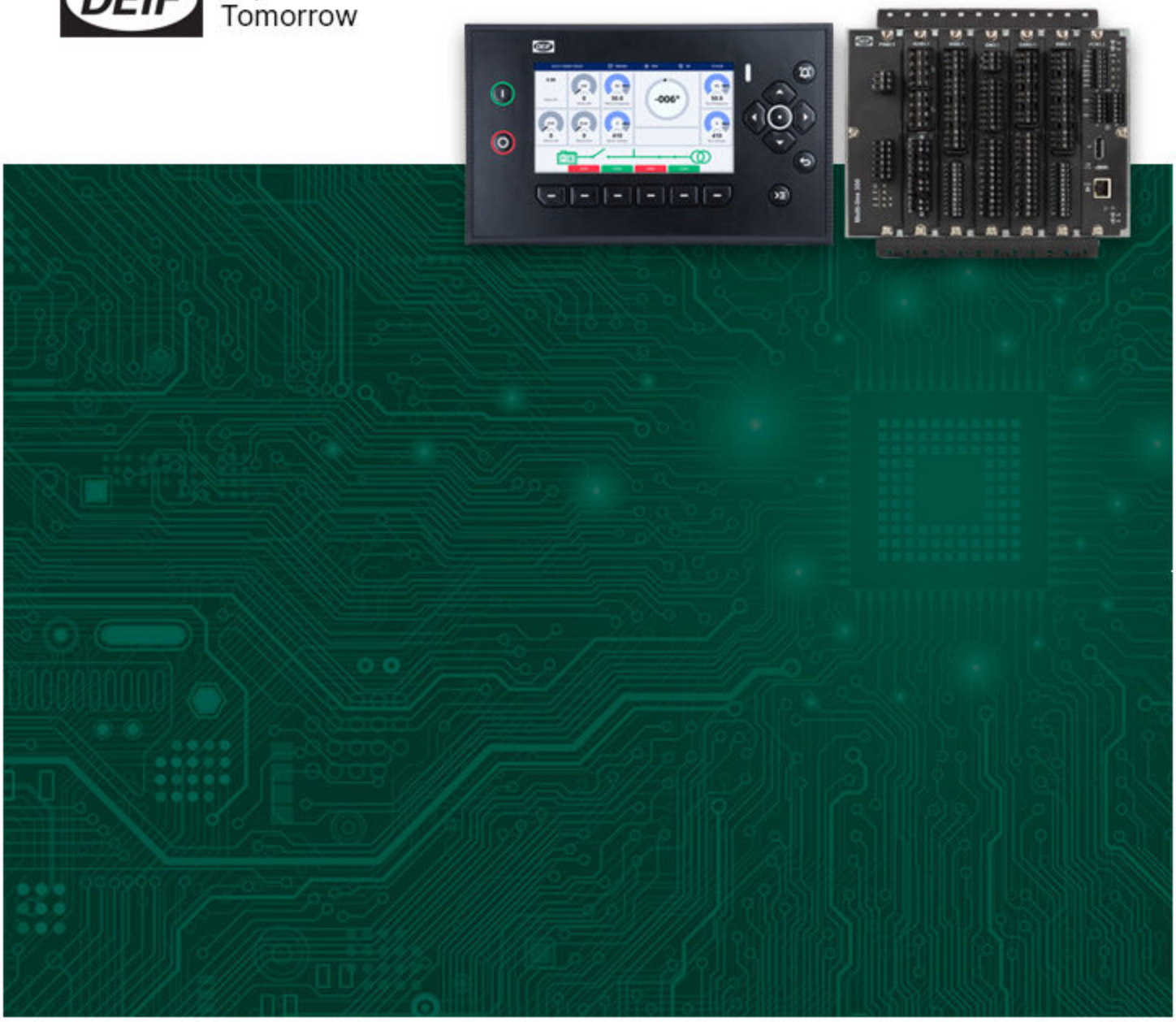
Omni

4189341464-A

iE 250 | iE 250 Marine | iE 350 | iE 350 Marine

Modbus server manual

DEIF Improve Tomorrow



1. About this manual

1.1 Intended users of the Modbus server manual

The Modbus server manual is intended for designers, integrators, and developers who want integrate an alternative interface to control the application or system.

1.2 Applicable products and software

The information in this document relates to these products and software versions:

	Version			
Product	**iE 250**	**iE 250 Marine**	**iE 350**	**iE 350 Marine**
Application software	2.0.17.x or later	2.0.17.x or later	2.0.17.x or later	2.0.17.x or later
PICUS	1.0.26.x or later	1.0.26.x or later	1.0.26.x or later	1.0.26.x or later

More information
See www.deif.com for the latest approved certifications on your product.

1.3 Legal information

Third party equipment

DEIF takes no responsibility for installation or operation of any third party equipment. In no event shall DEIF be liable for any loss of profits, revenues, indirect, special, incidental, consequential, or other similar damages arising out of or in connection with any incorrect installation or operation of any third party equipment.

Warnings

DANGER!

Access to controller settings with Modbus TCP

All controller settings can be accessed and modified through Modbus TCP.

This includes disabling critical controller protections by changing settings and alarms. Use the Modbus tables provided by DEIF to ensure that you do not disable critical protections.

NOTICE

Modbus and Emulation

Modbus control remains active even during Emulation mode.

If Modbus is allowed to control sources, these will continue to be controlled even if the controller is in Emulation mode.

Trademarks

DEIF and the DEIF logo are trademarks of DEIF A/S.
Bonjour® is a registered trademark of Apple Inc. in the United States and other countries.
*Adobe®, Acrobat®, and Reader®*are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States and/or other countries.

CANopen® is a registered community trademark of CAN in Automation e.V. (CiA).
SAE J1939® is a registered trademark of SAE International®.
CODESYS® is a trademark of CODESYS GmbH.
EtherCAT®, *EtherCAT P®*, *Safety over EtherCAT®*, are trademarks or registered trademarks, licensed by Beckhoff Automation GmbH, Germany.
VESA® and DisplayPort® are registered trademarks of Video Electronics Standards Association (VESA®) in the United States and other countries.
Google® and Google Chrome® are registered trademarks of Google LLC.
Linux® is a registered trademark of Linus Torvalds in the U.S. and other countries.
Modbus® is a registered trademark of Schneider Automation Inc.
Torx®, *Torx Plus®* are trademarks or registered trademarks of Acument Intellectual Properties, LLC in the United States or other countries.
Windows® is a registered trademark of Microsoft Corporation in the United States and other countries.

All trademarks are the properties of their respective owners.

Disclaimer

DEIF A/S reserves the right to change any of the contents of this document without prior notice.

The English version of this document always contains the most recent and up-to-date information about the product. DEIF does not take responsibility for the accuracy of translations, and translations might not be updated at the same time as the English document. If there is a discrepancy, the English version prevails.

Copyright

2. About Modbus

2.1 How it works

Modbus is a standard communication protocol between intelligent industrial devices. This is a standard method to represent and communicate data in intelligent industrial devices.

The controller includes a built-in Modbus TCP/IP server. The Modbus TCP/IP server allows external devices to communicate with the controller using the Modbus TCP/IP communication protocol.

