Modbus protocol Tester User Manual

Revision 2.0
March 2018

Contents

1 Introduction

What is Modbus Protocol Tester?

The major functional blocks of the Modbus Protocol Tester include the following:-

Modbus Master. Sends the most common Modbus messages.

Device Simulator. Is a software Modbus Slave. Responds with messages exactly like a physical device.

Controls. To organize test environment, connect and test.

Counters And Logs - Data collectors capture readings. Logs record events.

Displays - Display test data in format from selection of choices.

Why Was It Created

The Modbus Protocol tester is designed to deliver improved testing efficiencies. It is based on years of practical experience.

Who Should Use This Tool

Software Developers - Embedded Applications

The Tester is a handy tool for developers and Test Engineers to speed up their product development cycles:-

- Verify Modbus Specifications compatibility
- Benchmark device performance
- Stress the communication interface with bursts of messages
- Quickly organize comprehensive test cases.

Field Engineers - Instrumentation and Control

Extremely versatile for engineers and technicians working on live installations:-

- Transport Tester on a pen drive
- Organize poorly documented installations logically
- Quickly isolate problems and determine root cause.
- Comprehensive testing before commission

1.1 Features

- Supports Modbus TCP mode.
- Can be configured to control and monitor one Modbus TCP device. Multiple instances of the MPT can be used to communicate with multiple Modbus TCP devices.
- Presents a Multiple Document Interface using which multiple Data Groups can be configured for data acquisition from the device at different scan rates. Useful for creating logical data groups as well as presenting them in independent forms.

- All configurations can be serialized to a file. This enables storage of pre-configured groups that can be opened quickly when doing regression testing.
- Interface to change the status/value of a single or multiple Coils/Holding Registers. Can be used to test both "Single Write" and "Multiple Write" function codes of Modbus specification.
- Communication can be configured to reconnect to a device in case of drop in connection. Useful when doing long duration unattended testing of devices.
- Provision to display Modbus traffic (i.e. The transmitted and received packets) with an option to timestamp the same. Very useful in debugging unstable connections.
- User friendly, flexible and intuitive User Interface that provides user control over all visual parameters like fonts, colors, Window sizes, etc. All such customized parameters are saved on exit allowing for recall of the same on the next startup.
- Extensive event logging provides a powerful tool for debugging Modbus device connectivity issues both during development as well as deployment of field.
- Small footprint application enables field engineers to carry them in their pen drive and run it on any Windows computer.
- Light weight, multi-threaded design allows bombarding devices with Modbus requests to test their ability to handle bursts of Modbus requests.
- Displays real time communication statistics like the no. of requests sent, responses received, errors etc.
- Data interpretation layer (over and above the Modbus layer) permits interpreting and displaying Modbus register data as short and long integers, unsigned integers, float, double etc.

1.2 **Benefits**

Modbus Protocol Tester features, usage and benefits are summarized below.

Usage	**Enabling Feature**	**Advantage / Benefit**
Test complex configurations	Several instances of Modbus Protocol Tester can run simultaneously	Fast test cycles. One-pass comprehensive test of complete environment
Isolation and diagnosis of network problems	Ability to profile and manage individual devices	Fast regression testing.
Stress testing	Rapidly transmits and receives Modus messages	Increase product robustness.
Long duration testing	Reconnects automatically after communication interrupt	No need to repeat tests due to communication interrupt
Metrics capture for understanding communication delays	Timestamp traffic and display data at individual packet level	Debug unstable environments
Functional compatibility testing	Supports Modbus function codes: 01, 02, 03, 04, 05, 06, 15, 16	Assure Industry standard compliance
Test field installations	Small size software: 2.6 MB fits on pen drive	Carry software, not computer
Quick check connections and communications	Real time display of communications statistics	Immediate visibility to communications
View data in format of choice	Data Interpretation Layer offering	Data uniformity

	format choices	
Diagnose communication problems. Verify test execution. Compile test reports.	Event Logger. Time stamps and captures events throughout test cycle	Event log provides exact behavior of communications network: faster problem isolation and fix.

2 Getting Started

How To: Start Using Modbus Protocol Tester

This section will enable you to install MPT, quickly learn the basics and get you going with your first project in MPT.

2.1 How To: Install Modbus Protocol Tester

To install MPT on your computer simply run *SetupMSTH.msi* and follow the on-screen instructions. The only option to choose is the installation folder for the application. You would typically install this in the computer's *Program Files* folder. The setup program installs the following components on to your disk:

- The Modbus Protocol Tester application.
- A Modbus TCP Slave Device Simulator which you can use as a device while learning to use the MPT.
- The User's Manual for the Tester.

Subsequent to the installation, the above three components can be accessed using the Windows Start Menu at **Start Menu -> All Programs -> Modbus Protocol Tester**

Pre-requisites

The following are the pre-requisites for being able to install/run MPT:

- Operating System: Windows XP SP2 or Windows Vista (any version)
- A Network Interface Card (wired or wireless) for using MPT in Modbus TCP mode
- .NET Framework 3.0 or higher (can be downloaded in **Microsoft Download Center**)

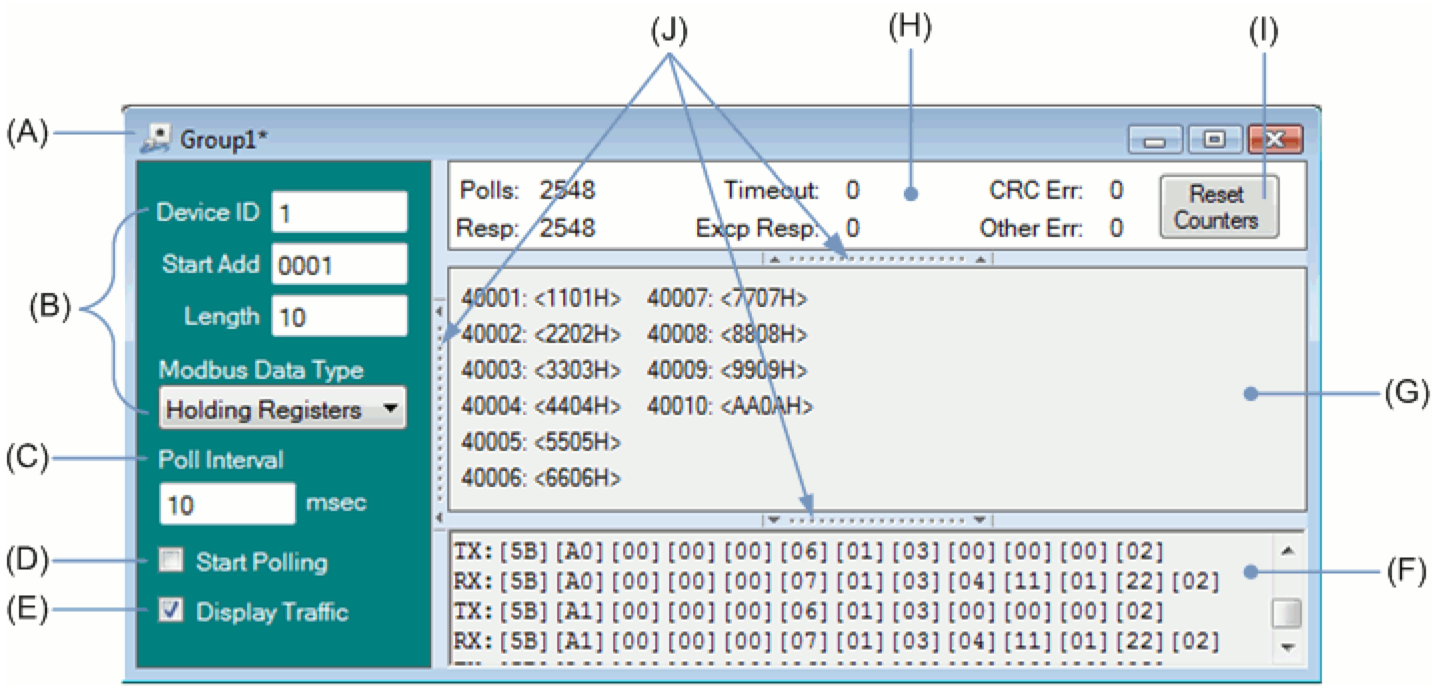
Important:

The MPT will not be able to connect to a Modbus TCP device if Firewall is enabled on your computer. Either disable Firewall or add Modbus port no. 502 under the Exceptions List of your Firewall.

2.2 Interface Basics - Overview: Modbus Data Group Form

- The Modbus Data Group form is the Window through which you view the data acquired from the Modbus device as well as change values of (i.e. Write to) tags in the device. Using this form you can:
- periodically acquire data from your Modbus device at a configurable poll interval
- Group data tags (but of the same data type) and view their latest acquired values
- Format the display to show the data in various formats (float, hex etc.)
- Change values of (i.e. write to) data tags in your device
- View raw Modbus packets being transmitted to and received from the device
- Track communication status by way of a set of *Event Counters*

The Data Group Forms can be saved (along with all the settings made in them including the size and position of the Window) and reopened at a later time.



Device ID: 1
Start Add: 0001
Length: 10
Modbus Data Type: Holding Registers
Poll Interval: 10 msec
Start Polling: ☐
Display Traffic: ☑

Polls: 2548 Timeout: 0 CRC Err: 0
Resp: 2548 Excp Resp: 0 Other Err: 0

```
40001: <1101H>   40007: <7707H>
40002: <2202H>   40008: <8808H>
40003: <3303H>   40009: <9909H>
40004: <4404H>   40010: <AA0AH>
40005: <5505H>
40006: <6606H>
```

```
TX: [5B] [A0] [00] [00] [00] [06] [01] [03] [00] [00] [00] [02]
RX: [5B] [A0] [00] [00] [00] [07] [01] [03] [04] [11] [01] [22] [02]
TX: [5B] [A1] [00] [00] [00] [06] [01] [03] [00] [00] [00] [02]
RX: [5B] [A1] [00] [00] [00] [07] [01] [03] [04] [11] [01] [22] [02]
```

A: Title Bar

The title bar displays the name of the Data Group Form. If the form is newly created (i.e. not yet saved) a default file name of the form **Group *N*** where 'N' is a running serial number (e.g. Group1, Group4 and so on). If a form has not been saved or if some changes have occurred in the form after it was last saved, a star (*) is suffixed to this name.

