VAASAN AMMATTIKORKEAKOULU
VASA YRKESHÖGSKOLA
UNIVERSITY OF APPLIED SCIENCES

Jin Cao

CONTROLLING TELECOM INSTRUMENTS USING MATLAB

Technology and Communication

2013

PREFACE

Firstly I would like to give my gratitude to my mentor Dr. Chao Gao, for his patient guidance and great encouragement during conducting the whole thesis. It is my honour for being given this valuable opportunity to work on this project.

Then many thanks to this school, VAMK, University of Applied Science, the place where I have spent precious four years. Additionally, I want to express my deep appreciation for all the faculties. This thesis is done in Techonobotina Research Centre. Vaasa, Finland, from 2012 December to 2013 April, and VAMK has provided all the equipment for this thesis.

Finally, I intend to exert my thanks to my parents for giving me unconditional support both economically and mentally, and the companion of all my friends.

Vaasa, Finland, 2013/4/12

Jin Cao

VAASAN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES
Bachelor of Engineering

ABSTRACT

Author	Jin Cao
Title	Controlling Telecom Instruments Using MATLAB
Year	2013
Language	English
Pages	45 + 1 Appendices
Name of Supervisor	Dr. Chao Gao

In the contemporary age, powerful measurement instruments are playing important roles in data measurement area. In the meantime, remote control instruments and process data have become a demand in the industry and education area. As remote control instrument could make instrument work under extreme environment that are not suitable for human beings, and developing of remote control software is inexpensive than hardware. In this thesis, my main focus are remote control of instruments and acquiring data from instruments to computer.

The sample instrument is Rohde & Schwarz FSP Spectrum Analyser, and MATLAB is the software development platform. The structure of the paper is performed as following. At beginning, the background information has been provided, including details toward software, hardware and programming languages. Then, second part introduced the GUI design and instrument connecting method, as well as MATLAB Instrument Control Toolbox. The third part specified the program functions' purpose, including connecting, tracing and settings part. Furthermore, an AM signal analysing example has been made, the details contain connecting instrument, setting parameters, tracing data, and adding markers. Based on the work of thesis, we can use the program to control and set the instruments remotely and find out the convenience to perform data measuring and processing work with the help of MATLAB and contemporary communication technology.

Keyword: MATLAB, Remote Control, Instrument, Spectrum Analyser

CONTENTS

LIST OF FIGURES, LISTINGS AND TABLES

LIST OF ABBREVIATIONS

ATT	Attenuation
GPIB	General Purpose Interface Bus
GUI	Graphical User Interface
IVI	Interchangeable Virtual Instruments
MATLAB	Matrix Laboratory
RBW	Resolution Band Width
REF	Reference
SCPI	Standard Commands for Programmable Instruments
TCP/IP	Transmission Control Protocol / Internet Protocol
UDP	User Datagram Protocol
VISA	Virtual Instrument Software Architecture
VXI	VME eXtensions for Instrumentation

1 INTRODUCTION

The introduction part contains the structure of this thesis, the intention and the structure of this project.

1.1 Structure of the thesis

This thesis started with an introduction about the intention of doing this project, and the overall structure of this project. After that, background information related with this project is introduced in Chapter 2, include development platform and programming techniques. Then, the program's flow chart and the program user interface design are explained in details. The fourth Chapter of this thesis illustrates some examples in the program that have covered the main function of the project. Moreover, fifth part of the thesis gives a specific demonstration about how to use this program by read out AM signal and analyse it. At last, the conclusion reviews the whole project and briefly state the conceivable develop areas of the project.

1.2 Intention of the project

The development of test and measurement instruments is explored considerably nowadays. With the massive expansion of communication technology, signal measurement and processing have become a major issue, it makes the measurement tools imperative for related works. A key concern in the measurement area today is to improve the user experience of using the measurement tools, and enhance the working efficiency at the same time.

Although current measurement tools have developed to be very accurate and powerful machine, however, as the limit of the instruments' hardware and embedded the operating system, the instrument is only capable for measuring the signal and displaying it on the small size screen of its own. In many cases, we need to measure data and process in software like MATLAB, TimeFlow. With the development of network technology, it is possible to control instruments and measure data remotely; this feature can greatly benefits the manipulator, which makes control multi instruments and remote collect data come true.

After specifying the two primary problems of modern instruments, here comes the motivation of doing this project. Firstly, control the instruments remotely by using current communication technology like cables, wireless signals. Secondly, make the measured data easily load into data processing programs for further process. Through working the project, we can have a relatively handy way to use instruments and analysing the signal from it.

1.3 Structure and objective of the project

The main part of the project is an application, which is built on MATLAB, together with the utilization of MATLAB driver and Instrument Control Toolbox. The reason why I choose MATLAB to be the main development tool is that MATLAB is a powerful data processing tool, which contains numerous mathematical functions. Moreover, MATLAB also handles image plotting and processing very well, and the main analysing object is the graph data obtained from the instrument. All that advantages of MATLAB made me choose it as my development environment. Figure 1 shows the overall structure of the project.

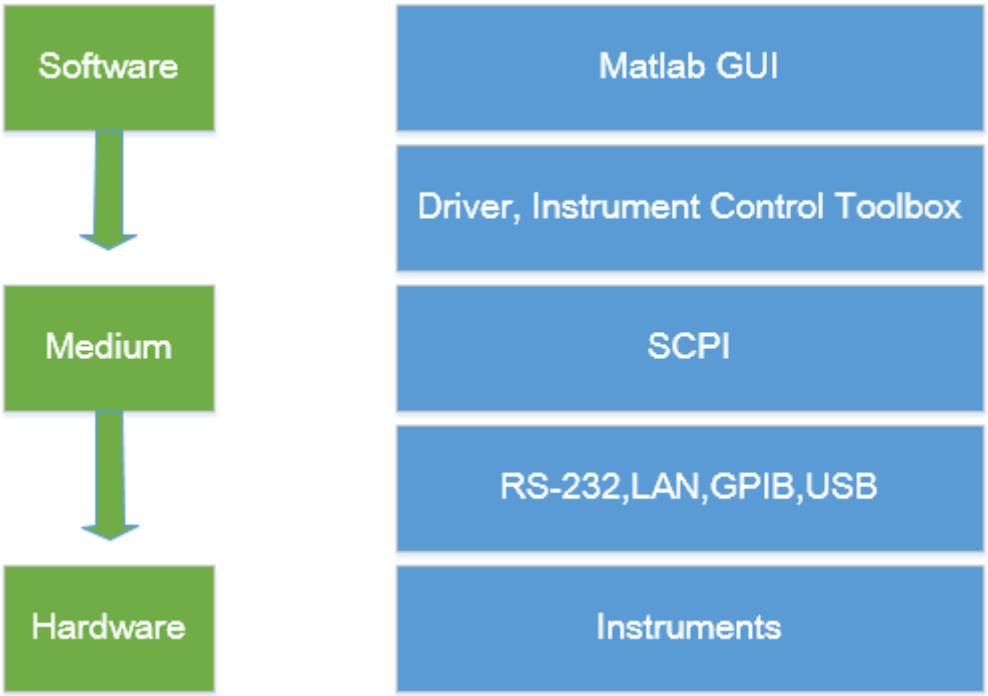

Figure 1 overall structure of the project

To explain it at first, is the common architecture, software and hardware connected by a medium. From the software, the core is a MATLAB program and its user interface, including SCPI (Standard Commands for Programmable Instruments). The medium here in this project is the RS-232 cable, and it also can be Ethernet, GPIB (General Purpose Interface Bus) if the instruments have the corresponding interfaces. For the hardware part, the instrument used in this project is Rohde & Schwarz FSP Spectrum Analyser. It likewise can be other instruments if the embedded system in the instruments have the interface and support remote control. [1]

More specifically, on the structure of the project, for the user interface part, MATALB GUI (Graphical User Interface) is used, and the function is also implemented in the MATLAB language. In order to connect to the instrument, MATLAB driver and Instrument Control Toolbox are also used in this program. They are VISA (Virtual Instrument Software Architecture) and GPIB drivers available for the connection as well. The medium of connection used in this project is RS-232 cable, and due to the working laptop does not have the RS-232 port, a USB to RS-232 adaptor and RS-232 cable were used in here.

