



5100 Modbus TCP Slave & RTU Serial Server Configuration

Blank Page

Table of Contents

Modbus	5
Modbus TCP Slave	5
Modbus Serial RTU	11
Test Summary	13
Assigning IP Address, Subnet Mask, and Gateway Address.....	13

Blank Page

Modbus

The Modbus Protocol is a messaging structure developed by Modicon in 1979. It is used for master-slave/client-server communication between intelligent devices and has become an industry standard. Details of the protocol may be found at the web site www.modbus.org for further details. There are numerous deviations of the Modbus protocol of which the 5100 supports those described within this document. Tools used to test the protocol are available from a number of sources. The 5100 was tested using those available from www.win-tech.com, namely their ModScan32 for RTU Slave testing and ModSim32 for Master.

This document discusses the configuration and testing when using Modbus TCP Slave (server) to interact with the 5100, while at the same time connecting to COM1 with a serial cable running the Modbus RTU Serial protocol. Note on both connections the 5100 operates as a slave (server), returning information as requested by the polling master.

Modbus TCP Slave

The Modbus TCP Slave protocol allows a TCP master to periodically poll the 5100 to collect desired information. The protocol allows for interfaces to such things as coils, analog, register, etc. Since the 5100 is able to access anything via its register interface, only the Holding Register commands are supported; Write Single Register (function code 0x06), Write Multiple Registers (function code 0x10), and Read Holding Registers (0x03).

				Function Codes		
				code	Sub code	(hex)
Data Access	Bit access	Physical Discrete Inputs	Read Input Discrete	02		02
		Internal Bits Or Physical coils	Read Coils	01		01
			Write Single Coil	05		05
			Write Multiple Coils	15		0F
	16 bits access	Physical Input Registers	Read Input Register	04		04
		Internal Registers Or Physical Output Registers	Read Multiple Registers	03		03
			Write Single Register	06		06
			Write Multiple Registers	16		10
			Read/Write Multiple Registers	23		17
			Mask Write Register	22		16
	File record access		Read File record	20	6	14
		Write File record	21	6	15	
Encapsulated Interface			Read Device Identification	43	14	2B

Modbus Function codes from Modbus.org, Modbus Application Protocol Specification, May 8, 2002

You should also note that Modbus registers are 16 bits in width and that of the 5100 are 32 bits, since Modbus is Big Endian, this means reading register 1 in the 5100, the high 16 bits equates to Modbus register 1 and the low 16 bits to Modbus register 2. Modbus register 3 would be the high 16 bits of the 5100 register #2. A maximum of 50 Modbus registers can be read at once, or 25 5100 sequential registers.

As a demonstration of the functionality of the 5100 Modbus TCP/Slave interface this section details the interface of Win-Tech's ModScan32 software and how it applies with regard to our product. As mentioned before, we only support the Holding Register interface. Upon installation of ModScan32 a screen such as Figure 7.3 will appear. Note that the 'Address' field is set to 1, but the display screen starts at 40001. This is Modbus nomenclature. 'Address' of 1 is the same as the upper 16 bits of the 5100 register 1. Note 'Length' is set to 50, the maximum allowable number of Modbus registers in a single read, Device ID is ignored since TCP is point to point.

