Power Automation ™

PMCS Modbus System Emulator

User Manual

Installation and user guide
DEH-xxx

GE Power Management Control System

Notice

The information contained in this document is subject to change without notice.

GE makes no warranty of any kind with regard to this material, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. GE shall not be liable for errors contained herein or incidental consequential damages in connection with the furnishing, performance, or use of this material.

Document Edition - First 06/03

POWER LEADER™ Meter, Electronic Power Meter 3710, and POWER LEADER™ Meter, POWER LEADER Modbus Monitor, POWER LEADER Electronic Power Meter, Electronic Power Meter 3710, Electronic Power Meter 3720, Spectra MicroVersaTrip, Enhanced MicroVersaTrip-C, Enhanced MicroVersaTrip-D, MDP Overcurrent Relay, Multilin Power Quality Meter (PQM), Multilin 239 Motor Protection Relay, Multilin 269 Plus Motor Management Relay, Multilin SR469 Motor Management Relay, Multilin SR489 Generator Management Relay, Multilin 565 Feeder Management Relay, Multilin 735 Feeder Relay, Multilin SR745 Transformer Management Relay, Multilin SR750 Feeder Management Relay, GE Fanuc Series 90/30 PLC, GE Fanuc Series 90/70 PLC and Spectra Electronic Control Module are products of GE.

Microsoft, Microsoft Access, Microsoft Excel, Microsoft PowerPoint, Microsoft Visual Basic, and MS-DOS are registered trademarks, and Windows NT, W2K and XP is a trademark of Microsoft Corporation.

Contents

Chapter 1
Welcome

Welcome to the GE PMCS Modbus Emulator, a second generation, real-time GE device simulator designed to simultaneously mimic the entire register map of any number of GE, or other, Modbus based devices. The Emulator is hosted on a separate PC and will simulate dynamic device values (Voltage, current, etc) as well as static values (firmware rev, etc). It will allow a user to create device 'scenarios' or value profiles for a specific device or a class of devices to simulate real time values and alarms. With the Emulator an integrator can pre-qualify virtually 100% of a SCADA application before installation at a customer's site.

I. Some of the major features:

Easy of Configuration: Importation of an existing PMCS GE32MODB or GE32MTCP server configuration(s) allows quick setup. A configuration tool is included for use with non-GE PMCS based system.

Preconfigured Device Profiles: All devices that are tightly integrated into the PMCS servers are pre-configured with a basic device value profile that can be easily modified for a users particular application.

Generic Devices: Most other Modbus based device (PLC's, etc) can be simulated also. The user must create a device profile file similar to a preconfigured device profile. Please see the "Generic Device" heading in Section 4 of this manual for specifics. Also reference the "Modbus RTU/IP" specs in the next Chapter 2.

Changing Dynamic values: All individual real time values such as amps, volts and energy can easily be configured to increment/decrement by a 'step' value between a min/max range

in a simple sawtooth profile. Profiles can be configured by device type(default) or individual device.

Manual value change: Any individual mnemonic (register mapped to a name) for a device can be changed in real time. This allows quick verification of event/alarm/status functionality of the SCADA. The Emulator also supports Modbus write codes, allows verification of SCADA commands/writes to a particular device register.

PMCS Device Event Codes: A user can create a device based 'event' that a will be seen by the PMCS alarm/event server. This feature is only available for Modbus Concentrator based devices that support this function.

DDE interface: Supports simple DDE pokes/reads to set a mnemonic of a specific device. This allows a user to create a custom data program (with VB or similar) to create complex, custom device data profiles or data driven events

I/O traffic display: Allows a user to quickly visually verify data requests/answers for any particular device. The latest data request is time stamped for ease of reference.

Realistic device update rates: While it is impossible to exactly mimic every devices specific Modbus query/response rates the Emulator allows the integrator to get a realistic estimate of how a real system will perform, allows any system changes or alterations to be identified early. Note: Modbus TCP/IP based devices may require additional configuration, see limitations below.

Modbus Data Error simulation: The user can 'inject' data errors into the Modbus traffic such as data timeouts, 'dead' devices and Modbus exception codes to see how the system responds

II. Limitations of the Emulator:

Logic Functions: The Emulator does not reproduce any device 'logic' functions, the Emulator's 'intelligence' is limited to its ability to change values between a registers min/max range. However with proper configuration of a registers' value range and step function this limitation can be mitigated. Example: Energy values can incremented realistically if the user can create the correct energy range and step value. Also, the user can create virtually any required functionality using VB scripting and the Emulators' DDE poke/request interface.

Waveforms: Currently the Emulator cannot reproduce waveforms. Any waveform retrieval attempts will fail, however it is theoretically possible to manually configure the waveform registers of the device with waveform samplings. This can be done by editing the devices csv profile or by 'poking' the values via the Emulators Modbus client interface or using the Emulators DDE interface.

Data Logs: The Emulator does not provide built-in support for any devices' automatic data logging capability; however the user may be able to input the data log manually into the device's CSV-configuration file or via DDE pokes.

Modbus TCP/IP based devices: The Emulator supports ModbusTCP/IP devices however due to the Modbus TCP/IP standard all Modbus TCP/IP based devices are configured on the PCs' single IP address with the default IP port 502. Since the Emulator can only listen to the Emulator's PC IP address this creates a situation where all Modbus TCP/IP based devices configured in the Emulator must be configured with the same IP address but different Modbus addresses. This forces all data requests into a 'serial' RTU fashion for all devices. The net effect will be unrealistic data update/refresh rates for ModbusIP based devices.

To accurately simulate Modbus TCP/IP devices the integrator should reconfigure each device in the Emulator as a RTU serial device on a dedicated Comport in the GE32MODB program. Each Individual IP address of each device is 'remapped' to its own single Comport. This allows a max of 254 individual IP address's to be remapped and will yield realistic update times. Please see

Chapter 4 for more details and other 'work around' solutions.

Also, only 1 Modbus master connection is allowed per IP port and polling MUST be done in a query/response pattern (the same format as Modbus RTU). Multiple, simultaneous queries on the same port are not supported with the current version of the server.

GE Modbus Concentrator based devices: The Comnet update cycle of the concentrator is not included in the Emulator so Concentrator based devices updates from the Emulator will be mostly likely ***twice*** as fast as real Concentrator based devices. Please take this into account when designing a system containing these devices.

Chapter 2
Concept and Specs

The PMCS Modbus system Emulator is a PC based software program that can simulate all Modbus values for the PMCS tightly integrated Modbus devices as well as most generic Modbus devices.

A standard PMCS system typically has the following configuration:

- The PMCS host PC running the SCADA and PMCS host software(DDE/OPC servers) and the NativeCom Comport-to-IP remapper (if using Modbus RTU devices)
- One or more GE Ethernet Gateways – for converting serial RS-485 Modbus networks to Ethernet, each network is mapped as a Comport at the host PC via the NativeCom remapper software
- GE Modbus meters/relays/breakers/PLC's or generic Modbus devices using RTU or IP
- An 10 or 10/100 Ethernet LAN

Figure 1 below illustrates a standard PMCS system:

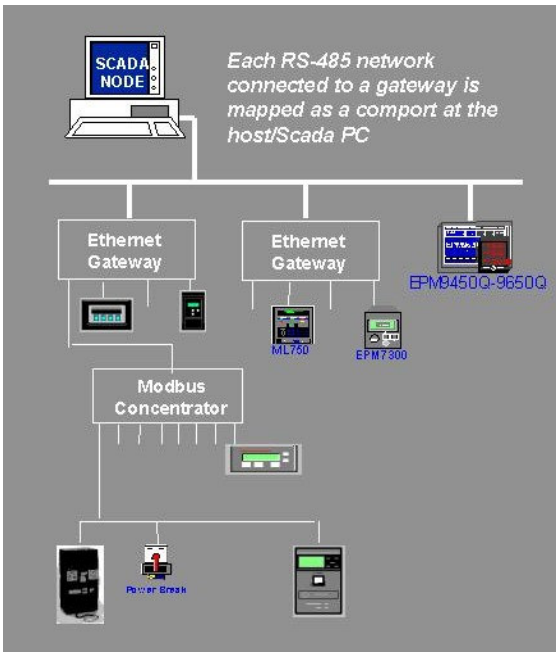


Figure 1. Typical PMCS system.

The PMCS Modbus System Emulator replaces all the Modbus devices (serial and IP based), the Ethernet gateways and the RS-485 networks in a typical system. The Emulator is preconfigured with all the Modbus register maps for standard PMCS devices and has the ability to simulate all the Modbus registers in those devices. In addition most 3rd party Modbus devices can be supported also. The Emulator uses the same topic and port configuration files from the GE32MODB and GE32MTCP servers thus allowing the user to quickly configure the Emulator for any size system. The Emulator is also scalable allowing it to be run multiple instances (same license) and on multiple PC's simultaneously (requires additional licenses).

Figure 2 below illustrates how an Emulator system replaces a standard PMCS system:

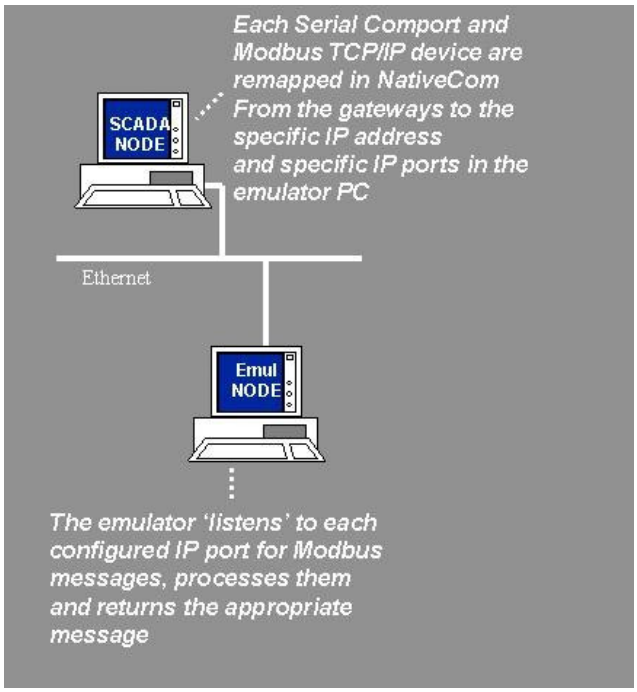


Figure 2. Typical Emulator system.

In addition if the Modbus Master PC has two serial ports available it is possible to run the Emulator on the same PC by connecting the ports together with a serial cable. The Master polls on one port and the Emulator listens on the other.

I. Requirements:

- Emulator PC with WinNT 4.0 sp5, Win2000/sp3 and WinXP/sp1 with at least the min requirements for the OS. Note: Emulator system resource utilization is dependent upon the size of the system being emulated and the device types being emulated. The Emulator PC used during testing was 1.2 GHz with 512Meg ram simulating a 300-Modbus RTU device system with a dozen or more dynamic registers per device. CPU usage averaged 55%. CPU utilization is the single highest used resource on the Emulator PC, therefore the faster the PC the better the Emulator performance.

- Each device or device type being emulated must have a csv 'profile' file created and configured for it. The Emulator requires a profile of registers, register names, initial values and min/max values (for dynamic registers). Without this profile all returned Modbus values will be zero.

- A second computer configured with PMCS (see requirements for PMCS) or similar Modbus master, NativeCOM v5.02 (or later) and the required NativeCom Generic Port Server license (included with the purchase of the Emulator).

