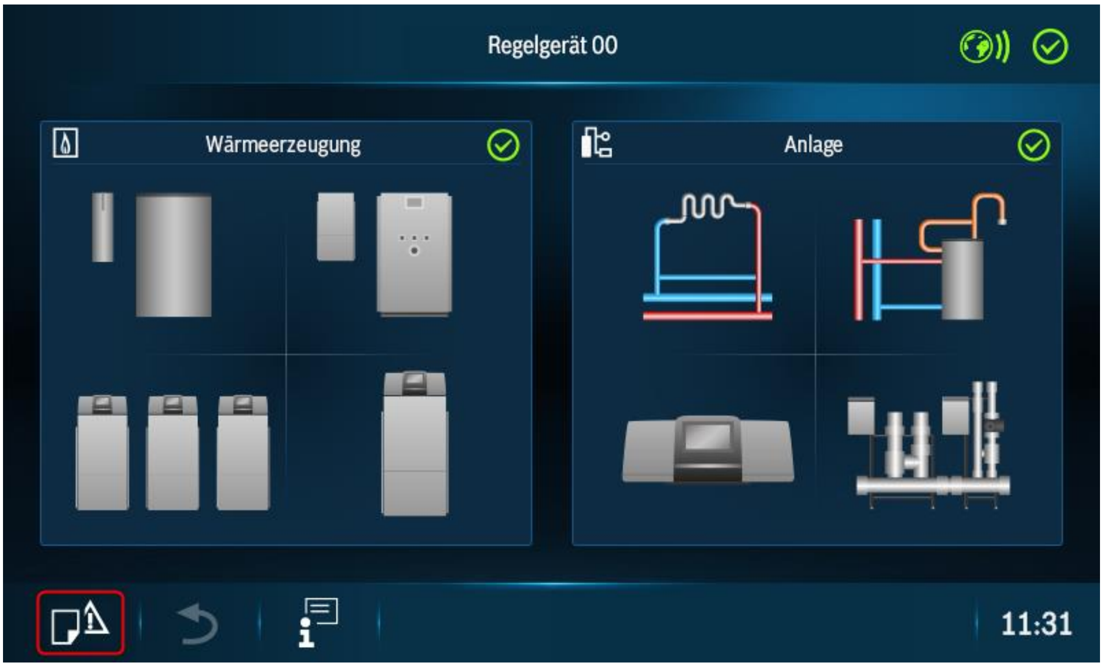


Figure 2-2: Overview with button to settings menu

Here you switch to the menu “Connectivity”

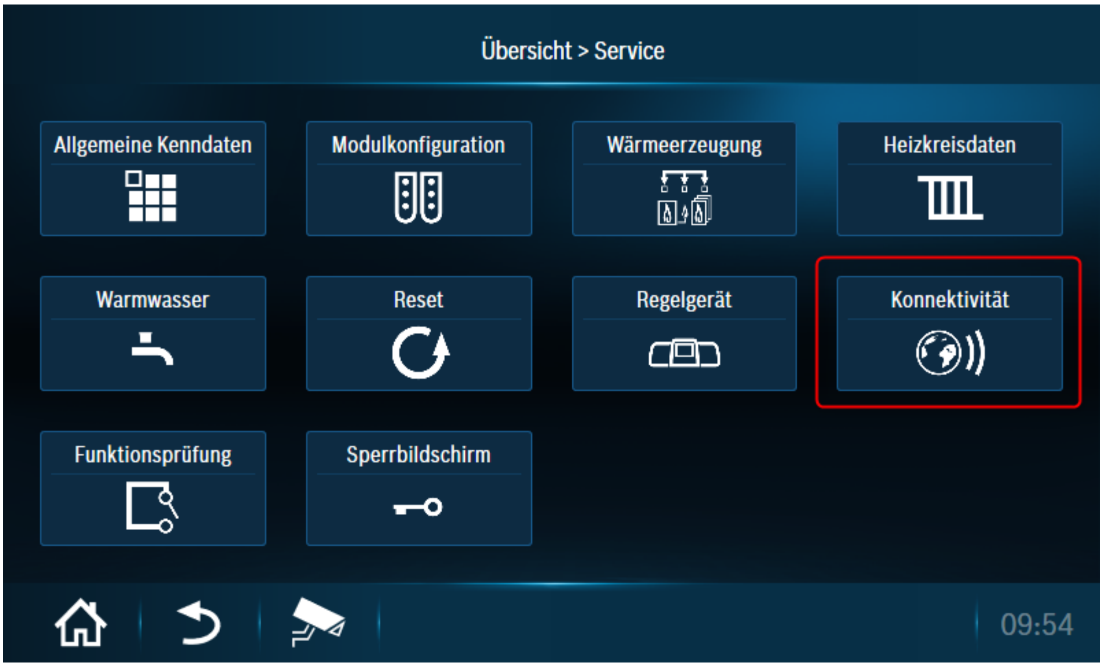


Figure 2-3: Settings Menu

In the following you have to make some settings for the Modbus TCP/IP communication at the controller. In particular, you must pay attention to the settings shown in bold, which may differ from the factory settings in your control device:

