IMPORTANT NOTE CONCERNING ADDRESSING IN MODBUS NETWORKS

This application note includes important information regarding variations in Modbus address nomenclature. If you experience difficulty getting Modbus communications to work when using the 2500P-ECC1, please read this note carefully.

Nomenclature

The original Modicon PLC's provided coils, discrete inputs, Input registers, and Holding registers as set of memory tables. Each register set could hold up to 9999 elements as shown below.

Modicon Address	Type	Modbus Function Type
1-9999	Read -Write	Discrete Output Coils
10001-19999	Read -Only	Discrete Input Contacts
30001-39999	Read -Only	Analog Input Registers
40001-49999	Read -Write	Analog Output Holding Registers

Modicon users began referring to the memory tables by their address range rather than the memory type.

The Modbus protocol does not use the Modicon PLC model. Instead, it uses a function code (similar to a task code) to designate which memory type to access. For example, there is a function code to read multiple holding registers. Further, Modbus addresses are not limited to 9999 per type. Instead, Modbus supports 65,536 addresses per Memory Type.

Nevertheless, some users continue to refer to the memory types by their Modicon PLC address and some software drivers let the user enter the Modicon nomenclature, converting to an actual Modbus address in the software.

The solution to this is to simply drop the top digit when configuring the server, so that 40010 becomes Holding register address 10. But there is one more wrinkle discussed below.

Modbus Base Address

In the Modbus protocol, the first element of each memory type is 1. The Modbus "on the wire" protocol represents the address as an offset. Thus memory address 1 is represented as 0 in the protocol. Protocol software is supposed to convert between offset and address. Thus, when an address is transmitted a value of 1 is subtracted from the memory address and when it is received a value of 1 is added back. Unfortunately, because documentation was sparse when Modbus began to be used outside of Modicon, some implementations did not make the conversion. While this works fine as long as the same company developed the master and the slave. But there can be problems if the Master and Slave implement it differently as shown in the following example.

Assume that the CTI server operation complies with the open Modbus specification adds 1 to the address transmitted in the protocol. If the Master does not perform the conversion from address to offset, then the address will be off by 1. For example, an address of 100 would be transmitted as 100; when received, the value will be converted to address 101. In this case, the master would need to access

a register that is one less to get the desired address (e.g. read holding register 40099 to obtain the value from HR 100.

A similar problem can occur with master implementations that consider the first “address” is 0, because that is the value transmitted in Modbus. In that case, a request to reading holding register 40000 will get the data from the first Modbus holding register address but it will be referred to as Holding Register address 1. In this case, the solution is recognizing the offset difference at the master. This case is easier to recognize, since one question will reveal it: “What address does the master software use to address the first holding register. “