- All hardware/software required to network the two PC's together for TCP/IP

- No network 'firewalls' between the PC's that block IP ports 502 and 9000~9256(default config, ports assignments can be customized)

II. Modbus RTU/IP Specs:

The Emulator supports all the devices, functions codes and register ranges of the GE PMCS Modbus servers:

Modbus types: RTU (all baud rates and interfaces) and TCP/IP (any port, default 504)
Modbus TCP/IP hosts: Only 1 ModbusIP master connection allowed per IP port, polling MUST be in a query/response pattern (the same pattern as Modbus RTU). Multiple, simultaneous queries on the same port are **not** supported with the current version of the PMCS server or the PMCS Emulator.
Comports supported: Max 256 available to Windows OS
Modbus Function codes: 01, 02, 03, 04, 05, 06, 15, 16
Modbus address ranges: 1 ~ 247
ModbusIP port Ranges supported: 0 ~ 65535, default Comport mapping starts at IP port 9000 (i.e. 9000+ Com 10 = 9010)
Modbus Exception responses: The Emulator does not support *automatic* generation of Modbus exception codes however it does support manual generation of all codes using the Emulator 'Control Panel'.
Read/Write: Full write support of all data types and formats for a PCMS system with the exception of 'Array' writes. Note: upon

shutdown the Emulator will not retain any writes to an 'Emulator' based device.
Individual Modbus Register size: 16bit, binary data
Register Numbering: Absolute with no offset, 0000 = 0000
Register Types: Unsigned 16 bit, Signed 16bit(I), Signed 32bit(L), IEEE Floating 32bit(F), ASCII string with 250 characters max(Sxx); See Appendix A for more information.
Special formats supported:
Individual bit read/writes(DX-x)
Array reads with values separated by a space with a max of 16bit registers(Axx).
Modbus Query Response Timing: Global setting starting at 0ms, default setting is 30ms
Dynamic register values: Any register configured with a Mnemonic can be configured for Dynamic updates. Dynamic updates force the register value to increment by a defined decimal 'step' amount at the 'Dynamic data update rate'.
Dynamic data update rate: Global for all registers set to be dynamic, range is 0 to 9999 seconds (0 = off for all updates, default = 2 seconds). All updates are simulatenous.
DDE interface: Any register configured with a Mnemonic can be configured for DDE pokes/requests. This allows the user to create more complex data scenarios, if required. Default ="N"
Ethernet TCP/IP interface: Uses Raw IP encapsulation (default RTU configuration) and the ModbusIP standard (for ModbusIP).
Serial Interface: Dictated by the PC serial interface hardware.

Quick Steps to Start Using the Emulator

The below steps are only intended to give a quick overview for using two PC's. More detailed setup instructions are given in later chapters of this manual.

1. Install the Emulator on a PC
2. Install and configure the Modbus RTU/IP Master on a second PC, i.e. for RTU use the GE32MODB and for IP use the GE32MTCP servers. Export the config.txt file from the servers or use the EmulatorConfig tool to create one. Copy the file to the Emulator PC
3. Install and license NativeCom on the PC running the Modbus RTU master, i.e. GE32MODB. NativeCom is not

required for ModbusIP masters, i.e. GE32MTCP. Configure NativeCom for the Comports used by the GE32MODB (or similar Modbus RTU master) and 'point' NativeCom to the Emulator PC's IP address.

4. Verify the Emulator PC and Modbus master PC are networked together and each PC can 'ping' the other.

5. Load the config.txt file into the Emulator and hit the run button. Start the Modbus Master.

Possible Emulator configurations

The Emulator can be configured for a variety of Modbus systems. Here are some examples.

- Emulator PC to second PC based Modbus RTU master (i.e. GE32MODB server) using NativeCOM over TCP/IP network (Ethernet typical) using IP 'encapsulation' with up to 256 RTU serial 'networks' supported as IP ports.

- Emulator PC to second PC based Modus RTU master on a RS232/485/422 network via serial Comports on both PC's. Limited to number of Comports on both PCs. Straight RS232 config requires null modem cable or adapter. This will also work with the Modbus master being a PLC or similar.

- Both Emulator and Modbus RTU master on same PC having two available Comports. Configure the Modbus Master for Serial Comport(X) and Emulator on Serial Comport(X+1) with an RS232 cable connecting both ports. A null modem cable is required.

- Emulator PC to second PC based ModbusIP master (i.e. GE32MTCP server) over TCP/IP network (Ethernet typical).

- Emulator PC over TCP/IP network to ModbusIP master being a PLC or similar.

Chapter 3
Installation and Configuration

I. PMCS/Modbus Master PC installation/Config:

For Modbus RTU masters utilizing more than 1 comports on the hosting PC a copy of NativeCom must be installed with the 'Generic Port Server Option' licensed. This license is included with the purchase of the Emulator.

The following steps must be applied on the PMCS/Modbus host PC:

1. Install the PMCS software (see installation procedures for your PMCS software) or other Modbus master.
2. Install NativeCOM v5.02 (or later). Available on the PMCS CD at x:\Other\Ethernet_gateway\NativeCom. It is also on the Emulator CD at x:\NativeCom
3. Configure NativeCom Comport mapping to the host computer. You must have the NativeCom Generic Port server license. This license is included in the purchase of the Emulator.

The PMCS software and SCADA system is configured in the standard PMCS fashion; however NativeCom is configured in a slightly different way. In a standard configuration the user selects/enters a GE Ethernet gateway IP address and maps comports to gateway serial ports. However, for the Emulator, the Emulator PC IP replaces the Ethernet gateway IP address and comports are mapped to a specific IP port on the Emulator PC IP. Note: NativeCom is only used with the GE32MODB server or other PC based serial Modbus RTU Master.

NativeCom Emulator Configuration Steps

1. Open the NativeCom configuration window from "Start>Programs>NativeCom> NativeCOM Configuration Utility". The NativeCom UI should appear.

1. Select “Add Multiple Ports’. The add multiple ports window should appear

2. Select “Add Generic Device”. If this is the first time using this feature you will be prompted to enter the license key to enable the generic port server feature. This key is included with the purchase of the Emulator.

3. Enter the network name or IP address of the Emulator PC. If you enter the IP address and the PMCS PC can resolve the Emulator PC’s network name you may see NativeCom convert the IP address to the network name.

4. Select the correct IP address/Hostname in the “Step 1” box

5. Enter a ‘Port Name’ in “Step 2”. This name is strictly for documentation and has no operational effect but a name must be entered. Example: “EmPorts”.

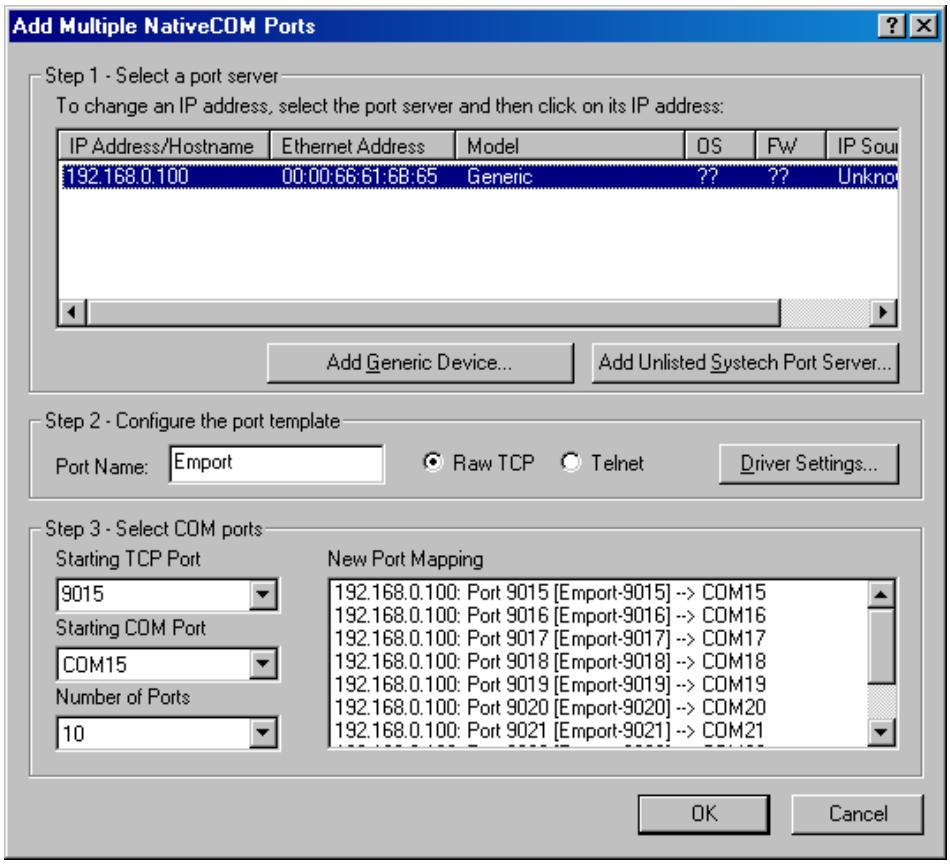
6. In “Step 2” there are two radio buttons next to the Port Name box. You **MUST** select “Raw TCP”. The Emulator only works with “Raw TCP”.

7. In “Step 3” select the “Starting COM Port” number. This will be the start of the Comport range the PC can access via NativeCom.

8. In “Step 3” in “Starting Port Server Port” take the “Starting Com Port” number entered above and add 9000 to it. For example, starting Com Port = 15 means the “Starting Port Server Port” = 9015

9. In “Step 3” in “Number of Ports” select the number of ports you need to create.

10. You should have a display similar to the one below



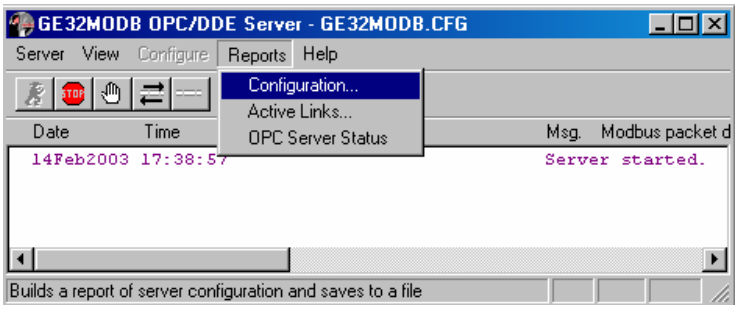
11. Press the “OK” button
12. After NativeCom creates the Comports the PC MUST be rebooted

Exporting the PMCS OPC/DDE Server Configuration

The PMCS server configuration is used by the Emulator to configure its internal ‘devices’ Modbus Modbus configuration. It contains register maps, types, function codes etc.

Please perform the following steps:

1. Configure the GE32MODB or GE32MTCP servers as normal.
2. Once the server is running go to menu item Reports>Configuration.



A “Save File” window will appear prompting the user to save the ‘config.txt’ file. This is file contains the entire server comport, device and device type settings and configuration. It is this file that is used to configure the Modbus Emulator. Save it directly to the Emulator directory on the Emulator PC (via Ethernet) or a location that can be retrieved from the Emulator PC. Note the Emulator does not require the file to be named “Config.txt”. This is just the default name.

II. Emulator Installation and Licensing:

Installation

1. Insert the PMCS Installation CD into the CD-ROM drive.
2. Click the Start button from the Windows Task Bar and select Run.
3. In the Run dialog box, enter “X:\setup” (where X is the appropriate CD-ROM drive designation).
4. The GE PMCS Emulator Setup Screen should appear. Once you have read the screen select “Next”.
5. After reading each setup screen select “Next”. When the setup prompts for the installation direction please enter the drive and directory that the current, if any, PMCS software resided. The Emulator default installation directory is “C:\GE_PMCS\EMULATOR”.
6. At the program folder window please choose the “GE_PMCS” program folder.

7. If applicable, accept the remaining defaults to complete the installation.

Uninstalling the Emulator

The Emulator comes with a complete uninstall feature that can be accessed from the Windows Control Panel “Add/Remove” programs feature. Any residual components can be safely deleted.

Licensing

General Electric Industrial Systems has added security technology that requires your PMCS Emulator software to be licensed in order to run past the trial period. A separate license is required for each computer that will receive the PMCS Emulator. Each eLicense Order ID that is sent with your order corresponds to one license.

You license your software as part of the automated PMCS installation process. When you launch the installation process the first screen that will appear will be your eLicense Option screen (see below). This screen allows you to either trial or fully license your PMCS software.

