SEMESTER VII

16BEBME702A VIRTUALINSTRUMENTATION DESIGN FOR MEDICAL SYSTEM **3 0 0 3 100**

OBJECTIVE:

- To introduce the students to the basics of virtual instrumentation.

INTENDED OUTCOMES:

- Make use of analysis tools
- Use the programming techniques.

UNIT I REVIEW OF VIRTUAL INSTRUMENTATION (9)

Historical perspectives, Need of VI, advantages, Define VI, block diagram and architecture of a virtual instrument, data -flow techniques, graphical programming in data flow, comparison with conventional programming.

UNIT II VI PROGRAMMING TECHNIQUES (9)

VIS and sub-VIS loops and charts, arrays, clusters and graphs, case and sequence structures, formula nodes, local and global variables, string and file I/O, Graphical programming in data flow, comparison with conventional programming.

UNIT III DATA ACQUISITION BASICS (9)

ADC, DAC, DIO, Counters & timers, PC Hardware structure, timing, interrupts, DMA, Software and Hardware Installation. GPIB/IEEE 488 concepts, and embedded system buses - PCI, EISA, CPCI, and USB & VXI. A

UNIT IV COMMON INSTRUMENT INTERFACES (9)

Current loop, RS.232C/RS.485, GPIB, System buses, interface buses: USB, PCMCIA, VXI, SCXI, PXI, etc., networking basics for office &.Industrial applications, Visa and lVI, image acquisition and processing. Motion control. ADC, DAC, DIO, DMM, waveform generator.

UNIT V USE OF ANALYSIS TOOLS (9)

Fourier transforms, power spectrum correlation methods, windowing & filtering, Major equipments- Oscilloscope, Digital Multimeter, Pentium Computers, Application in Biomedical field

Total : 45

TEXT BOOKS:

S.NO.	Author(s) Name	Title of the book	Publisher	Year of publication
1	Gary Jonson	Labview Graphical Programming	Second Edition, McGraw Hill, NewYork	1997
2	Lisa K.wells & Jeffrey Travis	Labview for everyone	, Prentice Hall Inc., New Jersey	1997

REFERENCES:

S.NO.	Author(s) Name	Title of the book	Publisher	Year of publication
1	Sokol off	Basic concepts of Labview 4	Prentice Hall Inc., New Jersey	1998
2	S.Gupta, J.P: Gupta	PC interfacing for Data Acquisition & Process Control	Instrument Society of America	1994
3	L.T.Amy	Automation System for Control and Data Acquisition	ISA	1992

KARPAGAM ACADEMY OF HIGHER EDUCATION
COIMBATORE-21
Faculty of Engineering
Department of Biomedical Engineering

LECTURE PLAN

Name of the staff : **M.Bhuvaneswari**
Designation : **Assistant Professor.**
Class : **IV-B.E BME**
Subject : **Virtual Instrumentation Design For Medical Systems**
Subject code : **16BEBME702A**

Sl.No.	Topics to be covered	Time Duration	Teaching aids
	INTRODUCTION		
1	Introduction to Virtual Instrumentation	1	
	UNIT-I REVIEW OF VIRTUAL INSTRUMENTATION		
2	History, Need & Advantages of VI	01	T1 4-7
3	VI Definition and Block Diagram	01	R4 5
4	Architecture of VI	01	W2
5	Data flow in LabVIEW	01	T1 38-42
6	Working in LabVIEW Environment	01	T1 43
7	Graphical Programming Language	01	R4 9
8	VI Comparison with Conventional Programming	01	W1
9	Eample Finder in LabVIEW	01	R4 10
10	Demonstration of example programs	01	R4
	UNIT-II VI PROGRAMMING TECHNIQUES		
11	VIS and sub-VIS	01	W1
12	Loops in LabVIEW	01	T113,14
13	Arrays & clusters with Eamples	01	T1 114-124
14	Charts & graphs	01	T1 506,556&563
15	Case Structure	01	T1 432
16	Sequence structures & formula nodes	01	T1 61-62,w2
17	String I/O		W1
18	local and global variables	01	T1 71-78
19	File I/O	01	T1 167
	UNIT-III DATA ACQUISITION BASICS		
20	ADC, DAC	01	T1 309-315
21	DIO	01	T1 297-298
22	Counters & timers	01	T1 137-149
23	PC Hardware structure, timing, interrupts		
24	DMA, Software and Hardware Installation	01	T1 5,320,373-374
25	GPIB/IEEE 488 concepts	01	T1 9,245
26	Embedded system buses – PCI, CPCI,	01	R4 38,W5
27	USB	01	R4 28,W5
28	EISA, VXI. A	01	W5
	UNIT-III COMMON INSTRUMENT INTERFACES		
29	Current loop, RS.232C/RS.485	01	W1
30	GPIB, System buses, interface buses: USB, PCMCIA	01	R4 26
31	VXI, SCXI, PXI, etc	01	R4 32-34
32	Networking basics for office	01	W1

