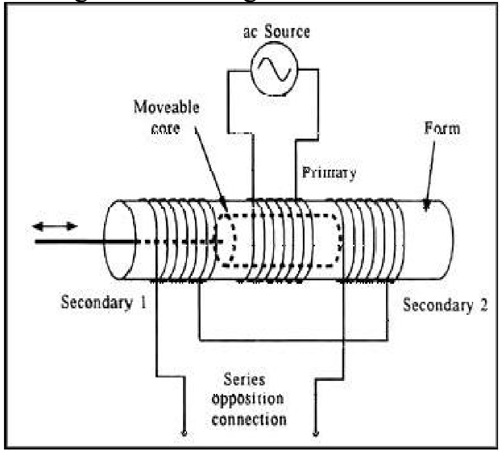
Objective

The objective of the project is to implement Virtual Instrumentation concept using LabVIEW. A virtual instrument is a simulation of an instrument either physically existing or is in a conceptual phase implemented through software's. 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. For a case study a spring pendulum system has been considered whose motion has to be analyzed using LabVIEW.

Introduction

LOADCELL:-Loadcell is a transducer,which converts force into measurable electrical output.there aremany type of loadcell,strain gauge based loadcell is most commonly used.

LVDT:-The LVDT is most widely used translate linear motion into electrical motion.This is an inductive transducer. Linear variable differential transformers (LVDT) are used to measure displacement. LVDTs operate on the principle of a transformer. As shown in Figure 2, an LVDT consists of a coil assembly and a core. The coil assembly is typically mounted to a stationary form, while the core is secured to the object whose position is being measured. The coil assembly consists of three coils of wire wound on the hollow form. A core of permeable material can slide freely through the center of the form. The inner coil is the primary, which is excited by an AC source as shown. Magnetic flux produced by the primary is coupled to the two secondary coils, inducing an AC voltage in each coil.



General LVDT Assembly

The main advantage of the LVDT transducer over other types of displacement transducer is the high degree of robustness. Because there is no physical contact across the sensing element, there is no wear in the sensing element.
Because the device relies on the coupling of magnetic flux, an LVDT can have infinite resolution. Therefore the smallest fraction of movement can be detected by suitable signal conditioning hardware, and the resolution of the transducer is solely determined by the resolution of the data acquisition system.

ACCELEROMETER:-An **accelerometer** is an electromechanical device that measure acceleration forces. An **accelerometer** is a device that measures proper acceleration. The proper acceleration measured by an accelerometer is not necessarily the coordinate acceleration (rate of change of velocity).

WHAT IS NI?

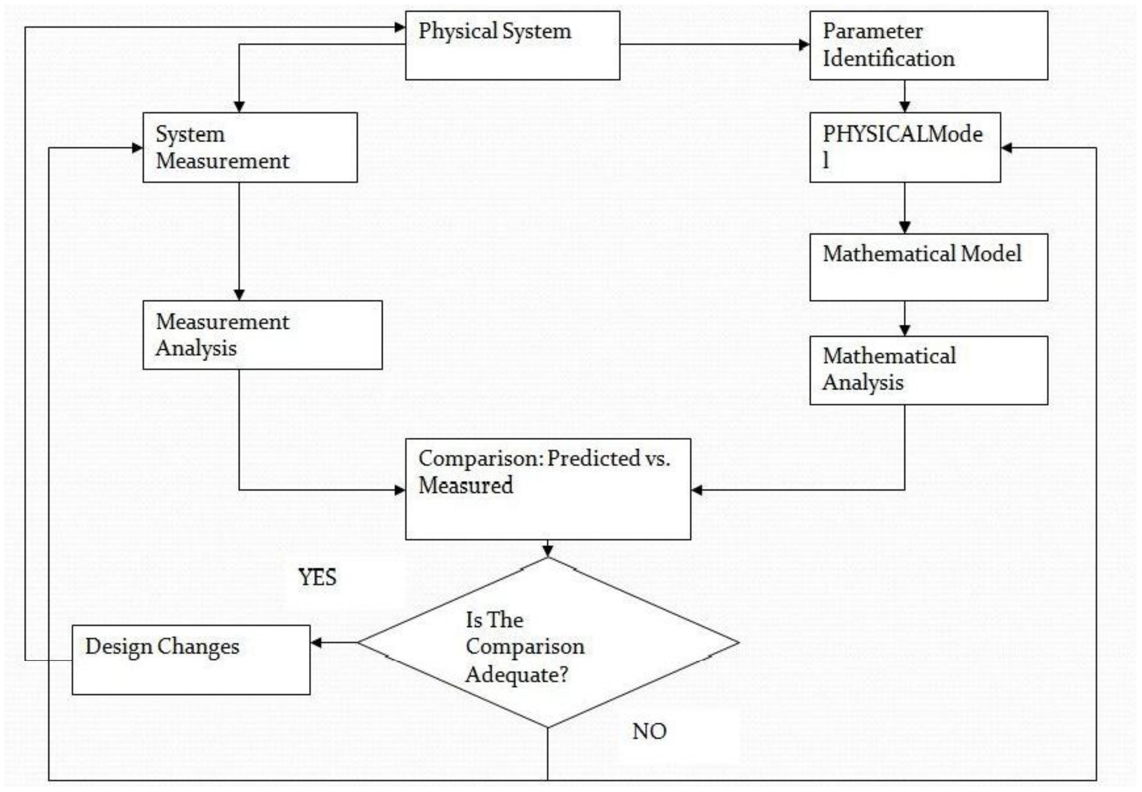
- NI provides a grapical system design platform for test ,control and embedded design application that is transforming the way engineering and scientists design,prototype and deploy systems.