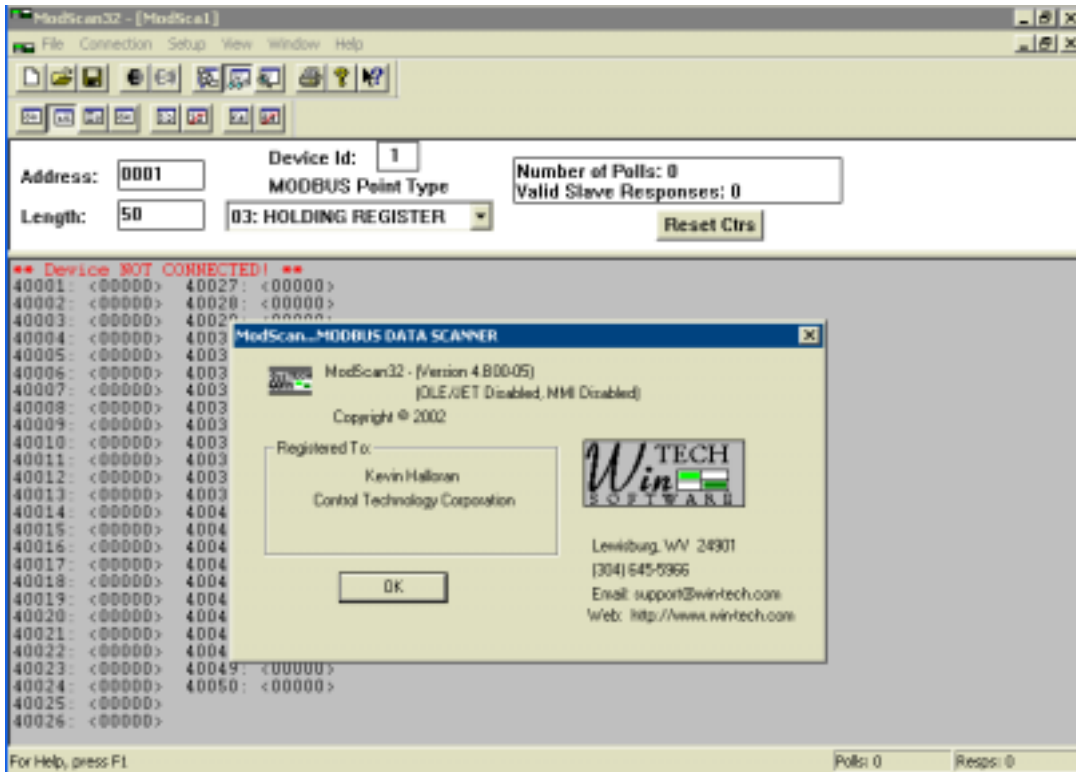


Figure 1.2: ModScan32 Master Scanning Program (only Holding Register supported)

Figure 9.3 shows the setup for an interface to a 5100 with a TCP address of 12.40.53.199 and the Modbus Slave running a server on the standard port of 502:

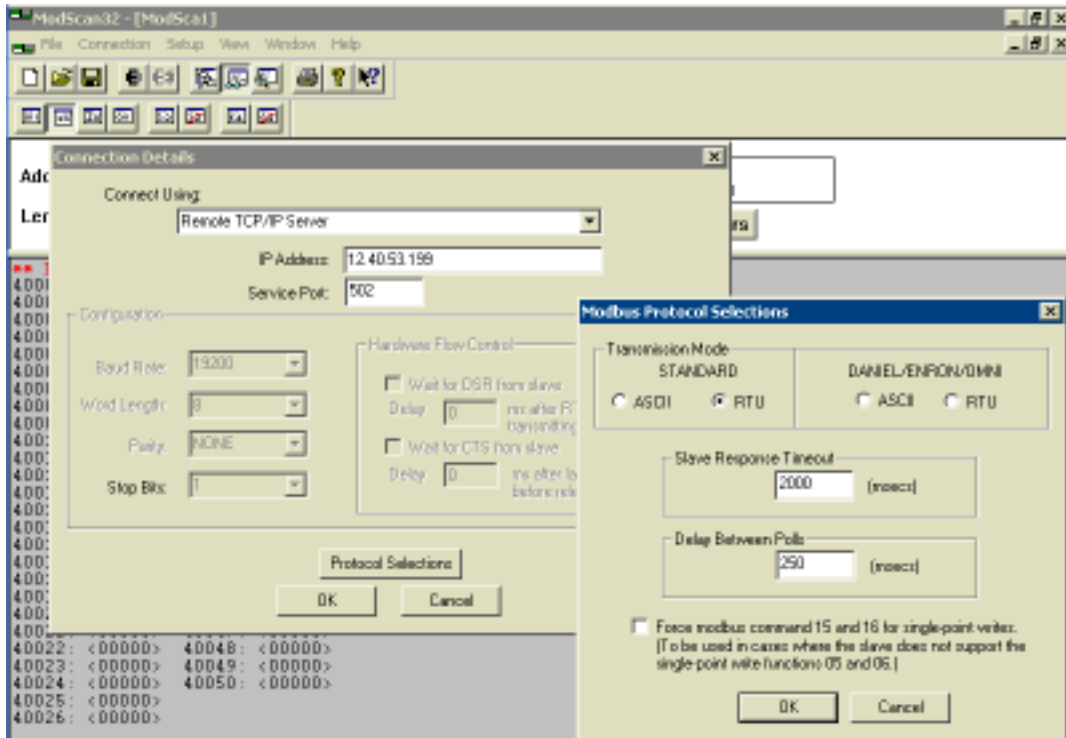


Figure 1.3: ModScan32 Master Scanning Program TCP Connection Setup

In order to do a single register write to a Modbus 16 bit register double click that register. Below shows changing Modbus register 40002 (Address 2) to a value of 5, this would translate to the lower 16 bits of Quickstep register 1. Remember Modbus Address 1 is the upper 16 bits.