B: Modbus Communication Parameters

- This set of parameters defines the Modbus data that will be acquired and displayed by this form.
- **Device ID**: The Modbus Slave (Unit) ID of the device that this form will communicate with. This field accepts any value between 1 to 247. Note that most Modbus TCP devices will ignore the Slave ID field in a Modbus TCP packet.
- **Start Add**: One Data Group Form can acquire and display data of any one type from a device. This field specifies the address of the first Modbus data item whose value will be acquired & displayed by this form.
- **Length**: is the number of Modbus data items starting at address **Start Add** whose values will be acquired & displayed by this form.
- **Modbus Data Type**: Users can choose the data type from four available choices - Coils, Discrete Inputs, Holding Registers Input Registers. Notice that one form can display data of one type only.

C: Poll Interval

The periodic interval at which this form sends a Modbus request to acquire data for the configured Modbus addresses.

⚠**Caution:**

Data acquisition is not guaranteed to happen precisely at this interval as the device being communicated to may have a much larger response time than the value of this parameter. A few other factors also affect the rate at which data can actually be acquired from the device - the network load, the number of Data Group Forms competing to communicate with the device and so on.

Note:

The controls for Modbus Communication Parameters and Polling Interval get disabled once polling is started. To change their values disable/stop polling first.

D: Start Polling

Checking this checkbox will cause the Data Group Form to start periodic data acquisition from the device. Unchecking it will stop the data acquisition.

Note:

- The **Start Polling** checkbox is enabled only if a connection has been made to the device.

- If an existing connection breaks (dropped by the device or breaks due to network disturbances) the Data Group forms stop the data acquisition even though **Start Polling** is checked. Data acquisition resumes automatically when a new connection is made to the device.

E: Display Traffic

Check this checkbox if you intend to see the raw Modbus packets being transmitted by the form to the device and the response packets received from the device.

F: Traffic Display Panel

This panel displays the raw Modbus packets being transmitted by the form (prefixed with *TX:*)to the device and the response packets received from the device (prefixed with *RX:*). Each byte of the packet is displayed as a hexadecimal number within two square brackets [ ].

G: Data Display Panel

This panel displays the current value of all the configured Modbus addresses in the currently selected **Data Format**. **More on the Data Display Panel.**

H: Event Counters, Panel

This panel displays a set of event counters to indicate the health and status of communication of this Data Group with the Modbus device. A brief description of these counters follows :>

- **Polls**: The total number of Modbus requests sent out by this data group. When polling is enabled this counter will increment roughly at the rate of the set polling frequency.
- **Resp**: Short for "Responses". The total number of valid Modbus responses received by this data group (excluding Exception Responses). If the communication is error free, this counter should follow the **Polls** counter.
- **Timeout**: The total number of Modbus requests for which the Data Group received no responses from the device. Ideally this counter should be zero. Possible reasons for the occurrence of timeouts are:
    1. Use of wrong device ID (one which does not match any devices on the network.
    2. Too small a timeout setting in Communication Setup form. This will cause the Tester to see no response even if the device did respond (but after the timeout period).
- **Excp Resp**: Short for "Exception Responses". The total number of Modbus requests for which the device responded with an Exception response. Exception responses indicate that the device does not support the requested function code or that at least one address in the requested address range does not exist on the device. While a non-zero value of this counter does not indicate a problem in the communication per se, it does indicate an error in the configuration.

- **CRC Err**: Short for "CRC Errors". This counter is unused in Modbus TCP communication and so is reserved for future use.
- **Other Err**: Short for "Other Errors". The total number of Modbus requests for which an unknown error occurred when trying to read a response.

I: Reset Counters Button

- This button is used to reset counters of this form to zero. You will use this when you start a test case afresh probably after fixing some error in the configuration or the setup.

J: Splitter Bars

- The splitter bars are used to reorganize the panels on this form to get an optimal view of all the information shown on the form. **How to use the splitter bars to customize the Data Group Form.**

2.3 Overview: Modbus Device Simulator

Overview: Modbus Device Simulator

The Modbus TCP Device Simulator is a Windows console application that can be used as a Modbus TCP device for you to get started with using the Tester. It has the following capabilities:

- ➢ Simulates a Modbus TCP device by listening for connection requests on port 502.
- ➢ Supported on Windows XP and Windows Vista.
- ➢ Supports the following Modbus Function Codes:
- ➢ Read Coils (FC 01)
- ➢ Read Discrete Inputs (FC 02)
- ➢ Read Holding Registers (FC 03)
- ➢ Read Input Registers (FC 04)
- ➢ Write Single Coil (FC 05)
- ➢ Write Single Register (FC 06)
- ➢ Write Multiple Coils (FC 15)
- ➢ Write Multiple Registers (FC 16)
- ➢ Simulates a database consisting of Modbus points of all four types (Coil, Discrete Input, Holding Register, Input Register). The holding registers are further organised so that they can be read in the following formats:
- ➢ Hexadecimal (2-byte value). E.g. 0x8ABC
- ➢ Short Integer (2-byte signed decimal value). E.g. -30020
- ➢ Unsigned Short Integer (2-byte unsigned decimal value). E.g. 35516
- ➢ Integer (4-byte signed decimal value). E.g. -12345678
- ➢ Unsigned Integer (4-byte unsigned decimal value). E.g. 12345678
- ➢ Float (4-byte IEEE 754 single precision real value). E.g. 124.35

How to launch the Device Simulator

You can launch the simulator using *one of two ways*:

1. Choose menu item **Tools->Start Device Simulator->Modbus TCP** from within the Tester
2. Using Windows Start Menu: Go to **Start Menu->All Programs->Modbus Protocol Tester** and select **Modbus TCP Slave Simulator**

Address map of the Device Simulator

The device simulator has a virtual internal database as below:

Address Range	Data Type	Startup Value
0 - 15	Coil	0,0,1,1 | 0,0,0,0 | 1,1,1,1 | 0,1,0,1
0 - 15	Discrete Input	1,1,0,0 | 1,1,1,1 | 0,0,0,0 | 1,0,1,0
0 - 9	Holding Registers	0x1101, 0x2202, 0x3303, 0x4404, 0x5505 0x6606, 0x7707, 0x8808, 0x9909, 0xAA0A
0 - 9	Input Registers	0x2202, 0x4404, 0x6606, 0x8808, 0xAA0A 0x1101, 0x3303, 0x5505, 0x7707, 0x9909
20 - 29	Holding Register (Float)	12.34f, 56.78f, 278.94f, 981.23f, 32.97f
40 - 49	Holding Register (4-byte Unsigned Integer)	12345678, 22114433, 33998877, 22112211, 0x55eeff66
60 - 63	Holding Register (String)	'COLWAY'

2.4 How To: Set Up Project in Modbus Protocol Tester

Follow these steps to start using MPT with your first Modbus device.

1. Launch Modbus Protocol Tester
2. **Setup communication parameters by choosing Communication->Setup menu**
3. **Make a connection to your device by choosing Communication->Connect menu**
4. Create a new Modbus Data Group (MDG) Form by choosing **File->New** menu
5. **Setup the Modbus Data Group Form for Data Acquisition**
6. Start data acquisition by checking **Start Polling** in the MDG form

3. Modbus Device Connection

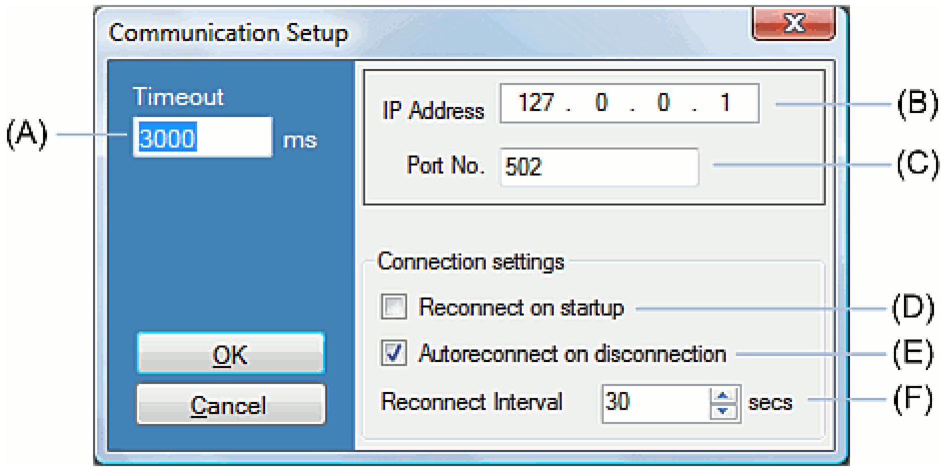
Connecting to a device requires setting up of communication parameters to be used for making the connection. This section gives you an overview of the Communication Setup form and procedures for making a connection to a Modbus TCP device.

3.1 How To: Set Up Communication Parameters

Before making a connection to a Modbus device there are a few parameters that must be setup. To do this, the Tester provides the **Communication Setup** form. This form can be opened by choosing **Communication->Setup** menu item or by clicking the Communication Setup Toolbar button .

Note:

If the Communication Setup form is opened when an active connection to the device exists, the form is displayed in a *read-only* mo values of settings. This means that connection parameters can be changed only in a *disconnected* state.



Timeout: 3000 ms

IP Address: 127 . 0 . 0 . 1

Port No. 502

☐ Reconnect on startup

☑ Autoreconnect on disconnection

Reconnect Interval: 30 secs

A: Timeout

This is the amount of time the Tester will wait to get a response from the device after it has sent a Modbus request to it. If no response is received within this time, a "timeout" is considered to have occurred and the transaction is aborted. The value specified is in millisecond.

B: IP Address

The IP address of the device to connect to.

