











# Acknowledgments

*I would like to thank my supervisor Alain Sylvestre for his guidance and support throughout this diploma thesis.*

*I would also like to thank my colleagues from CNRS for their encouragement and help during my studies in Grenoble.*

*But above all I would like to thank my parents and girlfriend. This work would not have been achievable nor possible without their support.*

# Abbreviations

## A

ASCII *American Standard Code for Information Interchange*  
ATN *Attention*

## C

CAMAC *Computer Automated Measurement And Control*  
CAN *Controller Area Network*  
CGI *Common Gateway Interface*  
CNRS *Centre National de la Recherche Scientifique*

## D

DAQ *Data acquisition*  
DAV *Data Valid*

## E

EOI *End Or Identify*

## G

G2Elab *Laboratory of Electrical Engineering of Grenoble*  
GPIB *General Purpose Interface Bus*

## H

HTML *Hyper Text Markup Language*  
HTTP *Hypertext Transfer Protocol*

## I

I/O *Input/Ouput*  
IEEE *Institute of Electrical and Electronic Engineers*  
IFC *Interface Clear*  
IP *Internet protocol*

## L

LabVIEW *Laboratory Virtual Instrument Engineering Workbench*  
LEMD *Laboratory of Electrostatic and Dielectric Materials*

## N

NDAC *Not Data Accepted*  
NRFD *Not Ready For Data*

## **P**

PC

*Personal Computer*

PCMCIA

*Personal Computer Memory Cards International Association*

PXI

*PCI eXtensions for Instrumentation*

## **R**

REN

*Remote Enable*

RS-232

*Recommended Standard 232*

## **S**

SCPI

*Standard Commands for Programmable Instruments*

SRQ

*Service Request*

## **T**

TCP

*Transmission Control Protocol*

## **U**

URL

*Uniform Resource Locator*

## **V**

VI

*Virtual Instrument*

VXI

*VME eXtensions for Instrumentation*

# Contents

ACKNOWLEDGMENTS	6
ABBREVIATIONS	7
CONTENTS	9
INTRODUCTION	12
ELECTRICAL CONDUCTION IN INSULATING MATERIALS	13
2.1 Overview	13
2.2 Team of G2Elab	13
2.3 Physical principle	13
PROGRAMMING IN LABVIEW	16
3.1 Introduction to LabVIEW	16
3.2 Introduction to Virtual Instruments	16
3.3 Building the Front Panel	17
3.3.1 Configuring Objects on the Front Panel	17
3.4 Building the Block Diagram	18
3.4.1 Relationship between Front Panel Objects and Block Diagram Terminals	18
3.4.2 Block Diagram Data Flow	18
3.4.3 Data Flow and Managing Memory	18
3.5 Local and Global Variables	19
3.5.1 Local Variables	19
3.5.2 Global Variables	19
3.5.3 Read and Write Variables	20
3.5.4 Initializing Local and Global Variables	20
3.6 Programmatically Controlling VIs	20
3.6.1 Capabilities of the VI Server	20
3.6.2 Building VI Server Applications	21
3.6.3 Application and VI References	22
3.6.4 Property Nodes	22
3.6.5 Implicitly Linked Property Nodes	23
3.6.6 Invoke Nodes	23
3.6.7 Manipulating Application Class Properties and Methods	23
3.6.8 Editing and Running VIs on Remote Computers	24
DATA ACQUISITION	25
4.1 Comparing DAQ Devices and Special-Purpose Instruments for Data Acquisition	25
4.2 DAQ Devices versus Special-Purpose Instruments	25
4.3 How Do Computers Talk to DAQ Devices?	26
4.3.1 Role of Software	28
4.4 How Do Computers Talk to Special-Purpose Instruments?	28

4.4.1	How Do Programs Talk to Instruments?	28
4.5	Types of Instruments	29
4.6	GPIB Communications	29
4.6.1	Controllers, Talkers, and Listeners	29
4.6.2	Hardware Specifications	30
<b>REMOTE LABORATORY USING LABVIEW</b>		<b>31</b>
5.1	Overview	31
5.2	Distance-Learning Remote Laboratories	31
5.3	Remote Laboratory in world	32
5.3.1	Stanford University – Cyberlab	32
5.3.2	Swiss Federal Institute of Technology – Lecture Enhancement	33
5.3.3	Dalhousie University – Virtual Laser Laboratory	34
5.4	LabVIEW Remote Panels	35
5.5	Enabling Remote Panel	36
5.6	Client Operation	39
5.6.1	Required Software	39
5.6.2	Application Control	39
5.6.3	Releasing Control	40
5.7	Application Administration	40
5.7.1	Remote Panel Connection Manager	40
5.7.2	Web Server: Configuration	41
5.7.3	Application Security	42
<b>KEITHLEY</b>		<b>44</b>
6.1	Electrometer/High Resistance Meter	44
6.2	Exceptional Performance Specifications	44
6.2.1	Wide Measurement Ranges	45
6.3	Why this one?	45
6.4	IEEE- 488 Reference	45
6.4.1	IEEE- 488 bus connections	45
6.5	Trigger Model (IEEE-488 operation)	46
6.5.1	Idle and initiate	46
6.5.2	Trigger model layers	46
6.6	Command words	50
6.6.1	Commands and command parameters:	50
6.6.2	Case sensitivity	52
6.6.3	Long-form and short-form versions	52
<b>SOFTWARE</b>		<b>53</b>
7.1	Introduction	53
7.2	Describe of Front panel	54
7.3	Describe of Block Diagram	65
7.4	Results	70
<b>CONCLUSION</b>		<b>75</b>



# Chapter 1

## Introduction

These master thesis were made foreign and their content wasn't known beforehand. Whole project was created only here, in Grenoble. Basics of programming in LabVIEW, detailed analyses of virtual and remote laboratories and their examples aren't shown here. For those pieces of information I recommend you to look into my semestral project. In a following text there will be analysed questions of remote laboratories, which are made in LabVIEW. You will be informed about items, which are necessary for advanced programming in language G. In addition, programming of measuring instrument Keithley 6517A will be explained in the following text. There are mentioned characteristics and possibility of communication through the use of IEEE-488. Physical base of the problem will be shortly analysed in order to make clear why this kind of software. More detailed information about dielectrical materials you can find in [16][17][18]. There is some information about obtaining data from real equipment. The main aim of the thesis has been to create a measuring system in LabVIEW, which is able to generate square-wave voltage and at the same time measure low data of current. Measurement should be time-unlimited because CNRS and its institution LEMD, for which the program was made, is engaged in research of dielectrical material's electrical properties. These materials change their electrical properties in time. Scientists from LEMD search materials for various firms, e. g. Schneider Electric, EDF, GEC Alsthom, Bolloré, ELF Atochem, Pirelli, Hydroquebec, Toxot, Varioptic a PME-PMI from region Rhone Alpes. LEMD cooperates with other French laboratories or foreign scientific institutions. (Canada, Norway, Russia, Spain, Rumania, etc.). A chief of the graduation thesis hasn't been interested in creating the remote laboratory too much, the main aim has been to make the working system, which is able to measure pikoamperes. Some results of measurement in this program will be shown in the end of the thesis.

# Chapter 2

## Electrical conduction in insulating materials

### 2.1 Overview

Insulating material generally called dielectrics are necessary to many applications. We could find them in mobilphone; they are used to make capacitors which are integrated in ICs(microelectronics). We couldn't transport current without these kind of material, without them it would be impossible to insulate high voltage, power and RF cables.

A dielectric is a material through which one current can't pass; it has a high resistivity. However, the resistivity of an insulator varies greatly with the purity and surface condition of the material, time of application of voltage, and in some cases with the magnitude of the applied stress. It usually decreases rapidly as the temperature is increased (in many cases changing by orders of magnitude) and may be reduced temporarily or permanently by irradiation.

### 2.2 Team of G2Elab

My internship proceeded within the team '*Dielectric Materials and Electrostatic*' at the *Laboratory of Electrical Engineering of Grenoble (G2Elab)*. In this team, researchers investigate the understanding of leakage/parasitic currents when insulating materials are subjected to high voltages and temperatures. These materials are mainly dedicated for high-voltage applications and the appearance of current induce a reinforcing electric field in the material. This electric field involves then a local heating in the bulk of the material which can bring to the breakdown of the insulator and the electric apparatus which it protects. Other insulating materials investigated are dedicated for integration in electronic and microelectronic systems. There still, leakage currents in these materials involve a dysfunction in the electronic circuit and it is so important to measure and analyze the origin of this current against the process for growing these materials, the temperature and the voltage.

### 2.3 Physical principle

One important parameter for measuring the current is the duration of recording of the current after the application of the voltage. Indeed; as a voltage is applied to an insulating material, it is observed that the current doesn't reach directly a constant value but it decreases as a function of time. This decrease is due to the so-called '*polarization mechanisms*' particularly present in polar insulating materials. As subjected to a voltage, dipoles in the bulk of the

materials can orient themselves and induce a current. This orientation can take a very long time (seconds, hours, weeks, years!) and so it is necessary to develop program for monitoring this current as a function of time. [19]

A schematic representation of the current is presented below as a voltage is applied on an insulating material and then when the voltage is switch off.

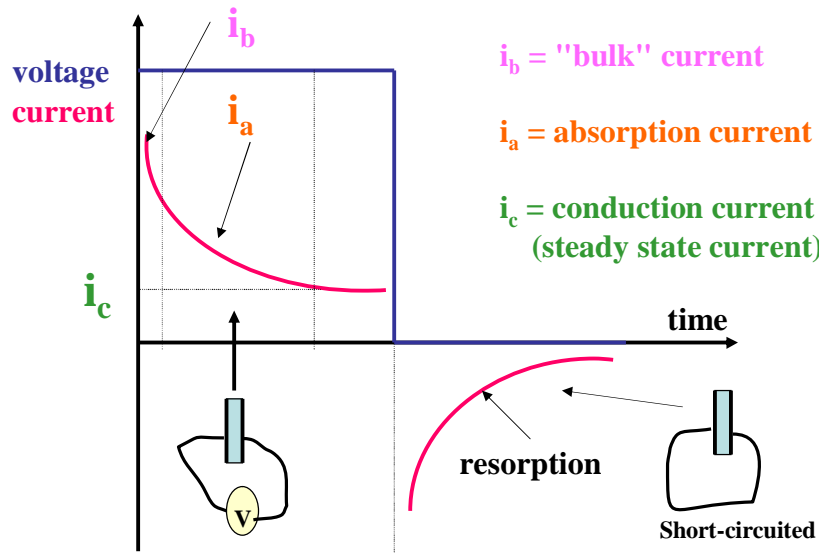
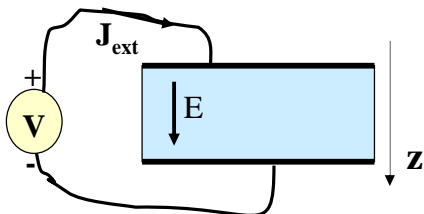


Fig. 1: Running of current

The expression of the current is expressed by this equation [19] :

$$\mathbf{J}_{\text{ext}}(z,t) = \underbrace{\dot{a}_i q_i u_i n_i(z,t) E(z,t)}_{\mathbf{J}_c(z,t)} - \underbrace{\dot{a}_i q_i D_i \frac{\nabla n_i(z,t)}{\nabla z}}_{\mathbf{J}_{\text{diff}}(z,t)} + \underbrace{\frac{\nabla D(z,t)}{\nabla t}}_{\mathbf{J}_{\text{dis}}(z,t)}$$

Conduction
diffusion
displacement



- $q_i$  = electrical charge
- $n_i(z,t)$  = carrier's density
- $u_i(z,t)$  = carrier's mobility
- $D_i$  = carrier's diffusion coef.
- $E(z,t)$  = local field
- $D(z,t)$  = displacement vector

I don't develop the significance of all these terms. We can just note that the current is the sum of three terms and that the last part of the current (labelled 'displacement current') is at the origin of the time-dependence of this global current.

More precisely, concerning the displacement current, it can be expressed by the expression below [19] :

$$\mathbf{J}_{\text{dis}}(\mathbf{z}, t) = \frac{\partial \mathbf{D}}{\partial t}$$

**Polarization vector**

**as :  $\mathbf{D}(\mathbf{z}, t) = \epsilon_0 \mathbf{E}(\mathbf{z}, t) + \mathbf{P}(\mathbf{z}, t)$**

time-constant  
decreases ~ 1 sec.

$$\mathbf{J}_{\text{dis}}(\mathbf{z}, t) = \epsilon_0 \frac{\partial \mathbf{E}}{\partial t} + \frac{\partial \mathbf{P}}{\partial t} = J_p = \dot{\sum}_i N_i q_i \frac{\partial l_i}{\partial t}$$

Displacement  
current  
in the vacuum  
(free charges)

Polarization  
current  
density  
(bind charges)

$l_i$  = local displacement  
of bind charges

$N_i$  = total density of  
bind charges

Finally, the time-dependent current is due to bind charges located in the bulk of the material. When an electric field or voltage is applied to the material, the moving of these charges can be low and explains the observation of a decrease of this current as a function of time.

# Chapter 3

## Programming in LabVIEW

### 3.1 Introduction to LabVIEW

LabVIEW is a graphical programming language that uses icons instead of lines of text to create applications. In contrast to text-based programming languages, where instructions determine program execution, LabVIEW uses dataflow programming, where the flow of data determines execution.

In LabVIEW, we build a user interface by using a set of tools and objects. The user interface is known as the front panel. We then add code using graphical representations of functions to control the front panel objects. The block diagram contains this code. In some ways, the block diagram resembles a flowchart. We can purchase several add-on software toolsets for developing specialized applications. All the toolsets integrate seamlessly in LabVIEW.

### 3.2 Introduction to Virtual Instruments

LabVIEW programs are called virtual instruments, or VIs, because their appearance and operation imitate physical instruments, such as oscilloscopes and multimeters. Every VI uses functions that manipulate input from the user interface or other sources and display that information or move it to other files or other computers. A VI contains the following three components:

- **Front panel**—Serves as the user interface.
- **Block diagram**—Contains the graphical source code that defines the functionality of the VI.
- **Icon and connector pane**—Identifies the VI so that we can use the VI in another VI. A VI within another VI is called a subVI. A subVI corresponds to a subroutine in text-based programming languages.

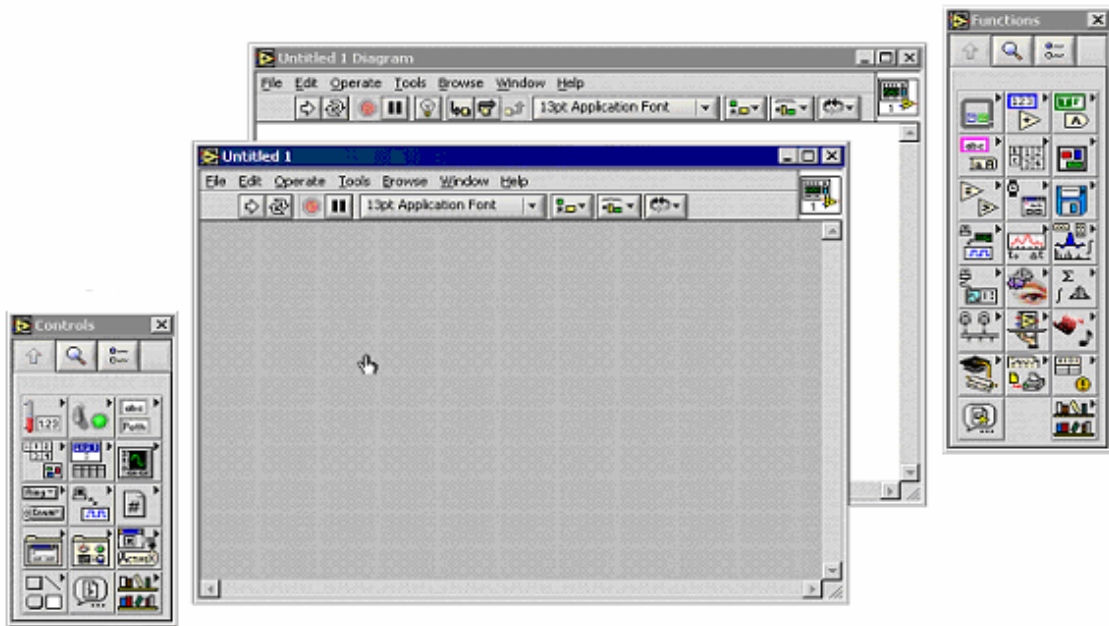


Fig. 2: Development environment of LabVIEW

Use the LabVIEW palettes, tools, and menus to build the front panels and block diagrams of VIs. We can customize the **Controls** and **Functions** palettes, and we can set several work environment options.

### 3.3 Building the Front Panel

The front panel is the user interface of a VI. Generally, we design the front panel first, then design the block diagram to perform tasks on the inputs and outputs we create on the front panel. Refer to paragraph 3.4 *Building the Block Diagram*, for more information about the block diagram.

We build the front panel with controls and indicators, which are the interactive input and output terminals of the VI, respectively. Controls are knobs, push buttons, dials, and other input devices. Indicators are graphs, LEDs, and other displays. Controls simulate instrument input devices and supply data to the block diagram of the VI. Indicators simulate instrument output devices and display data the block diagram acquires or generates.

Select **Window»Show Controls Palette** to display the **Controls** palette, then select controls and indicators from the **Controls** palette and place them on the front panel.

#### 3.3.1 Configuring Objects on the Front Panel

We can customize the front panel by using the control and indicator shortcut menus, by setting the order of front panel objects, and by using imported graphics. We also can manually resize front panel objects and set them to automatically resize when the window size changes. For more

information about creating and using controls, indicators, and type definitions you can look to my semestral thesis.

## **3.4 Building the Block Diagram**

After we build the front panel, we add code using graphical representations of functions to control the front panel objects. The block diagram contains this graphical source code.

### **3.4.1 Relationship between Front Panel Objects and Block Diagram Terminals**

Front panel objects appear as terminals on the block diagram. Double-click a block diagram terminal to highlight the corresponding control or indicator on the front panel.

Terminals are entry and exit ports that exchange information between the front panel and block diagram. Data we enter into the front panel controls enter the block diagram through the control terminals. When the VI finishes running, the output data flow to the indicator terminals, where they exit the block diagram, reenter the front panel, and appear in front panel indicators.

### **3.4.2 Block Diagram Data Flow**

LabVIEW follows a dataflow model for running VIs. A block diagram node executes when all its inputs are available. When a node completes execution, it supplies data to its output terminals and passes the output data to the next node in the dataflow path.

Visual Basic, C++, JAVA, and most other text-based programming languages follow a control flow model of program execution. In control flow, the sequential order of program elements determines the execution order of a program.

In LabVIEW, because the flow of data rather than the sequential order of commands determines the execution order of block diagram elements, we can create block diagrams that have simultaneous operations. LabVIEW is a multitasking and multithreaded system, running multiple execution threads and multiple VIs simultaneously. [3]

### **3.4.3 Data Flow and Managing Memory**

Dataflow execution makes managing memory easier than the control flow model of execution. In LabVIEW, we do not allocate variables or assign values to them. Instead, we create a block diagram with wires that represent the transition of data.

VIs and functions that generate data automatically allocate the memory for that data. When the VI or function no longer uses the data, LabVIEW

deallocates the associated memory. When we add new data to an array or a string, LabVIEW allocates enough memory to manage the new data.

Because LabVIEW automatically handles memory management, we have less control over when memory is allocated or deallocated. If VI works with large sets of data, we need to understand when memory allocation takes place. Understanding the principles involved can help we write VIs with significantly smaller memory requirements. Minimizing memory usage can help us increase the speed at which VIs run. [1]

## 3.5 Local and Global Variables

In LabVIEW, we read data from or write data to a front panel object using its block diagram terminal. However, a front panel object has only one block diagram terminal, and application might need to access the data in that terminal from more than one location.

Local and global variables pass information between locations in application that we cannot connect with a wire. Use local variables to access front panel objects from more than one location in a single VI. Reader can use global variables to access and pass data among several VIs.

### 3.5.1 Local Variables

Use local variables to access front panel objects from more than one location in a single VI and pass data between block diagram structures that we cannot connect with a wire. With a local variable, we can write to or read from a control or indicator on the front panel. Writing to a local variable is similar to passing data to any other terminal. However, with a local variable we can write to it even if it is a control or read from it even if it is an indicator. In effect, with a local variable, we can access a front panel object as both an input and an output.

For example, if the user interface requires users to log in, we can clear the **Login** and **Password** prompts each time a new user logs in. Use a local variable to read from the **Login** and **Password** string controls when a user logs in and to write empty strings to these controls when the user logs out.

### 3.5.2 Global Variables

Use global variables to access and pass data among several VIs that run simultaneously. Global variables are built-in LabVIEW objects. When we create a global variable, LabVIEW automatically creates a special global VI, which has a front panel but no block diagram. Add controls and indicators to the front panel of the global VI to define the data types of the global variables it contains. In effect, this front panel is a container from which several VIs can access data.

For example, suppose we have two VIs running simultaneously. Each VI contains a While Loop and writes data points to a waveform chart. The first VI contains a Boolean control to terminate both VIs. We must use a global variable

to terminate both loops with a single Boolean control. If both loops were on a single block diagram within the same VI, we could use a local variable to terminate the loops.

### 3.5.3 Read and Write Variables

After we create a local or global variable, we can read data from a variable or write data to it. By default, a new variable receives data. This kind of variable works as an indicator and is a write local or global. When we write new data to the local or global variable, the associated front panel control or indicator updates to the new data.

We also can configure a variable to be have as a data source, or a read local or global. Right-click the variable and select **Change To Read** from the shortcut menu to configure the variable to behave as a control. When this node executes, the VI reads the data in the associated front panel control or indicator.

To change the variable to receive data from the block diagram rather than provide data, right-click the variable and select **Change To Write** from the shortcut menu.

On the block diagram, we can distinguish read locals or globals from write locals or globals the same way you distinguish controls from indicators. A read local or global has a thick border similar to a control. A write local or global has a thin border similar to an indicator.

### 3.5.4 Initializing Local and Global Variables

Verify that the local and global variables contain known data values before the VI runs. Otherwise, the variables might contain data that cause the VI to behave incorrectly.