33	Industrial applications of LabVIEW	01	W2
34	Visa and lVI	02	T1 344-350
35	Image acquisition and processing	01	T1 572
36	Motion control.	01	T1 520-523
37	ADC, DAC, DIO, DMM, waveform generator		T1 309-314
UNIT-V USE OF ANALYSIS TOOLS			
38	Fourier transforms	01	T1 281
39	power spectrum	01	W1
40	correlation methods	01	T1 306
41	windowing & filtering	01	T1 344-350
42	Major equipments- Oscilloscope	01	T1 536, W1
43	Digital Multimeter	01	W4
44	Pentium Computers	01	W6
45	Application in Biomedical field	02	W7

Website:

W1- http://www.ni.com/pdf/manuals/373427j.pdf
W2- http://www.ni.com/en-us/shop/labview.html
W3- https://knowledge.ni.com/KnowledgeArticleDetails -IN
W4- https://zone.ni.com/reference/
W5-https://turbofuture.com/computers/buses
W6- https://www.computerhope.com/jargon/p/pentium.htm
W7- http://biolabview.blogspot.com/

TEXT BOOK

Sl.No.	Author(s)	Title of the Book	Publisher	Year of Publication
1	Gary Jonson	Labview Graphical Programming	Second Edition, McGraw Hill, NewYork	1997
2	Lisa K.wells & Jeffrey Travis	Labview for everyone	, Prentice Hall Inc., New Jersey	1997

REFERENCES

Sl.No.	Author(s)	Title of the Book	Publisher	Year of Publication
1	Sokol off	Basic concepts of Labview 4	Prentice Hall Inc., New Jersey	1998
2	S.Gupta, J.P: Gupta	PC interfacing for Data Acquisition & Process Control	Instrument Society of America	1994
3	L.T.Amy	Automation System for Control and Data Acquisition	ISA	1992
4	Jeffrey Travis,Jim Kring	LabVIEW for Everyone	Pearson Education	2009

Staff In-Charge **HOD/ECE**

UNIT I

Historical perspective, advantages, block diagram and architecture of a virtual instrument, data-flow techniques, graphical programming in data flow, comparison with conventional programming

INTRODUCTION

An instrument is a device designed to collect data from an environment, or from a unit under test, and to display information to a user based on the collected data. Such an instrument may employ a transducer to sense changes in a physical parameter, such as temperature or pressure, and to convert the sensed information into electrical signals, such as voltage or frequency variations. The term instrument may also be defined as a physical software device that performs an analysis on data acquired from another instrument and then outputs the processed data to display or recording devices. This second category of recording instruments may include oscilloscopes, spectrum analyzers, and digital millimeters. The types of source data collected and analyzed by instruments may thus vary widely, including both physical parameters such as temperature, pressure, distance, frequency and amplitudes of light and sound, and also electrical parameters including voltage, current, and frequency.