Name	Parameters to be selected	Importance
LAN1	**Modbus TCP/IP** (Factory setting: internet)	The controller activates the Modbus TCP/IP functionality, which is necessary for data transfer.
Modbus communication	**With Heart Beat / Without Heart Beat** (Factory setting: No)	The controller uses a Modbus TCP/IP connection with/without heartbeat. For further information, please refer to the chapter General Information on Modbus TCP/IP Heart Beat.
Time overrun	120 to 600 seconds (Factory setting: 180 seconds)	Defines the timeout of the Modbus TCP/IP heartbeat. If the time is exceeded until a new counter value is received, an error message is generated.
Write access	**On** (Factory setting: On)	Enables write access via Modbus TCP/IP and is therefore a prerequisite for external setpoint setting via Modbus TCP/IP.
Providing data from substations	**On** (Factory setting: Off)	Activates the provision of data from internal control components as well as from substations connected via Modbus TCP/IP. This parameter must always be activated for communication via Modbus TCP/IP!
Heat request only via Modbus (Only valid until Firmware 1.2.7.X)	**On** (Factory setting: Off)	If activated, a heat request can only be accepted via Modbus TCP/IP. Heat requirements that may be generated by built-in function modules are not taken into account. For further information, please refer to the chapter "Information about parameter “Heat requirement only via Modbus”". **Note**: This parameter is only valid until Firmware 1.2.7.X.
Address assignment	Static / DHCP	If you do not have a DHCP server in the machine-network, select static for the address assignment and then set the IP address, the network mask and the gateway - otherwise you can select DHCP. Hint: The limitation of the IP address is that you are not allowed to use the IP address range 172.31.42. X because it is used for internal purposes of the regulation.
Activate Modbus compatibility for Firmware lower 1.3.6	Off (Factory setting: Off)	Activates or deactivates the Modbus-compatibility mode. Please, use this mode only, if you already a controller to version 1.2.7. X head via Modbus TCP/IP by your GLT. Provided that the Modbus-compatibility mode was activated, you can Reuse the old Modbus data points / activation without any change in your existing solution.

If you have made all settings concerning the Modbus TCP/IP communication, you can apply the settings via the "Save" button. The "Cancel" button is used to discard any changes that have been made. As soon as you have saved the changed connectivity settings, the controller is ready for communication via Modbus TCP/IP.

To define which components can generate a heat request at the controller, please refer to the chapter "Information on 'Heat request via Modbus' and 'Internal heat request' parameters'" for the necessary configuration.

Note: Up to version 1.2.7.X, this setting was controlled via the "Heat request only via Modbus" parameter (see chapter "Information on 'Heat request only via Modbus' parameter" and chapter "Outdated software versions and their peculiarities").

2.4 Network behavior and diagnostic options of the controller

When using Modbus TCP/IP, the Modbus Device ID (Unit Identifier) of the controller is set to 255 by default. This can be changed from version 1.5.13 (see chapter "Necessary settings on the controller"). In addition, when using Modbus TCP/IP, any values are transmitted in the byte order Big-Endian (Most Significant Bit (MSB) First) also known as "order3210" or "or-derDCBA" (see chapter "Data types and Register Width").

Up to software version 1.5.X it was not possible to address the controller with the network diagnostic tool "Ping". This has been changed as of software version 1.5.X: Activate this function by setting the "Local Area Network" parameter to "Modbus TCP/IP" (see chapter "Necessary settings on the controller"). This way the controller responds to the ping request and you can see that the controller is reachable in the network.

2.5 Data types and Register Width

The controller supports the following data types, which must be used via Modbus TCP:

Data Type	Bits	Bytes	Value Range
Bit (bit)	1	0	From 0 to 1
Integer (int)	16	2	From 0 to 65535
Signed Integer (signed int)	16	2	From -32768 to +32767
Long (long)	32	4	From -2147483648 to +2147483647

The Modbus registers have in each case a width of 16 bits. On this occasion, the registers are built up as follows:

	Register															
Valency	MSB															LSB
Bit number	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Byte number	2								1							

Hint: If you read a register with the data type "Long", you must take into account that the value is divided into two successive registers, since the memory of only one register would not be sufficient to map such a large number. In this case the register to be read looks as follows:

	Register (High)																**Register (Low)**															
Valency	MSB															LSB	MSB															LSB
Bitnumber	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bytenumber	4								3								2								1							

2.6 Control of the controller via Modbus TCP/IP from and including version 1.3.X

Please use the following control of the controller from firmware "1.3.X".

Register type	Address	Bit	Data point name	Data type	Value Range	Importance
Holding Register	0		Heartbeat (in)	INT		If the connection is to be monitored for timeout, the counter that is different from the old value, which can be read via the Heart Beat (out) register, must be written on this register.
Holding Register	1		Heartbeat (out)	INT		If the register is read, the current counter value of the heart beat (in) can be called up.
Holding Register	400		Set-point flow temp. in °C	INT	From 0 to 120	Set-point of the currently required boiler flow temperature. **Hint:** The maximum system temperature depends on the installed boiler and there power.
Holding Register	401		Set-point system power in %	INT	From 0 to 100	Set-point of the topically demanded system output.
Holding Register	402		Mode of operation	INT	0 (Off) or 1 (Operation)	Operating mode for boiler request. 0 = Off = Boilers are off, consumers continue to be controlled 1 = Operation = normal operation over boiler heat request
Holding Register	405		Bit block for Cascade control	BIT		See the following table entries
Holding Register	405	0	Lead boiler	BIT	0 (inactive) or 1 (active)	If you implement your own cascade control system as BMS, you can sign the current boiler as the guide boiler via this bit. Through this, the follow-up times of the boiler circuit pump will be adjusted. **Hint**: The functionality is not implemented yet and is made available at a later time than software update.
Holding Register	405	1	Prioritization	BIT	0 (inactive) or 1 (active)	Allows the boilers to be prioritized. This is used, for example, for a faster warm water demand. **Hint**: The functionality is not implemented yet and is made available at a later software update.
Holding Register	405	2	Temperature-controlled regulation	BIT	0 (inactive) or 1 (active)	Activates or deactivates the temperature controlled operation of the boiler.

Holding Register	405	3	Power-controlled regulation	BIT	0 (inactive) or 1 (active)	Activates or deactivates the power controlled operation of the boiler.

Notes:

- Data type Integer (int) corresponds to 16 bits (Bit0 to Bit15), see chapter "Data types and register width".
- The addresses mentioned here refer to the fact that the controller is configured as master. If you use the controller as slave, please refer to chapter "Control via Modbus TCP/IP in the controller network".
- To set individual bits in a Modbus register (as required for Holding Re-gister 405), please refer to chapter "Supplementary information on setting or reading individual bits in a Modbus register".
- The data point "Setpoint system supply temperature in °C" affects the strategy in the controller. Here, the specific structure of the system or module configuration in the controller determines which heat generators are activated or deactivated by the controller.
- A power-steered control can only be used if one or more boilers of the same type are connected to the control device. This function is not possible with several different heat generators.