2 BACKGROUND

This chapter introduces the tools and technologies used in the project, including MATLAB, MATLAB GUI, MATLAB Instrument Control Toolbox, SCPI and Rohde & Schwarz FSP Spectrum Analyser.

2.1 MATLAB

MATLAB is a high technology calculate environment that released by Mathworks company from America, it mainly focusses on scientific computation, visualization, and interactive program design. Utilization of MATLAB is multi-functional, like analysing data, developing algorithms, creating models and applications. The other notable feature is MATLAB has integrated numerous built-in mathematical functions, it makes a complex mathematical calculation to be relatively easy and intuitive. [2]

MATLAB is also an object-oriented programming language, it includes classes, inheritance, packages. Moreover, MATLAB can be used concurrently with other languages like C and Java. This feature has greatly improved the expansibility of the project made by MATLAB. [3]

With the high speed development of IT industry, MATLAB is widely used in numerous areas. The industry involved with MATLAB consists of the signal process and communication, image and video processing, system control, test and measurement, computational finance, computational biology. Besides, MATLAB has lots of useful toolboxes which can be used especially in the designated area, both functionally and from the field point of view.

2.2 MATLAB GUI and Instrument Control Toolbox

GUI (graphical user interface) is a significant tool between user and computer. It directly related to the user experience. MATLAB has integrated all the supportive functions into the GUI environment and also provide appearance, property, and action responds method settings control. More importantly, it has a very powerful drawing function, and we can have a high quality curve graph for research. At the

same time, one vital feature for this project is MATLAB support serial port operation. [4]

Another tool used in this project is Instrument Control Toolbox, this toolbox enables us directly connect to instruments like the function generator, spectrum analyser. The toolbox consists of a set of functions which provide us easy access to the instruments without writing specific codes for the instruments. The fundamental advantage for this toolbox is that it supports the most common instruments drivers like IVI, VXIplug&play, and numerous communication protocols like GPIB, VISA, serial, UDP, and TCP/IP. All these wonderful specialty makes the toolbox a best companion when you dealing with problems with connecting instruments. [6]

As the communication medium we used in this project is RS-232C, because the instrument that we have GPIB and RS-232C port, and RS-232C is the most commonly used and relatively inexpensive solution in this day and age.

2.3 SCPI (Standard Commands for Programmable Instruments)

Standard Commands for Programmable Instruments, as it literally means, a standard commands criterion for programmable instruments. The reason we need this standard is in the 1960s [8], different manufacturers have different controlling commands for its own instruments, and this phenomenon has caused a problem that it is expensive to build every application for different manufactures. In order to cope with this instrument control command consistency problem, SCPI is built up on the base of the IEEE 488.2 standard. It defines a set of generic functions that can be used for a variety of instruments. For instance, query frequency value is 'SENSe:FREquency', and query the instrument information is '*IDN'.

After we have the standard for the instruments control commands, it is easy to develop an application that suits a series of instruments. Besides, only minor modifications in the code needed if we need to change the manufacture of the instruments, it dramatically saves time for the development process. [7]

2.4 Instruments used (R&S FSP Spectrum Analyser)

2.4.1 What is spectrum analyser

Spectrum analyser is an instrument that measures the amplitude of a signal in the frequency domain. It measures the spectrum structure of an electrical signal, mainly used for detecting the degree of signal distortion, signal modulation depth, spectral purity, frequency stability, cross modulation distortion, and also the parameters of the amplifier and filter circuit system. It also called frequency domain oscilloscope, frequency analyser, and harmonic analyser or Fourier analyser. The result can be displayed either digitally or analogy.

2.4.2 Rohde & Schwarz FSP Spectrum Analyser

Rohde & Schwarz is a Germany company which provides products on electronic signal test and measurement equipment, the company has business in mobile communication, broadcasting, military, and radio industry. The model used in this project is FSP30 Spectrum Analyser. It is a general purpose spectrum analyser, measure frequency range from 9 KHz to 40 GHz, resolution bandwidth range from 1 Hz to 10 MHz [9]

2.4.3 Example structure of the device commands

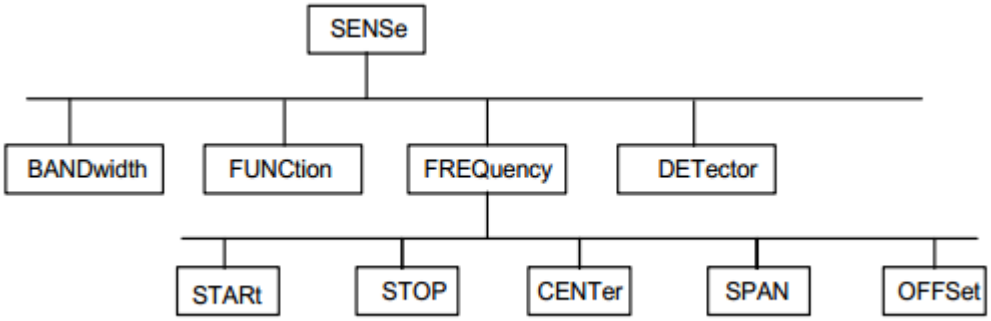

Figure 2 Tree structure of SENSE system [5]

Figure 2 shows the hierarchical structure of device commands. The word sense at the top indicates the system command name and the root, at the next level is the first branch which shows the specific detectable parameters, at the third level is

the sub collection of the parameter frequency, which contains the concrete and detail information of frequency.

Like the structure showed in the sense tree structure, the other subsystems of the instrument system have the analogous system structure. For instance, calculate subsystem, source subsystem, status system. In this project, the function of input parameters is achieved by input subsystem. Input subsystem handles the input values of the instrument. The state of the instrument is controlled by instrument sub-system. [11]

3 PROGRAM FLOWCHART AND UI DESIGN

3.1 Flowchart of the project

In this section, the flowchart of the program and the function of this project, the program design diagram are described.

3.1.1 Basic steps of establish serial port communication

To have a communication between the computer and the instrument, we need to have an object of the communication medium, in this case, it is RS-232 and the computer the port represents as COM number. According to the instruction of serial communication in MATLAB, we have the five steps to creating serial port communication in Figure 3.

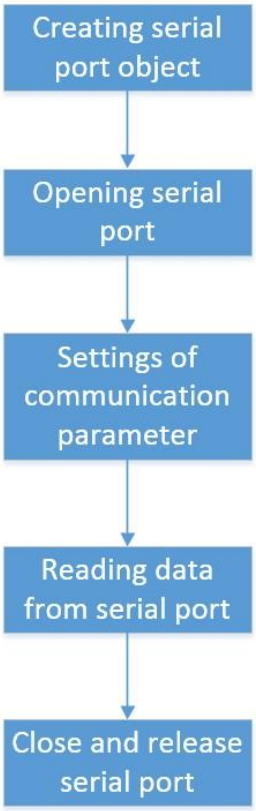

Figure 3 flow chart of creating serial communication

The procedures function example is as follows: [10]

1. Creating serial port object for the application

```
obj = serial('port number','PropertyName',PropertyValue)
```

2. Opening the serial port

```
fopen(obj)
```

3. Make settings of communication parameter like baud rate, parity bit.

```
Props = set(obj,'PropertyName',PropertyValue)
```

4. Reading data from serial port

`fgetl, fgets, fread, fscanf,` all these function can read from the serial port and they are used differently depending on your needs, like reading binary data or ASCII data.