Virtual Instrument

- Instrument which may or may not exist physically
- Simulated through various software used in practice, like, LabView, AMESim, etc.
- Testing is not done at implementation level Error correction is done while simulating the instrument through software

<u>LabVIEW Features</u>

- Not like the conventional hand written programming like C, Java Programming Languages.
- There are controls and blocks.
- The controls and blocks are connected through wiring.



- From the very early days computers found a role in process monitoring and control. The early computers were bulky and costly. Their application was limited to the top end of applications, and they were found only in a few large plants, experimental systems, etc. furthermore, the computational capabilities were minuscule by modern standards. One can safely say that the super computer of any age is less powerful than the simplest computer of a couple of decades later. The average calculator (or even a digital wristwatch) of today is more powerful than the earliest super computers.

- Subsequently as the prices dropped and the capabilities expanded, the use of computers also grew. Following the development of powerful and inexpensive personal computers this growth has exploded, and today a computer is more often than not, an integral part of the instrument.

- The progress in the application of computers in instrumentation can be divided into three distinct but inter-related areas-computer hardware, software, and interfaces.

- Computer hardware is those physical devices like motherboard, hard disk, RAM, Microprocessor etc. that are integrated together to accomplish multiple purpose.

- Computer software is programs that make possible the interaction between the hardware components so that some specific task may be accomplished.

- An interface is a physical device looks like a card through which other physical devices that are not essential part of the computer, can be connected. A user can interact or instruct the devices through interfaces.

- As the interest in the uses of computers grew users duplicated the front panel control functions of the existing instruments through the computer interface. One of the earliest such applications was the multi-channel analyzers. Thus the instruments could be controlled through the front panel as well as the computer. Then somebody had a bright idea-since almost all the time the instruments is being controlled by the computer, why not do away with the front panel altogether? And thus the first virtual instrument was born. This was the period when the most popular interconnection bus for test and measurement instruments was the GPIB and the use of the GPIB for setting up a test system can also be considered to be the first virtual instrument.

- Since then, VI has come a long way. Today it's a technique in its own right finding widespread application over a vast range of applications, ranging from small laboratory experiments, to large automation applications encompassing complex plants, and even town, districts and countries. At the lower end you have a scientist controlling a small experiment or instrument, on the other end, VI is used for managing the distribution of natural gas in Switzerland, and electricity in southern California.

- Thus Virtual Instrumentation is-Industry-standard computers equipped with user-friendly application software, cost effective hardware and driver software that together perform the functions of traditional instruments.

Why LabVIEW?

- LabVIEW (Laboratory Virtual Instrument Engineering Workbench) 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 the order of program execution, LabVIEW uses dataflow programming, where the flow of data through the nodes on the block diagram determines the execution order of the VIs and functions. VIs, or virtual instruments, is LabVIEW programs that imitate physical instruments.

- In LabVIEW, you build a user interface by using a set of tools and objects. The user interface is known as the front panel. You then add code using graphical representations of functions to control the front panel objects. This graphical source code is also known as G code or block diagram code. The block diagram contains this code. In some ways, the block diagram resembles a flowchart.