2.6.1 Temperature-controlled heat request via Modbus TCP/IP from version 1.3.X

To be able to send a temperature-controlled heat request via Modbus, you must first configure the controller accordingly. Please note the information in chapter "Information about parameters 'Heat request via Modbus' and 'Internal heat request'" if you want to control the heat request exclusively via the BMS.

If you now want to generate a temperature-controlled heat request, you must describe the following registers:

Register	Value of Register
Operation Mode (Holding Register 402)	1
Flow Temperature Set point (Holding Register 400)	X in °C
Leading Boiler (Holding Register 405; Bit$_0$)	1
Temperature controlled operation (Holding Register 405; Bit$_2$)	1
Power controlled operation (Holding Register 405; Bit$_3$)	0

Note:

- With the values from the above example, a temperature-controlled heat request with X °C, which is to work as a lead boiler, is sent to a boiler. For a system with FM-CM, the guide boiler function must be enabled.
- X in °C corresponds to your desired flow temperature.

2.6.2 Power-controlled heat request via Modbus TCP/IP from and including version 1.3.X

To be able to send a power request via Modbus, you must first configure the controller accordingly (see chapter "Information on parameters 'Heat request via Modbus' and 'Internal heat request'").

Then you can write to the following Modbus registers:

Register	Value of Register
Operation Mode (Holding Register 402)	1
Flow Temperature Set point (Holding Register 400)	X in %
Leading Boiler (Holding Register 405; Bit_0)	1
Temperature controlled operation (Holding Register 405; Bit_2)	0
Power controlled operation (Holding Register 405; Bit_3)	1

Note:

- With the values from the above example, a power-controlled heat request with X %, which is to work as a lead boiler, is sent to a boiler. In the case of a system with FM-CM, it is mandatory to enable the guide boiler function.
- You can only use the output-guided control if you have deactivated the internal heat request.
- X in % corresponds to your desired output..

2.6.3 Information on 'Heat request via Modbus' and 'Internal heat request' parameters

You can use the following parameters of the controller to define how a heat or power request can be generated at the controller. The parameters for configuring the behavior can be found in the Service menu > Heat generation > Strategy data > Basic settings. Depending on the parameterization, this may have an effect on the control.

If you have connected additional function modules (e.g. for a heating circuit) or substations to the control, you can activate the heat request for these function modules via the "Internal heat request" parameter.

If you want to generate a heat request via Modbus, you must activate the parameter "Heat request via Modbus". Depending on which heat requests you have activated, the behavior of the control is influenced as follows:

Parametres „Internal heat request“	Parametres „Heat request through Modbus“	Impact
Off	Off	No heat requests from the control unit are taken into account.
On	Off	Only heat requests of internal function modules are considered.
Off	On	Only heat requests via Modbus are considered (exclusive mode).
On	On	Internal and Modbus heat requests are taken into account (parallel mode).

2.6.3.1 Exclusive mode of heat demand

In the exclusive mode of the heat request, any heat requests from internal function modules are ignored. This gives you full control from the BMS of any heat demand generated by the controller. In this mode you can use a heat request via a temperature controlled as well as a power controlled heat request.

2.6.3.2 Parallel mode of the heat request

In the parallel mode of the heat request, both heat requests from internal function modules and heat requests via Modbus are taken into account. A heat request via Modbus, which comes from the BMS, can only add a heat request. Existing heat requests from other modules are not influenced by this. The setpoint is always the highest heat demand. You cannot use a power-controlled heat request in this mode.

2.7 Control via Modbus TCP/IP in the network

With the controllers of the Logamatic 5000/Control 8000 controller series, it is possible to operate other controllers or components from the Buderus modular system Logaflow HSM plus in a network. This creates a hierarchical structure in which there can be one head station (master) and several substations (slaves). Additionally it is possible with Logaflow HSM plus to create so called segment substations. A segment substation is a substation that can itself include further substations in the network.

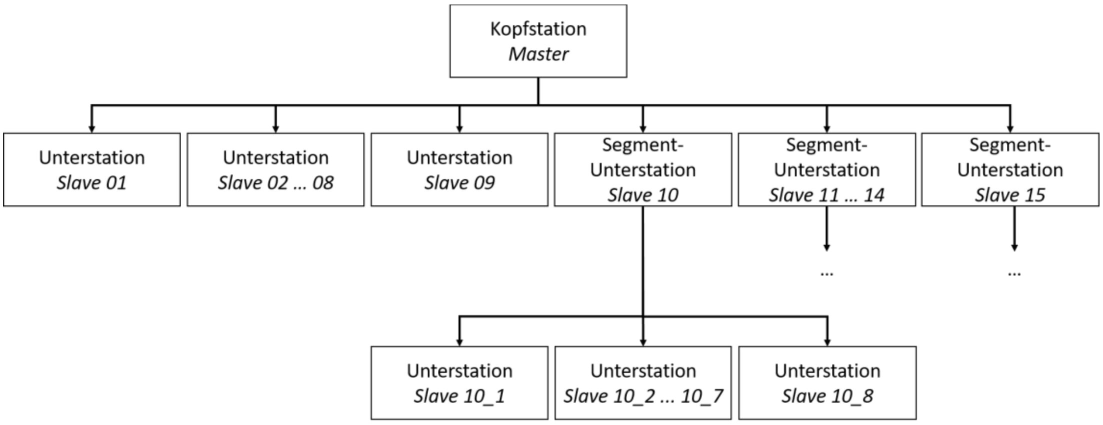


Figure 2-4: Functional principle of Network

2.7.1 Control via Modbus TCP/IP in the controller network

If you operate the controller in a controller network, i.e. you have connected a master controller and at least one slave controller, you can access the data points of the slave controller via the master controller. The behavior of the controller, whether it is to function as master or slave, is set via the rotary coding switch on the controller (see documentation "Service manual for the specialist - Logamatic 5311" or "Service manual for the specialist - Logamatic 5313").

