KEIT INDUSTRIAL ANALYTICS

IRmadillo™

Commissioning Guide:
Data Communication - Modbus TCP

www.keit.co.uk
support@keit.co.uk

Table of Contents

1. Scope

This guide explains how to set up data communications between the IRmadillo controller and your plant's Distributed Control System (DCS) using Modbus TCP. This guide requires that Keit has already set up your controller with its chemometric models and has enabled the Modbus connection.

Before starting, ensure that:

- Your IRmadillo is installed and running
  - If you have an external controller, ensure that is also installed and running
- Keit has provided you with the Register Mapping Document

If you have any questions, contact Keit using: support@keit.co.uk. More detailed information is in the following IRmadillo user manuals, found at https://keit.co.uk/user-manual/:

KeitSpec	**DOC0893**
IRmadillo ASM0627-10-Z (internal controller)	**DOC0935**
IRmadillo ASM0627-09-Z (external controller)	**DOC0945**

1.1. Overview

An overview of the analyser and communications layout is shown below for both types of controller.

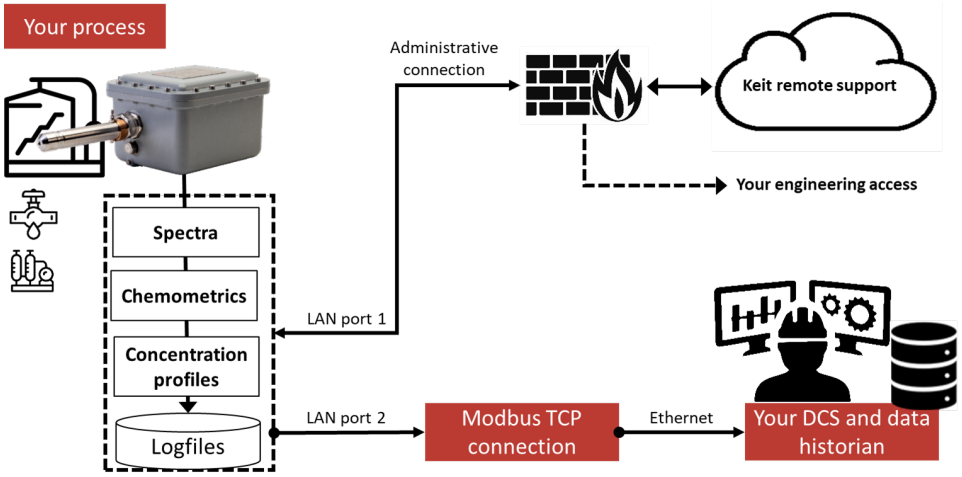


Figure 1 System schematic for ASM0627-10 (internal controller)

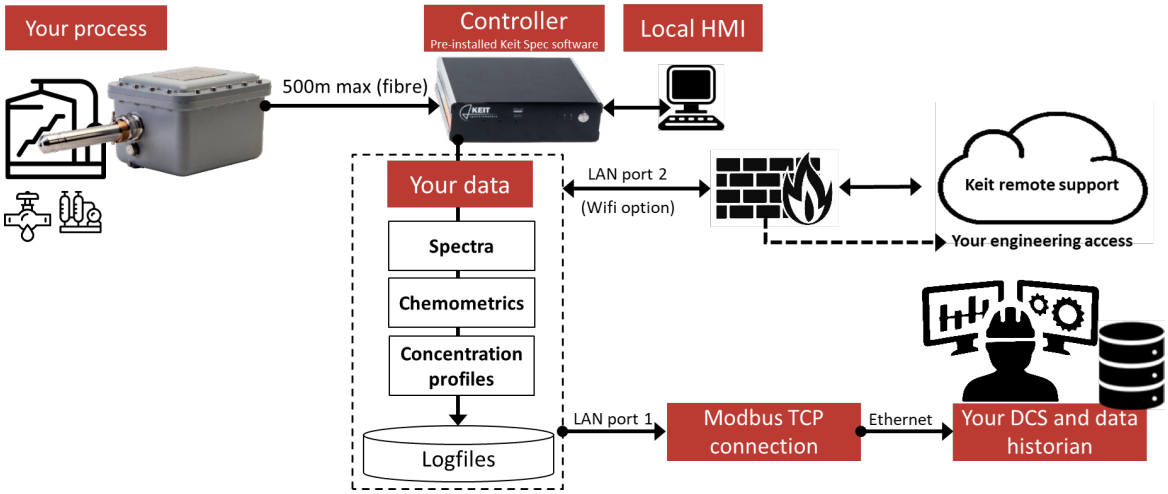


Figure 2 System schematic for ASM0627-09 (external controller)

2. Implementing Modbus TCP data communications

This section will show you how to set up the Modbus TCP data communications.

2.1. Finding the IP address of the Modbus TCP Server

2.1.1. In KeitSpec, with a Continuous Collection project open, click the *Configure Acquisition* tab, then toggle *Modbus TCP Server* to *ON*.

