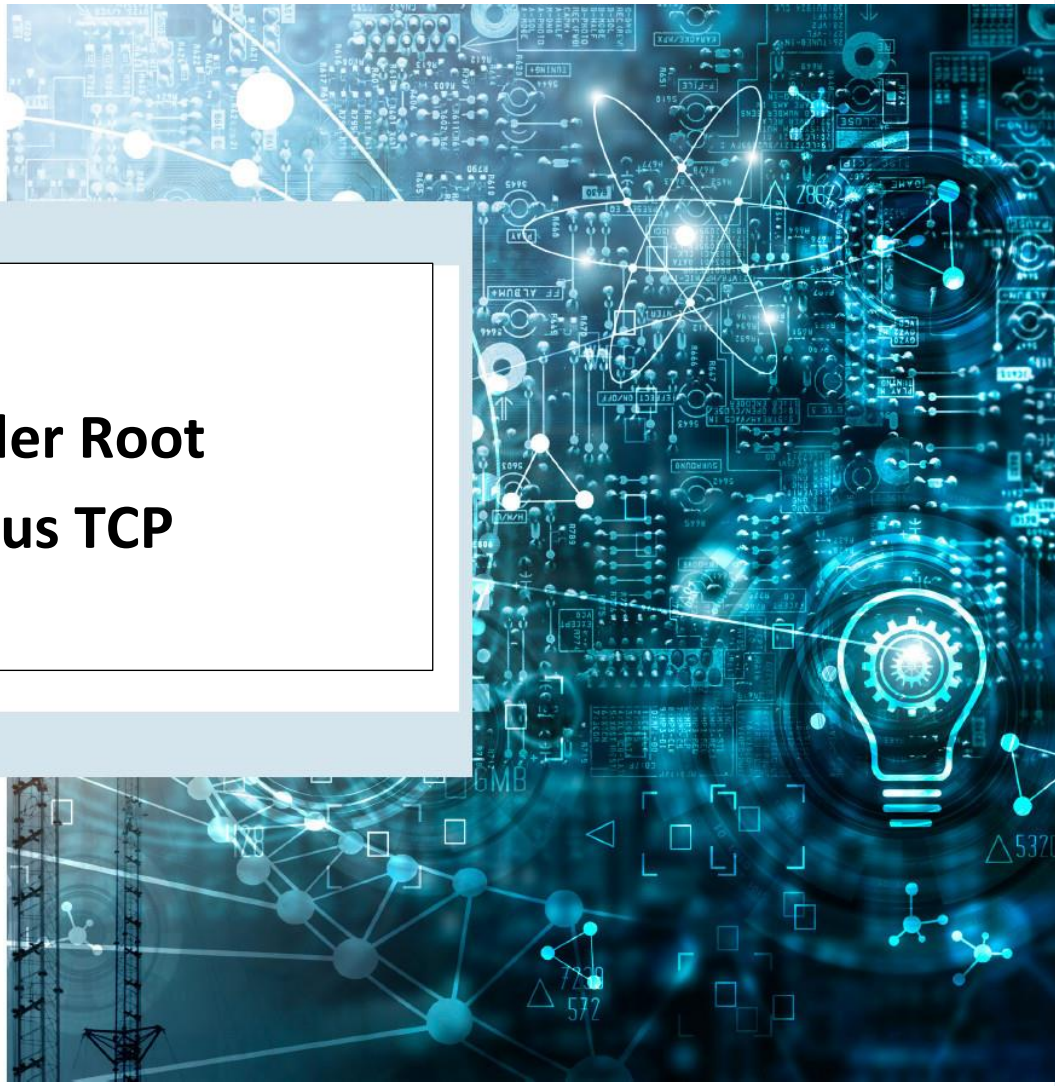
CHIPKIN



Case Study

EVO™ 5000 (Veeder Root Protocol) – Modbus TCP

Overview

This case study involves a client who was looking for a solution on how to read data from two 40,000-gallon fuel tanks using their Building Management System. Their objective was to integrate two **EVO™ 5000** (Veeder Root) fuel tanks into their BMS which could only read from the Modbus TCP protocol.

The unique aspect of this project was the FieldServer QuickServer Veeder root driver had not been built or tested with the client's **EVO™ 5000 INCON** devices. Our hope was that the Veeder Root driver would be compatible with these devices.

Initially, we were having issues communicating with the **EVO™ 5000.** We provided the customer with a wiring diagram to follow, and he was able to properly connect to the device. Even despite verifying that connection was proper, we were still unable to get a connection. But after many hours of support, Chipkin was able to resolve the issue and establish a flawless connection for our client.

Chipkin's approach to the solution

After listening to the customer's requirements, we suggested that he use an FS-QS-2020-0659 FieldServer QuickServer device. This device has the capability of facilitating data between the Veeder Root and Modbus/TCP protocols. It can read/serve up to 250 data points.

Along with the FieldServer QuickServer gateway, Chipkin recommended custom configuration services as a solution to complete the integration.

With this solution, the client provided Chipkin with all the information we needed to write a configuration file. We requested the number of tanks he needed us to integrate, whether there were any vapor or liquid sensors, and if he wanted us to use a relay. Veeder Root is an extensive protocol that can provide a wide range of data, depending on the customer's needs. Once we were certain of his requirements, we wrote a configuration file to poll for the data points from the **EVO™ 5000** device.

Chipkin used the information provided and much testing to generate a functional configuration file (config.csv). Once it was completed, we provided it to the customer along with the driver manual and instructions on how to download the file onto his FieldServer. Also, as a courtesy to the customer, we wrote a Modbus Map that would help the end-user when it came to mapping these points in their BMS. This map contained matching Modbus Registers with the corresponding **EVO™ 5000** data points. For example, Modbus register 40001 was the register that we used to store the 1st alarm point off of tank #1.

Communication & Data Integrity

We did experience communication issues but were able to resolve them by updating the config.csv file to include a parameter that reads TLS-450 data. Our driver default (unless specified) is set up to read TLS-350, which was the cause of the errors. Once we changed that parameter in the configuration file to read as TLS-450, we resolved the connection issues.

Another issue we faced during this project was how to resolve the issue of turning a FLOAT data type into Modbus Registers. **EVO™ 5000** (Veeder Root) represented their data as FLOAT values. We wrote the configuration to read them as such. However, the Modbus Protocol uses various function codes that only allow for 16-bit registers. So how did we turn FLOATS into registers? Simple. We used a feature in the FieldServer which allows us to convert FLOAT values into two 16-bit registers using a "move". Moves can be created in the configuration file. So, for every single FLOAT value, we moved them into two 16-bit Modbus TCP registers. When we wrote the Modbus Map, we also went into detail explaining to the customer the endianness. In the end, the customer was able to map the data appropriately using their BMS.

With the above conclusions, the integration was successfully completed, earning Chipkin a happy client.

As a comment, the client provided Chipkin with the following kind words:

"I think we are good. Thanks again for the continued support."