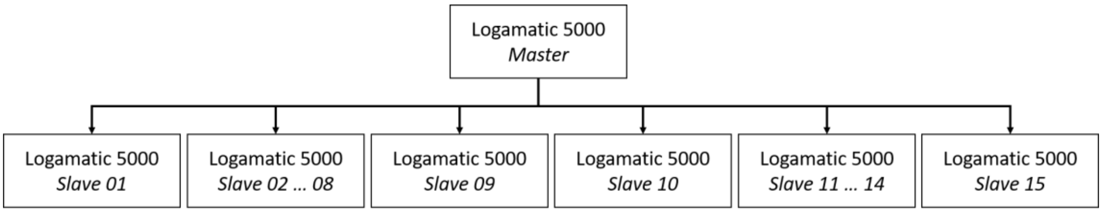


Figure 2-5: Overview of control unit network with Logamatic 5000

The data points between a master and a slave controller differ only in their Modbus address for addresses smaller than 8000. As soon as the controller acts as a slave, you can access the data points via the master controller using the following formula:

$$ModbusAddressSlave = ModbusAddressMaster + (Rotary\ coding\ switch * 500)$$

Note:

- The already calculated addresses can be found in the data point list (sheets Slave01 - Slave15).
- The formula can only be used for data points with address less than 8000. Data points from and including address 8000 are only available in the master.

Example:

Assuming you have two controllers (1x master and 1x slave) in use and have set up a controller network. The rotary coding switch on the master controller is set to position 0 (= master) and on the slave controller to position 1 (= slave 01). You have installed and configured a heating circuit module in slot 1 on the slave controller. If you now want to access the current flow temperature of heating circuit 1 from the slave controller, for example, you can calculate the address as follows:

$$ModbusAdressSlave = 102 + (1 * 500) = 602$$

- The Modbus address for the current flow temperature at heating circuit 1 is at input register 102 at the master controller.
- The position of the rotary coding switch on the slave controller is set to 1. The position is multiplied by 500 and added to the master Modbus address of the corresponding data point.
- The calculated address for requesting the current flow temperature of heating circuit 1 from the slave controller via the master controller is located on input register 602.

If you now request the calculated address (input register 602) at the master controller, you will receive the current flow temperature of heating circuit 1 from the slave controller.

Note:

- You will find the already calculated addresses of all positions of the rotary coding switch (slave 01-15 corresponds to the position of the rotary coding switch 1-15) also in the Modbus TCP/IP data point list.
- If you have connected one boiler each at the master and slave controller, you need a cascade module at the master controller to control both boilers via Modbus TCP/IP. If you have not installed a cascade module at the master controller and have set up a controller network, the boiler at the slave controller is blocked by the master controller. If you do not want to use the cascade module, you must implement the cascade functionality in your building management system. In this case, you must use both control devices as masters (no control device network may be created).

2.7.2 Control via Modbus TCP/IP in combination with Logaflow HSM plus

You can operate the control unit in a network with the Buderus Logaflow HSM plus modular system. For this, you need a connection between the master controller and at least one HSM plus device. You can then access the data points of the HSM plus via the master control device. The behavior of the controller, whether it is to function as master or slave, is set via the rotary coding switch on the controller. Further information can be found in the documentation "Service manual for the specialist -

Logamatic 5311/Control 8311" or "Service manual for the specialist - Logamatic 5313/Control 8313".

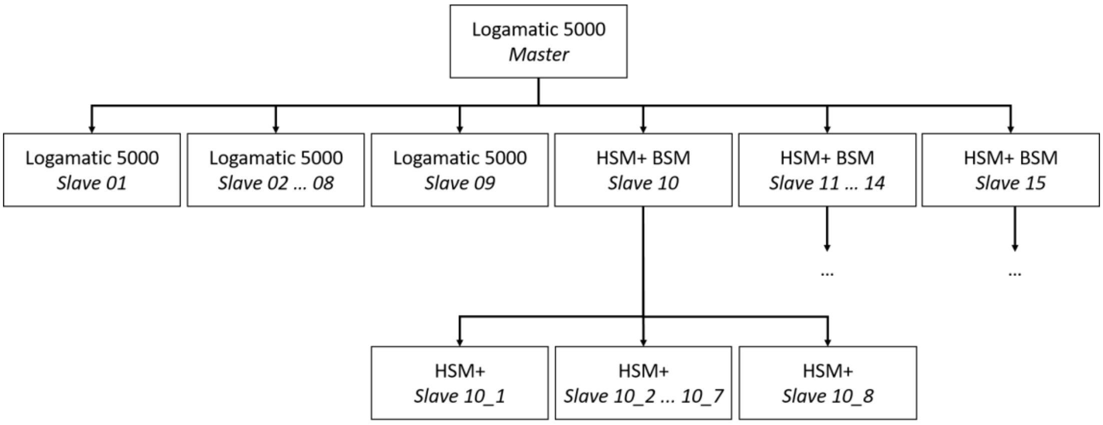


Figure 2-6: Overview network with Logamatic 5000 and Logaflow HSM plus

For the connection between Logamatic 5000 and Logaflow HSM plus the positions Slave10 to Slave 15 are provided. Up to six HSM plus BSM can be connected to one Logamatic 5000/Control 8000 (master). Each HSM plus BSM can be used as a segment substation. Thus, up to eight additional HSM plus devices can be connected to each HSM plus BSM.