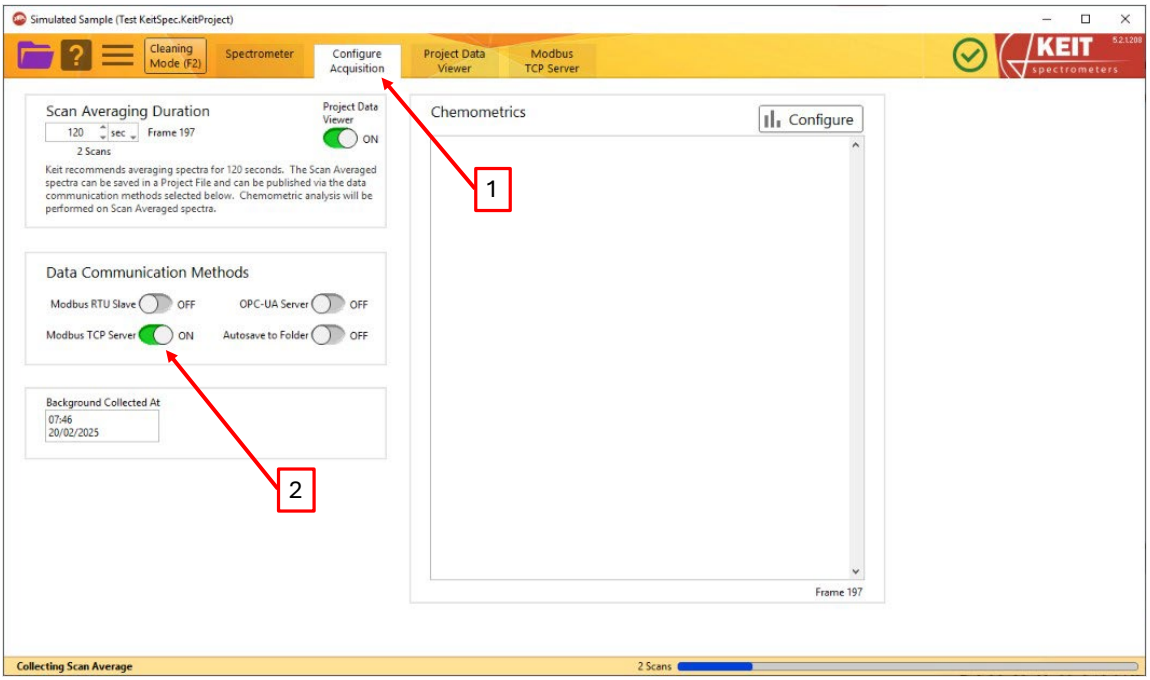


Figure 3 Toggle the Modbus TCP Server Tab

2.1.2. Click the *Modbus TCP Server* tab that appears. In the *Network Adapters* box, the IP address of the controller should be listed.

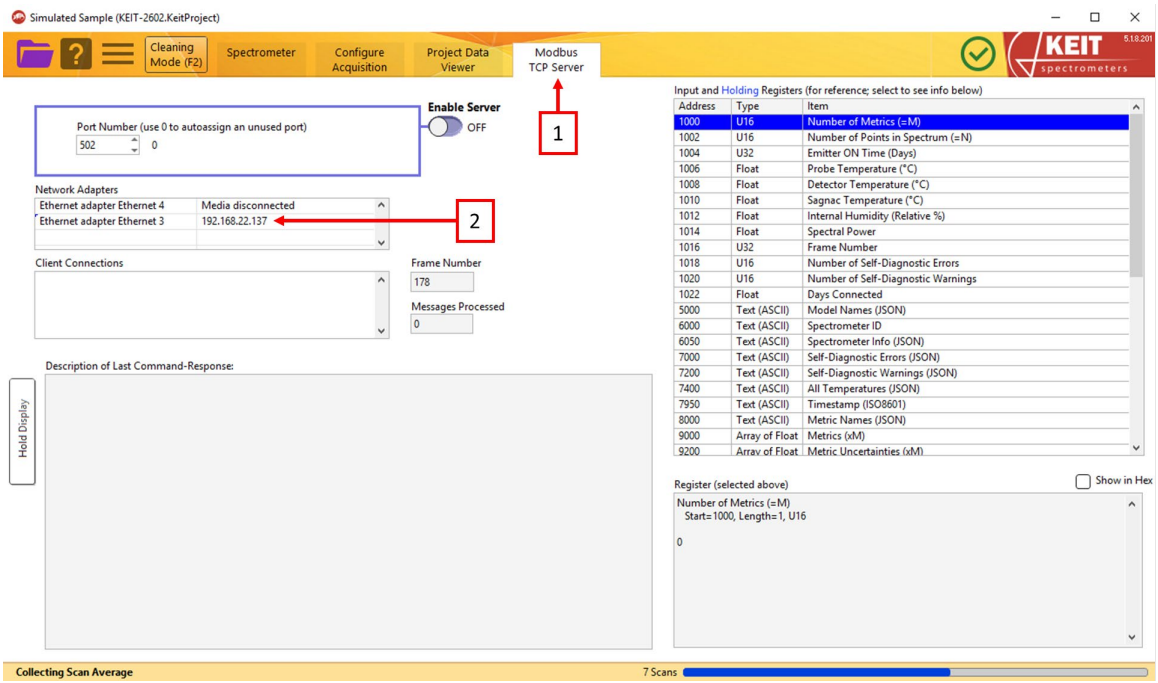


Figure 4 Identifying the IP address

2.2. Setting a static IP address

You may need to assign a specific IP address to your controller. You can do this as follows:

2.2.1. Type ”Control Panel” into the Windows search bar and press *Enter* to open the control panel. In the control panel window, click *Network and Internet* and then *Network and Sharing Center*.

2.2.2. Click *Change adapter settings*.

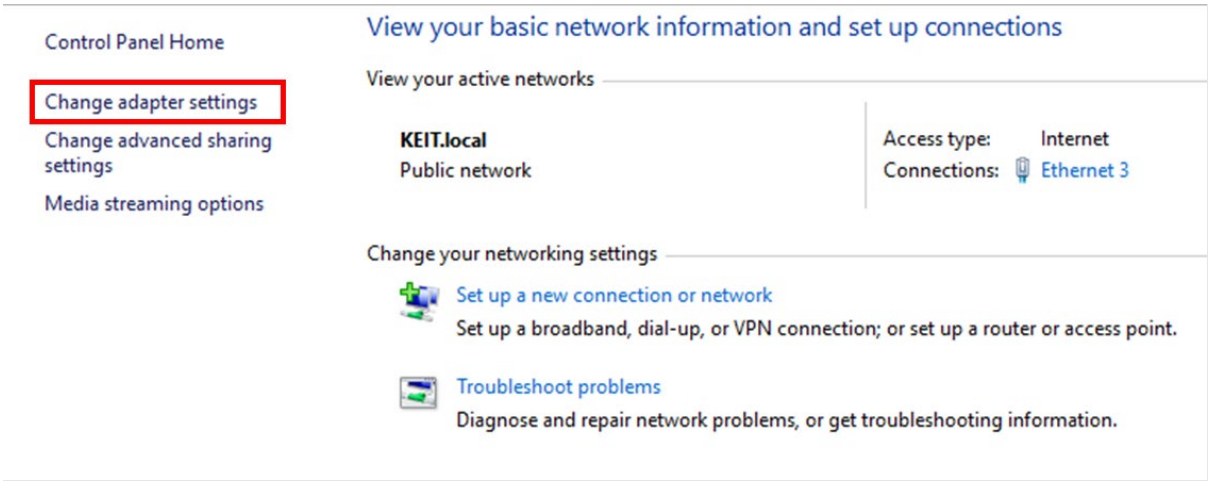


Figure 5 Network and sharing centre

2.2.3. Right-click on the relevant network connection (‘Ethernet 3’ in Figure 6, below) and click *Properties*. You will need to enter the administrator password, which can be found on the sticker on the front of your controller (or, if you have an integrated controller, is shared with you electronically).