- You can purchase several add-on software toolkits for developing specialized applications. All the toolkits integrate seamlessly in LabVIEW.

Virtual Instrument

- Instrument which may or may not exist physically.

- Simulated through various software used in practice, like, LabView, AMESim, etc.

- Testing is not done at implementation level.

- Error correction is done while simulating the instrument through software.

A VI contains the following three components:

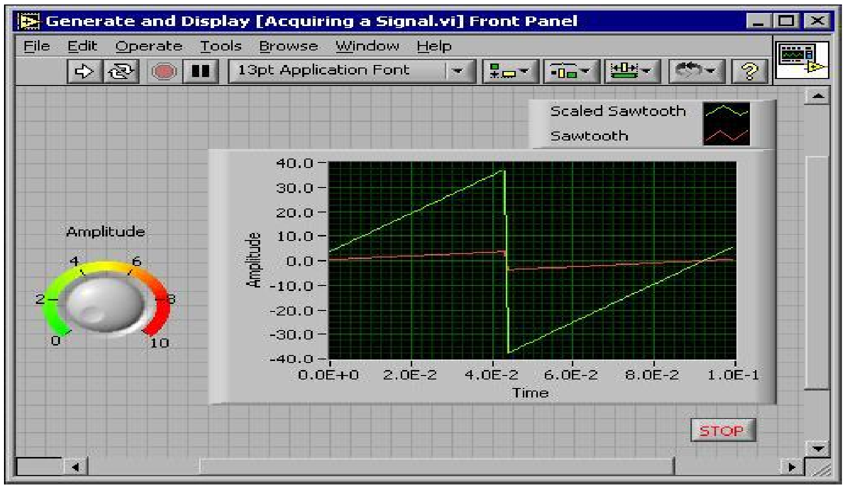
Front panel—Serves as the user interface.

Block diagram—contains the graphical source code that defines the functionality of the VI.

Icon and connector pane—identifies the interface to the VI so that you 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.

Front Panel

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

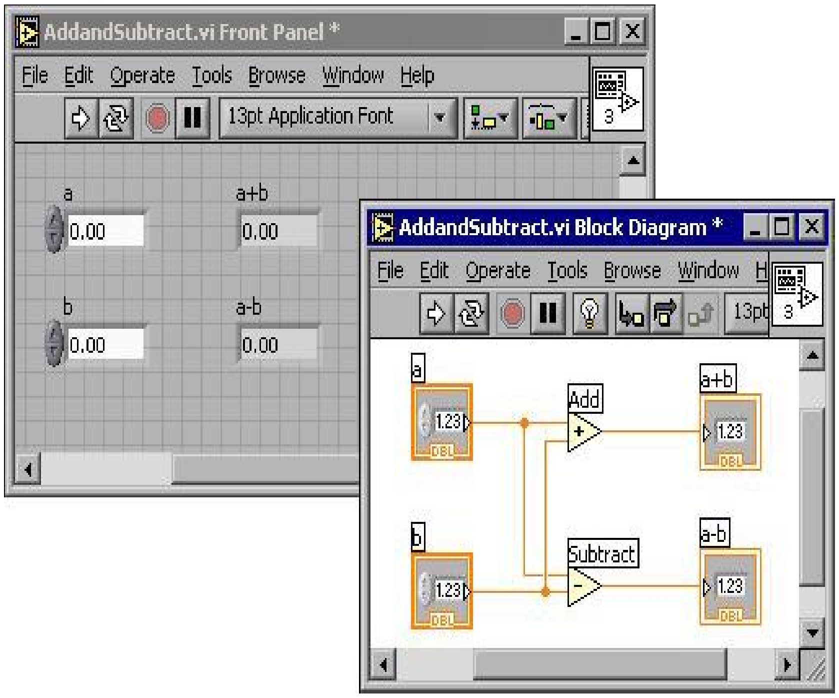


Example of a Front Panel

Block Diagram

After you build the front panel, you add code using graphical representations of functions to control the front panel objects. The block diagram contains this graphical source code, also known as G code or block diagram code. Front panel objects appear as terminals on the block diagram.

The following VI contains several primary block diagram objects—terminals, functions, and wires.