For example:

- A PLC can request that specific data is read from the controller, such as the settings for the nominal AC configuration.
- A PLC can send commands to the controller using the Modbus TCP/IP protocol.

This document only describes the information required to communicate with the controller using the Modbus TCP/IP protocol. For more information about Modbus in general and the Modbus TCP/IP protocol, refer to the documentation freely available at http://www.modbus.org.

2.2 Modbus tables

About the Modbus tables

The Modbus tables are stored in a Microsoft® Excel file that contains worksheets with Modbus data.

Worksheet name	Description
Descriptions	An overview of the other four worksheets. The information includes a description of each function group listed in the tables worksheets.
Discrete output coil	You can read or write information to the addresses that are listed in this worksheet. Use Modbus function code 01 to read whether a coil is on or off. Use Modbus function code 05 or 15 to toggle the coil value. Read-only addresses will return a 0 value if you try to write to them.
Discrete input contact	You can only read information from the addresses that are listed in this worksheet. Use Modbus function code 02 to read whether the contact is on or off.
Output holding register	You can read or write information to the addresses that are listed in this worksheet. Use Modbus function code 03 to read the information stored at the requested Modbus address(es). Use Modbus function code 06 or 16 to write information to the Modbus address(es). Read-only addresses will return a 0 value if you try to write to them.
Input register	You can only read information from the addresses that are listed in this worksheet. Use Modbus function code 04 to read the information stored at the requested Modbus address(es).
Controller text	An overview of texts associated to Modbus output values. This association is only available for selected Modbus addresses.

It is possible to map digital and analogue inputs/outputs to Modbus addresses.

NOTE All values shown are decimal values, unless specifically stated that a value is hexadecimal.

Download Modbus tables

iE 250

XLSX **iE 250** Core

www.deif.com/rtd/ie250/modbus

XLSX **iE 250** Premium

www.deif.com/rtd/ie250pm/modbus

iE 250 Marine

XLSX **iE 250 Marine** Core

www.deif.com/rtd/ie250marine/modbus

XLSX **iE 250 Marine** Power management

www.deif.com/rtd/ie250marinepm/modbus

iE 350

XLSX **iE 350** Core

www.deif.com/rtd/ie350/modbus

iE 350 Marine

XLSX **iE 350 Marine** Core

www.deif.com/rtd/ie350marine/modbus

XLSX **iE 350 Marine** Power management

www.deif.com/rtd/ie350marinepm/modbus

Modbus texts

XLSX **iE 250** **iE 350** Modbus texts

www.deif.com/rtd/omni/mt

2.3 Modbus communication

Modbus TCP/IP

To communicate to the controller with Modbus TCP, the following conditions must be met.

- The device interfacing with the controller must be connected to one of the following:
  - Any Ethernet connection on the controller.
  - Another controller in the DEIF network.
- The controller must have an IPv4 address.
- Modbus TCP communication software must be installed on the device communicating with the controller.

More information
See the iE 250 Installation instructions or iE 350 Installation instructions for how to wire the Ethernet connection to the controller.

3. Modbus in the controller

3.1 Modbus TCP protocol

The controller uses the Modbus TCP protocol to communicate with an external device over the Modbus network and through the internet. The communication protocol uses static IPv4 addresses to send information. Dynamic IPv4 addresses (created by a dynamic host configuration protocol server (DHCP server)) and IPv6 addresses are not supported by the controller for Modbus communication purposes.

More information
See the Operator's manual or the PICUS manual for how to configure the controller network settings.

3.2 Modbus communication port

By default the controller uses port 502 (standard for Modbus TCP protocol) for TCP communication. Create a custom Modbus server to use a different communication port.

Each controller can process up to 10 communication requests at a single time.

3.3 Controller identifier

The Modbus TCP protocol will always use the controller IPv4 address to identify the controller that the client wants to communicate with. However, some Modbus communication tools will still require/automatically add a Modbus server ID, also known as a unit identifier, for the unit that the server is communicating with. For these cases the controller accepts Modbus server IDs from 1 to 247. This is the case for all controllers in the network that communicate using the Modbus TCP protocol.

If two Modbus servers are enabled at the same time that use the same communication port, then a unique Modbus Server ID must be configured for each server.

Specific controller identifiers can be selected for the controller when you configure a custom server.

3.4 Data handling

NOTICE

Check Modbus protocol address information

Check the Modbus protocol address information using PICUS to ensure that you are referencing the correct Modbus address for the function that you are executing.

NOTE Always document and store changes that you make to the way that the controller interprets Modbus data.

Data format (endian)

To ensure that the correct data is retrieved from the controller, the request from the Modbus client must match the data format of the selected address. The data format is configured in the Modbus server, and are applied to the *Holding register* and the *Input register*.

Sign

In general, the integer data (16-bit and 32-bit) that is accessed from the controller through Modbus TCP are signed integer values.

Conversion

Data in the *Holding register* and *Input register* of the Modbus table is converted according to the conversion template selected for that address. When data is read using Modbus, then the *Formula* is used to convert the Modbus data. When data is written using Modbus, then the *Reverse formula* is used to convert the data into a form that can be stored in the Modbus protocol.

Conversions can also be used to force unit conversions on specific addresses.
NOTE *Reverse formulas* are NOT automatically determined.

Modbus data conversion example

The parameter nominal power factor is assigned to an unused address in a custom Modbus protocol. The controller can process inputs to the forth decimal value (for example, 0.8002) for the nominal power factor. To read and write values correctly using Modbus a conversion template *X * 10000* is assigned to the address. The *Formula* equal to *x*10000* and a *Reverse formula* equal to *x*0.0001*.

This means that when a value of 0.8002 is read from the controller, the displayed value is:
Result = *Formula* ⇒ Result = x * 10000 ⇒ Result = 0.8002 * 10000 ⇒ Result = 8002

To write a value of 0.85 to the controller using Modbus, the value that should be written to the controller is:
Result = *Reverse formula* ⇒ Result = x * 0.0001 ⇒ 0.85 = x * 0.0001 ⇒ x = 8500