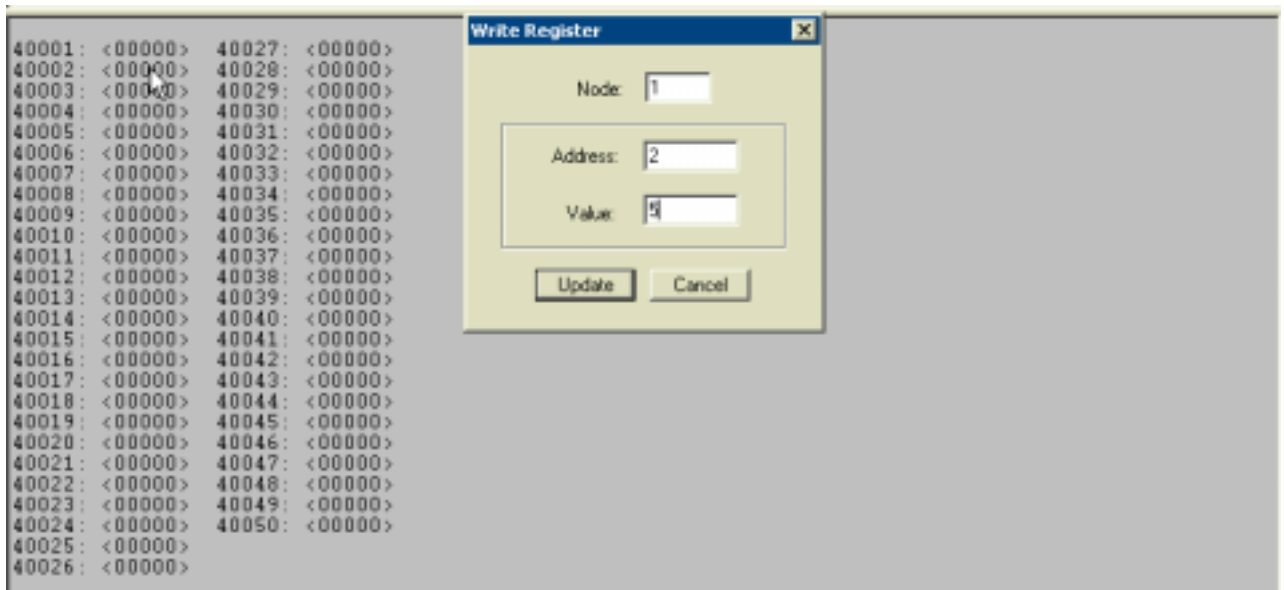


Figure 1.4: Single register write, value 5 to 40002

Changing a number of register all at once is known as a Write Multiple Register access. This can be done using the Extended Access option:

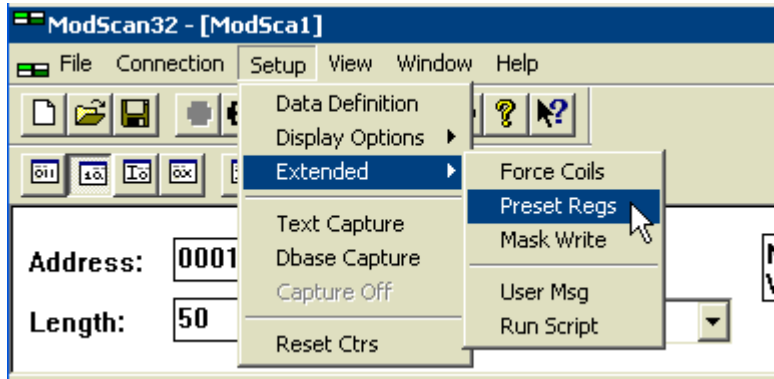


Figure 1.5: Write Multiple register (Preset Regs) selection

The Preset Multiple Registers will appear. Note that in TCP the 5100 ignores any slave or node identifiers since it is a single device and not acting as a gateway. Set the Modbus register you wish to start changes with and the number of registers to change, up to a maximum of that you are viewing:

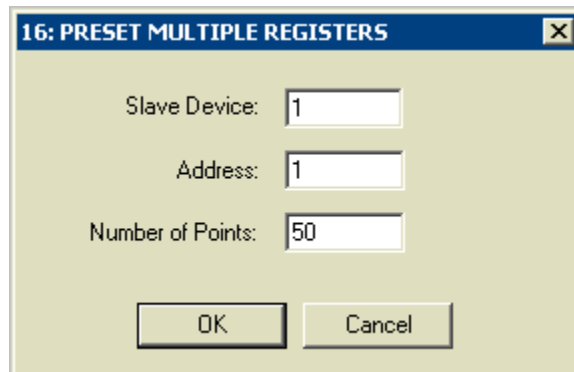


Figure 1.6: Preset Multiple register dialog

In this case we will change Addresses 1 to 10 to sequential numbers 1 to 10:

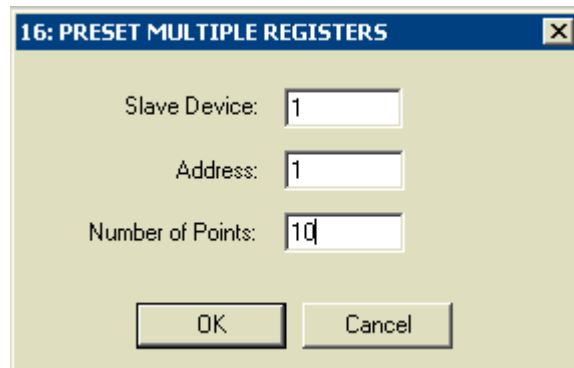


Figure 1.7: Select number of multiple writes to do

As shown below the current register values are displayed in the dialog box.