C: Port No.

The TCP port number on which the device services Modbus TCP connection requests. Port 502 is the Modbus standard port. There are only a handful devices which use a non-standard port number. If so, the port number used will be provided in the device's User Manual.

D: Reconnect on startup

Enabling this feature will cause the Tester to attempt connection to the device on startup. The connection will be made as per the last configured settings.

E: Autoreconnect on disconnection

Enabling this feature will cause the Tester to attempt reconnection whenever an existing connection to the device is dropped due to network or device problems. Auto reconnection is not made if the user manually disconnects using the **Disconnect** menu item or the toolbar button. The reconnection attempts will continue in the background at a periodicity set by the "Reconnect Interval" field (see below) until the connection is successful or the user disables this feature.

Note:

This feature is extremely useful when doing unmanned long duration testing on devices. If during such a test the connection is drop
reconnect to the device when the device reboots thus maintaining the continuity of the test. In the absence of this feature the test
reconnection events is logged into the event log file which can be analysed at the end of the test to detect if there was a communic

F: Reconnect Interval

This field is related to the "Autoreconnect on disconnection" field and is used to specify how often the Tester will attempt to reconnect to the device if the last attempt was unsuccessful.

3.2 How To: Set Up Communications for Modbus TCP

Procedure

1. Open the **Communication Setup** form by choosing **Communication->Setup** menu item. This form can be opened by clicking the Communication Setup Toolbar button too.
2. Enter the IP address of your Modbus TCP device. This information should be available from your device's User Manual.
3. Change the Port Number from the default value of 502 if your device services connection requests on a different port number. The standard port number for Modbus TCP connections is 502.
4. Set a communication timeout period in the **Timeout** text box. Exercise caution in setting this parameter as a value too small will cause timeouts even though the device is responding correctly to Modbus requests. Device response times greatly vary from anywhere between 10ms to tens of seconds depending on the speed of the device's microcontroller and the medium of communication. For direct Ethernet connections a typical value of timeout is between 1 second to 5 seconds. However for slower media like GPRS you may have to set this to anywhere between 10 seconds to even a minute.
5. If you want the Tester to automatically reconnect to the last configured device on startup, *check* the **Reconnect on startup** checkbox. This will save you the hassle of manually reconnecting every time the test harness is started.
6. If you want the Tester to automatically reconnect to the device if an existing connection drops due to network problems, *check* the **Autoreconnect on disconnection** checkbox. Set an interval for periodic retries if the reconnection attempt is unsuccessful in the **Reconnect Interval** text box. It is strongly recommended that this feature be enabled prior to starting long duration stability tests on your device.
7. Click **OK** to save the configuration.

Caution:

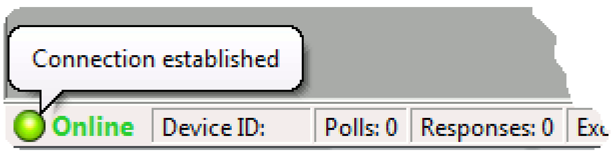
One of the most common causes of failure in Modbus TCP communication is to leave the Firewall enabled on the selected Modbus p
Firewall. Remember to disable the Firewall or add the configured Modbus port number (usually 502) under the Exceptions List of yo

3.3 How To: Connect Modbus Device

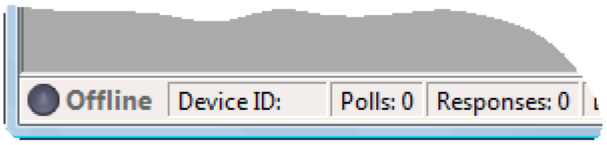
Procedure

1. Setup communication parameters to be used to connect to your device.
2. Choose **Communication->Connect** menu item or click the **Connect** toolbar button to connect to your device.
3. Verify that the connection was successful.
4. If the connection was successful, the status is reflected in the status bar as below.

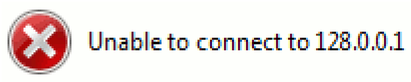
Successful connection


Connection established
Online | Device ID: | Polls: 0 | Responses: 0 | Exc

Connection failure


Offline | Device ID: | Polls: 0 | Responses: 0

5. If the connection was unsuccessful, a pop-up error message is also displayed as below.


Unable to connect to 128.0.0.1

Notes:

The **Connect** operation attempts to make a TCP connection to the Modbus device. Under different circumstances, a connection attempt may fail after several seconds of initiating it. The **Connect** toolbar button and the menu item are both disabled until the connect operation is completed, successfully or not.

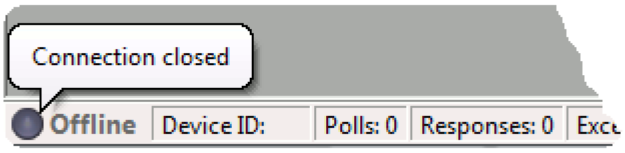
3.4 How To: Disconnect Device

Why would I want to break the connection to my device?

You might want to disconnect under the following circumstances:
 You want to connect to a different device.
 You want to change the communication parameters. These parameters cannot be changed when a connection is active.
 The device under test has stopped responding to requests and you want to retry with a fresh connection.

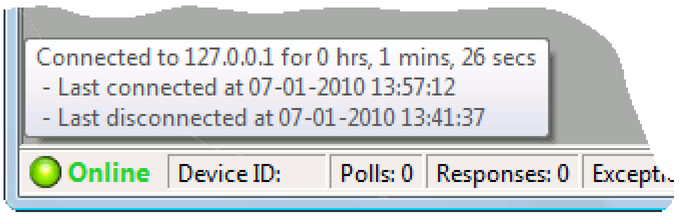
Procedure

- Choose **Communication->Disconnect** menu item or click the **Disconnect** toolbar button to disconnect from your device. A successful disconnection is indicated with a balloon on the status bar as below.



3.5 How To: View Connection Status

Hover your mouse pointer above the "Online" status label on the status bar. A tooltip appears providing details about the currently active connection as shown below.


Connected to 127.0.0.1 for 0 hrs, 1 mins, 26 secs
- Last connected at 07-01-2010 13:57:12
- Last disconnected at 07-01-2010 13:41:37

What information does this tooltip provide?

 Connected to: This displays the IP address of the device to which this connection has been made.
 Connected Duration: The duration in terms of total hours, minutes and seconds for which this connection is active.
 Last Connected At: The time at which this connection was made to the device.
 Last Disconnected At: The time at which the previous connection to the device was dropped.

Tips:

The Connected Duration, Last Connected and Last Disconnected information elements can be used to know if a connection was dropped in the midst of a long term test on your device as well as the duration for which the connection was down.

Tips:

The event log provides a lot more information about connection related events over a period of time. The **connection details** tooltip must be used as a quick reference to know the connection quality while the event log must be used for detailed analysis.

4. Acquire Data from a Device

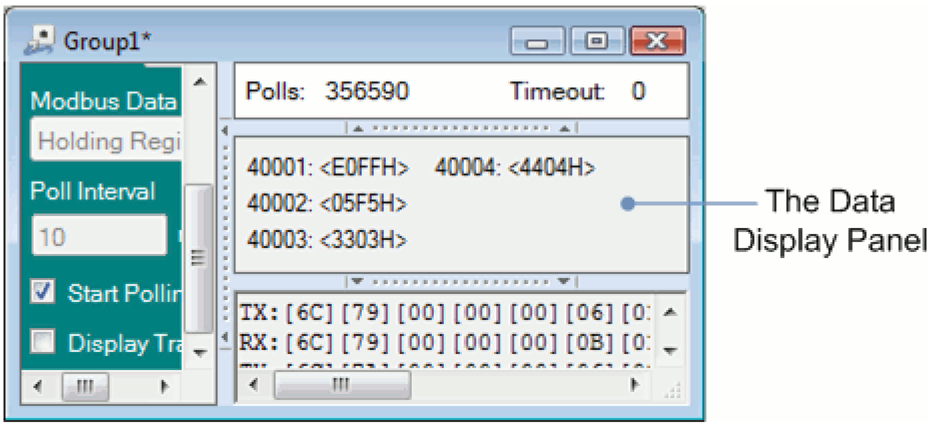
Data is acquired in the Modbus Protocol Tester using Modbus Data Group (MDG) forms. Each MDG form can be configured to acquire data of any one type (i.e. Coils, Discrete Inputs, Holding Register or Input Registers). So it is likely that your project would have several MDG forms each acquiring data for a related set of Modbus tags.

4.1 Overview: Data Display Panel

The Data Display Panel (DDP) is that area of the Data Group Form in which the data acquired from the Modbus device is displayed. The DDP:

- displays formatted data as per the current format selection.

- can be used to change the value of a single Modbus variable at a time (using WriteSingleCoil and WriteSingleRegister function codes)



```
40001: <E0FFH>   40004: <4404H>
40002: <05F5H>
40003: <3303H>
TX: [6C] [79] [00] [00] [00] [06] [0:
RX: [6C] [79] [00] [00] [00] [0B] [0:
```

Note:

The display for each Modbus address has two components. The *address* field and the *value* field. The *address* field is always display with the data type code as defined by the Modbus standard.

- 1 for Discrete Inputs

- 2 for Coils

- 3 for Input Registers

- 4 for Holding Registers

In what format is the data displayed?

Coils and Discrete Inputs are always displayed as binary values - '1' if the Coil/DI status is Hi or '0' if Low.
However register data are displayed in one of the following formats:

- Hexadecimal (as 2-byte value). E.g. 0x8ABC
- Short Integer (2-byte signed decimal value). E.g. -30020
- Unsigned Short Integer (2-byte unsigned decimal value). E.g. 35516
- Integer (4-byte signed decimal value). E.g. 12345678
- Unsigned Integer (4-byte unsigned decimal value). E.g. -12345678
- Float (4-byte IEEE 754 single precision real value). E.g. 124.35
- Double (8-byte IEEE 754 double precision real value). E.g. 124.35

4.2 How To: Set Up MDG Form For Data Acquisition

1. Start with a Modbus Data Group Form. You can create a new form or open an existing one.

 To Create a new **Modbus Data Group Form** choosing **File->New** menu item, click the **New Data Group** toobar button or type the CTRL+N shortcut key combination.