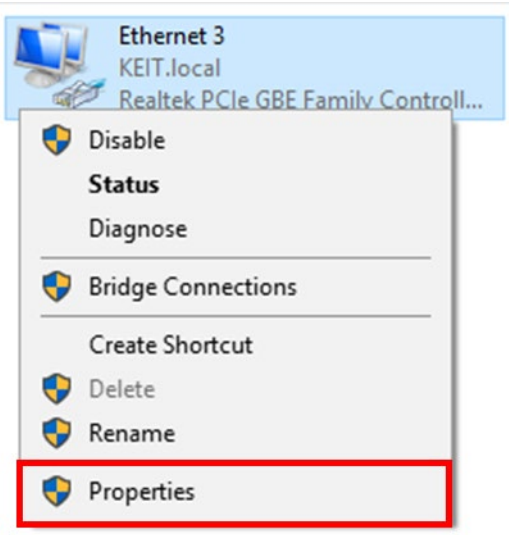


Figure 6 Change adapter settings

2.2.4. Click on *Internet Protocol Version 4 (TCP/IPv4)*, then click *Properties*.

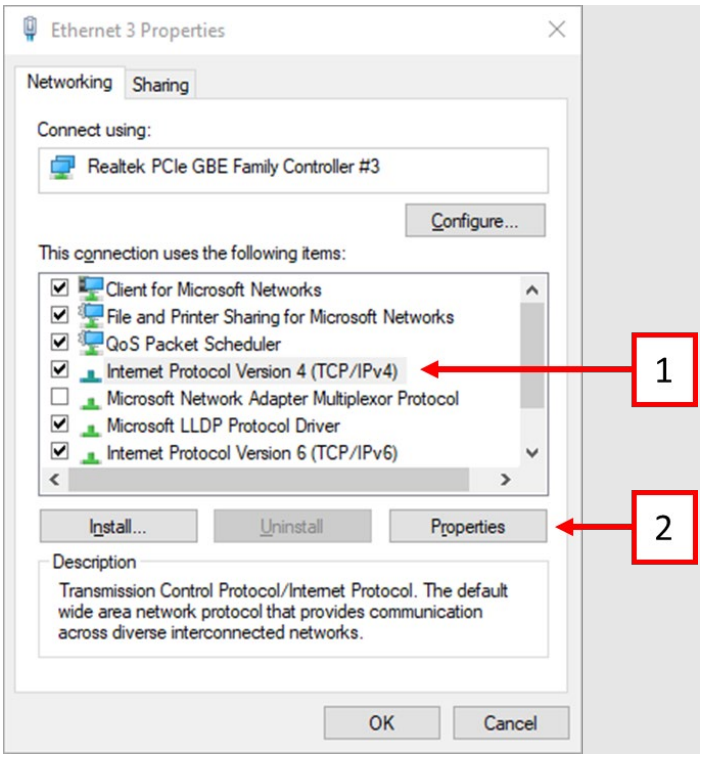


Figure 7 Ethernet properties

2.2.5. Select *Use the following IP address*. Enter the IP address, subnet mask and default gateway that you want to use, then click *OK*.

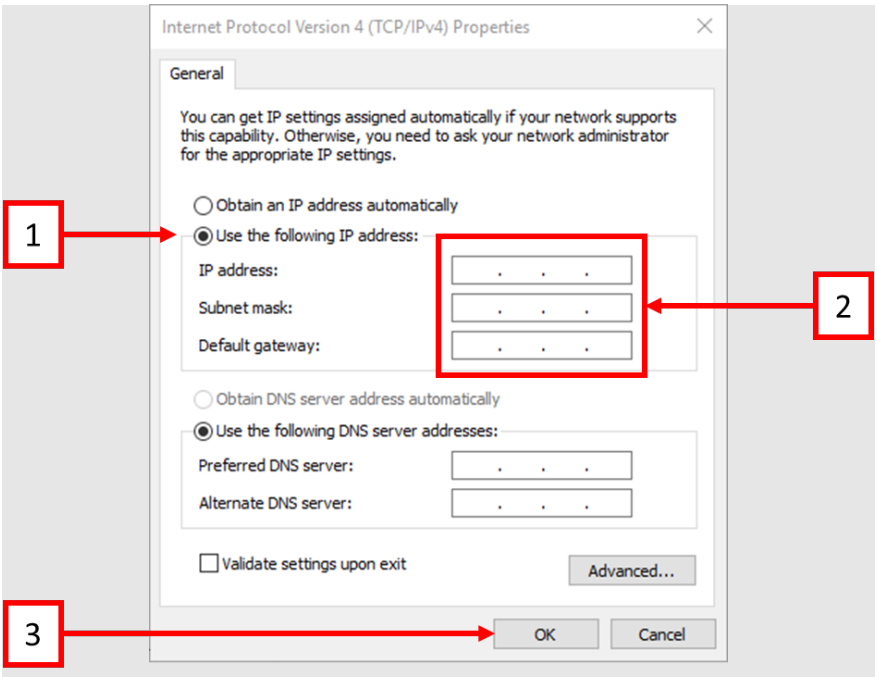


Figure 8 Setting a static IP address

2.2.6. Click *Close*. Your controller will now use the assigned IP address.

If needed, you may assign two different fixed IP addresses, one for each of the IRmadillo's ethernet ports.

2.3. Connecting your Modbus client to KeitSpec

Returning to the *Modbus TCP Server* tab of KeitSpec:

2.3.1. The default port number is 502. If you need to change the port number, this can be done when the server is disabled.