If we do not initialize the variable before the VI reads the variable for the first time, the variable contains the default value of the associated front panel object.

## 3.6 Programmatically Controlling VIs

We can access the VI Server through block diagrams, ActiveX technology, and the TCP protocol to communicate with VIs and other instances of LabVIEW so we can programmatically control VIs and LabVIEW. We can perform VI Server operations on a local computer or remotely across a network.

### 3.6.1 Capabilities of the VI Server

Use the VI Server to perform the following programmatic operations [1]:

- Call a VI remotely.

- Configure an instance of LabVIEW to be a server that exports VIs we can call from other instances of LabVIEW on theWeb. For example, if we have a data acquisition application that acquires and logs data at a remote site, we can sample that data occasionally from local computer. By changing our LabVIEW preferences, we can make some VIs accessible on theWeb so that transferring the latest data is as easy as a subVI call. The VI Server handles the networking details. The VI Server also works across platforms, so the client and the server can run on different platforms.
- Edit the properties of a VI and LabVIEW. For example, we can dynamically determine the location of a VI window or scroll a front panel so that a part of it is visible. We also can programmatically save any changes to disk.
- Update the properties of multiple VIs rather than manually using the **File»VI Properties** dialog box for each VI.
- Retrieve information about an instance of LabVIEW, such as the version number and edition. We also can retrieve environment information, such as the platform on which LabVIEW is running.
- Dynamically load VIs into memory when another VI needs to call them, rather than loading all subVIs when we open a VI.
- Create a plug-in architecture for the application to add functionality to the application after we distribute it to customers. For example, we might have a set of data filtering VIs, all of which take the same parameters. By designing the application to dynamically load these VIs from a plug-in directory, we can ship the application with a partial set of these VIs and make more filtering options available to users by placing the new filtering VIs in the plug-in directory.

### 3.6.2 Building VI Server Applications

The programming model for VI Server applications is based on refnums. Refnums also are used in file I/O, network connections, and other objects in LabVIEW.

Typically, we open a refnum to an instance of LabVIEW or to a VI. We then use the refnum as a parameter to other VIs. The VIs get (read) or set (write) properties, execute methods, or dynamically load a referenced VI. Finally, we close the refnum, which releases the referenced VI from memory.

Use the following functions and nodes located on the **Functions»Application Control** palette to build VI Server applications [5]:

- **Open Application Reference** - Opens a refnum to the local or remote application we are accessing through the server or to access a remote

instance of LabVIEW. Use the Open VI Reference function to access a VI on the local or remote computer.

- **Property Node** - Gets and sets VI or application properties. Refer to the *Property Nodes* section of this chapter for more information about properties.
- **Invoke Node** - Invokes methods on a VI or application. Refer to the *Invoke Nodes* section of this chapter for more information about methods.
- **Call By Reference Node** - Dynamically calls the referenced VI.
- **Close LV Object Reference** - Closes the refnum to the VI or application we accessed using the VI Server.

### 3.6.3 Application and VI References

We access VI Server functionality through references to two main classes of objects - the Application object and the VI object. After we create a reference to one of these objects, we can pass the reference to a VI or function that performs an operation on the object.

An Application refnum refers to a local or remote instance of LabVIEW. We can use Application properties and methods to change LabVIEW preferences and return system information. A VI refnum refers to a VI in the instance of LabVIEW.

With a refnum to an instance of LabVIEW, we can retrieve information about the LabVIEW environment, such as the platform on which LabVIEW is running, the version number, or a list of all VIs currently in memory. We also can set information, such as the current user name or the list of VIs exported to other instances of LabVIEW.

When we get a refnum to a VI, we load the VI into memory. After we get the refnum, the VI stays in memory until we close the refnum. If we have multiple refnums to an open VI at the same time, the VI stays in memory until we close all refnums to the VI. With a refnum to a VI, we can update all the properties of the VI available in the **File»VI Properties** dialog box as well as dynamic properties, such as front panel window position. We also can programmatically print the VI, save it to another location, and export and import its strings to translate into another language. [1]

### 3.6.4 Property Nodes

Use the Property Node to get and set various properties on an application or VI. Select properties from the node by using the Operating tool to click the property terminal or by right-clicking the white area of the node and selecting **Properties** from the shortcut menu.

We can read or write multiple properties using a single node. Use the Positioning tool to resize the Property Node to add new terminals. A small direction arrow to the right of the property indicates a property we read. A small direction arrow to the left of the property indicates a property we write. Right-click the property and select **Change to Read** or **Change to Write** from the shortcut menu to change the operation of the property.

The node executes from top to bottom. The Property Node does not execute if an error occurs before it executes, so always check for the possibility of errors. If an error occurs in a property, LabVIEW ignores the remaining properties and returns an error. The **error out** cluster contains information about which property caused the error.

### 3.6.5 Implicitly Linked Property Nodes

When we create a Property Node from a front panel object by right-clicking the object and selecting **Create»Property Node** from the shortcut menu, LabVIEW creates a Property Node on the block diagram that is implicitly linked to the front panel object. Because these Property Nodes are implicitly linked to the object from which it was created, they have no **refnum** input, and we do not need to wire the Property Node to the terminal of the front panel object or the control refnum. [1]

### 3.6.6 Invoke Nodes

Use the Invoke Node to perform actions, or methods, on an application or a VI. Unlike the Property Node, a single Invoke Node executes only a single method on an application or VI. Select a method by using the Operating tool to click the method terminal or by right-clicking the white area of the node and selecting **Methods** from the shortcut menu. The name of the method is always the first terminal in the list of parameters in the Invoke Node. If the method returns a value, the method terminal displays the return value. Otherwise, the method terminal has no value. The Invoke Node lists the parameters from top to bottom with the name of the method at the top and the optional parameters in gray at the bottom. [5]

### 3.6.7 Manipulating Application Class Properties and Methods

We can get or set properties on a local or remote instance of LabVIEW, perform methods on LabVIEW, or both. Figure 3 shows how to display all VIs in memory on a local computer in a string array on the front panel.

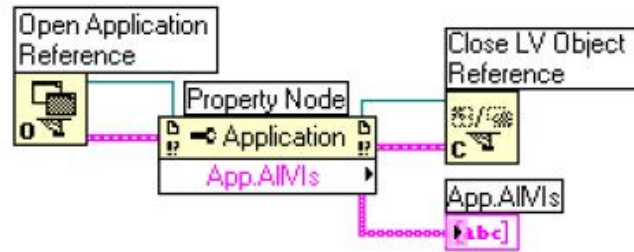


Fig. 3: *Displaying All VIs in Memory on a Local Computer*

To find the VIs in memory on a remote computer, wire a string control to the **machine name** input of the Open Application Reference function, as shown in Figure 4, and enter the IP address or domain name. We also must select the **Exported VIs in Memory** property because the **All VIs in Memory** property used in Figure 3 applies only to local instances of LabVIEW.

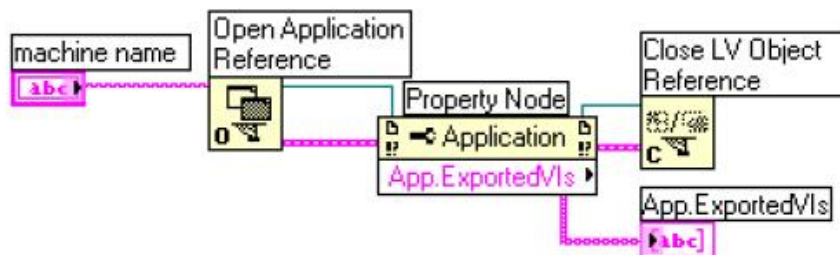


Fig. 4: *Displaying All VIs in Memory on a Remote Computer*

### 3.6.8 Editing and Running VIs on Remote Computers

An important aspect of both Application and VI refsnums is their network transparency. This means we can open refsnums to objects on remote computers in the same way we open refsnums to those objects on our computer.

After we open a refsnum to a remote object, we can treat it in exactly the same way as a local object, with a few restrictions. For operations on a remote object, the VI Server sends the information about the operation across the network and sends the results back. The application looks almost identical regardless of whether the operation is remote or local.

# Chapter 4

## Data Acquisition

### 4.1 Comparing DAQ Devices and Special-Purpose Instruments for Data Acquisition

Measurement devices, such as general-purpose data acquisition (DAQ) devices and special-purpose instruments, are concerned with the acquisition, analysis, and presentation of measurements and other data you acquire.

Acquisition is the means by which physical signals, such as voltage, current, pressure, and temperature, are converted into digital formats and brought into the computer. Popular methods for acquiring data include plug-in DAQ and instrument devices, GPIB instruments, VXI instruments, and RS-232 instruments. [1]

Data analysis transforms raw data into meaningful information. This can involve such things as curve fitting, statistical analysis, frequency response, or other numerical operations.

Data presentation is the means for communicating with system in an intuitive, meaningful format.

Building a computer-based measurement system can be a daunting task. There is a wide variety of hardware components we can use to monitor or control a process or test a device. Should we build on traditional rack-and-stack IEEE 488 equipment or look to modular VXI-based solutions? Or maybe we should consider a PC-based plug-in board approach. Which type of hardware meets our needs today and will be around for the long run? What are the differences between all the choices? This chapter will describe several types of hardware solutions.

### 4.2 DAQ Devices versus Special-Purpose Instruments

The fundamental task of all measurement systems is the measurement and/or generation of real-world physical signals. The primary difference between the various hardware options is the method of communication between the measuring hardware and the computer. In this chapter we will separate the discussion into two categories: general purpose DAQ devices and special purpose instruments.

General purpose DAQ devices are devices that connect to the computer allowing the user to retrieve digitized data values. These devices typically

connect directly to the computer's internal bus through a plug-in slot. Some DAQ devices are external and connect to the computer via serial, GPIB, or ethernet ports. The primary distinction of a test system that utilizes general purpose DAQ devices is where measurements are performed. With DAQ devices, the hardware only converts the incoming signal into a digital signal that is sent to the computer. The DAQ device does not compute or calculate the final measurement. That task is left to the software that resides in the computer. The same device can perform a multitude of measurements by simply changing the software application that is reading the data. So, in addition to controlling, measuring, and displaying the data, the user application for a computer-based DAQ system also plays the role of the firmware—the built-in software required to process the data and calculate the result—that would exist inside a special purpose instrument. While this flexibility allows the user to have one hardware device for many types of tests, the user must spend more time developing the different applications for the different types of tests.

Instruments are like the general purpose DAQ device in that they digitize data. However, they have a special purpose or a specific type of measurement capability. The software, or firmware, required to process the data and calculate the result is usually built in and cannot be modified. For example, a multi-meter can not read data the way an oscilloscope can because the program that is inside the multi-meter is permanently stored and cannot be changed dynamically. Most instruments are external to the computer and can be operated alone, or they may be controlled and monitored through a connection to the computer. The instrument has a specific protocol that the computer must use in order to communicate with the instrument. The connection to the computer could be Ethernet, Serial, GPIB, or VXI. There are some instruments that can be installed into the computer like the general purpose DAQ devices. These devices are called computer-based instruments. [2]

The following sections discuss the communication between computers and measurement hardware.

### **4.3 How Do Computers Talk to DAQ Devices?**

Before a computer-based system can measure a physical signal, a sensor or transducer must convert the physical signal into an electrical one, such as voltage or current. The plug-in DAQ device is often considered to be the entire DAQ system, although it is actually only one system component. Unlike most stand-alone instruments, we cannot always directly connect signals to a plug-in DAQ device. In these cases, we must use accessories to condition the signals before the plug-in DAQ device converts them to digital information. The software controls the DAQ system by acquiring the raw data, analyzing the data, and presenting the results.

Figure 5 shows two options for a DAQ system. In Option A, the plug-in DAQ device resides in the computer. In Option B, the DAQ device is external. With an external board, we can build DAQ systems using computers without available plug-in slots, such as some laptops. The computer and DAQ module

communicate through various buses such as the parallel port, serial port, and Ethernet. These systems are practical for remote data acquisition and control applications.

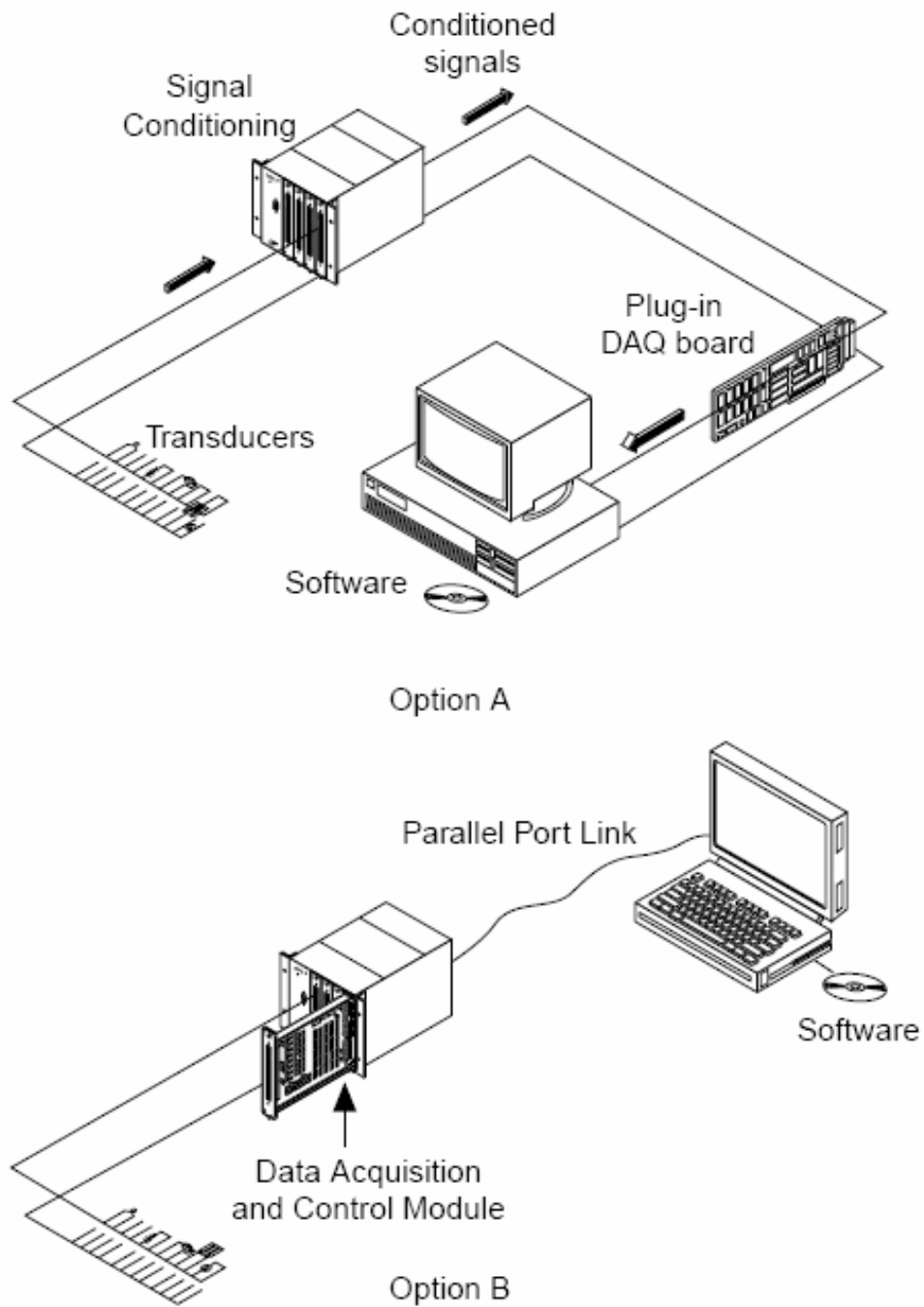


Fig. 5: DAQ System Components

A third option, not shown in Figure 5, uses the PCMCIA bus found on some laptops. A PCMCIA DAQ device plugs into the computer, and signals are connected to the board just as they are in Option A. This allows for a portable, compact DAQ system.

### **4.3.1 Role of Software**

The computer receives raw data. Software takes the raw data and presents it in a form the user can understand. Software manipulates the data so it can appear in a graph or chart or in a file for report. The software also controls the DAQ system, telling the DAQ device when to acquire data, as well as from which channels to acquire data.

Typically, DAQ software includes drivers and application software. Drivers are unique to the device or type of device and include the set of commands the device accepts. Application software (such as LabVIEW) sends the commands to the drivers, such as acquire a thermocouple reading and return the reading, then displays and analyzes the data acquired.

## **4.4 How Do Computers Talk to Special-Purpose Instruments?**

The fundamental task of an instrument is to measure some natural phenomenon. Unlike data acquisition, the signal the computer ultimately receives requires no conditioning. How the computer controls the instrument and acquires data from the instrument depends on how the instrument is built. Common types of instruments include the following [3]:

- GPIB
- Serial Port
- VXI
- PXI
- Computer-based instruments

All external instruments communicate with the computer through some type of bus where a communication protocol has been defined. The instrument has a set of commands that it understands. The user writes an application that sends commands to and receives data from an instrument. As the test system designer, we have to be concerned with the software connection with the instrument. That is, we have to understand how application and instrument communicate with each other. Additionally, we have to be concerned with the type of hardware connection to instrument.

### **4.4.1 How Do Programs Talk to Instruments?**

An instrument driver is a collection of functions that implement the commands necessary to perform the instrument's operations. LabVIEW instrument drivers simplify instrument programming to high-level commands, so we do not need to learn the low-level instrument-specific syntax needed to control instruments.

Instrument drivers are not necessary to use instrument. They are merely time savers to help us develop our project so we do not need to study the instrument manual before writing a program. Instrument drivers create the instrument commands and communicate with the instrument over the serial,

GPIB, or VXI bus. In addition, instrument drivers receive, parse, and scale the response strings from instruments into scaled data that can be used in test programs. With all of this work already done for us in the driver, instrument drivers can significantly reduce development time.

Instrument drivers can help make test programs more maintainable in the long-run because instrument drivers contain all of the I/O for an instrument within one library, separate from other code. We are protected against hardware changes and upgrades because it is much easier to upgrade test code when all of the code specific to that particular instrument is self-contained within the instrument driver. [3]

## 4.5 Types of Instruments

When we use a personal computer to automate test system, we are not limited to the type of instrument that we can control. We can mix and match instruments from various categories, such as serial, GPIB, VXI, PXI, and computer-based instruments along with instruments that are not discussed, such as image acquisition, motion control, Ethernet, SCXI, CAMAC, parallel port, CAN, FieldBus, and other devices. [3]

The things to be aware of with PC control of instrumentation are:

- What type of connector (pinouts) is on the instrument
- What kind of cable is needed (null-modem, number of pins, male/female)
- What electrical properties are involved (signal levels, grounding, cable length restrictions)
- What communication protocols are used (ASCII commands, binary commands, data format)
- What kind of software drivers are available

## 4.6 GPIB Communications

GPIB instruments offer test and manufacturing engineers the widest selection of vendors and instruments for general-purpose to specialized vertical market test applications. GPIB instruments have traditionally been used as stand-alone benchtop instruments where measurements are taken by hand. [1]

### 4.6.1 Controllers, Talkers, and Listeners

To determine which device has active control of the bus, devices are categorized as Controllers, Talkers, or Listeners and each device has a unique GPIB primary address between 0 and 30. The **Controller** defines the communication links, responds to devices requesting service, sends GPIB

commands, and passes/receives control of the bus. **Talkers** are instructed by the Controller to talk and place data on the GPIB. Only one device at a time can be addressed to talk. **Listeners** are addressed by the Controller to listen and read data from the GPIB. Several devices can be addressed to listen.[4]

#### 4.6.2 Hardware Specifications

The GPIB is a digital, 24-conductor parallel bus. It consists of eight data lines, five bus management lines (EOI, IFC, SRQ, ATN, REN), three handshake lines (DAV, NRFD, NDAC), and eight ground lines. The GPIB uses an eight-bit parallel, byte-serial, asynchronous data transfer scheme. This means that whole bytes are sequentially handshaked across the bus at a speed that the slowest participant in the transfer determines. Because the unit of data on the GPIB is a byte (eight bits), the messages transferred are frequently encoded as ASCII character strings. [12]

Additional electrical specifications allow data to be transferred across the GPIB at the maximum rate of 1MB/sec because the GPIB is a transmission line system. These specifications are [4]:

- A maximum separation of 4 m between any two devices and an average separation of 2 m over the entire bus.
- A maximum cable length of 20 m.
- A maximum of 15 devices connected to each bus with at least two-thirds of the devices powered on.

If we exceed any of these limits, we can use additional hardware to extend the bus cable lengths or expand the number of devices allowed.

Faster data rates can be obtained with HS488 devices and controllers. HS488 is an extension to GPIB and is supported by most National Instruments controllers.

# Chapter 5

## Remote laboratory using LabVIEW

### 5.1 Overview

Laboratories, which are found in all engineering and science programs, are an essential part of the education experience. Not only do laboratories demonstrate course concepts and ideas, but they also bring the course theory alive so students can see how unexpected events and natural phenomena affect real-world measurements and control algorithms. However, equipping a laboratory is a major expense and its maintenance can be difficult. Teaching assistants are required to set up the laboratory, instruct in the laboratory, and grade laboratory reports. These time-consuming and costly tasks result in relatively low laboratory equipment usage, especially considering that laboratories are available only when equipment and teaching assistants are both available.

What if some of the basic laboratory experiments could be made available 24 hours a day, seven days a week? What if students could have access to experiments from their home or student dormitories? What if a professor wanted his students to take a closer look at a classroom demonstration? What if a professor wanted to make a research demonstration available to students and others at irregular times? What if a professor deriving a complex equation for some application wanted his students to try different sets of parameters to bring out the essence of the model? What if a professor wanted his students to see an electronic circuit in action, and even give them control of the operating parameters? What if a research team wanted to make their expensive research equipment available to others when they went home for the evening?

All of these and many more exciting applications are easily achievable with the new technology available with National Instruments LabVIEW Remote Panels. With this standard feature of LabVIEW, a user can quickly and effortlessly publish the front panel of a LabVIEW program for use in a standard Web browser. Once published, anyone on the Web with the proper permissions can access and control the experiment from the local server. If the LabVIEW program controls a real-world experiment, demonstration, calculation, etc., LabVIEW Remote Panels turns the application into a remote laboratory with no additional programming or development time.