Virtual instrumentation is an interdisciplinary field that merges sensing, hardware, and software technologies in order to create flexible and sophisticated instruments for control and monitoring applications. The concept of virtual instrumentation was born in late 1970s, when microprocessor technology enabled a machine's function to be more easily changed by changing its software. The flexibility is possible as the capabilities of a virtual instrument depend very little on dedicated hardware – commonly, only application specific signal conditioning module and the analog-to-digital converter used as interface to the external world. Therefore, simple use of computers or specialized onboard processors in instrument control and data acquisition cannot be defined as virtual instrumentation. Increasing number of biomedical applications use virtual instrumentation to improve insights into the underlying nature of complex phenomena and reduce costs of medical equipment and procedures.

History of Instrumentation Systems

Historically, instrumentation systems originated in the distant past, with measuring rods, thermometers, and scales. In modern times, instrumentation systems have generally consisted of individual instruments, for example, an electromechanical pressure gauge comprising a sensing transducer wired to signal conditioning circuitry, outputs a processed signal to a display panel and perhaps also to a line recorder, in which a trace of changing conditions is linked onto a rotating drum by a mechanical arm, creating a time record of pressure changes. Complex systems such as chemical process control applications employed until the 1980s consisted of sets of individual physical instrumentswired to a central control panel that comprised an array of physical datadisplay devices such as dials and counters, together with sets of switches,knobs, and buttons for controlling the instruments.A history of virtual instrumentation is characterized by continuous increaseof flexibility and scalability of measurement equipment. Starting from firstmanual-controlled vendor-defined electrical instruments, the instrumentationfield has made a great progress toward contemporary computer-controlled,user-defined, sophisticated measuring equipment. Instrumentation had the followingphases:

– Analog measurement devices

– Data acquisition and processing devices

– Digital processing based on general purpose computing platform

– Distributed virtual instrumentation

The first phase is represented by early "pure" analog measurement devices, such as oscilloscopes or EEG recording systems. They were completely closed dedicated systems, which included power suppliers, sensors, translators, and displays. They required manual settings, presenting results on various counters, gauges, CRT displays, or on the paper. Further use of data was not part of the instrument package, and an operator had to physically copy data to a paper notebook or a data sheet. Performing complex or automated test procedures was rather complicated or impossible, as everything had to be set manually.

Second phase started in 1950s, as a result of demands from the industrial control field. Instruments incorporated rudiment control systems, with relays, rate detectors, and integrators. That led to creation of proportional– integral–derivative (PID) control systems, which allowed greater flexibility of test procedures and automation of some phases of measuring process. Instruments started to digitalize measured signals, allowing digital processing of data, and introducing more complex control or analytical decisions. However, real-time digital processing requirements were too high for any but an onboard special purpose computer or digital signal processor (DSP). The instruments still were standalone vendor defined boxes.

In the third phase, measuring instruments became computer based. They began to include interfaces that enabled communication between the instrument and the computer. This relationship started with the general-purpose interface bus (GPIB) originated in 1960s by Hewlett-Packard (HP), then called HPIB, for purpose of instrument control by HP computers. Initially, computers were primarily used as off-line instruments. They were further processing the data after first recording the measurements on disk or type. As the speed and capabilities of general-purpose computers advanced exponentially general-purpose computers became fast enough for complex real-time measurements. It soon became possible to adapt standard, by now high-speed computers, to the online applications required in real-time measurement and control. New general-purpose computers from most manufactures incorporated all the hardware and much of the general software required by the instruments for their specific purposes. The main advantages of standard personal computers are low price driven by the large market, availability, and standardization. Although computers' performance soon became high enough, computers were still not easy to use for experimentalists.