2.3.2. If your Modbus client requires a Unit ID (also called Device ID), enter 1. This cannot be changed.

2.3.3. Toggle ‘Enable Server’ to ‘On’.

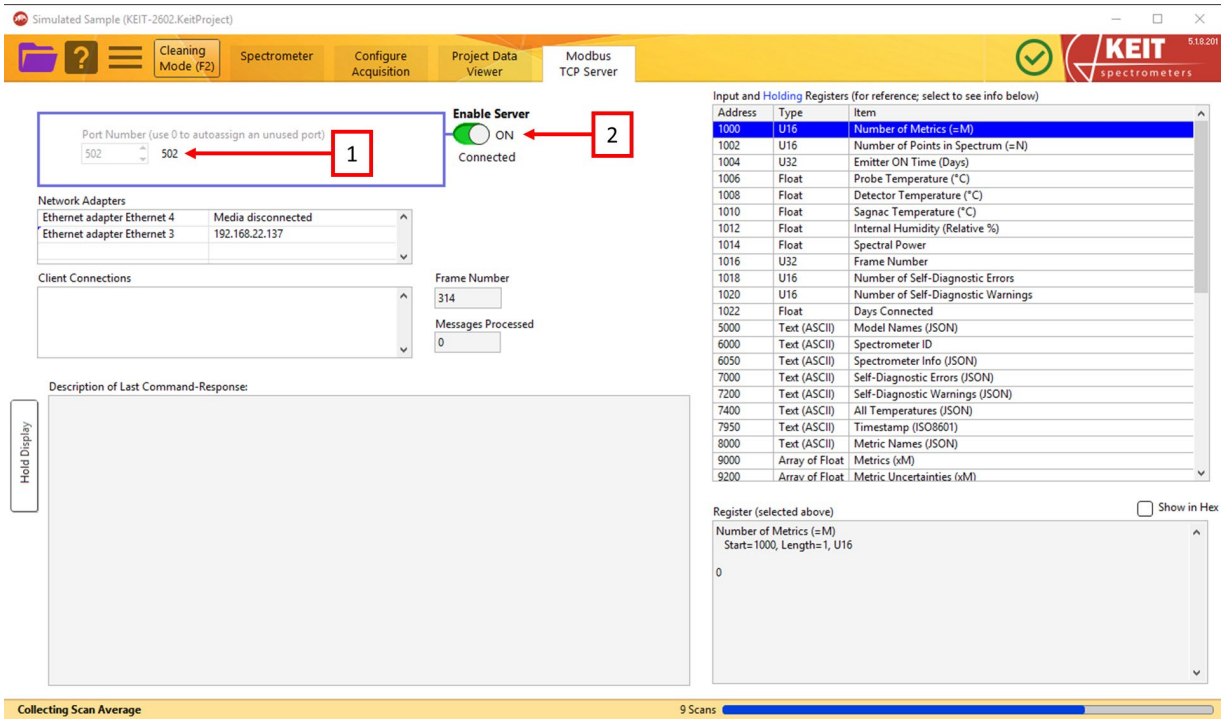


Figure 9 Server port settings

2.4. Representation of data in Modbus

2.4.1. Before testing the connection, know that data is read from ‘Input Registers’ unless stated otherwise.

2.4.2. Data is encoded as follows:

Data Type	Size (Registers)	Description
Float	2	IEEE 32-bit float in 2 Registers, Low Address=Most Significant U16. Sometimes referred to as a “switched float” or “IEEEFP (Big Endian)”.
Boolean	1	U16, False=0, True=1 (or higher)
U16	1	Unsigned 16-bit integer
U32	2	32-bit unsigned integer in 2 Registers, Low Address=Most Significant U16

2.5. Configure your client and connect

2.5.1. From your Modbus client, read two input registers starting at register 1016 (decimal). This will return the Frame Number as an unsigned 32-bit integer (U32). This is an internal counter – it will increment regularly whether it is being polled or not. The *Modbus TCP Server* tab in KeitSpec displays debug information in the lower two panels, as shown below:

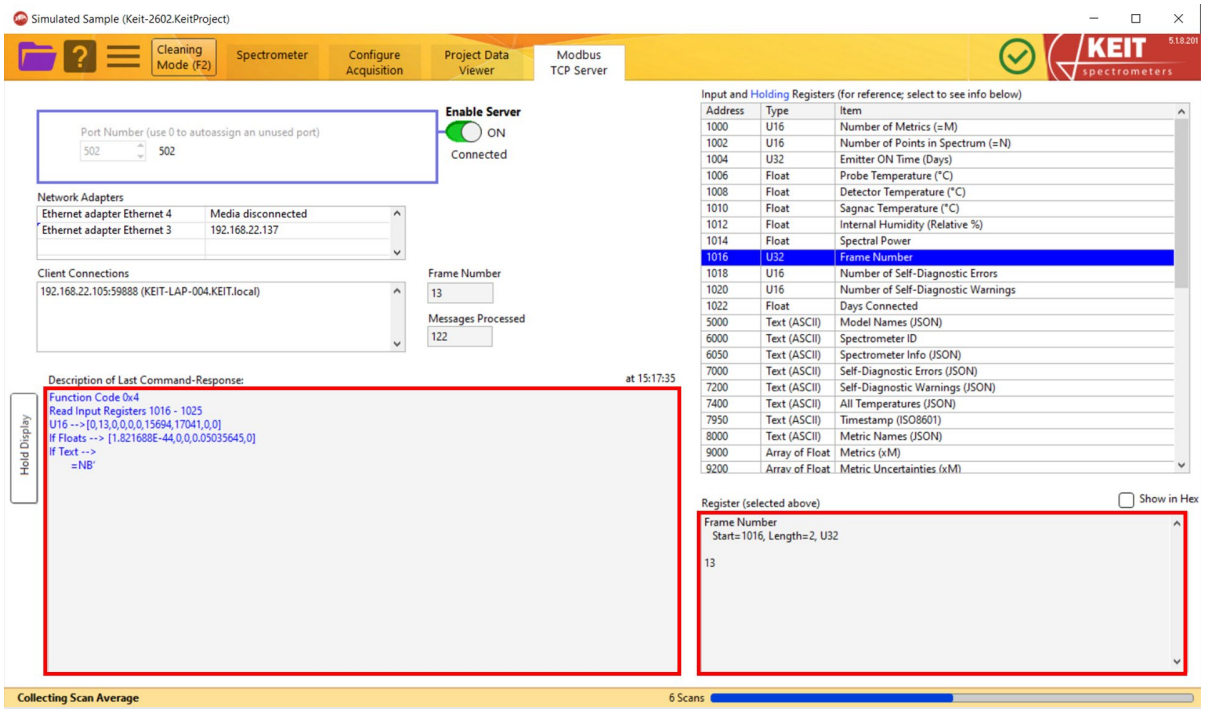
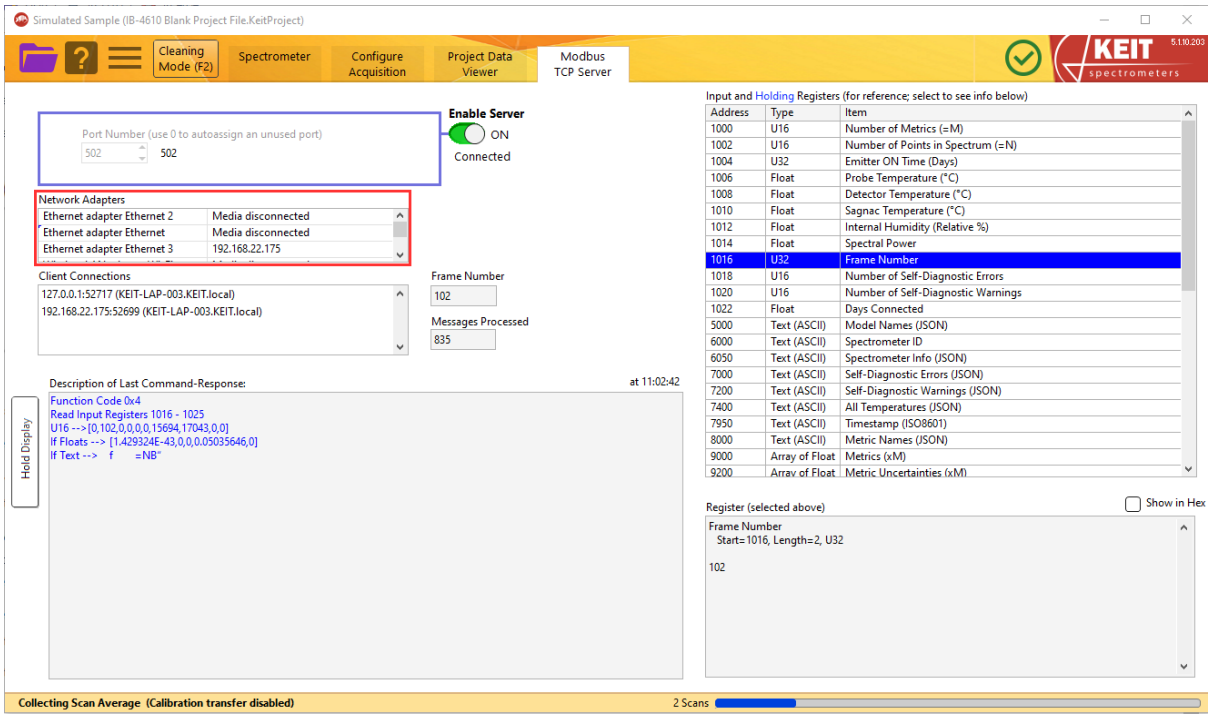