 To open an existing **Modbus Data Group Form** choosing **File->Open** menu item, click the **Open Data Group** toobar button or type the CTRL+O shortcut key combination and browse to/select the existing form (file name extension *.mdg).

2.Configure the Modbus Data Group Form to reflect the type of data to be acquired and displayed.

1. Set **Device ID** to the unit/slave ID of the device from which this form should acquire data. However note that most Modbus TCP devices ignore this field and will respond to requests
with any device ID except zero.
2.Choose an appropriate **Modbus Data Type** reflecting the nature of data to be acquired from the device. This information is generally published in the users manual as the *address map*
of your device. The Tester decides on the Modbus Function Code to be used for the requests based on this parameter. Selecting an incorrect data type will result in an Exception
Response from the device.
3.Choose the **Start Address** and **Length** fields to define the range of Modbus addresses for which data is to be acquired. Select this range to reflect a logical group of addresses in your
device. While the form permits configuration of a large number of addresses, it is suggested that all the addresses of interest be grouped and each group be displayed in different MDG
forms. This makes it easier to view and analyse the acquired data. If any of the addresses in the configured range does not exist in the device, an exception response is received from
the device causing data for existing Modbus addresses also to be unavailable.

Important:

The Start Address entered should be one more than the address as specified in the *address map* of the device. This is because the T
using it in encoding Modbus packets as required by the Modbus specifications.

4.Enter a duration (in milliseconds) for **Poll Interval** to set a periodic polling cycle. Setting too low a value can cause the device to be overloaded with Modbus requests and setting too
high a value can cause data to be sluggishly updated on the display.

3. Start data acquisition by *checking* the **Start Polling** checkbox or by clicking the right mouse button on the Data Display Panel and choosing the **Poll Device**
menu item in the context menu.

Tips:

If you have multiple Data Group Forms open and would like to start or stop polling in all of them, you can use the **Start Poll All Data**

Groups toolbar buttons to do this.

Verify that data acquisition is working fine.

 If the address range and data type configured in the Data Group Form exist on the device, you should see data being displayed on the form. If the form is unable to acquire data, it displays ???? in place of data.

 The upper section of the Data Group Form shows a set of **Poll Counters**. If data acquisition is working correctly the **Polls** counter must be equal to **Resp**(response) counter and the other remaining counters should not increment.

Notes:

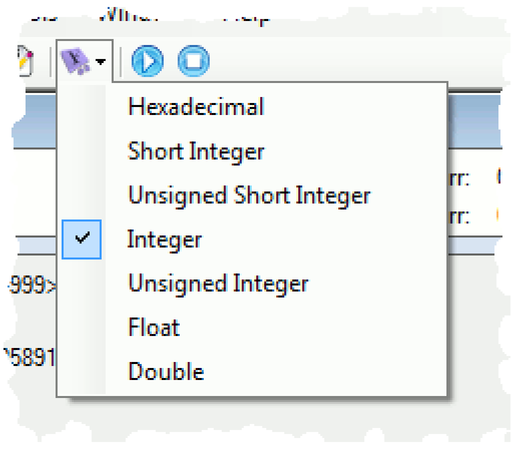
- You can create/open multiple Data Group Forms to view different sets of data. Each can acquire data at its own polling frequency.
- Stability of a device's communication interface under heavy load conditions can be tested by setting a very low value (<10ms) for **Poll Interval** so as to bombard the device with Modbus requests.
- Incorrect setting of **Modbus Data Type**, **Start Add** and/or **Length** can cause the device to respond with an Exception Response.

4.3 How to Change Data Display Format?

The format can be changed using one of two ways:

- Using the toolbar button

1. Select the Data Group Form whose data format is to be changed by clicking anywhere on the form.
2. Click the "Change Display Format" toolbar button. A drop down list box appears as below showing the current

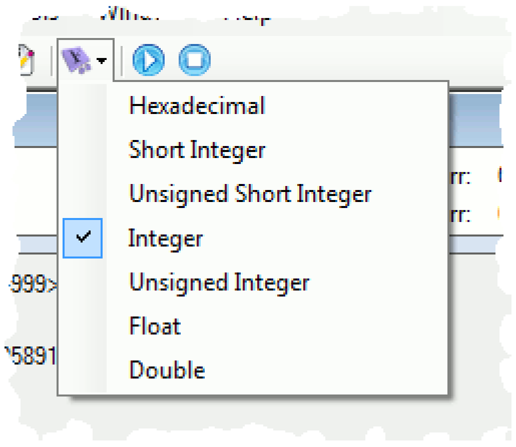
format with a tick mark 

.

3. Select the desired data format by clicking a relevant list item.

- Using the right-click pop-up menu

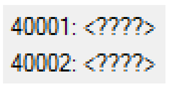
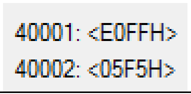
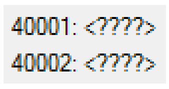
1.Right-click on the Data Display Panel of the Data Group Form whose data format is to be changed. A pop-up menu appears as shown below showing the current
format with a tick mark.



2.Select the desired data format by clicking a relevant pop-up menu item.

How is data displayed when no data is being polled or when an active connection is dropped while polling is in progress?

When a Data Group is first created and has not yet acquired any data from the device, the value fields for all configured addresses show question marks "?????" to indicate that they do not have data worth showing. However, if atleast one data set was acquired and then the polling was stopped, the DDP will display the last acquired data for all the addresses. However if a connection is dropped when polling was in progress, the display changes to "?????".

Freshly opened Form	Polling Off but atleast one data set acquired	Connection dropped while polling
 40001: <????> 40002: <????>	 40001: <E0FFH> 40002: <05F5H>	 40001: <????> 40002: <????>

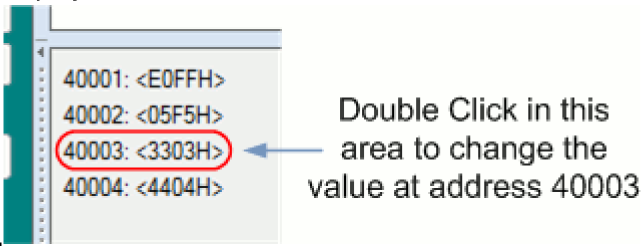
5. Write Data to Modbus Device

The Tester provides forms for changing the state of Coils as well as writing new values to Holding Registers. You can write to a single Coil or a Holding Register in your device using the Data Group Form but this method cannot be used to change the value/state of multiple Holding Registers/Coils at once. For this you will have to use the **Write Holding Register** and **Write Coils** forms respectively.

5.1 How To: Change Modbus Variables

You can change values at one address at a time using the Data Display Panel of an MDG form.
The *WriteSingleXXX* function codes are used for writing to the device using this method. If you intend to change the values of multiple addresses at a time use the **Write->Coils** and **Write->Holding Registers** menu options instead.

1. **Setup a MDG form to acquire data for the addresses of interest from the Modbus device.**
2. Using the left mouse button, within the Data Display Panel, double click the *address* or the *value* part of the Modbus variable whose value is to be changed.

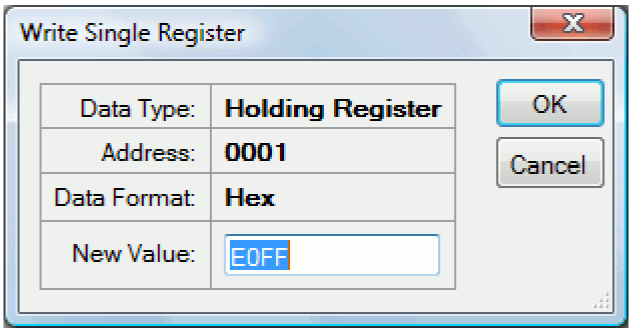


```
40001: <E0FFH>
40002: <05F5H>
40003: <3303H>
40004: <4404H>
```

Double Click in this area to change the value at address 40003

Depending on the Data Type of the address being modified a popup Window appears using which a new value can be written to this address.

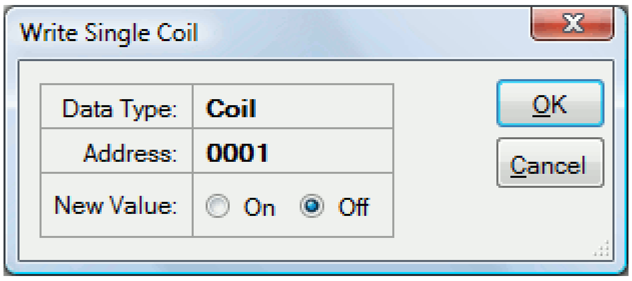
- The **Write Single Register** popup appears if the data type is **Holding Register**. In this popup, enter the value to be written to this address in the **New Value** text box. Press OK.



Write Single Register

Data Type:	Holding Register
Address:	0001
Data Format:	Hex
New Value:	E0FF

- The **Write Single Coil** popup appears if the data type is **Coil**. In this popup, set the new state for the coil to **On** or **Off** by clicking on the appropriate radio button. Press OK.



Write Single Coil

Data Type:	Coil
Address:	0001
New Value:	On [✓ Off]

3. Verify that the new value was indeed written to the device by examining the readback value in the Data Display Panel (ensure polling is enabled before doing this verification).