AddandSubtract.vi Front Panel *
File Edit Operate Tools Browse Window Help
13pt Application Font
a
0.00
a+b
0.00
b
0.00
a-b
0.00
AddandSubtract.vi Block Diagram *
File Edit Operate Tools Browse Window H
13pt
a
1.23
DBL
Add
a+b
1.23
DBL
b
1.23
DBL
Subtract
a-b
1.23
DBL

Example of a Block Diagram and Corresponding Front Panel

- ***Terminals***

The terminals represent the data type of the control or indicator. You can configure front panel controls or indicators to appear as icon or data type terminals on the block diagram. By default, front panel objects appear as icon terminals. For example, a knob icon terminal, shown as follows, represents a knob on the front panel.

The DBL at the bottom of the terminal represents a data type of double-precision, floating-point numeric. A DBL terminal, shown as follows, represents a double-precision, floating-point numeric control.

Terminals are entry and exit ports that exchange information between the front panel and block diagram. Data you enter into the front panel controls (**a** and **b** in the previous figure) enter the block diagram through the control terminals. The data then enter the Add and Subtract functions. When the Add and Subtract functions complete their calculations, they produce new data values. The data values flow to the indicator terminals, where they update the front panel indicators (**a+b** and **a-b** in the previous figure)

- ***Nodes***

Nodes are objects on the block diagram that have inputs and/or outputs and perform operations when a VI runs. They are analogous to statements, operators, functions, and subroutines in text-based programming languages. The Add and Subtract functions in the previous figure are examples of nodes.

- ***Wires***

You transfer data among block diagram objects through wires. In the previous figure, wires connect the control and indicator terminals to the Add and Subtract functions. Each wire has a single data source, but you can wire it to many VIs and functions that read the data. Wires are different colors, styles, and thicknesses, depending on their data types. A broken wire appears as a dashed black line with a red X in the middle. Broken wires occur for a variety of reasons, such as when you try to wire two objects with incompatible data types.

- ***Structures***

Structures are graphical representations of the loops and case statements of text-based programming languages. Use structures on the block diagram to repeat blocks of code and to execute code conditionally or in a specific order.

- ***Icon and Connector Pane***

After you build a VI front panel and block diagram, build the icon and the connector pane so you can use the VI as a subVI. The icon and connector pane correspond to the function prototype in text-based programming languages. Every VI displays an icon, such as the one shown as follows, in the upper right corner of the front panel and block diagram windows.

An icon is a graphical representation of a VI. It can contain text, images, or a combination of both. If you use a VI as a subVI, the icon identifies the subVI on the block diagram of the VI. You can double-click the icon to customize or edit it.You also need to build a connector pane, shown as follows, to use the VI as a subVI.

The connector pane is a set of terminals that correspond to the controls and indicators of that VI, similar to the parameter list of a function call in text-based programming languages. The connector pane defines the inputs and outputs you can wire to the VI so you can use it as a subVI. A connector pane receives data at its input terminals and passes the data to the block diagram code through the front panel controls and receives the results at its output terminals from the front panel indicators

<u>SCXI</u>

SCXI stands for signal conditioning extension for instrumentation. Signal conditioning is the general term for the process to make a signal more useful by improving precision,accuracy,signal to noise ratio.

TYPE OF SCXI

SCXI-1540:-The SCXI-1540 module is an eight channel module for interfacing with industry-standard LVDT,RVDT and resolvers.

SCXI-1520:-The SCXI-1520 module is an eight channel module for interfacing to strain-gauge bridges and other whaetstone bridge based sensors.

SCXI-1000:-This document explains how to install and configure SCXI signal conditioning modules in SCXI-1000,SCXI-1001,SCXI-1000DC AND PXI/SCXI combination chassis ,confirm the module and chassis are operating properly and setup multichassis systems.

<u>Scope of Work</u>

The essence of using the simulation of virtual instrument is that it is not physically implemented until it's rigorously tested and modified by the use of software. Hence it reduces the cost of implementation. Before the virtual instrument exploiting software came into practice, there had been a lump of money being wasted on testing and modification of an instrument because until an instrument is physically and completely developed there is no scope of receiving the output or generally results. There is a thick difference between the conceptual, i.e., theoretical and mathematical solutions, and the physically implemented device. Hence, if somehow the implemented device or instrument fails to meet the desired results or specified goals, the cost of development is a total wastage. And also there is no guarantee that the instrument will work properly in the next endeavors of implementation. But if the conceptual problems can be analyzed and tested virtually without being physically implemented the chances of failure reduces and thus in effect reduces the development costs. Assumptions and consideration taken, but was overlooked can be taken into account if there is an error found when simulating the virtual instruments through software. As there is no end of instruments, there is no end of scope, i.e., the range is wide!!!

