Measurement Automation in MATLAB

1 Introduction

This document will guide you through the process of automating your measurement setup with MATLAB. In the first section we will discuss how to setup connections to the measurement equipment so we can start sending commands and receive data from the measurement equipment. The second section discusses the general structure of the commands and how to use them. The third section will discuss how we can generate a class that simplifies sending and receiving data from the measurement equipment. Finally we will combine everything discussed into a measurement setup that measures the bode plot of an opamp.

2 Making a connection

In this section two different connection methods are discussed, the Prologix Ethernet to GPIB converter and VISA. They share many similarities, however there are some differences in the initialization for both connections. First we discuss how to connect to the Prologix controller, after which it is discussed how to set up a visa connection.

2.1 Prologix GPIB

The Prologix Ethernet to GPIB converter is a custom GPIB-LAN controller which allows control of GPIB measurement equipment over ethernet. The controller has some extra configuration options which could make the communication easier. First it is discussed how to verify if a connection can be made with the Prologix GPIB controller. Afterwards it is discussed how to configure different settings of the Prologix controller. The full documentation for the Prologix Ethernet controller can be found at `http://prologix.biz/downloads/PrologixGpibEthernetManual.pdf`

2.1.1 Finding the correct IP address

In the Westzaal the Prologix Ethernet to GPIB converter is connected to a separate Ethernet port. The first thing to verify is if the IP address range of the controller and the network port match. If this is not the case the controller cannot be accessed from MATLAB.

First we retrieve the IP address of the PC at the second Ethernet port. Open the command promt (CMD) and run ipconfig. The results should be similar as in figure 1.

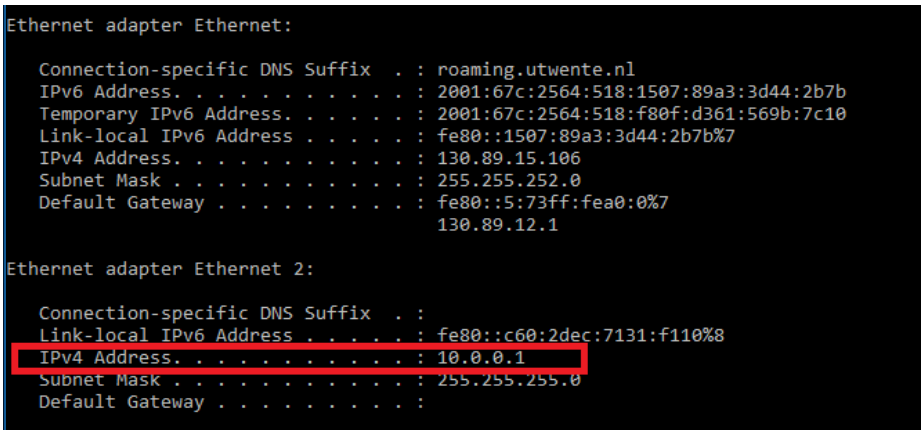

Figure 1: ipconfig results. The first ethernet adapter has a 130.89.XX.XX range, which is the standard UT range. The second ethernet adapter has a 10.0.0.XX range, which in the Westzaal is connected to the measurement equipment.

To find the IP address of the Prologix controller a tool called Netfinder is made available. Netfinder can be found at http://prologix.biz/downloads/netfinder.exe. It searches for all available Prologix Ethernet controllers and lists them. It allows you to see the settings and also change the IP address of the controller. The results are shown in figure 2.