### 5.2 Distance-Learning Remote Laboratories

A remote laboratory is defined as a computer-controlled laboratory that can be accessed and controlled externally over some communication medium. For this discussion, a remote laboratory is an experiment, demonstration, or

process running locally on a LabVIEW platform but with the ability to be monitored and controlled over the Internet from within a Web browser. In the simplest case, the remote laboratory server can be an experiment connected to a computer through a standard interface (DAQ, GPIB, serial, parallel, etc) and with the host computer connected to the Internet. The client can be any computer connected to the Internet running a simple browser. Once connected, the client will see the same front panel as the local host and also have the same program functionality.



Fig. 6: Internet Control of a Remote Laboratory

## 5.3 Remote Laboratory in world

Reader can find a lot of examples of remote laboratory using LabVIEW in the world.

### 5.3.1 Stanford University – Cyberlab

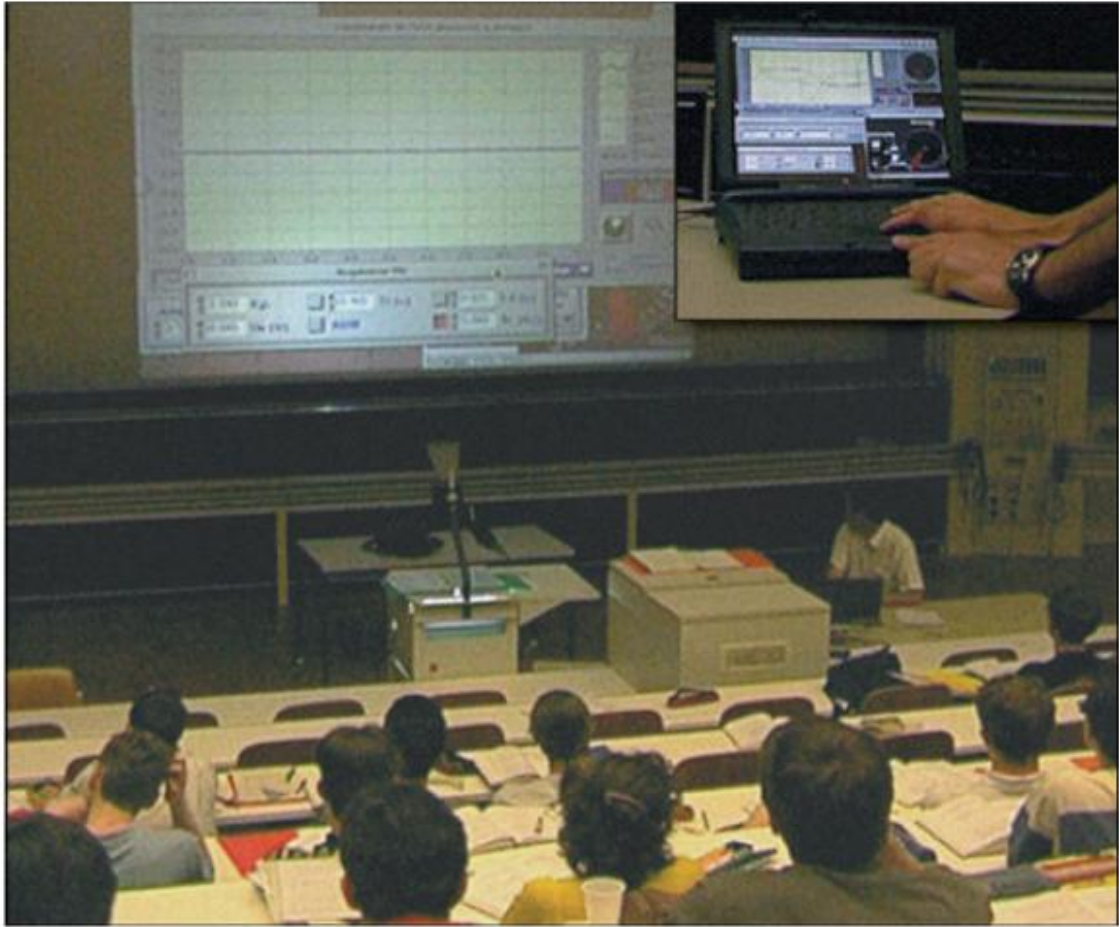
At Stanford University, students can log onto a remote optics laboratory to conduct an experiment to measure the physical properties of a laser diode. Cyberlab provides not only monitoring and control features, but also a laboratory scheduler, reference library and analysis tools. The NI LabVIEW Internet Toolkit is used to create CGI scripts for communication and control between the remote browser and the host server. The live embedded images are provided using NI-IMAQ software tools and a NI PCI-1408 image acquisition board. [8]



Fig. 7: Stanford University – Cyberlab

### 5.3.2 Swiss Federal Institute of Technology – Lecture Enhancement

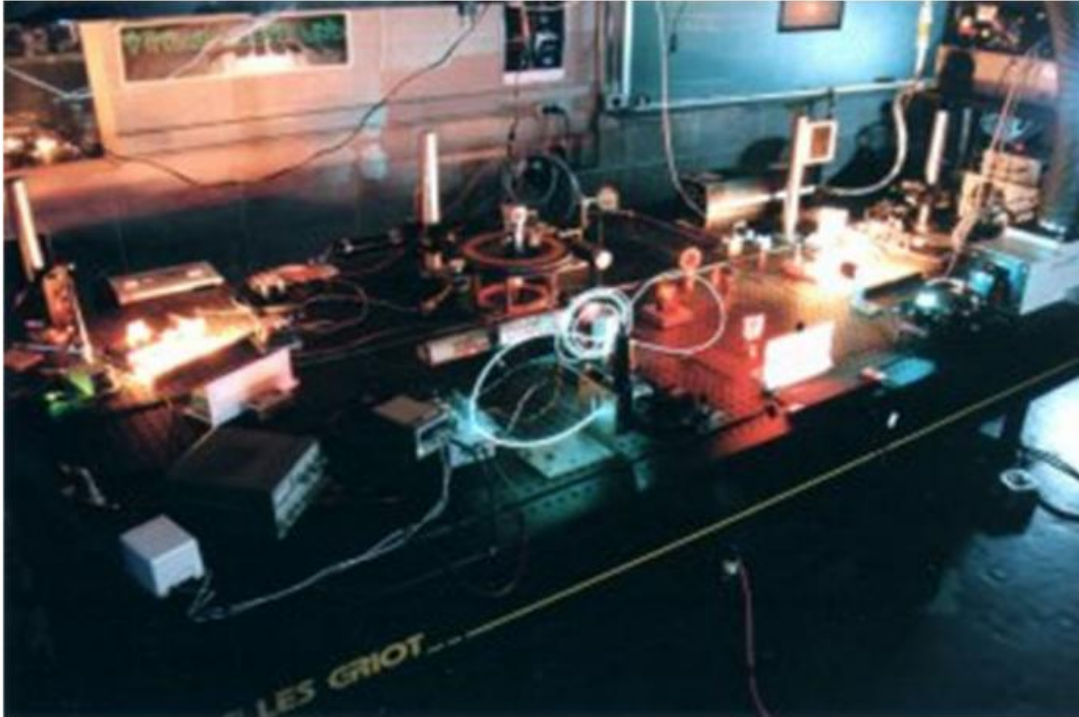
At the Swiss Federal Institute of Technology, mechanical engineering students and others can watch in the classroom as the professor uses his laptop computer as a client to control a remote mechanical system. At other times, students take control from home to study the classroom demonstration in detail. Control within LabVIEW is provided by "Call-By-Reference" of sub-VIs. Visual feedback is provided with video conferencing software. Both the video and LabVIEW servers run in parallel, independent of the local host computer.



*Fig. 8: Swiss Federal Institute of Technology – Lecture Enhancement*

### **5.3.3 Dalhousie University – Virtual Laser Laboratory**

At Dalhousie University, engineering and science students can log onto the Virtual Laser Laboratory 24 hours a day, seven days a week and conduct up to 10 laser experiments. There are four lasers, three stepping motors, two DC motors, numerous measuring points and three video cameras. One computer controls all of the instruments and experiments on a 4 by 8 optics table. A second computer controls the pan, tilt, and zoom of the main camera and provides jpg images of the table action. A third computer integrates all the images and control parameters to produce the exported pages. Each of the experiments has a unique presentation and set of control buttons. At login, a Java script is downloaded to the remote browser that interprets all the CGI commands sent over the network and provides interactivity through the browser.



*Fig. 9: Dalhousie University – Virtual Laser Laboratory*

## **5.4 LabVIEW Remote Panels**

In all the previous successes of remote laboratories, LabVIEW was used but extensive programming of Java, CGI, or other third-party software tools was required to bring local laboratory functionality to a browser environment. Now, with LabVIEW Remote Panels, remote execution is just a couple of clicks away. Without any additional programming, a LabVIEW program can be enabled for remote control through a common Web browser. With this new technology, the user simply points the Web browser to the Web page associated with the application. Then, the user interface for the application shows up in the Web browser and is fully accessible by the remote user.

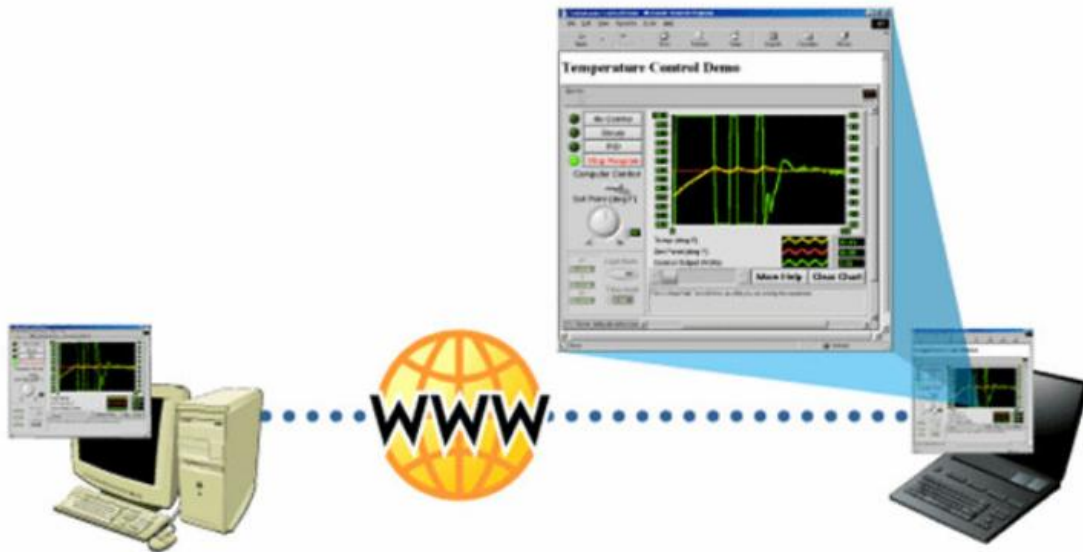


Fig. 10: LabVIEW Remote Panels

The acquisition is still occurring on the host computer, but the remote user has total control and identical application functionality. Other users can also point their Web browser to the same URL to monitor the application in progress. To reduce confusion, only one client can control the application at a time, but the client can pass control easily among the various clients at run-time. At any time during this process, the operator of the host machine can assume control of the application back from the client currently in control.

## 5.5 Enabling Remote Panel

Transforming application into a distance-learning remote laboratory is very easy. Enabling the Remote Panels feature of LabVIEW is a very simple process that walks the user through the creation of a Web page that automatically embeds the appropriate LabVIEW application into the new Web Page.

To transform application into a remote laboratory, make sure the VI that we want to publish is loaded into LabVIEW memory. Next, select the **Web Publishing Tool** option from the Tools menu. This window is the main window for interactively creating and publishing remote laboratory.

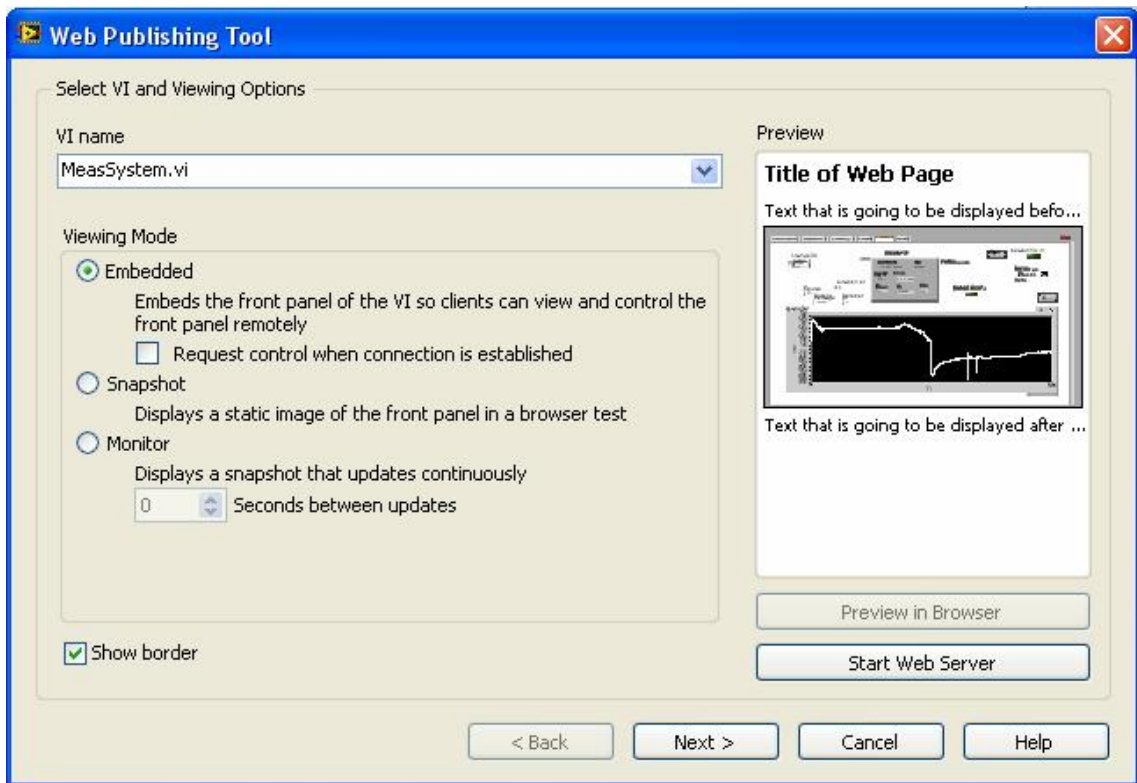


Fig. 11: Enabling Remote Panels I

In web publishing tool we can choose VI, which will be a remote laboratory as we can see on figure 11. Other options to choose is **Viewing mode**, it is possible to elect three mode:

- Embedded – Embeds the front panel of the VI, so clients can view and control the front panel remotely
- Snapshot – Displays a static image of the front panel in a browser test
- Monitor – Displays a snapshot that updates continuously

To another step is need to click on button **Next**.

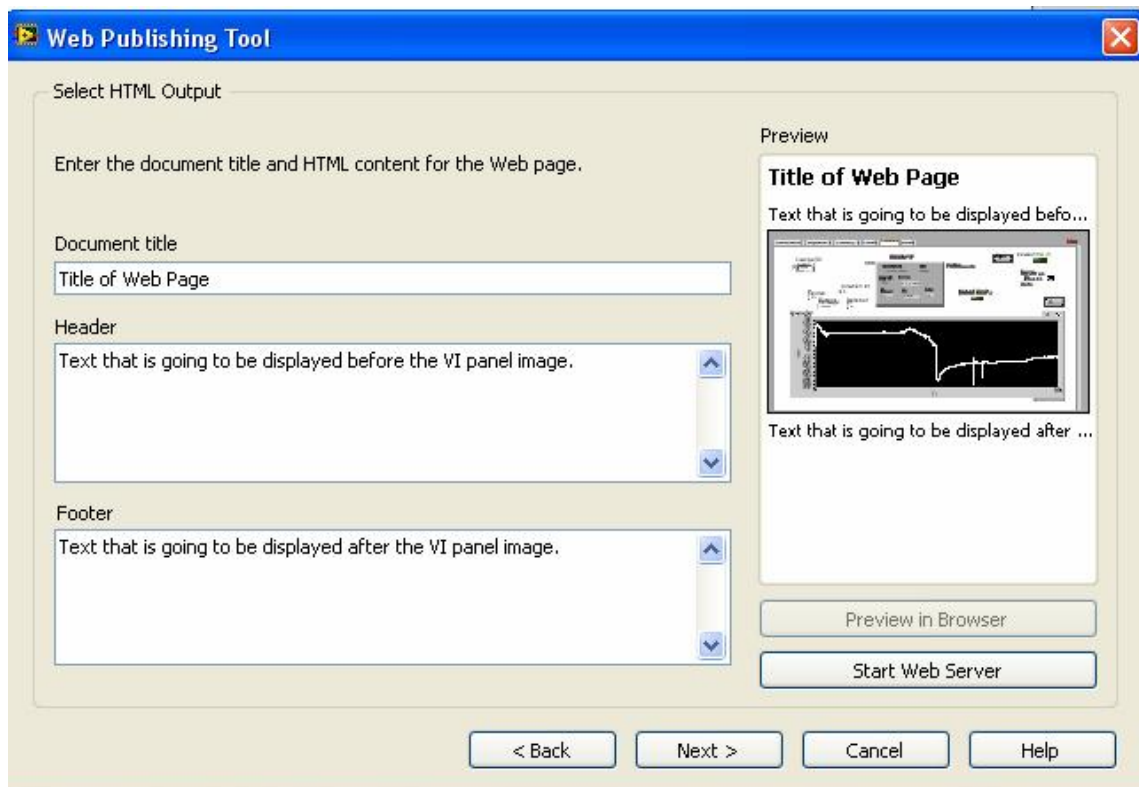


Fig. 12: Enabling Remote Panels II

In figure 12 we can see a next window of web publishing tool, there is a preview of web page. Document Title, Header, and Footer are all text fields that we can use to customize the Web page created with the publishing tool.

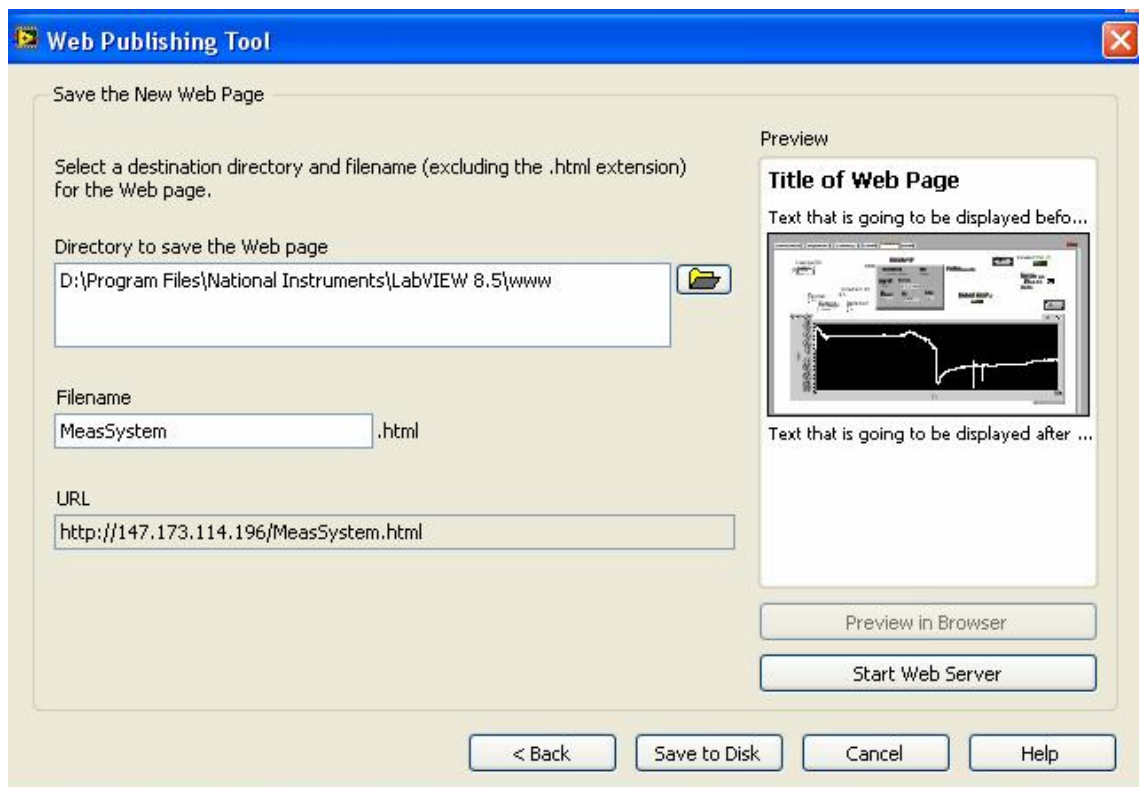
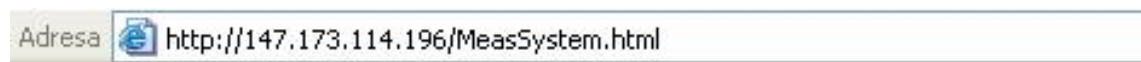


Fig. 13: Enabling Remote Panels III

To another step is need to click the on **Next** button. Clicking on **Save to Disk** places an HTML file called Document Title.html into the LabVIEW file folder called WWW by default. Saving Remote Panels HTML documents into this folder will ensure that the LabVIEW Web server can find them. Either keep the default name, or assign a new name and save the file. Last item in this windows is message box containing the URL address of enabled LabVIEW application.

The last step necessary to enable a remote laboratory is to select the **Start Web Server** button. When pressed, this button activates the built-in LabVIEW Web server, which will publish and control front panel images from the Internet. Lab is now ready for remote visitors.



## Title of Web Page

Text that is going to be displayed before the VI panel image.