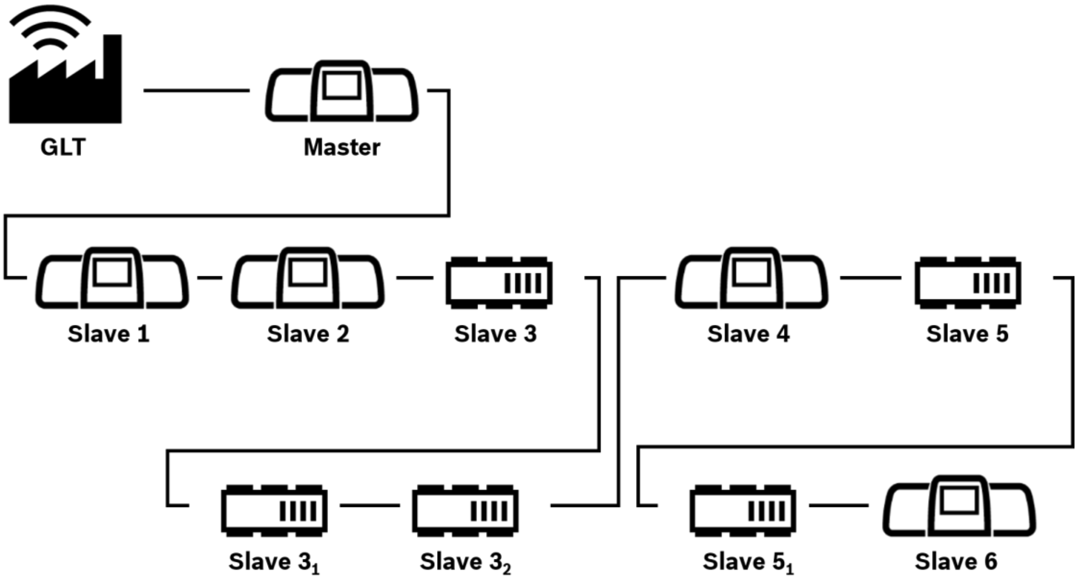


Figure 2-7: Sample Installation of a Network

Note:

Another controller of the Logamatic 5000/Control 8000 controller series is always integrated as a substation (slave). Use as a segment substation is not possible (see Figure 2 7).

Example:

If you connect a HSM plus BSM to a Logamatic 5000, it is automatically integrated as slave10 in the Logamatic 5000. A possible second, third, ... HSM plus BSM is integrated accordingly as slave11, slave12,... .

2.7.2.1 Access to Substation

All data points can be retrieved via the master using Modbus TCP. Since the addresses of the data points may differ depending on the system structure, you can use the following formulas to calculate the corresponding addresses:

First you have to calculate the required offset for the respective slave (10 - 15):

$$OffsetSlaveX = X * 500$$

You can now add this offset to the addresses of the HSM plus BSM. The result is the address of the data point with which you can access the respective BSM via the master.

$$AdresseViaMaster = OffsetSlaveX + Adresse$$

Example:

You want to retrieve the address "Input Register 1 - Own Flow Temperature" from slave11 via master.

$$\begin{aligned} OffsetSlave11 &= 11 * 500 = 5500 \\ AdresseViaMaster &= 5500 + 1 = 5501 \end{aligned}$$

2.7.2.2 Access to substation via segment substation

You can also access the substations of an HSM plus BSM (Slave10 - Slave15) via the master. For this you must also first calculate the offset and then add this with the address.

$$\begin{aligned} OffsetSlaveX &= (X - 10) * 8 * 500 \\ OffsetSubSlaveY &= (Y - 1) * 500 \\ OffsetSubSlaveX_Y &= 10000 + OffsetSlaveX + OffsetSubSlaveY \end{aligned}$$

You can now also add this offset to the addresses of the respective HSM plus device. The result is the address of the data point with which you can access the respective HSM plus device via the master.

$$AdresseViaMaster = OffsetSubSlaveX_Y + Adresse$$

Example:

You want to retrieve the address "Input Register 34 - HMI State" from slave 1 via slave10 of the master:

$$\begin{aligned} OffsetSlave10 &= (10 - 10) * 8 * 500 = 0 \\ OffsetSubSlave1 &= (1 - 1) * 500 = 0 \\ OffsetSubSlave11_5 &= 10000 + 0 + 0 = 10000 \\ AresseViaMaster &= 10000 + 34 = 10034 \end{aligned}$$

You can access the HMI State of slave station 1 of segment slave station 10 via the head-end station using the address "Input Register 10034".

Example:

You want to retrieve the address "Input Register 34 - HMI State" from substation 8 via the master using Slave12:

$$\begin{aligned} OffsetSlave12 &= (12-10)*8*500 = 8000 \\ OffsetSubSlave8 &= (8-1)*500 \quad = 3500 \\ OffsetSubSlave12_8 &= 10000 \ + 8000 + \ 3500 = 21500 \\ AdresseViaMaster &= 21500 + 34 \quad = 21534 \end{aligned}$$

You can access the HMI State of slave station 8 of segment slave station 12 via the head-end station using the address "Input Register 21534".

3. General information on the message concept of the controller

The controller is designed to display pending messages (indications or faults) to the user. Especially when using the controller with a BMS, it is important to interpret pending messages correctly and to be able to initiate any necessary measures in a targeted manner.

To simplify the display of messages, so-called "HMI statuses" have been added to all subcomponents of the controller. With the help of these HMI statuses, a possible cause of error can be narrowed down and the error weighting identified. Please read the chapter "General information on HMI statuses of the controller" for a detailed explanation of HMI statuses. An HMI status can also be used to identify any boiler malfunctions (see chapter "General information on boiler malfunctions").