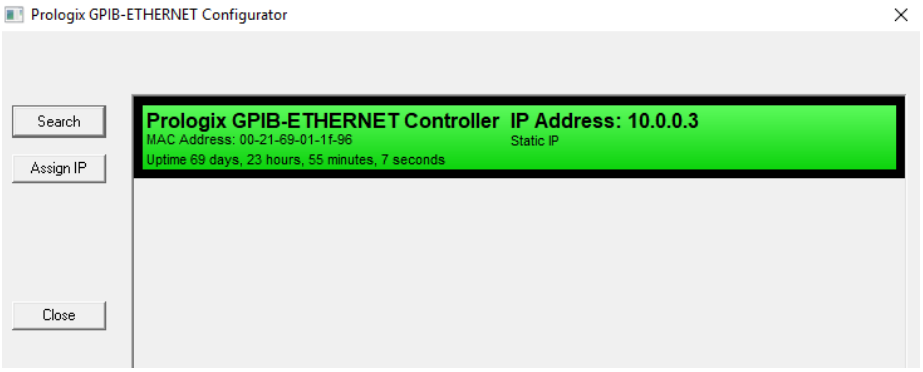

Figure 2: The results of the Prologix netfinder tool. A single GPIB-Ethernet controller was found at ip address 10.0.0.3

As long as the first parts match of the PC ip-address and the Prologix ip-address, in this case 10.0.0.XX, everything is set to connect to the Prologix controller. So, our PC has an ip address of 10.0.0.1 and the GPIB-Ethernet controller has an address of 10.0.0.3, therefor the PC can access the GPIB-Ethernet controller.

2.1.2 Common settings

Since the Prologix controller is not a standard GPIB controller it requires some additional code to function. We will discuss the 'mode', 'auto', 'eoi', 'read' and

'addr' here.

The mode of the Prologix controller sets the controller in either controller mode (1) or device mode (0). Since we want to control the measurement equipment we use the Prologix controller in controller mode. To set the Prologix in controller mode send **'++mode 1'**.

The auto settings is used to automatically read data after a write action is performed. This could be useful when reading lots of data. However, since we write almost as much as we read we disable automatic read. Send **'++auto 0'** to disable automatic read after write.

The eoi command enables or disables the automatic insertion of the EOI signal after the last character has been transferred over GPIB. Some measurement equipment requires this in order to properly detect the end of the message. In this case we will enable the eoi by sending **'++eoi 1'**.

A read from the measurement equipment last until:

- EOI is detected
- A specific character is read
- timeout

In our case most measurement equipment uses the EOI signal to notify the controller that no more data will be send. To configure the controller to read until EOI is set send **'++read eoi'** to the Prologix controller.

The last thing is to set the addr. In this case addr is the GPIB address of the measurement equipment. So to connect the Prologix controller to a machine on GPIB port 22 send **'++addr 22'**. There are many more configuration settings available, so see the manual of the Prologix controller if you are interested.

2.2 VISA

There are many different connection standards in use for measurement equipment, for example serial, usb, ethernet and GPIB. Luckily the VISA I/O API is available to help us. VISA stands for Virtual Instrument Software Architecture and is commonly used in industry for communication between computers and measurement equipment. The VISA API takes care of the communication between the computer and the measurement equipment, so we can focus on the actual commands that have to be send instead of also having to implement the connection specific communication protocol.

2.2.1 VISA connections

Before VISA can be used driver software have to be installed. In the case of the PCs in the Westzaal the drivers from National Instruments are installed. The drivers come with a tool called NI-Max which can be used to manage all your VISA compatible devices.

From figure 3 we get the visa resource name which is:
`TCPIP0::10.0.0.2::inst0::INSTR`. We need this visa resource name later to connect MATLAB to the oscilloscope. In this case the only visa compatible devices are the oscilloscope connected to the ethernet connection and a serial port COM1 with visa address `ASRL1::INSTR`. If we supply the visa API with an