5. Close the serial port and release the memory occupied

```
fclose(obj)
delete(obj)
```

3.1.2 Program serial communication flowchart

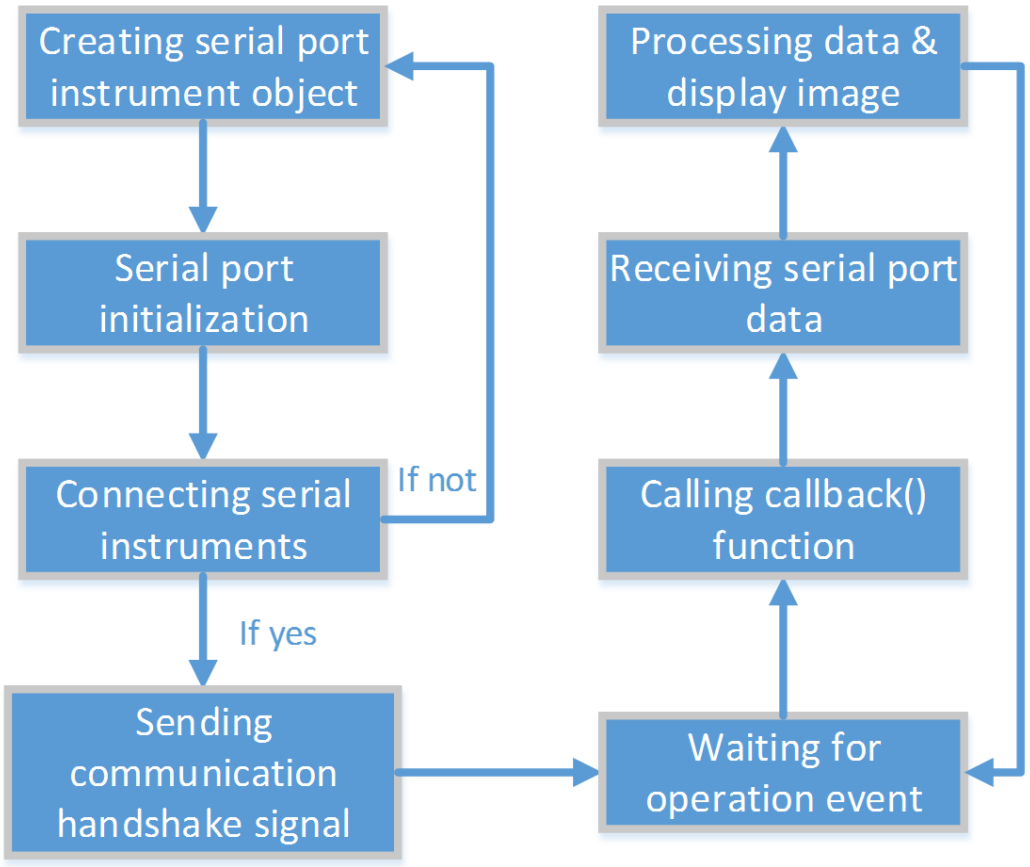

Figure 4 the program design flowchart

Figure 4 shows the overall program design procedure. In the program, each function is written into a call-back function, that made the program to be modularity and easy to expand or simplify.

3.2 User interface design according to the instrument

As this project is designed for spectrum analyser, the software instrument function part is special created for spectrum measurement, like the parameters about frequency and span setting, threshold line control, market control is the main part of the setting panel. Furthermore, the instrument connection setting part for most serial communication is identical, we only need to set the com port number, baud rate, data bit, parity bit, so these functions were designed in one panel. If we have the needs to change to another instrument like an oscilloscope, we merely need to

add the function panel part for controlling the oscilloscope and the connection settings can be remain the same.

3.2.1 Draft version of the program GUI

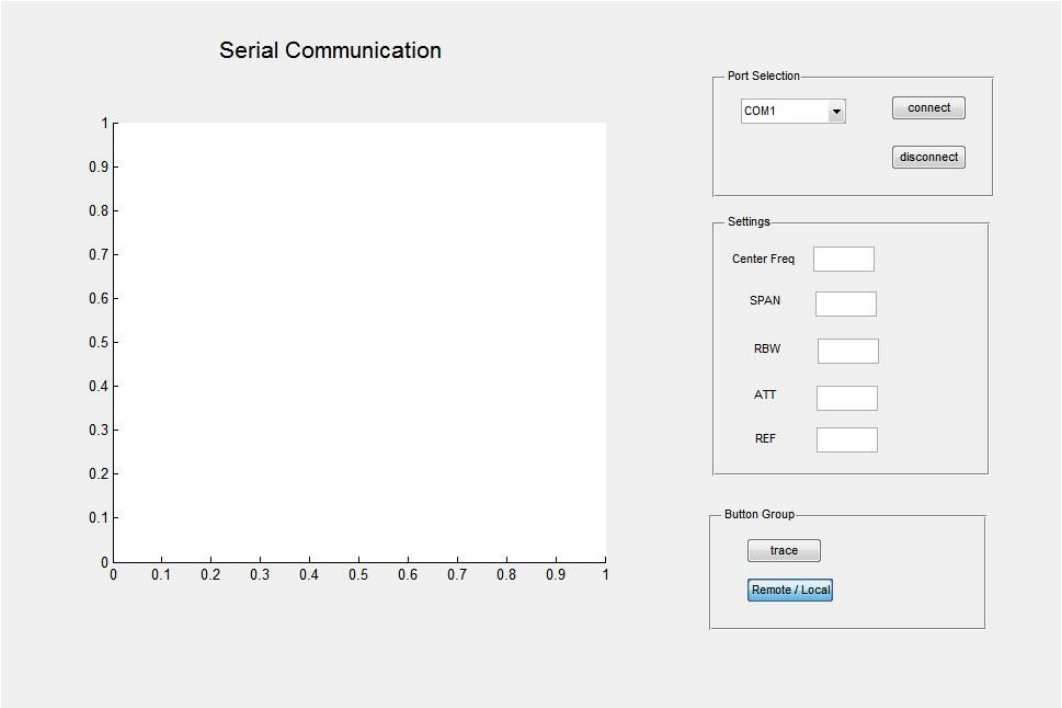

Figure 5 overview of the draft GUI sample

When the application was developing, the goal was to design the program to be an easy and intuitive tool. The core part of this issue is the graph data which obtained from the instrument, for that reason, a graphic panel for the data is the main part of the user interface.

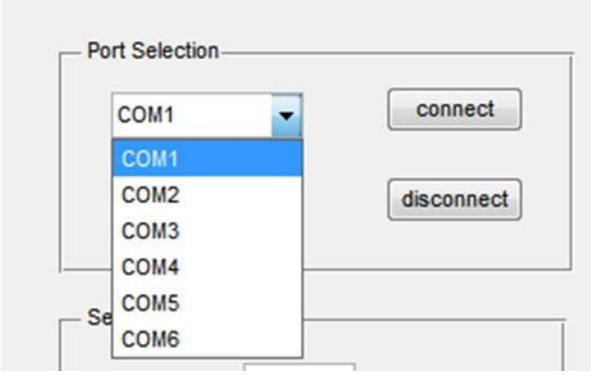

Figure 6 port selection draft version

Next comes the port setting part, in the draft version we can only choose the port number manually, and no data bit and parity bit options. In draft version, the design thinking is exploratory yet insufficient. Port control panel in Figure 6 is an example.

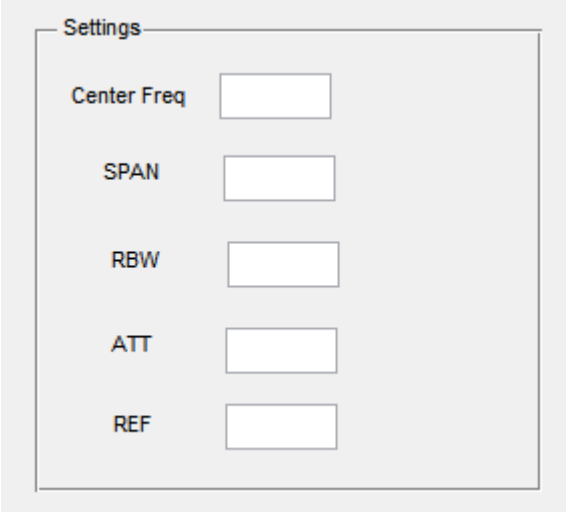

Figure 7 instrument settings draft version

In the instrument control setting part, the necessary function for controlling the spectrum analyser is achieved, centre frequency, SPAN, resolution bandwidth, attenuation, reference line.