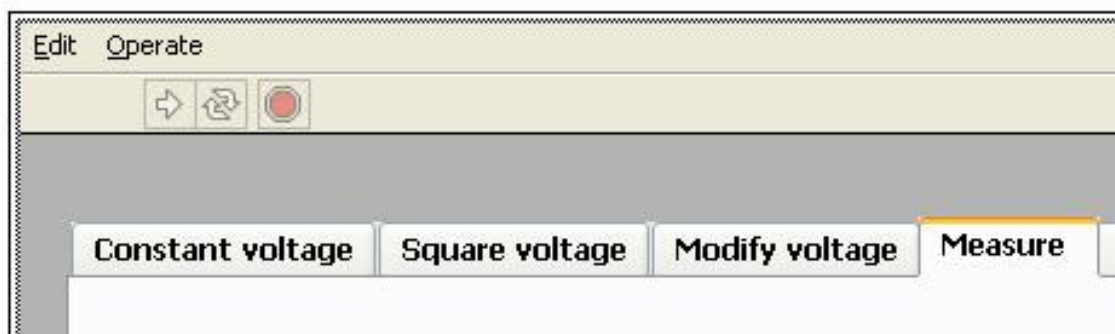


Fig. 14: Remote laboratory

In figure 14 we can see remote connection to VI, which is final result of the piervous text.

## 5.6 Client Operation

### 5.6.1 Required Software

To operate a LabVIEW program using remote panels, it is necessary to have the free LabVIEW run-time engine installed on the client computer. When a remote viewer logs onto the lab with the appropriate URL address, the LabVIEW front panel will appear in the browser, or reroute the user to install the run-time engine from the National Instruments Web site.

### 5.6.2 Application Control

Once connected to the remote laboratory, the client connection will automatically be in a monitor state. If another client is controlling the remote laboratory, the user will be able to monitor the actions of the controlling client.

To request control of the program, right click on the front panel and select Request Control. Once selected, one of two possible messages will appear. Either the user will be granted control (Control Granted), or the user will see a message indicating that control is currently granted to another user (Waiting for control: Either the server is locked or another client has control). If another client has control, the controlling client will be notified that control time has now become limited. Once the timeout occurs or the controlling client has released control, application control is automatically switched to the requesting client (Control Granted). Once the user has been granted control, all icons and controls will become active and running the LabVIEW application is exactly like running the application from the local environment.

### **5.6.3 Releasing Control**

When the remote viewer either moves on to a different URL address or relinquishes control by right clicking and selecting (Release Control), or when the remote laboratory times out, the remote laboratory is available to the next visitor.

## **5.7 Application Administration**

National Instruments LabVIEW Remote Panels comes with all the administrative tools necessary for a complete remote laboratory solution. A variety of tools are available for monitoring and logging of network traffic, Remote Panels license management, configuration of remotely accessible VIs, and configuration of Remote Panel control time limits.

### **5.7.1 Remote Panel Connection Manager**

The tool used to administer Remote Panels is the Remote Panel Connection Manager. With this tool, we can manage client traffic to a specific front panel. This window logs all network traffic and displays a graph with network throughput for all visible VIs and specific VIs. The Connection Manager also gives you the ability to disconnect a specific client with the touch of a button. In figure 15 we can see remote panel connection manager with two connected users.

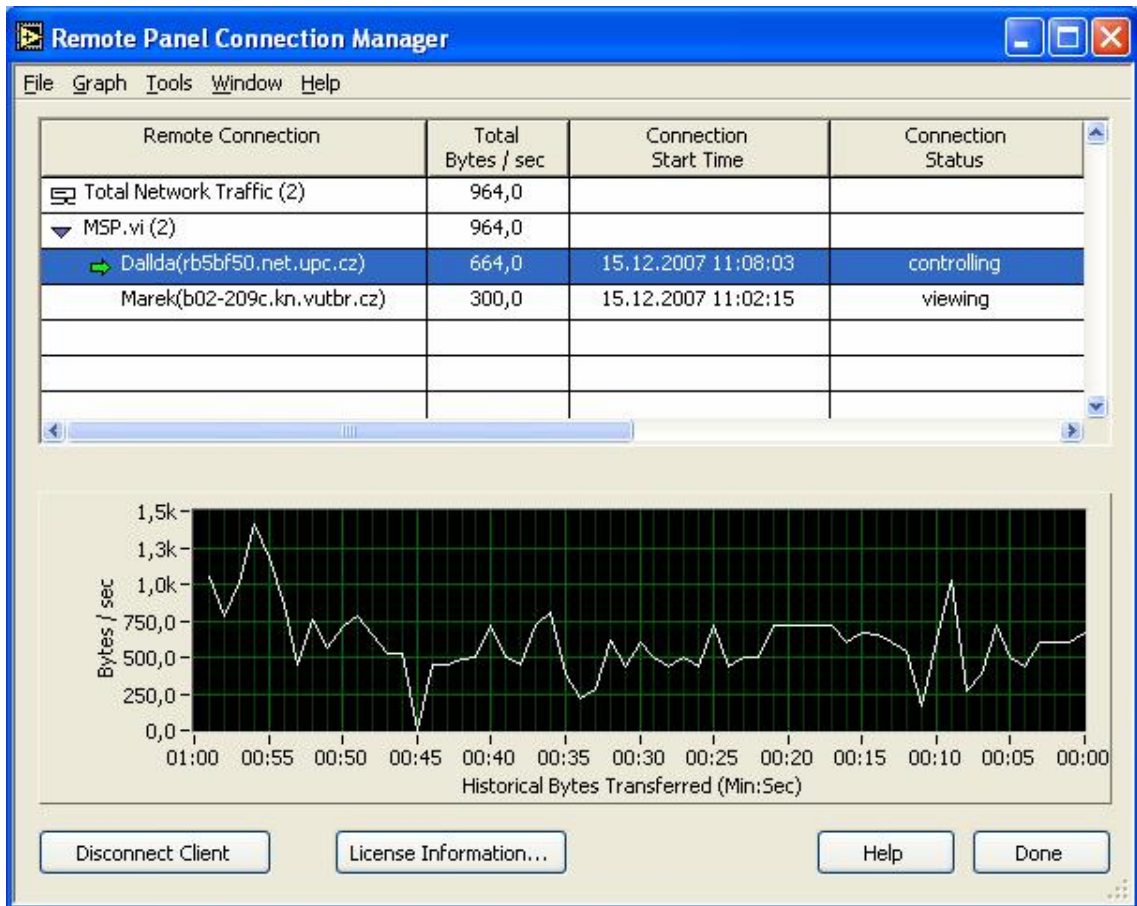


Fig. 15: Remote Panel Connection Manager

## 5.7.2 Web Server: Configuration

Another very important administrative tool is the **Web Server: Configuration** tool located within the Tools: Options menu in LabVIEW. We can use this tool to enable or disable the LabVIEW Web Server, specify the port to use for HTTP (default 80), enable or disable the log file, set the Web Server timeout (default 60 s), and even specify the LabVIEW Web Server root directory (Figure 16).

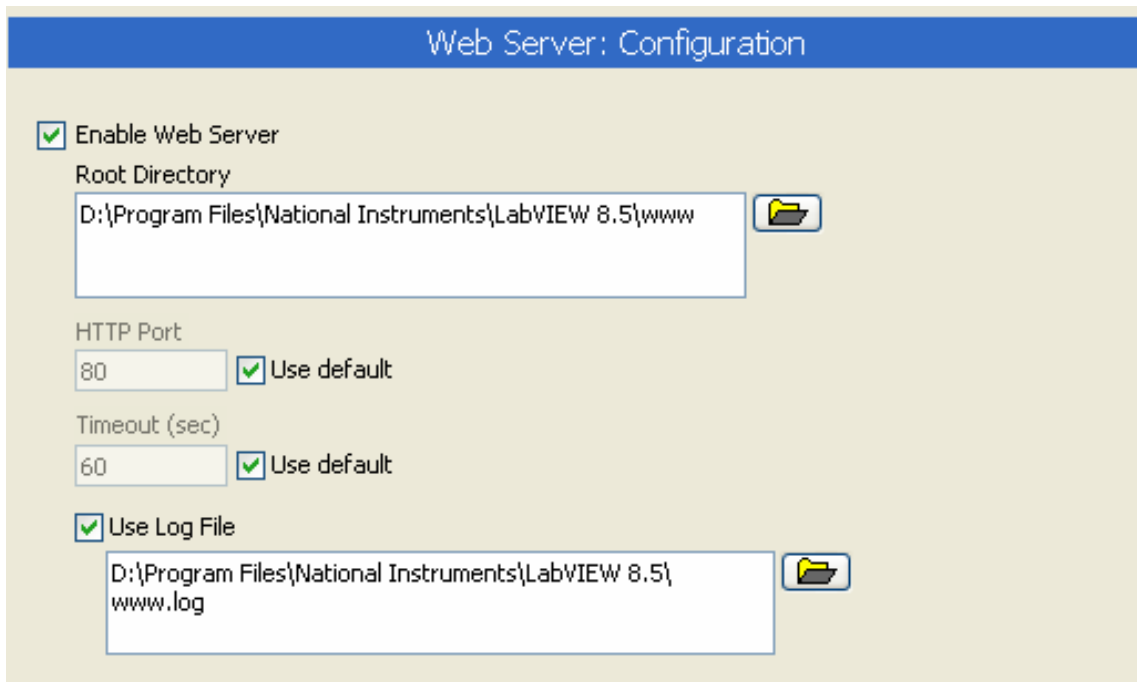


Fig. 16: Web Server: Configuration

### 5.7.3 Application Security

National Instruments LabVIEW Remote Panels has a very simple and powerful security tool. The **Web Server: Browser Access tool**, which is located within the Tools: Options menu in LabVIEW, can configure browser access to one of the following three options:

- Allow Viewing and Controlling
- Allow Viewing
- Deny Access

These three options can be applied to a specific computer name or IP address thus providing substantial security enhancements to our remote laboratory.

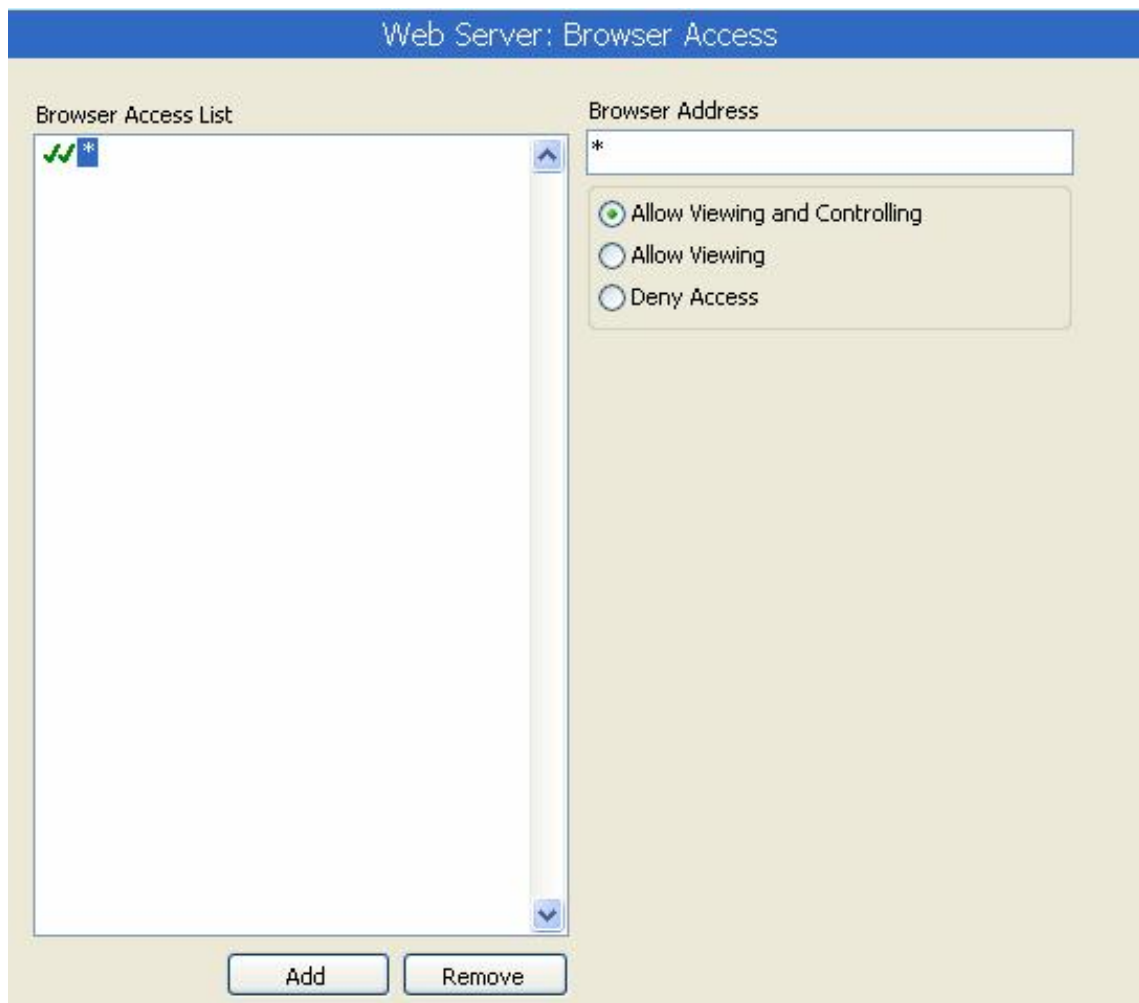


Fig. 17: Web Server: Browser Access

Because LabVIEW is running our remote laboratory, we could easily create and implement a higher level security database application for our remote laboratory. A VI could easily be created and implemented to prompt the student for a user name and password. That VI could look up access times in a database and determine whether to allow the student access to the remote laboratory. This example shows what is possible using LabVIEW and Remote Panels to create remote laboratory.

# Chapter 6

## Keithley

### 6.1 Electrometer/High Resistance Meter

Keithley's 5 $\frac{1}{2}$  digit Model 6517A Electrometer/High Resistance Meter offers accuracy and sensitivity specifications unmatched by any other meter of this type. It also offers a variety of features that simplify measuring high resistances and the resistivity of insulating materials. With reading rates of up to 125 readings/second, the Model 6517A is also significantly faster than competitive electrometers, so it offers a quick, easy way to measure low-level currents.



Fig. 18: Keithley 6517A

### 6.2 Exceptional Performance Specifications

The half-rack-sized Model 6517A has a special low current input amplifier with an input bias current of  $<3\text{fA}$  with just  $0,75\text{fA}$  p-p (peak-to-peak) noise and  $<20\mu\text{V}$  burden voltage on the lowest range. The input impedance for voltage and resistance measurements is  $200\text{T}\Omega$  for nearideal circuit loading. These specifications ensure the accuracy and sensitivity needed for accurate low current and high impedance voltage, resistance, and charge measurements in areas of research such as physics, optics, nanotechnology, and materials science. A built-in  $\pm 1\text{kV}$  voltage source with sweep capability simplifies performing leakage, breakdown, and resistance testing, as well as volume ( $\Omega\text{-cm}$ ) and surface resistivity ( $\Omega/\text{square}$ ) measurements on insulating materials. [15]

## 6.2.1 Wide Measurement Ranges

The Model 6517A offers full autoranging over the full span of ranges on current, resistance, voltage, and charge measurements:

- Current measurements from 1fA to 20mA
- Voltage measurements from 10 $\mu$ V to 200V
- Resistance measurements from 50 $\Omega$  to 10<sup>16</sup> $\Omega$
- Charge measurements from 10fC to 2 $\mu$ C

## 6.3 Why this one?

Because the Model 6517A is well suited for low current in areas of research such as physics materials science. Its extremely low voltage burden makes it particularly appropriate for use in solar cell applications, and its built-in voltage source and low current sensitivity make it an excellent solution for high resistance measurements of nanomaterials such as polymer based nanowires. Its high speed and ease of use also make it an excellent choice for quality control, product engineering, and production test applications involving leakage, breakdown, and resistance testing. Volume and surface resistivity measurements on non-conductive materials are particularly enhanced by the Model 6517A's voltage reversal method.

## 6.4 IEEE- 488 Reference

The IEEE-488 is an instrumentation data bus with hardware and programming standards originally adopted by the IEEE (Institute of Electrical and Electronic Engineers) in 1975 and given the IEEE-488 designation. In 1978 and 1987, the standards were upgraded to IEEE-488-1978 and IEEE-488.1-1987, respectively. The Model 6517A conforms to these standards.

The Model 6517A also conforms to the IEEE-488.2-1987 standard and the SCPI 1991 (Standard Commands for Programmable Instruments) standard. IEEE-488.2 defines a syntax for sending data to and from instruments, how an instrument interprets this data, what registers should exist to record the state of the instrument, and a group of common commands. The SCPI standard defines a command language protocol. It goes one step farther than IEEE-488.2 and defines a standard set of commands to control every programmable aspect of an instrument. [1]

### 6.4.1 IEEE- 488 bus connections

The Model 6517 can be connected to the IEEE-488 bus through a cable equipped with standard IEEE-488 connectors, an example is shown in figure 19. The connector can be stacked to allow a number parallel connections to one

instrument. Two screws are located on each connector to ensure that connections remain secure. Current standards call for metric threads, which are identified with dark colored screws. Earlier versions had different screws, which were silver colored.

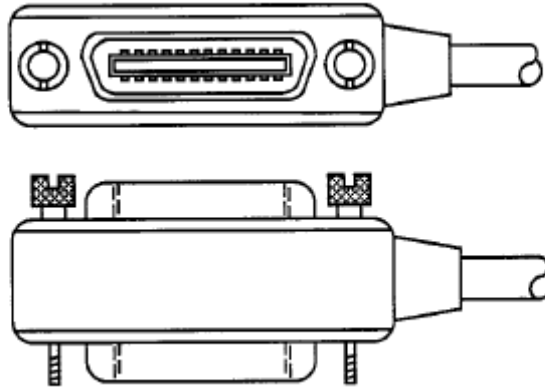


Fig. 19: IEEE-488 connector

## 6.5 Trigger Model (IEEE-488 operation)

The following information describes the operation process of the Model 6517A over the IEEE-488 bus. The flowchart in figure 20, which summarizes operation over the bus, is called the Trigger model. It is called the trigger model because operation is controlled by SCPI commands from the Trigger subsystem.

### 6.5.1 Idle and initiate

The instrument is considered to be in the idle state whenever it is not operating within one of the layers of the trigger model. The front panel ARM indicator is off when the instrument is in the idle state. While in the idle state, the instrument cannot perform any measure or scan functions. Over the bus, there are two SCPI commands that can be used to take the instrument out of the idle state: `:INITiate` or `:INITiate:CONTinuous ON`. Notice that with continuous initiation enabled (`:INIT:CONT ON`), the instrument will not remain in the idle state after all programmed operations are completed. However, the instrument can be returned to the idle state at any time by sending the `*RST` command, the `*RCL` command, or the `SYST:PRES` command.

### 6.5.2 Trigger model layers

As can be seen in figure 20, the trigger model uses three layers; Arm Layer 1, Arm Layer 2 and the Trigger Layer. For front panel operation, these layers are known as the Arm Layer, Scan Layer and Measure Layer. Once the Model 6517A is taken out of the idle state, operation proceeds through the layers of the trigger model down to the device action where a measurement occurs.

**Control sources** - In general, each layer contains a control source which holds up operation until the programmed event occurs. The control sources are summarized as follows[15]:

- **IMMediate** -With this control source selected, event detection is immediately satisfied allowing operation to continue.
- **MANual** -Event detection is satisfied by pressing the TRIG key. Note that the Model 6517A must be taken out of remote before it will respond to the TRIG key. Press LOCAL or send LOCAL 16 over the bus to take the instrument out of remote.
- **BUS** - Event detection is satisfied when a bus trigger (GET or \*TRG) is received by the Model 6517A.
- **TIMer** - Event detection is immediately satisfied on the initial pass through the layer. Each subsequent detection is satisfied when the programmed timer interval (1 to 999999,999 seconds) elapses. A timer resets to its initial state when operation loops back to a higher layer (or idle). Note that TIMer is not available in Arm Layer 1.
- **EXTernal** -Event detection is satisfied when an input trigger via the EXTERNAL TRIGGER connector is received by the Model 6517A.
- **TLINK** - Event detection is satisfied when an input trigger via the TRIGGER LINK is received by the Model 6517A.
- **RTClock** - Event detection is satisfied when the programmed time and date occurs. Note that the real-time clock is only available as a control source for Arm Layer 1.
- **HOLD** - With this selection, event detection is not satisfied by any of the above control source events and operation is held up.

**Control source loops** - As can be seen in the flowchart. Each layer has three paths that allow operation to loop around the control source. These three paths are described as follows [15]:

- **:DIRection (Source Bypass)** -When a source bypass is enabled (:DIRection SOURce) and the EXTernal or TLINK control source is selected, operation will loop around the control source on the initial pass through the layer. If programmed for another event detection in the layer, the bypass loop will not be in effect even though it is still enabled. The bypass loop resets (be in effect) if operation loops back to a higher layer (or idle).

In Arm Layer 1 and Arm Layer 2, enabling a source bypass also enables the respective output trigger. In the Trigger Layer, its output trigger is always enabled and occurs after every device action.

- **:IMMediate** - Each time an :IMMediate command is sent, operation loops around the respective control source. It is used when we do not wish to wait for the programmed event to occur (or when the HOLD control source is selected). Note that in Arm Layer 1 and the Trigger Layer, :IMMediate also loops operation around the delays.
- **:SIGNal** -Same function as an :IMMediate command.

**Delays** - Arm Layer 2 and the Trigger Layer have a programmable Delay (0 to 999999,999 seconds) that is asserted after an event detection. We can't change it during a measure.

