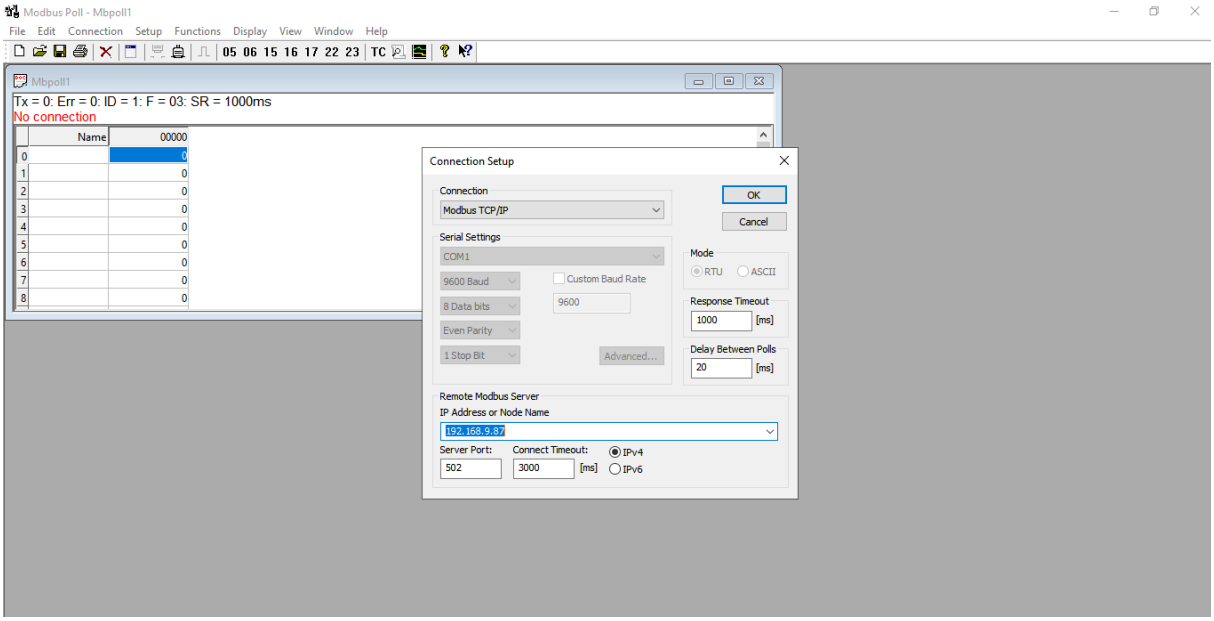
Connection

First make a connection and make sure that the correct connection mode (Modbus TCP), IP and port is set. I've left the other parameters at default.



Read out channel 1 (outlet 1) from outputActualVoltage

We need to check this register in the SPDM model:

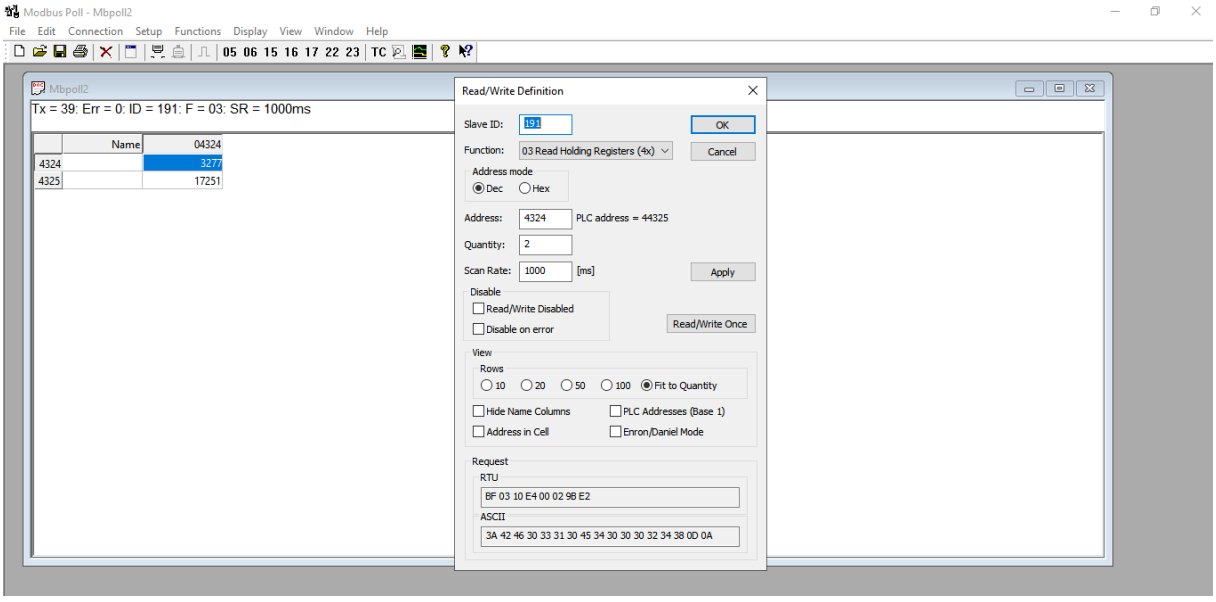
4324	output_measures	**omvoac**	outputActualVoltage	fd	2	27	54	•	ro	ALL