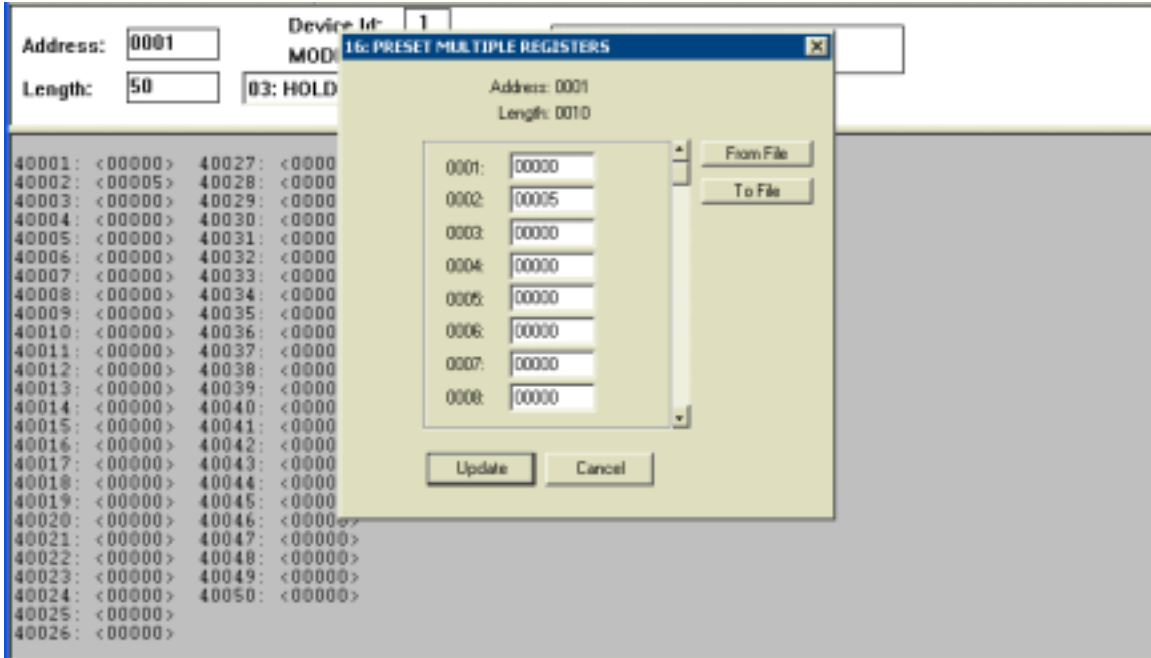


Figure 1.8: Preset Multiple register dialog viewing existing values

Note below that each register value has been changed, also we scrolled down so we could get to register 10. Click Update and note the changed register values from the previous display, 40002 is no longer 5 but now 2.

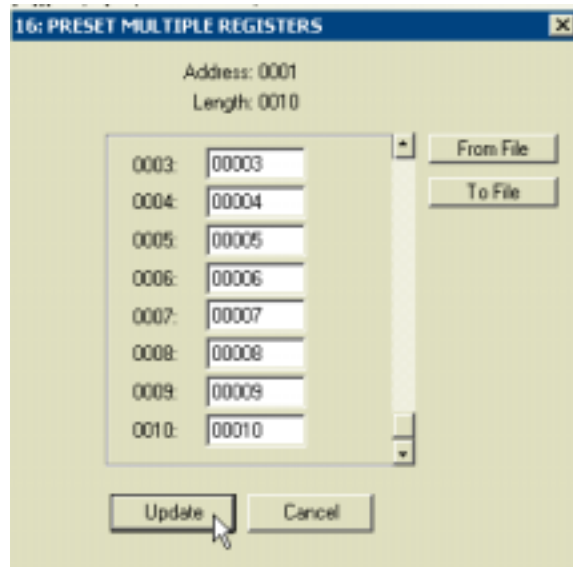


Figure 1.9: Preset Multiple new values entered

Upon clicking the Update key, the new values are written to the 5100 registers and new values read back using the Read Multiple Register command.