Figure 10 Checking the correct data is read

2.5.2. The value you read should match the number shown in the *Register* pane outlined in red above. If this is the case, then you have successfully established communications.

2.5.3. KeitSpec can handle a maximum of 254 simultaneous TCP connections, some of which are already used by other parts of the software. Please ensure that unused ModbusTCP connections are closed. Leaving large numbers of TCP connections open will cause KeitSpec to crash. You can see a list of all the ModbusTCP connections that are currently open in the 'Network Adapters' box.



2.6. Polling for new data using the frame number ‘heartbeat’

2.6.1. Register 1016 (decimal) is the ‘frame number’, which may be used as a heartbeat to indicate that fresh data is being returned by the IRmadillo. The register data are always available and will update every 120 seconds.

2.6.2. If you only want to read when there is new data, then read the frame number from input register 1016 (decimal). This changes whenever there is new data. It is a 32-bit unsigned integer [U32] and will increment by an integer number. The amount it changes by each time varies. At some point it will wrap around to zero.

2.6.3. If register 1016 does not change for >10 minutes, intervention may be required. Set up a suitable alarm/warning to flag the user.

2.7. Configure and test the historian

2.7.1. All of the items in the Register Mapping Document must be stored in the plant historian. They are not stored on the IRmadillo controller.

2.7.2. Check that the data recorded in the historian matches that given by the spectrometer. Please see Section 4 Testing your DCS configuration for more information.

2.8. Changing Settings and Performing Actions

2.8.1. If you need to change a setting in KeitSpec via Modbus, it is possible that KeitSpec will fail to return the handshake signal that your Modbus client may be expecting. Keit recommends reading the holding register back to confirm that the setting has been changed correctly.

2.8.2. One of the settings that can be changed is the Emitter state, holding register 110. If you set this to 0=OFF/false, the spectrometer will cease returning spectra and the frame counter will stop updating.

2.9. Example Register Mapping Document

Below is an example of a Register Mapping Document.

The spectrometer is shipped with a set of training models. The models are read as single precision floating point numbers from pairs of registers from 9000 onwards.

Training models can be used to help integrate the IRmadillo into your DCS earlier in the commissioning of the system. The models output fixed values to make setup easier and help with troubleshooting problems. The number of registers that need to be read will depend on what is intended to be measured by the spectrometer.

Multiple Confidence models are built to assess the health of the individual measurements and report they can be trusted. They act as Boolean status flags and should be presented to the end user along with the measurement data to prevent users relying on data that the model has flagged as lower confidence values. The training Confidence models will only report 0.

Register	Type	Name	Description
1016	U16	Frame Number	Heartbeat – changes when there is new data
1018	U16	Number of Self-Diagnostic Errors	Number of errors thrown by the spectrometer. If this is non-zero for more than an hour when the spectrometer has been running for at least 12 hours, contact Keit.
1020	U16	Number of Self-Diagnostic Warnings	Number of warnings thrown by the spectrometer. If this is non-zero for more than an hour when the spectrometer has been running for at least 12 hours, contact Keit.
9000	Float32	Model 1	Fixed output 1.111111
9006	Float32	Confidence 1	Calibration confidence (0 = within specification, 1 = Low confidence)
9008	Float32	Model 2	Fixed output 2.222222
9014	Float32	Confidence 2	Calibration confidence (0 = within specification, 1 = Low confidence)
9016	Float32	Model 3	Fixed output 3.333333
9022	Float32	Confidence 3	Calibration confidence (0 = within specification, 1 = Low confidence)
9024	Float32	Model 4	Fixed output 4.444444
9030	Float32	Confidence 4	Calibration confidence (0 = within specification, 1 = Low confidence)
9032	Float32	Model 5	Fixed output 5.555555
9038	Float32	Confidence 5	Calibration confidence (0 = within specification, 1 = Low confidence)
9040	Float32	Model 6	Fixed output 6.666666
9046	Float32	Confidence 6	Calibration confidence (0 = within specification, 1 = Low confidence)
9048	Float32	Model 7	Fixed output 7.777777
9054	Float32	Confidence 7	Calibration confidence (0 = within specification, 1 = Low confidence)
9056	Float32	Model 8	Fixed output 8.888888
9062	Float32	Confidence 8	Calibration confidence (0 = within specification, 1 = Low confidence)
9064	Float32	Model 9	Fixed output 9.999999
9070	Float32	Confidence 9	Calibration confidence (0 = within specification, 1 = Low confidence)

2.10. Handling Negative Concentration Values

The IRmadillo may occasionally report **negative concentration values** for some chemical species. These values are not physically meaningful and occur due to model extrapolation or noise when the true concentration is near zero.

Where this applies:

- These values are typically found in the **chemometric model outputs**, which are labelled as **Model #** in your mapping document:
    - Registers in the range **9002, 9010, 9018, …** (see Section 2.9 Example Register Mapping Document).