It is a fd with 2 bytes. Now we look up the correct Modbus Type by using the SPDM type from the SPDM model in the table of our Modbus specification.

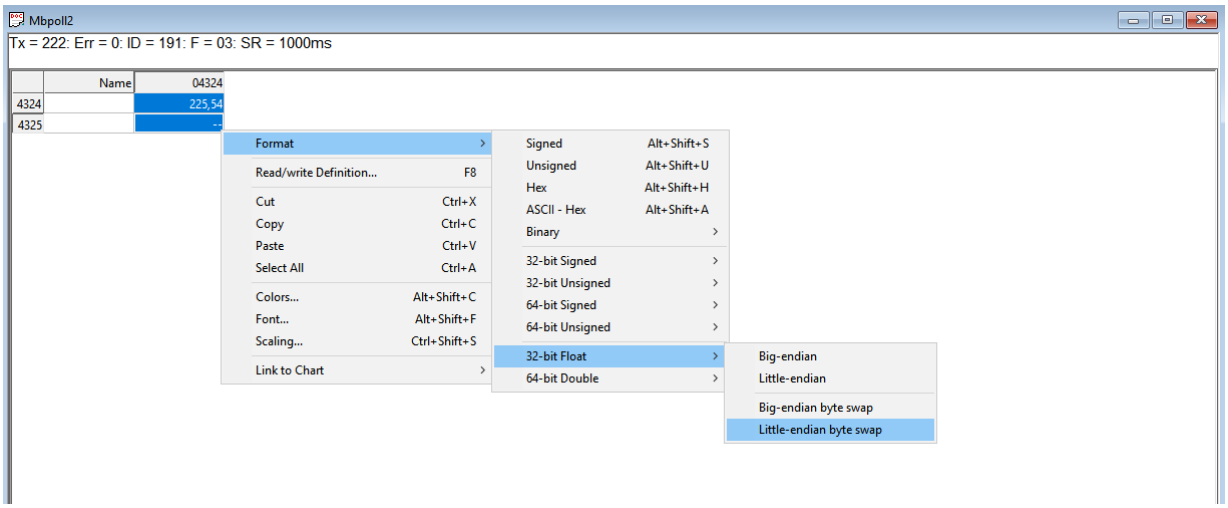
Each data type in SPDM is mapped to Modbus as in the following table.

SPDM Type	Modbus Type	Details
int, 1	1 register, integer	
int, 2	1 register, integer	
int, 3	2 consecutive registers, 32 bit integer*	Both registers must be accessed together.
int, 4	2 consecutive registers, 32 bit integer*	Both registers must be accessed together.
fd, 2	2 consecutive registers, 32 bit float*	Both registers must be accessed together.
ascii, even sized	(size/2) registers, string*	All registers must be accessed together.

So to get the value for a (fd, 2) channel we need to read 2 consecutive registers in Modbus and interpret it as a 32bit float. It will result in the following settings in Modbus Poll. Mind that the Slave ID needs to be set to the unit address of the PDU. Set View to 'fit to quantity' and press OK.