Note:

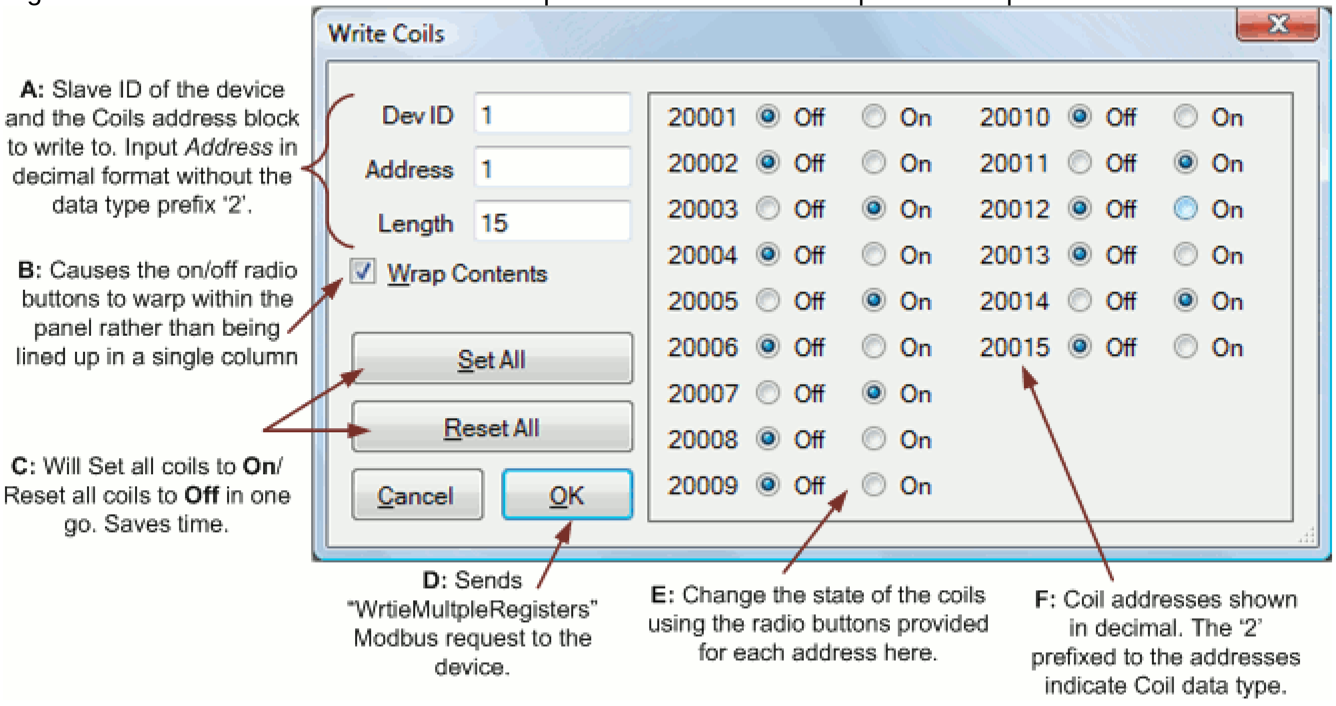
An attempt to write to a Discrete Input or an Input Register will cause an error message to be displayed.

5.2 How To: Write Multiple Coils

Modbus provides two functions to modify the state of Coils. Function Code 0x05 can be used to change the state of a single coil whereas Function Code 0x0F is for changing the state of multiple coils in one request. This section describes the procedure for changing the state of multiple coils using FC 0x0F.

The Write Coils form

Figure below shows a Write Coils form and provides a note on the important components of this form.



Write Coils

Dev ID: 1
Address: 1
Length: 15
Wrap Contents: ✓

Set All | Reset All | Cancel | OK

20001	● Off	○ On	20010	● Off	○ On
20002	● Off	○ On	20011	○ Off	● On
20003	○ Off	● On	20012	● Off	○ On
20004	● Off	○ On	20013	● Off	○ On
20005	○ Off	● On	20014	○ Off	● On
20006	● Off	○ On	20015	● Off	○ On
20007	○ Off	● On			
20008	● Off	○ On			
20009	● Off	○ On			

A: Slave ID of the device and the Coils address block to write to. Input *Address* in decimal format without the data type prefix '2'.

B: Causes the on/off radio buttons to warp within the panel rather than being lined up in a single column

C: Will Set all coils to **On**/ Reset all coils to **Off** in one go. Saves time.

D: Sends "WrtieMultpleRegisters" Modbus request to the device.

E: Change the state of the coils using the radio buttons provided for each address here.

F: Coil addresses shown in decimal. The '2' prefixed to the addresses indicate Coil data type.

Procedure

1. Open the **Write Coils** form by choosing menu item **Write->Coils** or by clicking the **Write Coils** toolbar button.
2. Set **Dev ID** to the Slave (Unit) ID of your device. Note that this parameter is ignored by most Modbus TCP devices.
3. Set **Address** to the Modbus address of the first coil in the block that will be modified
4. Set **Length** to the number of coils to be modified. Length cannot be larger than 1000. Setting a large value to length has performance tradeoffs as the numberof controls for modifying so many coils will take time to load into the panel. It is suggested that the length be limited to 200.
5. Modify the state of the coils as desired using the radio buttons. Use the **Set All** or **Reset All** buttons to quickly set (On) or reset (Off) all the coils.
6. Click **OK** to transmit the Write command to the device.

Tips:

You may use the **Wrap Contents** checkbox to control the flow of the radio buttons within the panel. If this is checked, each pair of radio buttons (along with their address labels) flow over to the next column on reaching the bottom of the panel. If not, all text boxes are placed in the same column and a scroll bar appears using which you can view text boxes that are beyond the bottom of the panel.

Tips:

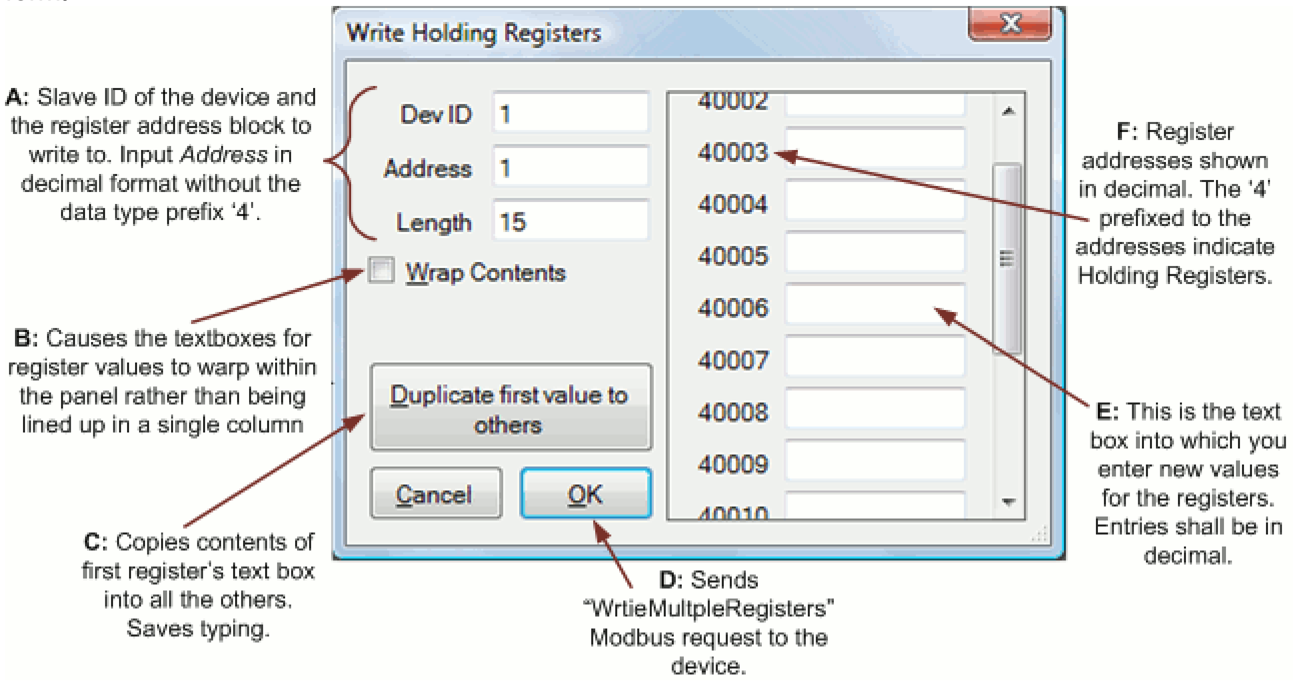
You may resize the **Write Coils** form to make more space so that all the addresses are visible without having to scroll. The radio buttons and their address labels reflow to fit into the resized form.

5.3 How To: Write Multiple Registers

Modbus provides two functions to modify the values of Holding Registers. Function Code 0x06 can be used to change the value of a single register whereas Function Code 0x10 is for changing the values of multiple Holding Registers in one request. This section describes the procedure for changing the state of multiple Holding Registers using FC 0x10.

The Write Holding Registers form

Figure below shows a Write Holding Registers form and provides a note on the important components of this form.



A: Slave ID of the device and the register address block to write to. Input *Address* in decimal format without the data type prefix '4'.

B: Causes the textboxes for register values to warp within the panel rather than being lined up in a single column

C: Copies contents of first register's text box into all the others. Saves typing.

D: Sends "WrtieMultpleRegisters" Modbus request to the device.

E: This is the text box into which you enter new values for the registers. Entries shall be in decimal.

F: Register addresses shown in decimal. The '4' prefixed to the addresses indicate Holding Registers.

1. Open the **Write Holding Registers** form by choosing menu item **Write->Holding Registers** or by clicking the **Write Holding Registers** toolbar button.
2. Set **Dev ID** to the Slave (Unit) ID of your device. Note that this parameter is ignored by most Modbus TCP devices.
3. Set **Address** to the Modbus address of the first Holding Register in the block that will be modified.
4. Set **Length** to the number of Holding Registers to be modified. The Length cannot be larger than 123.
5. Enter new values for each Holding Register in the text box provided against each address. You may use the **Duplicate first value to others** button if you intend to set all registers in this block to the same value.
6. Click **OK** to transmit the Write command to the device.

Tips:

You may use the **Wrap Contents** checkbox to control the flow of the text boxes within the panel. If this is checked, text boxes (along with their address labels) flow over to the next column on reaching the bottom of the panel. If not, all text boxes are placed in the same column and a scroll bar appears using which you can view text boxes that are beyond the bottom of the panel.