If negative values are not desired, configure your DCS to replace any negative values with zero before displaying or using them in control logic.

$$If\ x < 0\ then\ x' = 0$$

This can be implemented in most DCS systems using basic conditional logic or scripting.

2.11. Control systems

The Frame Number is the indicator from which it is possible to tell if the device is recording spectra and upon which control systems should be based. The frame number updates with every scan average (usually 120 seconds) whether it is being polled or not.

To minimise the risk of integral wind-up, PID systems should not expect a continuous stream of values. Concentrations and other readings will only update when the frame number updates, not more frequently.

In the rare event that KeitSpec loses connection with the IRmadillo, your DCS values will not change.

3. Troubleshooting Connectivity

If you encounter issues when setting up your Modbus TCP system, please check the following:

3.1. User Accounts

- When setting up Modbus TCP in KeitSpec you must use the *Spectrometer user* account, not *Administrator*.
- KeitSpec must only be run from the *Spectrometer user* account. Concurrent instances of KeitSpec on different accounts (e.g. the *Administrator* account) will cause issues.

3.2. Project setup

- A Continuous Collection project must be loaded in KeitSpec. DCS communications are not available in the Manual Sample Acquisition project type.

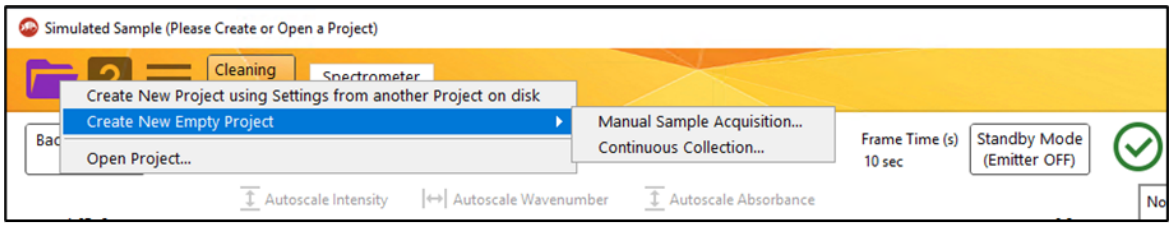


Figure 11 Creating a new project

- To see updating data, the project must be running and collecting spectra. If not, Modbus TCP will not register new information that is being collected. Make sure that the spectrometer is not in Standby Mode; when in Standby Mode, the *Standby Mode* button, highlighted below, will be red.

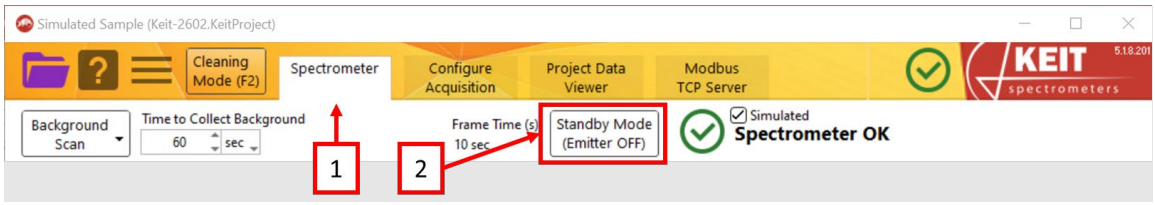


Figure 12 Turning standby mode on/off

3.3. Other Communication Issues

- You may find it useful to install a separate Modbus server program such as WireShark on your controller so you can test and debug the Modbus connection separately to the KeitSpec Modbus server.

- You may also find it useful to install a basic Modbus client, such as ModSIM, MODScan or Radzio!, to test and debug the connection separately to your DCS client.

- Communication issues may be caused by restrictive network configuration and firewall settings. Check the controller's firewall policy and ensure that KeitSpec has permission to communicate over your network type (domain, private or public).

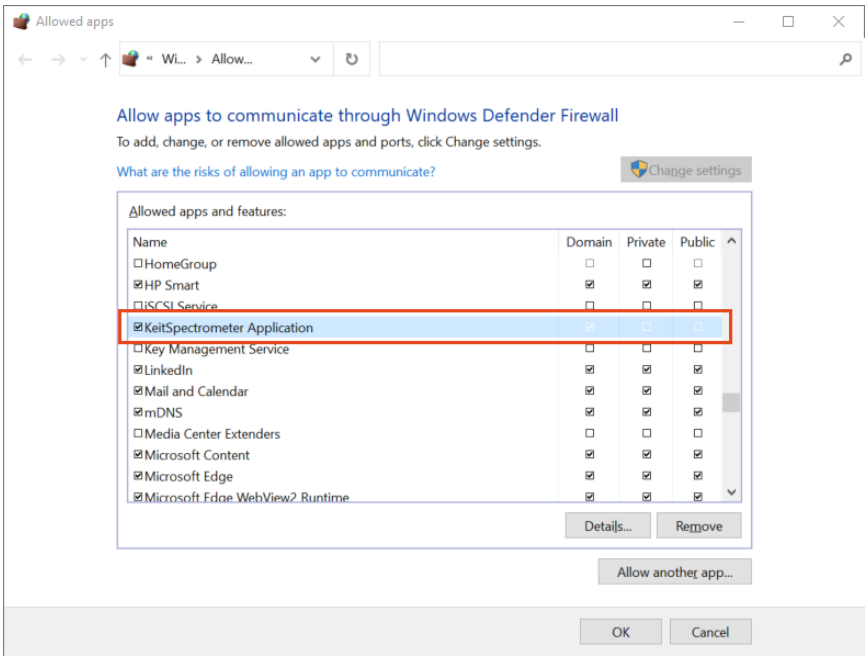


Figure 13 Firewall restrictions

- KeitSpec may sometimes take several seconds to respond to a command, which can lead to timeout errors in the Modbus client. If this is an issue, Keit recommends increasing your Modbus client's Connect Timeout to at least 30 seconds and Read (or Transaction) Timeout to at least 20 seconds. Examples are shown below.

