VAASAN AMMATTIKORKEAKOULU
VASA YRKESHÖGSKOLA
UNIVERSITY OF APPLIED SCIENCES

GOLD AYOOLA ABIODUN

PC CALIBRATION OF MEASURING INSTRUMENTS

Information Technology

2015

PREFACE

I will like to express my gratitude to the Almighty God for seeing through my journey as a student of Vaasa University of Applied Sciences. I will also like to thank Finland as a country for giving me the privilege of acquiring such a quality education at no cost.

I also will like to thank all the lecturers who have taught me all through my study. I am who I am in this profession because of every one of you. Most importantly, I will like to thank Dr Ghodrat Moghadampour for his professional and inspirational impact in my life.

Special thanks to Santiago Chavez for his professional, inspiration and care both as my lecturer and as my supervisor in this project. Thanks also to Jani Ahvonen and Timo Rene for their support all through the project.

Finally, I dedicate this project to the Almighty God once again and to my good friend Faleye Olusola Benjamin who passed away in his course of study.

VAASAN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES
Information Technology

ABSTRACT

Author	Gold Ayoola
Title	PC Calibration of Measuring Instrument
Year	2015
Language	English
Pages	74
Name of Supervisor	Santiago Chavez

Calibration involves the adjustment of measuring instruments basically by comparing the values obtained from a measuring instrument with a standard instrument whose output value is known.

This project is focused at developing an application used to calibrate measuring instruments (oscilloscope) in the laboratory. This application eases the traditional inputting of output value manually from the calibrator (Fluke 5500A in this case) to the oscilloscope (Agilent DSO5012A oscilloscope in this case). In addition, the application determines if the obtained measured voltage from the oscilloscope falls within a range of minimum and maximum voltage calculated based on the calibrator outputted voltage. The application finally displays this data on a table for easy analysis.

Furthermore, the project was developed on the .NET environment in which the graphic user interface (GUI) is a basic windows form. Visual Studio 2012 was used for the design, compilation and execution. C# language was used as the programming language.

Finally, testing and running the application shows its usefulness in easing the calibration of the measuring instrument (oscilloscope) in the laboratory as well as analyzing and keeping record of the values obtained.

TABLE OF CONTENTS

LIST OF FIGURES, TABLES AND CODE SNIPPETS

ABBREVIATIONS

AC	Alternating Current
CLR	Common Language Runtime
DC	Direct Current
FCL	Framework Class Library
GPIB	General Purpose Interface Bus
GUI	Graphic User Interface
NI	National Instrument
PC	Personal Computer
RTD	Resistance Thermometer
UI	User Interface
USB	Universal Serial Bus
VISA	Virtual Instrumentation Software Architecture

1 INTRODUCTION

Calibration is the process of comparing the measured value from a measuring instrument with that of another instrument with a known value. In the calibration process, the equipment with the known value is called the standard while the equipment whose value are been compared is called the test instrument. Standard vary from country to country. The calibration process involves careful selection of the standard which has less than 0.25 of the measurement uncertainty of the test instrument.

During the calibration process, there might be a need for adjusting the test instrument in case the values measured are not in conferment with the standard. This process of adjusting the test instrument is known as the adjustment process. In all, the whole process is known as the calibration process. The calibration process can be done manually or automatically depending on the devices involved.

Electronic measuring equipment is composed of various components for instance resistor, capacitor and transistor. The functioning value of this components naturally drift over time, therefore resulting in the need for a calibration process at interval. The calibration process is an important process for measuring equipment in order to ensure the accuracy of the equipment and also reduce the measurement error of the equipment.

1.1 Background

Advancement in technology as well as the development in programing skill has provided a platform for the automation of calibration process. Automation of the calibration process improves the documentation of the obtained calibrated values which makes it easy for decision making from the history recorded. Automation of the calibration process in total makes the calibration process easy in terms of sending and reading values from the instrument involved.

This project is focused on the development of an application for the calibration of measuring equipment. Based on the naturally drifting of the function value of the components that make up the measuring equipment, it is important to make the calibration process on an interval basis. This application therefore makes the calibration process easier and convenient as well as making the documentation of record easy.

1.2 Objective

The objective of this application is to develop a software that eases the calibration process of measuring instruments. In the case of this project, the application calibrates the oscilloscope using values (voltage and frequency) sent from the calibrator. Both instruments are connected to the PC using the GPIB – USB connector.

In addition, the application provides the user with a platform for selecting if the calibration would be done manually or automatically. In the manual section, the application provides the user with a platform where the user can input values (voltage and frequency) and send them to the calibrator. Also, the user can adjust the oscilloscope and finally measure the value (voltage and frequency) measured by the oscilloscope. While in the automatic section, the application provides the user with the privilege of either making a table of values (voltage and frequency) to be sent to the calibrator or load values from a saved file to the calibrator.

Furthermore, the application, makes a calculation of minimum and maximum voltage values based on the voltage sent from the calibrator. The application then checks if the voltage measured from the oscilloscope is within the range of the minimum and maximum value calculated.

Finally, the application provides the user with a platform for saving the data (outputted voltage and frequency from the calibrator as well as measured voltage and frequency from the oscilloscope) for historical references.

2 TECHNOLOGY AND PERFORMANCE ISSUE

2.1 System Reference

The application developed in this project was not developed based on any existing system, therefore, the whole application was designed and developed based on the requirement of the client. In this project, the application was developed using the .NET environment. The user interface (UI) is a basic windows form using Visual Studio 2012 as the designer. The programing language used throughout the development of this application was C# language. Other instruments used in the development of this application includes the following

- Fluke 5500A calibrator

- Agilent DSO5012A Oscilloscope

- GPIB Bus Cable

- GPIB – USB connector.

The setup of the devices can be seen in page 50 of this write up.

.Net Framework

The .NET framework is a programming software framework developed by Microsoft and it runs primarily on Microsoft Windows operating system platform. The .NET framework is a tool that makes programming easy for software developers. The .NET framework is composed of two main components which includes

- **FCL** (Framework Class Library): This is a standard library in the .NET framework consisting of several predefined classes, interfaces and value types.
- **CLR** (Common Language Runtime): This is a virtual machine in the .NET framework which compiles and executes the program. Technically, takes the program written by the programmer, compiles it and then converts it to machine language that can then be executed by the central processing unit (CPU). In addition to this feature, the CLR is also responsible for debugging, memory management and security in the .NET framework.

The .NET framework exhibit certain features which includes

- Cross Language Interoperability: This is a platform in which the .NET framework provides functionality between different programming languages with similar data structure.
- Portability: This platform makes it possible for other operating systems to implement the .NET framework.
- Security
- Memory management

C#

The C# programming language is a programing language designed and developed by Microsoft. The language is included and supported in the .NET framework. C# language is an object oriented programming language which suggest that it supports encapsulation, inheritance and polymorphism.

Visual Studio

Visual Studio is an integrated development environment developed by Microsoft. It is used in the development of console application, graphical user interfaces, web services, web application and windows forms.

Visual Studio supports the writing of native and managed codes supported by Microsoft Windows, Windows mobile, Microsoft Silverlight, .NET Compact framework and .NET framework. In addition, the code editor supports intelliSense and code refactoring. Also, the visual studio incorporates a bugger for both source and machine level debugging.

Fluke 5500a Calibrator

The fluke 5500A calibrator is a device used for the calibration of a wide range of devices which includes multimeters, thermometers (thermocouple and RTD) handheld wattmeter, data loggers and current lamp.

In addition, the fluke 5500A calibrator offers the traditional meter source function which includes voltage, current and resistance. The power output can be stimulated using either the dc or ac outputs which enables sourcing voltage or voltage plus current at the same time with precise phase control.

Technical specification for the fluke 5500A identifies a range of value the calibrator is capable of outputting. The following includes some of the values and their ranges

•	Voltage (DC)	0 to ±1020V
•	Current (DC)	0 to ±11A
•	Resistance	0 to 3.29.999 MΩ
•	Voltage (AC)	1mV to 1020V
•	Current (AC)	29mA to 11A

Agilent DSO5012A Oscilloscope

This is a two channel oscilloscope which support frequency of up to 100MHz. The DSO5012A supports the use of USB and also includes a GPIB interface.

GPIB

General Purpose Interface Bus is a digital communication bus specification that was developed by Hewlett Packard (HP) which is used for connecting short range communication devices. The GPIB is also referred to as IEEE-488.

The GPIB cable consists of a 24 pin connector which is used for a double headed design. Both ends of the GPIB cable are used including the male on one side and the female on the other side. The cable consists of a 16 signal line of which eight lines are used for bi directional communication, five lines for the bus management while the remaining eight are used for the handshakes. Therefore, a single physical GPIB cable is capable of managing the communication of fifteen (15) devices connected to it.

The GPIB bus cable has several advantages over the traditional RS232 in quite a number of ways which include

- Performance
- Reliability
- Productivity

NI VISA Driver

A driver is a set of programs that enable communication between a computer system and a device connected to it through any of its interface. The drivers can be used to trigger series of events in the device connected to the computer system which includes configuration, reading from and writing to the device.

Virtual Instrumentation Software Architecture (VISA) provides the interactive platform and enables communication to devices independently.

The NI VISA driver is the National Instrument implementation of the VISA I/O standard. The NI VISA includes libraries, interactive utilities as well as configuration programs needed for device application development.

2.2 Quality Function Deployment

Quality function deployment refers to the translation of requirements and specifications into design. The quality function deployment in this case is described in three categories which includes normal requirements, expected requirements and exciting requirements. In the case of this project, the categories and their enlisted features are as follow

Normal requirements

- Control state and output of the calibrator
- Read the measured output from the oscilloscope
- Construct a table showing the calibrator output and oscilloscope measured values
- Determine if the oscilloscope requires adjustment

Expected requirements

- Checking the cable connection and notifying the user if the cable are properly connected
- Friendly graphic user interface
- Extend functionality and exceptions (flexible).

Exciting requirements

- Checking if the scope connection on the calibrator is well connected
- Ability to fully adjust and control the calibrator and oscilloscope.

3 APPLICATION DESCRIPTION

The main function of this application is its use in the calibration of measuring instruments in the laboratory. In order to carry out this activity, the application must be able to write values to the calibrator (fluke 5500A in this case) and read values measured from the oscilloscope (Agilent DSO5012A in this case).

As discussed earlier, the application has been programmed with the C# language using the .NET framework. Therefore, a driver is required to read and write to the connecting GPIB bus cable. The driver used for this application was the NI VISA driver by National Instruments. The driver contains library of classes and method responsible for communicating with data within the GPIB bus cable.

3.1 Application requirements

The requirements for this application have been categorized as ‘must have’, ‘should have’ and ‘nice to have’ requirements.

The must have requirements of this application are as follows

- Writing and adjusting the Calibrator: The application should be able to write values input by the user to the calibrator. In addition, the application should be able to change the output state of the calibrator for instance AC, DC, sine wave generation, square wave generation, tri wave generation.
- Reading and adjusting the oscilloscope: the application must be able to read the values measured by the oscilloscope. In addition, the application must be able to adjust the display view of the oscilloscope based on the user’s desire. For instance, the user must be able to adjust the vertical voltage division scale of the oscilloscope using the application (5V/div, 2V/div, 1V/div, 500mV/div, 200mV/div, 100mV/div, 50mV/div, 20mV/div, 10mV/div, 5mV/div and 2mV/div). Furthermore, the application must automatically adjust the horizontal time division scale of the oscilloscope based on the frequency generated by the calibrator.

- Calculate the minimum and maximum expected voltage: the application must be able to estimate the minimum and maximum voltage expected to be measured by the oscilloscope. This voltages are calculated based on the voltage value generated by the calibrator.
- Displaying data on a table: the application must be able to display all the data which includes the calibrator voltage output, calibrator frequency output, oscilloscope measured voltage, oscilloscope measured frequency, oscilloscope channel in use and the oscilloscope vertical division scale.

The should have requirements of the application includes

- Friendly user interface: the application should be friendly in usage to the user. Navigation, selecting of addresses, sending of values to the calibrator and adjusting of the oscilloscope should be easy for the user to carry out.
- Detection of cable and interface status: the application should be able to prompt the user if the GPIB cable or the USB interface are not properly connected.
- Provision of a data table for automatic calibration: the application should be able to provide the user with a table in which the user can enter series of values (voltage and frequency) that can be loaded into the calibrator to enhance the ease of the calibration process.

The nice to have requirements of the application include the following

- Saving calibration value to a file: for an easy future calibration process, it would be nice if the application can save the calibration parameters created by the user into a file. This is useful as the data can be used again for the calibration process thereby reducing the stress of the user creating new data table all the time during the calibration process.
- Loading data from a file: the application operation would be nice if it is possible for the user to load data directly to the calibrator from a file containing data that have been created and saved by the user.
- Saving displayed data as a Microsoft excel file or text file: the application operation would be nice if the application can give the user the option of saving the displayed data on the table as either a Microsoft excel file or text file. This feature

also depends on the existence of Microsoft excel on the PC. In addition, this feature help the analysis of the calibration process in future purposes.

3.2 Function description

The user installs the application on a PC with windows operating system running the .NET framework. Once the devices (calibrator and oscilloscope) are properly connected using the GPIB bus cable and the GPIB –USB connector is properly connected to the PC, the application becomes fully functional upon lunching.

The use case diagram below explains further the functionality of the application at different stages as well as the activities to be carried out by the user. The main stages in the use case diagram includes the start page, calibration page and the creating data table page.

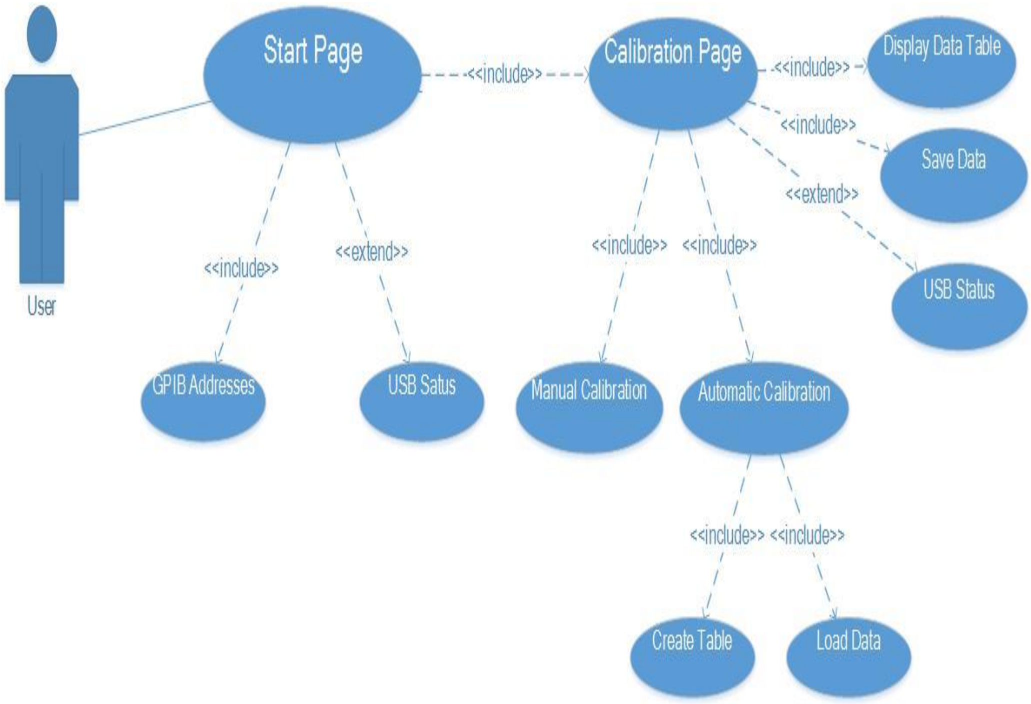


Figure 1: Use case diagram

Textual description of the use case diagram

Start Page

Use case name: Start Page

Participation actor: User

Entry Condition: This page is displayed to the user once the installation file of the application is lunched. In addition, the functionality of this page depends on the proper connection of the GPIB-USB connector to the GPIB bus cable, proper connection of the USB connector in the PC USB interface and powering on of the calibrator and the oscilloscope.

Event Flow: On this page, the user must give the correct GPIB address of the calibrator and the oscilloscope which is made available in the drop box provided in front of each device address.

Once the correct addresses have been selected for the right device, the user can proceed to the calibration page by clicking the connect button provided.

Error Message: an error message in this page can arise if there is an open connection in the GPIB bus cable or the PC USB interface.

Calibration page

Use case name: Calibration Page

Entry Condition: this page can only be accessed once all the criteria of the start page have been met and the connect button is clicked.

Event flow: on this page, the user is provided with an option of an automatic or manual calibration process. In the manual process, the user can send voltage and frequency values to the calibrator one at a time by clicking the output button provided in the application. Also, the user can adjust the vertical division scale of the oscilloscope.

On the other hand, in the automatic section, the user can either load data from an existing data file or create a new data table. Creation of a new table prompts the display of the ‘create table’ page which will be discussed later.

Finally on this page, the user is presented with the calibration result once the user clicks the measure button and the user also have the option of saving the results to a file for historical reference.

Error message: an error in this page can arise if the user tries to load data from a file that is not formatted for the application. Also, an error can also arise from disconnection of the GPIB bus cable as well as from USB – PC USB interface.

Create table page

Use case name: Create Table

Entry Condition: this page can only be accessed in the automatic mode of the calibration page.

Event Flow: Once in this page, the user is provided with a table which must be filled accordingly and correctly. The table is illustrated below:

Required Information	Description	Function
Voltage value	This is the voltage value sent to the calibrator. The value must be in consideration	Query Calibration process

	with the oscilloscope vertical voltage division scale displayed on the page.	
Frequency value	This is the frequency value sent to the calibrator.	Query Calibrator output state (DC and AC) Calibration process.
Channel Division	This is the range of vertical voltage division scale. This scale is predefined on this page. The application will automatically execute the calibration process by running each vertical voltage division scale with the corresponding voltage and frequency value entered.	Query Oscilloscope adjustment

Table 1: Table form information

Error message: error messages on this page can be prompted if the user enters a wrong data type in the voltage and frequency value. For instance, the application would flag a red color if any of the boxes is left empty and flag a blue color if any of the boxes contains a character.