Tips:

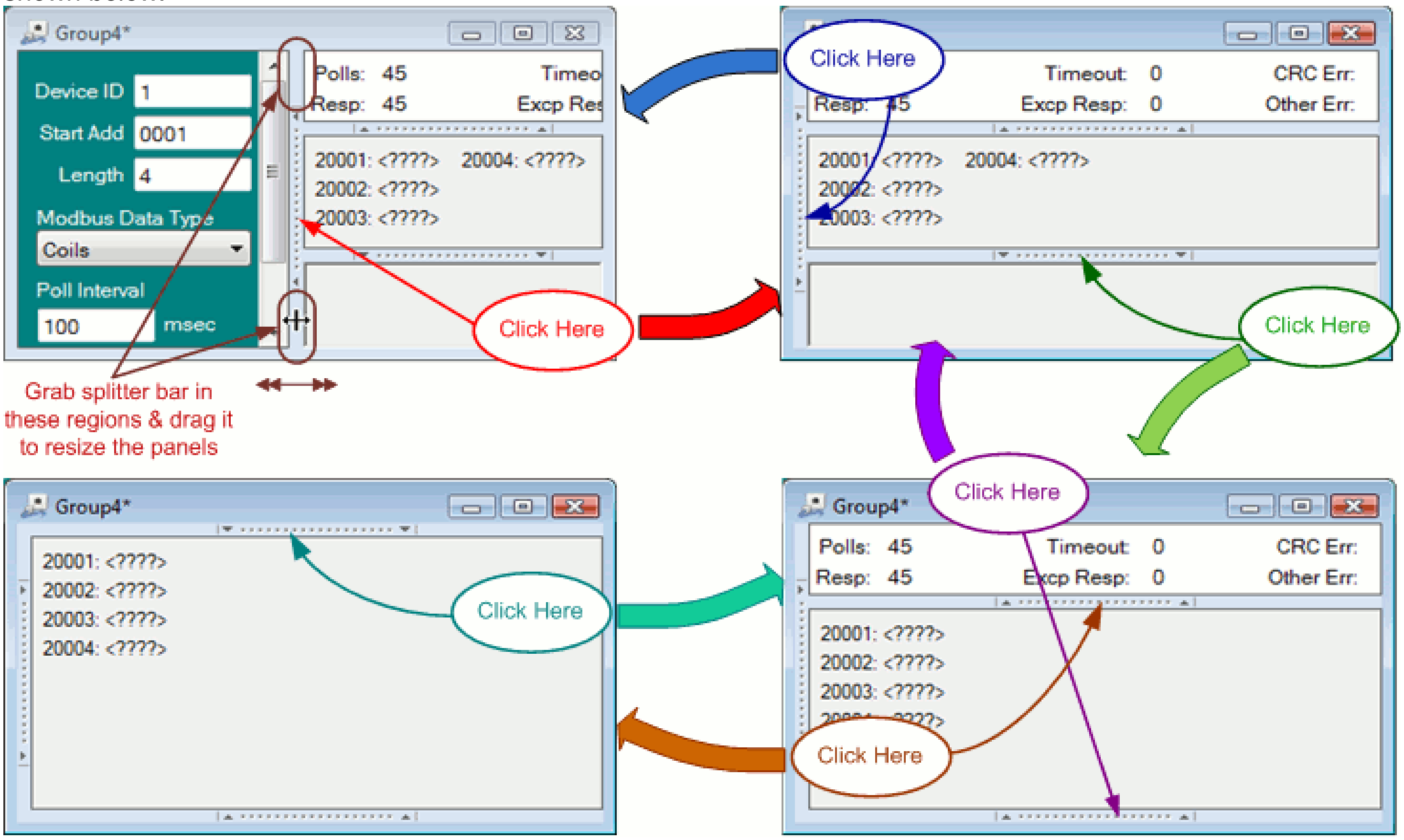
You may resize the **Write Holding Registers** form to make more space so that all the addresses are visible without having to scroll. The text boxes and their address labels reflow to fit into the resized form.

6. Customize Tester Interface

You can customize the data group forms to hide information elements that are not of interest to you or to change font and foreground/ background colors as per your personal preferences or to improve readability. These preferences are saved for each form on a per-user basis along with other settings of the Data Group Forms.

6.1 How To: Use Splitter Bars to resize or hide panels

After the configuration of a MDG form, you may want to hide the configuration panel or reduce its size so that more screen space is available for the Data Display Panel. In some other case, you may not be interested in Modbus traffic and so would want to hide the Traffic Display Panel. This can be done using the splitter bar as shown below.



6.2 How To: Change Colors

Procedure

1. Right-click the mouse anywhere within the Data Display Panel. The context menu for the panel will pop up.
2. Click on the **Back Colour** menu item. The Colour selection dialog box will appear.
3. Click on a colour rectangle of your choice to select a *Basic colors*. If your colour is not among the basic colours click on the **Define Custom Colors** button and select your colour from the provided colour spectrum.
4. Click OK.

Change foreground (i.e. font) colour of the Data Display Panel

Follow the procedure as in **Change background colour of the Data Display Panel** but in step (2) choose **Fore Colour** menu item instead of **Back Color**.

Change background and foreground colours of the Traffic Display Panel

Follow the procedures for changing background and foreground colours of the Data Display Panel but run the procedures on the Traffic Display Panel instead.

6.3 How To: Change Fonts

Procedure

1. Click the right mouse button anywhere within the Data Display Panel. The context menu for the panel will pop up.
2. Click on the **Font** menu item. The font selection dialog box will appear.
3. Choose the desired **Font**, **Font Style** and **Size**.
4. Click OK.

How to change the font style for the Traffic Display Panel?

Follow the procedure as in **Change font style for the Data Display Panel** but run the procedure on the Traffic Display Panel instead.

Tips:

Using a fixed-width font will ensure that the characters in the display are properly aligned from one row to another irrespective of the characters being displayed.

7. Event Logger

What is the Event Logger?

The Event Logger feature writes important activities and events occurring in the application to a log file. For e.g. the MPT successfully connecting to a device is an event and this information is logged into the file. Event logging can be enabled or disabled. The level of log messages can also be controlled. All event logs are time stamped allowing a user to analyze the sequence of events occurring against a timeline.

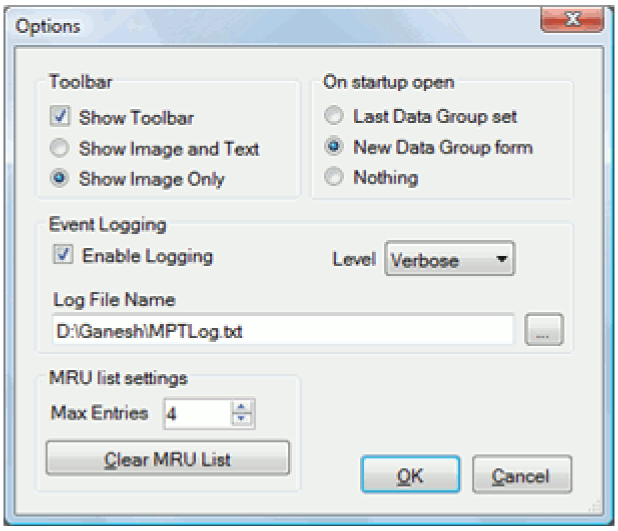
What kind of activities and events get logged?

There are four kinds of events/activities that get logged: Errors, Warnings, Informational Messages, Verbose Messages. The logger can be configured to log at one of these levels.

- Errors are events when MPT is unsuccessful in completing an activity or encounters and unexpected situation. An example is MPT is unable to establish a connection to the device.
- Warnings are non-critical errors that occur in the application. An example is the user attempting to start a device simulator when one is already running.
- Informational Messages provide logs of routine activities in the MPT and can be reviewed to check and ensure that all is well in the application. An example is the MPT successfully connecting to a device.
- Verbose Messages provide detailed logs of activities in the application with an intention of providing sufficient information of the sequence of events occurring in the MPT for the purpose of debugging errors and failures in the application. For e.g. a verbose log is made when the user saves a Modbus Data Group form.

How to enable Event Logging and set the Logging Level

1. Open the *Options* dialog box by choosing menu ***Tools->Set Options***


Options

Toolbar
- [✓] Show Toolbar
- ○ Show Image and Text
- ● Show Image Only

On startup open
- ○ Last Data Group set
- ● New Data Group form
- ○ Nothing

Event Logging
- [✓] Enable Logging
- Level: Verbose
- Log File Name: D:\Ganesh\MPTLog.txt

MRU list settings
- Max Entries: 4

2. Check the *Enable Logging* check box
3. Enter a file name for the log file including the full directory path. Alternatively, click the *Browse* button beside the file name text box to browse to a folder. Note that the MPT will not overwrite but will append log messages to this file.
4. Select the logging level from the *Level drop* down list box. Please note that setting the log level to:

Error logs only Error messages.

Warning logs Error and Warning messages.

Information logs Error, Warning and Informational messages.

Verbose logs all types of events including Verbose messages.

How to view the Event Log

The Event log file is a text file and so can be viewed with any text editor. You can choose the menu ***Tools->View Event Log*** to open the Event log file in notepad.

Description of an Event Log entry

An entry in the Event log file has the following format:
```
MPT <Log Level>: <Event ID>: <Date: DD-MM-YYYY> <Time: hh:mm:ss.ms> <Log message
text>
```
Where:

- Log Level: Can be Error, Warning, Information or Verbose.
- Event ID: Is a numeric identifier for the event. This field is currently unused and reserved for future use.
- Date: The date when the log was made in DD-MM-YYYY format (e.g. 12-01-2010).
- Time: Time with millisecond resolution at which the log was made in HH:mm:ss.ms format (e.g. 09:45:30.768). Hour is printed in 24 hour format.
- Log message text: A text message describing the log.