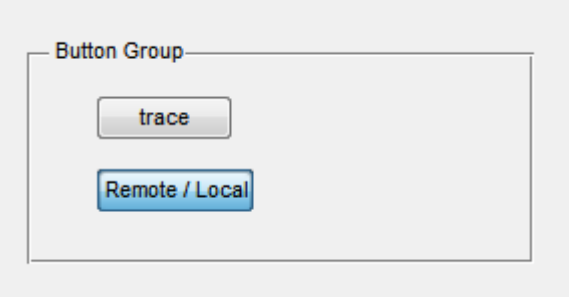

Figure 8 button group draft version

In this draft version, the program does not have much functions, the trace button simply traces the instrument's data and displays it on the axis. Moreover, remote and local button converts the status of the instrument, either in remote mode and local mode. And more functions and improvements will be introduced in beta version chapter.

3.2.2 Beta version of the program GUI

In this beta version, vast improvements on functionality and usability is done after further study on the spectrum analyser. The main purpose is to be more functional and efficient. The marker part is a completely new added part which could achieve the marker control function, to locate the maximum point in the curve, and minim point as well. Furthermore, the instrument function part is also expanded with some new features, like the input bar, we can entering raw SCPI commands and the program will display the feedback in a message box, just in case some uncommon functions, and it also useful with instrument testing purpose. The button group part is also better than the one in draft version. The specific information about the change is in later this chapter. Figure 9 shows an overview of the new designed GUI.

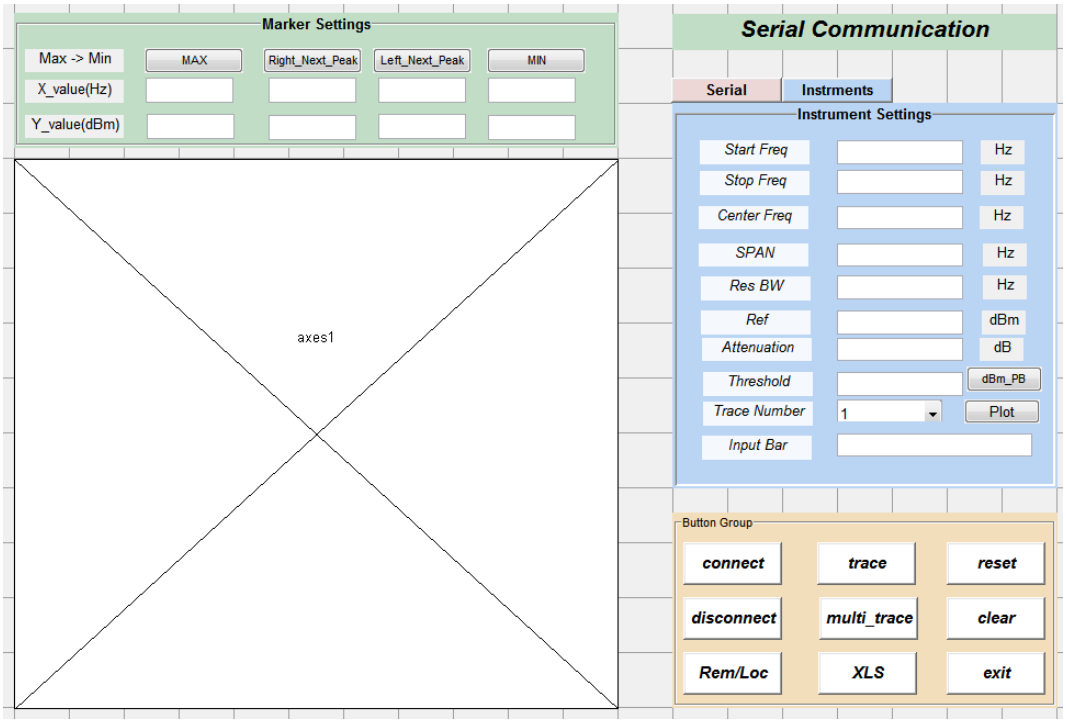

Figure 9 GUI overview of beta version

The details of the change is as follows:

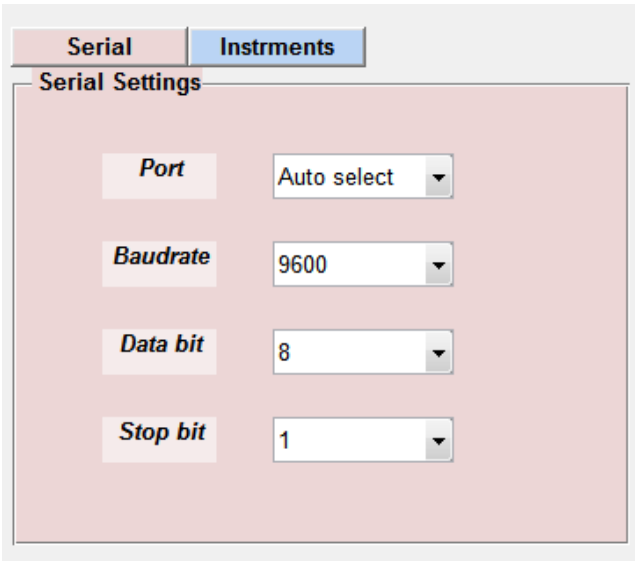

Figure 10 serial settings of beta version

In this version, the function of baud rate selection, data bit and stop bit selection is added, in this way, the program can be utilised in most serial communication environment and be more flexible when dealing with different kind of instruments.

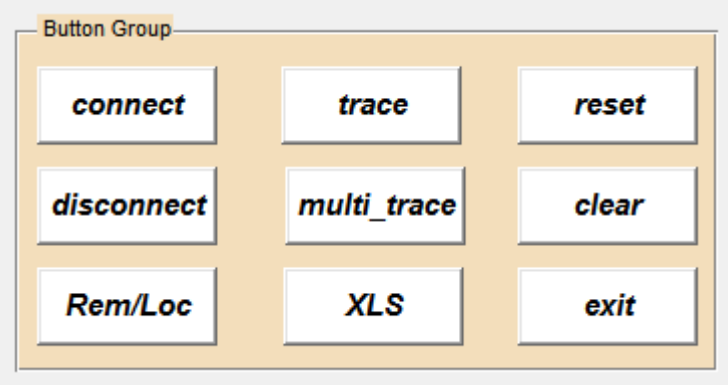

Figure 11 button group of beta version

Button group has been expanded with some new features. Table 1 is about the specific information about the buttons' function.

Table 1. Button functions

Button name	Function
trace	Trace the current data in the instrument and plot out the graph in the GUI's plot area.
multi_trace	Instead of trace and plot out the acquired data, it saves the data in MATLAB, and can be plotted as user requested later.
disconnect	Disconnect the instrument, delete the object of instrument.
Reset	Reset the instruments settings to the default.
Clear	Clear variables in the program.
XLS	Saving the multi_trace value data into an excel file.
exit	Clear the memory, delete all the objects, and close the program.
Rem/Loc	A toggle button that changes the instrument mode, remote or local.