Trial/Demo Period

You have the option to trial the PMCS Emulator for an unlimited number of 10 minute intervals. The trial selection is found on first license screen that appears when the Emulator is launched. All functionality of the Emulator is available during the trial. Basic settings and all configurations made for the software will be maintained from one use to the next. A single trial is registered to each PC and may not be uninstalled and re-installed for another trial.

Licensing via the Web

Emulator licensing ***must*** be done via the Internet. This option will automatically connect you to General Electric Industrial Systems’ eLicense distribution system. Once connected, an automated process will retrieve your unique PMCS eLicense key.

Selecting the Web License option from the License Option brings you to the Web License Screen. This screen requires you to enter your eLicense Order ID.

This information is contained on the Order Confirmation Sheet that accompanied your PMCS Emulator software or it can be obtained from your Purchase Confirmation e-mail. Enter the Order ID and click on Get License to initiate the Web connection to General Electric Industrial Systems' eLicense distribution system.

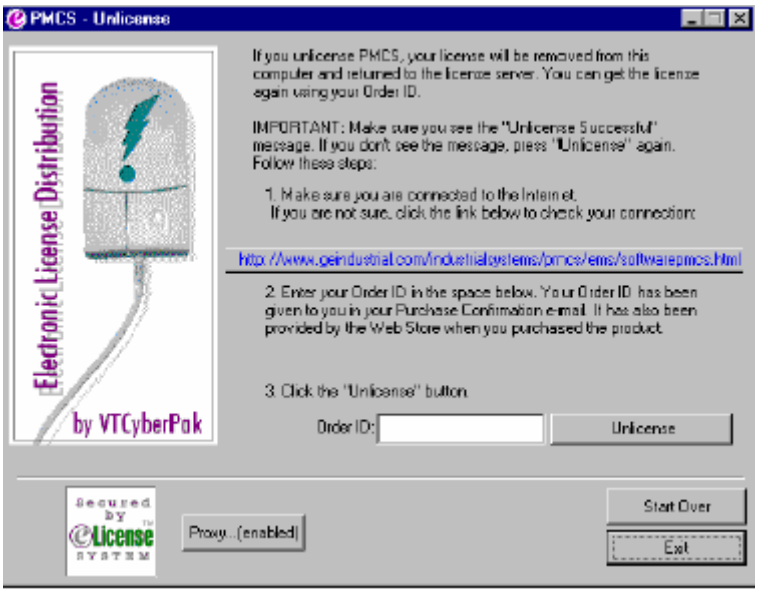
License by Phone

If you experience an online licensing issue please contact GE licensing The Toll-Free Assistance Line (888) - GE4-SERV or for international callers please use (540)-378-888. This system can be used to retrieve your unique eLicense key from a General Electric Industrial Systems customer support representative (please select Technical Support, then Software Licensing from the phone menu). You will be asked over the phone for your System Identifier, Version ID, and Order ID and in return you will be given your eLicense key. The eLicense Order ID from the License Confirmation sheet and the eLicense key given over the phone are required to Install the eLicense.

Similar to the trial, each eLicense is registered to the computer In which case, the eLicense Order ID can only be used for one PMCS computer. The software must be unlicensed to be re-licensed with the same eLicense Order ID on another computer. Select Launch from your License Installation Success Screen to continue installing PMCS.

Un-elicensing the Emulator

You must unlicense your software to use the same single license eLicense Order ID for another computer – you must have an internet connection and the Emulator license must have been installed via the Internet. To retrieve your license for use on another PMCS installation, simply right click on any PMCS executable on your desktop and select unlicense.



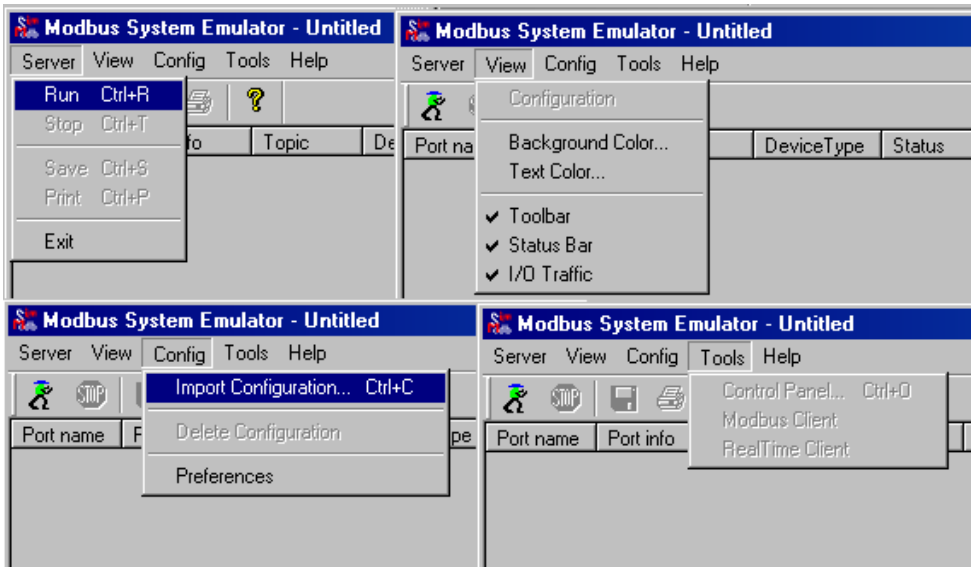
The screen will request the eLicense Order ID to be entered. After entering it select the “UnLicense” button. A screen will appear to indicate that you have successfully un-eLicensed and you may now use the eLicense Order ID for use on another PMCS installation.

Please note that you *can not* retrieve an eLicense for re-use through the unlicense process past 3 years of initial Licensing.

III. Configuring the Emulator:

Exploring Menus and Commands

The menu features most of the same functions as the Toolbar



Definitions for all the menu items follow;

Server:

Configuration – View the loaded device configuration
Run – Starts the Emulator once the config.txt file has been imported
Stop – Stops the Emulator
Save – Saves the contents of the I/O screen as a txt file
Print – Prints the current I/O display
Exit – Self-explanatory

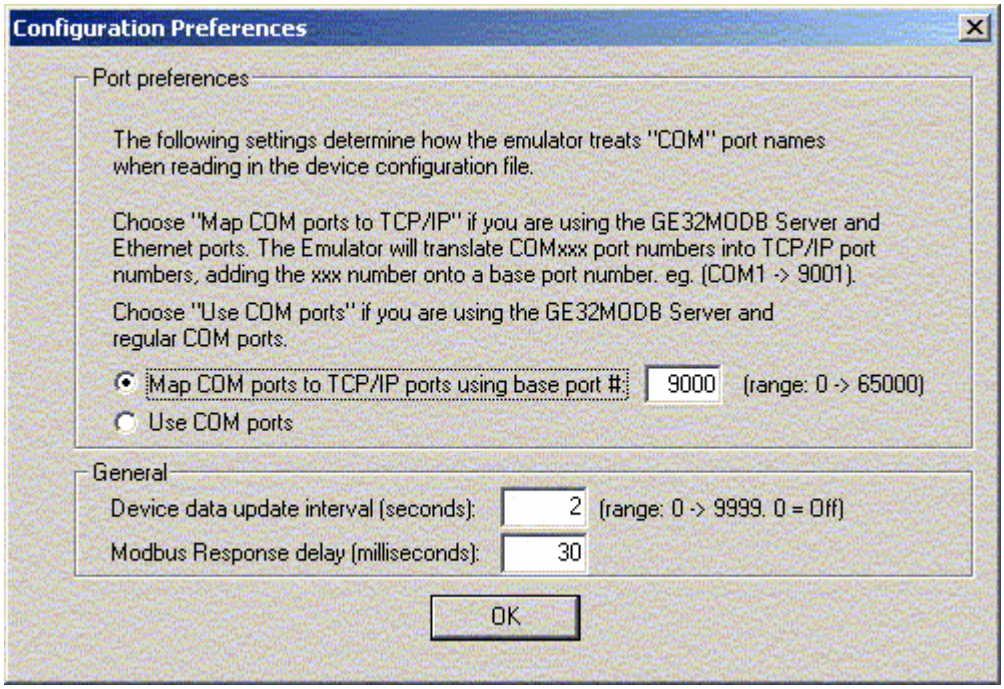
View:

Background, Text color, Toolbar and *Status* bar are self-explanatory.
I/O traffic – Turns on/off the I/O traffic display update. When turned off the display is no longer updated with fresh data, however the Emulator is still processing Modbus messages. Turning off the I/O traffic reduces the CPU usage on the Emulator PC. The “Current Time” and “Elapsed Time” columns will display “I/O Updates DISABLED” when the I/O traffic is disabled.

Config:

Import Configuration – Import the config.txt file created with the PMCS Modbus server(s)
Delete Configuration – Self-explanatory
Preferences – evokes the ‘configuration preferences’

window as seen below and can only be evoked *before* a configuration is imported.



Port preferences – there are two possible selections:
*Map COM ports to *:* Used with NativeCom on the Modbus Master PC and forces the remapping of Comports in the config file as IP ports on the Emulator PC's TCP/IP address. The user can modify the default base IP port to comport mappings. Starting port 9000 is the recommended default setting.
Use COM ports: forces the Emulator to listen to real Comports configured on the Emulator PC.

General –
Device data update interval: Global time interval when ALL dynamic device data changes occur. Setting this to '0' places all dynamic values into a static, unchanging min value state.
Modbus Response delay: Inserts a delay for all Modbus responses. This allows a more realistic simulation of device responses.

Tools:

Control Panel – displays a dialog box with two tabs. The first tab, “Exceptions”, allows the user to ‘inject’ Modbus Exception errors into the Modbus traffic for a particular device. The second tab, “Activate/Deactivate”, gives the user the ability to activate and deactivate selected devices and/or entire Com/IP ports.

Modbus Client – displays a dialog box (only available in "Run" mode) that allows the user to read and write values from/to an individual device’s registers. Note that you can only write to a device register that has a mnemonic associated with it in the devices’ or device type .csv file. Without a .csv file to define register range values, all registers will have a values of zero. On this dialog box, the "Set" button will fix a register to a static value, until the "Reset" button is pressed. The “Reset” button will reset the mnemonic to the min value (for static registers) or into the dynamic profile (for dynamic registers).

Real-Time Client – displays a dialog box (only available in "Run" mode) which shows the current reading of all configured registers for a single (selectable) device.

Configuring the Emulator Startup Environment

1. To start the Emulator, select the Emulator icon in the GE_PMCS program group. This will start the PMCS Modbus System Emulator. The Emulator always starts in an un-configured state. Configuration of the Emulator requires the ‘config.txt’ file created with the “Exporting the PMCS DDE/OPC server(s) configuration” step at the beginning of this chapter For non-PMCS systems the configuration can be created with the config tool included with the Emulator
2. Open menu item “*Config>Preferences*”. The ‘Configuration Preferences’ window should appear
3. In the “*Port Preferences*” section;
   if using the GE32MODB server with NativeCom or with the GE32MTCP server on the host PC make sure the “Map COM ports….” Radio button is selected. Leave the default base port at ‘9000’ (changing it is for

advanced configurations).
If using with a PC serial port choose “Use COM ports”, with this selection the Emulator will listen to the serial port(s) on the Emulator PC. You must use a straight, pass-thru, serial cable to connect the Emulator to the PC running the Modbus master.