Nearly all of the early instrument control programs were written in BASIC, because it had been the dominant language used with dedicated instrument controllers. It required engineers and other users to become programmers before becoming instrument users, so it was hard for them to exploit potential that computerized instrumentation could bring. Therefore, an important milestone in the history of virtual instrumentation happened in 1986, when National Instruments introduced LabVIEW 1.0 on a PC platform. LabVIEW introduced graphical user interfaces and visual programming into computerized instrumentation, joining simplicity of a user interface operation with increased capabilities of computers. Today, the PC is the platform on which most measurements are made, and the graphical user interface has made measurements user-friendlier. As a result, virtual instrumentation made possible decrease in price of an instrument. As the virtual instrument depends very little on dedicated hardware, a customer could now use his own computer, while an instrument manufactures could supply only what the user could not get in the general market.

The fourth phase became feasible with the development of local and global networks of general purpose computers. Since most instruments were already computerized, advances in telecommunications and network technologies made possible physical distribution of virtual instrument components into telemedical systems to provide medical information and services at a distance.

Possible infrastructure for distributed virtual instrumentation includes the Internet, private networks, and cellular networks, where the interface between the components can be balanced for price and performance.

The introduction of computers into the field of instrumentation began asa way to couple an individual instrument, such as a pressure sensor, to acomputer, and enable the display of measurement data on virtual instrumentpanel on the computer screen using appropriate software. The instrumentalso contained buttons for controlling the operation of the sensor. Thus, suchinstrumentation software enabled the creation of a simulated physical instrument,having the capability to control physical sensing components.

Virtual Instrumentation

Virtual instrumentation achieved mainstream adoption by providing a new model for building measurement and automation systems. Keys to its success include rapid PC advancement; explosive low-cost, high-performance data converter (semiconductor) development; and system design software emergence. These factors make virtual instrumentation systems accessible to a very broad base of users.

Definition

A virtual instrumentation system is software that is used by the user to develop a computerized test and measurement system, for controlling an external measurement hardware device from a desktop computer, and for displaying test or measurement data on panels in the computer screen. The test and measurement data are collected by the external device interfaced with the desktop computer. Virtual instrumentation also extends to computerized systems for controlling processes based on the data collected and processed by a PC based instrumentation system.

Block diagram and architecture of a virtual instrument

A virtual instrument is composed of the following blocks:

– Sensor module

– Sensor interface

– Information systems interface

– Processing module

– Database interface

– User interface

Figure shows the general architecture of a virtual instrument. The sensor module detects physical signal and transforms it into electrical form, conditions the signal, and transforms it into a digital form for further manipulation. Through a sensor interface, the sensor module communicates with a computer. Once the data are in a digital form on a computer, they can be processed, mixed, compared, and otherwise manipulated, or stored in a database.

Then, the data may be displayed, or converted back to analog form for further process control. Virtual instruments are often integrated with some other information systems. In this way, the configuration settings and the

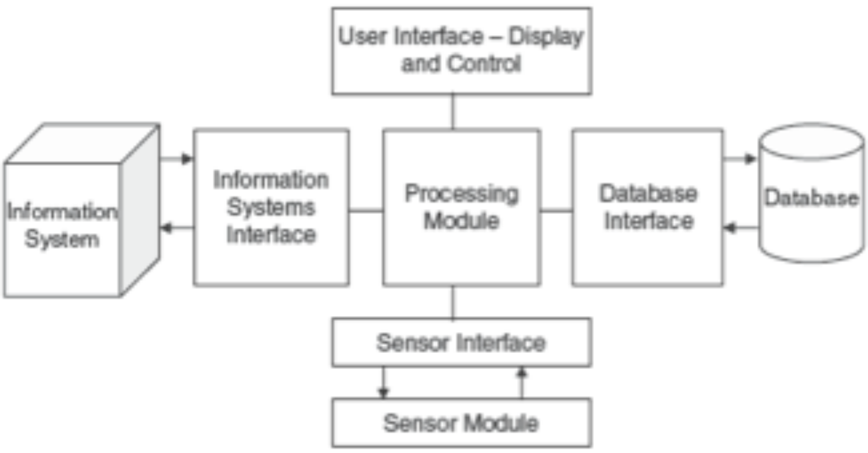


Fig 1: Architecture of a virtual instrument

data measured may be stored and associated with the available records. In following sections each of the virtual instruments modules are described in more detail.