SCXI Accelerometer Input Modules

NI SCXI-1530, NI SCXI-1531

- Programmable gain per channel (1,10,100)
- Programmable lowpass 4-pole Bessel filter per channel (2.5, 5, 10 and 20 kHz)
- 4 mA current source, 24 V compliance
- IEPE conditioning – software configurable
- Simultaneous sample and hold
- BNC direct connectivity
- Autocalibration
- Multiple inputs
    - 4 channels – SCXI-1530
    - 8 channels – SCXI-1531
- Random scanning
- NI-DAQ driver software simplifies configuration, scaling, and measuremen

Overview

The National Instruments SCXI-1530/1531 signal conditioning modules are designed for Integrated Electronic Piezoelectric (IEPE) accelerometers and microphones. These modules share a common architecture in which each input channel includes a programmable AC instrumentation amplifier, 4-pole Bessel lowpass filter, and excitation current source. These modules also offer simultaneous sampling to preserve interchannel phase relationships. Each module can multiplex its signals into a single channel of the DAQ device and you can add modules to increase channel count.With random scanning capabilities, you select only the channels from which you want to acquire data as well as scan channels in any order. These modules also offer parallelmode operation for faster scanning rates. Each module offers BNC connectors to simplify signal connection.

Analog Input

To simplify configuration and ensure measurement accuracy, the analog inputs of the NI SCXI-1530/1531 consist of programmable instrumentation amplifiers, 4-pole Bessel lowpass filters, and simultaneous sample and hold using track-and-hold (T/H) circuitry. You can program each channel individually for input ranges of ±10, ±1, or ±0.1 V. You can program the lowpass filter of each channel for 2.5, 5, 10, or 20 kHz. The 4-pole Bessel lowpass filters provide a sharp cutoff while maintaining interchannel phase information. Each channel also has a 0.2 Hz highpass filter to block the DC offset of IEPE sensors. All analog inputs offer T/H circuitry to perform simultaneous sampling. To determine the allowable scanning rate,

Conditioning

Each channel of the SCXI-1530/1531 has a 4 mA, 24 V compliant current source to power IEPE accelerometers/ microphones. By using an active current source, the excitation current of the module remains constant regardless of loading by the sensor.You can programmatically disable the current source for each channel. This is helpful for measuring the system noise of your environment, or using selected input channels for other input signal types.

SCXI 8-Channel LVDT Input Module

NI SCXI-1540
• 8 channels
• Programmable input range per channel (0.05 to 6 Vrms)
• Programmable 1 or 3 Vrms excitation per channel at 2.5, 3.3, 5, or 10 kHz
• Multichannel synchronization
• 333 kS/s maximum sampling rate (250 Hz output bandwidth)
• Autocalibration in case excitation required differs from level and frequency SCXI-1540 provides
• 4 or 5-wire connections
• Synchronization to external excitation frequency available
• Random scanning
• Lowpass filter (250 Hz)
• NI-DAQ driver software simplifies configuration, measurement, and scaling.

SCXI for LVDT Measurements

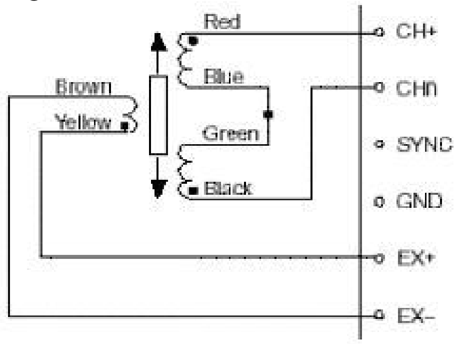
SCXI is a signal conditioning platform for PC-based data acquisition (DAQ) systems used in instrumentation applications. An SCXI system consists of a shielded chassis that houses a combination of signal conditioning input and output modules, which perform a variety of signal conditioning functions. You can connect many different types of transducers, including LVDTs, directly to SCXI modules. SCXI operates as a front-end signal conditioning system for PC plug-in DAQ devices (such as PCI or PCMCIA) or DAQ modules in PXI measurement and automation systems.