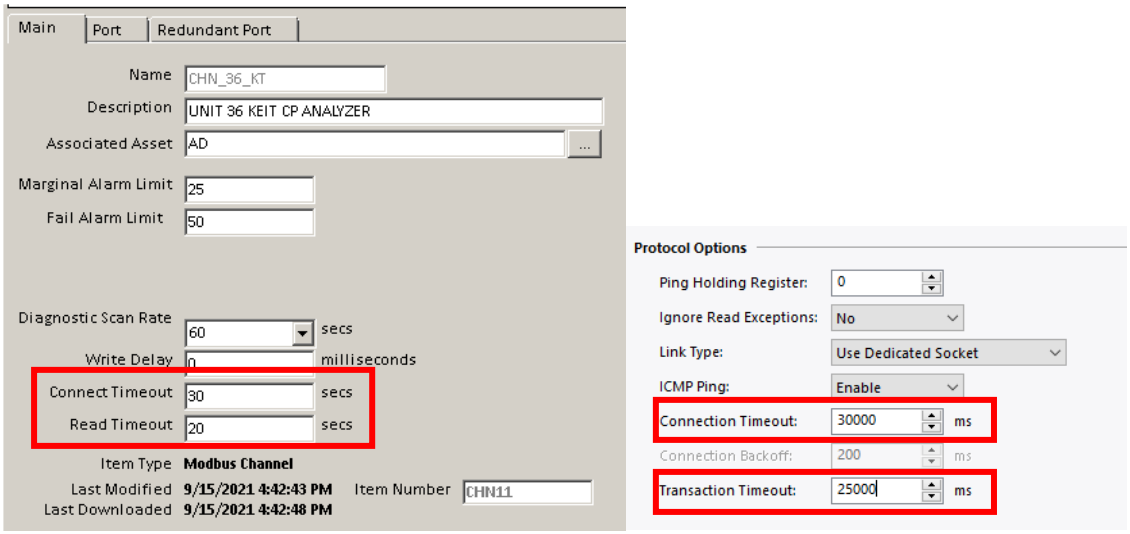


Figure 14 Timeout and read settings

- If you are experiencing intermittent communication between KeitSpec and your DCS, or if you see a Device Name Error in either your DCS client or another program that you are using for troubleshooting, it may be necessary to add your Modbus client to the hosts file on the controller. The hosts file can be found in:

  `Windows\System32\drivers\etc`

and can be edited with WordPad, Notepad++ or a similar text editor. Add the IP address and name of the PC running your Modbus client to the end of this file.

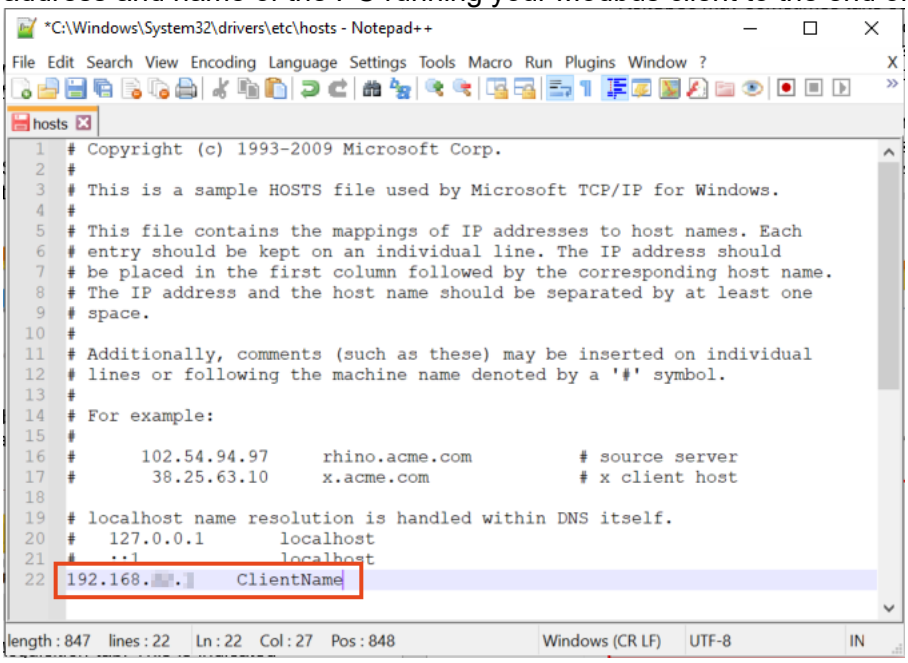


Figure 15 Adding Modbus client to the controller

4. Testing your DCS configuration

An incorrect configuration of your DCS can lead to the concentrations read by your DCS not matching the outputs of the models in KeitSpec. If this is the case, please check the following:

4.1. Setting up a Test mode

KeitSpec can be configured with dummy chemometric models that output constant values over Modbus TCP. This may be useful for troubleshooting.

4.1.1. Go to the 'Configure Acquisition' tab.

4.1.2. Click on the 'Configure Chemometrics' button.

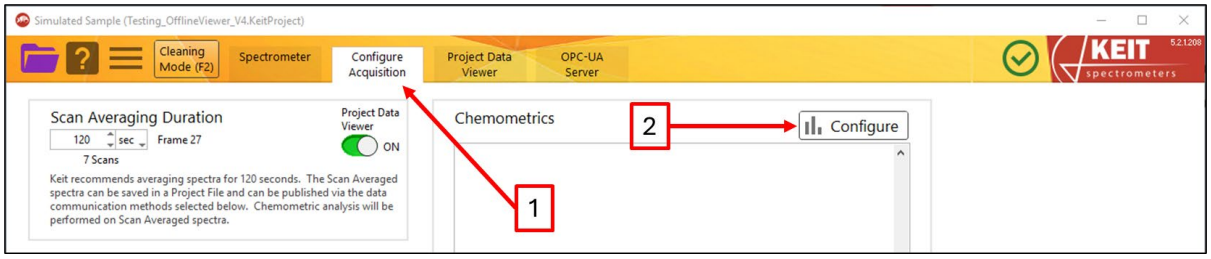


Figure 16 Configure Acquisition

4.1.3. A 'Chemometrics Editor' window will appear. You may already have some chemometric models loaded; this is OK. Add a new model by clicking on the blue plus icon and then select 'Add New Model >> Calculation'.

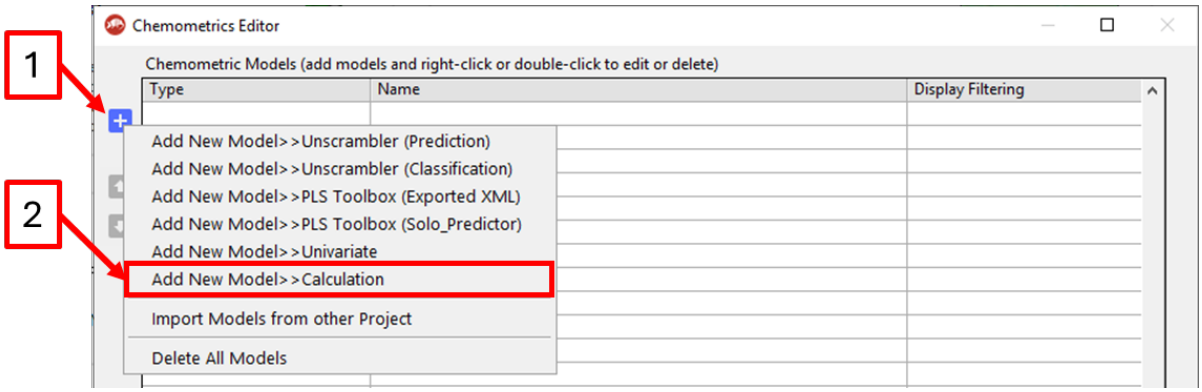


Figure 17 Adding a Calculation model

4.1.4. Type a sensible name for your model in the 'Name' box and enter the value in the 'Expression' box, then click 'OK'. The 'Expression' box will turn green when the syntax is correct.