4. Under the “*General*” section;
enter how many seconds between data updates. This will dictate how often the Emulator will change/increment ALL Modbus Register Mnemonics configured for ‘dynamic update’. Setting this value to zero forces all values to be static. All mnemonics configured to be dynamic will update at the same time, regardless of how often they are polled by the master. Note: Device update times in the SCADA are also dependent upon the configuration of the Modbus network (total number of devices, device per RS-485 LAN, etc)

5. Open menu item “*Config>Import Configuration*”. Once the ‘Open file’ window appears select the ‘*.txt’ file created with the ‘Reports’ function of either the PMCS GE32MODB or GE32MTCP Modbus servers (See section titled “**Exporting the DDE/OPC server(s) configuration”**). Note: A single instance of the Emulator can only import and use 1 config file at a time. If data from both GE Modbus servers need to be run than start a second Emulator instance and import the ‘config.txt’ file from the second server

6. After the Emulator is loaded with the ‘config.txt’ file it is ready to run

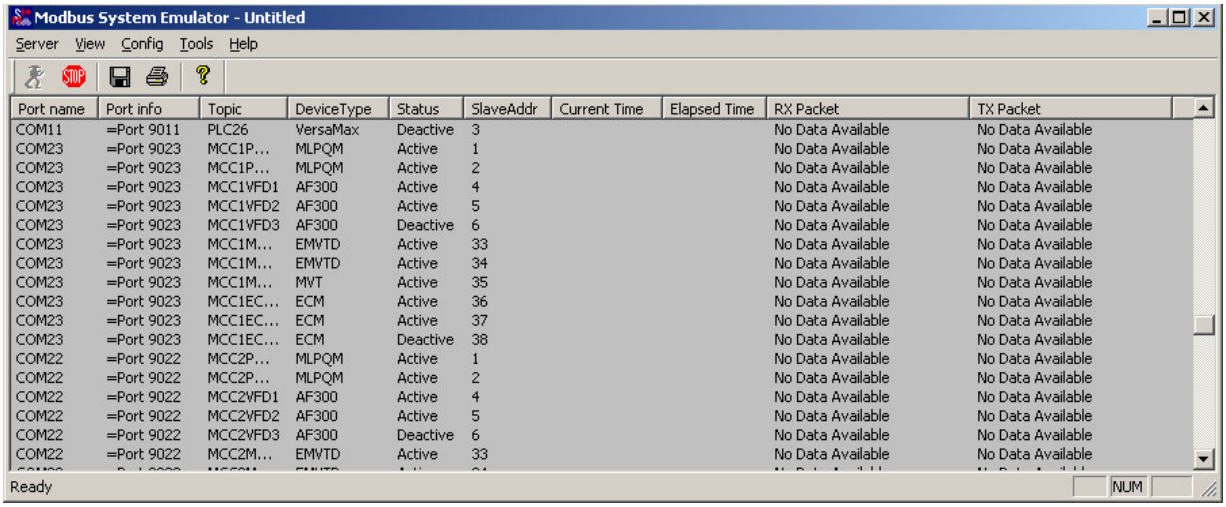
7. To delete the configuration either shutdown the Emulator or go to Config>Delete Configuration

Running the Emulator

1. Once the Emulator startup environment has been configured the Emulator is ready to run.

2. Go to menu item ‘*Server>Run’*

3. The Emulator window will display the configuration that has been loaded into it and is ready to receive Modbus requests from an appropriately configured Modbus host using NativeCom, ModbusIP or direct serial port. The window will look similar to the one below.



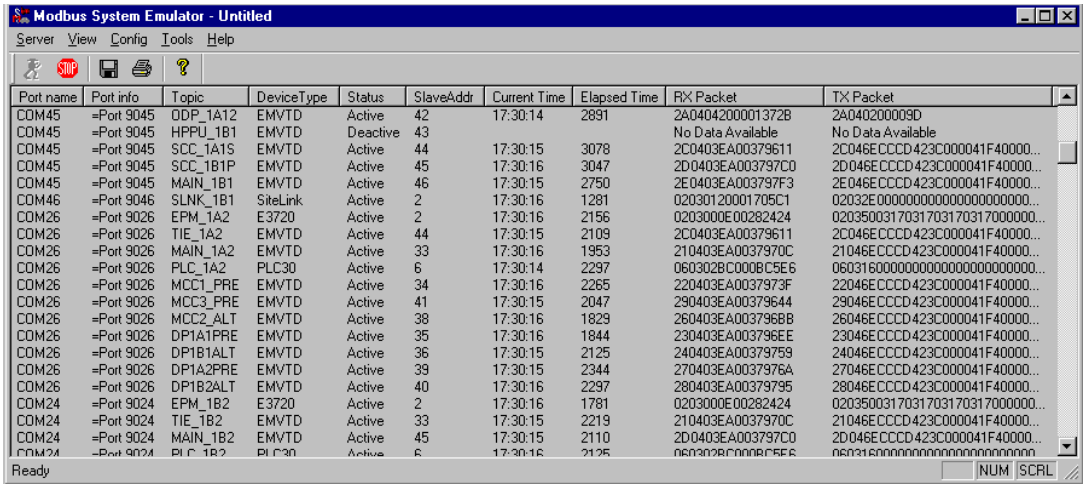
Port name	Port info	Topic	DeviceType	Status	SlaveAddr	Current Time	Elapsed Time	RX Packet	TX Packet
COM11	=Port 9011	PLC26	VersaMax	Deactive	3			No Data Available	No Data Available
COM23	=Port 9023	MCC1P...	MLPQM	Active	1			No Data Available	No Data Available
COM23	=Port 9023	MCC1P...	MLPQM	Active	2			No Data Available	No Data Available
COM23	=Port 9023	MCC1VFD1	AF300	Active	4			No Data Available	No Data Available
COM23	=Port 9023	MCC1VFD2	AF300	Active	5			No Data Available	No Data Available
COM23	=Port 9023	MCC1VFD3	AF300	Deactive	6			No Data Available	No Data Available
COM23	=Port 9023	MCC1M...	EMVTD	Active	33			No Data Available	No Data Available
COM23	=Port 9023	MCC1M...	EMVTD	Active	34			No Data Available	No Data Available
COM23	=Port 9023	MCC1M...	MVT	Active	35			No Data Available	No Data Available
COM23	=Port 9023	MCC1EC...	ECM	Active	36			No Data Available	No Data Available
COM23	=Port 9023	MCC1EC...	ECM	Active	37			No Data Available	No Data Available
COM23	=Port 9023	MCC1EC...	ECM	Deactive	38			No Data Available	No Data Available
COM22	=Port 9022	MCC2P...	MLPQM	Active	1			No Data Available	No Data Available
COM22	=Port 9022	MCC2P...	MLPQM	Active	2			No Data Available	No Data Available
COM22	=Port 9022	MCC2VFD1	AF300	Active	4			No Data Available	No Data Available
COM22	=Port 9022	MCC2VFD2	AF300	Active	5			No Data Available	No Data Available
COM22	=Port 9022	MCC2VFD3	AF300	Deactive	6			No Data Available	No Data Available
COM22	=Port 9022	MCC2M...	EMVTD	Active	33			No Data Available	No Data Available

Chapter 4
Using the PMCS Modbus Emulator

I Display:

With the Emulator running but no client (Modbus server) connected to it the display will look similar to the one at the end of the previous chapter. In the previous chapter the Emulator configured and running and NativeCom configured correctly the next step is to start the PMCS server. If you wish to run both the GE32MODB and the GE32MTCP servers simultaneously you must run two separate instances of the Emulator, one with the MODB server configuration and the other with the MTCP configuration.

With the Modbus server is started and a client requesting data from it the Emulator window will look similar to the one below.



Port name	Port info	Topic	DeviceType	Status	SlaveAddr	Current Time	Elapsed Time	RX Packet	TX Packet
COM45	=Port 9045	ODP_1A12	EMVTD	Active	42	17:30:14	2891	2A0404200001372B	2A040200009D
COM45	=Port 9045	HPPU_1B1	EMVTD	Deactive	43			No Data Available	No Data Available
COM45	=Port 9045	SCC_1A1S	EMVTD	Active	44	17:30:15	3078	2C0403EA00379611	2C046ECCCD423C000041F40000...
COM45	=Port 9045	SCC_1B1P	EMVTD	Active	45	17:30:16	3047	2D0403EA003797C0	2D046ECCCD423C000041F40000...
COM45	=Port 9045	MAIN_1B1	EMVTD	Active	46	17:30:15	2750	2E0403EA003797F3	2E046ECCCD423C000041F40000...
COM46	=Port 9046	SLNK_1B1	SiteLink	Active	2	17:30:16	1281	02030120001705C1	02032E0000000000000000000000...
COM26	=Port 9026	EPM_1A2	E3720	Active	2	17:30:16	2156	0203000E00282424	020350031703170317031700000000...
COM26	=Port 9026	TIE_1A2	EMVTD	Active	44	17:30:15	2109	2C0403EA00379611	2C046ECCCD423C000041F40000...
COM26	=Port 9026	MAIN_1A2	EMVTD	Active	33	17:30:16	1953	210403EA0037970C	21046ECCCD423C000041F40000...
COM26	=Port 9026	PLC_1A2	PLC30	Active	6	17:30:14	2297	060302BC000BC5E6	060316000000000000000000000000...
COM26	=Port 9026	MCC1_PRE	EMVTD	Active	34	17:30:16	2265	220403EA0037973F	22046ECCCD423C000041F40000...
COM26	=Port 9026	MCC3_PRE	EMVTD	Active	41	17:30:15	2047	290403EA00379644	29046ECCCD423C000041F40000...
COM26	=Port 9026	MCC2_ALT	EMVTD	Active	38	17:30:16	1829	260403EA003796BB	26046ECCCD423C000041F40000...
COM26	=Port 9026	DP1A1PRE	EMVTD	Active	35	17:30:16	1844	230403EA003796EE	23046ECCCD423C000041F40000...
COM26	=Port 9026	DP1B1ALT	EMVTD	Active	36	17:30:15	2125	240403EA00379759	24046ECCCD423C000041F40000...
COM26	=Port 9026	DP1A2PRE	EMVTD	Active	39	17:30:15	2344	270403EA0037976A	27046ECCCD423C000041F40000...
COM26	=Port 9026	DP1B2ALT	EMVTD	Active	40	17:30:16	2297	280403EA00379795	28046ECCCD423C000041F40000...
COM24	=Port 9024	EPM_1B2	E3720	Active	2	17:30:16	1781	0203000E00282424	020350031703170317031700000000...
COM24	=Port 9024	TIE_1B2	EMVTD	Active	33	17:30:15	2219	210403EA0037970C	21046ECCCD423C000041F40000...
COM24	=Port 9024	MAIN_1B2	EMVTD	Active	45	17:30:15	2110	2D0403EA003797C0	2D046ECCCD423C000041F40000...
COM24	=Port 9024	PLC_1B2	PLC30	Active	6	17:30:16	2125	060302BC000BC5E6	060316000000000000000000000000...