A typical entry in the log file looks as below:
MPT Information: 0 : 27-03-2010 12:21:11.633 Global settings changed

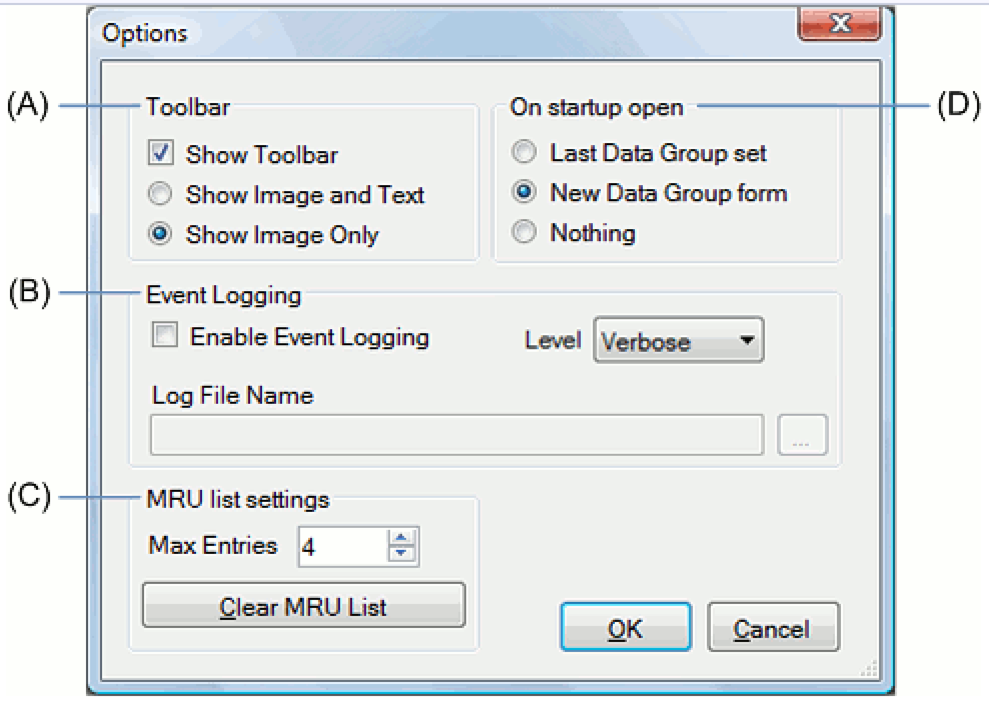
8. Setting Protocol Tester Options

The Options Form can be used to set up a few configuration parameters of the Modbus Protocol Tester. The form can be opened by choosing **Tools->Set Options** menu item or by clicking the "Set Options" toolbar

button .

Caution:

All option settings are stored by the Tester and automatically applied the next time it is lauched. It is not necessary to make the set



Toolbar
[✓] Show Toolbar
Show Image and Text [✓ Show Image Only]

On startup open
Last Data Group set [✓ New Data Group form] Nothing

Event Logging
[ ] Enable Event Logging
Level: Verbose
Log File Name:

MRU list settings
Max Entries: 4
Clear MRU List

A: Toolbar Options

This set of options controls how the toolbar is displayed.

- **Show Toolbar Checkbox:** Checking this item will cause the toolbar to be visible. If not the toolbar is hidden.
- **Show Image and Text:** Selecting this radio button will cause the toolbar's text to appear beside its image.
- **Show Image Only:** Selecting this radio button will cause the toolbar's image alone to be displayed and no text.

- B: Event Logging Options

- This set of options should be used to enable or disable event logging and also to set the name and path to the event log file. Click here for more on event logging.
- **Enable Event Logging:** If checked, important events are logged into the log file. Else, no events are recorded.
- **Log File Name:** You may type in the name of the event log file with full path here. Clicking the ... button pops up a file selection dialog box using which you can conveniently browse the file system and choose a path and a file name for the log file.

Caution:

The Tester requires write permissions to the file and folder path entered here. If this is not granted, the Tester throws up an error a

C: MRU List Settings

Most Recently Used list settings.

- **Max Entries:** The maximum number of file names that will be listed under the MRU menu. Beyond this, a new file name will replace an old file name on a First In First Out basis.
- **Clear MRU List:** Clicking this button will erase all entries in the MRU list.

D: Startup Behavior Settings

- These settings control which Data Group Form is opened by the Tester when it is launched.
- **Last Data Group set:** Selecting this radio button will cause the Tester to store the list of all Data Group forms that were open just before it was closed and reopen them automatically when it is launched next.
- **New Data Group form:** Selecting this radio button will cause the Tester to create a new Data Group form (unsaved) when it is launched.
- **Nothing:** Selecting this radio button will cause the Tester to display an empty main form (i.e. no Data Group forms are created or opened) when it is launched.

9. Modbus Test Automation

9.1 Introduction

Mb_Automation allows user to send multiple MODBUS commands at a time to the slave and logs the data into a text file. MB_Automation routines provide both read and write access to one or more modbus slave devices through the script generated using XML file. Test scripts consist of ASCII text data field separated by Node elements for each function codes. They may be constructed using any XML editor application. A test script entry consists of at least 15 data fields with sub child nodes.

9.2 Features of MB_Automation

- Mb_Automator supports around 11 function codes such are,

  i. Read Coils (0x01)

  ii. Read Discrete Input (0x02)

  iii. Read Holding Registers (0x03)

  iv. Read Input Registers (0x04)

  v. Write Single Coil (0x05)

  vi. Write Single Register (0x06)

  vii. Read Exception Status (0x07)

  viii. Write Multiple Coils (0x0F)

  ix. Write Multiple Registers (0x10)

  x. Report Slave ID (0x11)

  xi. Read/Write Multiple Registers (0x17)

- Validation of Read Coils with pre-defined reference data.
- Validation of Write Coils & Registers.
- User friendly XML script.
- Data logging of real time communication statistics like sent requests, responses received, errors etc.
-

9.3 Syntax for automation profile

Sample automation profile (Xml Script file) consists of one parent element called **"<MbProfile>"** and there are around minimum 15 child elements (child elements may increase or decrease based on user requirement) out of which 11 child elements are used for MODBUS commands and the rest 3 child elements are mandatory to run Mb_Automation. Following are the three mandatory syntax.

- **Repeat Count**

  Using <RepeatCount> element user can decide the number of times script to run.

  XML format:

```
<RepeatCount>
<NumOfTimes_Script_Run>2</NumOfTimes_Script_Run>
</RepeatCount>
```

- **Log File**

  User can define the source path to store the log file and can change the log filename.

  XML format:

```
<LOGFILE>
        <Filename>D:/logFile.txt</Filename>
</LOGFILE>
```

- **Modbus Mode**

  Set Modbus communication mode to "RTU" or "TCP" depending on the communication preference.

  XML format:

```
< ModbusMode>
        <Mode>RTU</Mode>
</ModbusMode>
```

- **mbTcpMode**

  User can set the communication parameters for TCP communication.

  XML format:

```
<mbTcpMode>
        <DeviceIP>127.0.0.1</DeviceIP>
        <tcpPortNo>502</tcpPortNo>
        <Timeout>1000</Timeout>
</mbTcpMode>
```

- **Serial Port**

  User can set the communication parameters for the initialization of serial communication.

  XML format:

```
<SERIALPORT>
                <Parity>0</Parity>
                <Stopbits>0</Stopbits>
                <Databits>1</Databits>
                <Baudrate>3</Baudrate>
                <Timeout>1000</Timeout>
```

```
                    <Portname>COM2</Portname>
            </SERIALPORT>
```

9.4 XML syntax for MODBUS Commands

IMPORTANT NOTE: The MODBUS commands missing below have been explained in section '**Validation of Coils and Registers'**.

- **Write Single Coil (0x05)**

  Syntax for Write Single Coil:

```
<MbTxn>
        <FC>0x05</FC>
        <DeviceID>1</DeviceID>
        <StartAddr>1</StartAddr>
        <Length>1</Length>
        <Data>
              <Value>1</Value>
        </Data>
</MbTxn>
```

- **Write Single Register (0x06)**

  Syntax for Write Single Register:

```
<MbTxn>
        <FC>0x06</FC>
        <DeviceId>1</DeviceId>
        <StartAddr>1</StartAddr>
        <Length>1</Length>
        <Data>
              <Value>0xAABB</Value>
        </Data>
</MbTxn>
```

- **Report Slave ID (0x11)**

  Syntax for Report Slave ID:

```
<MbTxn>
        <FC>0x11</FC>
        <DeviceID>1</DeviceID>
</MbTxn>
```

- **Read Exception Status (0x07)**

  Syntax for Read Exception Status:

```
<MbTxn>
        <FC>0x07</FC>
        <DeviceID>1</DeviceID>
</MbTxn>
```

- **Read/Write Multiple Registers (0x17)**

  Note: Data value can be entered horizontally separated by **comma (,)** inside a tag **"<Value></Value>"** also user can create another new tag **"<Value></Value>"** in the next line (if required) and can continuing entering the data value sequentially as shown below.

Syntax for Read/Write Multiple Registers:

```
<MbTxn>
            <FC>0x17</FC>
            <DeviceID>1</DeviceID>
            <StartAddrReadReg>1</StartAddrReadReg>
            <StartAddrWriteReg>1</StartAddrWriteReg>
            <QtyReadReg>5</QtyReadReg>
            <QtyWriteReg>5</QtyWriteReg>
            <Data>
                    <Value>0x1111, 0x2222, 0x3333</Value>
                    <Value> 0x4444, 0x5555</Value>
            </Data>
</MbTxn>
```

9.5 Validation of Coils and Registers

Important Note: Reference data/write registers or coils values can be entered horizontally separated by **comma (,)** inside a tag **"<Value> </Value>"** and also user can create another new tag **"<Value></Value>"** in the next line (if required) and can continuing entering the reference data value sequentially as shown below.