3.3 Class description of the application

The classes of the application have been categorized based on the forms presented. In this application, the main entity is the main user.

Main User: this includes students, staffs and technician in the laboratory. The main user is responsible for plugging the GPIB cables to both the calibrator and the oscilloscope, connecting the USB to the PC USB interface and lunching the application from the installation file.

Upon lunching of the application, the main user is presented with the start page in which the user must select the correct calibrator GPIB port address and the oscilloscope GPIB port address from the drop box present on the page. Once this information have been properly entered, the user then click the connect button which proceeds with the display of the calibration page.

On the calibration page, the user can perform the calibration in the automatic or manual mode operation depending on the user's choice. In the automatic mode, the user have the opportunity of creating a new table for loading the data to be sent to the calibrator or loading the data from a file containing the data in the right format the application can recognize.

On the other hand, the manual mode presents the user with the voltage and frequency text box in which the user enters the values and sends them intermittently to the calibrator. On sending each value to the calibrator, the user also have the opportunity of adjusting the oscilloscope vertical voltage division scale. In case of an AC output, the application automatically calculates the horizontal time division scale of the oscilloscope to display the largest full period that can be obtained at the sent frequency.

Finally, the user is presented with a table showing the measured voltage and frequency value, calibrator outputted voltage and frequency values, minimum and maximum expected value of the oscilloscope and at the end, the status of the oscilloscope for each measurement.

Below is a figure showing the overall class diagram of the application

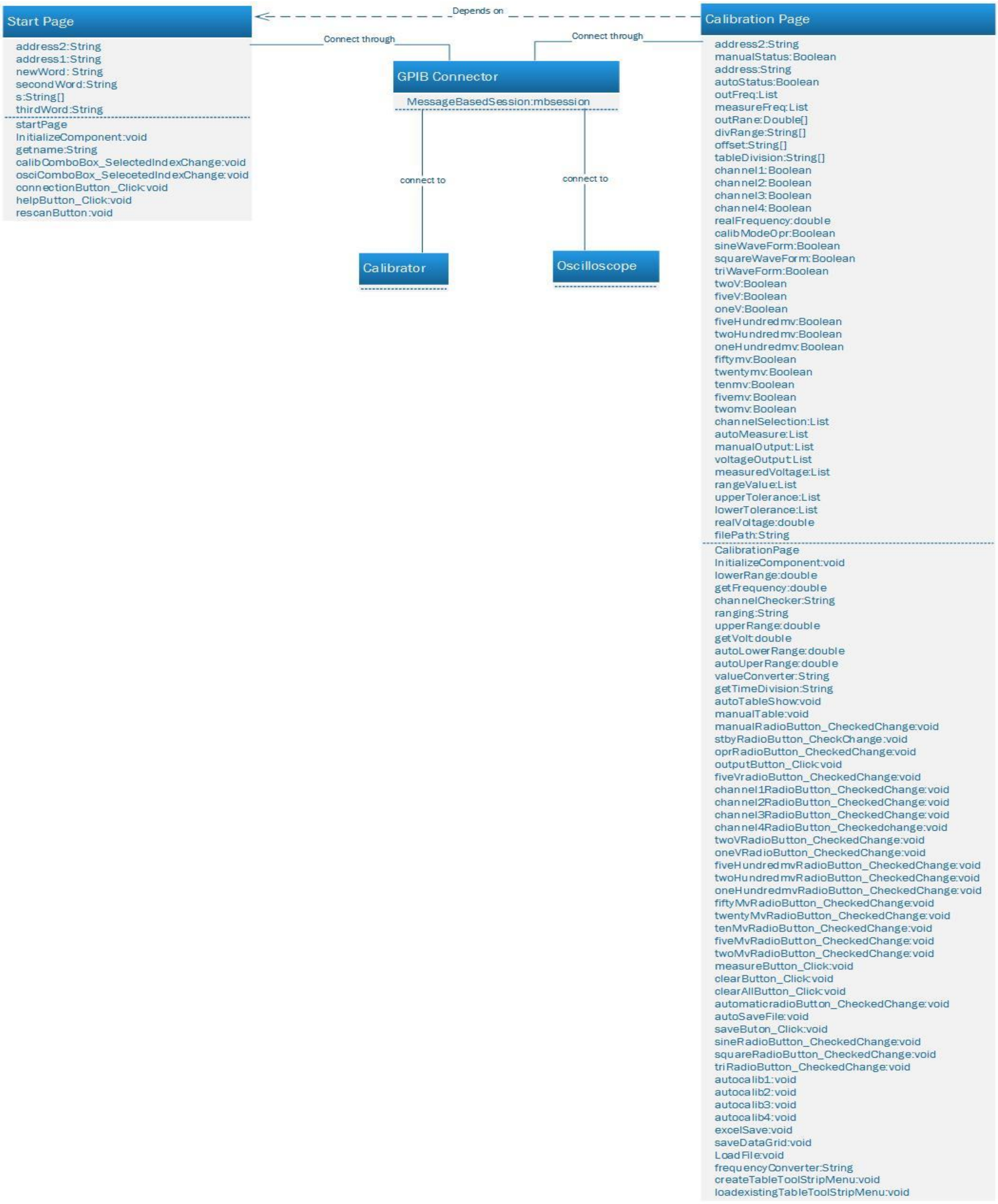


Figure 2: Class diagram

3.4 Individual class description

StartPage class

This class is responsible for identifying the GPIB addresses of the devices connected to the PC through GPIB. This class also allows selection of the calibrator address and oscilloscope address. In addition, this class displays the name of the device based on the selected address. Finally, the class provides a rescan handler which recheck if there are any new devices connected to the GPIB bus cable. This class contains several methods and event handlers.

Methods

- *getName()*: (argument: string, Return Type: String) this method is responsible for getting the name of the device once the address is selected. This method returns a string of the IDN of the devices.

Event handlers

- *calibComboBox_SelectedIndexChanged():* this event handler is responsible for selecting the address of the calibrator.
- *osciComboBox_SelectedIndexChanged()*: this event handler is responsible for selecting the address of the oscilloscope.
- *connectionButton_Click()*: this event handler is responsible for displaying the calibration page. This handler also ensures that the address are properly selected.
- *rescanButton_Click()*: this event handler is responsible for rescanning the address of the devices. This method returns an array of addresses which includes the address of the GPIB devices.

CalibrationPage class

This class is responsible for querying the GPIB devices. This class is also responsible for adjusting the oscilloscope. In addition, this class is responsible for sending values (voltage and frequency) to the calibrator. Furthermore, this class displays the result data on a table for analysis.

The class responsibility extends to saving of the content of the result data to a file, loading data vales from a file, clearing data on the table, selecting either automatic or manual calibration mode and displaying the table for creating new data values. Included in this class are several methods and event handlers.

Methods

- *lowerRange()*: (argument type: double, return type: double). This method is responsible for calculating the lowest voltage value expected from the oscilloscope based on the voltage sent from the calibrator. This method returns a double value which is the lowest value expected.
- *getFrequency()*: (argument type: double, return type: double).This method is responsible for checking the selected index of the frequency drop box and making sure that the frequency value is converted to the standard frequency value. This method returns the standard frequency.
- *frequencyConverter()*: (argument type: double, return type: string).This method takes in the frequency as an argument and converts them to a string and also returns them in a readable format e.g KHz, MHz.
- *channelChecker()*: (argument type: null, return type: string).This method is responsible for checking which of the channel radio button is selected and returns a string indicating the channel selected.
- *ranging():* (argument type: null, return type: string).This method checks which of the channel range radio button is selected and returns a string indicating the channel range selected.
- *upperRange()*: (argument type: double, return type: double). This method calculates the maximum voltage value expected to be measured by the oscilloscope and then returns the value as a double.
- *getVolt()*: (argument type: double, return type: double)This method takes the voltage value in the voltage text box and converts it to the standard unit. This method then returns the standard voltage based on the selected index.
- *valueConveter()*: (argument type: double, return type: string). This method is responsible for converting the voltage values to a format that is easy to understand e.g mV and V.

- *getTimeDivision()*: (argument type: double, return type: string). This method is responsible for calculating the vertical time division range of the oscilloscope by using the frequency value sent to the calibrator and converting it to an exponential format that is understandable by the oscilloscope.
- *manualTable()*: (argument type: List<double> , List<string> , List<string> c, List<double>, List<double>, return type: void). This method is responsible for displaying the table of the channel selected, range selected, frequency sent, frequency measured, voltage sent, voltage measured, maximum voltage expected and minimum voltage expected on the datagrid view.
- *excelSave()*: (argument type: null, return type: void). This method is responsible for writing the contents of the data grid view to an excel file.
- *saveDataGrid()*: (argument type: null, return type: void). This method is responsible for saving the content of the data grid view to a text file.
- *autocalib1()*:(argument type: null, return type: void). This method is responsible for writing to the calibrator and reading value from channel 1 of the oscilloscope in the automatic calibration mode.
- *autocalib2():*(argument type: null, return type: void). This method is responsible for writing to the calibrator and reading value from channel 2 of the oscilloscope in the automatic calibration mode.
- *autocalib3():* (argument type: null, return type: void). This method is responsible for writing to the calibrator and reading value from channel 3 of the oscilloscope in the automatic calibration mode.
- *autocalib4()*:(argument type: null, return type: void). This method is responsible for writing to the calibrator and reading value from channel 1 of the oscilloscope in the automatic calibration mode.
- *getChannelNo():* (argument type: string, return type: char): this method is responsible for determining the number of channel present on the oscilloscope.

Event handlers

- *manualRadioButton_CheckedChanged()*: This event handler is responsible for putting the program in manual mode once checked.

- *automaticRadioButton_CheckedChanged()*: this event handler is responsible for putting the program in the automatic calibration mode once checked.
- *stbyRadioButton_CheckedChanged()*: This event handler is responsible for putting the calibrator in standby mode once it is checked.
- *oprRadioButton_CheckedChanged()*: This event handler is responsible for putting the calibrator in operation mode once it is checked.
- *outputButton_Click()*: This event handler is responsible for sending the frequency and voltage values entered by the user to the calibrator. This event handler also checks and make sure the parameters entered by the user are in the correct format.
- *fiveVradioButton_CheckedChanged()*: This event handler is responsible for adjusting the horizontal range of the oscilloscope to the 5V/div once checked.
- *twoVRadioButton_CheckedChanged()*: This event handler is responsible for adjusting the horizontal range of the oscilloscope to 2V/div once checked.
- *oneVradioButton_CheckedChanged()*: This event handler is responsible for adjusting the horizontal range of the oscilloscope to 1V/div once checked.
- *fiveHundredmVRadioButton_CheckedChanged()*: This event handler is responsible for adjusting the horizontal range of the oscilloscope to 500mV/div once checked.
- *twoHundredMvRadioButton_CheckedChanged()*: This event handler is responsible for adjusting the horizontal range of the oscilloscope to 200mV/div once checked.
- *oneHundredmVRadioButton_CheckedChanged()*: This event handler is responsible for adjusting the horizontal range of the oscilloscope to 100mV/div once checked.
- *fiftyMvRadioButton_CheckedChanged()*: This event handler is responsible for adjusting the horizontal range of the oscilloscope to 50mV/div once checked.
- *twentyMvRadioButton_CheckedChanged()*: This event handler is responsible for adjusting the horizontal range of the oscilloscope to 20mV/div once checked.
- *tenmVRadioButton_CheckedChanged()*: This event handler is responsible for adjusting the horizontal range of the oscilloscope to 10mV/div once checked.

- *fivemVRadioButton_CheckedChanged()*: This event handler is responsible for adjusting the horizontal range of the oscilloscope to 5mV/div once checked.
- *twomVRadioButton_CheckedChanged()*: This event handler is responsible for adjusting the horizontal range of the oscilloscope to 2mV/div once checked.
- *channel1RadioButton_CheckedChanged()*: This event handler is responsible for ensuring that the channel one (1) of the oscilloscope is active.
- *channel2RadioButton_CheckedChanged()*:This event handler is responsible for ensuring that the channel two (2) of the oscilloscope is active.
- *channel3RadioButton_CheckedChanged():*This event handler is responsible for ensuring that the channel three (3) of the oscilloscope is active.
- *channel4RadioButton_CheckedChanged():*This event handler is responsible for ensuring that the channel four (4) of the oscilloscope is active.
- *measureButton_Click()*: This event handler button is responsible for sending query to the oscilloscope to acquire the voltage and frequency measured by the oscilloscope and also displays the data on the table.
- *clearButton_Click()*: This event handler button is responsible for clearing the selected row on the datagrid view.
- *clearAllButton_Click()*: This event handler is responsible for clearing all the data on the datagrid view once pushed.
- *sineRadioButton_CheckedChanged()*: This event handler is responsible for changing the wave form generated by the calibrator to a sine wave once checked. This event handler is responsive if the signal is an AC signal and not DC.
- *squareRadioButton_CheckedChanged()*:This event handler is responsible for changing the wave form generated by the calibrator to a square wave once checked. This event handler is responsive if the signal is an AC signal and not DC.
- *triRadioButton_CheckedChanged():*This event handler is responsible for changing the wave form generated by the calibrator to a triangle wave once checked. This event handler is responsive if the signal is an AC signal and not DC.
- *loadExistingTabletolStripMenuItem_Click():* this event handler is responsible for loading data to the calibrator from an existing file containing data in the format the application can recognize.

- *createNewTableToolStripMenuItem_Click():* this event handler is responsible for displaying the table page.

TableForm class

This class is responsible for handling the creation of a new table of data values that would be sent to the calibrator. The class is also responsible for saving the data created in the table to a file.

Methods

- *autoSave():* (argument: List<string> voltage, List<string> frequency, return type: void). This method is responsible for saving the voltage and frequency created in the table into a text file.

Event handlers

- *Loadbutton_Click()*: this event handler is responsible for loading the voltage and frequency created in the table to the calibrator.
- *Savebutton_Click()*: this event handler is responsible for saving the data in the table to a text file. The event handler carries out this activity by calling the autoSave() method.

3.5 Behavioral description

The application conforms to the UML (Unified Modelling Language). In this application, the sequence of event can be defined in various stages,

Selecting the address of the GPIB devices

- Connect the GPIB devices (calibrator and oscilloscope) to the PC with the devices running
- Lunch the application
- On the start page, select the appropriate address for the calibrator and the oscilloscope from the drop box.

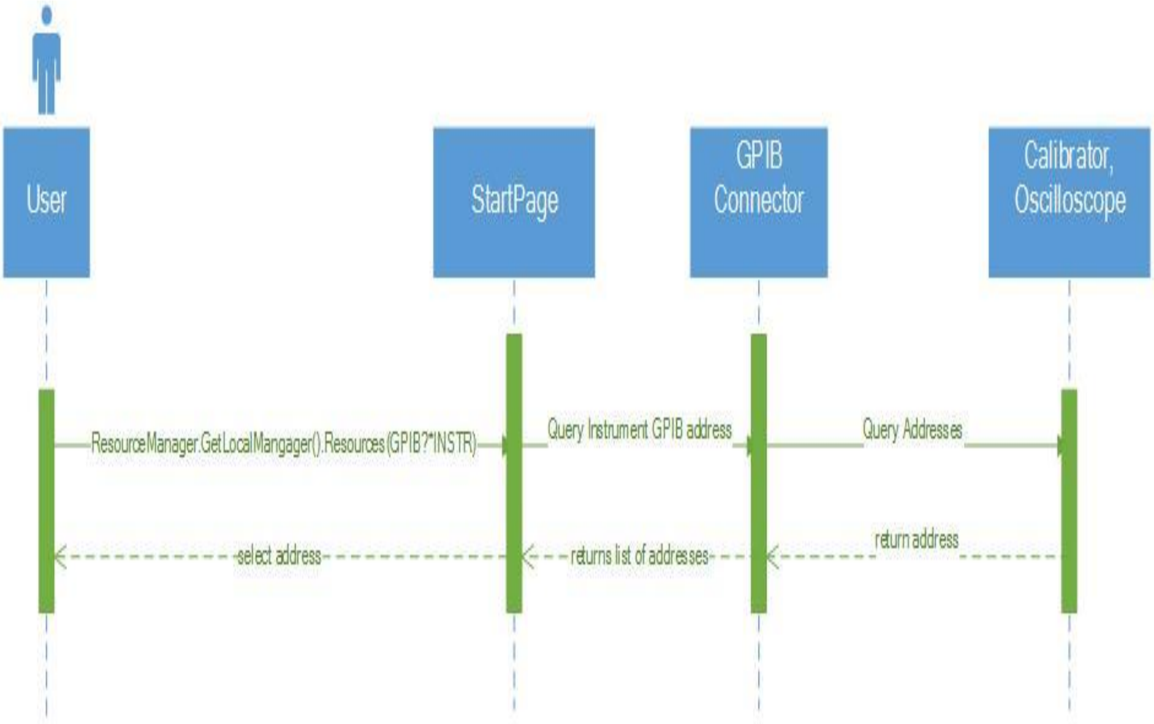

User
StartPage
GPIB Connector
Calibrator, Oscilloscope
ResourceManager.GetLocalMangager().Resources(GPIB?*INSTR)
Query Instrument GPIB address
Query Addresses
select address
returns list of addresses
return address

Figure 3: GPIB address selection

Getting the names of the devices

- Connect the GPIB devices (calibrator and oscilloscope) to the PC with the devices running
- Lunch the application
- The application sends a query to the device whose address has been selected to retrieve the IDN of the device using the *getName()* method on the start page.

Sending values manually to the calibrator

- Connect the GPIB devices (calibrator and oscilloscope) to the PC with the devices running
- Lunch the application

- On the start page, select the appropriate address for the calibrator and the oscilloscope from the drop box.
- Click the connect button
- On the calibration page, select the manual mode (this enable the section for outputting values to the calibrator and also enables the section for adjusting the oscilloscope).
- Enter the desired values in the voltage and frequency box.
- Click the output button. This make the application send the query of the values to the calibrator using the GPIB connector class
- In case the calibrator is sending an AC signal, the waveform can be adjusted by selecting any of the waveform radio button (sine, square and tri) by sending the query to adjust the wave form generated using the GPIB connector class.