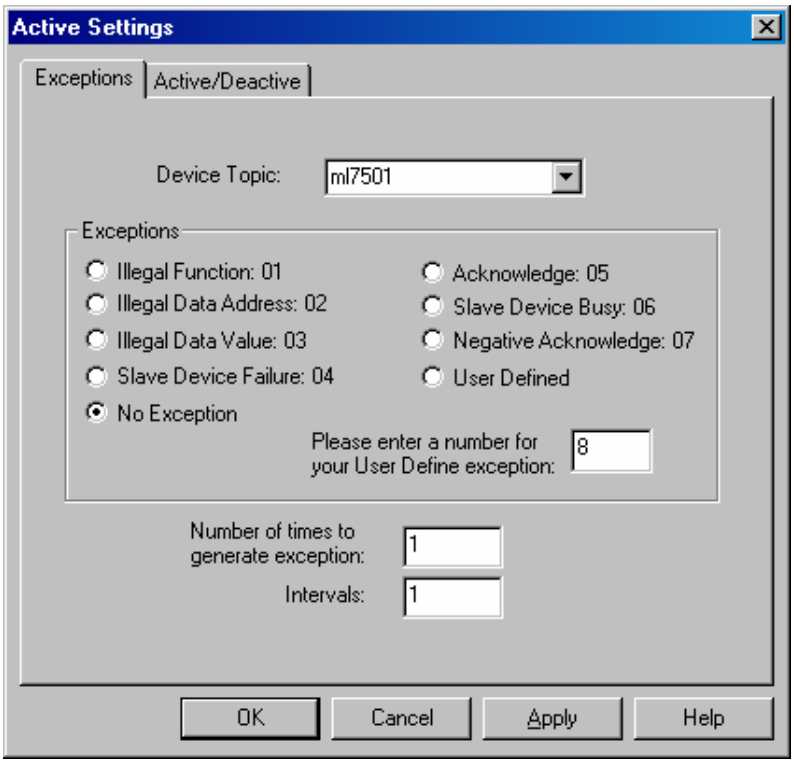
Column Headings and their Meanings

Port Name – Designates the comport the device is associated with in the GE32MODB or NetCom port in the GE32MTCP

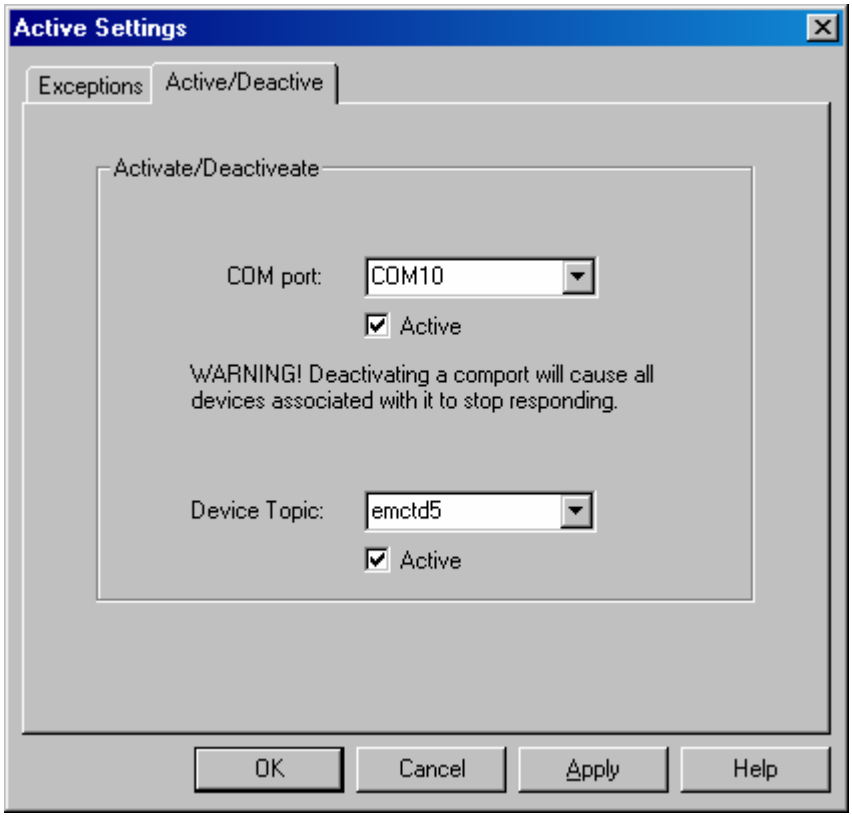
Port Info – Displays the IP port the Emulator is listening to; default mapping is Comport number + 9000 = IP port, i.e. com45 + 9000 = IP port 9045, GE32MTCP config will show "= Port 502"
Topic – Device name in the PMCS server associated with the device being emulated
Device Type - Device type associated with the device name
Status – Displays if the device is active or deactivated in the server and Emulator. A deactivated device will not respond to requests.
Slave Address – Modbus slave address of device being emulated
Current Time - Time stamp of the last received Modbus request
Elapsed Time – Time between the LAST TWO Modbus requests
RX Packet – Actual Modbus request from the device
TX Packet - Actual Modbus response to the device for the above request. Note: TX Packet size could be larger than TX packet display buffer. This would cause the display to be truncated.

II Using the 'Tools' Menu

Control Panel – displays a dialog box containing options available to the user to 'inject' data errors into the Modbus traffic.



The “Exceptions” tab allows a user to a inject Modbus error code into the message traffic of a specific device. The user can select from standard Modbus exception codes or create a custom, user defined, code. The number of error occurrences can be controlled as well as their spacing between subsequent Modbus requests (for the same device). To use select the device topic, the exception type (number) and the number and rate of occurrences. Next hit “Apply”. Only one device at a time can be configured. Attempting to apply an exception routine for another device while an exception routine is being performed will reset the current exception to the new one.



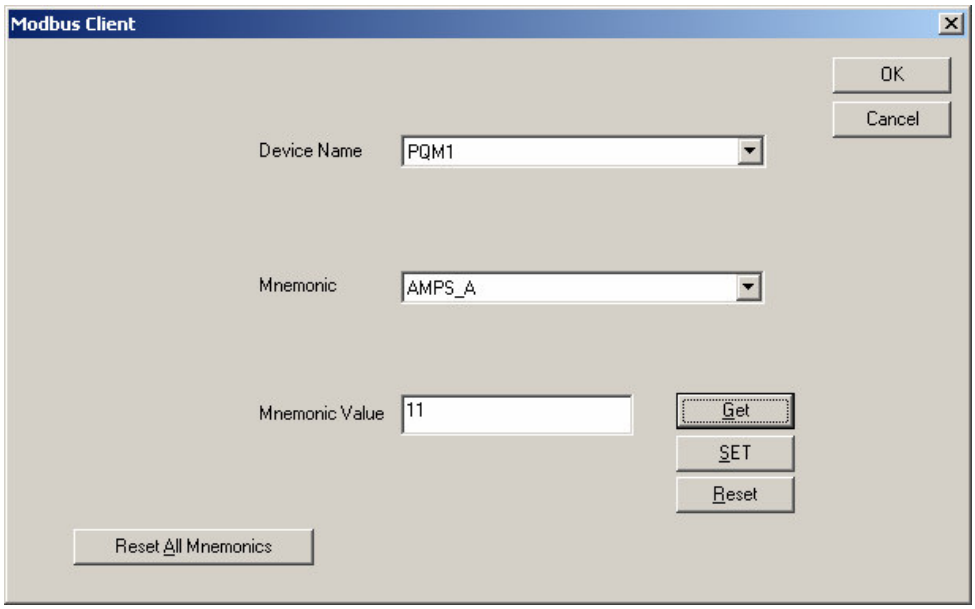
The “Activate/Deactivate” tab allows a user to Activate/Deactivate a specific device or an entire Comport. For PMCS: When a device is deactivated it will ‘time out’ in the PMCS Modbus server(s) and will be declared ‘dead’ when the “Maximum Qry Retries” server threshold has been reached. The device will be re-polled at the servers “Dead Device Scan Interval”.

To Activate/Deactivate a specific device select it from the drop down menu, unselect the ‘Active’ check box below it and press “Apply”. The devices should begin ‘timing out’ in the Modbus server (if data for it is being requested).
Activation/Deactivation of a Comport will cause all the devices on the specified Comport to go ‘dead’. Note: If a device is set for ‘deactive’ in the Modbus server, setting it for ‘active’ in the Emulator will have no effect on the server setting.
To Activate/Deactivate a comport select it from the drop down menu, unselect the ‘Active’ box and press ‘apply’.

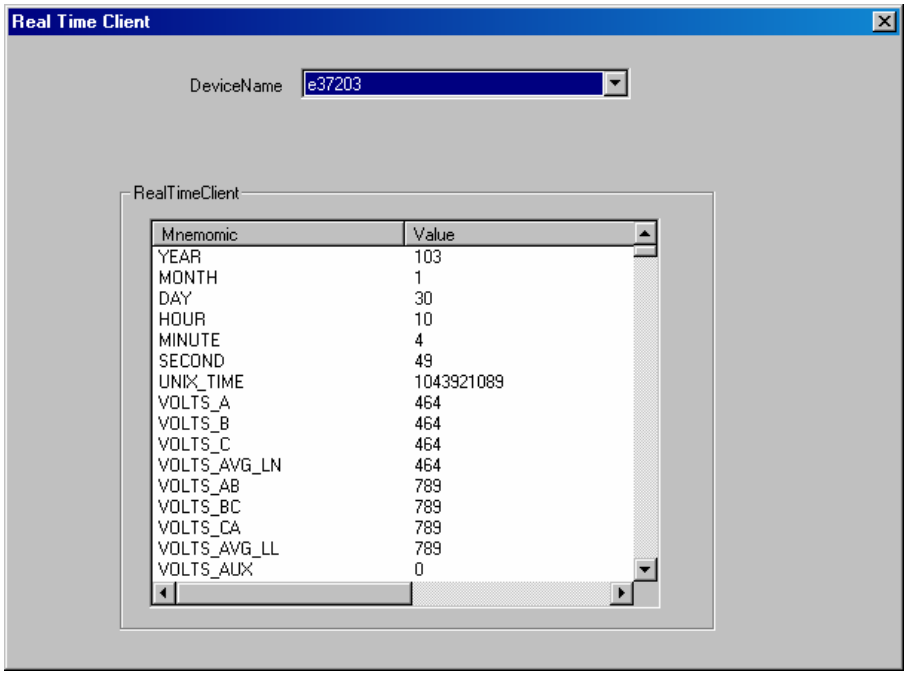
Modbus Client – Allows a user to read/write values to/from an individual device’s mnemonics/registers. In this dialog box, the "Set" button will set a register to the static value entered into the value box, until the "Reset" button is pressed. At reset the register will return to its initialization or min value. The “Reset All Mnemonics” resets all the devices’ mnemonics back to their initialization values.

A user can only write to a device register(s) that has a mnemonic name associated with a register(s) in the devices’ or device type profile.csv file. The profile .csv file is used to define registers, register names (mnemonics), initial value or min/max range values. Values written must be in the range of the defined mnemonics Modbus register(s). For example a mnemonic defined as a 16 bit unsigned register has a range of (0~65,535), a 16bit signed has a range of (-32,737 ~ +32,767). Values written outside of the registers range will yield incorrect values at the Modbus Master.

Note: All value written are non-retentive. Any value changed via the Modbus Client will be lost upon shutdown of the Emulator



Real Time Client – Allows a use to view a quick view of all the mnemonic values, including dynamic registers for a specific device. To use select devices from the 'DeviceName' drop down box. The display box show all the values for all the mnemonics configured for the device chosen.



III Creating Device CSV Profiles

Each device **or** device type in the Emulator must have a device value 'profile' csv file in order to emulate a particular device or device type. The profile contains information such as registers, register types, mnemonics, mnemonic values, if the mnemonic is dynamic or static and, if dynamic, its incremental step function. A profile contains values for all the Modbus registers of a device type or a specific device. The Emulator loads the device or device type profile csv files when the it is placed in 'Run' mode. A profile for each device type in PMCS 6.11a has been included with the Emulator and is contained in the Emulators 'data' directory. Several profiles from common generic Modbus devices have been included also. The profile name must match either the device topic name (for a specific device) or the device type name (for all devices of that type). **Note: In order to accurately emulate a device or device type all values should be based on typical values seen in the registers for the real device.** A sample EPM9650Q.csv file for the GE device type "EPM9650Q" is below.

	A	B	C	D	E	F
1	DEVICE_NAME	R4X0000S16	0107 Nexus 1250			
2	FW_STR_1	R4X0008S16				
3	FW_STR_2	R4X0010S16				
4	FW_STR_3	R4X0018S16				
5	FW_STR_4	R4X0020S16				
6	FW_STR_5	R4X0028S16				
7	FW_STR_6	R4X0030S16				
8	FW_STR_7	R4X0038S16				
9	FW_STR_8	R4X0040S16				
10	BOOT_VERNO	R4X0048L	808464952			
11	RUNTIME_VERNO	R4X004AL	808531766			
12	DSP_BOOT_VERNO	R4X004CL	1110454324			
13	DSP_RUNTIME_VERNO	R4X004EL	808530996			
14	ON_TIME_1	R4X0050L	335741206			
15	ON_TIME_2	R4X0052L	303506442			
16	CURRENT_TIME_1	R4X0054L	335741207			
17	CURRENT_TIME_2	R4X0056L	136853259			
18	CUR_DAYOF_WEEK	R4X0058	5			
19	TS_VOLT_AN	R4X007AL	7680000	8320000	2	30000
20	TS_VOLT_BN	R4X007CL	7690000	8310000	2	30000
21	TS_VOLT_CN	R4X007EL	7700000	8400000	2	30000

Column Heading Definitions

The profile csv file contains 7 defined columns:
Column A: Mnemonic name – must match exactly the mnemonic name in the server, mnemonics must not contains any blank spaces and can be no longer than 20 characters.
Column B: Register number and format - must match exactly the format in the server, see 'Register format', Appendix A for more info.
Column C: Min value or static value – must match register type (integer, string, etc), default = 0 for numbers, 'null' for strings.
Column D: Max value – used only if register is intended to be dynamic. This value MUST be larger than the min value.
Column E: Static (default) or Dynamic, 1 = freeze at min, 2 = dynamic, default = 1
Column F: Step function used during dynamic value increment, default = 1
Column G: DDE enable, - "Y" enables DDE poke/request, Default = "N"

If columns C, D, E, F or G are left blank the Emulator will assume default values for them.