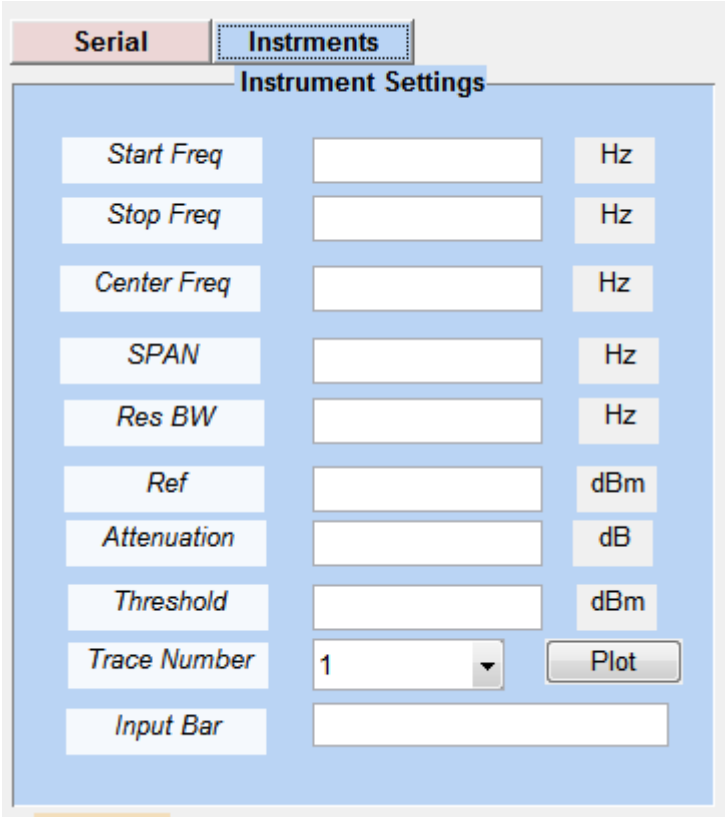

Figure 12 instrument setting of beta version

In the instrument setting panel, the program has added with the start and stop frequency, threshold line, trace number, and input bar as showed in the picture. Actually, we can control the instrument's frequency settings by either start and stop frequency or centre frequency and SPAN, the reason why added these options is in different measurement tasks, using one of them can be very productive and easy. More importantly, an input bar has been added, this acts like the terminal, if there is a command need to test or unusual function needs to be used we can simply type the command in without modifying the program.

4 MAIN PROGRAM FUNCTION LIST

4.1 Connecting instrument part

4.1.1 Detecting COM port

Auto detect available COM port is a humanized function for users. In this part, the 'instrhwinfo' function was used to find all the available serial port and display it in the list. In case some different circumstance situations, the option was given as manually select the port number from COM1 to COM12. In the picture above, COM5 is the auto detected an available port. The sample code and Figure 13 show the function.

Listing 1: detecting COM port example

```
% auto detect the available serial ports and display in the port
list
% define the manual select port string
port_str =
{'COM1','COM2','COM3','COM4','COM5','COM6','COM7','COM8','COM9','C
OM10','COM11','COM12'};
% find out the available serial ports
serialPort = instrhwinfo('serial');
% generate the port string to the port selection menu
set(handles.pop_port_number, 'String', [{'Auto select'} serial-
Port.SerialPorts,{'Manual select'} port_str]);
```

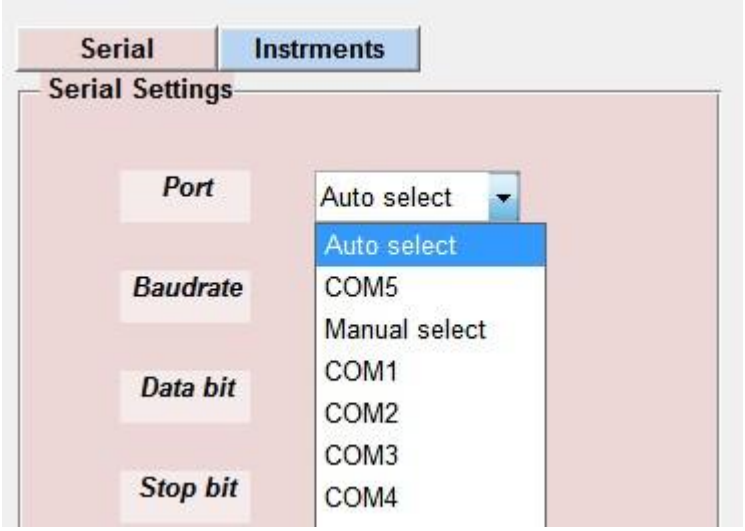

Figure 13 auto select serial port

4.1.2 Port initialization and parameter setting

In this part, port initialization and instrument object parameter are explained. In the code below, first we get the port selection number from GUI, and then get all the strings in the port selection, after we have the number and strings, we define the exact string to the instrument object's property. All the properties here like COM port number, baud rate, data bit stop bits are all set in the same form. In port initialization, we can change the parameters of the serial port settings, in that way, we can make the program adapt to different instruments' requirement for the serial communication.

Listing 2: port initialization and parameter example

```
global fsp;
% ----------------------port initialization-----------------------
all_com = get(handles.pop_port_number,'String');
com_value = all_com{get(handles.pop_port_number,'Value')};
fsp = serial(com_value);
% ----------------------baudrate initialization---------------------
baudrate_number = get(handles.pop_baudrate,'Value');
baudrate_string = get(handles.pop_baudrate,'String');
baudrate_value = baudrate_string(baudrate_number);
set(fsp,'baudrate',str2double(baudrate_value));
% ----------------------data bit initialization---------------------
databit_number = get(handles.pop_databit,'Value');
databit_string = get(handles.pop_databit,'String');
databit_value = databit_string(databit_number);
set(fsp,'databits',str2double(databit_value));
% ----------------------stop bit initialization---------------------
stopbit_number = get(handles.pop_stopbit,'Value');
stopbit_string = get(handles.pop_stopbit,'String');
stopbit_value = stopbit_string(stopbit_number);
set(fsp,'stopbits',str2double(stopbit_value));
```

4.1.3 Open the port

In this section, the input buffer size and time out value is defined by the parameter of the instrument object. In here the input buffer size is defined as 2048, and the default is 512. This will be explained in the next chapter in 4.2.2. The timeout value, if it is too big, it takes longer time waiting for the instrument's response, if it is too small, the instrument cannot respond in such a short time, and this will cause the reading and timeout error.

After opening the instrument object, there is a query to ask the instrument's identification, from the feedback of the instrument, we can make sure the instrument we connect is exactly the one that we expected. An example of the code is as follows.

Listing 3: An example of open the port

```
% define the input buffer size and time out for serial object
fsp.InputBufferSize = 2048;          % input buffer size in bytes
fsp.Timeout = 3;                     % timeout in seconds

% Connect to instrument object, fsp.
fopen(fsp);

% Get feedback from the instrument object fsp.

% query the identification of the instrument
fprintf(fsp,'*IDN?');
% read identification from instrument
output = fscanf(fsp);
```

4.2 Set parameter and acquiring data part

In this section, the codes of parameter setting and data acquiring are introduced. More specifically, the parameter setting is showed with an example of setting the centre frequency, and the pre-settings of obtaining graphical data, codes obtain trace data are explained.

4.2.1 Frequency settings

The settings part's principle is almost the same, for that reason, only one significant example for setting the centre frequency value is showed. The process begins with getting the value from user input, and write the value into instrument object.

Listing 4: A parameter writing example

```
function start_freq_value_Callback(hObject, eventdata, handles)

global fsp;
% getting the input string from GUI
handles.start_frequency = get(hObject,'String');

% convert the obtained string to double for comparison
```

```
sf = str2double(handles.start_frequency);

% make sure the input value is in the range of the instrument
if (sf >= 9000) && (sf <= 30e9)
    % writing the input value into the instrument
    fprintf(fsp,['FREQ:STAR ' handles.start_frequency 'HZ']);
else
    warndlg('The input value is out of range. [9k, 30G]','Error!');
end
guidata(hObject,handles);
```

4.2.2 Pre-setting before read out data

Before acquiring data from the instrument, it is needed to define several settings like spectrum mode, sweep method, synchronization method. In this part, the three mentioned settings were explained.