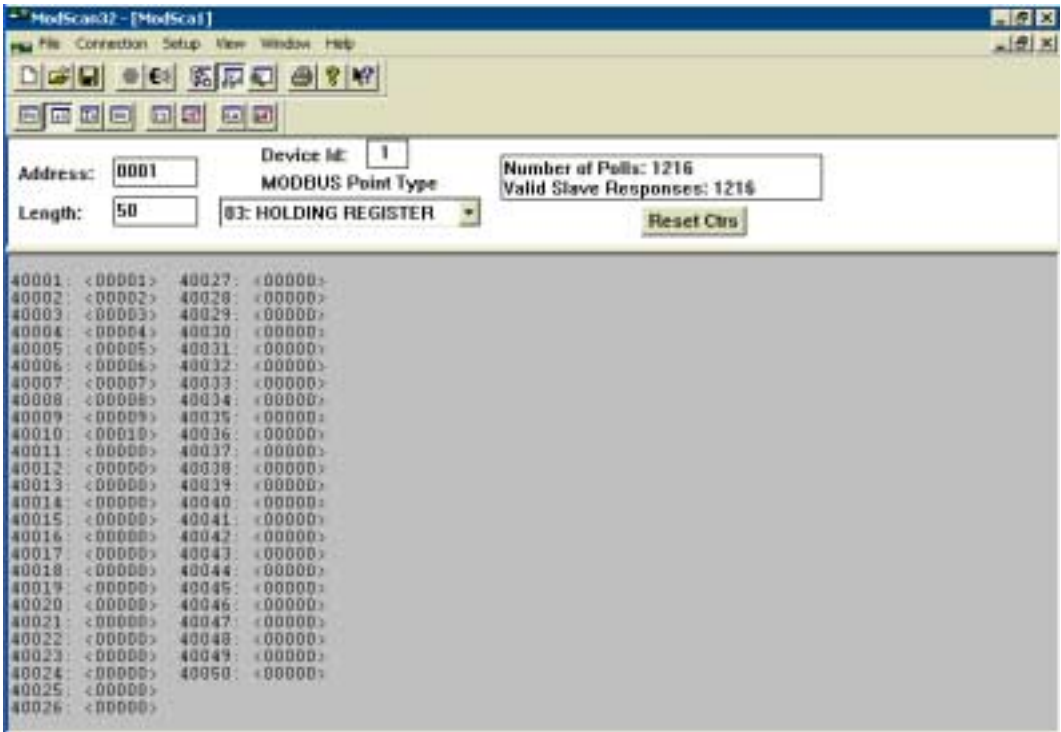


Figure 1.10: New values written and read back, Quickstep registers 1 to 5, Modbus 1 to 10

Should any errors occur a Modbus exception will occur. One such common error is attempting to read too many registers or illegal registers. Below is what is returned if > 50 Modbus registers are attempted:

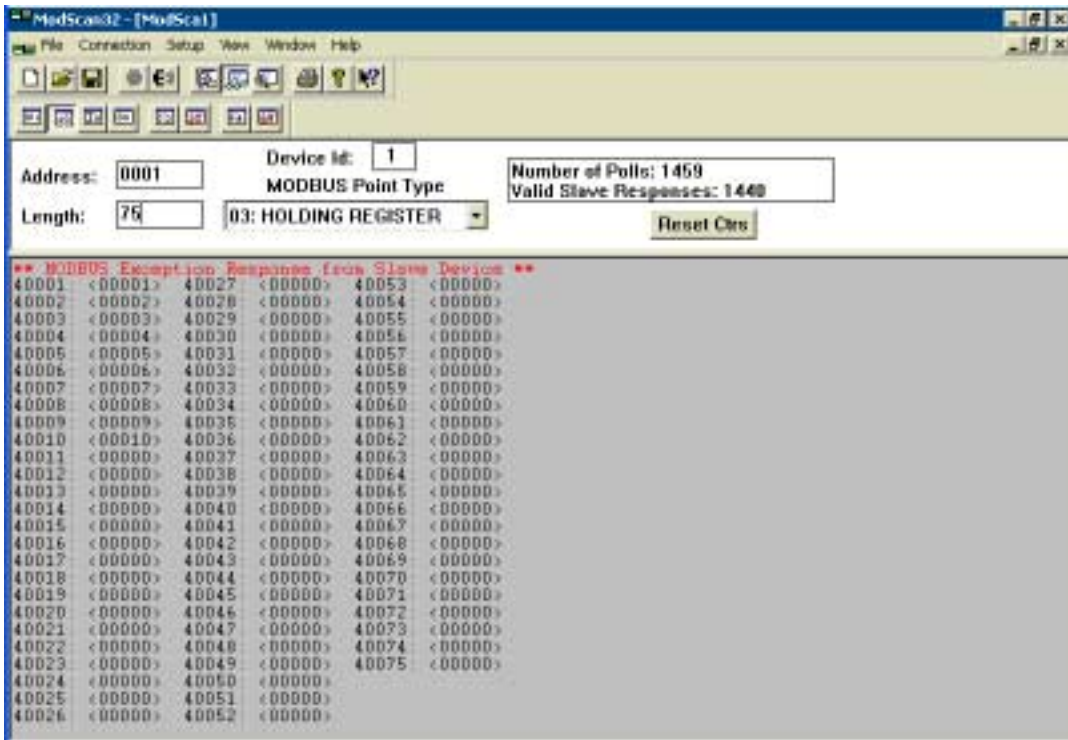


Figure 1.11: Modbus Exception Example > 50 registers

Editing the 75 appropriately will update the error. Below is an example of displaying registers in the 13002 block of the controller. 13002 is the system tic counter, real time clock/date values can also be seen incrementing in other register dynamically. Note that 26003 is the high 16 bits of 13002 and 26004 (13002 X 2) is the base lower 16 bits.