SCXI Signal Coditioning System

The National Instruments SCXI-1540 8-channel LVDT module provides the necessary conditioning to measure signals from transformer-based ratiometric position sensors, including LVDTs, rotary variable differential transformers (RVDTs), and resolvers. The SCXI-1540 offers both 4 and 5-wire connections for LVDTs and RVDTs. In addition, this module offers autocalibration without external hardware using NI-DAQ driver software. Each of these modules can multiplex its signals into a single channel of the DAQ device, and modules can be added to increase channel count. With random scanning capabilities, you can select only the channels from which you want to acquire data.

How to Connect LVDTs

LVDTs typically come in 4-wire (open wire) and 5-wire (ratiometric wire) configurations. Wires from the sensor connect to a signal conditioning circuit that translates the output of the LVDT to a measurable voltage. The two circuits in the figures below depict the external connections to the conditioning circuit.



Wire Connection of an LVDT to a Signal Conditioning Circuit

4-wire and 5-wire configurations are differentiated by the way the signals from the first and second secondaries are conditioned. In the 4-wire configuration, only the voltage difference between the two secondaries is measured. The following equation relates the measured voltage to the displacement, where G is the gain or sensitivity:

$$displacement = G \times (V_{CH+} - V_{CH-})$$

The benefit of using a 4-wire configuration is that you require a simpler signal conditioning system. "This is at the expense of temperature stability and phase coherence between the primary excitation voltage and the resulting secondary voltages. Temperature changes can alter the LVDT's magnetic induction efficiency. This causes a change in the perceived voltage for a given displacement. Because the 4-wire scheme is also sensitive to phase changes between the primary and the resulting secondary voltage, long wires or a poor excitation source can also cause problems."

The 5-wire configuration is less sensitive to both temperature changes and phase differences between the primary and the secondaries. "The reason for the temperature stability lies in the fact that the voltage changes due to the changes in magnetic induction efficiency affect voltages VCH+ and VCH- equally with respect to ground and thus null the effects of temperature."[2] Similarly, phase information is determined at the signal conditioning circuitry without needing to reference the phase of the primary excitation source. Therefore, longer wires can be used between the LVDT and the signal conditioning circuitry. The following equation relates the measured voltage to the displacement, where G is the gain or sensitivity:

$$displacement = G \times \frac{(V_{CH+} - V_{CH-})}{(V_{CH+} + V_{CH-})}$$

Each of the eight analog inputs consists of an instrumentation amplifier, a variable gain stage, a demodulation circuit, and a 250 Hz lowpass filter. Excitation voltage can be set for 1 or 3 Vrms and a frequency of 2.5, 3.3, 5 or 10 kHz.

About my work

Whole experiment can be Actually the whole experiment can be done using by Image prcessing because labview has one of the feature sense and vision.then I try to do this experiment otherwisely,as digital interfacing with NIelvis series-II.Using this NIELVIS

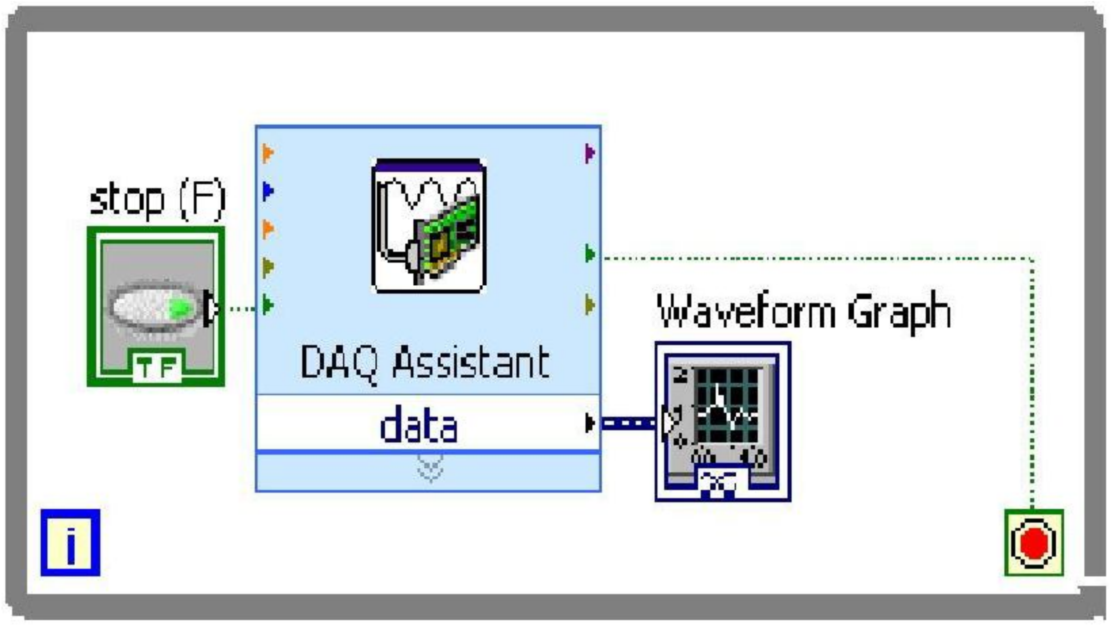
SERIES-II,first resistor circuit create using multisim and interface with my PC.Then I try to do some difficult circuit such as five stage amplifier,I also do this circuit using multisim.Then I study the whole configuration of NI ELVIS SERIES –II,and try to do some voltage measurement using this NI-ELVIS SERIS –II INSTRUMENT,then also do some current measurement.