Since the profile csv file must match the device type configuration in the PMCS DDE/OPC server the best method to create a simple profile is to export the mnemonic list of the device type from the PMCS server. Please see server documentation on creating/editing generic device types.

Comments

Any value/text that has a "/" in front of it is considered to be a comment and will be ignored by the Emulator. Comments must be to the far right of a row or in a dedicated row.

@FLOAT 32bit IEEE Floating Number Function

The Emulator can accommodate two different types of implementation for the IEEE 32bit floating number format.

@FLOAT1 format: The default type used by most GE devices, including all GE Modbus Concentrator based devices. The format is: (1st byte, 0 byte) first register, (3rd byte, 2nd byte) second

register, for each floating-point register pair. This is the default configuration.

@FLOAT2 format: Used by the GE UR relay and some third party devices. The format is: (3rd byte, 2nd byte) second register, (1st byte, 0 byte) first register, for each floating-point register pair.

To enable the @FLOAT2 function manually enter “@FLOAT2” in the first, topmost, entry in the devices profile csv file. Once the Emulator reads this all “F” formatted registers for the device will use this format. *If this entry is not found the “@FLOAT1” format will be used.*

DDE Poke/Request Interface

The Emulator supports DDE Pokes/Requests to activated mnemonics. The user can create custom scripts using third party programs (Excel, SCADA, VB, etc) to poke/request mnemonic values via DDE. To enable DDE enter a "Y" in column G for the register(s) you want to poke/read. Once a mnemonic is enabled for DDE it automatically becomes ‘static’ even if it is defined as dynamic. A value change can only be done via DDE or the Emulator client. Also, only enable the DDE function for required mnemonics. The DDE interface does NOT support the DDE Advise function. There is a simple DDE client included with the Emulator to use.

The formula for all DDE transactions is
“Emulator|TopicName!Mnemonic”

An additional DDE function is the ability to reset all mnemonics in a device topic to its initialization values, if the value is dynamic then value will reset to its min value and then resume incrementing. Poking a ‘1’ resets the mnemonics.

The DDE formula for resetting all mnemonics is
“Emulator|TopicName!DDEReset”

Creating Dynamic Value Registers

To create a dynamic register:
Choose the Mnemonic that needs to be dynamic
Enter a max value in Column D larger than Column C
Enter the value "2" into Column E
Column F is the step function, default is '1', value will increment by

this number. This size of this value may be critical to how accurate the emulated device simulates the real device type. Make sure the step value is realistic.

The update interval (in seconds) is determined by "data update interval" under the *'Configuration Preferences*' window. This value is global and updates all dynamic registers simultaneously regardless of the devices Modbus polling status. To change this value there cannot be a configuration in the Emulator. Value profile is a classic sawtooth between the Min/Max values.

Note: There may be duplicate registers in a device's profile.csv file. All parameters for duplicates MUST match:
For example:
AMPS_A_RMS, R3X1800F, 10, 100, 2, 5
S1_AMPS_A_RMS, R3X1800F, 10, 100, 2, 5
The min/max ranges must match for both sets:

Dynamic update can only be applied to single element, numerical registers. Dynamic updates for arrays and strings are NOT supported.

A mnemonic set for dynamic values can still have a value entered via the Emulator Modbus client but once a value is 'set' it become static at that value until the 'reset' button is applied. The 'reset' command returns the mnemonic to its default dynamic profile.

Modbus Register Maps

Please see Appendix A of the GE PMCS manual GEH-6509.pdf for GE PMCS device register maps and formatting rules. This manual is included on the Emulator CD. For device specific formatting rules please reference the specific devices user manual, integration manual or communication manual

Emulator Modbus Register Formatting

Please see Appendix B for register formatting rules and see Chapter 6 of GEH-6510 PMCS server manual for more information. This manual is included in the Emulator directory.

Creating Device Profiles

The best way to create a device profile is using the GE PMCS server or, if the server is unavailable, using the Emulator Config tool included with the Emulator.

To create a device profile:

1. The device type must already be configured in the PMCS server or Emulator Config tool. If this is a new, 3rd party Modbus device it is highly recommended that communication with a real device be tested first to verify correct configuration of the devices' Modbus map and formats.

2. If using the PMCS server, stop the server and go to configure>configure and select the "Device Type Info" button. If using the Emulator Config tool go directly to the "Device Type Info" button.

3. Select the device type needed in the "Configured Device Type" window on the left.

4. Press the "Mnemonics" button the right. The Mnemonics window for the device should appear.

5. Select the "Export Mnemonics" window. The "Export Mnemonics" Save-As window will appear.

6. If you wish all the devices of this device type to have the same profile then save the resulting csv file as "*DeviceType*.csv" where *DeviceType* is the exact name of the device type picked in step 3. Values populated in the resulting file will only be used by all the devices of the same device type picked in step 3. The only exception to this will be a TopicName specific device created in step 7 below

7. If you wish a specific device to have a different profile than others of the same type than save the file as "*TopicName*.csv", where *TopicName* is a specific device name configured as the device type picked in step 3. Values populated in the resulting file will only be used by the device with the same *TopicName*.

8. Close all windows opened in the steps above.

Populating Profiles with Data from Real Devices

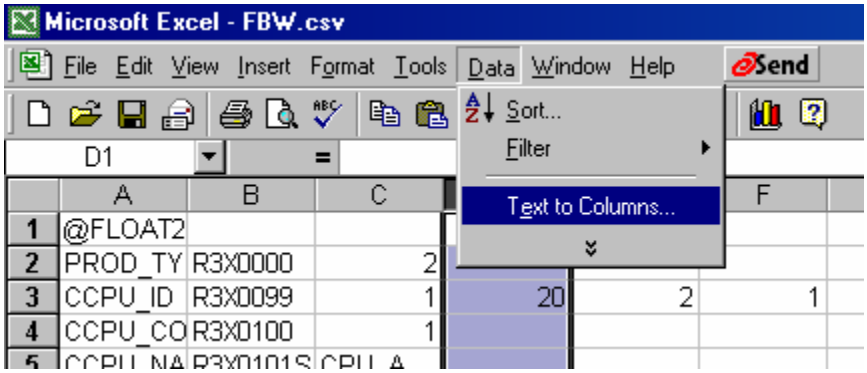
The quickest and easiest method of creating realistic values for a specific device or device type is to capture the values from the device itself via the PMCS DDE server. The procedure below is the same one used to create the value profiles for all standard PMCS device profiles included with the Emulator. Use the following steps:

1. Configure the PMCS server to communicate to the device.
2. Export the devices Mnemonics from the DDE server, this is done from the server using "View>Configuration>DeviceTypeInfo", if the server is not in 'Run' mode you must use "Configure>Configure>DeviceTypeInfo".
3. When the "Device Type" window appears select the device on the left
4. Click on the "Mnemonics" button on the right. Note: A mnemonic is required in order to simulate a value.
5. When the "Mnemonics" window appears select the "Export Mnemonics" button and save the resulting CSV file to an easily accessible directory
6. Open the csv file with Excel.
7. In the first row, third column enter the following formula:
   ="=GE32MODB|<*DEVICENAME*>!"&A1
   where <*DEVICENAME*> is the name of the device in the DDE server. The result should look like a DDE link to the server; i.e.
   ' =GE32MODB|METER1!AMPS_A '
8. Highlight the cell created above and click on the lower right of the cell. Now 'drag' the cell downward in the same column. Drag it to the last row entry in the file. You should now see the entire row populated with DDE links.
9. To activate the links select the entire column created above then 'copy 'and 'paste special'>values to the column next to it.

10. You should now have a column of DDE text links not associated with the column that created them

11. To change this column into true DDE links highlight the entire column

12. Go to Excel menu item ‘Data>Text to Columns’



13. The “Convert Text to Columns” window should appear.

14. Click on the “Finish” button

15. All highlighted cell should now be DDE links advising the server for data

16. Once all the data is collected highlight all the cells then ‘copy ‘and ‘paste special’>values to the column next to it.

17. Delete the DDE column

18. There should now be three columns, Column A are the mnemonics, Column B are the registers assigned to the mnemonics and Column C are the values captured from the device. To make any one particular mnemonic dynamic please see the section above titled “Creating Dynamic Value Registers”

19. Save the file as either <devicename>.CSV or <devicetype>.CSV. If saved as <devicename>.CSV than the values in the file will only be used for the specific device named by the <devicename>. If saved as <devicetype>.CSV the file will become “global” for all devices of that type, unless a specific device has its own <devicename>.CSV file.

Device Generated Alarm/Events

Many GE meters and relays support internally generated alarms and or events. These internal alarms/events can be for items such as ‘breaker closed’, ‘current unbalance’, ‘internal error’, etc. Alarm/events generated by devices are presented as ‘hex’ codes that the Modbus master must read and translate. The Emulator has the ability to replicate these ‘hex’ codes in the proper format and sequence but the user must have a thorough understanding of the device being emulated in order to understand the generated the code. Currently the Emulator only supports alarm/event code generation for the following GE devices; EMVTC/D, MVT, PLEPM, MPD, ECM and PLM. Additional devices will be added in future versions of the Emulator.

The generation of the hex codes is via a DDE item generated by the Emulator for the above devices. This tag exists only in the Emulator and is not found in the devices csv profile.

To generate a Modbus hex code alarm/event for a device a DDE Poke must be sent to the Emulator with the following configuration:

EMULATOR|*DevName*!GENERATE_EVENT = “*nn*:d1:d2:d3:d4:d5:d6”
DevName is the topic name of the device
where *nn* is a device event code(in decimal)
and d1~d6 are optional parameters for the event. Please see device event documentation for these parameters.

The Emulator only stores 1 event per device at a time.

Alarm/Event Hex code generation for other devices is possible if the user creates a custom VB script to properly DDE poke/request the correct registers for a device. Please see the devices event documentation to reference how to create the script.

Chapter 5 Advanced Topics

I Modbus TCP/IP devices and configuration:

The Emulator supports Modbus TCP/IP functions in a manner similar to Modbus RTU. All Modbus functionality supported with Modbus RTU is supported by the Emulator under Modbus TCP. However the following methodologies must be applied:

Since there is only 1 IP address on the host PC it can only emulate a single Modbus TCP device at the standard Modbus IP port(502). Any additional devices must be run with a separate Emulator on a separate PC. This method, while workable for a couple of devices, rapidly exceeds the number of PC's available. The following 'work arounds' can eliminate this limitation. Also, The Emulator does not support multiple, simultaneous Modbus masters or connections on the same IP port.. Only one master per port is allowed.

1. Simulate each Modbus TCPIP device as a Modbus RTU device on its own serial port: For GE PMCS systems please configure and use the GE32MOD server. Configure each Modbus device on its own Comport. Since the GE32MOD server handles each Comport independently the end result is parallel polling of the devices in a manner similar to how the GE32MTCP server polls independent IP based devices. The SCADA software must be pointed to the GE32MODB server. When the system is implemented in the field the SCADA must be redirected to the GE32MTCP server.

2. For other Modbus TCP masters: Either use a method similar to above or, if available, configure each device to use a different IP ports (other than 502). The Emulator can then be configured to listen to these other ports. This is easily done by editing the device and comport/Netcom section at the bottom of config.txt file. Instructions on how to do this are included next in this chapter.