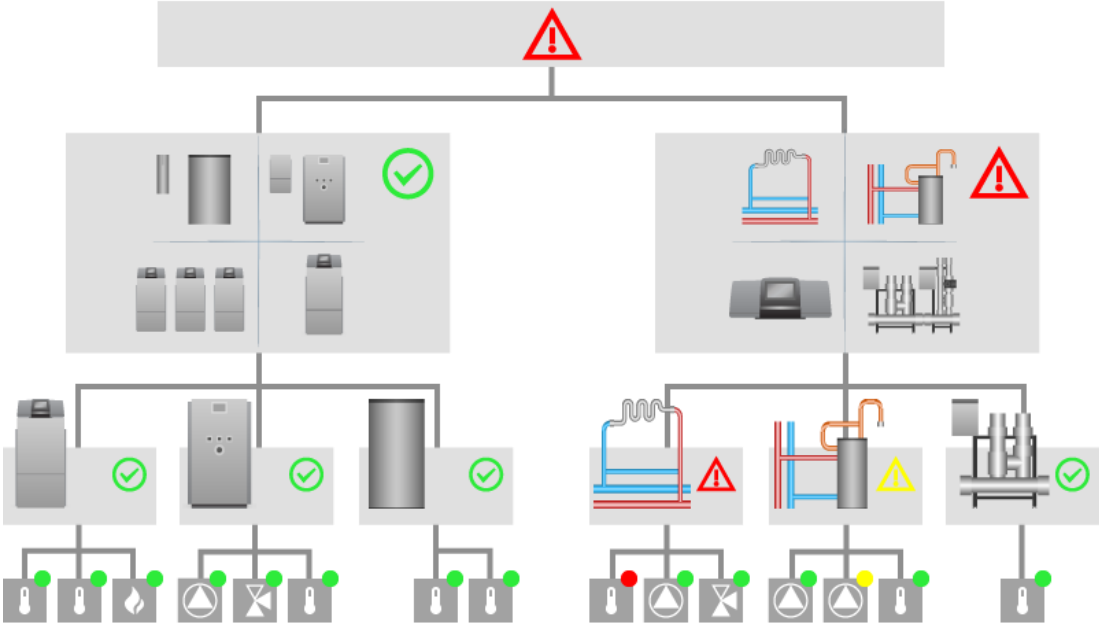


Figure 3-1: Notification Concept of the Controller

Since the HMI statuses can only be used for a rough assessment of the error and often more detailed information is required for the initiation of a suitable measure, there are additional Modbus TCP registers which make the pending message identifiable via a message key. For this please refer to chapter "General information about the evaluation of current messages at the controller".

Note: If the connected boiler is in malfunction or manual mode, the controller will not send any heat requests to the boiler. As soon as the boiler is ready for operation again, the boiler is controlled according to the specifications.

3.1 General information on HMI Status of the control device

An HMI status can be used to determine the current status of the controller and its components. Each software component in the controller has such an HMI status. This shows the current status of the respective component via five possible states. As soon as one of the software components detects an irregularity in control operation, the corresponding HMI status is set. The higher-level software component recognizes the HMI status, adds to it if necessary and also passes it on to its higher-level software component until the respective status has reached the highest software component level. The principle of passing on the HMI status allows the software component that originally influenced the HMI status to be reliably identified and the exact cause of the error to be analyzed.

Five HMI statuses are available within the controller, which use a color assignment:

HMI Status	Colour	Meaning
OK ✓	Green / Blue 🟢	The system / component is working properly.
Warning ⚠	Yellow 🟡	A deviation from the normal state has been detected on the system/component, e.g. manual operation has been activated or the boiler has too high a flue gas temperature.
Critical ⚠	Orange 🟠	The system/component can currently still be operated, but timely intervention is required.
Error ⚠	Red 🔴	The system / component has a malfunction. An intervention is required.
Unknown ⚠ ✓	Gray ⚪	The state of the system / component cannot be determined.

Note: Only the colors green/blue, yellow and red are displayed on the hardware LED on the controller.

3.2 General Information on Boiler Malfunctions

In addition to the HMI status of individual software components of the controller, the heat generator provides additional information on operating messages. The message classes of the boiler influence the HMI status of the corresponding component in the controller. There are three different message classes:

Boiler Fault	HMI Status	Meaning
Maintenance Notice ⚠	Warning (Yellow) 🟡	Signaling of a pending maintenance task.
Blocking Fault ⚠	Warning Yellow 🟡	Fault that leads to a temporary shutdown of a system (e.g. a heating system), which restarts automatically as soon as the malfunction is no longer present.
Locking Malfunction ⚠	Error (Red) 🔴	Malfunction that leads to a shutdown of a system (e.g. a heating system). The fault must be acknowledged at the boiler before the system can resume work.

3.3 General information on the evaluation of current messages on the controller

As of controller version 1.5.X, up to four currently pending messages of the controller can be read out via Modbus TCP/IP. To evaluate and retrieve these messages, the following registers have been added to the data point list:

Error Key X: This information specifies the message key.[1]

The values can be interpreted using the document "Logamatic 5000/Control 8000 series message key list". This document is available in the ".csv" file format. This enables an automated readout or processing of the message keys. You can obtain the document from your Buderus contact partner.

Some messages contain additional information which, however, is only necessary in individual cases. If you need this additional information for a detailed error description, you can contact your Buderus contact person.

---

[1] The X is to be replaced with the numbers 1 to 4

4. General information about Modbus TCP/IP Heart Beat

The communication is monitored via the heart beat by means of a counter. If the counter is not changed by the communication partner (in this case the BMS) within a certain period of time, the control system detects a break in communication. In this case, the error message "Connection to building control system disturbed" is displayed in the menu "Overview of error messages".

In the following two variants are described to realize the Modbus TCP/IP communication:

Variant 1: With Heart Beat

If you operate the "Modbus communication" parameter with the "with heart beat" setting, the BMS must monitor the communication via a counter. For this purpose, the BMS must write to the Modbus register "Heart Beat (in)" of the controller. As soon as this register is written to, the timeout of the heart beat is reset. The timeout can be set via the HMI of the controller.

The current Heart Beat value can be read, validated accordingly and incremented via the "Heart Beat (out)" register. The "Heart Beat (out)" register always reflects the current value from the "Heart Beat (in)" register. The validation can therefore check whether the numerical value sent to the "Heart Beat (in)" register has been adopted accordingly in the "Heart Beat (out)" register and use the value to increment the counter. The value in the "Heart Beat (in)" register should be reset in time before the overflow.

Variant 2: Without Heart Beat

If you do not have the possibility to implement the heartbeat on the BMS side, you can also use the "no heartbeat" setting. In this case, the controller assumes that the connection to the BMS is continuous. In this case, the detection of a communication interruption is not available.