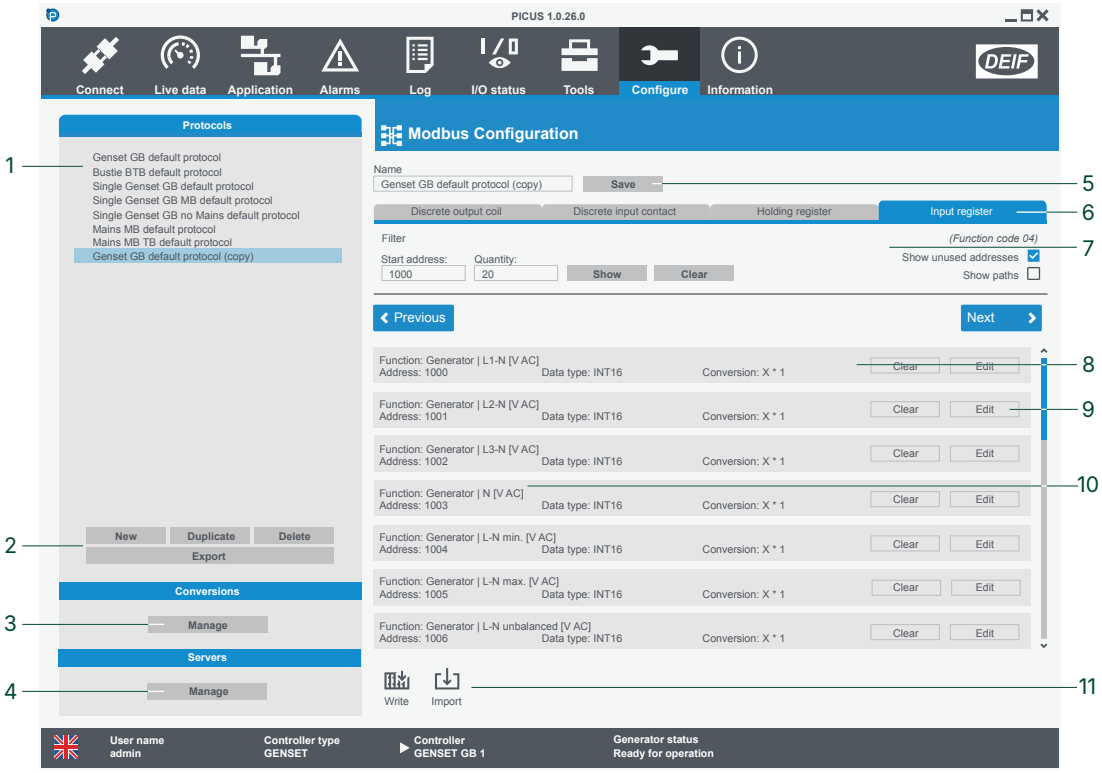
Refresh rate

Data stored in the Modbus addresses is refreshed at the following maximum rates:

Data	Maximum refresh rate	Function group example
AC measurements	20 ms	[A-side] AC measurements
Values	40 ms	Alarm parameter: Enable

3.5 Configure Modbus with PICUS

3.5.1 Protocols page



No.	Item	Notes	
1	Protocol list	Shows the protocols on the controller.	
2	Commands	**New** protocol.	**Duplicate** the selected protocol.
		Delete the selected protocol.	**Export** the protocol.
3	Conversions page	Change to the Modbus conversion page.	
4	Servers page	Change to the Modbus servers page.	
5	Protocol name	Name of Modbus protocol.	
6	Supported Modbus functions	**Discrete output coil**: Read and write addresses in binary data.	**Discrete input contact**: Read only addresses in binary data.
		Holding register: Read and write addresses in boolean, 16 and 32-bit integer, float or bit map data.	**Input register**: Read only addresses in boolean, 16 and 32 bit integer, float or bit map data.
7	Address filter	Filter to display up to 1000 consecutive addresses for a Modbus function.	
8	Modbus address details	**Unused address**: A function can be assigned if a duplicated protocol.	**Reserved address**: Function assigned is not configurable. The function cannot be restored if it is cleared.
		Function: Controller path of the function assigned.	**Address**: Modbus address of the function.
		Data type: The data type associated. *	**Conversion**: Scaling or conversion associated. *
9	Address configuration commands (Default protocols cannot be edited)	**Set**: function to an unused address.	**Edit**: function assigned to the selected address.
		Clear: function assigned to the selected address.	

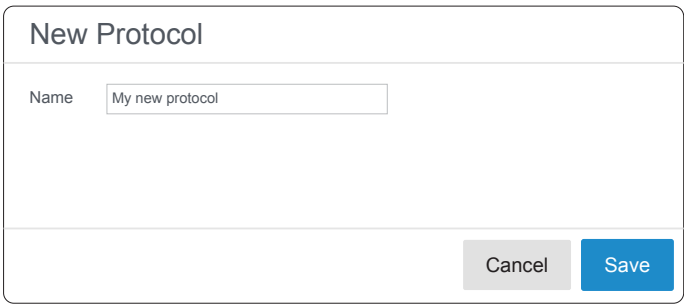
No.	Item	Notes	
10	Function path	Full function path displayed by default.	
		Collapse: the function name.	**...**: expand the function path.
11	Modbus function commands	**Write** changes to the selected function to the controller.	**Import** a Modbus function to replace the selected function.

NOTE * Only available in the Holding and Input registers. Scaling is not available for binary values.

Create, edit, or export a protocol

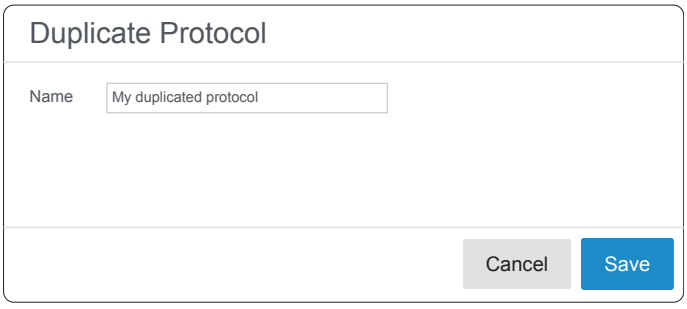
The controller default protocol cannot be edited or removed.

Create a new protocol

1. Select **New**.
2. Enter a name:
   - 

3. Select **Save**.
4. Select the new protocol to access the Modbus functions.
5. Select a Modbus function to configure.
6. Configure Modbus addresses individually with the filter and **Set** address configuration command, or import an existing Modbus function.

Duplicate an existing protocol

1. Select a Modbus protocol to duplicate.
2. Select **Duplicate**.
3. Enter a name:
   - 

4. Select **Save**.
5. Select the new protocol to access the Modbus functions.
6. Select a Modbus function to configure.
7. Configure Modbus addresses individually with the filter and **Set** address configuration command, or import an existing Modbus function.

Edit a protocol

Edit a used address

1. Select the protocol to configure from the protocol list.

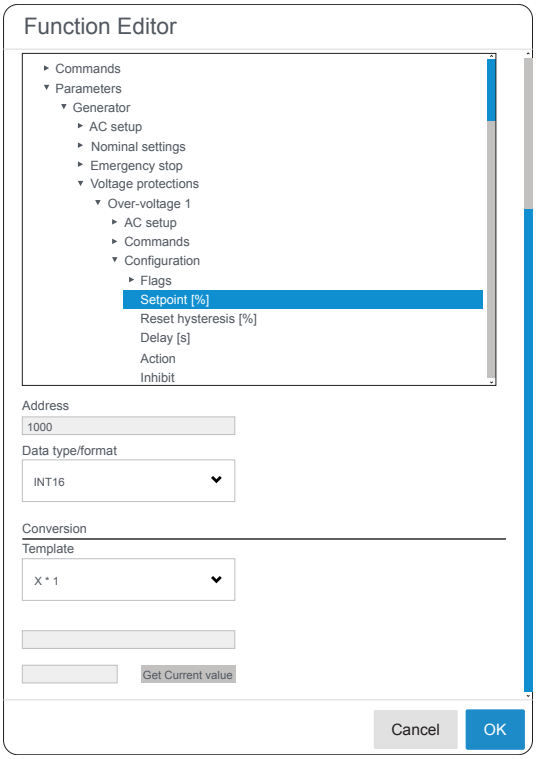
2. Select the Modbus function to configure.
3. Use the filter to select the address range to configure.
   - Type in the start address and the number of addresses (including the Start address) to read from the controller.
   - If Show Unused Addresses is **not enabled**, then only configured addresses are shown.
   - The amount of addresses shown can be less then the value entered in Quantity.
4. Select **Edit** to configure the selected address.
5. Select **Write** to write the changes to the controller.