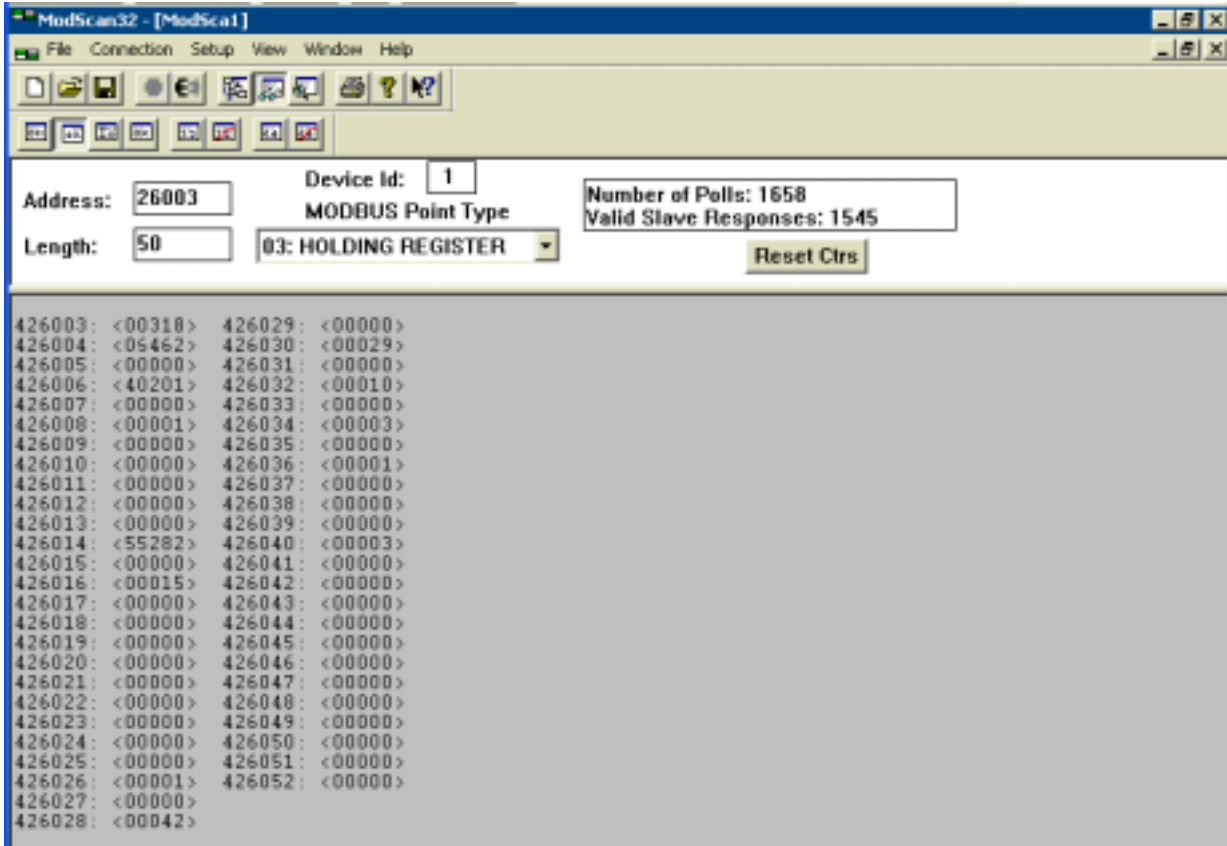


Figure 1.12: Display of 5100 system tic, dynamically updating

Modbus Serial RTU

The Modbus Serial RTU protocol functions exactly like that of Modbus TCP with regards to how to access information and ModScan32 operation (see figure 1.13 for serial port setup versus TCP). There are some key differences since an RS232 connection is used versus a network connection. They are as follows:

1. Only COM1 can be used for the Modbus Serial RTU protocol. COM2 uses an intelligent controller chip which does not currently support the protocol. COM2 support may be added in the future.
2. The virtual TCP communication ports (when interacting with a terminal server) may also be used but only for point to point operations with a single address