About the sweep method, we have continuous and single two options, in this program, single sweep method are deployed for two reasons. The first reason is when we use a continuous sweep, the instrument need to calculate the trace data's parameters, it takes a lot of instrument's computational resource which causes the delay in transmitting data. The other reason is when we obtain data from an instrument, the ideal data are a complete and desired piece, so the trace time is extremely important. However, in continuous sweep mode, it is difficult to keep the perfect timing all the time. Besides, real time display in MATLAB also is a challenge for the computer's calculation ability. [12]

One important thing needs to be explained here is the sweep point number. As we use single sweep mode to obtain data from the instrument, the number of 501 means in one single sweep the instrument generate 501 points for the data displayed on the screen. Each sweep point is 32 bits floating number which is equal to 4 bytes, and 501 sweep points take 2004 bytes space in total. The input buffer size has to be bigger than 2004, so 2048 bytes as the input buffer size is set in the program.

The measurement with synchronization is achieved by WAI command. When we receive data from the instrument, we must make sure that the instrument has com-

pleted the measurement and be able to transmit a single sweep. WAI is an IEEE 488.2 common command, which provides an easy way to synchronize the instrument. With the added WAI command, WAI is executed after a complete sweep operation. [13]

Listing 5: An example of pre-setting when acquiring data

```
% ---------------------Instrument setting before acquire data-----
% set instrument to spectrum analyser mode
fprintf(fsp,'INIT:SEL SAN');

% switch on the update of all display elements during remote con-
trol
fprintf(fsp,'SYST:DISP:UPD ON');

% number of measurement samples acquired during a sweep
fprintf(fsp,'SWE:POIN 501');

% turn off continuos sweep mode
fprintf(fsp,'INIT:CONT OFF');

% starts the measurement with synchronization
fprintf(fsp,'INIT;*WAI');
```

As the x-axis for the spectrum analyser demonstrates the frequency domain, in order to have the proper frequency domain value, the start frequency value and the SPAN value were taken to calculate the correct frequency domain. The formula is as follows:

$$\text{step value} = \frac{SPAN}{length\ of\ traced\ data}$$

After obtained the step value, and we know the start frequency value, we can calculate the x-axis range is from the start frequency to the start frequency plus SPAN minus one step value.

x axis range =(start frequency):(step value):(start frequency + SPAN – step value)

Next the data transmitting type as binary data and the transmitting port one are defined. Actually, we have binary and ASCII two formats to choose, with ASCII

format, only the list of raw amplitude values is exported, and it is relatively difficult to control the exact length of the exported data. The method applied in this program is binary data, and the details of the procedure will illustrate later in this chapter.

Listing 6: Example of axis value and trace data type

```
% getting the start frequency and SPAN value for x-axis
fprintf(fsp,'FREQ:START?');
start_freq=str2num(char(fread(fsp)'));
fprintf(fsp,'FREQ:SPAN?');
span=str2num(char(fread(fsp)'));

% define the data acquiring method
fprintf(fsp,'FORMAT REAL,32');
fprintf(fsp,'TRAC1? TRACE1');
```

When we use binary format to obtain data, according to the programming examples in the specification, there are three steps to read out the data.

1. The number of digits in the length specification is read out.
2. The length specification itself is read out.
3. The trace data itself is read out. [14]

After we have the y axis data, we need to shift the dimension of the data from X*1 to 1*X format due to the x axis data is 1*X format. For the calculation of x axis the detail explanation is in the earlier this chapter. The example of the read out data code is as follows.

Listing 7: An example of acquiring data

```
% getting the lenth of the data
% the number of digits in the length specification is read out
fread(fsp,1);
number=fread(fsp,1);
number_value=str2num(char(number));

% the length specification itself is read out
result=fread(fsp,number_value);
data_length=int32(str2num(char(result')));
mod_part=mod(data_length,buffer);
mod_number=mod_part/4;                  % 4 bytes makes a point
```

```
                                        % 32-bits floating number
% acquiring y axis data

% acquire record to MATLAB
data=fread(fsp, double(mod_number),'single');
% reshape data for ploting
trace_data = reshape(data,1,mod_number);

% caculate x axis data

% calculate each step of x axis
step = span/length(trace_data);
% generate x axis number
x_freq = start_freq:step:(start_freq+span-step);
```

4.3 Other functions

In this chapter, some other functions of the project is introduced, like plotting, disconnect instrument, marker control.

4.3.1 Plotting data part

In the plotting figure part, there is a value detection to display the unit of frequency for the better human readable axis. The units are Gigahertz, Megahertz, kilohertz and Hertz, all the common used frequency units are included.

Listing 8: An example of detect frequency unit

```
% determine x axis units in GHz MHz KHz Hz
if start_freq > 1e9
    x_freq = x_freq/1e9;
    plot(x_freq,trace_data);
    xlabel(handles.axes1,'frequency (GHz)');
else if start_freq > 1e6
    x_freq = x_freq/1e6;
    plot(x_freq,trace_data);
    xlabel(handles.axes1,'frequency (MHz)');
else if start_freq > 1e3
        x_freq = x_freq/1e3;
        plot(x_freq,trace_data);
        xlabel(handles.axes1,'frequency (KHz)');
    else
        plot(x_freq,trace_data);
        xlabel(handles.axes1,'frequency (Hz)');
    end
    end
end
```

4.3.2 Disconnect the instrument

In this example, the operation of disconnect the instrument is explained. When disconnect the instrument, first put the instrument back to local mode, and then delete the instrument object. The reason why we add this function is there are situations that we need to disconnect the instrument and connect to another instrument. Here is an example of the disconnect button function.

Listing 9: an example of disconnect the instrument

```
% --- Executes on button press in pb_disconnect.
function pb_disconnect_Callback(hObject, eventdata, handles)

% Disconnect from instrument object, fsp.
global fsp;
fprintf(fsp,['@LOC']);      % set instruments back to local mode
fclose(fsp);
% Clean up all objects.
delete(fsp);
clear fsp;
guidata(hObject,handles);
```

4.3.3 Marker control

In frequency measurement, it is an essential function to have markers to label the value like peak value or valley value. With the spectrum analyser that we used, we have a calculate marker control subsystem provides numerous control functions with four markers.

In the marker control part, because we can store the values in MATLAB, so it is possible to control one marker for different purposes. In this program, we have max value detection, and right/left second biggest value, and minim value detection. By controlling the marker, we will have corresponding values read from the instrument and display it in the GUI. By recording the value into MATLAB, it is intuitive to see those values. Here is an example code of marker control button function of move the marker to the next right peak.

Listing 10: An example of marker control function

```
% --- Executes on button press in pb_mark_two.
function pb_mark_two_Callback(hObject, eventdata, handles)

global fsp;

% define the marker move property
fprintf(fsp,['CALC:MARK1:MAX:RIGH ']);

% obtain X-axis value of Marker 1
% query the marker 1's x axis value
fprintf(fsp,['CALC:MARK1:X? ']);
% read value from instrument
output_x = fscanf(fsp);
% display the value in program GUI
set(handles.two_x_mark_value,'String',str2num(char(output_x)));

% obtain Y-axis value of Marker 1
% query the marker 1's y axis value
fprintf(fsp,['CALC:MARK1:Y? ']);
% read value from instrument
output_y = fscanf(fsp);
% display the value in program GUI
set(handles.two_y_mark_value,'String',str2num(char(output_y)));

guidata(hObject,handles);
```

4.3.4 Toggle button

There have situations that we need to release the control of instrument or we need to doing operation directly on the instrument, for this reason, the button to switch the control state of the instrument is added. The method is done by change the value of the toggle between zero and one. Here is an example of the toggle button function.