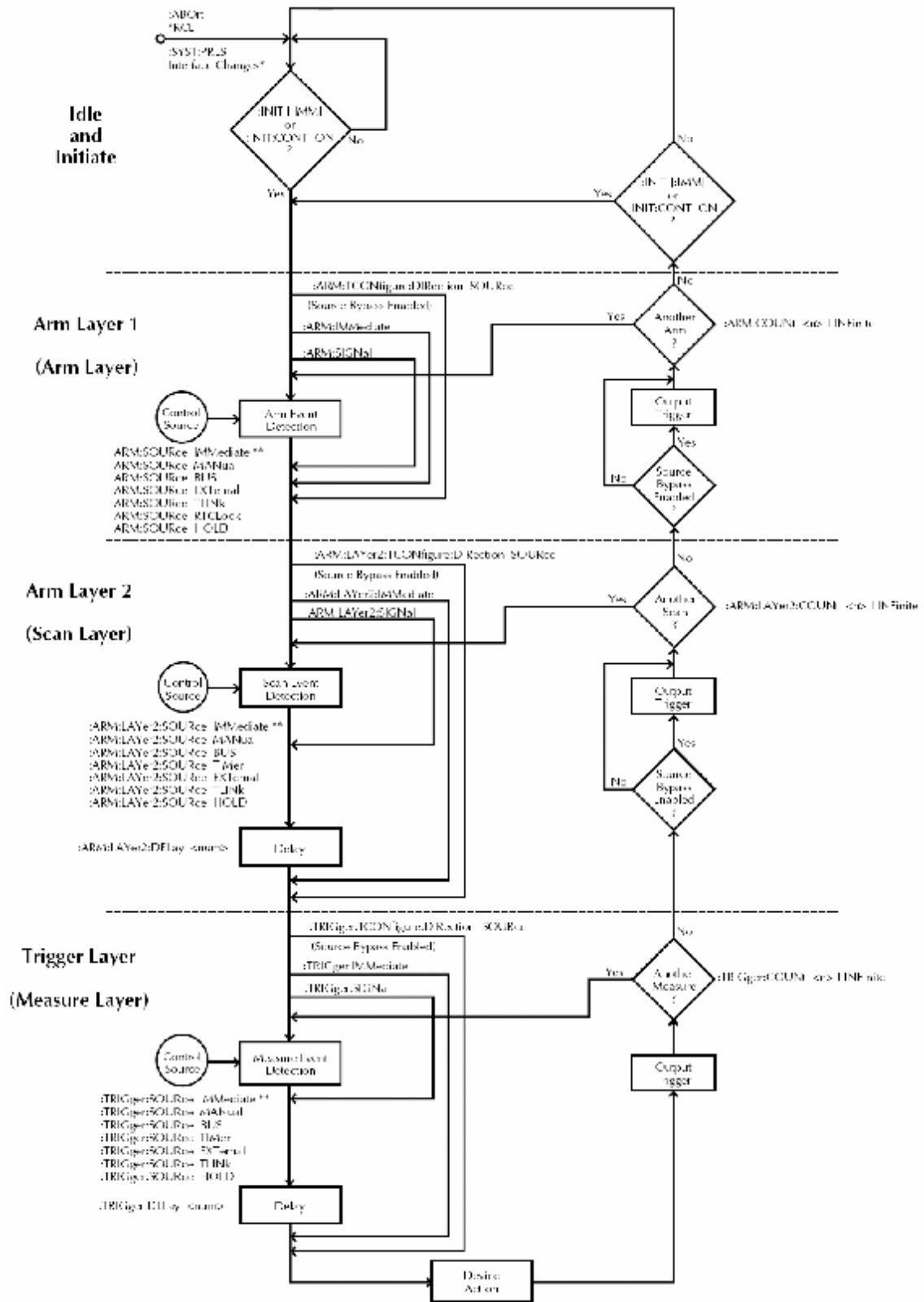


Fig. 20: Trigger model

**Device Action** - The primary device action is a measurement. However, the device action could include a function change and a channel scan (if scanner is enabled). A channel is scanned (closed) before a measurement is made. When scanning internal channels, the previous channel opens and the next channel closes (break-before-make). Also included in the device action is the internal settling time delay for the relay.

**Output Triggers** - In Arm Layers 1 and 2, the output triggers are enabled only if their respective source bypasses are also enabled. If a TLINK control source is selected, the output trigger pulse is available on the selected TRIGGER LINK output line. For all other control source selections, the trigger pulse is available at the METER COMPLETE connector.

In the Trigger Layer, the output trigger is always enabled and occurs after every device action. If the control source is set for EXTeMal, IMMEDIATE, MANual, BUS or TIMer, the output trigger pulse is available at the METER COMPLETE connector. If the TLINK control source is selected, output trigger action occurs on the selected TRIGGER LINK output line as follows [15]:

- If the asynchronous Trigger Link mode is selected, the output trigger pulse is available on the programmed output line.
- If the semi-synchronous Trigger Link mode is selected and the source bypass is disabled (:trig:tcon:dir acc), the Trigger Link line is released (goes high).
- If the semi-synchronous Trigger Link mode is selected and the Source Bypass is enabled (:trig:tcon:dir sour), the Trigger Link line is pulled down low and then released.

**Counters** - All three layers use programmable counters which allow operation to return to or stay in the respective layer. For example, programming the Trigger Layer counter for infinity (:trig:coun inf) keeps operation in the Trigger Layer. After each device action and subsequent output trigger, operation loops back to the Trigger Layer control source. A counter resets when operation loops back to a higher layer (or idle).

## 6.6 Command words

The following information covers syntax for common SCPI commands. For information not covered here, refer to the SCPI standards. Program messages are made up of one or more command words.

### 6.6.1 Commands and command parameters:

Common commands and SCPI commands may or may not use a parameter. Examples:

*SAV <NRf>	Parameter (NRf) required.
*RST	No parameter used.
:INITiate:CONTinuous <b>	Parameter (<b>) required.

:SYSTem:PRESet

No parameter used.

Note that there must be at least one space between the command word and the parameter.

**Brackets [ ]**: There are command words that are enclosed in brackets ([ ]). These brackets are used to denote an optional command word that does not need to be included in the program message. For example:  
:INITiate[:IMMEDIATE]

The brackets indicate that :IMMEDIATE is implied (optional) and does not have to be used. Thus, the above command can be sent in one of two ways:

:INITiate or :INITiate:IMMEDIATE

Notice that the optional command is used without the brackets.

**Parameter types:** Some of the more common parameter types are explained as follows [15]:

- **<b> Boolean:** Used to enable or disable an instrument operation. 0 or OFF disables the operation, and 1 or ON enables the operation. Example:  
:CURRent:DC:RANGe:AUTO ON            Enable auto-ranging.

- **<name> Name parameter:** Select a parameter name from a listed group. Example:

<name> = NEVer  
          = NEXt  
          = ALWays  
          = PRETrigger

:TRACe:FEED:CONTrol PRETrigger

- **<NRf> Numeric representation format:** This parameter is a number that can be expressed as an integer (e.g., 8), a real number (e.g., 23.6) or an exponent (2.3E6). Example:

:STATus:MEASurement:ENABLE Set bit B2 of enable register

- **<n> Numeric value:** A numeric value parameter can consist of an NRf number or one of the following name parameters: DEFault, MINimum or MAXimum. When the DEFault parameter is used, the instrument is programmed to the \*RST default value. When the MINimum parameter is used, the instrument is programmed to the lowest allowable value. When the MAXimum parameter is used, the instrument is programmed to the largest allowable value.

Examples:

:TRIGger:TIMer 0.1	Sets timer to 100msec.
:TRIGger:TIMer DEFault	Sets timer to 0.1 sec.
:TRIGger:TIMer MINimum	Sets timer to 1m sec.
:TRIGger:TIMer MAXimum	Sets timer to 999999.999sec.

- **<list> List:** Specify one or more switching channels. Examples:  
:ROUte:SCAN (@1:10) Specify scan list (1 through 10).  
:ROUte:SCAN (@ 2,4,6) Specify scan list (2,4 and 6).

### 6.6.2 Case sensitivity

Common commands and SCPI commands are not case sensitive. We can use upper or lower case, and any case combination. Examples:

```
*RST = *rst
:SCAN? = :scan?
:SYSTem:PRESet = :system:preset
```

### 6.6.3 Long-form and short-form versions

A SCPI command word can be sent in its long-form or short-form version. The command subsystem tables in this section provide the commands in the long-form version. However, the short-form version is indicated by upper case characters. Examples:

:SYSTem:PRESet	Long-form
:SYSTPRES	Short-form
:SYSTem:PRES	Long and short-form combination

Note that each command word must be in long-form or short-form. and not something in between. For example, :SYSTe:PRESe is illegal and will generate an error. The command will not be executed.

# Chapter 7

## Software

### 7.1 Introduction

In introduction I've made an analyse of a submission and I've explained a direction of the thesis. There was created a program in which scientists in CNRS, institution LEMD search characteristics of dielectrical materials. In my opinion, it isn't necessary to analyse a whole source code. It would make the theses unclear. A main VI has been made and next 18 subVI have been made in it. A whole is made as a large and interconnection system. If you have ever made something in a development system LabVIEW or at least have read my semestral theses you know, that resulting programs are large. During creating this system it was necessary to solve much more complicated operations than creation of a matrix. That's why in text source code isn't mentioned. For its scheme we would need at least one tree. Farther in thesis the system will be describet in a user's way. How it was mentioned, the creation of remote laboratory in labview takes tens of minutes. It's why the farther text will deal rather with the program itself than its presentation in web. In the end this program will be used as a remote laboratory too, what hasn't been planned on. This remote laboratory will be available only for scientists from CNRS. A user can choose from three modes of course of tension according to his or her needs. Next he or she has two possibilities how to measure it. And the last possibility of program is the analyse of measuring data.

## 7.2 Describe of Front panel

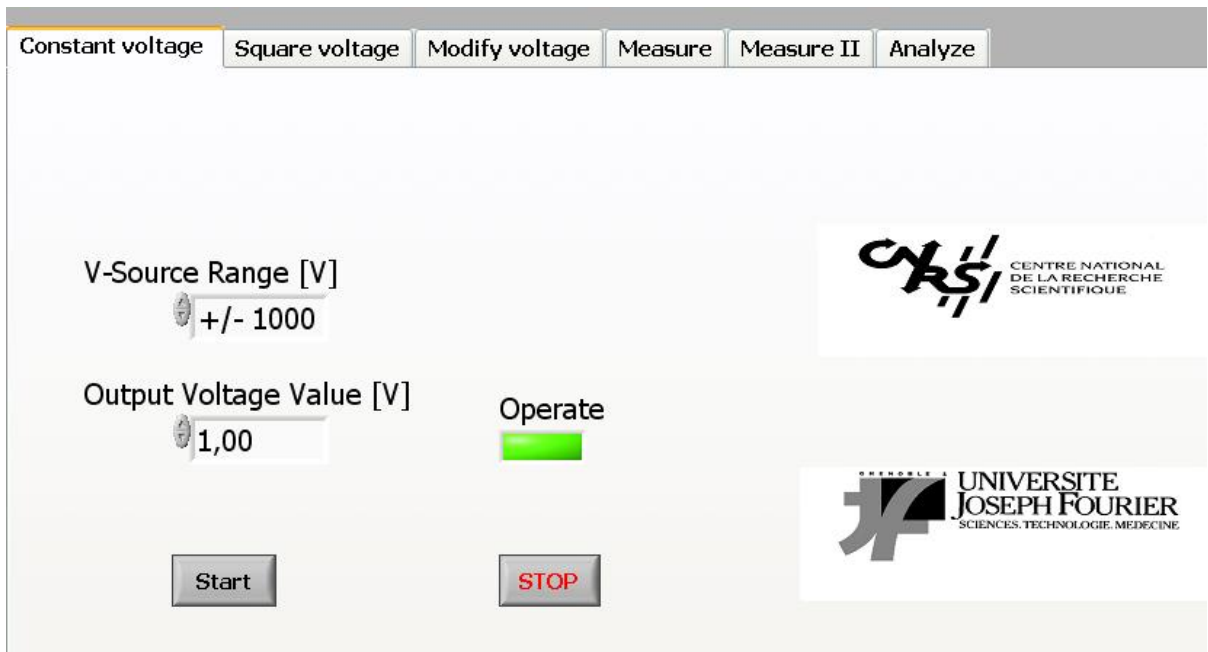


Fig. 21: Constant Voltage

In a figure 21 we can see the first fold, which appears with a program start. The user has got several possibilities here. He can choose parameters of constant voltage, that means value of voltage during measurement doesn't change. The user chooses a range of voltage, either +/- 100V or +/- 1000V. Next he enters concrete value of voltage. If user chooses the range +/- 100V and enters value of voltage e.g. 500V, the program is made in a such way so as instrument wouldn't be damaged. It means that in that case on terminals VSource appears voltage 100V, which is the last in allowed range. Then he presses START, immediately led with sign (name) OPERATE that means that source of constant voltage is active. However, if the user has a look at a measuring instrument Keithley, he can see, that any voltage doesn't appear at an output. And this is correct, because program doesn't realize the order until user runs the measurement itself. Reasons are written in a chapter 2. Creation of this running of voltage is the easiest and instrument Keithley is determined to it.

Since this source code is the least large we can present it, we can see it in the figure 22. All orders are closed in CASE structure and they will be realized as soon as the measurement is run. This way is entered an occurrence which it is waited for. Then with PICK LINE and RING CONTROL is chosen a range and is changed into string together with an input constant, what is an command for setting up of a range. Farther we can see next structure CASE, which is realized if range overflow isn't active. Next command activates Vsource. Whole this segment of the command we send to the measuring instrument with subVI.

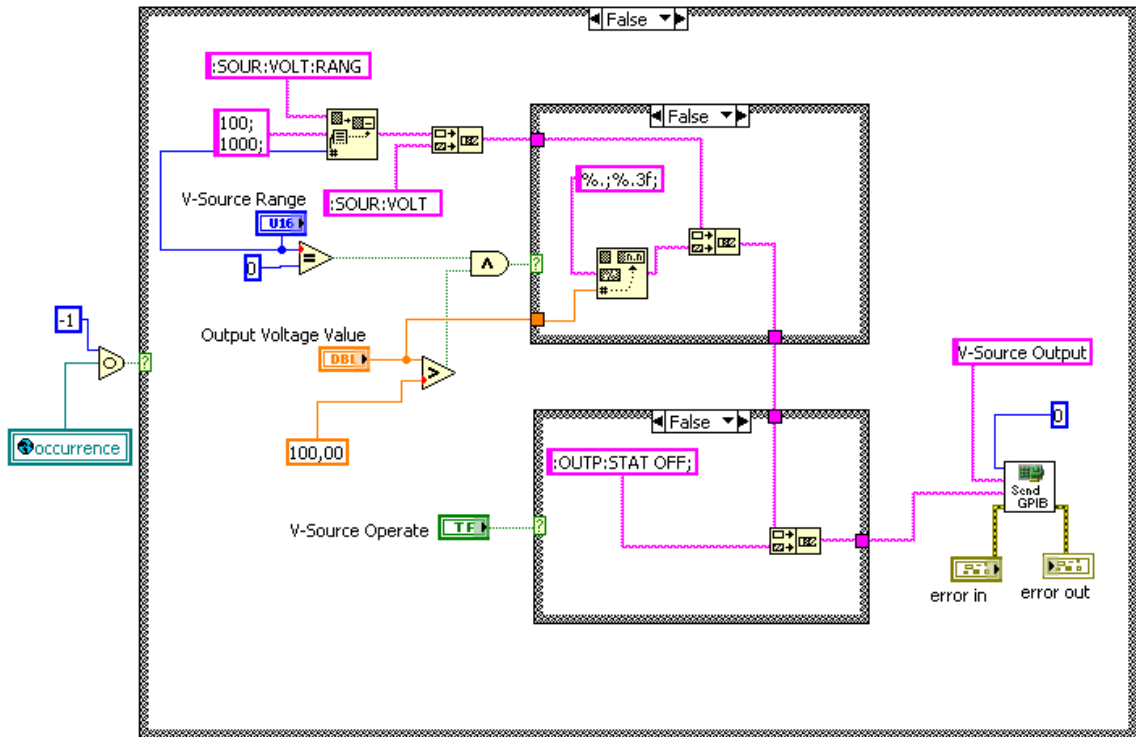


Fig. 22: Vsource.vi block diagram

In next fold, figure 23, user can choose parametres of a square wave, this fold sets up values SquareGen.vi. We can set up values of amplitude, range of voltage, frequence and DC offset, next we press START and then, like in the fold **CONSTANT VOLTAGE**, led OPERATE lights up but waits for user's running the measurement. At generating square wave signal we set some limiting properties, those of them, which are stated by a producer you can find in [15]. Many testing measurements were made to find out properties of whole system. Now we can see some results.

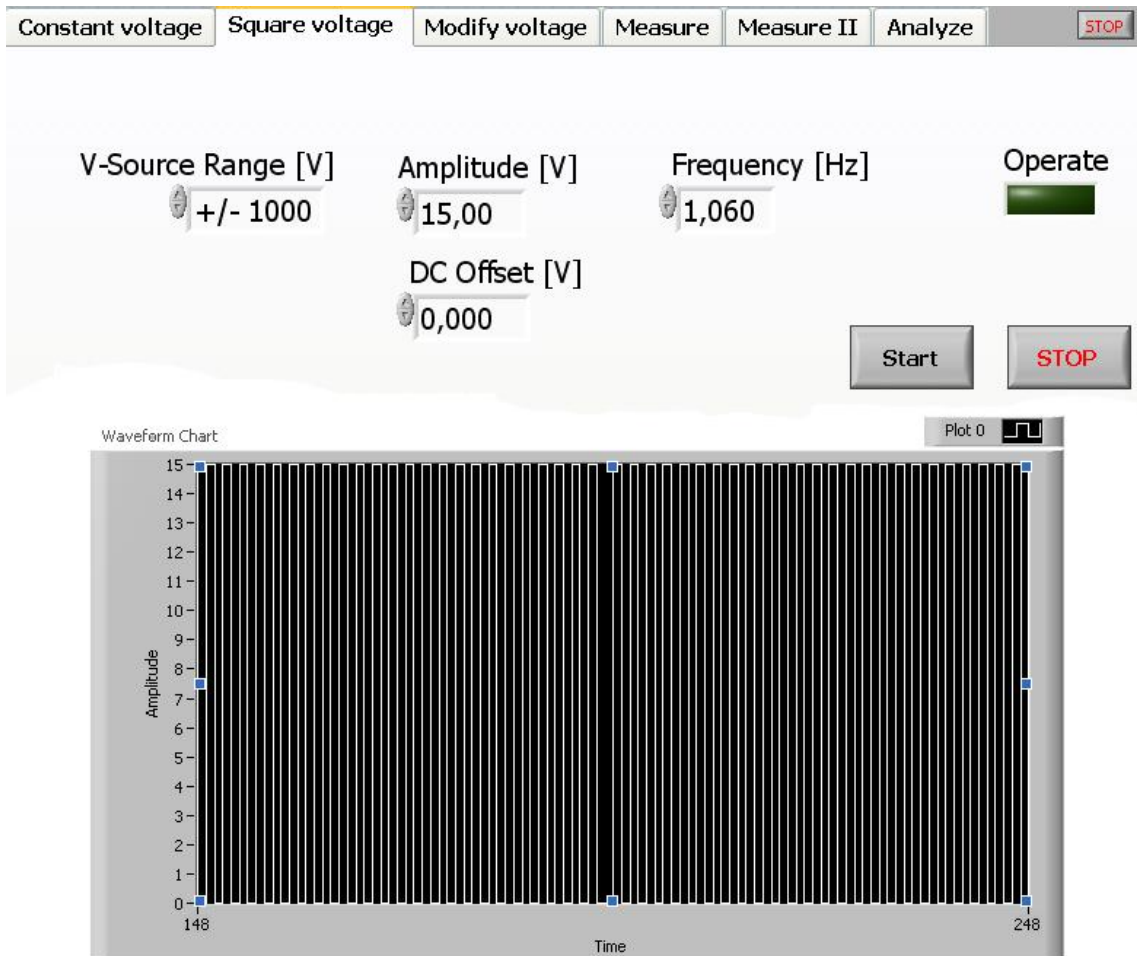
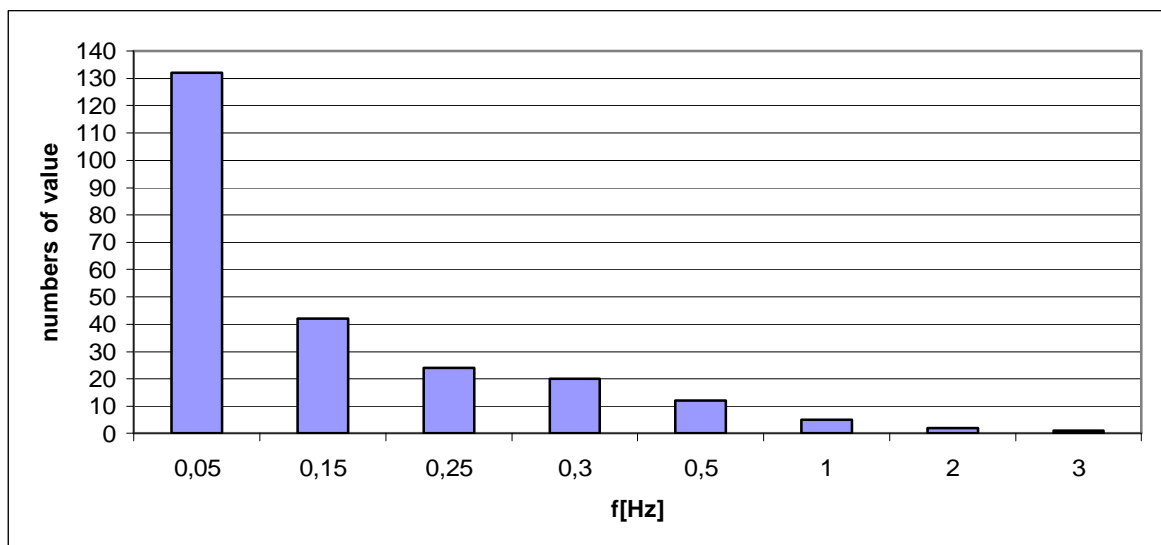
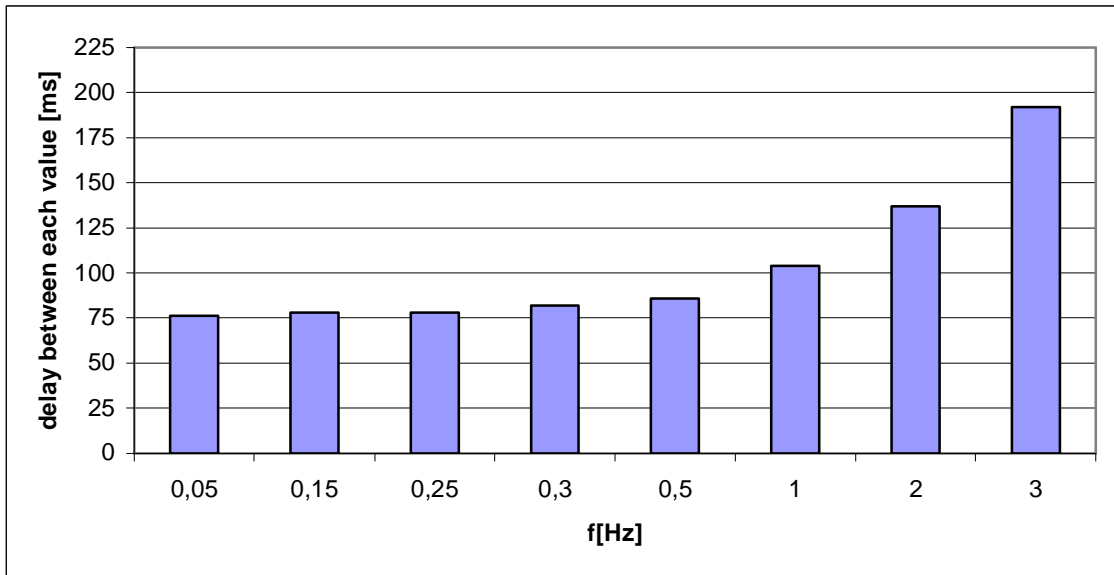


Fig. 23: Square Voltage

In a graph 1 we can see dependence of a number of values, which we can measure during an impulse, on a frequency. That means, if we choose square wave signal with frequency 0,5Hz, so we have got a period 2 seconds, the duration of impulse will be equal a half of the period i. e. 1 second, during this time we are able to get 12 values, as we can read in graph.