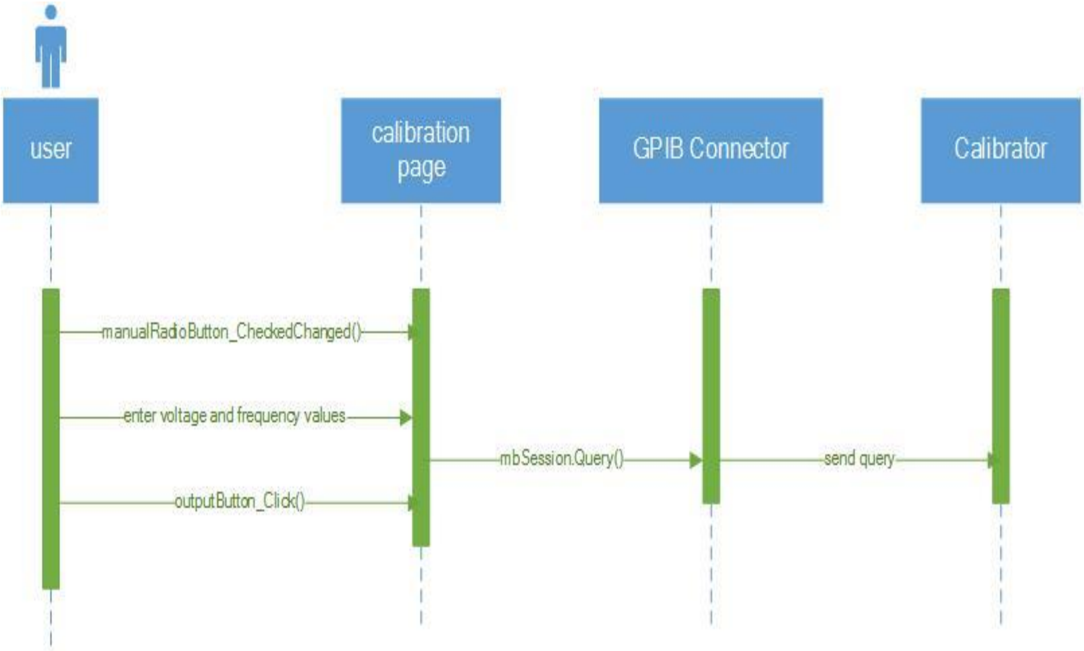


Figure 4: Sending queries to the calibrator

Adjusting the oscilloscope

- Connect the GPIB devices (calibrator and oscilloscope) to the PC with the devices running

- Launch the application
- On the start page, select the appropriate address for the calibrator and the oscilloscope from the drop box.
- Click the connect button
- On the calibration page, select the manual mode (this enable the section for outputting values to the calibrator and also enables the section for adjusting the oscilloscope).
- Select the appropriate channel
- Select the vertical voltage division scale desired from the channel range section of the application. The query is sent to the oscilloscope using the GPIB connector class.
- The application automatically adjust the horizontal time division of the oscilloscope in a situation where the signal is AC by making the time division five times the frequency sent from the calibrator using the *getTimeDivision()* method.

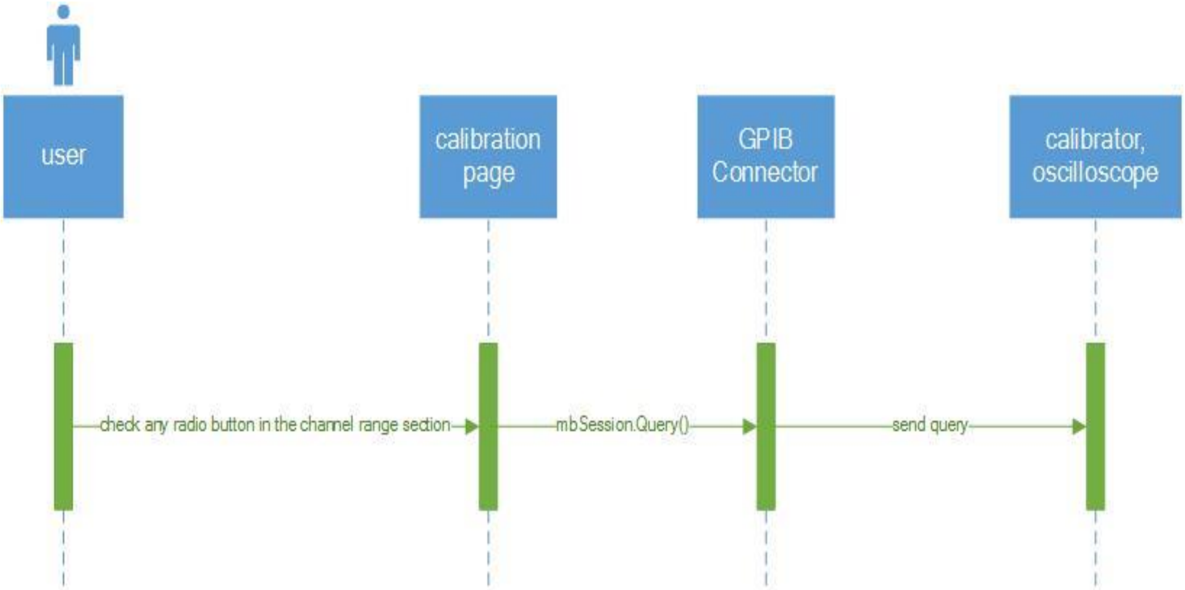


Figure 5: Adjusting the oscilloscope

Creating a new data table and loading

- Connect the GPIB devices (calibrator and oscilloscope) to the PC with the devices running
- Launch the application
- On the start page, select the appropriate address for the calibrator and the oscilloscope from the drop box.
- On the calibration page, the application is in the automatic calibration mode by default
- Click on the menu strip on the application showing the automatic section and select the create new table
- A table is presented. Fill the table correctly and click the load button
- The application returns to the calibration page. Select the appropriate channel for calibration and click the measure button. The calibration process starts by sending the data created into the calibrator and the calibration progress can be monitored on the progress bar showing on the application.

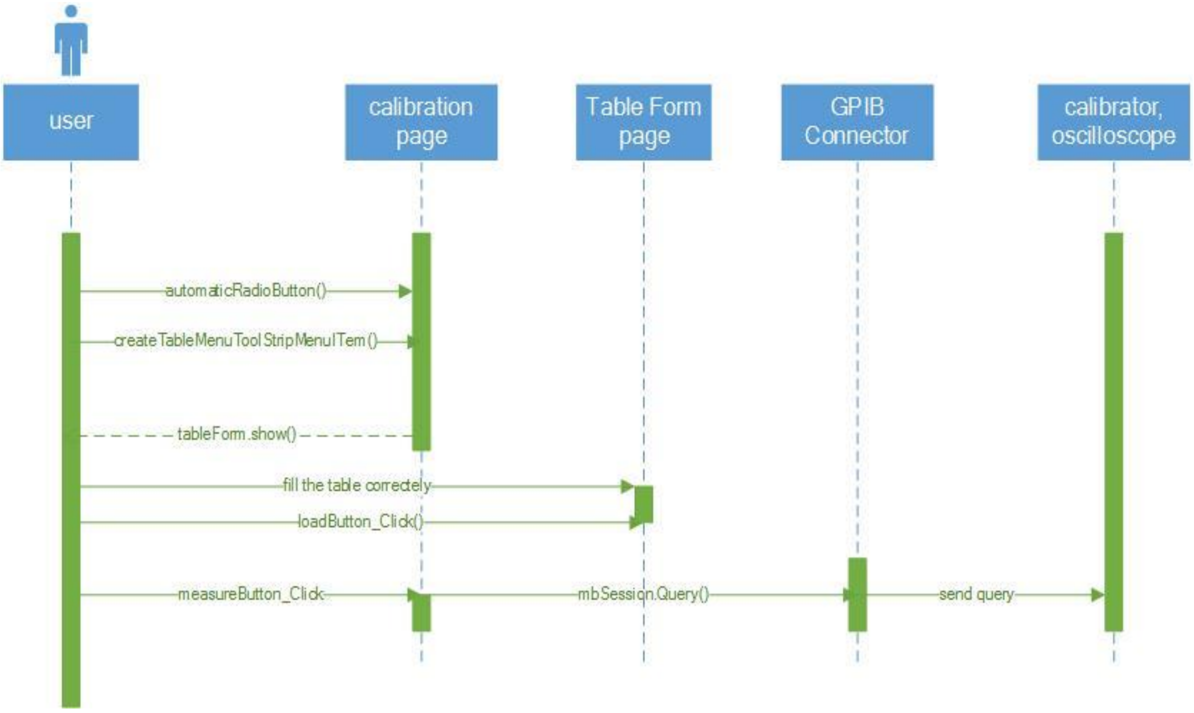


Figure 6: Creating a new data table

Saving the content of the data table to a file

- Connect the GPIB devices (calibrator and oscilloscope) to the PC with the devices running
- Launch the application
- On the start page, select the appropriate address for the calibrator and the oscilloscope from the drop box.
- On the calibration page, the application is in the automatic calibration mode by default
- Click on the menu strip section on the application which is only enable in the automatic mode and select the create new table
- A table is presented. Fill the table correctly and click on the save button
- The application presents a file dialog box where the desire file location can be selected.

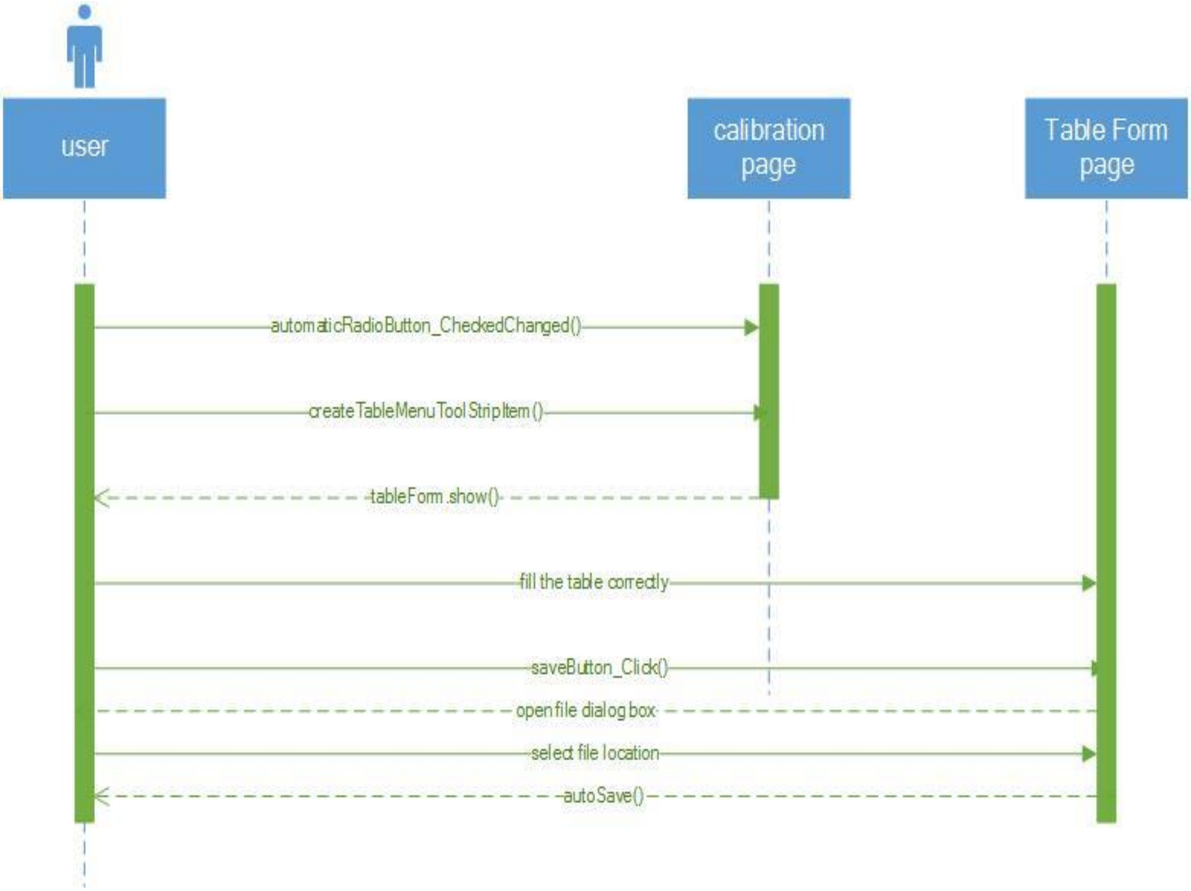


Figure 7: Saving content of the data table to a file

Loading data from an existing file

- Connect the GPIB devices (calibrator and oscilloscope) to the PC with the devices running
- Launch the application
- On the start page, select the appropriate address for the calibrator and the oscilloscope from the drop box.
- On the calibration page, the application is in the automatic calibration mode by default
- Click on the menu strip on the application which is only enabled in the automatic mode and select the load data from an existing file
- The application presents a file dialog box in which the user can select the file containing the data for calibration.
- Once the file have been selected, select the appropriate channel to be calibrated and push the measure button.
- The calibration process proceeds thereafter.

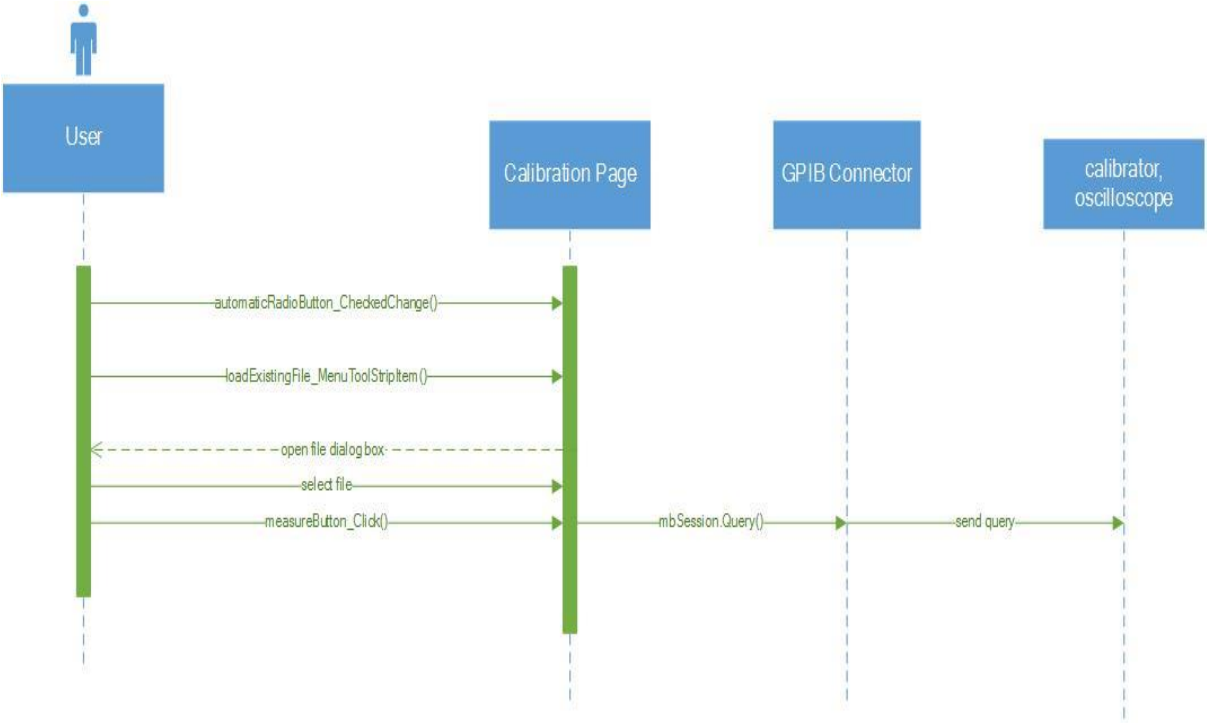


Figure 8: Loading data from an existing file

Clearing selected row from the displayed data on the data gird view

- Connect the GPIB devices (calibrator and oscilloscope) to the PC with the devices running
- Launch the application
- On the start page, select the appropriate address for the calibrator and the oscilloscope from the drop box.
- This feature is only available in the manual calibration mode.
- Once the data grid view has been populated with data, the user can delete an undesired row by clicking on the row and pushing the clear button.
- It is important to note that data deleted from the grid view cannot be recovered.