Listing 11: An example of toggle button

```
% --- Executes on button press in pb_toggle.
function pb_toggle_Callback(hObject, eventdata, handles)

global toggle;
global fsp;
if toggle == 1
      % set instruments back to local mode
      fprintf(fsp,['@LOC']);
else if toggle == 0
      % set instruments back to remote mode
```

```
      fprintf(fsp,['@REM']);     end
end
% change the toggle value to its inverse
toggle = ~toggle;
guidata(hObject,handles);
```

5 ANALYZE SIGNAL SAMPLE

In this section, an example was made which shows the function of the project by acquiring, analysing an AM signal.

5.1 Instruments needed

The equipment used in this example is as follows:

- Rohde & Schwarz FSP30 Spectrum Analyser
- HEWLETT PACKARD E4420B ESG series Signal Generator
- Radio Frequency cable (RF Cable connectors)
- Computer installed with MATLAB version R2011b

5.2 Sample signal settings

Amplitude modulation is an old and simple modulation technique in telecommunications. One of the common utilization is AM radio. In the AM signal analysis, carrier frequency, sidebands frequency are two important factors when we judge the AM signal. In this program, we can see the carrier frequency and sideband frequency of the signal in the graph.

The sample signal settings is as follows:

- Frequency: 100.0 MHz
- Amplitude: -40.0 dBm
- Depth: 50.0%
- Rate: 2.0 KHz
- Waveform: sine

5.3 Measuring signal

In this part, the detailed information about using this program is showed.

5.3.1 Step 1 connecting instrument

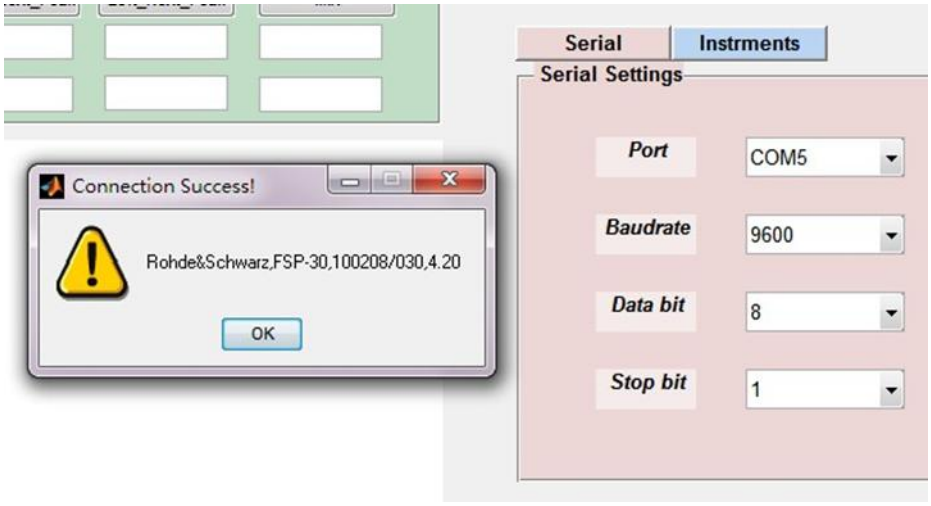

Figure 14 sample connection successful

Firstly, after start the program, we choose the auto select port, COM5, default instrument baud rate 9600, data bits 8, stop bit 1 and click connect button. If the port is available and the connection between computer and instrument is right, we should see a pop up box remind us the instrument's type and the connection is successful.

5.3.2 Step 2 setting parameters

After we have the connection, we can change the right panel to instrument panel to set the signal settings. According to the sample signal parameter mentioned in chapter 5.2, we set the centre frequency 100 MHz, reference line as -40 dBm, SPAN 10 KHz,

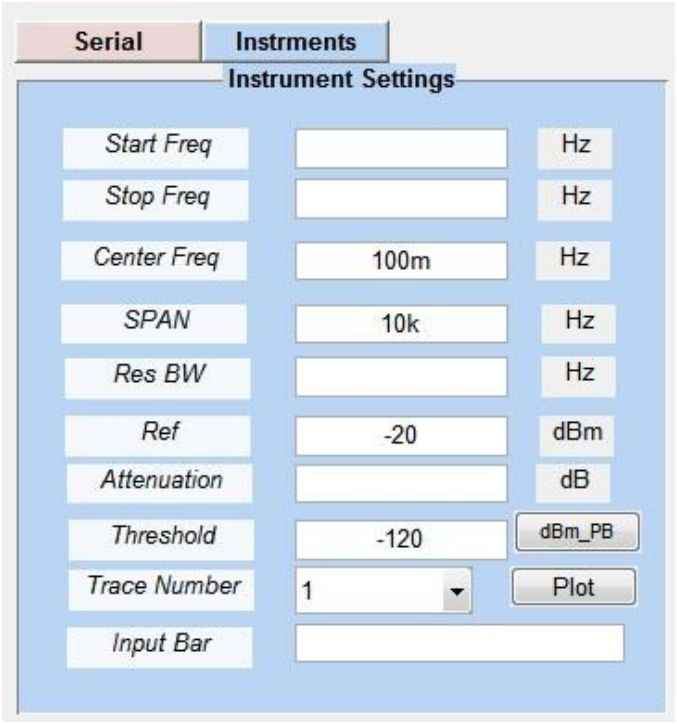

Figure 15 setting parameters example

5.3.3 Step 3 acquiring data

When all the parameters and settings are finished, click the 'trace' button or 'multi_trace' button to read out the trace data. For the button 'trace', if data is successfully read out, the data will directly showed on the program's GUI part, and for the button 'multi_trace', if it is successfully read out, a message box will pop up to show you the number of the trace and the data is saved to the program. By using the trace number selection and 'plot' button, we can draw out the trace line.