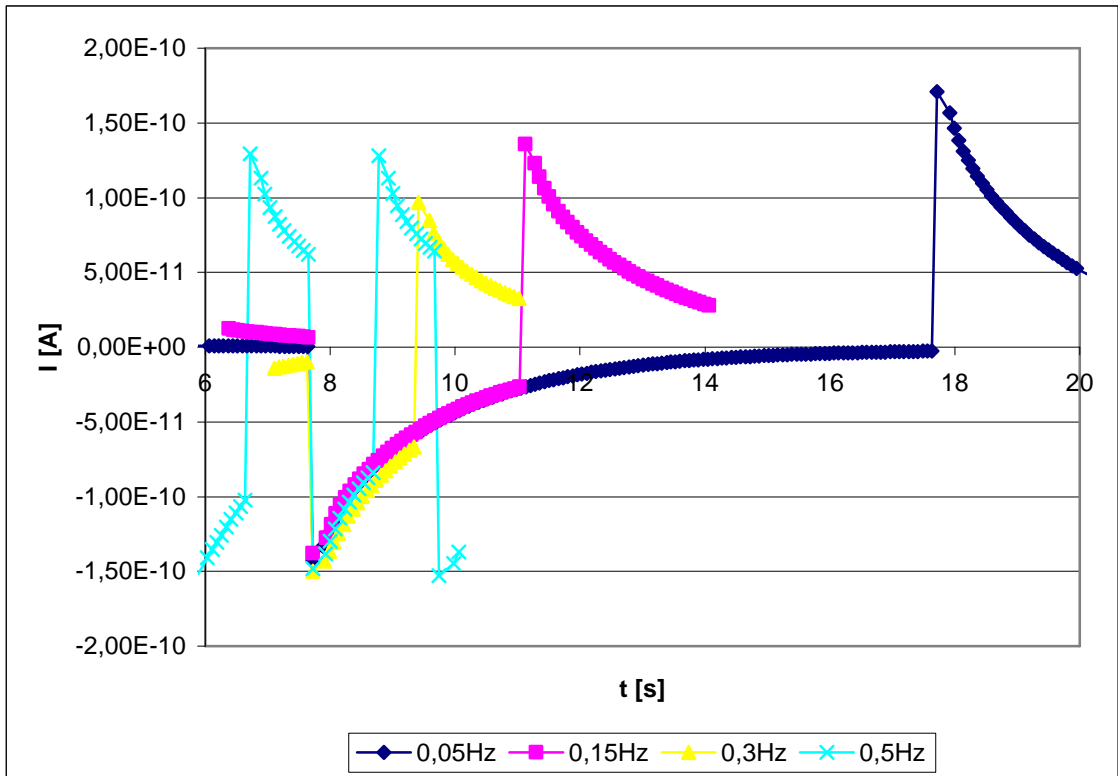


Graph 1: Numbers of value depends of frequency



Graph 2: Delay between each value

Next graph 2 search dependence of being delay between individual values during measurement, on a frequency. User can determine the time step for measurement. Both of graphs were made of values which had got during measurement with delay 0 seconds, so during the fastest measurement. Both of graphs clearly show that useful values of frequency are smaller than 0,5 Hz, in this interval we are able to point out the running as it is shown in graph 3. Maximum frequency of square wave signal we still were able to get at least one value was 3Hz and delay between individual values was 192 ms. From test measurements we learnt that we have to add a constant 68ms to each chosen value of delay. It's time, which measuring instrument needs to processing value (to write to a buffer etc.). Well, if user chooses the delay 0 seconds, he will measure in the fastest way, that it is possible (thus with delay 68s between individual values).



Graph 3: Course of current depends of frequency

A fold **MODIFY VOLTAGE** is determined to choice of generating of voltage, running of which the user creates himself. You can find it with name VsourceVer2.vi. on enclosed CD. It is possible to put any value of time and voltage in the table (max +/- 1000V with step 50mV). Minimum difference between two values of time should be 200ms. Then you can press START as in previous cases. However, now generating starts immediately, whole simulated course is repeated till user presses key STOP. This is suitable e.g. for creating of triangular course.

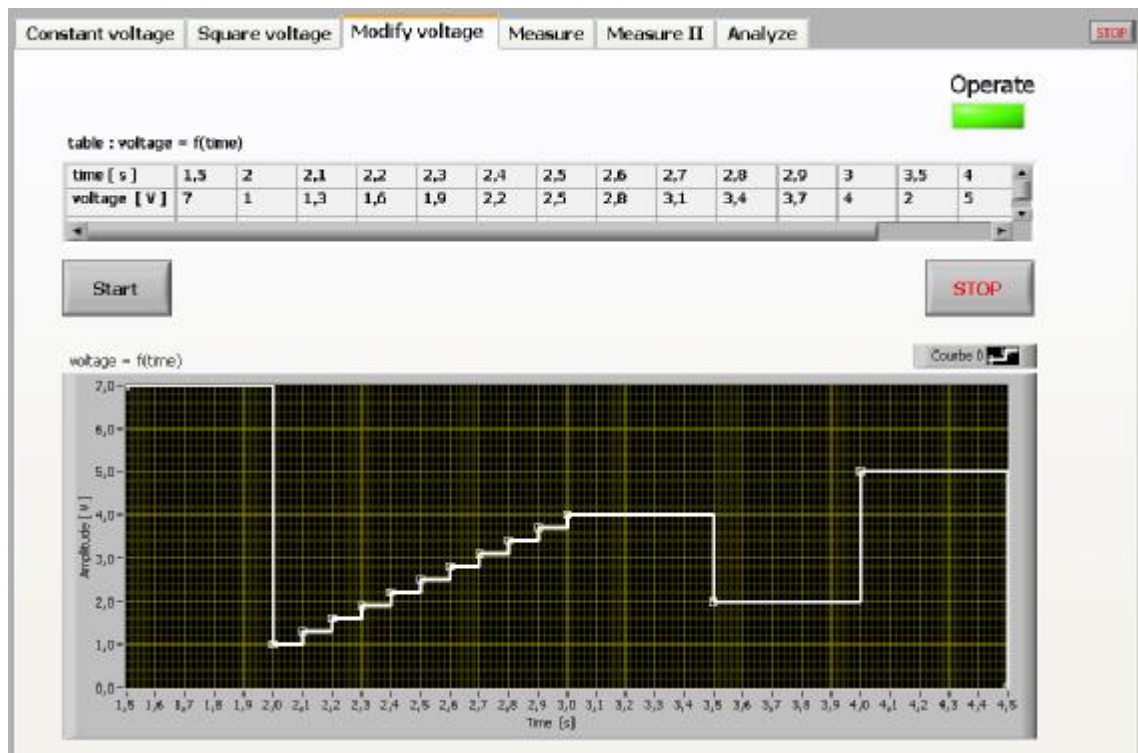


Fig. 24: Modify Voltage

Now we have chosen one of three types of a course of voltage and we can start measuring of electric current. We click the fold **MEASURE**(Figure 25), it's the first type of measurement, which is determined for short periods of measurement, maximum 50 minutes. Length of measurement is determined by user himself according to parametre of delay and numbers of measurements. E. g., if user chooses delay between individuals measurements 200ms and chooses a number values, which he wants to get on 5000, resulting time of measurement will be  $t=0,2*5000=16,7\text{min}$ . Maximum number of values which he can get is 8566 (size of buffer). Thus theoretically maximum time of measurement with clearly visible course of current  $0,2*8566 = 26,5 \text{ min}$ .

User can choose range of current or keep possibility AUTO. Next user should fill in optional field author, comments a date. Next he sets when measured results shall be saved, in a format txt. Led out of limit protection indicates range overflow. By unsuitable choice of range of current you can achieve the situation when the first values in the course of voltage change are out of range, the measuring instrument identifies them as value 999999, however, the other values are in picoamperes and that's why the graph would be distortion. Therefore they are taken out.

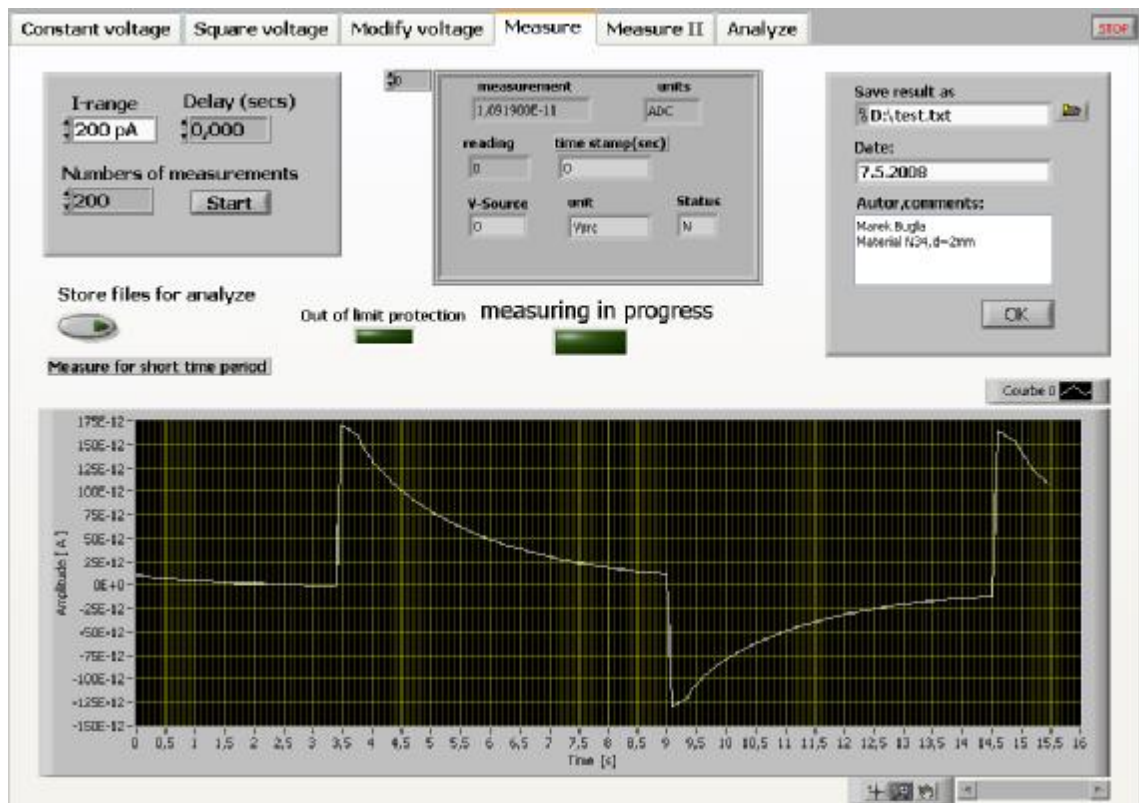


Fig. 25: Measure

An array, in which we can see measured values, is called measurements, there is a measured value, a unit, a number of measurement, time, when measurement was made, a value of Vsource, its unit and status that means, that the measured value was normal, overflow, underflow, out of limit, zero check enabled. Before program starts the user has to decide if he is going to analyse a resulting signal. If you want to do an analyse, you click a button STORAGE FILES FOR ANALYZE. Then you press start and the event, which vsource.vi or vsource2.vi is waiting for, occurs. The voltage is activating and immediately an electric current starts to be measured. The user has to wait for the finish of measurements, that indicates a led called measuring in progress. Then we can have a look at the measured values in the graph and exact values in array measurements.

Next fold is similar to previous but isn't dependent on any previous. This measurement is intended for square wave signal with a period longer than one hour. Maximum value of the period isn't limited and theoretically it was designed for infinitude. The user first chooses signal data in a left part. If he takes a number of period zero, the measurement will be running only in impulse duration, which will have a value of voltage which the user has chosen in the array output voltage value. If he chooses other period, to plus impulse the part with voltage zero will be added. Then he fills in items which he has known. After pressing a key start it is necessary to wait for the running of the measurement to the end. Complication of this measurement rests on the length of measurement. How it was said in an introduction, ideas of length of measurement changed during creating so it was necessary to make the second time unlimited measurement. Since it isn't possible to change a delay between individual values (see chap.6), it was necessary to create a cycle of

independent measurements. Constants of individual measurement cycles were deduced from knowledge of running of current. We can see values in following table.

Tab. 1: Constant for measuring cycles

number of measurement cycle	n of value	delay [s]	real delay [s]	duration[min]
1. measurement	1000	0,1	0,168	2,8
2. measurement	600	3	3,068	30,68
3. measurement	180	10	10,068	30,204
4. measurement	180	20	20,068	60,204

These constants were derived from test measurements and were consulted with a supervisor of the thesis. For all materials I have measured, this setting of values was sufficient. In case the measurement of a material with different running of current (slower falling characteristics), constants can be modified. In the figure number 26 we can see a structure CASE, in the concrete values for the first cycle of measurement, here it is possible to rewrite values into required ones. Next cycle of measurement you get by clicking an arrow in right. But it isn't a finish of a modification of a block diagram.

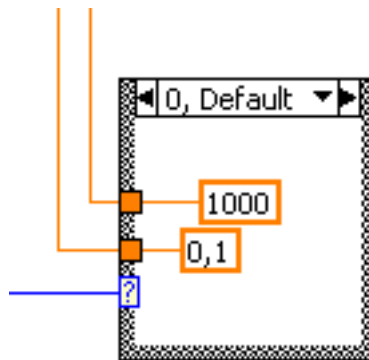


Fig. 26: Constants of measuring cycles

Since all measurement consists of  $n$  independent measurements, it was necessary to join them together in time, because each measurement always starts on zero. A result block diagram you can see in the figure number 27.

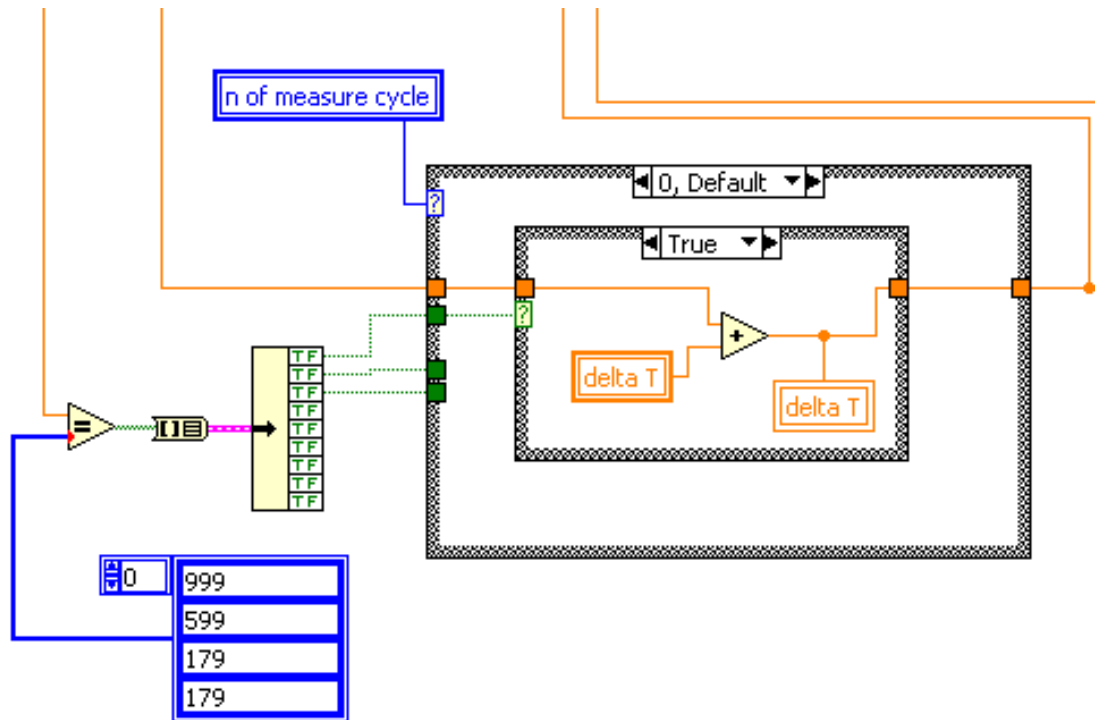


Fig. 27: Jointing of time

There are two CASE structure. The first depends on cycle order of measurement, the second depends on number of reading. In figure 26, the value of number of measurements is 1000 and in the figure 27 it is 999, a difference is due to numbering of measuring instrument keithely. The first measurement signs 0, the second 1 etc. As soon as it is block EQUAL number 999 that means, that the first cycle of measurement is finishing, and that's why a time value at that moment is saving into delta T. In the following cycle this constant is added to each time value. So the user has to change values which are written in cluster (marked blue).

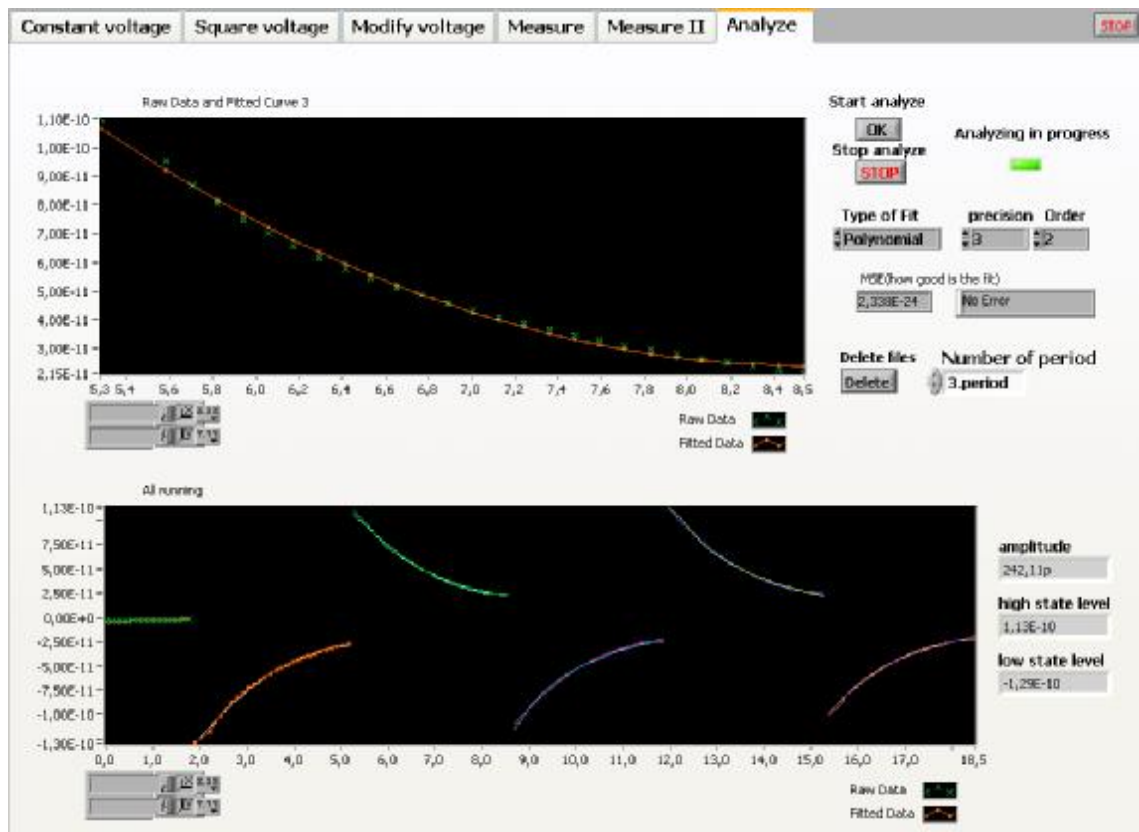


Fig. 28: analyze

The last fold is analyze, figure number 28. In this window the user can analyse measured running. First he chooses an interpolation type, either linear, polynomial or exponential. He can establish an exactitude. If the user chooses polynomial interpolation, next he can set up a order. Interpolations can be changed only when the analyse is not active. Then he presses a button start and watch results in two graphs, in the first he watches a detail of one period. It is possible to switch between periods with NUMBER OF PERIOD. A number of analyse periods is limited on 10, because in LabVIEW it is not possible to create controls or indicators in dynamic way. On the second graph in this window we can watch the whole course, for better orientation between periods. Next to the graph there is an array, when the user can read amplitude values, high state level and low state level. If the user has a look below the graph with the whole course, he will find equations describing the interpolations in periods. Reader can see this on figure 29. According to chosen the interpolation type there are result equations. For finishing the analyse the user presses a button stop, this step must be done always. Since analysed values are scan from a file (analyze1.txt, analyze2.txt atd.) it is necessary to delete them after finishing, we can do it by pressing a button delete files. In case the user wants to use these files, he can find them in an addresser, when the main program is. In individual files we can find current values for each half of period. If user during measurement didn't save files for analyse, it isn't possible to do anything in this folder.

