GUIDE TO VNA AUTOMATION IN MATLAB USING THE TCP INTERFACE

08/03/2018

Background

Usually, Copper Mountain Technologies instruments are locally automated on the same Windows-based PC to which the instrument is connected and the VNA control software is installed. Such functionality is readily achieved on a Windows-based PC by using the Component Object Model (COM), or ActiveX interface for automation. All that is required for automation of the instrument is the applicable VNA control software and its respective Programming Manual. The last page of the VNA control software installation wizard presents the option to register a COM interface server. If this box is checked and a confirmation dialogue pops up to confirm the registration, users are ready to begin programming.

However, some users require remote automation, instead, where they can control an instrument from a separate computer, and some users are running the Linux-compatible versions of the VNA control software and cannot use the COM interface to automate their measurements. To support this need, in addition to COM interface support, Copper Mountain Technologies instruments support the Transmission Control Protocol (TCP) interface. It can be used instead of the COM interface for local automation, or instead of the Distributed Component Object Model (DCOM) interface for remote automation.

There are numerous advantages to using TCP. First, the COM/DCOM interface is a Microsoft product, and is only available on Windows PCs. For remote automation, when compared with DCOM, TCP is considered much less difficult to use, making the remote automation easier to setup. TCP is the standard for internet communication among hosts running applications on a network, and therefore requires no special permissions or firewall settings to implement, unlike DCOM. Users can activate the TCP socket by simply navigating to the corresponding softkey menu.

Using TCP, it is possible to easily communicate with any Copper Mountain Technologies instrument currently active on a local area network. When a user wishes to control the “server” computer (the one that runs the VNA control software), they can send messages from a “client” computer (which doesn’t even require the VNA control software). Over TCP, these commands are sent as unsigned eight-bit integers which the VNA control software interprets as Unicode Standard Commands for Programmable Instruments (SCPI) strings, which are available in a series of Programming Manuals available from http://coppermountaintech.com/ . This application note illustrates the implementation of a TCP connection to one of our VNAs utilizing MATLAB, but it is possible to implement TCP connectivity in multiple programming languages.

Initializing and Connecting to a TCP Socket

To get started, first determine the IP address of the “server” computer.

- To locally automate an instrument using the same computer that the VNA control software is running on, the IP address to use is “127.0.0.1” or “localhost”.
- To remotely automate an instrument by connecting to a different computer, find the IP address on

the “server” computer by opening the command line and typing “ipconfig”. The computer's IP address will be listed as “IPv4 Address”.

Then, open the VNA control software and click the following softkeys:

System > Misc Setup > Network Setup > Network Setup

Set the TCP Socket option and ensure that the Port is 5025.

Testing Connection to a TCP Socket

If the TCP socket was activated, running this sample code on the “server” computer should return no errors:

```
try
      vna = tpclient('127.0.0.1', 5025);
catch ME
      disp('Error establishing TCP connection.');
      disp('Check that the TCP server is on');
      return
end
```

Standard Command Format

Once the connection between the “client” and “server” computers has been established, it should be possible to send the VNA “server” machine any command from the VNA’s respective Programming Manual. The following code demonstrates the format required to interface commands from the “client” computer to the “server”:

```
if(instrument(1) ~= 'R')
      write(vna, [uint8(['SOURce1:POWer:LEVel:IMMediate:AMPLitude ',
      power_level_dbm]), nl]);
      pause(0.1);
end

%Configure the measurement type
write(vna, [uint8(['CALCulate1:PARameter:DEFine ', parameter]), nl]);
pause(0.1);
write(vna, [uint8(['CALCulate1:SELected:FORMat ', format]), nl]);
pause(0.1);
write(vna, [uint8('TRIGger:SEQeunce:SOURce BUS'), nl]);
pause(0.1);

tic
for iter = 1:num_iter
```

```
16 -        write(vna, [uint8('TRIGger:SEQuence:SINGle'), nl]);
17 -        pause(0.1);
18 -        pause(0.1);
```

When sending commands over the TCP connection, first write the necessary code to send a string of integers in Unicode format to the machine. In the code above this appears as:

```
write(vna, [uint8('command'), nl]);
```

The `uint8()` command is used to convert the string array into unsigned 8-bit integer format because the `write()` command only accepts integers. `uint8()`, therefore, converts a string to the corresponding Unicode numbers. The variable `nl` is declared to be integer 10 to implement new lines more quickly; all SCPI commands are terminated with a new line.

Notice that it would be more elegant to write a function that does these operations, rather than having to type out `[uint8('command'), nl]` every time a command is to be sent. Likewise, there is a space after the end of every "write" command string. In code lines 7 and 9, notice these spaces after `DEFine` and `FORMat`, included within the string. This is to ensure proper spacing in SCPI format such that the variable that appends the command array (`parameter` and `format`, respectively), are in the proper spot to be written to the VNA interface. Recognize, also, that unlike COM interfacing, the channel selected for the `CALCulate` command in lines 7 and 9 is directly appended after the call.

An example of a function that executes the necessary steps to send a command is shown below:

```
 e
 2 -  %SEND_STR This function is used to send commands over a TCP connection.
 3 -  %     VNA is a tcpclient object that is occupied by the VNA. COMMAND is
a
 4 -  %     string for a SCPI command (i.e. 'SENSe:FREQuency:STARt 100MHz').
 5 -
 6 -  message = [uint8(command), 10];
 7 -  write(vna, message)
 8 -  pause(0.1)
```

This example further shows how TCP socket automation can be used with less typing.

Conclusion

We hope you have found this guide to be helpful. If there are any other settings you are wondering about, please contact us at support@coppermountaintech.com and we will gladly help get you up and running.