Sensor Module

The sensor module performs signal conditioning and transforms it into a digital form for further manipulation. Once the data are in a digital form on a computer, they can be displayed, processed, mixed, compared, stored in a database, or converted back to analog form for further process control. The database can also store configuration settings and signal records. The sensor module interfaces a virtual instrument to the external, mostly analog world transforming measured signals into computer readable form. A sensor module principally consists of three main parts:

– The sensor

– The signal conditioning part

– The A/D converter

The sensor detects physical signals from the environment. If the parameter being measured is not electrical, the sensor must include a transducer to convert the information to an electrical signal, for example, when measuring blood pressure.

The signal-conditioning module performs (usually analog) signal conditioning prior to AD conversion. This module usually does the amplification, transducer excitation, linearization, isolation, or filtering of detected signals.

The A/D converter changes the detected and conditioned voltage into a digital value. The converter is defined by its resolution and sampling frequency. The converted data must be precisely time-stamped to allow later sophisticated analyses. Although most biomedical sensors are specialized in processing of certain signals, it is possible to use generic measurement components, such as data acquisition (DAQ), or image acquisition (IMAQ) boards, which may be applied to broader class of signals. Creating generic measuring board, and incorporating the most important components of different sensors into one unit, it is possible to perform the functions of many medical instruments on the same computer.

Sensor Interface

There are many interfaces used for communication between sensors modules and the computer. According to the type of connection, sensor interfaces can be classified as wired and wireless.

– Wired Interfaces are usually standard parallel interfaces, such as GPIB, Small Computer Systems Interface (SCSI), system buses (PCI eXtension for Instrumentation PXI or VME Extensions for Instrumentation (VXI), or serial buses (RS232 or USB interfaces).

– Wireless Interfaces are increasingly used because of convenience. Typical interfaces include 802.11 family of standards, Bluetooth, or GPRS/GSM interface. Wireless communication is especially important for implanted sensors where cable connection is impractical or not possible. In addition, standards, such as Bluetooth, define a self-identification protocol, allowing the network to configure dynamically and describe itself. In this way, it is possible to reduce installation cost and create plug-and-play like networks of sensors. Device miniaturization allowed development of Personal Area Networks (PANs) of intelligent sensors Communication with medical devices is also standardized with the IEEE 1073 family of standards. This interface is intended to be highly robust in an environment where devices are frequently connected to and disconnected from the network.

Processing Module

Integration of the general purpose microprocessors/microcontrollers allowed flexible implementation of sophisticated processing functions. As the functionality of a virtual instrument depends very little on dedicated hardware, which principally does not perform any complex processing, functionality and appearance of the virtual instrument may be completely changed utilizing different processing functions. Broadly speaking, processing function used in virtual instrumentation may be classified as analytic processing and artificial intelligence techniques.

– Analytic processing. Analytic functions define clear functional relations among input parameters. Some of the common analyses used in virtual instrumentation include spectral analysis, filtering, windowing, transforms, peak detection, or curve fitting. Virtual instruments often use various statistics function, such as, random assignment and biostatistical analyses. Most of those functions can nowadays be performed in real-time.

– Artificial intelligence techniques. Artificial intelligence technologies could be used to enhance and improve the efficiency, the capability, and the features of instrumentation in application areas related to measurement, system identification, and control. These techniques exploit the advanced computational capabilities of modern computing systems to manipulate the sampled input signals and extract the desired measurements. Artificial intelligence technologies, such as neural networks, fuzzy logic and expert systems, are applied in various applications, including sensor fusion to high-level sensors, system identification, prediction, system control, complex measurement procedures, calibration, and instrument fault detection and isolation. Various nonlinear signal processing, including fuzzy logic and neural networks, are also common tools in analysis of biomedical signals. Using artificial intelligence it is even possible to add medical intelligence to ordinary user interface devices. For example, several artificial intelligence techniques, such as pattern recognition and machine learning, were used in a software-based visual-field testing system.