- present. In other words the communications traffic of other Modbus nodes should not be present (can be on COM1). This is necessary because Modbus specifies a 3.5 character quiet time between packets and a maximum of 1.5 intercharacter delay during the continuous transmission of a packet data stream. The virtual ports can not guarantee these timing constraints, although from a high level protocol viewpoint, the ports do comply.
3. By default the Modbus protocol is disabled on the serial and virtual ports at power up. To enable the port it must be the active port in the 12000 register and the Modbus Slave address value must be written to register 12320. Note that by default the slave port address is 2 and that any value written as the Modbus slave address will be that used on all serial ports, system wide. Note that writing a value of 0 to 12320 will disable Modbus on that port only and not effect the system wide address.
 4. When Modbus is enabled on a serial port using CTCMON no further communications will be available on that port except with Modbus. In other words you will loose your CTCMON link if talking on the same port.

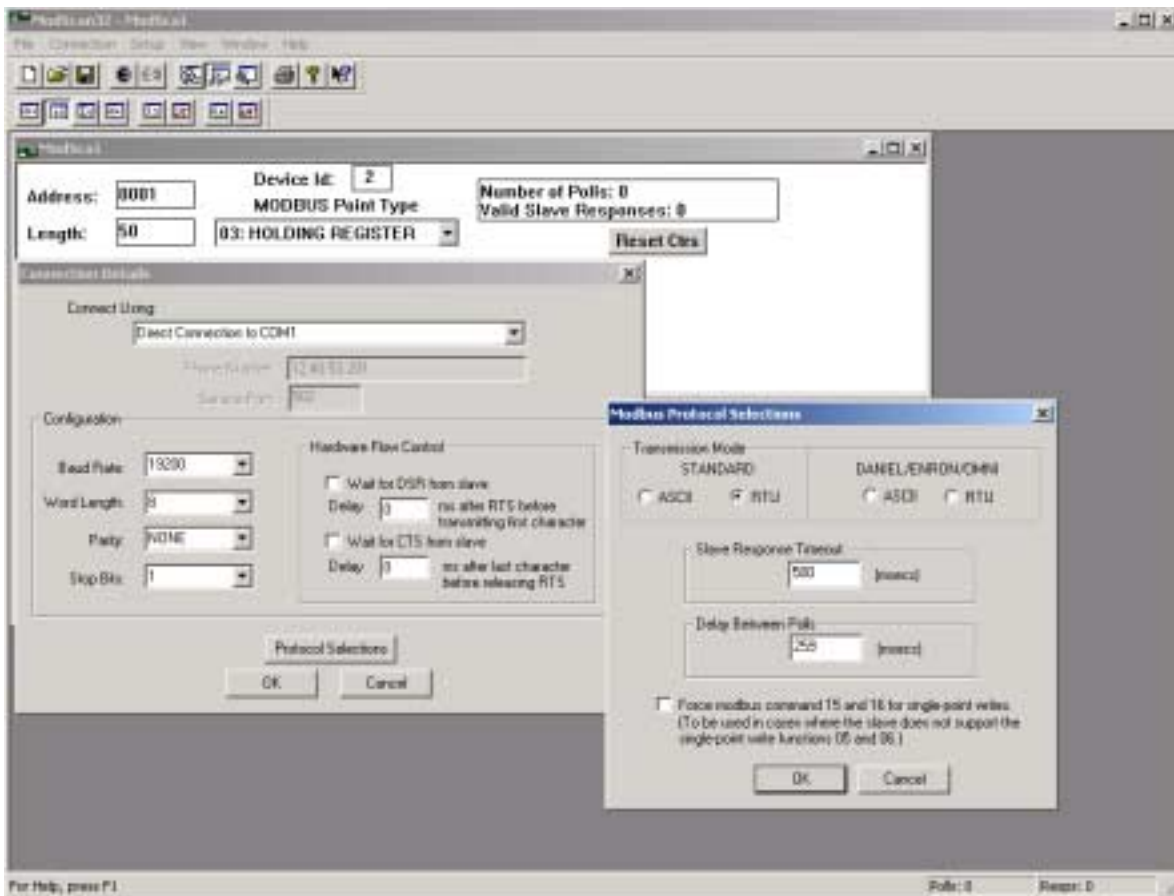


Figure 1.13: ModScan32 Master Scanning Program Serial Connection Setup

Since at power up Modbus is disabled on the COM1 serial port, it must be enabled by writing the desired Modbus address the 5100 is to respond to. This is written to register 12320. Typically done via a Quickstep program, for test purposes it can be also be done

with CTCMON or in the test case presented here, via the Modbus TCP Connection. When done with the TCP connection simply do a write operation to Modbus address 24640 (12320 X 2) of the desire serial RTU address and COM1 will immediately respond to Modbus requests.