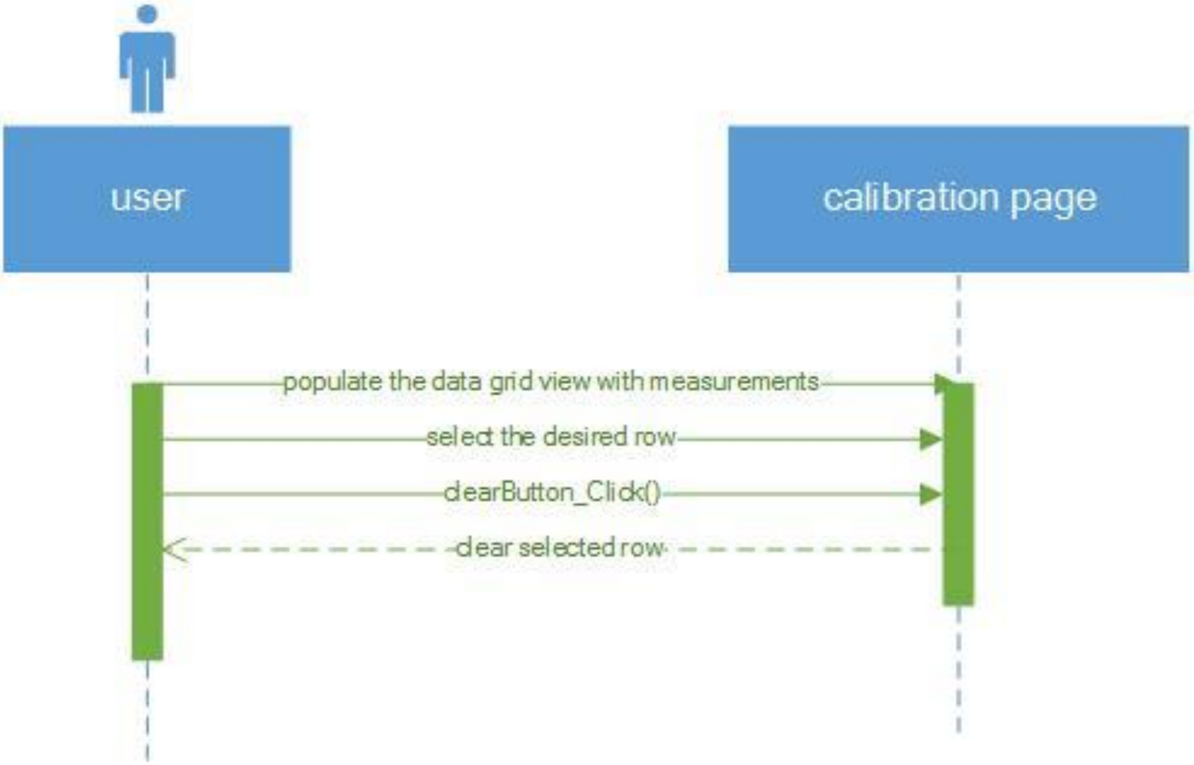


Figure 9: Clearing selected row from the data view

Clearing all data from the data grid view

- Connect the GPIB devices (calibrator and oscilloscope) to the PC with the devices running
- Launch the application
- On the start page, select the appropriate address for the calibrator and the oscilloscope from the drop box.
- This feature is available for both the automatic and manual calibration mode.
- Once the data grid view is populated and the information is not desired, push the clear all button
- It is important to note that data deleted from the grid view cannot be recovered.

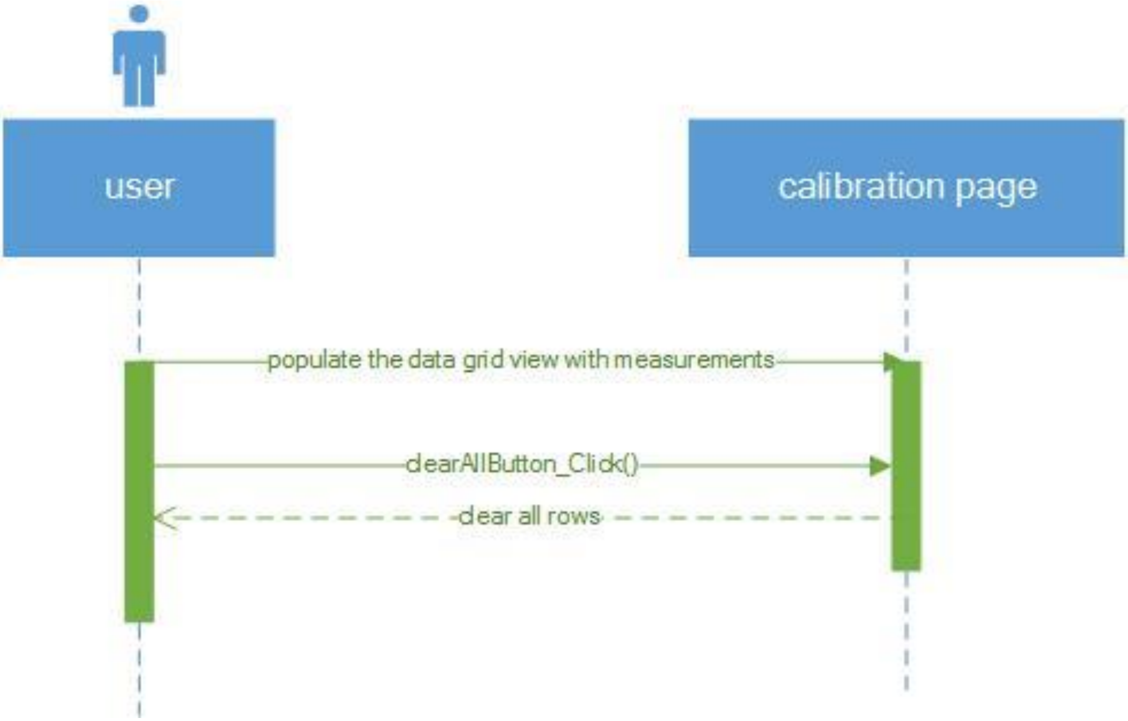


Figure 10: Clearing all rows from the data table

Saving data on the data grid view to as an excel document

- Connect the GPIB devices (calibrator and oscilloscope) to the PC with the devices running
- Launch the application
- On the start page, select the appropriate address for the calibrator and the oscilloscope from the drop box.
- This feature is available for both the automatic and manual calibration mode.
- Once the data gird view has been populated with the data, select the save button
- If Microsoft excel have been installed on the PC, a prompt message is presented to the user if the data should be save as an excel document or not
- To save the file as an excel document, the user should select yes
- This is proceeded with a file dialog box to specify the location for saving the file on the PC.

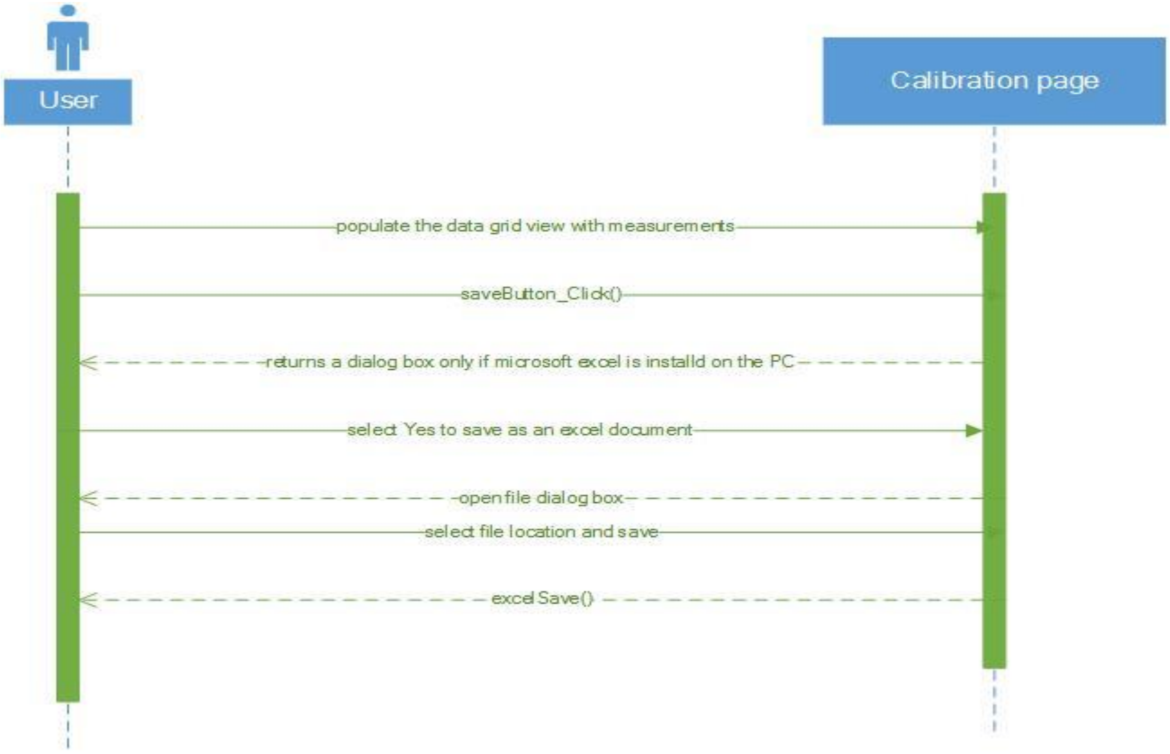

User
Calibration page
populate the data grid view with measurements
saveButton_Click()
returns a dialog box only if microsoft excel is installd on the PC
select Yes to save as an excel document
open file dialog box
select file location and save
excel Save()

Figure 11: Saving the data table content to as an excel file

Saving the data on the data grid view as a text file

- Connect the GPIB devices (calibrator and oscilloscope) to the PC with the devices running
- Launch the application
- On the start page, select the appropriate address for the calibrator and the oscilloscope from the drop box.
- This feature is available for both the automatic and manual calibration mode.
- Once the data gird view has been populated with the data, select the save button.
- Select the save button, if Microsoft excel is not installed on the PC, the application will automatically save the data as a text file.
- In case Microsoft excel is installed and the prompt message is presented, select no.
- A file dialog box is present to specify the location for saving the file.

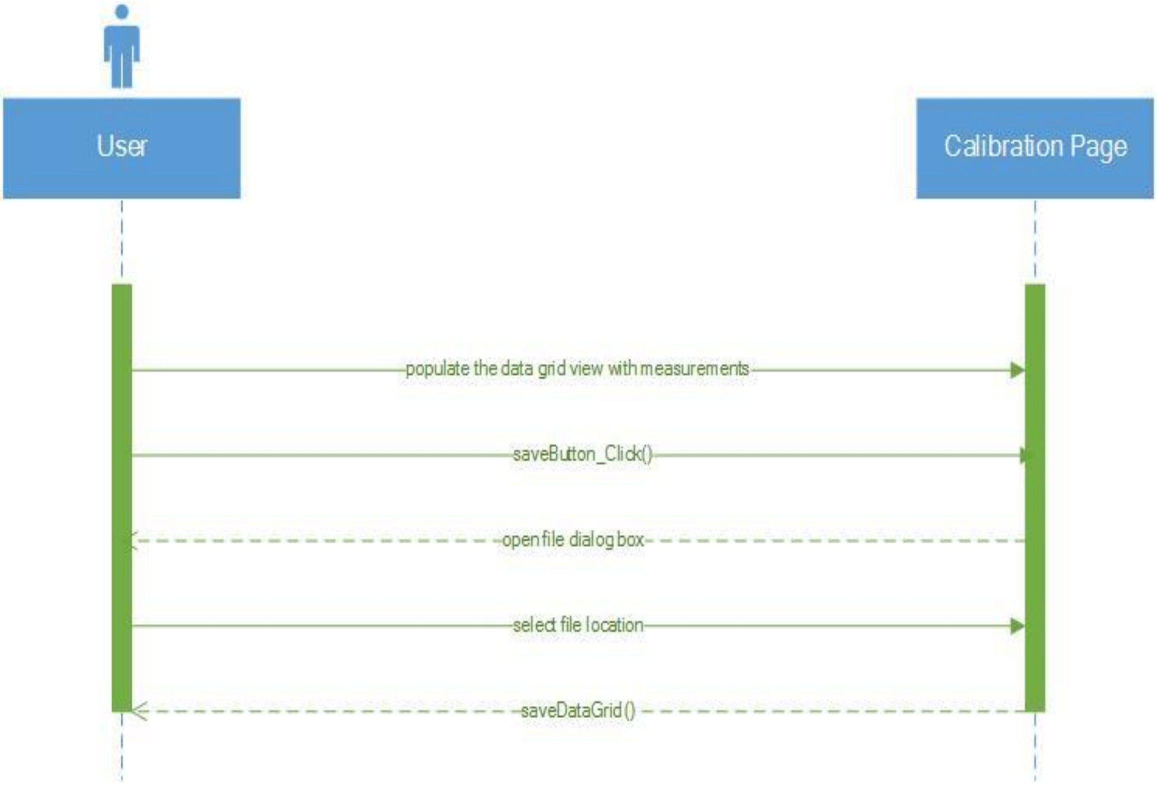


Figure 12: Saving the data table content as a text file

4 IMPLEMENTATION

The entire application was developed using the .NET framework. The GUI was implemented as a simple windows form which was designed using the Visual Studio 2012. The programing language used was C# for the programming. In this section, the implementation of the classes and method is discussed.

4.1 Communicating with the devices

The entire project was designed using the NI Visa driver provided by National Instrument. The driver contains classes and function which are relevant for the successful communication of the PC and the connected devices (calibrator and oscilloscope). One of the major class used during this project for communicating with the instrument is the 'MessageBasedSesion'. Therefore an instance of this class was created.

```
MessageBasedSession mbSession=(MessageBasedSession)ResourceManager.GetLocalManager().Open(adress);
```

Code Snippet 1 : Establishing a connection link between the PC and the

Once the connection is established. Queries are used to send and receive data from the devices. The instance of the class 'MessageBasedSesion' is used alongside the method 'Query' to send and receive information from the devices. The code below shows the querying of the device for the name of manufacturer.

```
string idnName = mbSession.Query("*IDN?");
```

Code Snippet 2: Querying the identity of the device

4.2 Start page

This is the first page presented to the user upon lunching of the program. The user is obliged to enter the required information which include selecting the calibrator address and oscilloscope address before clicking the connect button.

Getting connected devices GPIB addresses

This section involves obtaining a list of the GPIB address in an array and enrolling them in the drop box list of the calibration address and oscilloscope address. The list of GPIB devices connected to the PC is obtained using the resource name ‘GPIB?*INSTR’.

```
Public Calibration()
        {
            InitializeComponent();
            try
            {
                S = string[])ResourceManager.GetLocalManager().FindResources("GPIB?*INSTR");
                calibComboBox.Items.AddRange(s);
                osciComboBox.Items.AddRange(s);
            }
            catch (Exception)
            {
                //MessageBox.Show("please check the connecting cables and ensure the correct addresses are entered", "OK", MessageBoxButtons.OK, MessageBoxIcon.Information);
            }
}
```

Code Snippet 3: Retrieving the GPIB addresses

Retrieving the name of devices

This method gets the name of the device once the address has been selected. The method then loads the name as a label text. This helps the user to know what device have been selected.

```
  try
                {
                    mbSession = (MessageBasedSession)ResourceManager.GetLocalManager().Open(adress);// this is how the PC is connected to the device using the GPIB address
                    string idnName = mbSession.Query("*IDN?");
                    string[] word = idnName.Split(',');
                    theName = word[0];
                    theModel = word[1];
                   // MessageBox.Show(idnName, "OK", MessageBoxButtons.OK, MessageBoxIcon.Information);
                }
                catch (InvalidCastException)
                {
                    MessageBox.Show("please check address entered", "OK", MessageBoxButtons.OK, MessageBoxIcon.Information);
                }
                catch (Exception esp)
                {
                    MessageBox.Show(esp.Message.ToString()+"\nplease check the connecting cables and ensure the correct addresses are entered", "OK", MessageBoxButtons.OK, MessageBoxIcon.Information);
                }
                return theName + " " + theModel;
            }
        }
```

Code Snippet 4: Getting names of devices

4.3 Calibration page class

This class is responsible for the whole calibration process. In this class various methods are necessary for the successful executions in this application.

Scaling the time Division of the oscilloscope.

When sending an AC signal with a frequency, the horizontal time division of the oscilloscope needs to be adjusted to present the largest visible time period on the oscilloscope. In order to achieve this, the time division is scaled at a time equivalence of five times the frequency sent from the calibrator. The result is converted to an exponential format which is recognized by the oscilloscope and this value is then queried to the oscilloscope.

```
public string getTimeDivision(double frequency)
           {
                if (frequencycomboBox.SelectedIndex == 1)
{
                     frequency = frequency * 1000;
                }
                if (frequencycomboBox.SelectedIndex == 2)
{
                     frequency = frequency * 1000000;
                }
                double firstValue = frequency*5;
                double finalResult = 1/(double)firstValue;
                string value = finalResult.ToString("E3",
CultureInfo.InvariantCulture);
                return value;
         }
```

Code Snippet 5: Scaling the time division

Sending data to the calibrator.

Sending a signal from the calibrator to the oscilloscope using the scope outlet of the calibrator involves setting of the scope registry in the calibrator to ‘1’. This is made possible by sending a query ‘SETTLED’ to the calibrator. The signal from the calibrator can either be a DC signal or an AC signal. In sending an AC signal, the scope mode of the calibrator must be in ‘WAVEGEN’ while in sending a DC signal, the scope mode of the calibrator must be in ‘VOLT’.

```
mbSession = (MessageBasedSession)ResourceManager.GetLocalManager().Open(address);
mbSession.Write("SETTLED");

mbSession.Write("SCOPE WAVEGEN;");
mbSession.Write("OUT " + voltageTextBox.Text.ToString() + voltcomboBox.SelectedItem.ToString() + ", " +
frequencyTextBox.Text.ToString() + frequencycomboBox.SelectedItem.ToString());
mbSession = (MessageBasedSession)ResourceManager.GetLocalManager().Open(address2);
string value = getTimeDivision(realFrequency);
mbSession.Write(":TIMebase:SCALe " + value);
Thread.Sleep(milliseconds);
mbSession.Write("CHANnel2:OFFSet " +
value2.ToString("E2", CultureInfo.InvariantCulture));
```

Code Snippet 6: Sending an AC signal

If the signal is a DC, then the function is as below

```
mbSession = (MessageBasedSession)ResourceManager.GetLocalManager().Open(address);

mbSession.Write("SETTLED");

mbSession.Write("SCOPE VOLT; OUT " + voltageTextBox.Text.ToString() +
voltcomboBox.SelectedItem.ToString() + ", 0Hz");
```

Code Snippet 7: Sending DC signal

Displaying data to the data grid view

This function is responsible for displaying the data on the data grid view. In this function, a class of data table is instantiated. The columns of the table is loaded with its content as well as its row.

```
dataGridView1.DataSource = null;
status.Clear();
tb = new DataTable();
tb.Columns.Add("CHANNEL");
tb.Columns.Add("RANGE");
tb.Columns.Add("OUT FREQ");
tb.Columns.Add("MEASURED FREQ");
tb.Columns.Add("OUT VOLTAGE");
tb.Columns.Add("MEASURED");
tb.Columns.Add("MAXIMUM");
tb.Columns.Add("MINIMUM");
tb.Columns.Add("STATUS");

// here we add the values of the rows to the table


tb.Rows.Add(c[i], g[i], frequencyConverter(f1[i].ToString()), frequencyConverter(f2[i]),
voltageOutput[i] +
voltcomboBox.SelectedItem.ToString(), valueConverter(r[i]), valueConverter(upperTolerance[i]),
valueConverter(lowerTolerance[i]), status[i]);
```

Code Snippet 8: Loading the data table with data

The whole table is then set as the data source of the data grid view.

```
dataGridView1.DataSource = tb;
dataGridView1.AutoResizeColumns();
dataGridView1.Show();
```

Code Snippet 9: Loading the datagridview with the data table content

Adjusting of the vertical voltage scale of the oscilloscope.