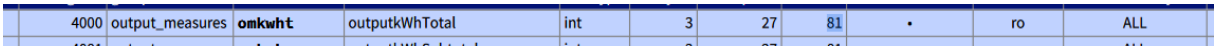
There are now two values in the two that are read correctly, but they are not interpreted yet. As the table in our Modbus specification states, it should be interpreted as a 32bit Float. If we set the format to Little-endian byte swap we find the correct value.



- If you want to read out channel 2; as the first register at 4324 took two consecutive registers, channel two will be available at 4326 (because it uses 2 bytes per channel) and will again take two consecutive registers to read and so on.

Read out channel 1 (outlet 1) from outputkWhTotal

Connect to the PDU with Modbus as in the previous example. Let's lookup outputkWhTotal in the SPDM model.

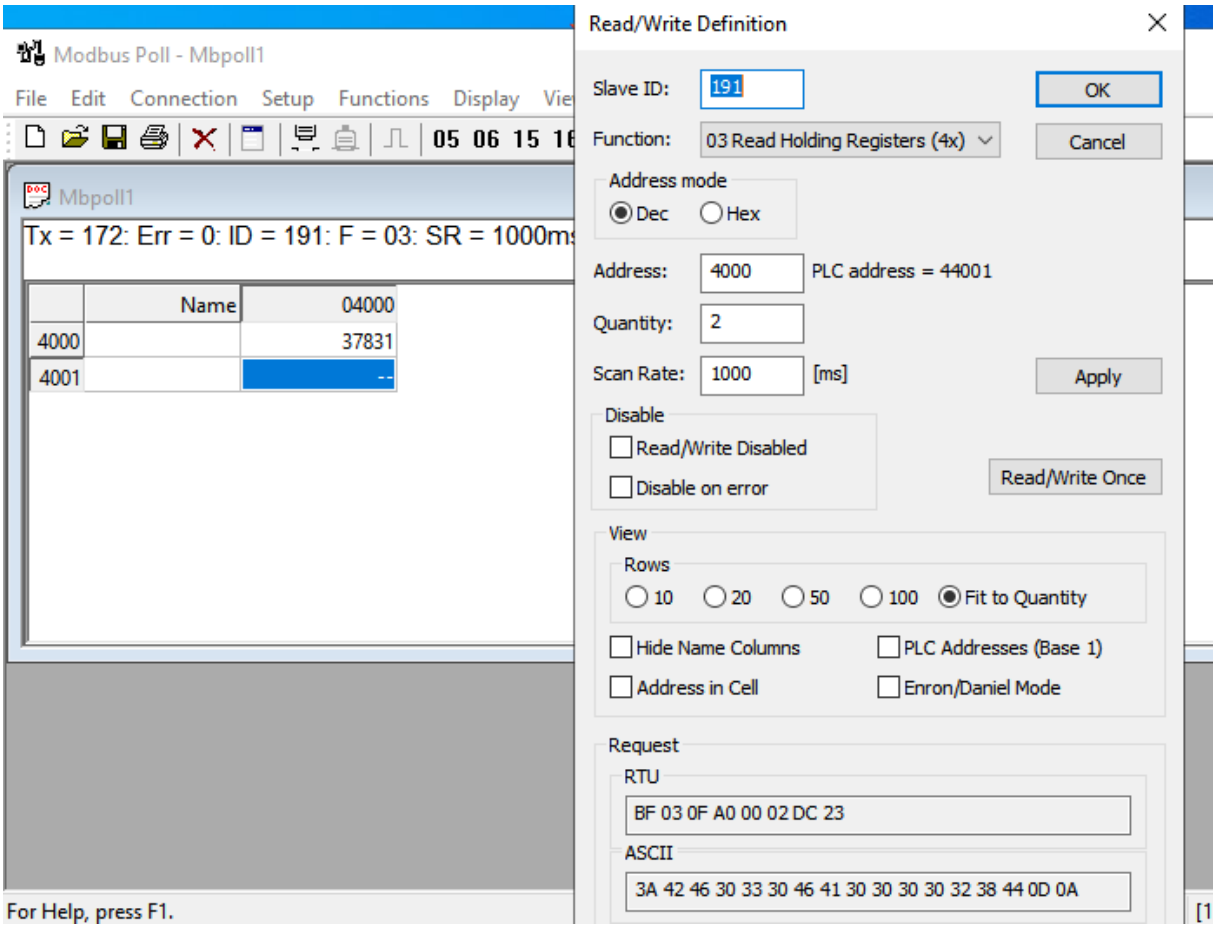


It is an int with 3 bytes as we can see. Now we go to our Modbus specification:

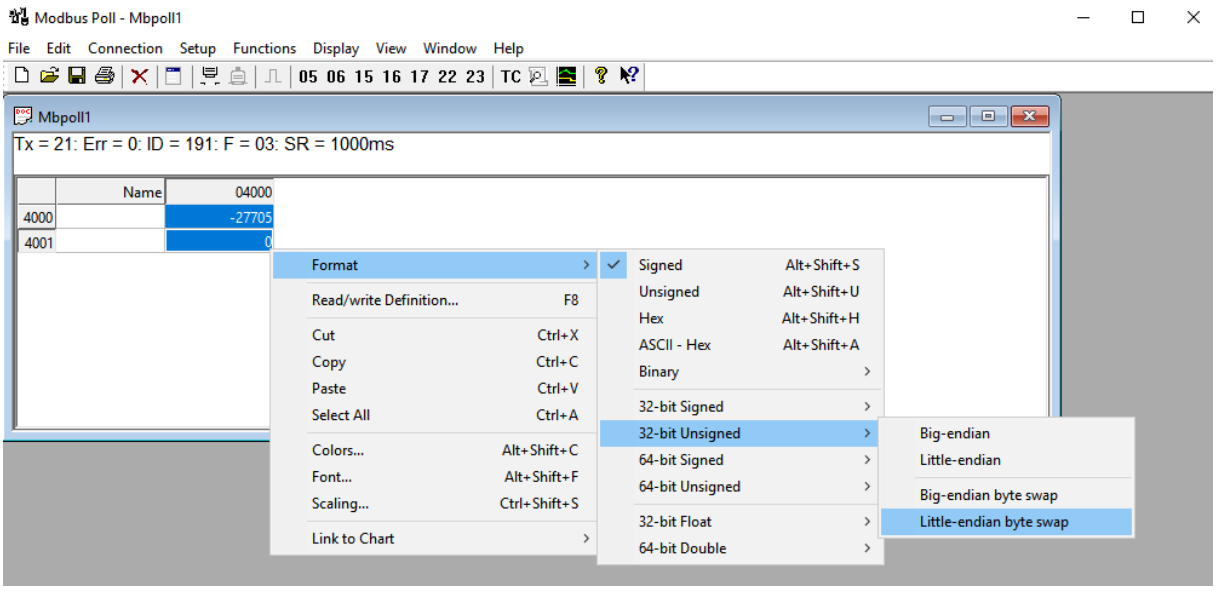
Each data type in SPDM is mapped to Modbus as in the following table.

SPDM Type	Modbus Type	Details
int, 1	1 register, integer	
int, 2	1 register, integer	
int, 3	2 consecutive registers, 32 bit integer*	Both registers must be accessed together.
int, 4	2 consecutive registers, 32 bit integer*	Both registers must be accessed together.
fd, 2	2 consecutive registers, 32 bit float*	Both registers must be accessed together.
ascii, even sized	(size/2) registers, string*	All registers must be accessed together.

So as it is defined in the SPDM model as an (int,3) we need to read it as 2 consecutive registers that need to be interpreted as an 32 bit integer.



We read the value of channel one using the settings above.



We interpret the value as stated in the Modbus specification and get the correct value for this PDU. It took 2 consecutive registers to read channel 1 at 4000, but watch out, channel 2 will be at 4003 (because it uses 3 bytes per channel) but will take again 2 consecutive registers to read as stated by the Modbus specification.

	Name	04000
4000		37831
4001		--

Extended registers

As stated in the Modbus specification and the SPDM model, a register can be of type extended. A measurement for the first 27 outlets starts at 4000 for example, but the registers of the second half of outlets (27) are available starting at 14000. Only outlets that are actually on the PDU will have measurements of course.

- Take a look at the corresponding Excel file, here you can see the range of Modbus registers for the extended 27 registers. Channel 28 can be reached at starting register + 10000 and so on when the channels are read individually.