## Equations of Fit

Equation 1

$$y = -4,179E-12 + 1,567E-12x - 193,277E-15x^2$$

Equation 2

$$y = -268,496E-12 + 88,636E-12x - 8,172E-12x^2$$

Equation 3

$$y = +587,482E-12 - 131,651E-12x + 7,691E-12x^2$$

Equation 4

$$y = -1,219E-9 + 201,766E-12x - 8,521E-12x^2$$

Equation 5

$$y = +1,853E-9 - 239,054E-12x + 7,814E-12x^2$$

Equation 6

$$y = -2,642E-9 + 283,239E-12x - 7,656E-12x^2$$

Fig. 29: Equations of Fit

### 7.3 Describe of Block Diagram

Structure of main program is showed at picture 30. The whole program is closed into a loop in order to infinite running. After the program starts up, a new occurrence is created, which is saved as a global variable. This global variable solves the problem of waiting for measure (which was described in paragraph 7.2). In whole system many CONTROL components are used. From those components are created local variables. TABCONTROL determinates on which fold we are currently located. Almost every fold has structure as CASE structure showed in picture 30. After pushing START button concrete operations are executed, after pushing STOP button other concrete operations are executed. There are other functional buttons in the system.

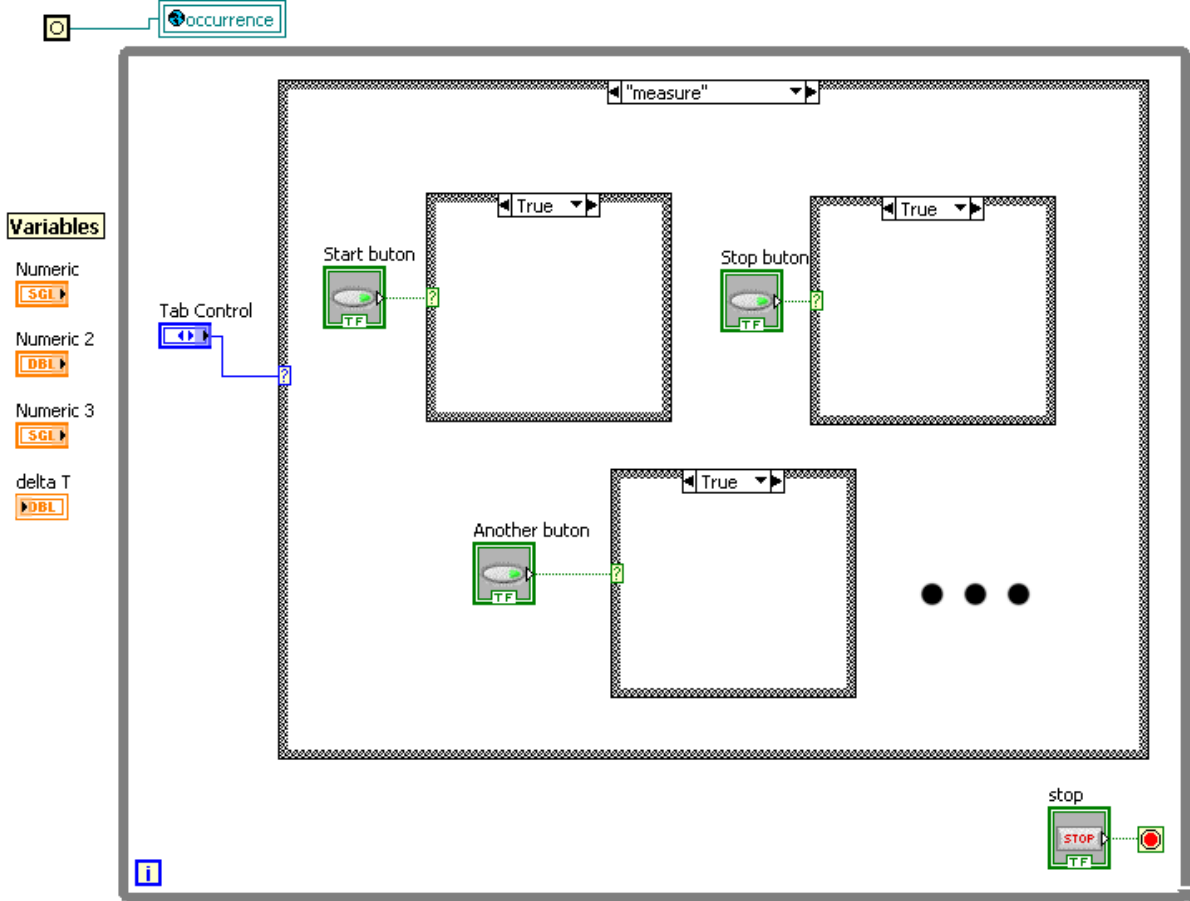


Fig. 30: Structure of system

First three folds seem similar. It was necessary to create parallel start-ups of some VI, because it was necessary to change voltage during the measurement. At the figure 31 is showed part of fold SQUARE VOLTAGE. It is opened by VI called SquareGen.vi and its reference is connected with to the sides, which sets relevant values according to name form field called CONROL NAME. WAIT UNTIL DONE means that the main VI, which started-up parallel subVI SquareGen.vi, doesn't wait for its end. Of course, those running parallel processes are necessary to be ended properly by pushing the STOP button. Shortly, the first three folds are run parallel to the main VI of appropriate subVI, particularly VsourceVI, SquareGenVI and VsourceVer2.vi.

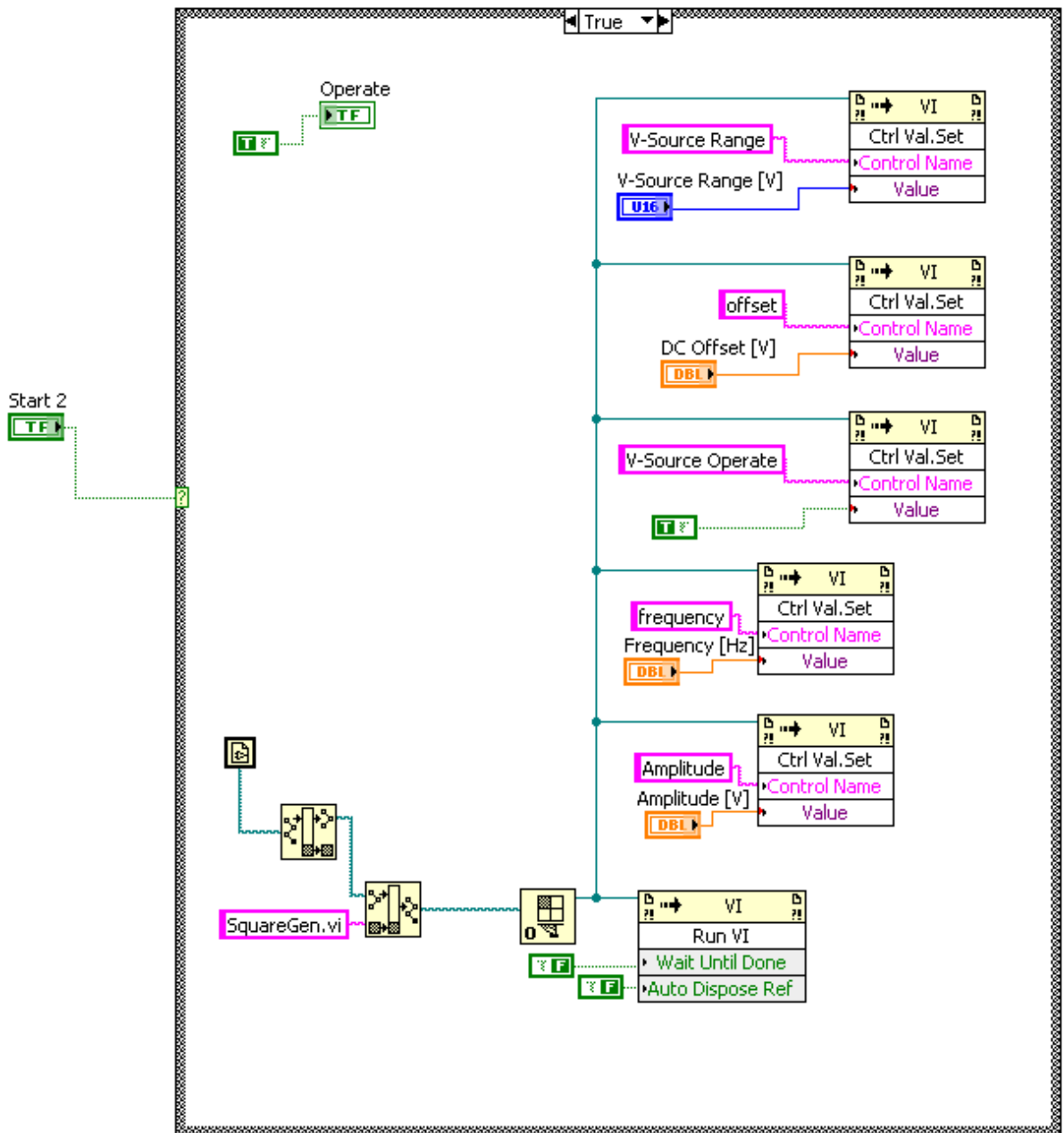


Fig. 31: Fold Square Voltage

At picture 32 a fractional part of fold MEASURE is shown. There are two subVI by which are the measurements performed. SubVI DCCconf.vi sets current measuring instrument Keithley, Another subVI carries out the necessary procedures for successful measurement – relevant registry setting, configuration of layers, which were mentioned at chapter 6. Furthermore, previously mentioned occurrence setting is apparent from this figure.

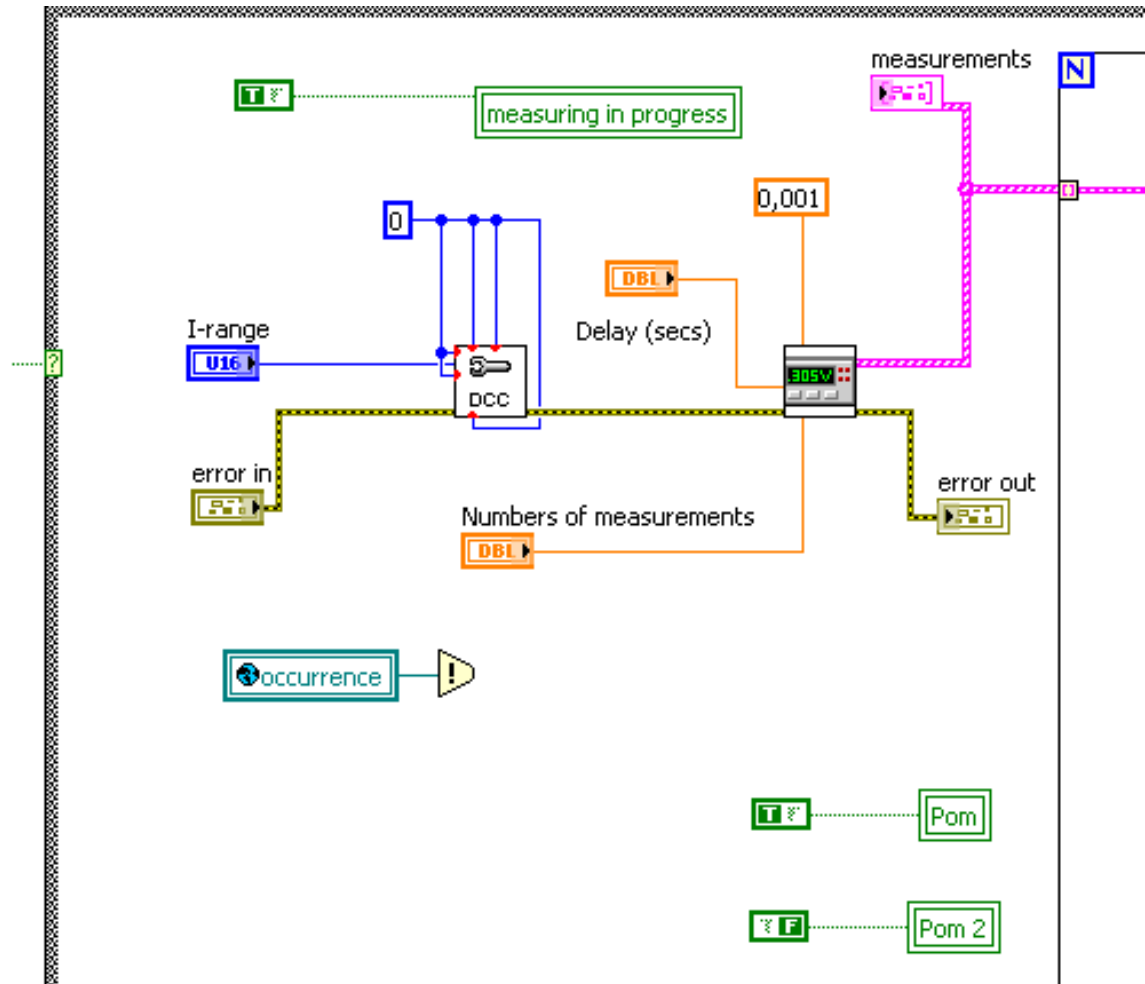


Fig. 32: Fragment of Measure

The following figure shows the solution of saving of each parts of period depended on Vsource voltage. It was necessary to save results to another file when the Vsource voltage changed. Of course, reader can get across about simple idea of simplified solution of this issue, when just the polarity of Vsource would be tested. So, in case of negative polarity, one file would be used, otherwise, the second file would be used. But a problem appears in the second positive value of Vsource, because after the first pass, analyze1.txt file is created in CASE STRUCTURE when it is true, after the next pass of this structure, analyze2.txt file would be created. It's absolutely necessary to save measured values into the same file. Creation of new file is acceptable only if the value of Vsource changes. To this new file, the values affected by Vsource chase, will be saved. If the value of Vsource will be again positive, its's not possible to save the values in analyze1.txt file, which was created after the first pass of positive Vsource impulse. That is why was sophisticated flip-flop system created, which allows this. The easiest way to understand function is to simulate this component in new VI.

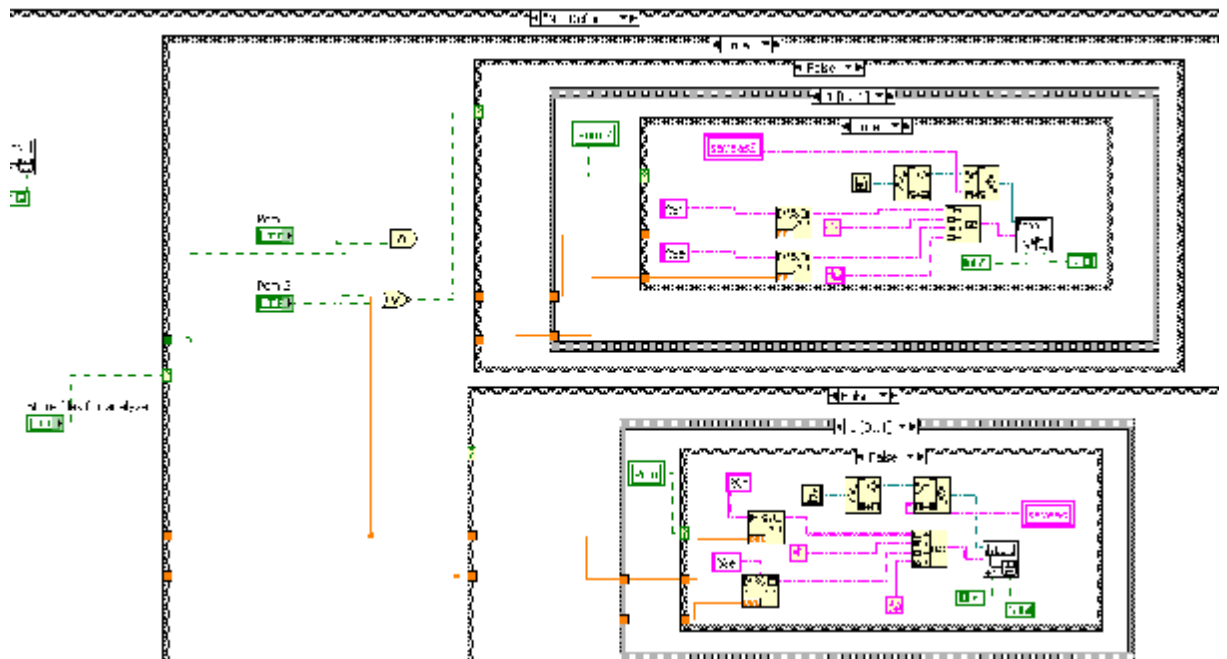
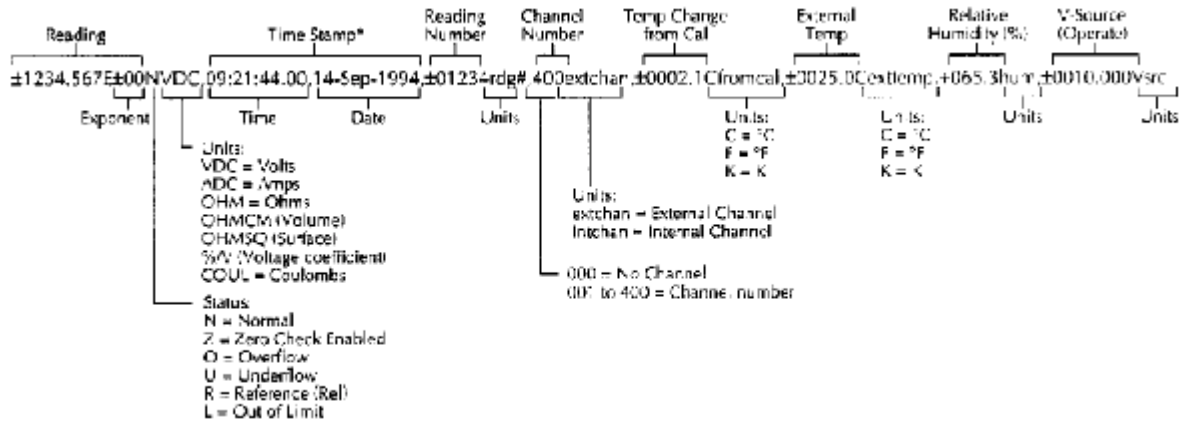


Fig. 33: Saving data to file



\* Real time clock times/ama shown  
 Fig. 34: ASCII data format

Figure 34 explains ASCII format of data received from Keithley measuring instrument. For our case, it was not necessary to take into account all data elements. So, the resulting data have this form:

-0000.958E-12ADC,+0000000.000000secs,+00000rdng#,+0010.000Vsrc

The way how are the data transformed and processed is solid in VI called measure.vi, especially in block that is shown in figure 35.

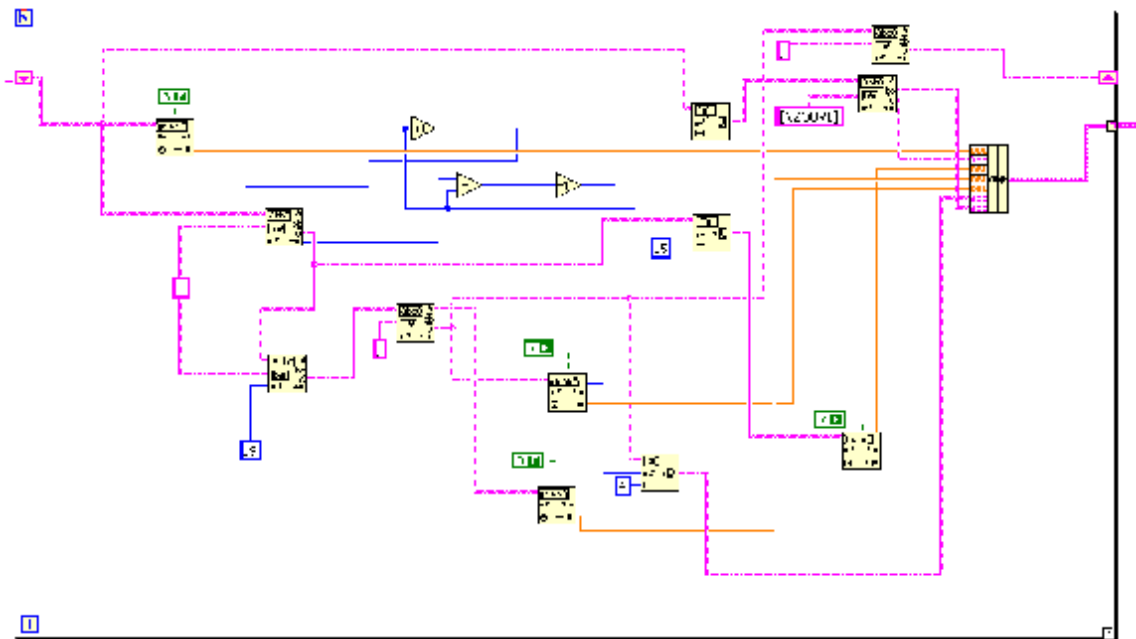


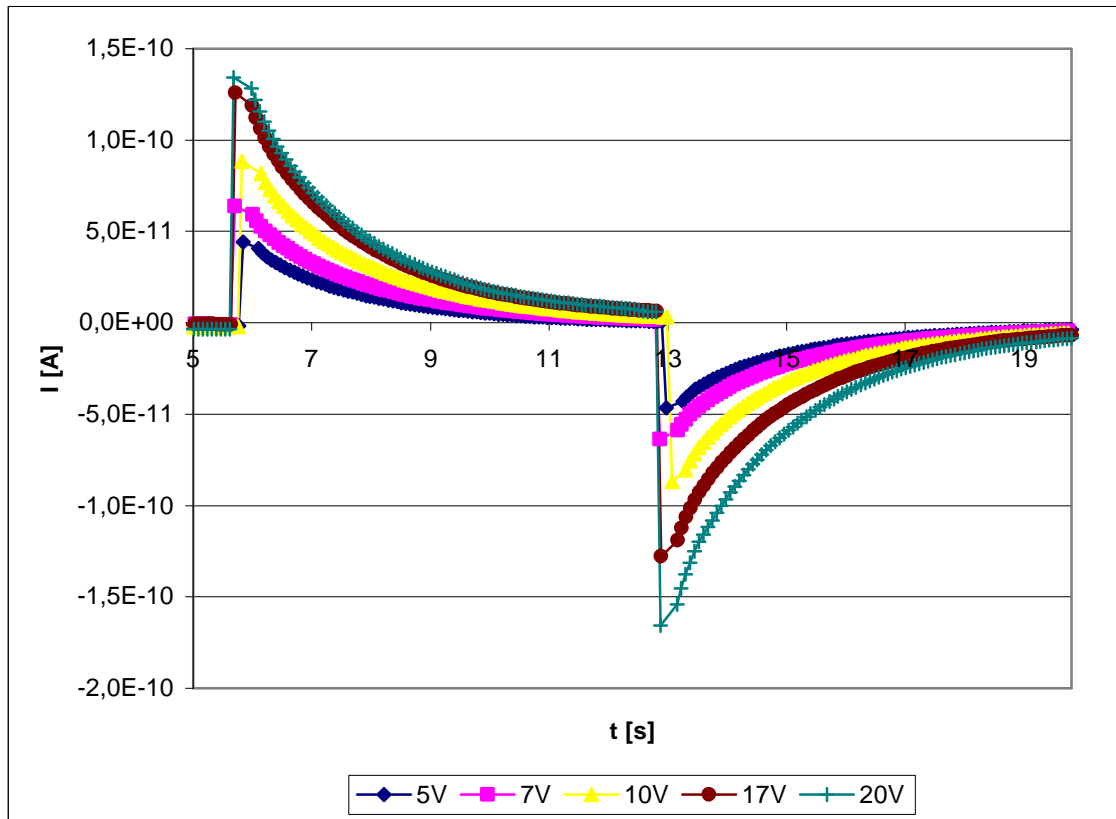
Fig. 35: Analysis of received data

We took a N packets of datas, so it is necessary to repeating this loop N times. It was use element Fract/Exp String To Number, which interprets the characters 0 through 9, plus, minus, e, E, and the decimal point (usually period) in string starting at offset as a floating-point number in engineering notation, exponential, or fractional format and returns it in number. We can see Match Pattern, which searches for regular expression in string beginning at offset, and if it finds a match, splits string into three substrings. Another element, which is

use in this block diagram is string subset. Returns the substring of the input string beginning at offset and containing length number of characters. The last element is a bundle, which assembles a cluster from individual elements.