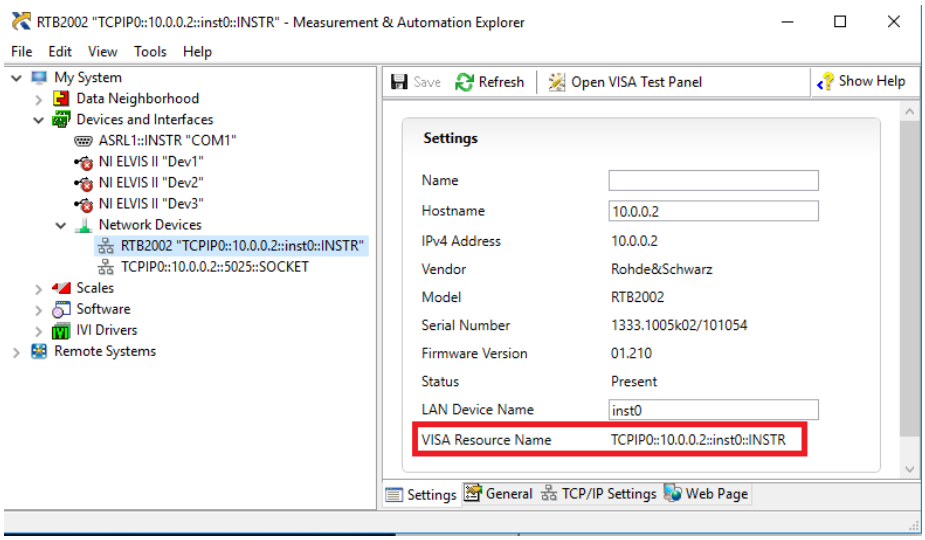

Figure 3: NI-Max program. It shows the Settings for the Rohde& Schwarz RTB2002 scope.

address it automatically determines what kind of connection is used, removing the need for separate code for each type of connection.

If the scope is not visible in NI-max check the IP address of the scope. You can open the ethernet settings of the scope by pressing the icon on the top right (ethernet icon). Then select the cog next to the ip address that becomes visible. This should open the Ethernet settings. Another possibility is to press Menu-¿Setup-¿Interface-¿Ethernet. Then press the parameter button. The screen shown in figure 4 should become visible. Make sure that the IP address settings match the settings of the PC ethernet adapter.

2.3 Setting up a connection in MATLAB

For the Prologix GPIB and VISA interfaces communication is performed in a similar way. The difference is in the interface used. VISA creates an abstract interface, so only an address is required. For the Prologix GPIB controller a separate TCP-IP connection has to be established.

To start the communication a object that contains information about the connection has to be made. For the Prologix GPIB controller TCP-IP connection object is needed. To create a TCP-IP connection in MATLAB use the TCP-IP function.

A useful command to reset all instrument connections is **instrreset**. If you have an open connection but have removed all objects (clear) use this to reset all open connections.

1 Create a connection object to the Prologix GPIB controller at ip-address 10.0.0.X to port 1234 by using *tcpip('10.0.0.X',1234)*. Save the result to variable prologixAddress.

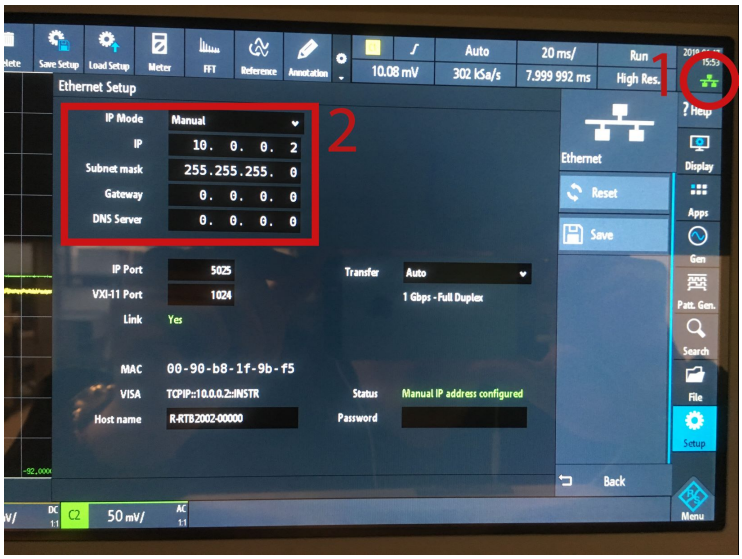

Figure 4: IP settings of the scope. First press the icon on the top right (ethernet icon). Then select the cog next to the ip address that becomes visible. This should open the Ethernet settings.