The display view of the oscilloscope is sectioned into eight grid view. Therefore, the vertical scaling of the voltage must be adjusted in order to display the voltage on the screen. In doing this, appropriate voltage division scale is selected for the voltage. The oscilloscope consist of eleven voltage division which includes 5V/div, 2V/div, 1V/div, 500mV/div, 200mV/div, 100mV/div, 50mV/div, 20mV/div, 10mV/div, 5mV/div and 2mV/div. In this project, once the signal is a DC signal, the oscilloscope offset is set to four (4) times the value of the vertical voltage scale while in the case of sending an AC signal, the offset is set to zero (0). The value queried to the oscilloscope are usually in the exponential format as shown below

```
string[] divRange = new string[] {"+5.00E+00",
"+2.00E+00", "+1.00E+00", "+500E-03", "+200E-03",
"+100E-03", "+50E-03", "+20E-03", "+10E-03", "+5E-
03", "+2E-03"}; // channel range for the oscillo-
scope in NR3 format
```

Code Snippet 10: Vertical voltage division format

```
string[] offset = new string[] { "+20.0000E+00",
"+8.00000E+00", "+4.00000E+00", "2.00000E+00",
"800.000E-03", "400.000E-03", "200.000E-03",
"80.000E-03", "40.000E-03", "20.0000E-03", "8.0000E-
03" }; // channel offset for the oscilloscope in NR3
format
```

Code Snippet 11: Offset value format

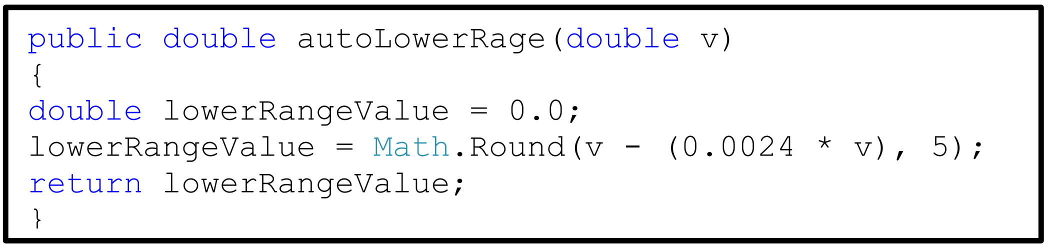
The following codes show how the vertical voltage division of the oscilloscope is adjusted to 1V/div.

```
mbSession = (MessageBasedSession)ResourceManager.Get-
LocalManager().Open(address2);
mbSession.Write("CHANnel1:SCALe " + divRange[2]);
Thread.Sleep(milliseconds);
if (realFrequency > 0)
{

String value = getTimeDivision(realFrequency);

mbSession.Write(":TIMebase:SCALe " + value);
Thread.Sleep(milliseconds);
mbSession.Write("CHANnel1:OFFSet " +
value2.ToString("E2", CultureInfo.InvariantCulture));
}
else
{
mbSession.Write("CHANnel1:OFFSet " + offset[2]);
```

Code Snippet 12: Adjusting the oscilloscope of 1V/div

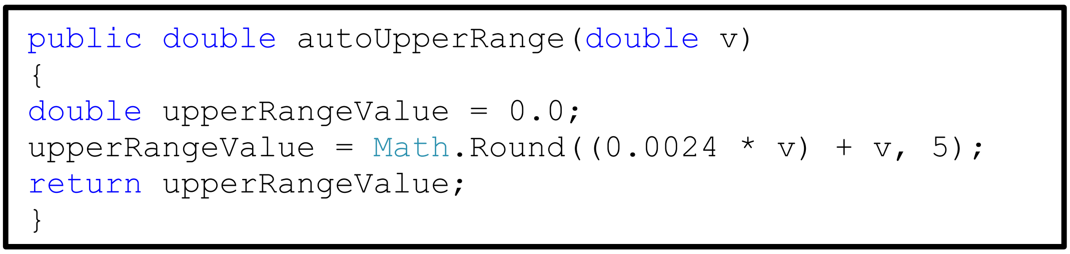
Calculation of the minimum and maximum expected voltage.

The application calculates the minimum and maximum voltage expected to be measured by the oscilloscope. In this application, these values are only applicable to the DC signals generated from the calibrator and not AC signal. For this application the minimum value is 0.24% less than the voltage sent from the calibrator while the maximum voltage is 0.24% more that the voltage sent from the calibrator. The method responsible for this process are shown below.



```
public double autoLowerRage(double v)
{
double lowerRangeValue = 0.0;
lowerRangeValue = Math.Round(v - (0.0024 * v), 5);
return lowerRangeValue;
}
```

Figure 13: Calculation of the minimum expected voltage



```
public double autoUpperRange(double v)
{
double upperRangeValue = 0.0;
upperRangeValue = Math.Round((0.0024 * v) + v, 5);
return upperRangeValue;
}
```

Figure 14: Calculation of the maximum expected voltage

5 TESTING

The entire application of the project was developed for the Technobothia laboratory. The testing process was therefore carried out at the laboratory in order to certify the functionality of the application. The testing was carried out in various stages.

5.1 Startup page

This is the first page presented to the user on launching of the application. On this page, the user is requested to select the calibrator and oscilloscope address correctly before proceeding to the calibration process. It is important to note that this page also requires correct setting up of the instruments.

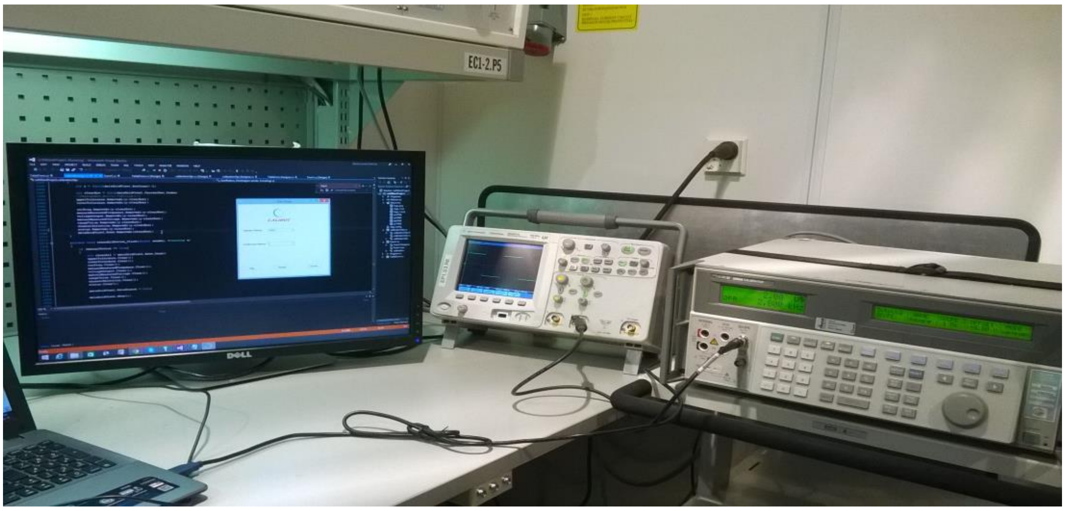


Figure 15: Instrument setup

Once the application installation file is opened, the startPage class is invoked and the figure below is presented

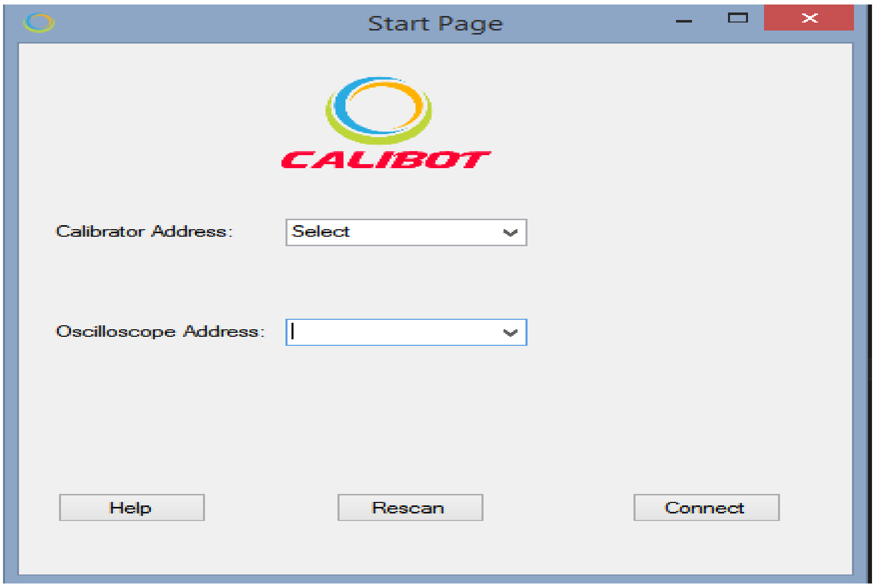


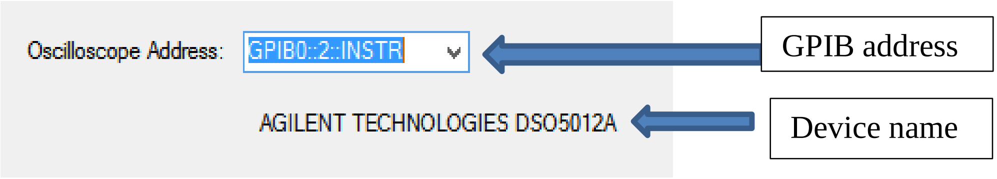
Figure 16: Start page of the application

On this page, once the address are selected from the drop down list, the name of the device with the address is displayed using the *getName()* method of the startPage class by sending '*IDN?' query to the device whose address have been selected. This makes it easy for the user to know which instrument has been selected.

```
try
{
mbSession = (MessageBasedSession)ResourceManager.GetLocalManager().Open(adress);// this is how the PC is connected to the device using the GPIB address
string idnName = mbSession.Query("*IDN?");
```

Code Snippet 13: Method for getting the name of the device

The result of this method is shown below



Oscilloscope Address: GPIB0::2::INSTR

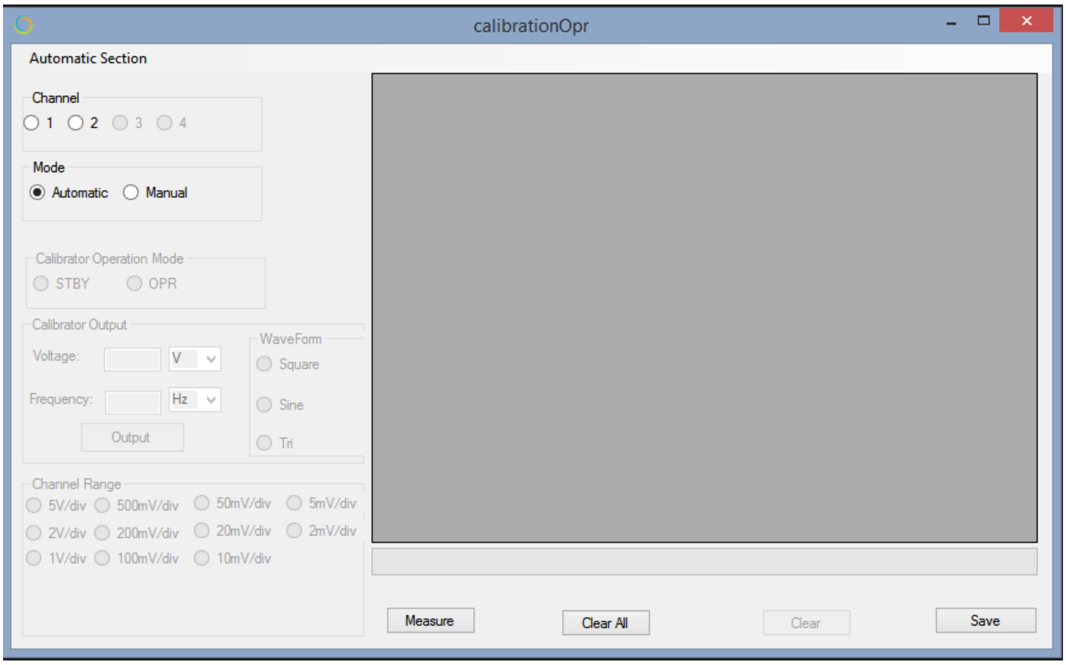
AGILENT TECHNOLOGIES DSO5012A

Figure 17: GPIB address and device name display

This page is concluded by clicking the connect button which will result in the display of the calibration page.

5.2 Calibration page

This is the page presented to the user upon completion of the start page. The page is represented in the diagram below



Mode: ● Automatic ○ Manual

Figure 18: calibration page

On this page, the user is able to perform the calibration either automatically or manually. In the automatic calibration process, the user can either create a new table of data or load the data values from a file containing the data to be sent. The application then sends this values to the calibrator and at the same time obtain the corresponding measurement by the oscilloscope.

On the other hand, the manual calibration mode involves the user sending data from the application to the calibrator and taking the measurements one at a time.

It is important to note that during the calibration process, when the frequency sent is changed, the application ensures that the oscilloscope is adjusted to display the largest period that can be obtained by that frequency.

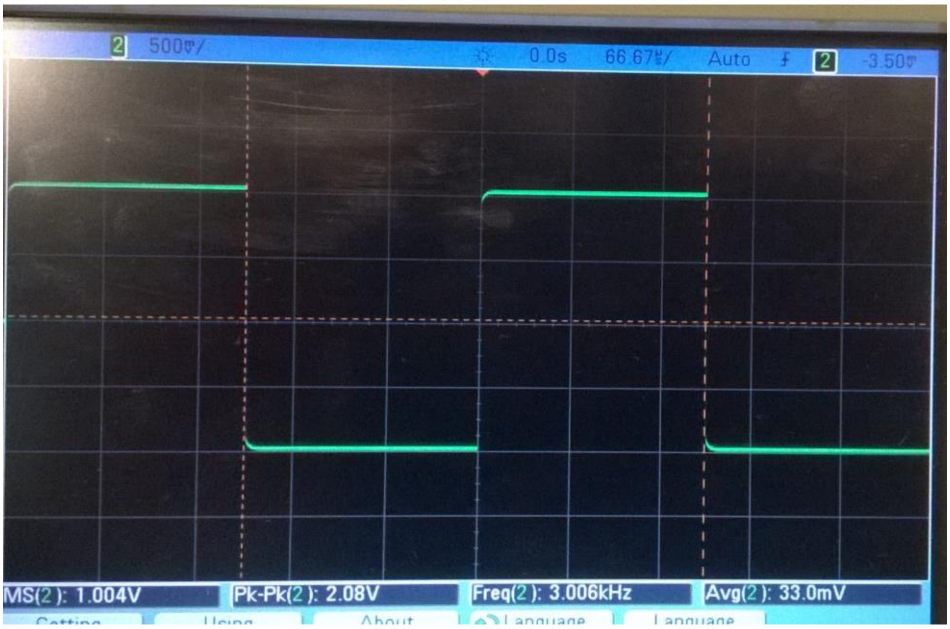
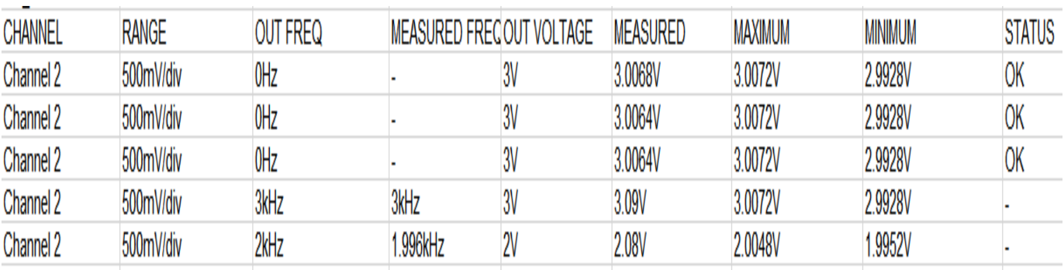


Figure 19: largest period obtainable with 3kHz

Furthermore, the result of the calibration process is displayed on the page which includes the channel which is being calibrated, frequency and voltage sent from the calibrator, frequency and voltage measured by the oscilloscope, minimum and maximum voltage values expected to be measured by the oscilloscope and the status of the oscilloscope based on the measurement obtained from the oscilloscope. In the result displayed, it is important to note that the application do not measure the frequency from the oscilloscope when a DC signal is generated by the calibrator and also the status is not checked when an AC signal is generated.



