# USING THE SUPER PLC AS A MODBUS GATEWAY

## B.1 Introduction

A Nano-10, FMD or F-series PLC with r77 firmware or later can be configured to act as the "MODBUS/TCP GATEWAY" for other Modbus serial devices while continuing to function as a Super PLC!

A Modbus/TCP gateway essentially translates a Modbus/TCP command packet it receives from its Ethernet port into a Modbus RTU serial command and sends it out through its serial port (RS232 or RS485). If the Modbus RTU slave sends back a serial response, the gateway will in turn translate the RTU response data back to a Modbus/TCP response packet for transmission to the client via the Ethernet port.

As such the Modbus/TCP gateway enables any non-Ethernet equipped serial Modbus slave device, either TRi or 3rd party, to be directly accessible by a Modbus/TCP client via the Ethernet or the Internet. In addition, the user PLC program does not need to handle the gateway function at all as the CPU performs the translation functions automatically and transparently to the PLC's program.

Best of all, while acting as a gateway, the Nano-10, FMD or F-series PLC program can **simultaneously act as a Modbus master PLC** and read/write to any register inside any of the connected Modbus RTU slaves! The CPU firmware automatically schedules the order of the command/response packets whether they originated from the client or from the PLC itself, so that the response to the Modbus/TCP command from the client is in the correct order and in a timely manner.

## B.2 Application Ideas for Modbus/TCP Gateway

The Modbus/TCP gateway function can be very useful for many large area control systems, such as a building automation system. A master FMD or F-series PLC is linked to many Modbus RTU slave controllers via RS485 bus distributed across an entire building. The master PLC can perform sophisticated control functions since it has read/write access to **ALL** the Modbus RTU slaves it connects to. At the same time, a Modbus TCP client software such as a Building Management System (BMS) can access any register in the master PLC or **ANY** of the slave Modbus RTU controllers directly via the master PLC acting as a gateway.

## B.3 Configuring The PLC As Modbus/TCP Gateway

The command used to define a serial port to become Modbus/TCP Gateway serial port is as follow:

```
SETSYSTEM 12, n
```

Where n = 1, 2 or 3 for COMM1, COMM2 and COMM3 port.

This statement must be run once (could be during start up via 1st.Scan pulse) to configure the serial port #n to be used to send out RTU commands and received RTU responses. If SETSYSTEM 12,n is not run then the Modbus gateway function will be disabled.

Here is how it works: A Modbus/TCP client (such as a SCADA, HMI etc) would send a Modbus/TCP request with a specific 8-bit Modbus slave ID to the gateway PLC. If the ID **matches** the native ID that is assigned to the gateway PLC then the PLC will react normally by sending its own Modbus/TCP response packet back to the client. The PLC will not send any RTU command out of the gateway serial port. This means that the client can access the master PLC's own register as per normal.

However, if the client sends a Modbus/TCP packet with a different ID from that of the gateway PLC's ID, then the gateway PLC will translate the command into a Modbus RTU command and send it out of the gateway serial port #n defined by the "SETSYSTEM 12, n" statement. The PLC will also wait for a response from the RTU slave via the gateway serial port, and if it does receive a response, it will translate it into Modbus/TCP response packet and return it to the client.

**Note:**

1) If the Modbus/RTU slave being addressed does not respond to the RTU command, then the gateway PLC will wait till time-out (default is about 150ms) and then resend the Modbus RTU command and wait for a response again. By default, the gateway PLC will repeat this procedure twice and if it still doesn't receive a response after repeated attempts, it will send back a "TARGET DEVICE FAILED TO RESPOND" error packet back to the Modbus/TCP client. (i.e. It will set the $7^{th}$ bit of the function code to "1" and send back an exception code "0B" hex).

2) The CPU firmware is designed to handle only one Modbus gateway translation for each Modbus/TCP connection per ladder logic scan. This is to ensure that the CPU is not completely bogged down by its Modbus gateway job and will have time to run its own program. If the Modbus/TCP clients attempt to overload the PLC with more requests than it can handle the PLC will frequently return "SLAVE DEVICE BUSY" error response packets back to the client. (i.e. It will set the $7^{th}$ bit of the function code to "1" and send back an exception code "06").

## B.4 Fine-Tuning The Modbus/TCP Gateway Function

It is important to know that every time the PLC runs a Modbus/TCP to Modbus RTU translation, its own program scan time increases due to the need for the CPU to wait for a response from the slave Modbus RTU devices. The delay becomes much more pronounced when a Modbus slave device fails to respond (e.g. not online). The system designer must therefore take this into consideration when designing the system:

1) The Modbus/TCP client should not overload the Modbus gateway with unnecessarily frequent Modbus requests. Whenever the PLC reports a "TARGET DEVICED FAILED TO RESPOND" function, the client program should back off for a while before re-trying communication with the Modbus/RTU slaves again.

2) You can change the number of wait states the gateway will wait for a response from the Modbus/RTU slave by running:

```
SETSYSTEM 1, w
```

Where w = number of serial port wait states for each COMM port as follow:

| Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|
| Ethernet | | COMM3 | | COMM2 | | COMM1 | |

Thus each COMM port could be configured to wait from 0 to 3 wait states for a response. The default settings for w = &H55 (or 01010101 binary) which means each COMM port as well as the Ethernet port uses 1 wait state of about 150ms..

3) You can change the number of retries (default = 2) the gateway will attempt to get a response from the Modbus RTU slave by running:

```
SETSYSTEM 1, n
```

Where n = number of retries for each COMM port as follow:

| Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|
| Ethernet | | COMM3 | | COMM2 | | COMM1 | |

Thus each COMM port could be configured to re-attempt to send the command from 0 to 3 times if it does not get a response from a Modbus RTU slave.

The default settings for n = &HAA (or 10101010 binary) which means each COMM port as well as the Ethernet port will retry twice (making a total of 3 attempts). E.g. If you run SETSYSTEM 2, 0, the gateway will not resend the RTU command on any of its COMM port if does not receive a response on the first attempt. This can help to reduce the CPU scan time if a Modbus RTC slave does not respond.

4) We strongly recommend that the serial port #n that is to be used as a gateway serial port be configured to be “No protocol” by running the SETPROTOCOL n, 10 statement once. This is to ensure that the serial data returned from the RTU slaves will never be interpreted wrongly by the master PLC as incoming commands, which could lead to errors. (The same advise applies to PLC programs that use the serial port to run READMODBUS, WRITEMODBUS, READMB2 and WRITEMB2 commands).

## B.5 Modbus/TCP Gateway Sample Program

Please download the following self-explanatory I-TRiLOGI sample program that demonstrates the new Modbus/TCP Gateway capability via any of the selected COMM port:

http://www.lt-automation.com/modbusgateway.zip