Now we have a object containing the tcp-ip connection information. The next step is to open this connection. Only when the connection is open data can be read and send.

2 Open the connection by using fopen(prologixAddress)

After opening the connection it is time to use it to write data to the Prologix controller and also read some data. In almost all cases before data can be read a command requesting the data has to be send. We are going to request and read the version of the software of the Prologix controller. First write data to the Prologix controller.

3a) Use fprintf(prologixAddress,'++ver') to request the firmware version of the prologix controller.
3b) Use version = fscanf(prologixAddress) to read the requested data from the prologix controller.

If you get data back this means that you successfully established a connection to the Prologix controller. The next step is to connect to the function generator and read its ID. Figure out what the GPIB address is of the function generator. This can be found in the MENU, normally in a section called I/O. This differs per scope.

4 Request the ID of the function generator by querying '*IDN?'.
a) Set the mode of the Prologix controller to 1 (++mode 1)

b) Set auto to 0 (++auto 0)
c) Set EOI to 1 (++eoi 1)
d) Set read to stop at eoi (++read eoi)
c) Set the GPIB address to the correct address (++addr X)

The last step is to close the connection again. Directly closing the connection after it is established prevents you from losing track of which connections are already open.

4 Close the connection with fclose(prologixAddress)

By now you should have successfully connected to the Prologix controller, read its firmware version and closed the connection. Now try to also read the ID of the oscilloscope through visa. The difference is that instead of creating the connection object with the tcpip function use the visa function. An example of the syntax could be scopeAddress = visa('ni',`'TCPIP0::10.0.0.2::inst0::INSTR'`). In the Westzaal the National Instruments visa drivers are installed, therefor the first argument should be 'ni'. To request the ID of the scope send `'*IDN?'` and then read the data.

5 Request the ID of the oscilloscope by using VISA.

3 SCPI Commands

There are many types of different measurement equipment and also many different manufacturers of measurement equipment. To prevent every manufacturer from designing its own standards most measurement equipment uses Standard Commands for Programmable Instruments(SCPI). Most same types of measurement equipment, e.g. signal generators, share a lot of the same commands, therefor making it easier to replace the measurement equipment with a different version.

3.1 Command structure

There are two types of SCPI commands, a set command and a query command. The set and query command can easily be distinguishes of each other since a query command always ends wit a question mark (?). This results in :FREQ X being a set command to set the frequency to X and :FREQ? a request for the current frequency value.

SCPI commands are structured as tree as shown in figure 5. A different level in the hierarchy is denoted by a colon (:).

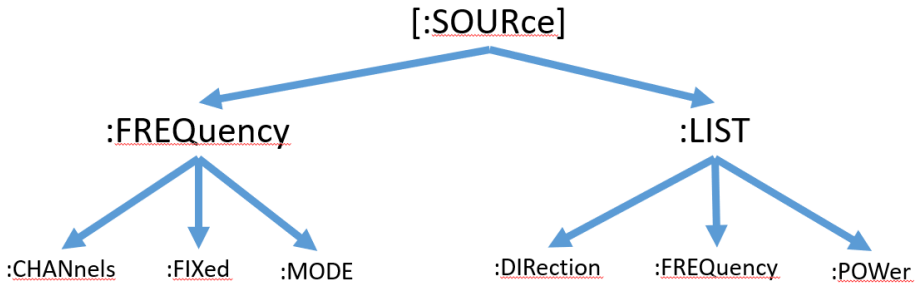

Figure 5: SCPI tree structure. In this way all commands are sorted,e.g. everything that has to do with measure is sorted under :MEAS

The command syntax as shown in figure 5 shows some letters as capital letters, while others are in lower case. It is only required to send the capital letters so `:FREQuency:CHANnels` is the exact same command as `:FREQ:CHAN`. Everything between square brackets can also be omitted.