Editing the ‘Config.txt’ file for use with ModbusIP devices

When converting from Modbus RTU to Modbus IP remember the rules derived from the above section:

- There can be only 1 TCP/IP address for all the Modbus IP devices but if your Modbus master allows it there can be different IP ports assigned. The GE PMCS GE32MTCP servers IP port is fixed at 502. It cannot be changed.
- Since there can be only 1 IP address each device must have its own Modbus address, unless different IP ports can be assigned.
- Only 1 Modbus IP master connection is allowed and it must poll data in an ‘RTU’ fashion; Query/Response, Query/Response, etc. This is regardless of IP port assignment.
- Data updates for all devices on the same IP port will be similar to Modbus RTU updates for all devices on the same Comport.

If using the GE PMCS GE32MTCP Modbus IP server the ‘config.txt file is generated using the same configuration export function as the GE32MODB. The following rules must be used in configuring the GE32MTCP server:

- Configure a NetCom port for the IP address of the Emulator PC. The IP address must be used, network names are not supported.
- **All** devices configured in the GE32MTCP server **MUST** be configured for the same NetCom port configured above.
- Each device on the same NetCom must have an unique Modbus address.

If the PMCS GE32MTCP server is not used than the initial configuration can be made with the ‘EmulatorConfig’ tool included with the Emulator or with the GE32MODB. Please see the EmulatorConfig tool user guide for instructions on its use. Pay close attention to the section on Modbus IP device configuration.

The config tool does not create the Modbus IP configuration directly. It creates a Modbus RTU config that must be converted to a Modbus IP config by editing the resulting config.txt file.

To convert a 'config.txt' file generated by the EmulatorConfig tool or generated by the GE32MODB server perform the following steps:

1. Generate the 'config.txt' file with the EmulatorConfig tool or the GE32MODB.

2. Open the file with notepad or similar program.

3. Scroll down to the very bottom of the file until the 'Comport' section is found. It will look similar to this:

   COMPORTS :
   ComPortName :COM100
   Port Params :COM100:19200,n,8,1
   ComPortName :COM10
   Port Params :COM10:19200,n,8,1
   ComPortName :COM11
   Port Params :COM11:19200,n,8,1

4. Delete the entire section and replace it with:

   COMPORTS :
   ComPortName :NetCOM1
   Port Params :Not Configured
   Notes :Not Configured
   ComPortName :NetCOM1
   Port Params :192.168.0.5:502
   Notes :

   Edit the line "*Port Params :192.168.0.5:502"* and replace the IP address, *192.168.0.5*, with the IP address of the Emulator PC. Leave the "*:502*" at the end of the IP address. This is the IP port assignement.

5. In the text document the next section defines the device configuration and Comports used. Perform a 'search and replace' to replace all Comport assignments with the new Modbus IP assignments. Here is an example:

   DevName :METER1
   DevTpName :E7300
   ComPortName :COM10
   Modbus Port Config String :COM10:19200,n,8,1

Edit it to look like:

DevName :METER1
DevTpName :E7300
ComPortName :NetCOM1
Ethernet Port Config String :192.168.0.5:502

Also verify the each device in the configuration on the same NetCom has its own unique Modbus address as identified in the “SlaveAddr:” line of each device config.

6. An example of a serialconfig.txt and an IPconfig.txt are included in the Emulators document directory and on the Emulator install CD for reference.

7. Note that if the Modbus IP master being used supports more than one IP port configuration than more than one Netcom can be configured. The same IP address must be used but change the port assignment “:502” to the new port.

II Specific Device configurations:

Details concerning specific PMCS devices

GE EPM3720/EPM3710 meter 32bit register workaround**:**

The standard PMCS configuration for these two devices is a 32bit register format for all Modbus registers verses the standard 16bit register format. The Emulator cannot recognize 32bit registers. To properly emulate the EPM3720/3710 they must be recreated in the PMCS server as generic devices.
1) Create two new generic devices in the PMCS server called “EPM3720EM” and “EPM3710EM”.
2) Add Modbus Functions codes 3 and 16 to the function code configuration of the new generic devices.
3) Import the “Reg_EPM37xx_01.csv*” and “Mne_ EPM37xx_01.csv *” files included in the Emulators’ “ExtraCfgs” directory into the config of each device using the import feature of each item.
4) The Emulator contains already profile CSV files for these two generic devices using the EPM3710.csv and EPM3720.csv files.

Note: The above procedure will allow emulation of the majority of device values however there may be issues with any function requiring the 32 bit Modulus 10000 format. Examples of these are energy values, etc.

GE MVT Trip unit KWh register:

The MicroVersaTrip(MVT) unit lacks an energy demand register that is required for PMCS applications such as CAM or PMCS wizards. To compensate for this the PMCS Modbus servers have an internal, virtual energy demand 'register' that can be accessed. To increase the accuracy of this register the server requires exclusive use of the MVT KWh register internal to the MVT. As a result the KWh register is **not** included in the servers' MVT configuration, even though the register exists in the MVT. However to emulate KWh with the Emulator both the KWh register and mnemonic must be added the MVT section configuration text file. In addition the mnemonic and register must be added to the MVT.CSV profile used by the Emulator.
The following register and mnemonic must be added to the MVT config in the config.txt file:

```
MnemonicName 47 : KWh
    RegFormat  :R31058f
```

And the MVT "DynamicValueRegisterGroup" register group 'End Address' range must be extended from 1055 to 1059.
This register must be added to the Emulator config.txt only, do NOT change the MVT configuration in the PMCS servers. Or if changed to create the config.txt file it must be change back to the original settings after the file is created.

GE ION based devices:

Devices such as the GE EPM7500, 7600 and 7700 use a protocol called "ION" which the PMCS Emulator does NOT emulate. However these devices can be simulated from a SCADA perspective if configured as a generic Modbus device. This will allow the SCADA package to be configured for items such as alarms/trends/scripts, etc. In the "ExtraCfgs" directory are the config files for the EPM7700 devices and profile files are included in the data directory for the EPM7700. The 7500 and 7600 can be recreated in a similar fashion. Note: The register map configured for the EPM7700 is purely fictitious and was created solely for the use of the Emulator. It has **no** relation to any real

Modbus register map in the device.

Generic Modbus devices:

Please configure the generic device using the PMCS server or the Emulator Configuration utility included with the Emulator. Remember to configure mnemonics for the device.

3RD Party Modbus Masters (PLC's, etc):

In addition the Emulator can be used with most third party Modbus masters located on PC's or other masters such as PLC's, HVAC controllers, etc. In this mode the Emulator would be configured with the devices listening to the RS232 port on the PC. This port is then connected to the host serial network either directly or via an RS232-RS485/422 converter. Or, for Modbus IP systems the connection will be via a TCP/IP network. With a converter the Emulator can be 'multi-dropped' with other devices allowing a wide range of testing to be done. Or it can be configured for ModbusIP masters simply by having the PC connected to the masters TCP/IP network.

Chapter 6
Troubleshooting

FAQ

"I can't get the Emulator to work"

1. For Ethernet configurations: Verify you can 'ping' the Emulator PC from the Modbus Master PC
2. Verify the Emulator is in 'Run' mode
3. Verify the Emulator has the correct configuration loaded
4. For Ethernet configurations: Verify that the required Comports on the Server PC in NativeCom are 'pointing' toward the IP of the Emulator PC
5. Verify that the NativeCom port configuration for each comport is set for "Raw TCP". This is a very common mistake.
6. Verify that the IP ports assigned to each Comport is equal to "9000 + Comport #" (for example Comport 23 is set to TCP port 9023).
7. For TCP/IP config: Verify that the Emulator was started and running BEFORE the Modbus Master was started. Occasionally an IP port(s) can 'hang' if the Emulator is not running first. Please shut down the Emulator and Modbus Server and restart them with the Emulator started and running first.
8. For Serial Port config: Verify cable is good and has a null modem adapter (for RS232). While the Modbus Master is attempting to poll the emulator 'stop' and then 'run' the emulator(Might be required, depending upon hardware).

"The export config.txt function of the PMCS server(s) does not work or Emulator Config Tool will not create the config.txt file"

This has been reported by several integrators but has not been reproduced in the lab. It has been reported that the following work around fixes the issue.

1. Rename the servers GE32MODB.cfg in the x"/GE_PMCS/server directory to GE32MODB.tmp
   Start the server, it will prompt for the creation of a GE32MODB.cfg, Select 'yes' and allow the file to be created.

2. Close the server, delete the newly created GE32MODB.cfg file
   Rename the original file, now named GE32MODB.tmp, back to GE32MODB.cfg
   Restart the server
   Export of the config file should now work correctly

3. If the above does not work please use the included "EmulatorConfig" utility. Use the 'topic.cfg" file you created with the GE32MODB server.

"I get an error when I attempt to run the Emulator stating 'Unable to open the file x:/…/Emulator/xxx.csv. This file does not exist'.

1. A missing device type 'profile.csv' file causes this error. The profile contains specific device information such as initialization values, dynamic values, etc. Continuing without a 'profile.csv' file in the Emulator will cause it to respond with '0' for any/all modbus requests to devices configured for that device type. To clear the error create a 'profile.csv' file for the device and place it in the Emulators 'data' directory.

"I see a message 'open port failure' in the Modbus server"

1. The Modbus server PC cannot open the required Comport. Verify the comports configured in the Modbus server match the generic ports configured in NativeCOM. Once that is confirmed please review items 1~7 of "I can't get the Emulator to work"

"I can connect to an Emulator device but none of its dynamic values are changing"

1. There is no mnemonic configured for the register(s). Please create a mnemonic in the device or device types profile.CSV file. The Emulator must be stopped and the configuration reloaded for any CSV file changes to be accepted. Note that the mnemonic must also exist in the config.txt

2. Verify the mnemonic is configured correctly to be dynamic. Verify min/max are in the correct columns and the value '2' is in the dynamic config column. Please see the section titled "Device CSV profiles" for more information.

3. Check the 'dynamic update rate'. It may be set for a value larger than expected causing an excessive wait for the value to update.

4. Check for mnemonics that have duplicate Modbus addresses. While duplicate addresses are supported they are not recommended as they may 'confuse' the Emulator as to which one needs to be updated. If there is an issue please delete duplicates from both the config.txt and the profile.csv

"If I change a dynamic mnemonic with the Emulator 'Modbus Client' menu item I can see the initial change but after that the value remains static at the changed value"

1. This is normal behavior, changing a value from the Emulator Modbus Client locks out the update routine. To go back to a dynamic update profile press the 'reset' button on the client for that mnemonic.

"How much Ethernet bandwidth does the Emulator use?"

1. The Emulator bandwidth usage is very small, even with the largest of systems. Usage is based on a per Comport being used.. Average usage runs roughly 5Kb/sec per IP port. A 47 port system with over 300 devices on it used approximately 233Kb of bandwidth on a 10Mb LAN. If each of these devices was a Modbus TCP device with each device on a dedicated IP port the expected bandwidth would be about 1.5Mb. This is

dependent upon polling rate, etc. For Modbus IP devices please see Chapter 5 for proper Emulator configuration.

"My non-GE Modbus master is working but I am receiving some Com errors and it indicates there are non-responding devices"

1. Verify the Emulator configuration is correct. Verify the Modbus address's and ports are matched correctly.

2. The Emulators' default setting of its "modbus response delay" is set for a 30ms response delay. Some third party servers require a longer delay. Please increase this setting to see if it helps alleviate the issue

3. Please check the 'health' of the Ethernet connection. An intermittent connection can cause Com issues.

"One or more devices in the Emulator have stopped responding to Modbus requests from the Modbus Master"

1. It is possible that the Ethernet IP port is 'hung' in the OS IP stack. Stop the Emulator and delete and reload the configuration. You may also need to stop and restart the Modbus master.

"When I query an emulated devices clock and date register the value appears to be static, shouldn't the clock be incrementing?"