Clear a used address

1. Select the protocol to configure from the protocol list.
2. Select the Modbus function to configure.
3. Use the filter to select the address range to configure.
   - Type in the start address and the number of addresses (including the Start address) to read from the controller.
   - If Show Unused Addresses is **not enabled**, then only configured addresses are shown.
   - The amount of addresses shown can be less then the value entered in Quantity.
4. Select **Clear** to remove the function associated to the address.
5. Select **Write** to write the changes to the controller.

Set a function to an unused address

1. Select the protocol to configure from the protocol list.
2. Select the Modbus function to configure.
3. Use the filter to select the address range to configure.
   - Type in the start address and the number of addresses (including the Start address) to read from the controller.
   - Show Unused Addresses must be **enabled** to see empty addresses.
4. Select **Set** to open the Function Editor.
5. Select the function to associate to the Modbus address:



   - Functions that don't match the Data type/format for the address cannot be selected.
   - The Data type/format can be selected for register addresses.
   - A conversion formula must be selected for register addresses.
   - Test the selected conversion with **Get Current value**.

6. Select **OK**.
7. Select **Write** to write the changes to the controller.

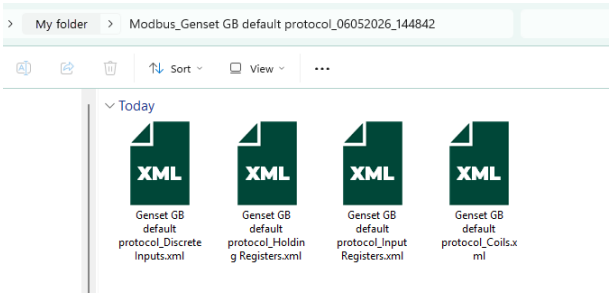
Import a protocol

If you import a function it overwrites existing data without a warning notification.

1. Select the protocol to import.
   - The controller only accepts Modbus functions that use the correct xml-format.
   - Only custom protocols or copies of default protocols can be imported.
2. Select the Modbus function to import data to.
3. Select **Import** .
4. Select the file to import and select **Open**.
5. Select **Dismiss** to close the confirmation window when the import is complete.

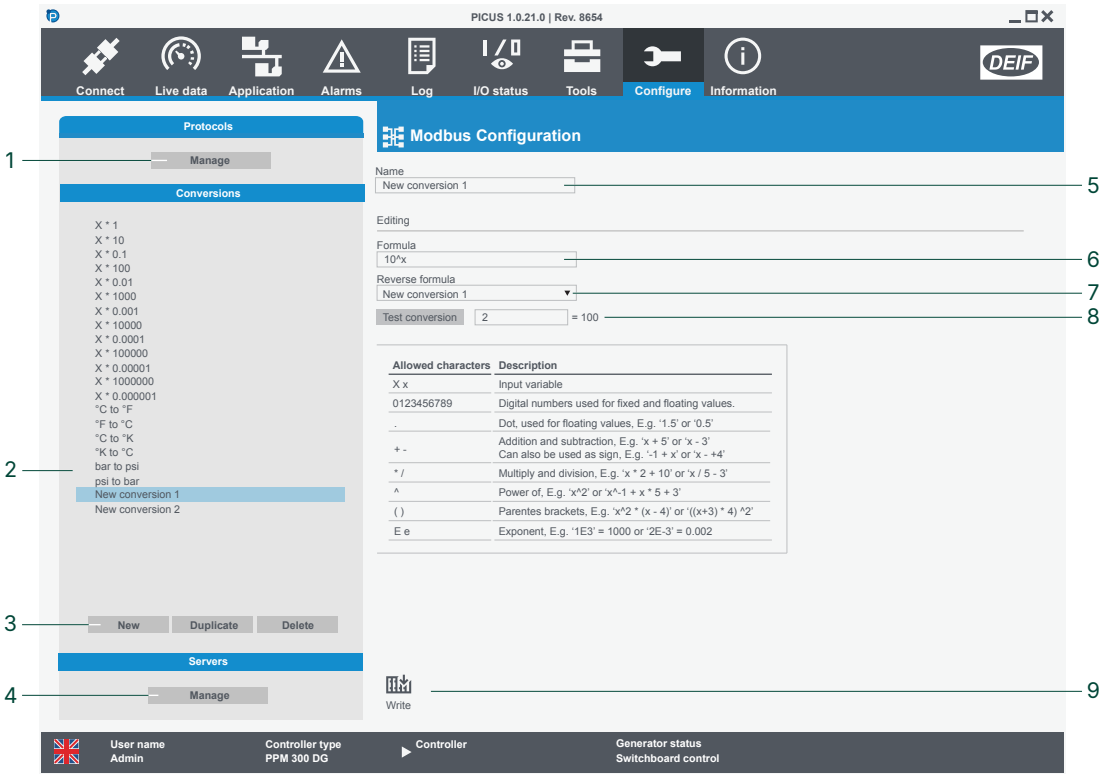
Export a protocol

Exported protocols are saved as four xml files (one for each function).

1. Select the protocol to export from the protocol list.
2. Select **Export** to open the location selection window.
3. Select a location to store the Modbus functions.
4. Select **Select folder**.
5. The protocol is exported to the folder you selected.
   - Example: *
   - 

NOTE * The XML files are named for your product, the above example is for iE 250.