4 Creating Objects in MATLAB

As discussed previously sending commands always requires you to open a connection, send the data and close the connection. To make these steps easier, we are going to write a class for the GPIB-Ethernet controller. This class will contain 3 functions: a constructor, a sendData() and a queryData(). Matlab already has templates available for generating classes. Press New → class to generate the template. The basic template is shown in figure 6.

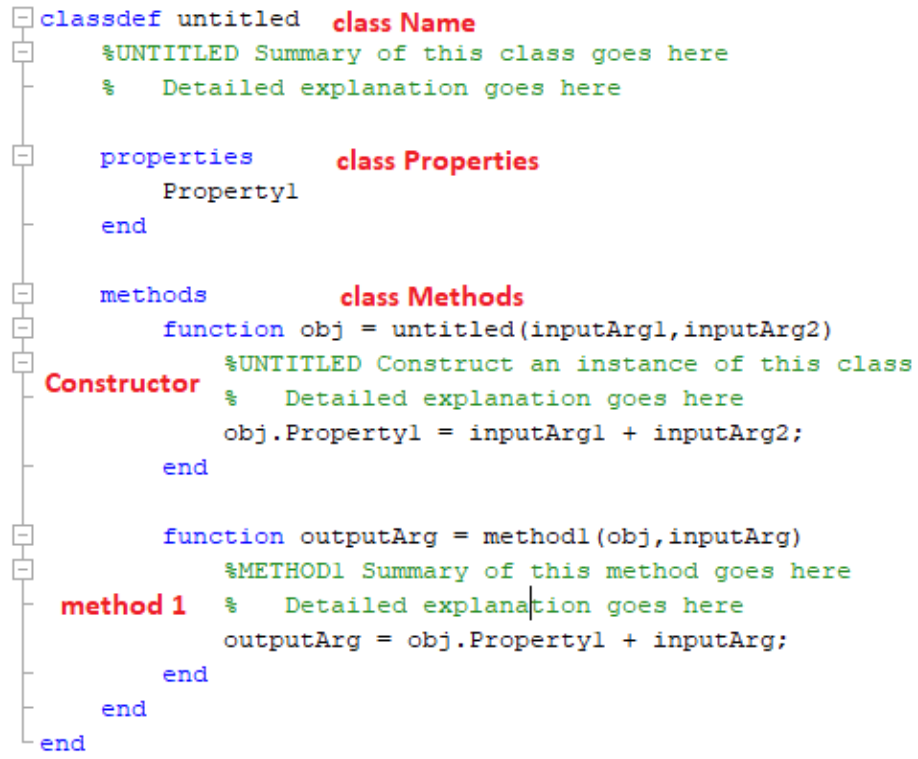

Figure 6: Standard class template from matlab. It can be generated by pressing New → class.

In the properties section the class variables should be stored. The methods section contains the methods of the class. We are going to edit this template and create a class that handles data transfers for us. First we should pick a name for our class; in this case we chose dataConnection.

1. Change the untitled in both the classdef and the first method (the constructor) to dataConnection.

The dataConnection now has a single property, Property1. We need a properties to store the location where the data should be written to. It is best practice to give the property a clear and unambiguous name. Since we want to store the address for the data connection to read and write from we change Property1 to address.

2. Change Property1 to address.

The constructor is called directly when a new class in instantiated. We can require some input arguments from which we use during the initialization of the class. In this case we require an address where the data has to be written to.

3. Change the input arguments for the constructor function to include the address.

Now that we have some input arguments it is time to store them as well. We can now set the address property with the address input argument from the constructor. If you want to access properties of the object itself use **obj.property** . See the example in the class Template how to do this.

4. Store the input argument address value into the class property address. The variable can be accessed by obj.address .

Now that the constructor is done we can start writing the methods of the class. In this case we want two methods: sendData(obj,data) and queryData(obj,command). The template class already contains a template for a function that returns data, e.g. the queryData(obj,command) method. Every method inside the class has as first input argument the obj, which refers to the object itself. This allows the methods of the class to access its own properties.