Database Interface

Computerized instrumentation allows measured data to be stored for off-line processing, or to keep records as a part of the patient record. There are several currently available database technologies that can be used for this purpose.Simple usage of file systems interface leads to creation of many proprietaryformats, so the interoperability may be a problem. The eXtensible MarkupLanguage

(XML) may be used to solve interoperability problem by providinguniversal syntax. The XML is a standard for describing document structureand content. It organizes data using markup tags, creating self-describing documents,as tags describe the information it contains. Contemporary databasemanagement systems such SQL Server and Oracle support XML import andexport of data. Many virtual instruments use DataBase Management Systems(DBMSs). They provide efficient management of data and standardized insertion,update, deletion, and selection. Most of these DBMSs provided StructuredQuery Language (SQL) interface, enabling transparent execution of thesame programs over database from different vendors. Virtual instruments usethese DMBSs using some of programming interfaces, such as ODBC, JDBC,ADO, and DAO.

Information System Interface

Virtual instruments are increasingly integrated with other medical information systems, such as hospital information systems. They can be used to create executive dashboards, supporting decision support, real time alerts, and predictive warnings. Some virtual interfaces toolkits, such as LabVIEW, provide mechanisms for customized components, such as ActiveX objects, that allows communication with other information system, hiding details of the communication from virtual interface code. In Web based applications this integration is usually implemented using Unified Resource Locators (URLs). Each virtual instrument is identified with its URL, receiving configuration settings via parameters. The virtual instrument then can store the results of the processing into a database identified with its URL.

Presentation and Control

An effective user interface for presentation and control of a virtual instrumentaffects efficiency and precision of an operator do the measurements andfacilitates result interpretation. Since computer's user interfaces are much easiershaped and changed than conventional instrument's user interfaces, it ispossible to employ more presentation effects and to customize the interfacefor each user. According to presentation and interaction capabilities, we canclassify interfaces used in virtual instrumentation in four groups:

– Terminal user interfaces
– Graphical user interfaces
– Multimodal user interfaces and
– Virtual and augmented reality interfaces

Terminal User Interfaces

First programs for instrumentation control and data acquisition hadcharacter-oriented terminal user interfaces. This was necessary as earliergeneral-purpose computers were not capable of presenting complex graphics.As terminal user interfaces require little of system resources, they were implementedon many platforms. In this interfaces, communication between auser and a computer is purely textual. The user sends requests to the computertyping commands, and receives response in a form of textual messages.

Graphical User Interfaces

Graphical user interfaces (GUIs) enabled more intuitive human–computerinteraction, making virtual instrumentation more accessible. Simplicity ofinteraction and high intuitiveness of graphical user interface operations madepossible creation of user-friendlier virtual instruments. GUIs allowed creationof many sophisticated graphical widgets such as graphs, charts, tables, gauges,or meters, which can easily be created with many user interface tools.

Multimodal Presentation

In addition to graphical user interfaces that improve visualization, contemporarypersonal computers are capable of presenting other modalities suchas sonification or haptic rendering. Multimodal combinations of complementarymodalities can greatly improve the perceptual quality of user interfaces.

Virtual Instruments versus Traditional Instruments

Stand-alone traditional instruments such as oscilloscopes and waveform generators are very powerful, expensive, and designed to perform one or more specific tasks defined by the vendor. However, the user generally cannot extend or customize them. The knobs and buttons on the instrument, the built-in circuitry, and the functions available to the user, are all specific to the nature of the instrument. In addition, special technology and costly components must be developed to build these instruments, making them very expensive and slowto adapt.