5. General information about the conversion of the Modbus address scheme

If you want to use the Modbus Modicon address scheme instead of the decimal address scheme, you may use the following formula to convert the addresses:

$$ModiconAdress = RegisterTyp + DecimalAdress + 1$$

For register type "Holding register" take the number 40000. For register type "In-put register" take the number 30000.

Example:

For example, you want to address the data point holding register with decimal address 400 "Setpoint system supply temperature in °C" in the Modicon schema:

1. You determine the value of the register type:

$$RegisterTyp = \text{Holding} - \text{Register} \ = \ 40000$$

2. You calculate the address for the Modicon scheme:

$$ModiconAdress = \ 40000 \ + \ 400 \ + \ 1 \ = \ 40401$$

So you can use the address 40401 if you want to use the Modicon scheme to describe the data point "Setpoint system supply temperature in °C.

Note: The decision whether to use the decimal or modicon scheme depends on the software and configuration of the BMS used. Please contact the manufacturer of your BMS for more information on the correct scheme.

6. Additional information for setting or reading single bits in a Modbus Register

If you do not have the possibility to set single bits in a register, you may alternatively enter an integer corresponding to the desired bit pattern into the Modbus register. You can also read out a complete register as an integer and interpret it as a bit pattern.

A Modbus register (2 bytes) consists of 16 bits (Bit_0 to Bit_{15}). Big-endian (Most Significant Bit First (MSB)) is used as byte order. If you now want to set individual bits in a Modbus register, you can determine the required integer (≙ bit pattern) using the following table:

Bit Number	**Bit_{15}**	**Bit_x …**	**Bit_7**	**Bit_6**	**Bit_5**	**Bit_4**	**Bit_3**	**Bit_2**	**Bit_1**	**Bit_0**
Valence	32768	…	128	64	32	16	8	4	2	1
Bit to be set										

Note: For reasons of presentation, the values of all bits have not been specified. The significance of a bit can be determined using the following formula:

$$Valency = 2^{Bitnumber}$$

Example for calculating the Valence

You want to determine the value of Bit_{15}:

$$Valency\ Bit_{15} = 2^{15} = 32768$$

Note:

- The following examples use the Modbus data point list from version 1.3.X. However, the described procedure can be transferred to any Modbus register and thus also to the old data point list.
- The conversion of integers to bit patterns and vice versa can also be calculated using a calculator with programming mode.
- The setting and reading of bits can also be done via corresponding bit operations.
- Numbers with a b are specified in binary format and numbers with a d in decimal format.

6.1 Setting individual bits in the Modbus Register

Sie möchten die temperaturgeführte Wärmeanforderung über Modbus TCP/IP verwenden. Hierfür müssen Sie das Bit_2 im Holding-Register 405 aktivieren. Die Tabelle würde in diesem Fall wie folgt aussehen:

Bit Number	**Bit$_7$**	**Bit$_6$**	**Bit$_5$**	**Bit$_4$**	**Bit$_3$**	**Bit$_2$**	**Bit$_1$**	**Bit$_0$**
Valence	128	64	32	16	8	4	2	1
Bit to be set						X		

Calculated integer or bit pattern:

$$Integer = Valence\ Bit_2 = 4$$
$$Bit\ Pattern = 0000000000000100_b$$

If, on the other hand, you would like to set several bits (e.g. temperature-controlled [Bit2], priority [Bit1] and guide tank [Bit0]), the table and corresponding calculation would look like this:

Bit Number	**Bit$_7$**	**Bit$_6$**	**Bit$_5$**	**Bit$_4$**	**Bit$_3$**	**Bit$_2$**	**Bit$_1$**	**Bit$_0$**
Valence	128	64	32	16	8	4	2	1
Bit to be set						X	X	X

Calculated Integer or Bit Pattern:

$$Integer = Valence\ Bit_2 + Valence\ Bit_1 + Valence\ Bit_0 = 4 + 2 + 1 = 7$$
$$Bit\ Pattern = 0000000000000111_b$$

6.2 Reading single bits from Modbus registers

If you want to read individual bits from a Modbus register, e.g. to read out the current errors on the safety chain, proceed as follows:

First, read out the register contents as an integer. The read integer must then be transferred to a bit pattern. To transfer the integer to a bit pattern, please proceed as follows:

Now divide the read integer by the valence, starting with the highest bit (division with remainder). If the quotient is equal to zero, you can divide the integer with the significance of the next smaller bit. As soon as the quotient is not equal to zero, the current bit is set and you can use the remainder of the divisi-on as a dividend for the calculation with the significance of the next smaller bit. You apply this scheme until you have divided the integer or the remainder by all bits. The result is the corresponding bit pattern.

For this you can use the following formula (division with remainder):

$$Bit_{NumberX} = \frac{Integer\ or\ Remainder}{Valence\ Bit_{NumberX}}$$

Example:

If, for example, you want to determine the current errors of the safety chain, you would use input register 225 for this. Assuming you read the integer 35 from the register, the division with remainder for this example would look like this:

Bit_{15} = 35 / Valence Bit_{15} = 35 / 32768 = 0

Bit_{14} = …

… = …

Bit_{5} = 35 / Valence Bit_{5} = 35 / 32 = 1 Remainder 3

Bit_{4} = 3 / Valence Bit_{4} = 3 / 16 = 0

Bit_{3} = 3 / Valence Bit_{3} = 3 / 8 = 0

Bit_{2} = 3 / Valence Bit_{2} = 3 / 4 = 0

Bit_{1} = 3 / Valence Bit_{1} = 3 / 2 = 1 Rest 1

Bit_{0} = 1 / Valence Bit_{0} = 1 / 1 = 1

The result of the integer 35 is the following bit pattern: 0000000000100011_b

7. Additional information about converting values to ASCII characters

If, for example, you want to convert the fault code (= display code, input register 60) into ASCII characters, please proceed as follows:

1. Read the integer value from register 60.
2. Convert the integer into the binary format as described in chapter "Supplementary information for setting or reading single bits in a Modbus controller".
3. Now split the binary format into the single bytes (= 8 bit). Each byte corresponds to one letter. 4.
4. Now convert each byte back into an integer.
5. Look in an ASCII-Table[2] which letter is represented by the respective integer

Example:

You read the integer 12880 from the input register register 60 (fault code / display code).