5a) Give the method the correct name and change it arguments list
5b) Implement the queryData function
5c) Create the sendData() method and implement it

Now it is time to test if the class actually works. Create a new MATLAB script and initialize a new dataConnection class. For example a new connection to the scope could look like : scopeCon = dataConnection('visaScope'). To access the methods in the class use scopeCon.queryData().

6 Test the connection to the scope by querying *IDN?

5 Final Assignment: Measurements

Now we are all set to finally perform some measurements. To put everything into practice we are going to measure the bode plot of the opamp circuit shown in figure 7. Use your previously written code in this assignment.

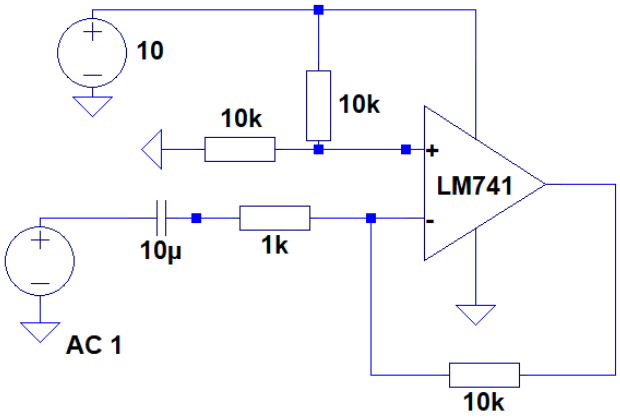

Figure 7: Schematic of the opamp circuit to be characterized. Expected is a gain of -10.

1) Build the circuit as shown in figure 7

Now that we have the circuit it is time to build the rest of the measurement setup as shown in figure 8. Connect a bnc splitter to the signal generator and connect one of the outputs to the OPAMP and the other one to CH1 of the oscilloscope. Connect the output of the opamp to CH2 of the scope.

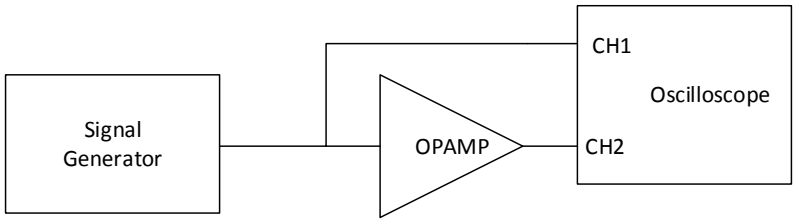

Figure 8: Schematic overview of the full measurement setup

2) Build the measurement setup as shown in figure 8

To measure the bode plot two things are required, the gain of the circuit and the phase shift. We are interested in the phase shift between CH1 and CH2, so we trigger the scope on CH1.

3) Send the SCPIs commands to the scope to trigger on the rising edge of CH1. Hint: Search the manual of the scope for the SCPI command TRIGGER:

The next step is to instruct the scope to take some measurements. In this case we are interested in the phase difference between CH1 and CH2 and the ratio between the amplitudes of CH1 and CH2.

4) Send the SCPIs commands so the scope performs the measurements. Hint: Search the manual of the scope for the SCPI command MEASUREMENT:. You have to set the MAIN and the SOURCE

Now that the scope measures the phase and voltage amplitudes we should instruct it to send the data to us.

4) Query the measurement results. Hint: Search the manual of the scope for the SCPI command MEASUREMENT$< n >$:Result.

The next step is to control the frequency of the signal generator. This will allow us to change the frequency during a measurement.

5) Set the frequency of the generator by using MATLAB. Hint: use :FREQ

At this point we have all the separate parts to do a fully automated measurement: circuit,frequency control, phase of output signal and the amplitudes of both the input and output signals. It is now time to write a loop that performs the measurement and plots the bode plot.

6) Measure the bode plot of the opamp circuit and plot the results

If you sweep the input frequency it might be required to change the time scale of the scope as well. To set the time scale use the following command `TIMEBASE:SCALE value`. To guarantee that you always have the same number of periods the *value* could be set to 1/f.