## 7.4 Results

The following paragraph shows the some results of measurements.

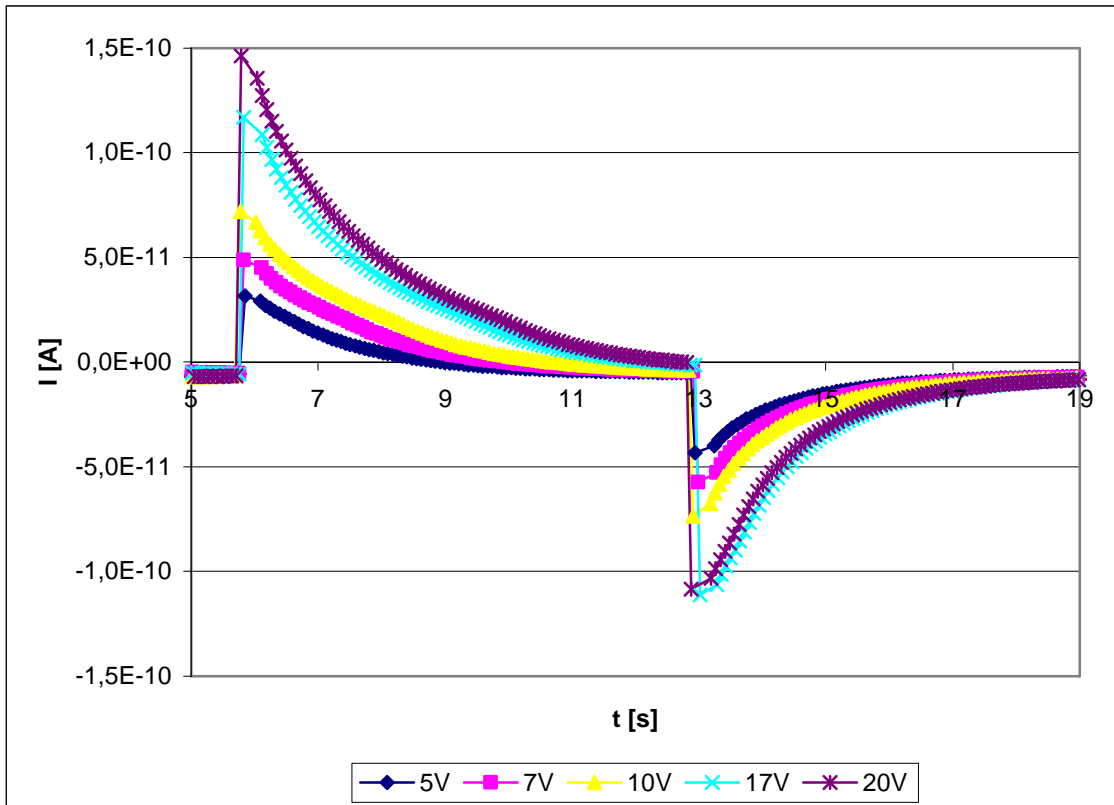


Graph 4: Epoxy resin

In graph number 4, reader can see dependence of current on voltage. Frequency 0,07Hz was applied for each measure; amplitude of square signal was successively 5, 7, 10, 17 and 20 V. The shape of every curve is the same. Peak values of current rises with increasing voltage as we can see on following graph. The name of this material is Epoxy resin and is determined for high voltage applications.

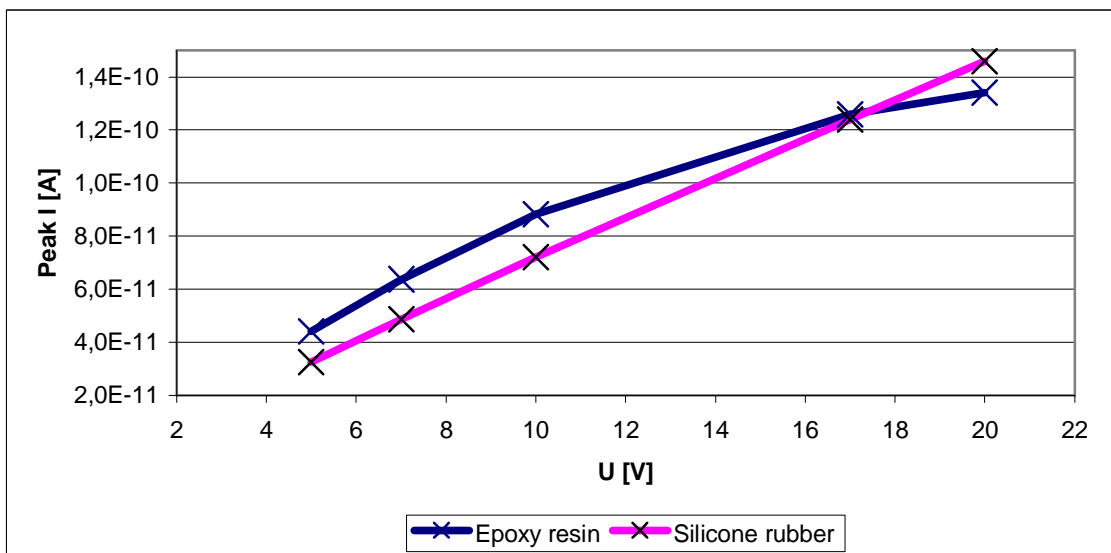
In following text we can see fragment of text file (5V.txt):

```
marek bugla, Epoxy resin 5V, 15.5.2008
read time current Vsource Status
0 0,000000 1,206000E-12 0 N
1 0,069668 1,083000E-12 0 N
2 0,138500 9,860000E-13 0 N
3 0,212310 8,700000E-13 0 N
```



Graph 5: Silicone rubber

Graph 5 shows relation between current and voltage changing in time. Frequency 0,07Hz was applied for each measure; amplitude of square signal was successively 5, 7, 10, 17 and 20 V. The name of this material is silicone rubber and is determined for high voltage applications.



Graph 6: compare materials

Graph 6 shows relation between peak values of current and voltage. Current values rise with increasing voltage. For silicone rubber this characteristic has linear progress, for epoxy resin it has exponential tendency.

The following figure show a characteristics of Varioptic material. Varioptic has developed a product offer for a wide range of markets from camera phones, consumer devices to industrial applications, which feature some crucial advantages over traditional technologies:

- Robustness
- Cost
- Size
- Ease of integration
- Power consumption
- Speed

The liquid lenses that they develop are based on the electrowetting phenomenon described below :

A water drop is deposited on a substrate made of metal, covered by a thin insulating layer. The voltage applied to the substrate modifies the contact angle of the liquid drop. The liquid lens uses two isodensity liquids, one is an insulator while the other is a conductor. The variation of voltage leads to a change of curvature of the liquid-liquid interface, which in turn leads to a change of the focal length of the lens.

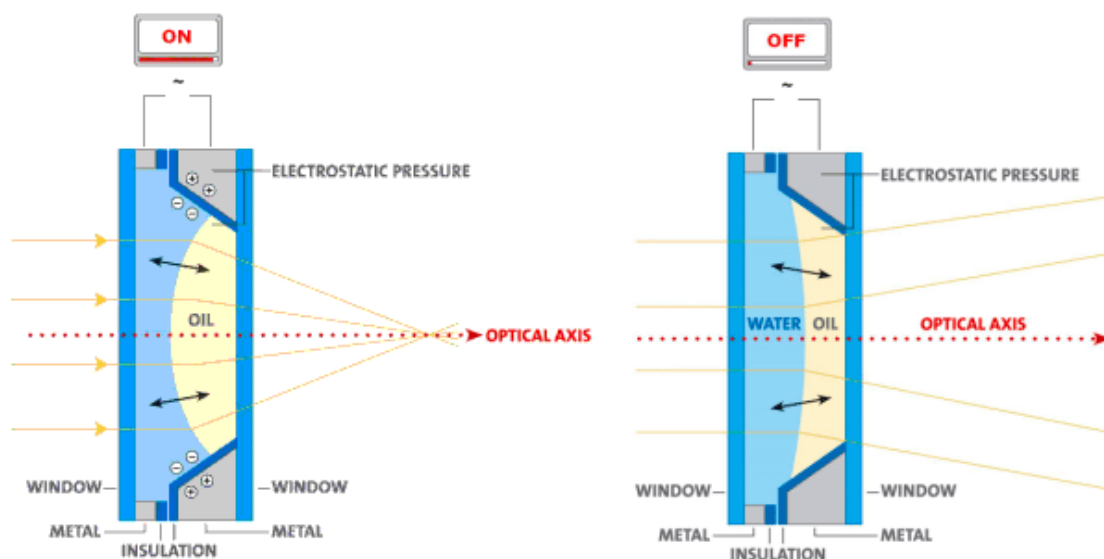
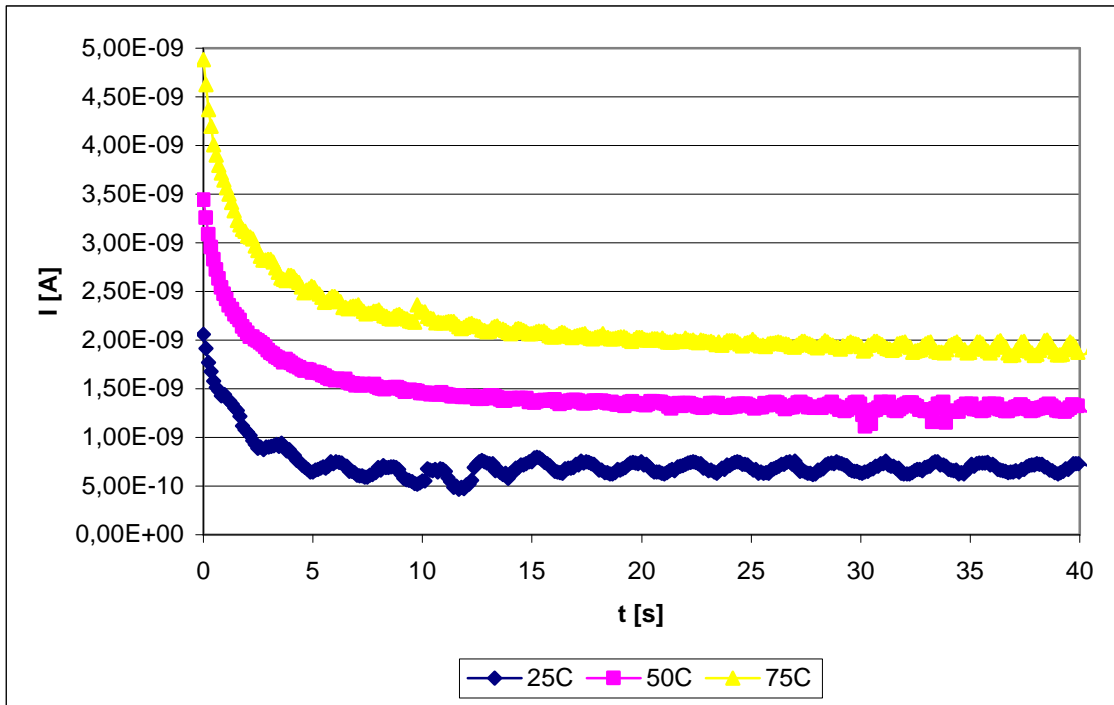


Fig. 36: Principle of function



Graph 7: Material, which is part of Liquid Lense

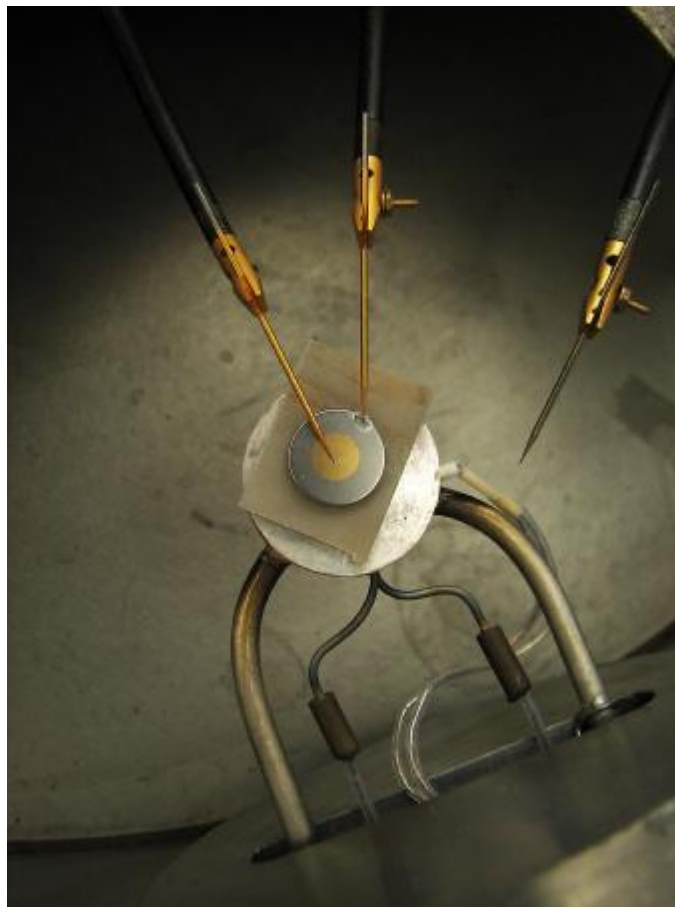
In figure 7 we can see a depends of current on the temperature. With increasing temperature increasing current too. It was apply three value of temperature, 25, 50 and 75 °C. The sample was in thermal equipment, which we can see on figure 37. This equipment sustains constant temperature, which operator can set up on Linkam.



Fig. 37: Keithley connected to thermal equipment



*Fig. 38: View of whole*



*Fig. 39: View of the sample*

# Chapter 8

## Conclusion

At the beginning of this thesis I described in which way I will solve problems in this thesis. Now I would like to summarize what I achieved in this project highlight difficulties which appeared.

How was mentioned, this master thesis was created in abroad. Before my departure, the only thing I knew was that it is necessary to create wide measure system in LabVIEW. After few first meeting with my supervisor it was clear, that they aren't interesting in any complicated solution of remote laboratory. I knew how easy is to public programs created in LabVIEW on the internet, in fact how to create virtual or remote laboratory. I didn't want to change subject of my thesis, because I knew I will work with real devices which I will control via LabVIEW. From mentioned arrangement it is possible to create remote laboratory. I would be apposite to let reader of this thesis look into my semester work where is in detail described distinction between virtual and remote laboratory, because also subject of thesis is not accurate. This subject in the right meaning I accomplished in my semester thesis, where I created virtual laboratory. I learned from supervisor of my semester work Ing. Pfeifer, that it is necessary to create laboratory to measure some known electric task (exercise). It means to scan real variables from concrete electric circuit. Because it is about real quantity it means, that we are talking about remote laboratory. Not about virtual laboratory, how it is in project specification. For closer description of differences between these two systems I refer to semester project. There is also an example of virtual laboratory, which realizes calculation of currents in electric circuit with method of loop currents.

It needed to create system, by which it is possible to generate voltage and measure current in range of pikoamperes by Keithley device. Generated voltage can has several waveforms, what is shown in chapter 7. The value of square signal period was changing during creating program. In first discussions it was mentioned 1ms, but this value was changed too quickly. It is technically impossible to get data from measure device this fast. So I started working with periods about 5 seconds, what is showed at graph 5. Maximum frequency, when it is possible to get at least one value of current, is 3Hz. After this, I find out, that period supposed to be for example 4 hours. Values changed 240 000 times in comparison with previous idea! Because of that, it was a must to change whole system and create new time independent measure (fold Measure II).

I created functional program, which allow scientists from CNRS examine characteristics of dielectric material. Program allows choosing from three waveforms of voltage. Also two types of measure can by chose; short-term (0 - 50 min) or long-term (30min - infinity). System is part of extensive project. In next step of realization program supposed to be improve by possibility of

changing temperature, because how it showed at graph 7 current is depends on temperature.

In final phase I was allowed to create remote laboratory. Measures can take long time, in range of tens hours. So application of remote laboratory was obvious. I explained to my project supervisor, what is possible to make by this kind of remote laboratory. We could start, check, see data, measure and evaluate results of whole system, all by remote.

For a weak point of system I concern that is not possible to stop measuring. Unfortunately I wasn't able to solve this problem. In next step of developing this system, this is the first thing should be focused on. Because when wrong parameters are entered, we would wait e.g. 4 hours for finish task. The only possible how to stop measure is to turn off and on again measuring device Keithley.

# Bibliography

- [1] NATIONAL INSTRUMENT. *LabVIEW User Manual* [on-line] . Available from www : <<http://www.ni.com/labview/>>
- [2] NATIONAL INSTRUMENT. *DAQ User Manual* [on-line]. Available from www: <<http://www.ni.com/DAQ/>>
- [3] NATIONAL INSTRUMENT. *LabVIEW Graphical Programming Course* [on-line]. Available from www: <<http://cnx.org/content/col10241/1.2>>
- [4] NATIONAL INSTRUMENT. *GPIB(IEEE 488) Solutions* [on-line]. Available from www: <<http://www.ni.com/gpib>>
- [5] PECHOUŠEK, J. *Základy programování v prostředí LabVIEW*. 1. vydání. Olomouc: skripta UPOL, 2004. 81 s. ISBN 80-244-0800-7.
- [6] BENCOMO, S.D. *Control learning: Present and future. IFAC 15th Triennial World Congress, 2002, Barcelona, Spain.*
- [7] HORÁČEK, P. *Laboratory experiments for control theory courses: A survey. Annual Reviews in Control*
- [8] LAMBERTUS H. ,Bjornson E.,Rizal D.. *CyberLab: Internet Assisted Experimentation* [on-line]. Available from www: <<http://kaos.stanford.edu/pdffiles/Cyberlab.pdf>>
- [9] K. C. CHU, LEUNG D. *Flexible Learning Via Web-Based Virtual Teaching and Virtual Laboratory Systems* [on-line]. Available from www: <<http://scholar.lib.vt.edu/ejournals/JOTS/v29/v29n2/chu.pdf>>
- [10] AKTAN B.,BOHUS C., CROWL A., SHOR M.. *Distance learning applied to control engineering laboratories, IEEE Transactions on Education, Vol 39, No. 3, August 1996.*
- [11] *GPIB TUTORIAL* [on-line]. Available from www: <<http://www.hit.bme.hu/~papay/edu/GPIB/tutor.htm>>
- [12] WIKIPEDIA, the free encyclopedia. *IEEE-488* [online]. Dostupné z WWW: <<http://en.wikipedia.org/wiki/IEEE-488>>
- [13] University of Oregon. *Virtual laboratory* [on-line]. Dostupné z www: <<http://jersey.uoregon.edu/vlab/index.htm>>
- [14] BURGET P., DOLEJŠ O., HANZÁLEK Z., KIRCHMANN B.. *Remote Programming of Control Systems*. 2004, Grenoble, France.
- [15] KEITHLEY INSTRUMENT, INC.. *Electrometer User's Manual* [on-line] . Available from www : <<http://www.keithley.com>>

- [16]** AGROUI K., COLLINS G. *Characterisation of EVA encapsulant material by thermally stimulated current technique*. 2003, Ohio, USA.
- [17]** C. JUHASZ, B. M. Z. KAMARULZAMAN, S. M. VAEZI-NEJAD. *Theoretical considerations of thermally stimulated discharge techniques in amorphous semiconductors*. 1991, London, UK.
- [18]** NEAGU E.R.. *Use of the final thermally stimulated discharge current technique to analyze space charge behavior in dielectrics*. 2003, Iasi, Romania.
- [19]** SYLVESTRE A.. *Electrical conduction in insulating materials*. 2003, Grenoble, France.