After that I want to some work such as those multisim creating circuits I want to interface with NI ELVIS SERIES –II Instrument.then I want to do interface with my PC,then also interface with labview ,I configure the circuit diagram in the front panel ,then I watch the o/p of the circuit.

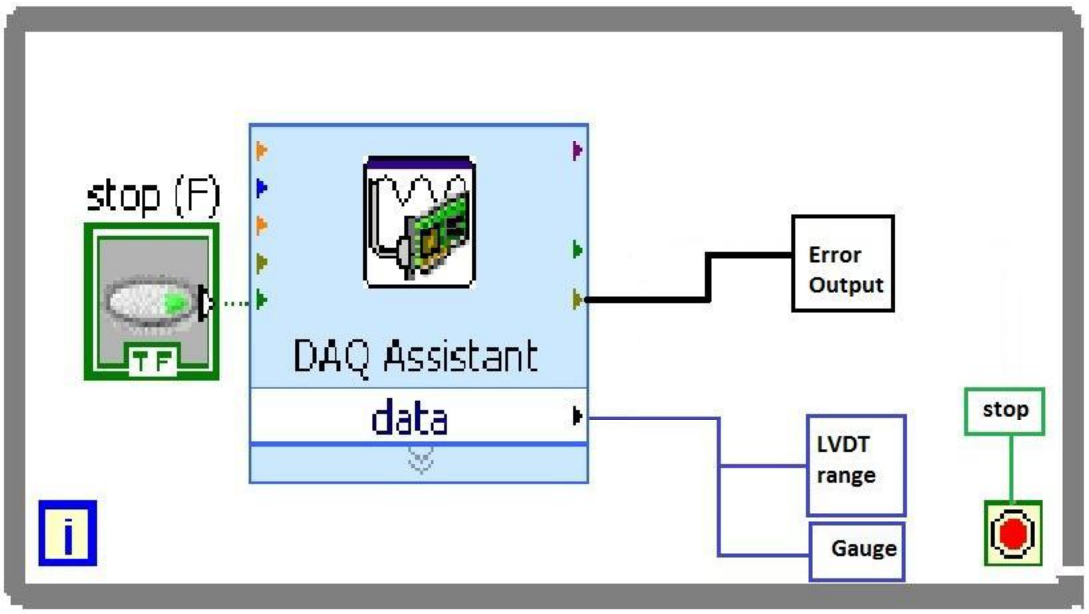
In the other way I will connect with some NIelvis series-ii,and I also inter face with my PC,then I will see the result,for which voltage change or which current change ,I will get what result.From this result I will get a clear idea what will be the characteristics and some output oriented result .From that I will get a clear idea ,using this I go forward with this experiment.

In the same way I will do my experimental components Loadcell,LVDT,Accelerometer,I will first configure them in the front panel of the labview,then I will ty to see the output of the each circuit configure separately.

<u>BLOCK DIAGRAM OF LOADCELL,LVDT,ACCELEROMETER</u>



Block diagram of loadcell, accelerometer



Block diagram of LVDT

<u>DAQ Assistant</u>

LabVIEW has the capability to connect and interact with a large number of hardware devices. This modules introduces you to two Express VIs that make acquiring data and communicating with traditional, third-party instruments easier.