1. The Clock/Date registers of a device in the Emulator are not tied to the PC clock or any other type of update mechanism. However, the Modbus master should be able to write to write to these registers in the Emulator to update them in runtime. Note that when the Emulator is shutdown it will not retain these updates and will revert to the profile CSV values upon restart. If needed the user could create a VB app or excel app to perform a DDE 'poke' of the correct time and date to the appropriate mnemonics/registers.

”The 32bit IEEE floating point registers in an emulated ‘device’ does not yield correct values”

1. Verify that the floating point register types of the Modbus Master and device type match. For PMCS systems the “@Float1” format is most used. This is also the fixed IEEE floating format for any generic based devices in the GE PMCS Modbus servers. If a generic device uses the @Float2 format it cannot be configured as a generic device in the GE PMCS Modbus servers. Please see Chapter 4, section titled “@FLOAT 32bit IEEE Floating Number Function” for more info on the “@Float1” and “@Float2” formats.

2. If using the “@Float2” format please open the device profile csv file with ‘notepad’ and check of any characters after the “@Float2” entry. Delete any if found and save as a csv file.

Appendix A – Modbus Protocol Reference

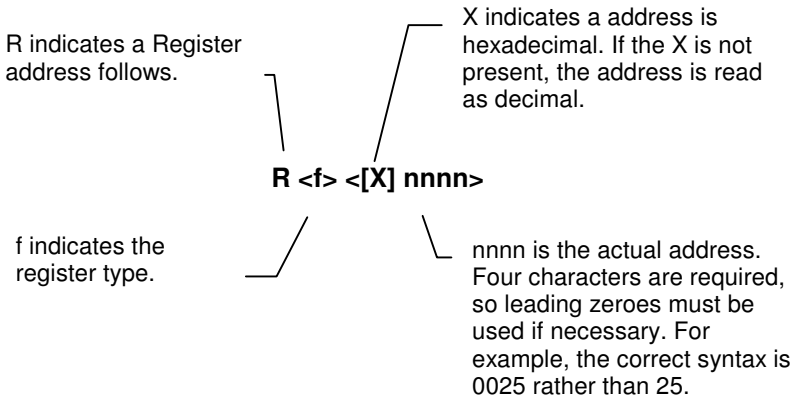
Please reference first two chapters of GE PMCS manual GEH-6508.pdf “Modbus Concentrator Protocol Reference” included in the documents section of the Emulator directory as well as the documents directory on the Emulator CD.

Appendix B – Emulator Modbus Register Addressing Conventions

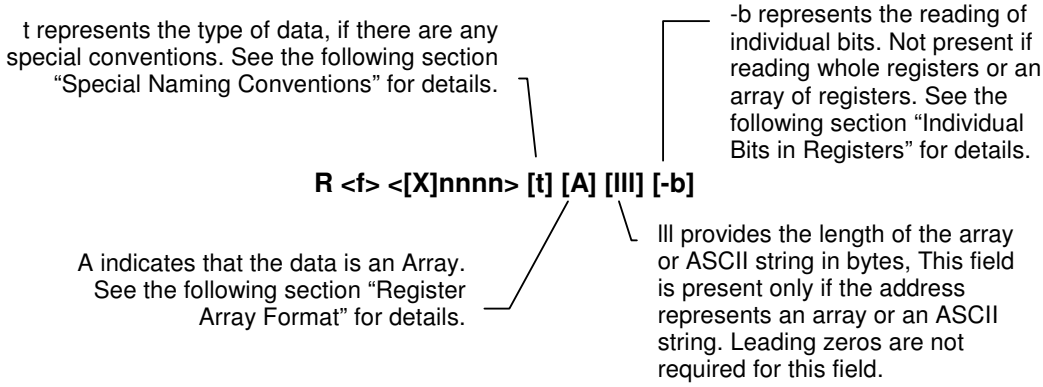
Data-Addressing Conventions

The PMCS DDE Server and Emulator are capable of interpreting both decimal and hexadecimal addresses. This allows access to the Modbus RTU/IP protocol’s extended register mode. The two addressing schemes are identical with one exception; in hexadecimal mode, an “X” is inserted prior to the address number to indicate that the address following is in hexadecimal format. The R character is ALWAYS present. Items in < > represent a variable numeric value. Capital letters in brackets [ ] indicate a hard character that does not change; it is either present or not. Lower-case letters in brackets [ ] indicate switches that may or may not be present; refer to the following sections for details.

The basic addressing scheme is as follows:



Switches may be used to modify addresses. The possible switches are shown below, and are detailed in the following sections:



Standard Data Organization

Data is organized according to data type, numeric range, tag type, and access type.

Data Types

There are four data types typically used by the GE devices. These four data types are the possible values for ‘F’ in the address.

(Each data type is organized in a separate table for each device in this manual):

1. Dynamic Value
2. Setpoint
3. Command Coil
4. Fixed Value

Each data type is assigned a range of register numbers, tag type, and access as shown below:

Emulator Data Organization					
Data Type	**Use**	**Register Range (hex)**	**Register Range (decimal)**	**DDE Tag Type**	**Type of Access**
Command Coil	1. Commands a device to take action. 2. Reads the status of an action or discrete input.	R0X0000 – R0XFFFF	R00000 – R09999	Discrete	Read and Write
Contacts	Reading contact/discrete inputs	R1X0000 – R1XFFFF	R10000 – R19999	Contacts/ Discrete	Read and Write
Dynamic Value	Read frequently, such as metering values which change constantly	R3X0000 – R3XFFFF	R30000 – R39999	Analog	Read Only
Fixed Value	Read only once at power-up. Info such as Product ID and configuration options	R4X0000 – R4XFFFF	R40000 – R49999	Analog	Read Only
Setpoint	Read/Set infrequently	R4X0000 – R4XFFFF	R40000 – R49999	Analog	Read and Write

Exa**mples**

Here are some examples of different types of register numbers:

Register number	**Represents**
R00005	Coil command, number 5, with Read/Write access to the user
R31005	Dynamic value, number 1005, Read Only access
R43010	Fixed value or Setpoint, number 3010, with Read /Write access to the user

Register Types

The four types of register groups that support some of the Modbus function codes are R0, R1, R3, and R4. The table below describes the types, registers, supported codes, and uses.

Type Code	Type of Register	Supported Function Codes	Use
R0	Coils	01	Reading coil status
R0	Coils	05	Setting/forcing/ executing coils
R0	Coils	15	Setting/forcing multiple coils
R1	Contacts or discrete inputs	02	Reading contact/discrete inputs
R3	Actual value or input register	04	Reading actual value or input registers
R4	Setpoint or holding register	03	Reading setpoint or holding registers
R4	Setpoint or holding register	06	Presetting single setpoint register
R4	Setpoint or holding register	16	Presetting multiple registers

You will need the device's Modbus RTU/IP protocol specification for the correct register formatting and functions codes.

Special Naming Conventions

Special handling of data from devices can be done by using the following conventions:

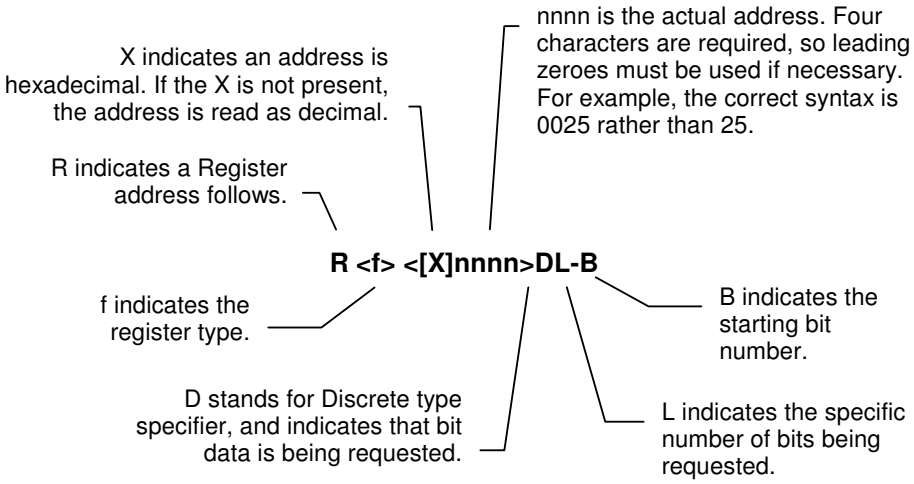
Long Words and Special Numbers

By default, a register item is treated as an unsigned integer. To treat the contents of any register differently, refer to the table below:

Special Data Item	Naming Convention	Example
Unsigned 16-bit Integer	Default	R40001
16-bit Signed Integers with values between -32,767 and 32,767	Append letter I to item number.	R40001I
32-bit Signed Integers (Long Integers)	Append L to item number. Comprised of two 16bit registers.	R40001L
32-bit floating point numbers	Append F to item number. Comprised of two 16bit registers. IEEE floating point standard applies.	R40001F
Modulus 10000 Used in 32-bit register mode for GE EPM 3710 and 3720 meters.	Append E to the item name ***NOTE:*** This format **cannot** be used by the Emulator since it does NOT support 32-bit register mode, Only 16bit registers are supported. See EPM 37xx workaround in chapter 5.	R40010E
ASCII data string	Append S to item number. [lll] field immediately after S character represents the number of characters to read. If no length is specified ([lll] field is not provided), only one register of characters (2 or 4) will be read. The High byte represents the first character, and the Low byte represents the second character. ***NOTE:*** No array type is allowed with S data items, nor are ASCII strings supported for coil registers. ***NOTE 2:*** For 16-bit mode devices (most devices) there are 2 characters per register. For 32-bit mode devices, there are 4 characters per register (not used by the Emulator). ***NOTE 3:*** The maximum value for the S string is 250.	R40010S020
Array Format	Append A to item number. Up to 100 sixteen-bit registers or 50 thirty-two-bit registers can be read as a block. Enter the starting register address, and append it with type specifier “A”, followed by the length field.	R30501A12

Individual Bits in Registers

Individual bits in registers can be read/written to in the Emulator as discrete mnemonic tags by using the below register notation in the devices register definition in the devices ‘profile.csv’ file. In addition the Emulator also supports discrete writes from the Modbus master (3rd party masters only, the GE PMCS Modbus servers do not support discrete writes).



Examples

Register Number	Represents
R40001D1-0	Specifies least significant bit of first holding register
R30008D1-15	Specifies most significant bit of an input register
R40001D2-5	Specifies 6th and 7th from the least significant bit of first holding register.

Register Array Format

If multiple data items are being requested from a single topic, it is more efficient to request a block of contiguous registers than to place multiple requests for single registers. This is referred to as *register array format*. The register array format is used for the following types of applications:

- To read a block of register values into/from a column of cells in a worksheet (such as Microsoft Excel).
- to pass waveform data to a client application (refer to GEH-6509, PMCS DDE Server Interface Reference, for details)

The rules for register arrays are as follows:

1. A register array, or series of consecutive registers, can be treated as a block of numeric values. Up to 100 sixteen-bit registers or 50 thirty-two-bit registers can be read as a block. Enter the starting register address, and append it with type specifier “A”, followed by the length field. For example, the register address R30501A12 accesses registers 501 through 512 as a block.

2. When using the Emulator ‘Modbus client’ to write a value to a register array, it must be in the form of a character string containing a value for each register in the array. The register values must be separated by spaces. For example, for R40001A6, the value string could be written in the profile csv file or in the Emulator ‘Modbus Client’ as:

   1 2 3 4 5 6

   Each individual value will be contained in a specific Modbus register.

   R40001 = 1
   R40002 = 2
   R40003 = 3
   R40004 = 4
   R40005 = 5
   R40006 = 6

   Note: The Emulator does NOT support array writes from any Modbus Master.

(This page left blank intentionally.)

GE Consumer & Industrial

General Electric Company
215, Anderson Avenue, Markham, ON, L6E 1B3, CANADA
USA & CANADA: 1-800-547-8629 Global: (905) 294-6222 Fax: (905) 201-2098
Email: multilin.tech@indsys.ge.com

DEH-XXX