3.5.2 Conversions page

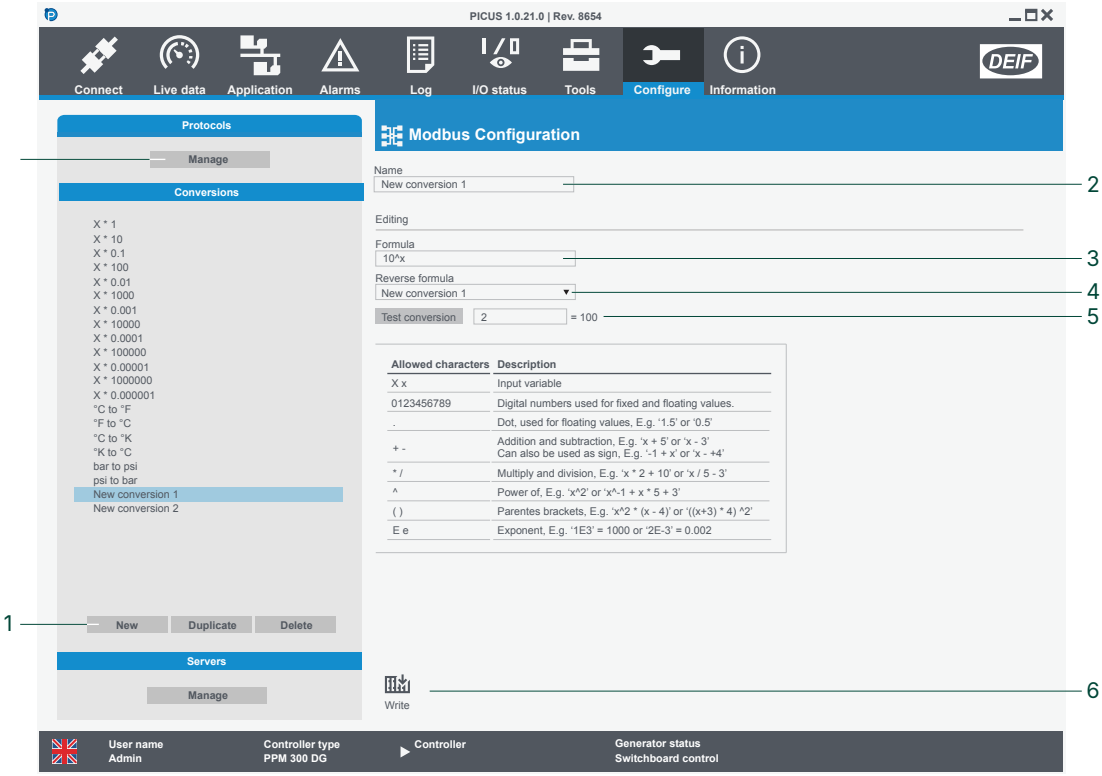


No.	Item	Notes
1	Protocol page	Change to the Modbus protocols page.
2	Conversions list *	Shows the conversions (scaling and unit) on the controller.
3	Commands	**New** conversion. **Duplicate** the selected conversion.
		Delete the selected conversion.
4	Servers page	Change to the Modbus servers page.
5	Conversion label	Name of a custom conversion.
6	Formula **	The conversion formula applied when you read a Modbus address.
7	Reverse formula	Conversion formula applied when you write a value to a Modbus address. The Reverse formula is always selected from the existing conversions.
8	Conversion test	Select a value for x to test the result of the Formula.
9	Modbus function commands	**Write** the conversion to the controller.

NOTE * The controller default conversions cannot be edited or removed.
** The Formula is a function of x, where x represents the raw value of the Modbus address.

Create or edit a conversion

Create a new conversion



1. Select **New**.
2. Enter a name for the conversion.
3. Type the formula for the conversion as a function of x.
   - The Formula is the conversion used when you read the data.
   - "x" is the value read by the controller for the function assigned to the address.
4. Select the Reverse formula from the list of existing formulae.
   - The Reverse formula is the conversion used when you write the data.
   - If the Reverse formula is not available, then a new conversion must be created where the Formula contains the desired Reverse formula.
5. Optional: Type a number in the Test conversion field and select **Test conversion** to see an example of the result of your new conversion (Formula).
6. Select **Write** to write the changes to the controller.

If there is an error with the Formula or Reverse formula, then the conversion defaults to x*1 for both the Formula and Reverse formula.

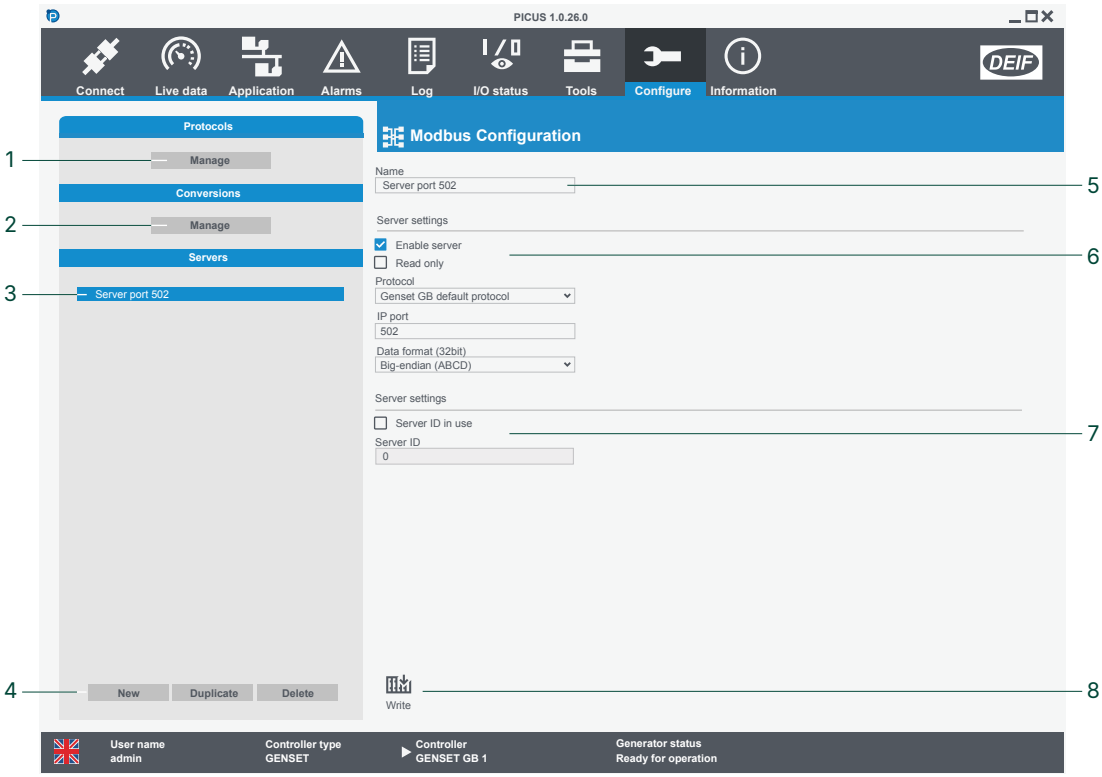
Duplicate a conversion

1. Select the conversion to duplicate and select **Duplicate**.
2. Optional: Enter a new name.
3. Select **Write** to write the changes to the controller.

Edit a conversion

1. Select the conversion to edit.
   - Default conversions cannot be edited.
2. Make the desired changes.
3. Select **Write** to write the changes to the controller.