Acquiring a Signal in NI-DAQmx

You will use the DAQ Assistant Express VI to create a task in NI-DAQmx. NI-DAQmx is a programming interface you can use to communicate with data acquisition devices. Refer to the **Getting Started with LabVIEW»Getting Started with DAQ»Taking an NI-DAQmx Measurement in LabVIEW** book on the **Contents** tab in the *LabVIEW Help* for information about additional ways to create NI-DAQmx tasks.

Creating an NI-DAQmx Task

In NI-DAQmx, a task is a collection of one or more channels, which contains timing, triggering, and other properties. Conceptually, a task represents a measurement or generation you want to perform. For example, you can create a task to measure temperature from one or more channels on a DAQ device.

Complete the following steps to create and configure a task that reads a voltage level from a DAQ device.

1. Open a new, blank VI.
2. On the block diagram, display the **Functions** palette and select **Express»Input** to display the **Input** palette.
3. Select the DAQ Assistant Express VI, shown below, on the **Input** palette and place it on the block diagram. The DAQ Assistant launches and the **Create New Express Task** dialog box appears.

4. Click **Acquire Signals»Analog Input** to display the **Analog Input** options.
5. Select **Voltage** to create a new voltage analog input task.The dialog box displays a list of channels on each installed DAQ device. The number of channels listed depends on the number of channels you have on the DAQ device.
6. In the **Supported Physical Channels** list, select the physical channel to which the device connects the signal, such as **ai0**, and then click the **Finish** button. The DAQ Assistant opens a new dialog box, shown in the following figure, that displays options for configuring the channel you selected to complete a task.
7. In the DAQ Assistant dialog box select the **Configuration** tab and locate the **Voltage Input Setup** section.
8. Locate the **Settings** tab. In the **Signal Input Range** section enter 10 for the **Max** value and enter -10 for the **Min** value.
9. Locate the **Timing Settings** section at the bottom of the **Configuration** page. From the **Acquisition Mode** pull-down menu, select **N Samples**.
10. Enter a value of 1000 in the **Samples to Read** text box.
11. Click the **OK** button to save the current configuration and close the DAQ Assistant. LabVIEW builds the VI.

Graphing Data from a DAQ Device

Complete the following steps to plot the data from the channel on a waveform graph and change the name of the signal.

1. On the block diagram, right-click the **data** output and select **Create»Graph Indicator** from the shortcut menu.
2. Display the front panel and run the VI three or four times. Observe the waveform graph. **Voltage** appears in the plot legend at the top of the waveform graph.
3. On the block diagram, right-click the DAQ Assistant Express VI and select **Properties** from the shortcut menu to open the DAQ Assistant.
4. Right-click **Voltage** in the list of channels and select **Rename** from the shortcut menu to display the **Rename a channel or channels** dialog box.
5. In the **New Name** text box, enter First Voltage Reading, and click the **OK** button.
6. In the **DAQ Assistant** dialog box, click the **OK** button to save the current configuration and close the DAQ Assistant.
7. Display the front panel and run the VI. **First Voltage Reading** appears in the waveform graph plot legend.
8. Save the VI.

<u>Why While Loops:</u>

Similar to a Do Loop or a Repeat-Until Loop in text-based programming languages, a While Loop, shown at left, executes a subdiagram until a condition is met. The While Loop executes the subdiagram until the conditional terminal, an input terminal, receives a specific Boolean value. The default behavior and appearance of the conditional terminal is **Stop if True**, shown at left. When a conditional terminal is **Stop if True**, the While Loop executes its subdiagram until the conditional terminal receives a TRUE value. You can change the behavior and appearance of the conditional terminal by right-clicking the terminal or the border of the While Loop and selecting **Continue if True**, shown at left, from the shortcut

menu. When a conditional terminal is **Continue if True**, the While Loop executes its subdiagram until the conditional terminal receives a FALSE value. You also can use the Operating tool to click the conditional terminal to change the condition.
You also can perform basic error handling using the conditional terminal of a While Loop. When you wire an error cluster to the conditional terminal, only the TRUE or FALSE value of the **status** parameter of the error cluster passes to the terminal. Also, the **Stop if True** and **Continue if True** shortcut menu items change to **Stop if Error** and **Continue while Error**.
The iteration terminal (an output terminal), shown at left, contains the number of completed iterations. The iteration count always starts at zero. During the first iteration, the iteration terminal returns **0**. Add shift registers to the While Loop to pass data from the current iteration to the next iteration. Refer to the *Shift Registers and the Feedback Node in Loops* section of this chapter for more information about adding shift registers to a loop.