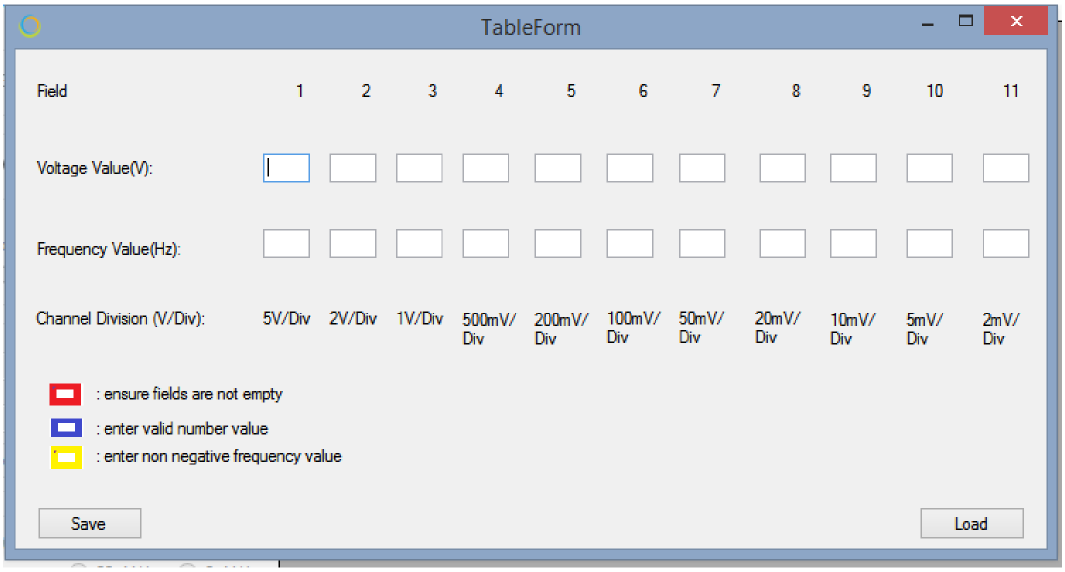
CHANNEL	RANGE	OUT FREQ	MEASURED FREQ	OUT VOLTAGE	MEASURED	MAXIMUM	MINIMUM	STATUS
Channel 2	500mV/div	0Hz	-	3V	3.0068V	3.0072V	2.9928V	OK
Channel 2	500mV/div	0Hz	-	3V	3.0064V	3.0072V	2.9928V	OK
Channel 2	500mV/div	0Hz	-	3V	3.0064V	3.0072V	2.9928V	OK
Channel 2	500mV/div	3kHz	3kHz	3V	3.09V	3.0072V	2.9928V	-
Channel 2	500mV/div	2kHz	1.996kHz	2V	2.08V	2.0048V	1.9952V	-

Figure 20: Result of calibration process

Finally, the result of the calibration process is being able to be saved to either a text file or as an excel document if Microsoft excel is installed on the PC used.

5.3 Table form page

This page is accessible when the user is in the automatic calibration mode of the calibration process. The page is shown in the figure below



Field	1	2	3	4	5	6	7	8	9	10	11
Voltage Value(V):											
Frequency Value(Hz):											
Channel Division (V/Div):	5V/Div	2V/Div	1V/Div	500mV/ Div	200mV/ Div	100mV/ Div	50mV/ Div	20mV/ Div	10mV/ Div	5mV/ Div	2mV/ Div

🟥 : ensure fields are not empty

🟦 : enter valid number value

🟨 : enter non negative frequency value

Figure 21: Data table form

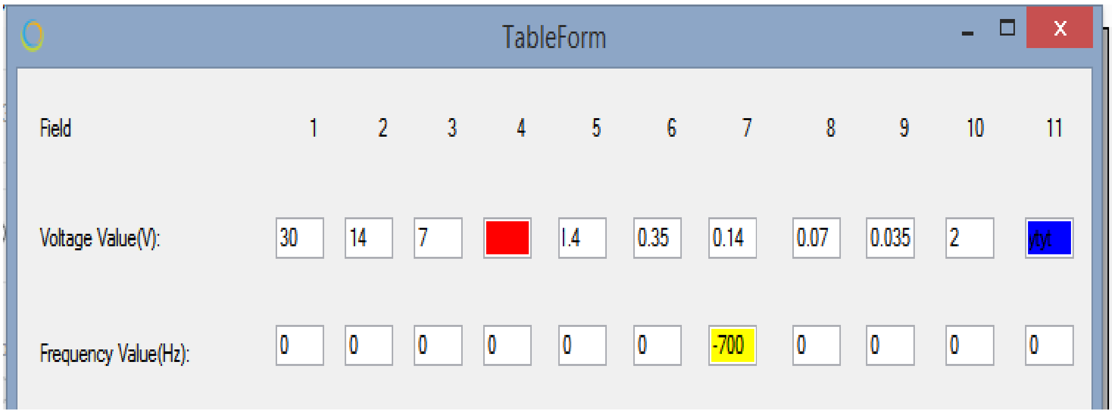
On this page, the user has the privilege of filling the boxes and loading the data directly to the calibrator or filling the table and saving its content to a file for future use.

In addition, this page has been designed to execute the calibration process by running the voltage and frequency value across different vertical voltage divisions of the oscilloscope. Therefore, it is important to note that filling the voltage value in the box requires consideration of the vertical voltage division of the oscilloscope.

Finally, the table form page is designed to notify the user in case of an error while filling the form. Errors in this case includes

- Empty boxes
- Negative frequency value
- Entering character

All this errors stated above are notified to the user with a color indicator. Empty boxes are indicated with a red color, invalid characters are indicated with a blue color and negative frequency value are indicated with a yellow color. The figure below demonstrates the error handling of the table form page.



Field	1	2	3	4	5	6	7	8	9	10	11
Voltage Value(V):	30	14	7	🟥	1.4	0.35	0.14	0.07	0.035	2	ytyt
Frequency Value(Hz):	0	0	0	0	0	0	-700	0	0	0	0

Figure 22: Error notification

6 USER MANUAL

6.1 System overview

The project application is designed to help the automatic calibration of measuring instrument in the laboratory. The application allows the user to send values (voltage and frequency) to the calibrator and also allows the user to obtain the value measured by the oscilloscope to be displayed on a screen. In addition, the application calculates a minimum and maximum voltage value expected of the oscilloscope based on the voltage value sent from the calibrator. Finally, the application allows the user to save the obtained data to a file. The application operates on computers with Microsoft windows operating system.

6.2 System configuration

The application operates on computers with windows operating systems. It is compatible with windows 7 and higher version with the .Net framework. The application requires a USB connection through the GBIP – USB connector to the calibrator and oscilloscope in order to write values to the calibrator and read values from the oscilloscope. Data from the application can be saved either as an excel document if Microsoft excel is installed on the computer or saved as a text file. The application can be used immediately without any further configuration.

6.3 User access levels

Everyone with the application installer can use the application.

6.4 Contingencies

In case of power outage, data are not saved in internal memory of the operation device therefore data not save to a file are lost. In case of the USB connection loss, the application is unable to communicate with the calibrator and oscilloscope.

Finally, in case wrong addresses are selected for the devices then wrong queries are sent which will result in an error message.

6.5 Installation and connection

The application is bundled as one executable file. The executable file must run on a Microsoft Windows compatible computer.

To install, run and access the application executable file.

1. Download the setup.exe file

The executable is made available by the developer to the client through a USB stick.

2. Save the file in the desired location and install

6.6 Using the application

System setup

The setup should be as below

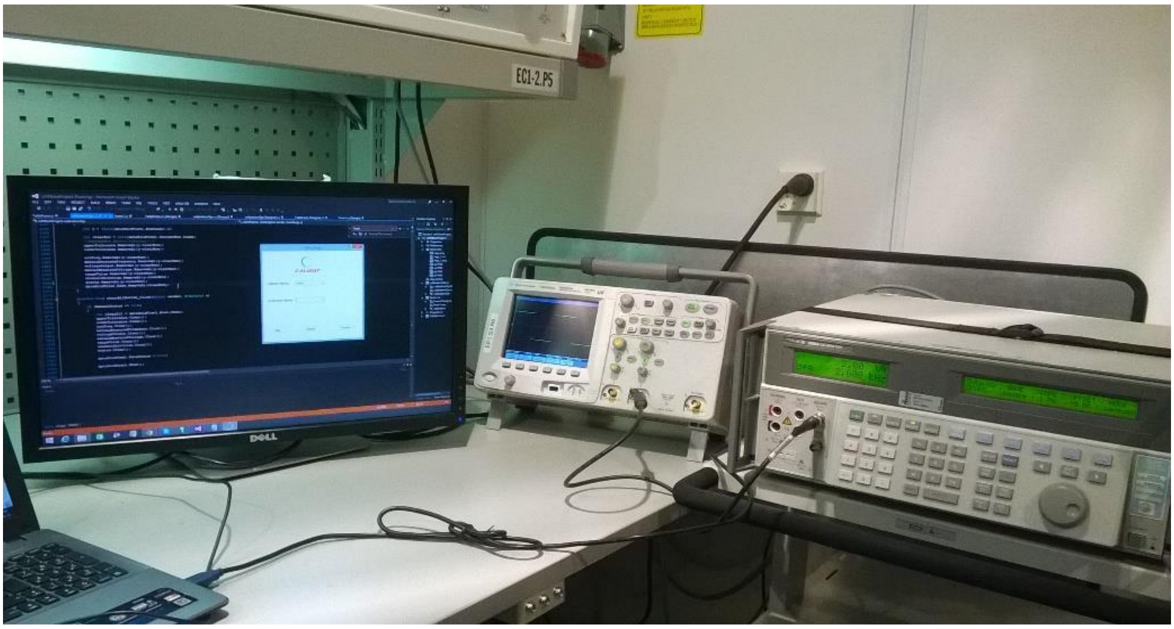
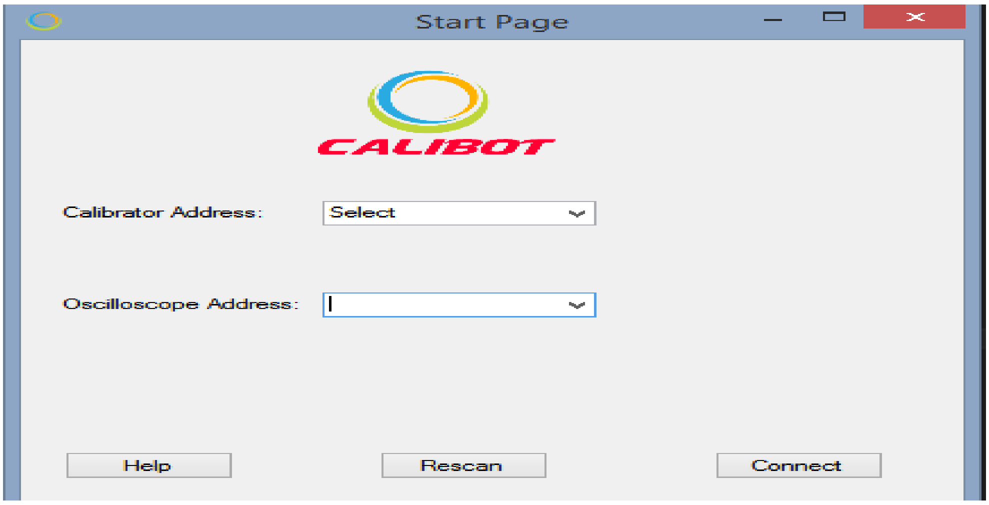


Figure 23: System setup

Once the application is installed, the application can be opened by double clicking on the software Project file.

The corresponding start page will be displayed



Calibrator Address: Select

Oscilloscope Address:

Figure 24: Start page

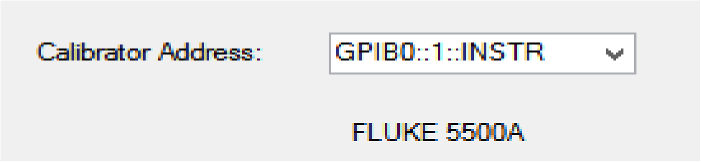
Start page.

- **Selecting the calibrator address**

1. Click the drop down box in front of the calibrator address to get a list of GPIB addresses of devices connected through GPIB cable.

Note: in case the list is empty, then check the USB cable and click the rescan button

2. Once the address is selected, the name of the device selected is displayed on the screen.



Calibrator Address: GPIB0::1::INSTR

FLUKE 5500A

Figure 25: Calibrator address

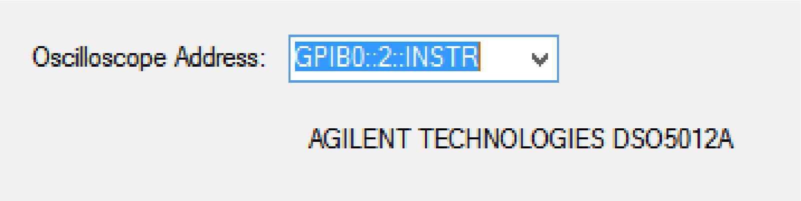
Note: ensure that the calibrator address is rightly selected as this can affect the whole performance of the application.

- **Selecting the oscilloscope address.**

1. Click on the drop box in front of the oscilloscope address to get a list of GPBI addresses of devices connected through the GPIB cable.

Note: in case the list is empty, then check the USB cable and push the rescan button

2. Once the address is selected, the name of the device selected is displayed on the screen



Oscilloscope Address: GPIB0::2::INSTR

AGILENT TECHNOLOGIES DSO5012A

Figure 26: Oscilloscope address

Note: ensure that the oscilloscope address is rightly selected as this can affect the whole performance of the application.

Calibration page

The calibration page is loaded once the connection button is clicked from the start page.

Note: The connection button should only be clicked when the calibrator address and oscilloscope address have been properly entered. Clicking the connect button will display the calibration page. The calibration page is as below

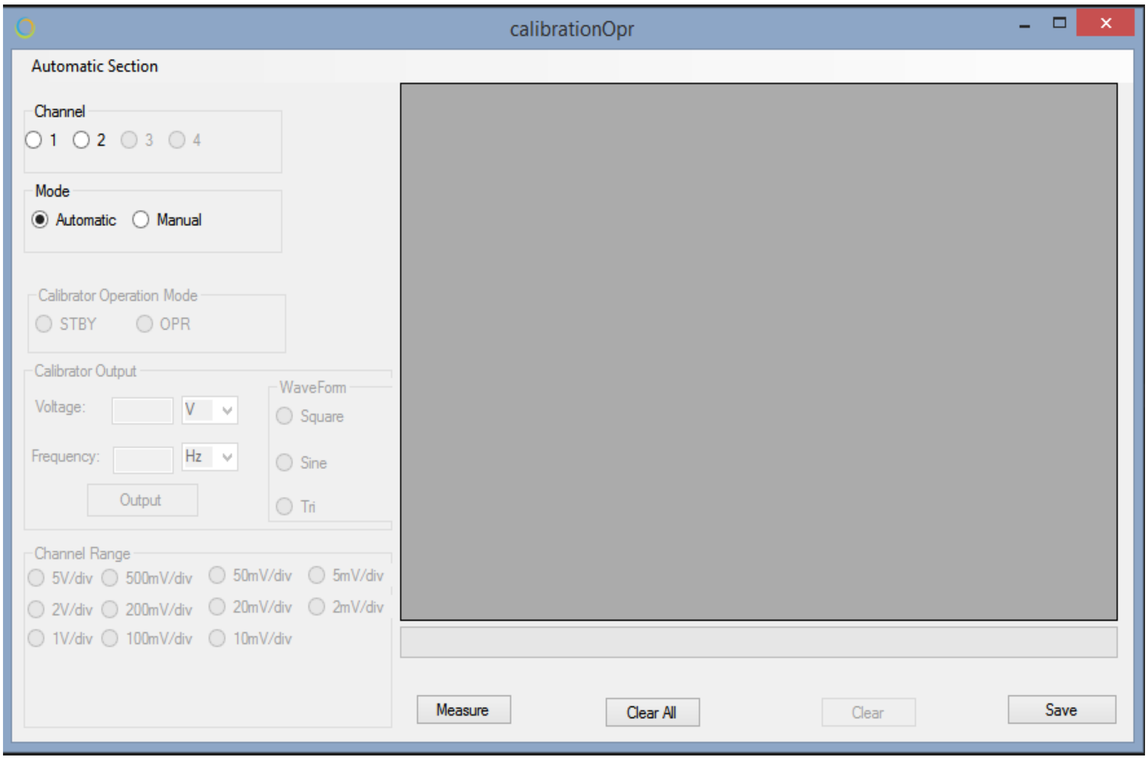


Figure 27: Calibration page

Automatic mode

- **Creating a new Data Table**

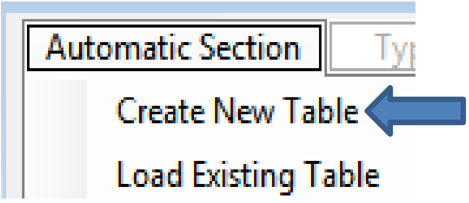
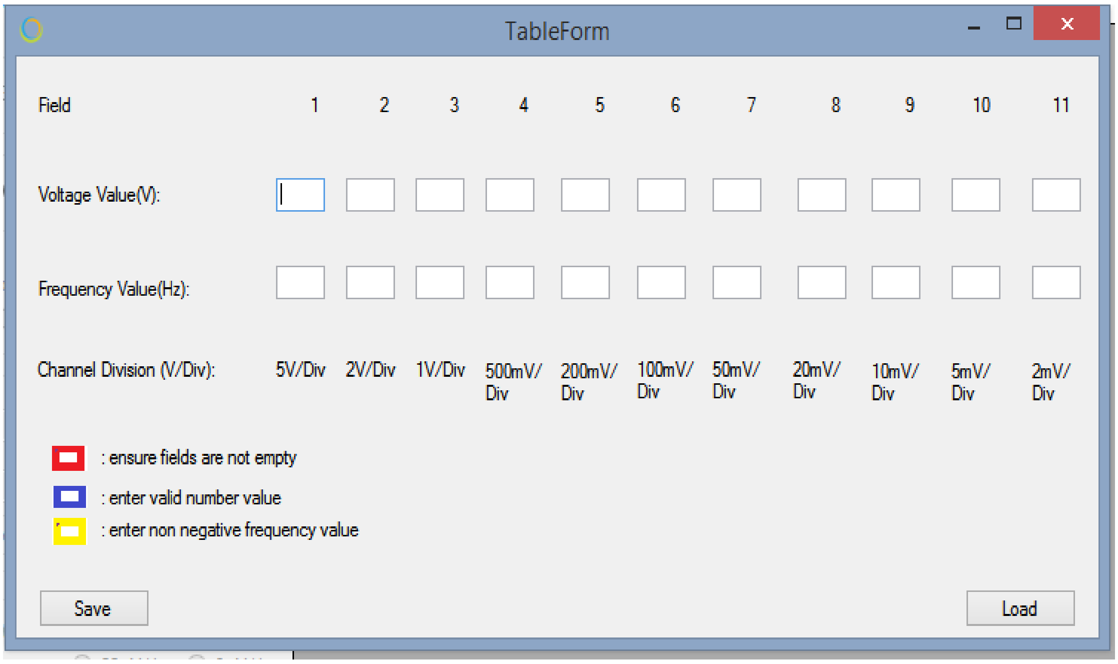


Figure 28: Selecting the 'create new table'

1. Clicking the "create new table" will prompt the application to display the corresponding table.



Field	1	2	3	4	5	6	7	8	9	10	11
Voltage Value(V):											
Frequency Value(Hz):											
Channel Division (V/Div):	5V/Div	2V/Div	1V/Div	500mV/ Div	200mV/ Div	100mV/ Div	50mV/ Div	20mV/ Div	10mV/ Div	5mV/ Div	2mV/ Div

🟥 : ensure fields are not empty

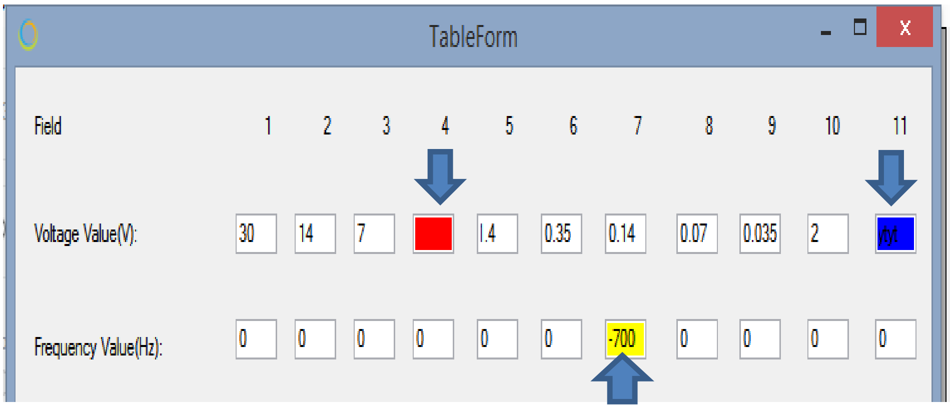
🟦 : enter valid number value

🟨 : enter non negative frequency value

Figure 29: Create new table

2. The user must fill the form properly by entering valid values for the voltage as well as the frequency value for the various vertical voltage division scale of the oscilloscope as specified on the table.