3.5.3 Servers page

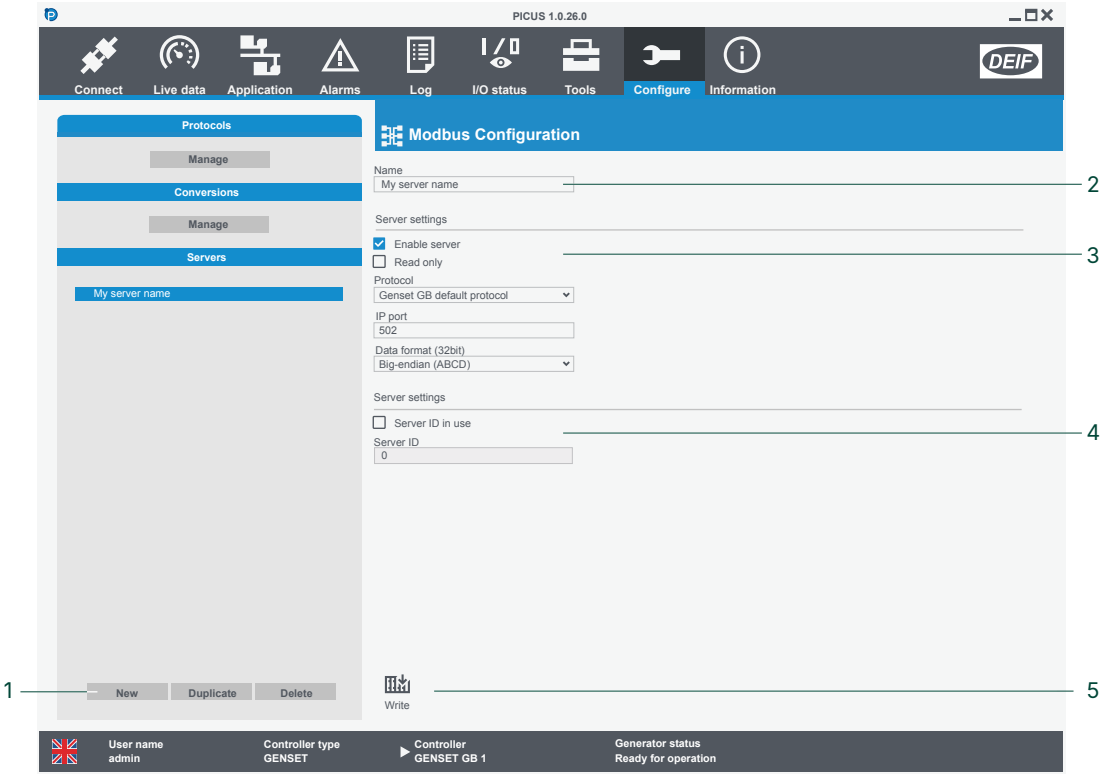


No.	Item	Notes	
1	Protocol page	Change to the Modbus protocols page.	
2	Conversions page	Change to the Modbus conversions page.	
3	Server list	Shows the servers on the controller.	
4	Commands	**New** server.	**Duplicate** the selected server.
		Delete the selected server.	
5	Server name	Name of selected server.	
6	Server settings	**Enable server**: Enable the selected server as active on the controller.	**Read only**: Enable all of the Modbus addresses as read only addresses and function codes 05, 06, 15 and 16 do not respond.
		Protocol: Select the Modbus protocol that is associated with the server.	**IP port**: The communication port for the server. *
		Data format (32bit): Byte order of the data sent with Modbus.	
7	Server settings	**Server ID in use**: Enabled the server uses the specified Server ID. If multiple servers are enabled and use the same IP port, then this parameter must be enabled.	
		Server ID: The unique Server ID associated with the Modbus server. If Server ID in use is not enabled, then the Server ID is 0.	
9	Server commands	**Write** the server to the controller.	

NOTE * The default Modbus port is port 502. If multiple servers are active and use the same port, then each server must have a unique Server ID.

Create or edit a server

Create a new server



1. Select **New**.
2. Enter a name for the server.
3. Configure the Server settings section:
   - **Enable server**: Activate or deactivate the server.
   - **Read only**: If **Enabled** then all of the Modbus addresses are read-only addresses.
   - **Protocol**: The Modbus protocol used on the server. Select from a list of existing protocols.
   - **IP port**: The communication port for Modbus communication. If more than one active server uses the same IP port, a Slave ID must be configured for all servers.
   - **Data format (32bit)**: Select the data format for 32-bit addresses (32-bit integer, float).
4. Optional: Configure the Slave settings section.
   - **Slave ID in use**: Only **Enable** this if you have multiple enabled servers that use the same communication port.
   - **Slave ID**: Select the ID number for the slave unit. ID number must be unique for every server that use the same communication port.
5. Select **Write** to write the changes to the controller.

Duplicate a server

1. Select the server to duplicate.
2. Select **Duplicate**.
3. Optional: Enter a new name.
4. Select **Write** to write the changes to the controller.

Edit a server

1. Select the server to edit.
2. Configure the settings.
3. Select **Write** to write the changes to the controller.

3.6 Modbus controller texts

Certain Modbus addresses provide status information and other textual data from the controller. Use the returned value is a reference to look up the corresponding text in the Modbus texts file.

Modbus tables

Modbus address (input register)	PLC address	Function group
13900	313901	Controller status text
13950	313951	Popup text
13952	313953	Popup text
13954	313955	Popup text
13956	313957	Popup text

Modbus texts

Value	Text
...	...
90140007	GB block not possible in SWBD
90140008	Switchboard control
90140009	1st priority not possible in SWBD
...	...

Example: Controller status text

`Input register (04) [FC04] - Modbus address 13900 Texts - Controller status text`

The input register for address `13900` returns the value `90140008`.

Search/look up for the value `90140008` in the Modbus texts file it gives the text "Switchboard control".

4. Specific Modbus function groups

4.1 CustomLogic: Modbus signal

You can find the function group *CustomLogic: Modbus signal* in the Discrete output coil (01; 05; 15) and the Discrete input contact (02) worksheets of the Modbus table. The function group allows you to interact with the CustomLogic of the controller using Modbus.

When you read a value from these addresses, the controller will return a value to show if the flag for the signal is active (true, 1) or not active (false, 0). When you write a value to the addresses in the Discrete output coil, the value stored in the address changes to the new value.

NOTE You cannot write values to Modbus signals that have been assigned to coils in CustomLogic.

More information
See **CustomLogic** in the PICUS manual for how to assign a Modbus signal to CustomLogic elements.

4.2 Breaker priority: Buffered value

4.2.1 How it works

```
Requires Power management licence
```

How it works

You can find the function group *Breaker priority: Buffered value* in the Holding register of the Modbus table. The function group acts as a temporary storage area for the breaker priority values that will be written to the controller using the function group *Breaker priority: Write values*.

When you read a value from these addresses, the breaker priority that you want to assign to the breaker that is stored in the address is returned to you. When you write a value to these addresses, the value is stored and ready to be written to the controller when you activate *Breaker priority: Write values*.