Ex1:

```
<MbTxn>
            <FC>0x01</FC>
            <DeviceID>1</DeviceID>
            <StartAddr>1</StartAddr>
            <Length>16</Length>
            <CompWrMuCoils>0</CompWrMuCoils>
            <Data>
                    <Value>0, 1, 1, 1, 1, 0, 1, 1</Value>  ⇕  Reference
                    <Value>1, 0, 1, 1, 0, 1, 1, 1</Value>      Data
            </Data>
</MbTxn>
```

Ex2:

```
<MbTxn>
            <FC>0x10</FC>
            <DeviceID>1</DeviceID>
            <StartAddr>1</StartAddr>
            <Length>6</Length>
            <Data>
                    <Value>0x0FAA, 0xAADD</Value>  ⇕  Write
                    <Value>0xCCAA, 0xFFFF</Value>      Register Values
            </Data>
</MbTxn>
```

- ### Read Coils (0x01)

  MB_Automator provides a feature to cross verify the successful transaction of read coils over MODBUS.

  Read Coils can be validated by providing a reference data in the XML file. The following shows the syntax for validating the read coils.

  During validation of read coils with **reference data**, make sure that "**<CmpWrMulCoils>**" tag is assigned to zero (0) as shown below.

  XML format for (0x01):

```
<MbTxn>
            <FC>0x01</FC>
            <DeviceID>1</DeviceID>
            <StartAddr>1</StartAddr>
            <Length>16</Length>
            <CmpWrMulCoils>0</CmpWrMulCoils>
```

```
          <Data>
                    <Value>0, 1, 1, 1, 1, 0, 1, 1</Value>   Reference
                    <Value>1, 0, 1, 1, 0, 1, 1, 1</Value>     Data
          </Data>
</MbTxn>
```

Note: If user wants to skip the validation/comparison of read coils. Just delete the entire sub child node (referenced data) with the tag name called **"<Data> </Data>".**

- **Write Multiple Coils (0x0F)**

  MB_Automator provides a feature to cross verify the successful transaction of write multiple coils over MODBUS.

  XML format for (0x0F):

```
< MbTxn>
                    <FC>0x0F</FC>
                    <DeviceID>1</DeviceID>
                    <StartAddr>1</StartAddr>
                    <Length>16</Length>
          <Data>
                    <Value>0, 1, 1, 1, 1, 0, 1, 1</Value>   Write Coils
                    <Value>1, 0, 1, 1, 0, 1, 1, 1</Value>     Values
          </Data>
</ MbTxn>
```

  In order to validate the write multiple coils, just call **Read Coils (0x01)** function code without any reference data for read coils in the XML file immediately after the write multiple coils.

  During validation of write multiple coils, make sure that there is no reference data is associated with read coils function code and make sure that "**<CmpWrMulCoils>**" tag is assigned to one (1) as shown below.

  Syntax for Read Coils (0x01) without any reference data for validation of Write Multiple Coils:

```
<MbTxn>
                    <FC>0x01</FC>
                    <DeviceID>1</DeviceID>
                    <StartAddr>1</StartAddr>
                    <Length>16</Length>
                    <CmpWrMulCoils>1</CmpWrMulCoils>
</MbTxn>
```

  Note1: During validation/comparison of write multiple coils the **"StartAddr"** and **"Length"** of **"Read Coils (0x01)"** should be same as the **"Write Multiple Coils".** If different value is given then validation result fails.
  Note2: If user wants to skip the validation/comparison of write multiple coils. Then assign zero (0) to "**<CmpWrMulCoils>"** tag in read coils function code.

- **Read Discrete Input (0x02)**

  MB_Automator provides a feature to cross verify the successful transaction of read discrete input over MODBUS.

  Read Discrete Input can be validated by providing a reference data in the XML file. The following shows the syntax for validating the read discrete input.

  XML format for (0x02):

```
<MbTxn>
                    <FC>0x02</FC>
                    <DeviceID>1</DeviceID>
                    <StartAddr>1</StartAddr>
                    <Length>16</Length>
```

```
        <Data>
                <Value>0, 1, 1, 1, 1, 0, 1, 1</Value>     Reference
                <Value>1, 0, 1, 1, 0, 1, 1, 1</Value>       Data
        </Data>
</MbTxn>
```

Note: If user wants to skip the validation/comparison of read discrete input. Just delete the entire sub child node (referenced data) with the tag name called **"<Data> </Data>".**

- **Read Holding Registers (0x03)**

  MB_Automator provides a feature to cross verify the successful transaction of Read Holding Registers over MODBUS.

  Read Holding Registers can be validated by providing a reference data in the XML file. The following shows the syntax for validating the Read Holding registers.

  During validation of read holding registers with **reference data**, make sure that "**<CmpWrMulReg>**" tag is assigned to zero (0) as shown below.

  XML format for (0x03):

```
<MbTxn>
                <FC>0x03</FC>
                <DeviceID>1</DeviceID>
                <StartAddr>1</StartAddr>
                <Length>10</Length>
                <CmpWrMulReg>0</CmpWrMulReg>
        <Data>
                <Value>0x0FAA, 0xAADD, 0xBBCC</Value>        Reference
                <Value>0xCCAA, 0xDDDD, 0xFFFF</Value>          Data
        </Data>
</MbTxn>
```

  Note: If user wants to skip the validation/comparison of Read Holding register. Just delete the entire sub child node (referenced data) with the tag name called **"<Data> </Data>".**

- **Write Multiple Registers (0x10)**

  MB_Automator provides a feature to cross verify the successful transaction of write multiple registers over MODBUS.

  XML format for (0x10):

```
<MbTxn>
                <FC>0x10</FC>
                <DeviceID>1</DeviceID>
                <StartAddr>1</StartAddr>
                <Length>3</Length>
        <Data>
                <Value>0x0FAA, 0xAADD, 0xBBCC</Value>       Write Register
        </Data>                                                   Values
</MbTxn>
```

  In order to validate the write multiple registers, there is no necessity to provide reference data. To validate, just call **Read Holding Registers (0x03)** function code immediately after the write multiple registers without any reference data for Read Holding Registers in the XML file.

  During validation of write multiple registers, make sure that there is no reference data is associated with read holding register function code and make sure that "**<CmpWrMulReg>**" tag is assigned to one (1) as shown below.

  Syntax for Read Holding Register (0x03) without any reference data for validation of Write Multiple Registers:

```
<MbTxn>
                <FC>0x03</FC>
                <DeviceID>1</DeviceID>
                <StartAddr>1</StartAddr>
                <Length>3</Length>
                <CmpWrMulReg>1</CmpWrMulReg>
</MbTxn>
```

Note1: During validation/comparison the **"StartAddr"** and **"Length"** of "Read holding registers (0x03)" should be same as the "write multiple registers (0x10)". If different value is given then validation result fails.

Note2: If user wants to skip the validation/comparison of write multiple registers. Then assign zero (0) to "**<CmpWrMulReg>"** tag in read holding registers function code.

- **Read Input Registers (0x04)**

  MB_Automator provides a feature to cross verify the successful transaction of Read Input registers over MODBUS.

  Read Input Registers can be validated by providing a reference data in the XML file. The following shows the syntax for validating the Read Input registers.

  XML format for (0x04):

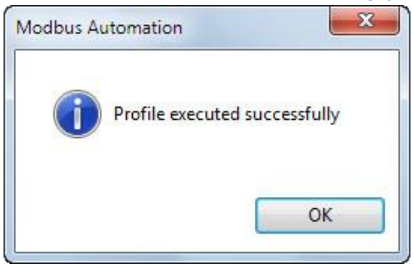
```
< MbTxn>
                <FC>0x04</FC>
                <DeviceID>1</DeviceID>
                <StartAddr>1</StartAddr>
                <Length>5</Length>
        <Data>
                <Value>0x0FAA, 0xAADD, 0xBBCC</Value     ⇕     Reference
                <Value>0xCCAA, 0xFFFF</Value>                  Data
        </Data>
</ MbTxn>
```

  Note: If user wants to skip the validation/comparison of Read Input registers. Just delete the entire sub child node (referenced data) with the tag name called **"<Data> </Data>".**

9.6 How to execute Modbus Test Automation?

1. Click on Modbus Test Automation toolbar button.
2. Load the sample automation profile supplied with the installer (can be found in the GUI installation folder. Copy into some local folder)

   Note: Ensure for proper communication settings in the sample automation profile. One of the most common causes for termination of automation test could be because of improper communication settings in the automation profile.
3. Click on **'Run Automation Profile'** button.
4. On successful automation test. Following pop up window will be displayed.

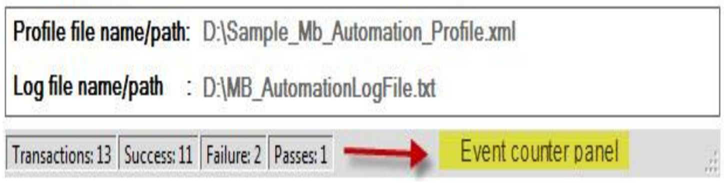


Profile executed successfully

In case of automation test failed a pop up error message will be displayed.

Additional information on Modbus Test Automation

- On successful automation test. Sample automation profile name/path & log file name/path information are available. As one shown below.



Profile file name/path: D:\Sample_Mb_Automation_Profile.xml

Log file name/path : D:\MB_AutomationLogFile.txt

Transactions: 13 | Success: 11 | Failure: 2 | Passes: 1

- **Event Counter Panel**

This panel displays a set of event counters to indicate status of communication of automation test with the Modbus device. A brief description of these counters follows:

a) **Transaction:** The total number of Modbus commands executed.
b) **Success:** The number of Modbus commands executed successfully.
c) **Failure:** The number command failed to execute (could be because of Read/write timeout, unsupported function code and so on).
d) **Passes:** Number of times automation profile executed.

10. Troubleshooting Connection Problems

Symptom	**Possible Causes**	**Possible Remedies**
Unable to connect to device	• Incorrect IP address • Device not powered ON • Ethernet cable to the device is loose or disconnected	• Verify IP address setting in Communication Setup Window matches the device IP • Ping the device to see if the physical connection is OK • Try connecting to he device directly bypassing any router or switch (use a crossed Ethernet cable)
MDG form displays "?????" in place of data	Polling has not been started in the MDG form	Enable polling by checking the "Start Polling" checkbox in the MDG forms