Note: the application will return the table if an error is made such as an empty box (indicated with a red color), a character (indicated with a blue color) and a negative frequency value (indicated with a yellow color).

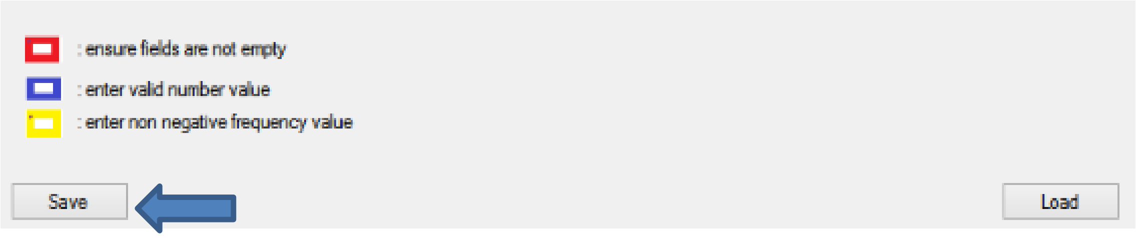


Field	1	2	3	4	5	6	7	8	9	10	11
Voltage Value(V):	30	14	7	🟥	1.4	0.35	0.14	0.07	0.035	2	ytyt
Frequency Value(Hz):	0	0	0	0	0	0	-700	0	0	0	0

Figure 30: Create new table error display

The user can either load and run the values in the table directly or save to a file.

- **Save button**



🟥 : ensure fields are not empty

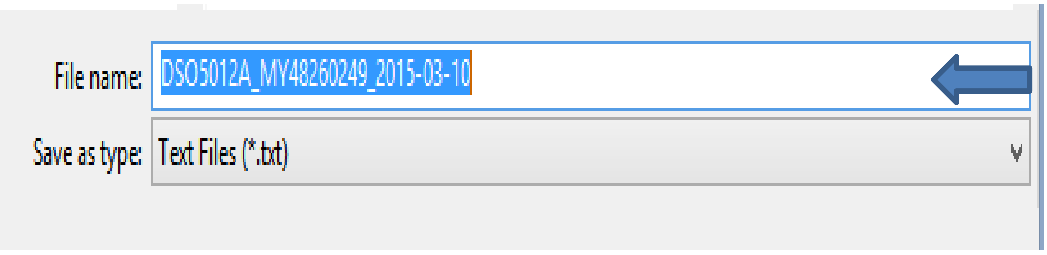
🟦 : enter valid number value

🟨 : enter non negative frequency value

Figure 31: Saving created table

If the user desires to save the data in the table, then the user clicks the save button, the application provides the user with a save file dialog box to specify the file path.

Note: the filename is automatically generated by the application which consist of the oscilloscope model number, serial number and the present date.



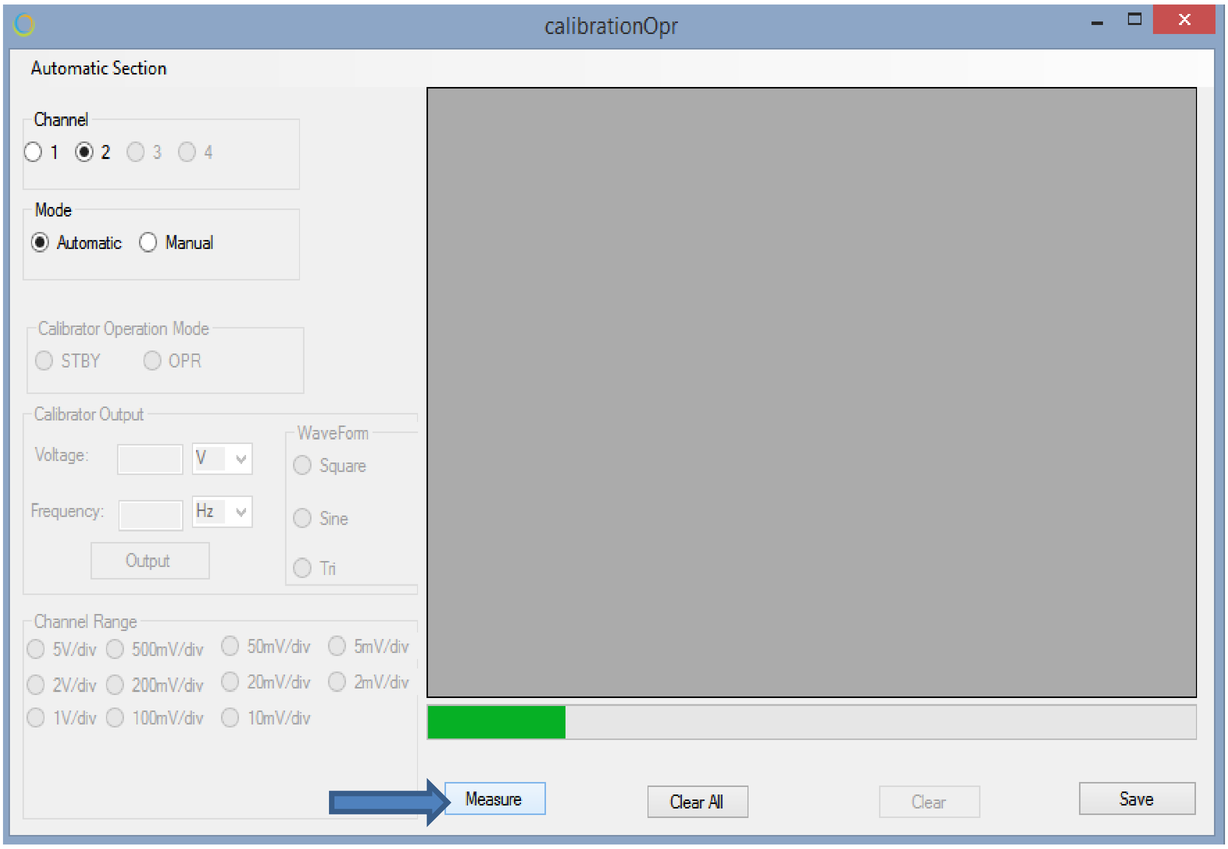
File name: DSO5012A_MY48260249_2015-03-10

Save as type: Text Files (*.txt)

Figure 32: Save table

- **Load button**

If the user decides to click the load button, the calibration page is presented again and the user should select the channel to calibrate and click the measure button. The calibration process starts and the calibration progress can be monitored with the progress bar movement.



calibrationOpr

Automatic Section

Channel
○ 1 ● 2 ○ 3 ○ 4

Mode
● Automatic ○ Manual

Calibrator Operation Mode
○ STBY ○ OPR

Calibrator Output
Voltage: V
Frequency: Hz

WaveForm
○ Square ○ Sine ○ Tri

Channel Range
○ 5V/div ○ 500mV/div ○ 50mV/div ○ 5mV/div
○ 2V/div ○ 200mV/div ○ 20mV/div ○ 2mV/div
○ 1V/div ○ 100mV/div ○ 10mV/div

Figure 33: Automatic process

- **Loading Data from a file**

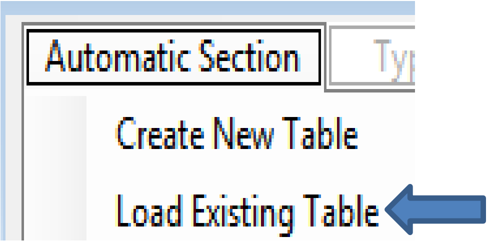


Figure 34: Loading from existing file

1. Selecting the ‘load existing table’ from the menu strip presents the user with an open file dialog box where the user can select a text file containing the data to be loaded

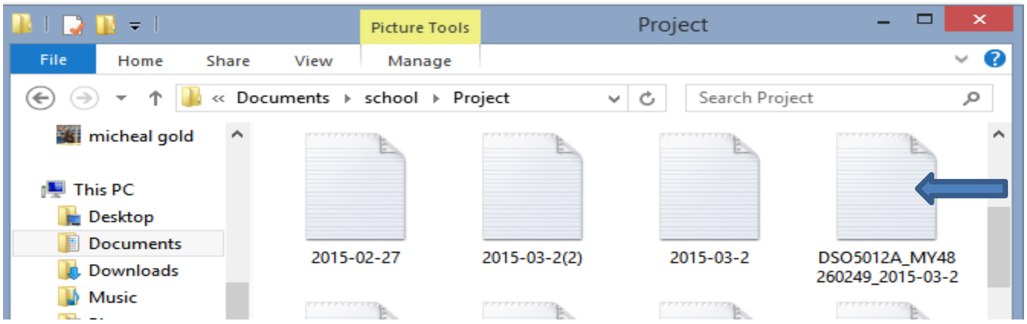


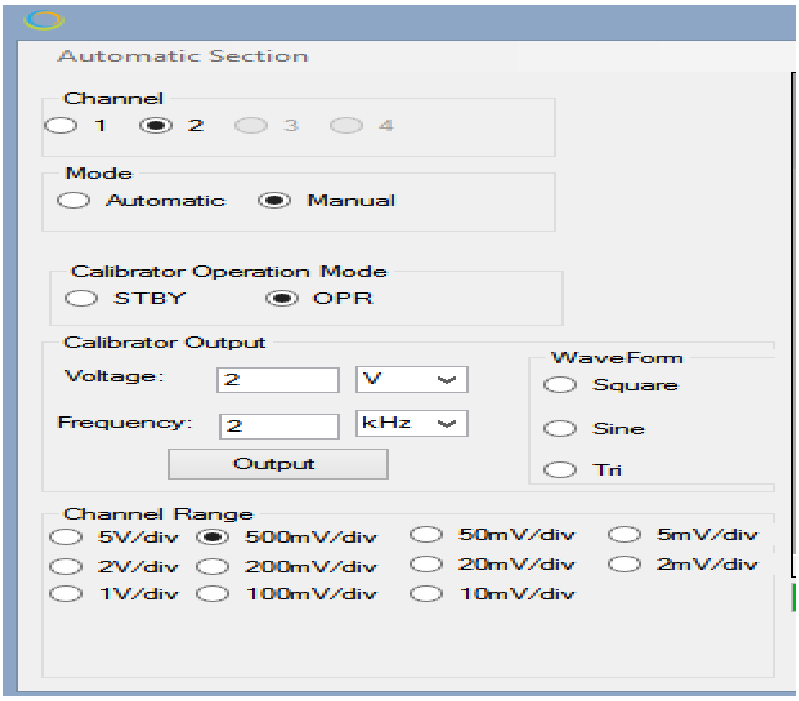
Figure 35: File dialog to select file to load data

Note: only load files that have been saved using the program. Any other text file can result in error either to the application or to the devices.

2. Select the appropriate channel.

3. Click the measure button.

Manual mode



Automatic Section

Channel
○ 1 ● 2 ○ 3 ○ 4

Mode
○ Automatic ● Manual

Calibrator Operation Mode
○ STBY ● OPR

Calibrator Output
Voltage: 2 V
Frequency: 2 kHz
Output

WaveForm
○ Square
○ Sine
○ Tri

Channel Range
○ 5V/div ● 500mV/div ○ 50mV/div ○ 5mV/div
○ 2V/div ○ 200mV/div ○ 20mV/div ○ 2mV/div
○ 1V/div ○ 100mV/div ○ 10mV/div

Figure 36: Manual mode view

- **Sending values to the calibrator**

1. Enter a valid voltage value in the voltage box provided and select the unit from the drop box in front (either V or mV).

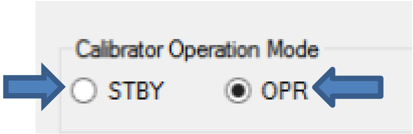
Note: the program recognizes both dot and comma as decimal point. Ensure characters are not entered in the voltage value.

2. Enter a valid frequency value in the frequency box and select the unit from the combo box in front (either Hz, KHz, MHz).

Note: ensure the frequency value is not a negative value and also not a character.

3. Click the output button to send the values to the calibrator.

4. To send the value from the calibrator to the oscilloscope, press the OPR radio button and to stop sending the value from the calibrator to the oscilloscope, press the STBY radio button.



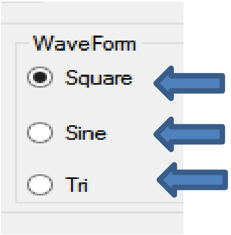
Calibrator Operation Mode

○ STBY ● OPR

Figure 37: Standby or operation mode selection

- **Setting the wave generated by the calibrator**

If the signal produced by the calibrator is AC, select the waveform as desired from the waveform radio button (sine, square, tri).



WaveForm

● Square

○ Sine

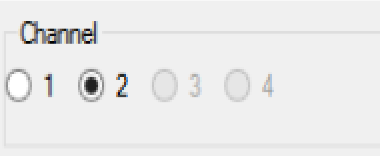
○ Tri

Figure 38: waveform selection

Note: the waveform is not active if the signal produced by the calibrator is DC

- **Adjusting the oscilloscope**

1. Select the appropriate channel under calibration.



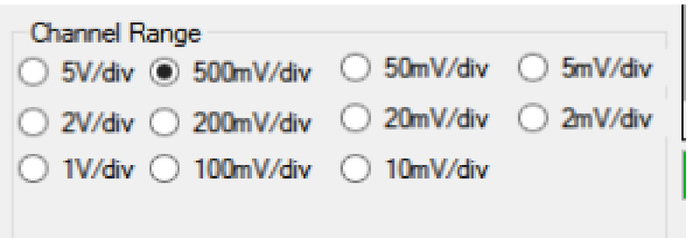
Channel

○ 1 ● 2 ○ 3 ○ 4

Figure 39: Channel selection

Note: by default, the application detects the total number of channels available on the oscilloscope being calibrated and enables on that number of channels for the user.

2. Once the calibrator is sending signal to the oscilloscope. The channel range is used to adjust the vertical scale division of the oscilloscope.



Channel Range

○ 5V/div ● 500mV/div ○ 50mV/div ○ 5mV/div

○ 2V/div ○ 200mV/div ○ 20mV/div ○ 2mV/div

○ 1V/div ○ 100mV/div ○ 10mV/div

Figure 40: Channel range selection

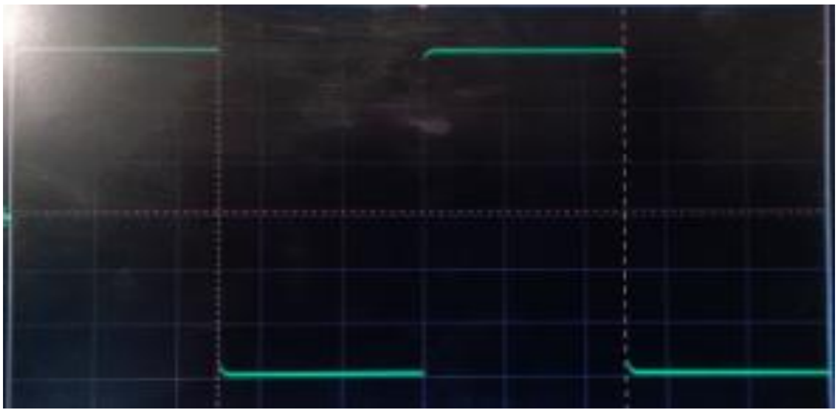


Figure 41: Oscilloscope period view

Note. If the signal is AC, the program automatically adjust the oscilloscope horizontal time division to display the largest visible period that can be produced by the oscilloscope.

- **Measuring the value**

1. Once the desired value enter have been sent to the calibrator and the oscilloscope have been adjusted to the desired visible output.
2. Click the measure button to display the voltage and frequency measured by the oscilloscope as well as the voltage and frequency sent by the calibrator, the maximum and minimum voltage value expected of the oscilloscope and finally the status of the oscilloscope.