CAUTION

Breaker priorities and the Modbus addresses

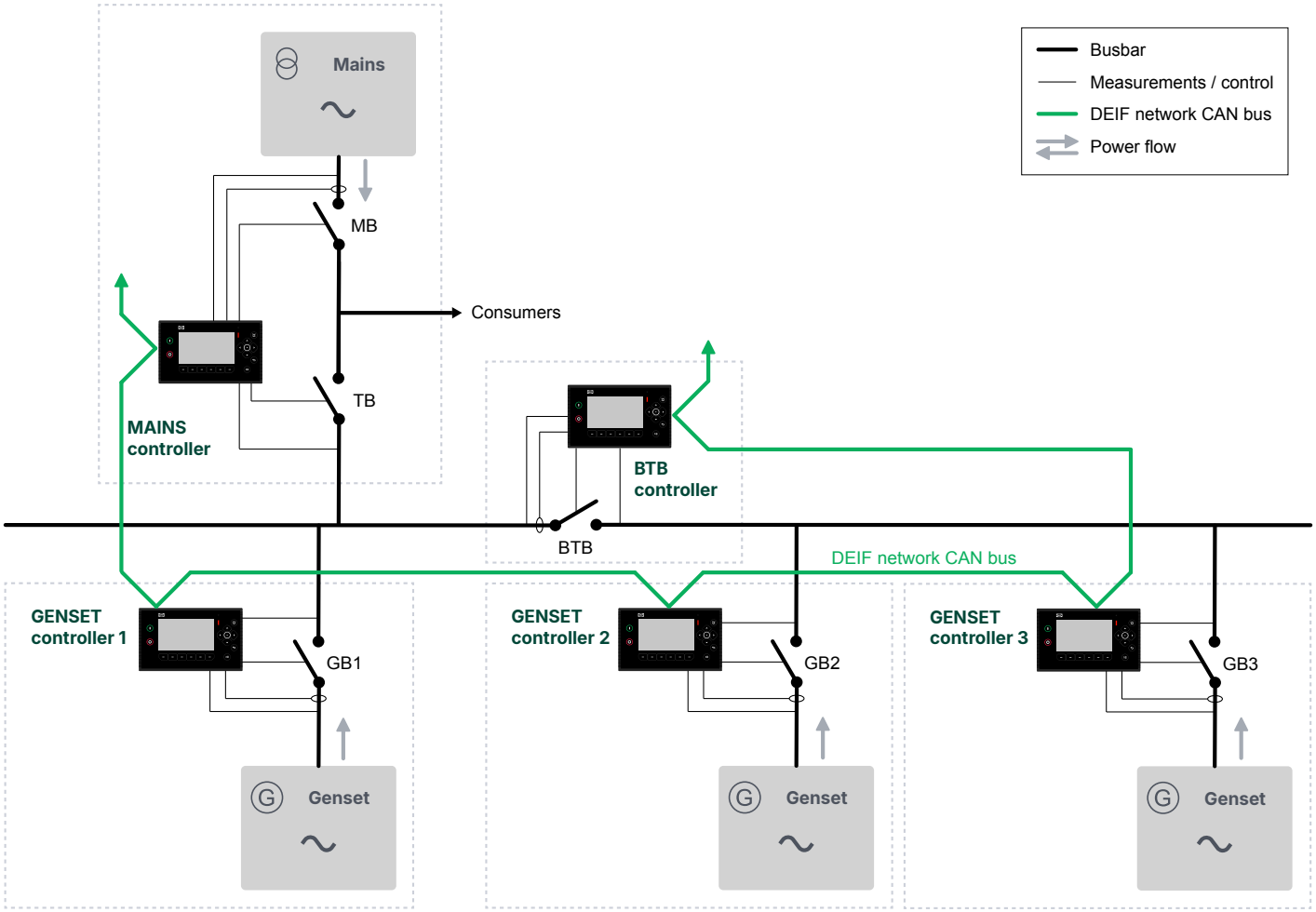
The breaker priorities and the Modbus address associated to a specific breaker is dependent on the **single-line application drawing**. If you change the **single-line application drawing**, you will change the associated Modbus addresses. If you add or remove GENSET controllers from the **single-line application drawing**, the breaker priorities can change.

4.2.2 Breaker priority allocation

These rules apply when breaker priorities are assigned:

- Only **GENSET** controllers receive a breaker priority value that is greater than zero (0).
- All other controller types receive a breaker value of zero (0).
- Marine applications: An **EMERGENCY genset** controller uses two Modbus addresses, the first for the tie breaker and the second for the generator breaker. Both breakers have a breaker priority of zero (0).
- Breaker priorities are assigned to the first available breaker priority, according to the order in which controllers are added to the single-line application drawing.
- Breaker priority Modbus addresses are assigned to the first available breaker priority Modbus address, according to the order in which controllers are added to the single-line application drawing.

4.2.3 Example land application



In this example, it is assumed that the single-line application drawing was drawn by placing the components in the drawing in the following order:

1. Genset controller 1
2. Mains controller
3. Bus tie breaker controller
4. Genset controller 2
5. Genset controller 3

This means that the breakers were assigned the values and priorities:

Component	Modbus address (Holding register and input register)	Breaker priority: Buffered value	Breaker priority: Value
Genset controller 1	14001	0	1
Mains controller	14002	0	0
Bus tie breaker	14003	0	0
Genset controller 2	14004	0	2
Genset controller 3	14005	0	3

The Modbus addresses are assigned to the breaker for the controller. The Modbus addresses are assigned to the components in the order that they were inserted into the single-line application drawing. The Modbus address(es) assigned to a component will not change when the controller ID changes.

Only genset breakers will be assigned a breaker priority value that is between 1 and 128. All other components and addresses which are unassigned (for example 14008 in the example above) have a breaker priority value of 0. Breakers with a breaker priority of 0 assigned to them, cannot be changed.

If a component is removed from the single-line application drawing, the Modbus address becomes free and can be reassigned. The breaker priorities are automatically reassigned for all the remaining components in the single-line application drawing.

For example if we remove Genset controller 1, and Genset controller 3 from the example above, the table will look as follows:

Updated breaker priority values and Modbus addresses after removing components

Component	Modbus address (Holding register and input register)	Breaker priority: Buffered value	Breaker priority: Value
-	14001	0	0
Mains controller	14002	0	0
Bus tie breaker	14003	0	0
Genset controller 2	14004	0	1
-	14005	0	0

If we add Genset controller 1 to the single-line application drawing and then add Genset controller 3, the table will look as follows:

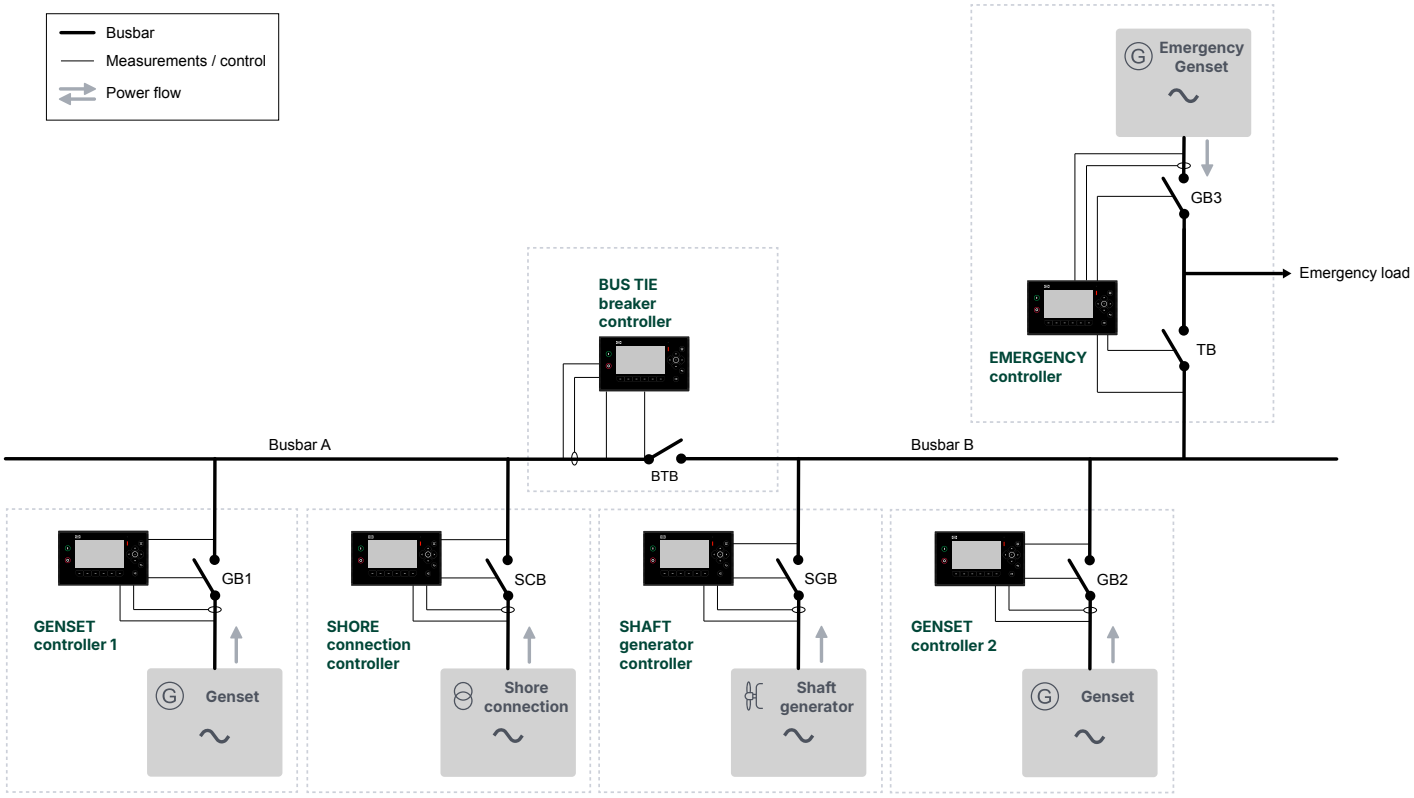
Updated breaker priority values and Modbus addresses after adding components