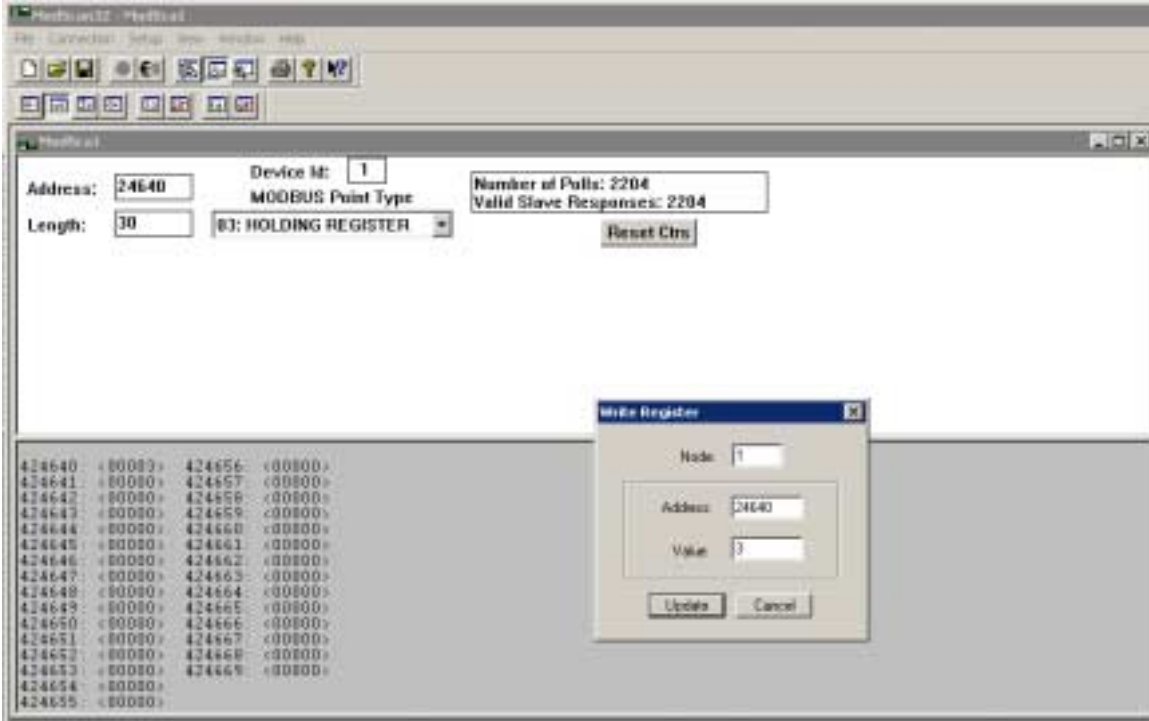


Figure 1.14: ModScan32 Master TCP changing RTU serial address to 3

Above shows device address 3 being written to Modbus register 24620, thereby setting the address for COM1 to respond to. Device ID #1 is connected via TCP.

Test Summary

1. Visit www.win-tech.com and download/install ModScan32 per their instructions.
2. Install CTCMON and set the appropriate IP address, subnet mask, and gateway (if needed) within the 5100 via the serial port. Details are below for the proper register settings:

Assigning IP Address, Subnet Mask, and Gateway Address

To communicate using UDP, TCP/IP, or Modbus/TCP, an IP address and Subnet Mask must be set on the controller. If the controller is to communicate with devices that are not part of the local subnet, then a Gateway Address must also be set. To determine the IP address to be used, consult your IT department.

A. Set the IP Address in 20048-20051

If IP Address is 12.40.53.200:

20048 = 12

20049 = 40

20050 = 53

20051 = 200

B. Set the Subnet Mask in 20064-20067:

If Subnet Mask is 255.255.0.0:

20064 = 255

20065 = 255

20066 = 0

20067 = 0

C. Set the Gateway, 20080-20083, a gateway of 0 (default), disables it.

Gateway 12.40.53.204

20080 = 12

20081 = 40

20082 = 53

20083 = 204

D. After setting the appropriate IP information write a 1 to register 20096 (this may respond with an error, that is normal and can be ignored). This writes the new values to Flash (and deletes the 5100.ini file).

E. Cycle the controller power. Changes will be effective on power up.

3. Invoke ModScan32 and configure as per figure 1.2 and 1.3 for TCP operation.
4. With TCP communications established poke the Serial RTU address into Modbus register 24620. This is done by double clicking that address on the ModScan32 screen whereupon the "Write Register" dialog will appear, figure 1.14. Com1 is now running the RTU Serial protocol and will only respond to the address value entered. Make sure you are set for Holding Registers and the Length field is less than or equal to 50.
5. Invoke another copy of ModScan32 on the same or different computer, with the serial port (COM1) connected to the 5100. Configure as per figure 1.13.
6. Modify values as desired by double clicking the screen. As changes are made via RTU serial they will appear on the TCP side, and vice versa.