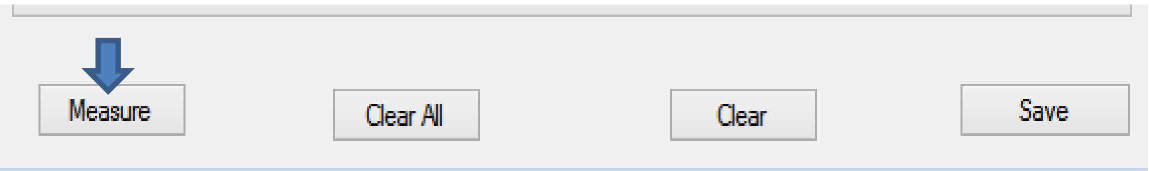
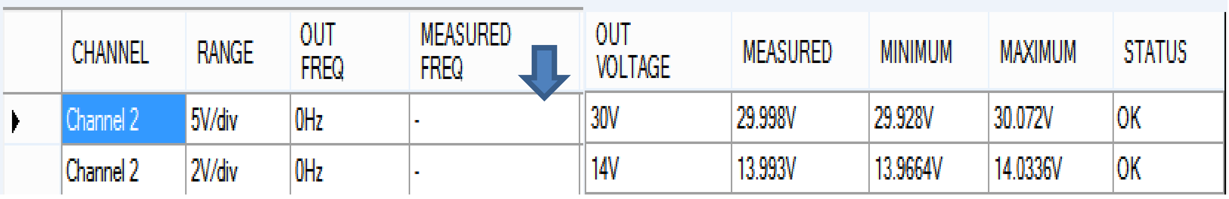


Figure 42: Measuring the value

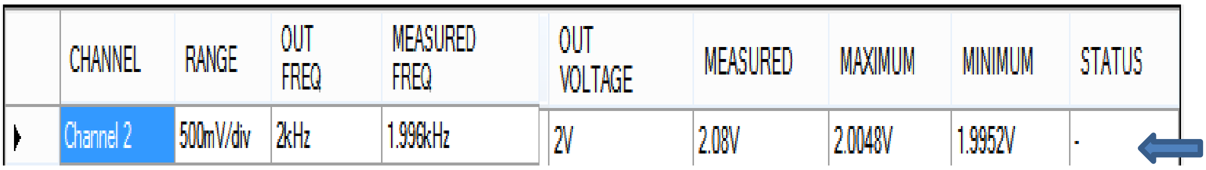
Note: The frequency from the oscilloscope is not measured if the signal is DC as in the figure below.



	CHANNEL	RANGE	OUT FREQ	MEASURED FREQ	OUT VOLTAGE	MEASURED	MINIMUM	MAXIMUM	STATUS
▸	Channel 2	5V/div	0Hz	-	30V	29.998V	29.928V	30.072V	OK
	Channel 2	2V/div	0Hz	-	14V	13.993V	13.9664V	14.0336V	OK

Figure 43: Displayed data for DC

Note: The application only checks the status if the signal is in DC and not in the case of AC signal as can be seen in the figure below.

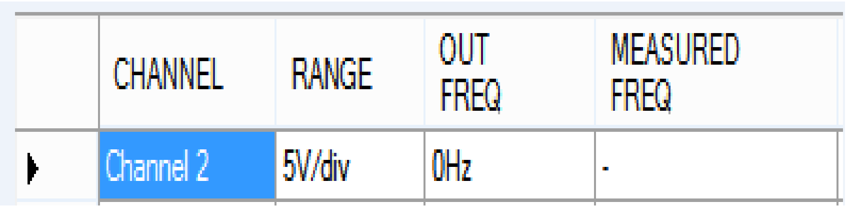


	CHANNEL	RANGE	OUT FREQ	MEASURED FREQ	OUT VOLTAGE	MEASURED	MAXIMUM	MINIMUM	STATUS
▸	Channel 2	500mV/div	2kHz	1.996kHz	2V	2.08V	2.0048V	1.9952V	-

Figure 44: Displayed data for AC

- **Clearing displayed data**

1. Highlight the row to be cleared



	CHANNEL	RANGE	OUT FREQ	MEASURED FREQ
▶	Channel 2	5V/div	0Hz	-

Figure 45: Highlighting row

2. Select the clear button to clear the selected row

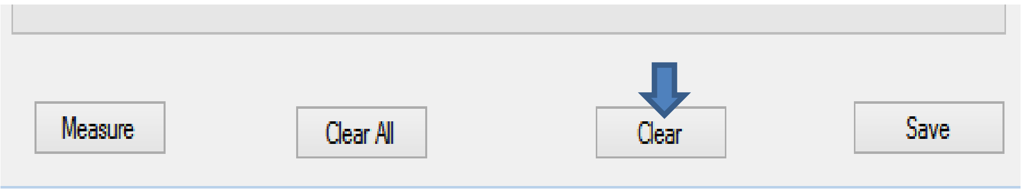


Figure 46: Clear button selection

- **Clearing all the displayed data**

To clear all the data displayed on the page, click on the clear all button below the displayed data

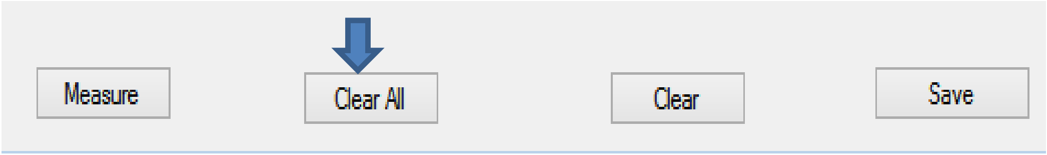


Figure 47: Clear all button selection

- **Saving the data displayed**

Once the desired data have been obtained, click the save button to save the data

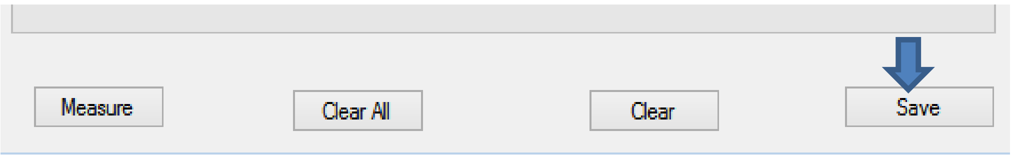
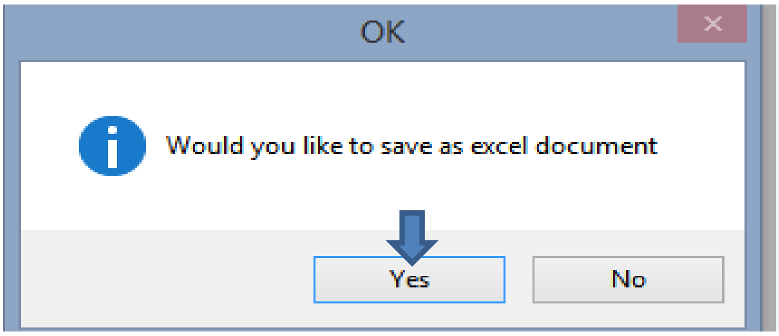


Figure 48: saving displayed data

Note: if Microsoft excel is installed in the computer, a message box is prompted asking if the user will like to save the data as an excel document or not.

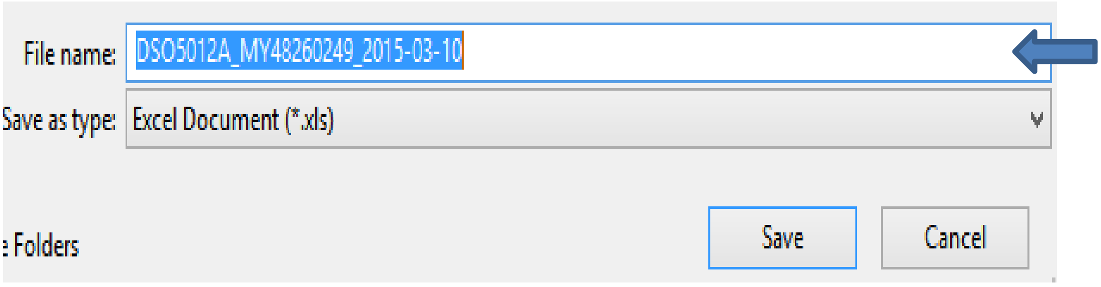
Note: if Microsoft excel is not installed on the PC, the application automatically saves the data as a text file and does not present the message box.



Would you like to save as excel document

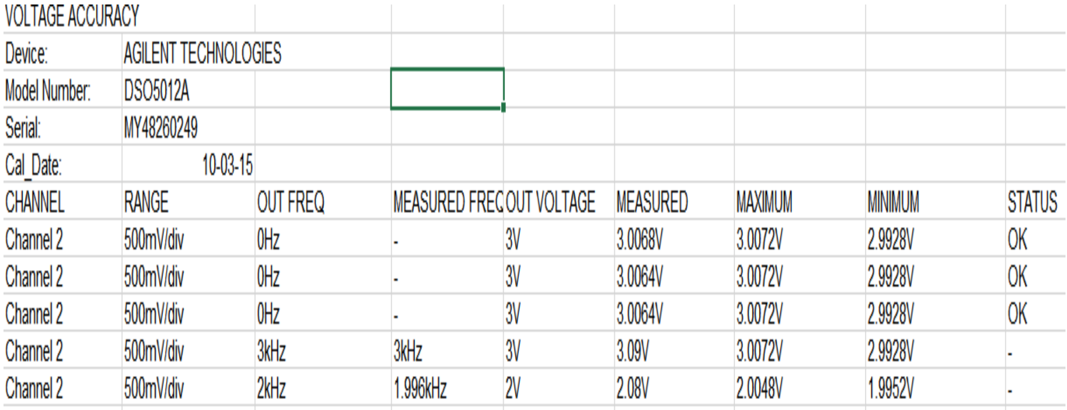
Figure 49: Message box for excel

If yes is selected, the program provides a file dialog box to specify the desired file location. By default, the file name contains the model number, serial number of the oscilloscope including the present date.


File name: DSO5012A_MY48260249_2015-03-10

Save as type: Excel Document (*.xls)

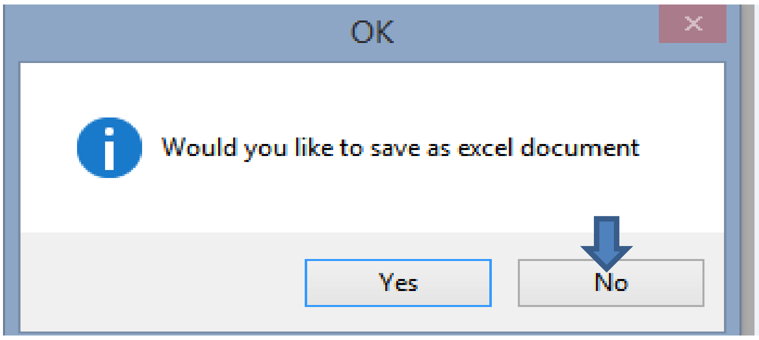
Figure 50: Save dialog box for file location selection



VOLTAGE ACCURACY								
Device:	AGILENT TECHNOLOGIES							
Model Number:	DSO5012A							
Serial:	MY48260249							
Cal_Date:	10-03-15							
CHANNEL	RANGE	OUT FREQ	MEASURED FREC	OUT VOLTAGE	MEASURED	MAXIMUM	MINIMUM	STATUS
Channel 2	500mV/div	0Hz	-	3V	3.0068V	3.0072V	2.9928V	OK
Channel 2	500mV/div	0Hz	-	3V	3.0064V	3.0072V	2.9928V	OK
Channel 2	500mV/div	0Hz	-	3V	3.0064V	3.0072V	2.9928V	OK
Channel 2	500mV/div	3kHz	3kHz	3V	3.09V	3.0072V	2.9928V	-
Channel 2	500mV/div	2kHz	1.996kHz	2V	2.08V	2.0048V	1.9952V	-

Figure 51: Saved excel document

Note. By default, the application saves the data and includes the oscilloscope manufacturer name, model number, serial number and the date.

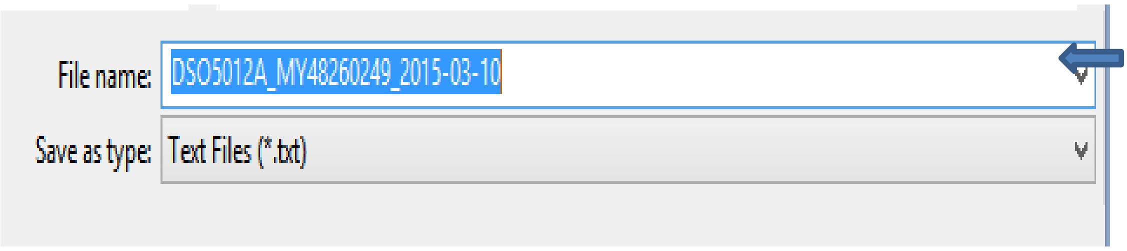


Would you like to save as excel document

Figure 52: Saving file to text file

If no is selected, the application saves the data as a text file.

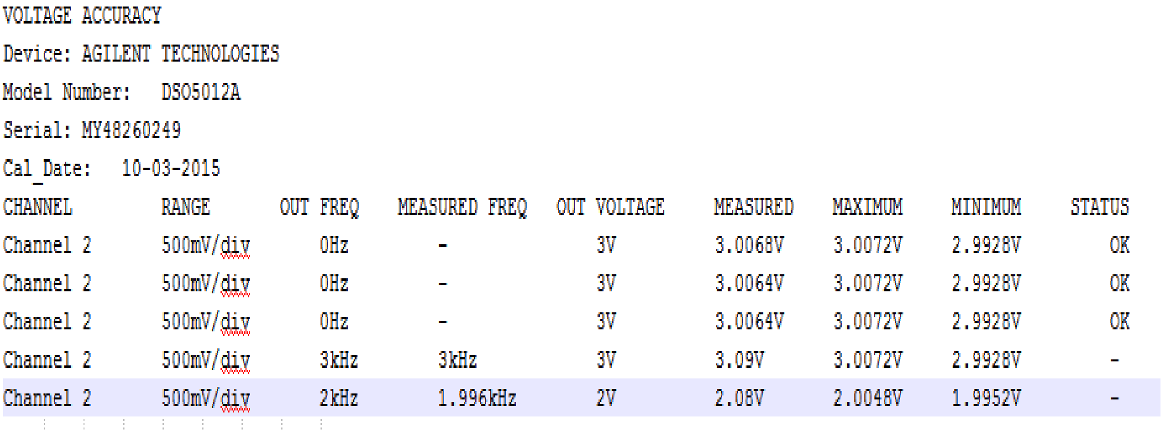
Note: the filename also contains the model number, serial number of the oscilloscope as well as the present date.



File name: DSO5012A_MY48260249_2015-03-10

Save as type: Text Files (*.txt)

Figure 53: save file dialog for specifying file location



```
VOLTAGE ACCURACY
Device: AGILENT TECHNOLOGIES
Model Number:   DSO5012A
Serial: MY48260249
Cal_Date:   10-03-2015
CHANNEL      RANGE        OUT FREQ   MEASURED FREQ   OUT VOLTAGE   MEASURED   MAXIMUM   MINIMUM   STATUS
Channel 2    500mV/div    0Hz        -               3V            3.0068V    3.0072V   2.9928V   OK
Channel 2    500mV/div    0Hz        -               3V            3.0064V    3.0072V   2.9928V   OK
Channel 2    500mV/div    0Hz        -               3V            3.0064V    3.0072V   2.9928V   OK
Channel 2    500mV/div    3kHz       3kHz            3V            3.09V      3.0072V   2.9928V   -
Channel 2    500mV/div    2kHz       1.996kHz        2V            2.08V      2.0048V   1.9952V   -
```

Figure 54: Saved text document

Note: By default, the application saves the data and includes the oscilloscope manufacturer name, model number, serial number and the date. Also, the text file is separated by tabulator for easy exportation of the file to excel.

7 CONCLUSION

The development of the application was a success. The application met all the requirements and functionalities. Calibration of the instrument (oscilloscope) was made easily with the use of the application.

The display of the results on the screen enabled easy decision making in the calibration of the instrument. In addition, saving of the data obtained from the calibration process to a file made it easy to view the history of the instrument status.

7.1 Challenges

Of course, there were series of challenges encountered during the development of the application. One of which was the connecting driver which was only compactible with the visual studio 2012 and above. This factor limited the use of the connecting driver in PC installed with lesser aged visual studio.

In addition, during the testing of the application, it was observed that the process cannot be interrupted once the calibration was going on. This factor is more conspicuous in the automatic calibration process in which the loop of data is loaded by the application therefore limiting the user from stopping the process as desired.

Furthermore, the instruments have a default setting which cannot be changed on lunching of the application. For instance, the calibrator by default generates a square wave on initialization which cannot be change on lunching the application. This situation was addressed by also setting the application to generate square waves on lunching. Though, the waveform can later be changed once the application is up and running to the other waveform such as sine and tri which are made available on the application.

7.2 Possible improvement

The calibration solution provided by this application is limited to the Agilent Oscilloscope. Though the application is able to calibrate other series of oscilloscope from this manufacturer. However, it would be nice if the application could calibrate any other instrument such as multimeters, voltmeters and so on.

Therefore, an obvious improvement to the application will be its ability to identify and calibrate any measuring instrument in the laboratory.

8 REFERENCES

/1/ .NET framework overview.

http://www.microsoft.com/net

/2/ Visual Studio

https://msdn.microsoft.com/en-us/library/dd831853.aspx

/3/ C# and .NET framework

https://msdn.microsoft.com/en-us/library/67ef8sbd.aspx

/4/ NI VISA

http://www.ni.com/visa/

/5/ Agilent DSO5012A Oscilloscope

http://www.keysight.com/en/pd-1077667-pn-DSO5012A/5000-series-oscilloscope-100-mhz-2-channels?cc=FI&lc=fin

/6/ Agilent DSO5012A Programming Command

Agilent InfiniiVsion 5000 Series Oscilloscope Programmer’s Guide, January 21, 2010.

/7/ Fluke 5500A Calibrator

http://us.flukecal.com/products/obsolete-products/5500a-multi-product-calibrator

/8/ Fluke Operation Manual

Fluke 5500A Multi Calibrator Operation Manual, December 1994.

/9/ GPIB

http://sine.ni.com/np/app/main/p/ap/ictrl/lang/en/pg/1/sn/n17:ictrl,n21:25/fmid/6003/