1. Convert the integer to binary format:
   0011001001010000_b
2. Now divide the binary number into bytes:
   00110010_b , 01010000_b
3. Now convert the single bytes to decimal:
   $00110010_b = 50_d$

   $01010000_b = 80_d$
4. According to the ASCII table, the numbers can now be converted into characters:

   50_d = „2“

   80_d = „P“

If you read a 12880 from input register 60, this corresponds to the error code "2P". If you also read a 564 from input register 59 (additional code), for example, this corresponds to the error message "Temperature rise boiler sensor too fast (>70K/min)".

Note: In addition to faults, operation codes are also displayed in the "Fault codes" register (input register 60).

---

[2] https://de.wikipedia.org/wiki/American_Standard_Code_for_Information_Interchange#ASCII-Tabelle

8. Outdated software versions and their peculiarities

The control unit will be developed further successively. Functions are added or changed in order to provide more safety, comfort and extended possibilities in future versions. Therefore, it is always recommended to update your controller to the latest available version (see chapter "Checking the current controller firmware").

If, however, an older software version is available on your controller, you will find information in the following chapters about the special features to be observed.

8.1 Control of the controller via Modbus TCP/IP up to and including version 1.2.7.X

Please use the following control of the controller up to software version "1.2.7.X".

Register type	Address	Data point Name	Data Type	Value Range	Meaning
Holding Register	0	Heart Beat (in)	int	0 to 65535	If the Modbus connection is to be monitored for connection interruptions or timeouts (Modbus communication with Heart Beat), the counter must be written to this register, which is not equal to the old value that can be read via the Heart Beat (out) register.
Holding Register	1	Heart Beat (out)	int	0 to 65535	If this register is read, the current counter value of the heartbeat (in) can be retrieved (for verification of the transmitted counter value on the BMS side).
Holding Register	35	Operating mode	int	0 = Off, 1 = Keep Warm **2 = operation**	Operating mode for boiler request. 0 = Off = Boiler is off, consumers are still controlled. 1 = Keep warm = Keep warm function for industrial cascade applications with separate keep warm pump (currently not relevant for Logamatic 5000). 2 = Operation = normal operation via heat request of the boiler.

Register type	Address	Data point Name	Data Type	Value Range	Meaning
Holding Register	2	System temperature Temperature specification in °C	int	0 to 100 (the maximum temperature depends on the System)	Setpoint of the system supply temperature. This setpoint only applies explicitly if the "Heat request only via Modbus" mode is deactivated.
Holding Register	36	Flow temperature set point in °C	int	0 to 100 (the maximum temperature depends on the type of boiler)	Setpoint of the current boiler flow temperature. This setpoint is valid as an exclusive setpoint if the "Heat request only via Modbus" mode is enabled.
Holding Register	37	Set point power in %	int	0 to 100	Set point of the current boiler output.

Note:

- It is recommended to perform a firmware update of the controller and to use the control from version 1.3.X (see chapter "Control of the controller via Modbus TCP/IP").
- If you perform a firmware update on the controller you are already using, you can still use the subsequent control via Modbus TCP/IP by activating the use of old Modbus addresses (see chapter "Necessary settings at the controller").
- Data type Integer (int) corresponds to 16 bits (Bit0 to Bit15), see chapter "Data types and register width".
- The addresses specified here refer to the fact that the controller is configured as a master. If you use the controller as slave, please refer to chapter "Control via Modbus TCP/IP in the controller network".Temperaturgeführte Wärmeanforderung über Modbus TCP/IP bis einschließlich Version 1.2.7.X

To be able to send a heat request via Modbus, you must first configure the controller accordingly.

Depending on which mode you have selected for the "Heat request only via Modbus" parameter, you must send a different address for the flow temperature setpoint value.

Note: To make a heat request via Modbus The "System temperature temperature specification" register is only relevant if the "Heat request only via Modbus" mode is deactivated. If the "Heat request only via Modbus" mode is activated, the "Setpoint flow temperature" register must be used for the temperature specification.

Variant "Heat request only via Modbus" deactivated:

Register	Value of Register
System temperature Temperature specification (Holding Register 2)	X in °C

Note: X in °C corresponds to your specification.

Variant "Heat request only via Modbus" activated:

Register	Value of Register
Operating Mode (Holding Register 35)	2
Set point power (Holding Register 37)	100 in %
Set Point – flow temperature (Holding Register 36)	X in °C

Note: X in °C corresponds to your specification.

8.1.1 Power controlled heat request via Modbus TCP/IP up to and including version 1.2.7.X

To be able to send a power request via Modbus, you must first configure the controller accordingly. Then you can transmit the following values to the following Modbus registers:

Register	Value of Register
Operating Mode (Holding Register 35)	2
Set point power (Holding Register 37)	X in %
Set point – flow temperature (Holding Register 36)	100 in °C

Note: X in °C corresponds to your specification.

8.1.2 Information on parameter 'Heat request only via Modbus

If you have connected additional function modules (e.g. for a heating circuit) or substations to the controller, these may generate a heat request. The parameter "Heat request only via Modbus" can be used to define the handling of heat requests in connection with Modbus TCP/IP.

If you have deactivated the parameter "Heat request only via Modbus", the connected function modules / substations can send a heat request. A heat request via Modbus, which comes from the BMS, can only add a heat request. Existing heat requests from other modules are not influenced by this. The highest heat request is always used as the setpoint.

If the parameter "Heat request only via Modbus" is activated, additionally connected modules / substations cannot send a heat request. With this setting, heat requests are only carried out by the BMS. The heat request is thus exclusively available to the BMS, which has full control of all heat requests.

A power request, unlike a heat request, is always available exclusively to the BMS. Therefore, you cannot perform a power-controlled heat request if you have deactivated the parameter "Heat request only via Modbus".