Component	Modbus address (Holding register and input register)	Breaker priority: Buffered value	Breaker priority: Value
Genset controller 1	14001	0	2
Mains controller	14002	0	0
Bus tie breaker	14003	0	0
Genset controller 2	14004	0	1
Genset controller 3	14005	0	3

The table above shows that the breakers are assigned the first open Modbus address in the Modbus table. This means that it is possible for a genset controller to have its breakers assigned to Modbus addresses that do not follow directly on one another.

When you want to change the breaker priorities by using Modbus, write the desired priority value to the Modbus address in the function group *Breaker priority: Buffered value*. When you are satisfied with the breaker priorities, activate *Breaker priorities: Write values* to write the values to the controller. Only values between 1 and 128 are accepted inputs for breaker priorities. Breakers that already have a priority of 0 assigned to them, cannot be changed. You cannot write the breaker priorities to the controller if there are duplicate non-zero entries in *Breaker priority: Buffered value*.

4.2.4 Example marine application



In this example, it is assumed that the single-line application drawing was drawn by placing the components in the drawing in the following order:

1. Genset 1
2. Shore connection
3. Bus tie breaker
4. Shaft generator
5. Genset 2
6. Emergency genset

This means that the breakers were assigned the values and priorities:

Component	Modbus address (Holding register and input register)	Breaker priority: Buffered value	Breaker priority: Value
Genset 1	14001	0	1
Shore connection	14002	0	0
Bus tie breaker	14003	0	0
Shaft generator	14004	0	0
Genset 2	14005	0	2
Emergency genset	14006	0	0
	14007	0	0

The Modbus addresses are assigned to the breaker for the controller. This means an emergency genset always uses two Modbus addresses for breaker priority, one for each breaker. The Modbus addresses are assigned to the components in the order that they were inserted into the single-line application drawing. The Modbus address(es) assigned to a component will not change when the controller ID changes.

Only genset breakers will be assigned a breaker priority value that is between 1 and 128. All other components and addresses which are unassigned (for example 14008 in the example above) have a breaker priority value of 0. Both the breakers for an emergency genset always have a breaker priority of 0. Breakers with a breaker priority of 0 assigned to them, cannot be changed.

If a component is removed from the single-line application drawing, the Modbus address becomes free and can be reassigned. The breaker priorities are automatically reassigned for all the remaining components in the single-line application drawing.

For example if we remove Genset 1 and the emergency genset from the example above the table will look as follows:

Updated breaker priority values and Modbus addresses after removing components

Component	Modbus address (Holding register and input register)	Breaker priority: Buffered value	Breaker priority: Value
-	14001	0	0
Shore connection	14002	0	0
Bus tie breaker	14003	0	0
Shaft generator	14004	0	0
Genset 2	14005	0	1
-	14006	0	0
-	14007	0	0

If we add the emergency genset to the single-line application drawing and then add Genset 1, the table will look as follows:

Updated breaker priority values and Modbus addresses after adding components

Component	Modbus address (Holding register and input register)	Breaker priority: Buffered value	Breaker priority: Value
Emergency genset (TB)	14001	0	0
Shore connection	14002	0	0
Bus tie breaker	14003	0	0
Shaft generator	14004	0	0
Genset 2	14005	0	1
Emergency genset (GB3)	14006	0	0
Genset 1	14007	0	2

The table above shows that the breakers are assigned the first open Modbus address in the Modbus table. This means that it is possible for an emergency genset to have its breakers assigned to Modbus addresses that do not follow directly on one another. Because Genset 1 has a higher Modbus address (14007) than Genset 2 (14005), by default it is assigned a lower priority than Genset 2 after the change was made in the single-line application drawing.

When you want to change the breaker priorities by using Modbus, write the desired priority value to the Modbus address in the function group *Breaker priority: Buffered value*. When you are satisfied with the breaker priorities, activate *Breaker priorities: Write values* to write the values to the controller. Only values between 1 and 128 are accepted inputs for breaker priorities. Breakers that already have a priority of 0 assigned to them, cannot be changed. You cannot write the breaker priorities to the controller if there are duplicate non-zero entries in *Breaker priority: Buffered value*. The tables below show the results after new breaker priorities were written to the buffered values, and after the buffered values were written to the controller.

Breaker priority values after writing new values to the buffer addresses

Component	Modbus address (Holding register and input register)	Breaker priority: Buffered value	Breaker priority: Value
Emergency genset (TB)	14001	0	0
Shore connection	14002	0	0
Bus tie breaker	14003	0	0
Shaft generator	14004	0	0
Genset 2	14005	2	1
Emergency genset (GB3)	14006	0	0
Genset 1	14007	1	2

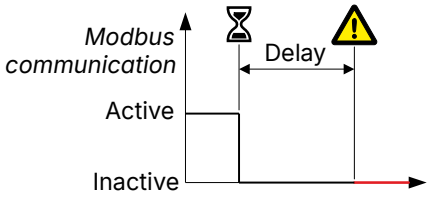
Breaker priority values after writing the buffered values to the controller

Component	Modbus address (Holding register and input register)	Breaker priority: Buffered value	Breaker priority: Value
Emergency genset (TB)	14001	0	0
Shore connection	14002	0	0
Bus tie breaker	14003	0	0
Shaft generator	14004	0	0
Genset 2	14005	2	2
Emergency genset (GB3)	14006	0	0
Genset 1	14007	1	1

5. Modbus alarm

5.1 Modbus communication timeout

The controller activates this alarm if there are no Modbus requests within the delay time.



`Communication > Modbus > Modbus communication timeout`

Parameter	Range
`Delay`	0.1 s to 1 h