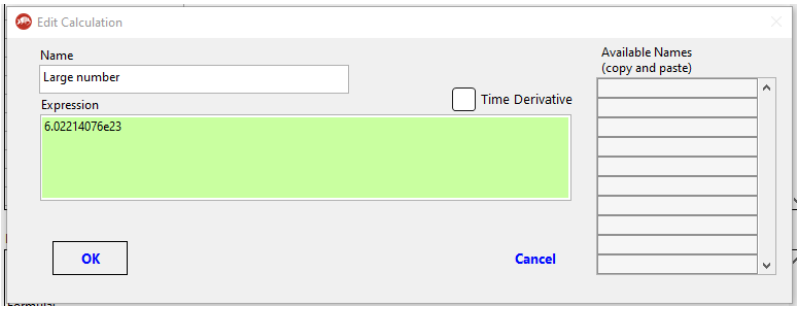


Figure 18 setting up a calculation

4.1.5. You can add as many models as you think useful. It will be useful to ensure that some of the numbers have a large number of decimal places. When you have finished adding models, click the ‘Done’ button.

4.1.6. When the next averaged frame is available, the outputs of your models will appear in the ‘Chemometrics’ window. This may take up to two minutes.

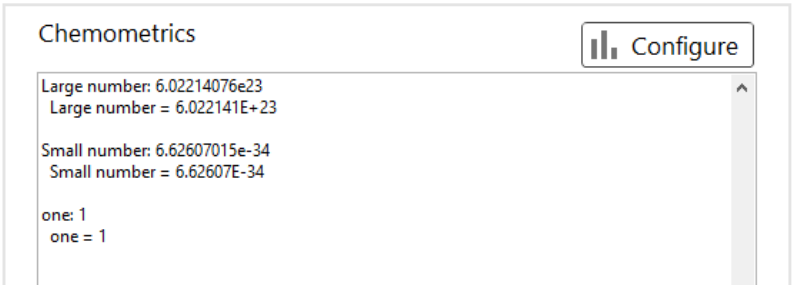


Figure 19 Model output

4.1.7. These numbers will now be available over your DCS. Each of these numbers will have its own Modbus register, appended to the end of the registers for the models you already have. For example, if the last model listed in your DCS mapping document is output on register 9016 and you have added three constant models, these models will be on registers 9018, 9020 and 9022.

4.2. Checking your Outputs

This example uses QModMaster to read the Modbus registers. QModMaster is set up as in the settings shown in the screengrab below (1): the base address is 0 (zero-based addressing) and QModMaster is set to big-endian.

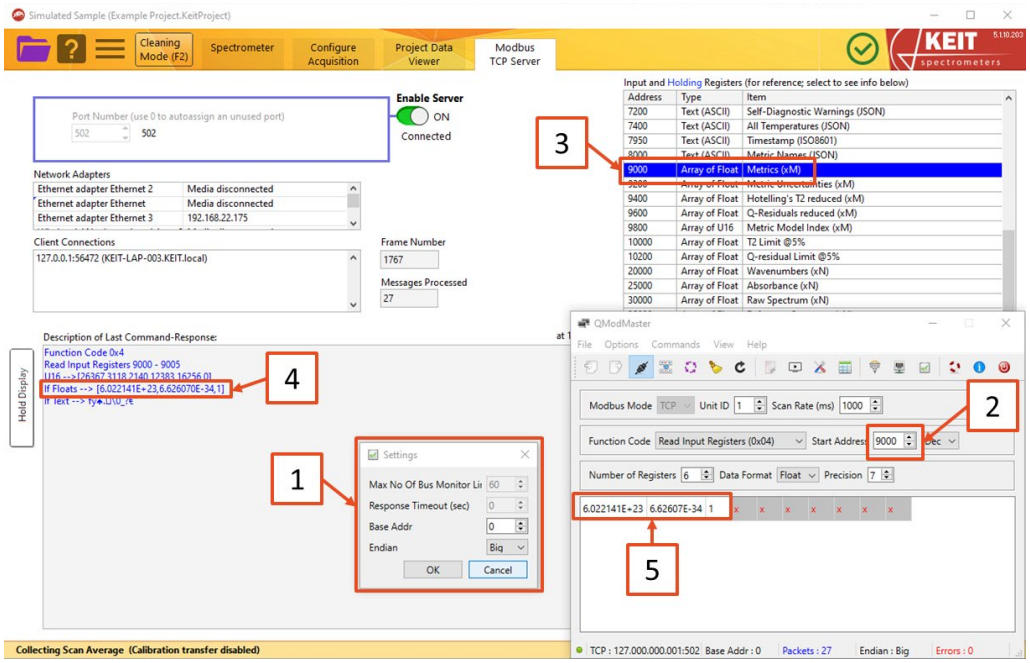


Figure 20 Modbus TCP output of models

We are reading the outputs of chemometric models from register 9000 (2). As shown in KeitSpec (3), this is an Array of Floats. Therefore, in the Description of Last Command-Response tab, look at the ‘If Floats’ line (4).

Compare these values to the values in QModMaster (5). These values should match exactly to as many decimal places as you are reading (in this case 7). If these values do not match, some things to check are given in the following sections.

4.3. Register Offsets

Some Modbus clients, e.g. Enron, have register offsets by default, meaning that your DCS will read a register that is a number of registers higher or lower than those requested. Please ensure that your DCS has no register offset. This may also be called zero-based addressing.

4.4. Endianness

The outputs of KeitSpec are big-endian. Each model output occupies two Modbus registers, and the endianness defines the order in which these registers are read. Reading the model outputs as little-endian usually gives results that are very obviously wrong, but in the situation where the endianness is wrong and there is a single register offset, you may get readings which are correct to three decimal places and then incorrect for the remainder.

The endianness setting in your DCS may also be called the register order.