5.4 Measured signal sample graph

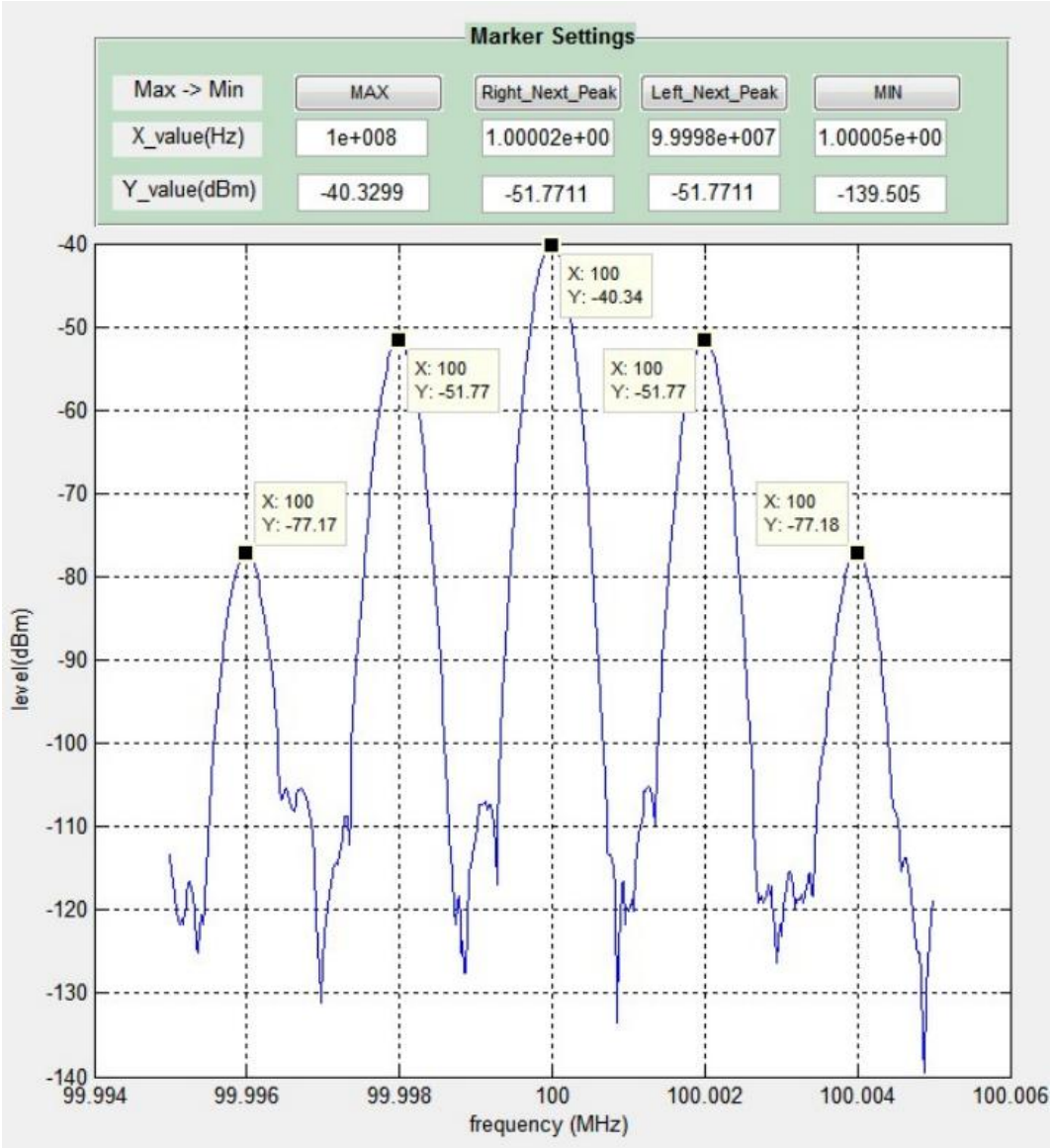

Figure 14 measured signal sample

From the measured signal, we can see that each peak point's value is displayed on the marker area in marker settings panel.

The operation of getting marker value started from the 'MAX' button, the button controls the marker 1 to locate on the maximum value of the curve. Then we can obtain the second biggest peak on the right side of the biggest value by clicking the 'Right_max_value' button, and if we need the third biggest value on the right side, we can click the button again. For the left side peak value, the method is the

same, just click the button and we can see the marker 1's position is change from peak to peak, when at the desired peak, the value can be seen from the marker setting panel.

If we need to compare several signals, the function 'multi_trace' can do the job. Pressing the 'multi_trace' button for the curve we want to compare, and after that plot the curves from the trace number selection bar and we will get a small figure for each trace data. In that case, it is relatively easy to observe, making comparison. Figure 17 is an example of comparing two example signals with different SPAN parameter.

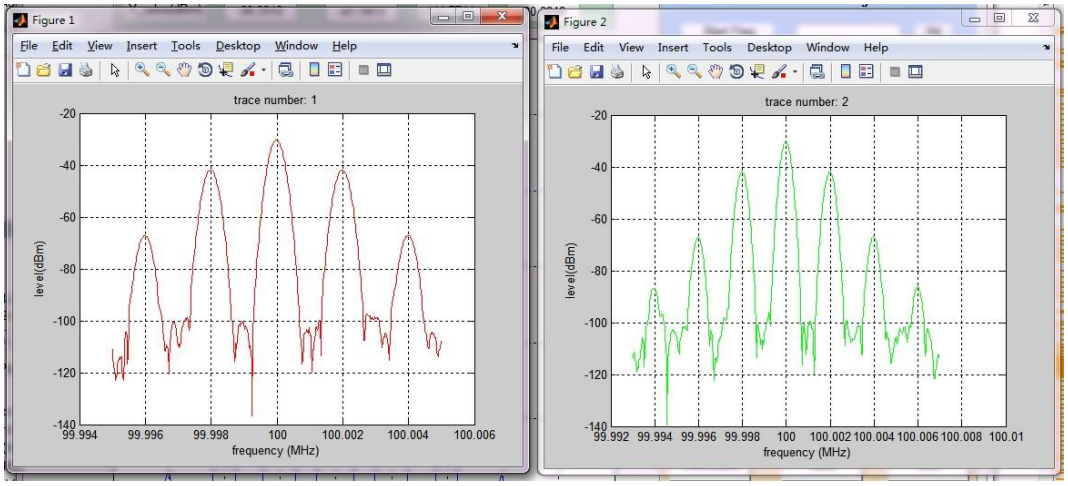

Figure 17 graph comparison

6 CONCLUTION

6.1 Review of the project

With all the work accomplished in this project, the function of remotely controlling Rohde & Schwarz FSP30 spectrum analyser and acquiring data into MATLAB have been achieved. More specifically, remote controlling the setting of centre frequency, start/stop frequency, SPAN, threshold line, reference line, resolution bandwidth, attenuation all operate well. The data in the spectrum analyser can be correctly taken over and displayed in MATLAB, with x-axis frequency domain and y-axis signal strength domain. Furthermore, with the multi trace function, up to eight traces can be stored in MATLAB, for research comparisons and corresponding work.

From this project, we shall have a conclusion that the combination of software and hardware that makes the work environment better. Furthermore, remote controlling instruments could be a trend for the instrument research direction, with the help of good design human-machine interaction software, it is possible for each one of us manipulates the instruments without knowing much professional knowledge or reading the manual of the instrument. Remote control has the potential to change the way we using the instruments nowadays and also give many possibilities to design more complex and intelligent instruments.

About the usage of remote control instruments would be widely deployed in the coming years. When integrated with automated computer, auto controlled instruments can be developed and used in some extreme environments.

6.2 Possible orientation of remote control instruments

After doing the project, there is one part could be added in the future. When we have the data in MATLAB, a varies of signal processing functions can be built into the application which would make the signal acquiring, signal presenting and signal processing be done in one application.

The other possible developing area is to make the medium wirelessly. For instance, a Wi-Fi modular can be integrated into the instrument, in that case, the instrument is just like a computer, we can exchange data with the instrument without the limitation of cable and have a much more convenient way to process data.

REFERENCES

/1/ Video, Deploying Applications with MATALB <URL: http://www.mathworks.se/videos/deploying-applications-with-matlab-68742.html> Accessed on 2013/3/08

/2/ MATALB, The Language of Technical Computing <URL: http://www.mathworks.se/products/matlab/> Accessed on 2013/3/09

/3/ MATLAB Programming Language

<URL:http://www.altiusdirectory.com/Computers/matlab-programming-language.php> Accessed on 2013/3/12

/4/ MATLAB GUI <URL: http://www.mathworks.se/discovery/matlab-gui.html> Accessed on 2013/3/13

/5/ Rohde & Schwarz GmbH & Co. KG (2009). Operating Manual for the R&S FSP Spectrum Analyser. Chapter 5.6 Structure and Syntax of the Device Messages

/6/ MATLAB Instrument Control Toolbox overview <URL: http://www.mathworks.se/products/instrument/ > Accessed on 2013/3/16

/7/ what is SCPI <URL: http://www.jpacsoft.com/scpi_explained.htm > Accessed on 2013/3/17

/8/ Standard Commands for Programmable Instruments (SCPI) Volume 1: Syntax and Style VERSION 1999.0

/9/ Rohde & Schwarz FSP30 <URL: http://www.rohde-schwarz.com/en/product/fsp-productstartpage_63493-8043.html > Accessed on 2013/3/18

/10/ Serial port I/O communication <URL: http://matlab.izmiran.ru/help/techdoc/matlab_external/ch_seria.html#105616 > visited on 2013/3/19

/11/ Rohde & Schwarz GmbH & Co. KG (2009). Operating Manual for the R&S FSP Spectrum Analyser. Chapter 6.1 description of commands, introduction

/12/ Johannes Ganzert (2007). Hints and Tricks for Remote Control of Spectrum and Network Analyzers Application Note Chapter 3 Optimizing Remote Control Operation

/13/ Johannes Ganzert (2007). Hints and Tricks for Remote Control of Spectrum and Network Analyzers Application Note Chapter 5 Measurement Synchronization

/14/ Spectrum Analyzer Quick Start Guide Rohde & Schwarz Test and Measure ment Appendix Reading Out Trace Data

.