Traditional Instruments	Virtual Instruments
Vendor defined	User-defined
Function specific,stand alone with limited connectivity	Application oriented system with connectivity to networks,peripherals, and applications
Hardware is the key	Software is the key
Expensive	Low cost , reusable
Closed, fixed functionality	Open, flexible functionality
Slow turn on technology	Fast turn on technology
Minimal economics of scale	Maximum economics of scale
High development and maintenance cost	Software minimizes development and maintenance cost

Table 1: Traditional Instruments Vs Virtual Instruments

Advantages of VI

The virtual instruments running on notebook automatically incorporate their portable nature to the Engineers and scientists whose needs, applications and requirements change very quickly, need flexibility to create their own solutions. We can adapt a virtual instrument to our particular needs without having to replace the entire device because of the application software installed on the PC and the wide range of available plug-in hardware.

Performance

In terms of performance, LabVIEW includes a compiler that produces nativecode for the CPU platform. The graphical code is translated into executablemachine code by interpreting the syntax and by compilation. The LabVIEWsyntax is strictly enforced during the editing process and compiled into theexecutable machine code when requested to run or upon saving. In the lattercase, the executable and the source code are merged into a single file. Theexecutable runs with the help of the LabVIEW run-time engine, which containssome precompiled code to perform common tasks that are defined bythe G language. The run-time engine reduces compile time and also providesa consistent interface to various operating systems, graphic systems, hardwarecomponents, etc.

Platform-Independent Nature
A benefit of the LabVIEW environment is the platform-independent nature of the G-code, which is (with the exception of a few platform specific functions) portable between the different LabVIEW systems for different operating systems (Windows, MacOSX, and Linux). National Instruments is increasingly focusing on the capability of deploying LabVIEW code onto an increasing number of targets including devices like Pharlap OS-based LabVIEWrealtime controllers, PocketPCs, PDAs, Fieldpoint modules, and into FPGAs on special boards.

Flexibility
Except for the specialized components and circuitry found in traditional instruments, the general architecture of stand-alone instruments is very similar to that of a PC-based virtual instrument. Both require one or more microprocessors, communication ports (for example, serial and GPIB), and display capabilities, as well as data acquisition modules. These devices differ from one another in their flexibility and the fact that these devices can be modified and adapted to the particular needs.

Lower Cost
By employing virtual instrumentation solutions, lower capital costs, system development costs, and system maintenance costs are reduced, increasing the time to market and improving the quality of our own products.

Plug-In and Networked Hardware
There is a wide variety of available hardware that can either be plugged into the computer or accessed through a network. These devices offer a wide range of data acquisition capabilities at a significantly lower cost than that of dedicated devices.

The Costs of a Measurement Application
The costs involved in the development of a measurement application can be divided into five distinct areas composed of hardware and software prices and several time costs. The price of the hardware and software was considered as the single largest cost of their most recent test or measurement system. However, the cumulative time costs in the other areas make up the largest portion of the total application cost.

Reducing System Specification Time Cost

Deciding the types of measurements to take and the types of analysis to perform takes time. Once the user has set the measurement specifications, the user must then determine exactly the method to implement the measurement system. The time taken to perform these two steps equals the system specification time.

Lowering the Cost of Hardware and Software

The price of measurement hardware and software is undoubtedly the most visible cost of a data-acquisition system. Many people attempt to save money in this area without considering the effect on the total development cost.

Minimizing Set-Up and Configuration Time Costs

Once the users have specified and purchased measurement hardware, the real task of developing the application begins. However, the user must first install the hardware and software, configure any necessary setting and ensure that all pieces of the system function properly.

Dataflow Programming

Lab VIEW 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. For a dataflow-programming example, consider a block diagram that adds two numbers and then subtracts 50.00 from the result of the addition as illustrated in Fig. In this case, the block diagram executes from left to right, not because the objects are placed in that order, but because the Subtract function cannot execute until the Add function finishes executing and passes the data to the Subtract function. Remember that a node executes only when data are available at all of its input terminals, and it supplies data to its output terminals only when it finishes execution.

In the following example in Fig. 2.consider which code segment would execute first – the Add, Random Number, orDivide function. In a situation where one code segment must execute beforeanother and no data dependency exists between the functions, the user canuse other programming methods, such as error clusters, to force the order ofexecution.

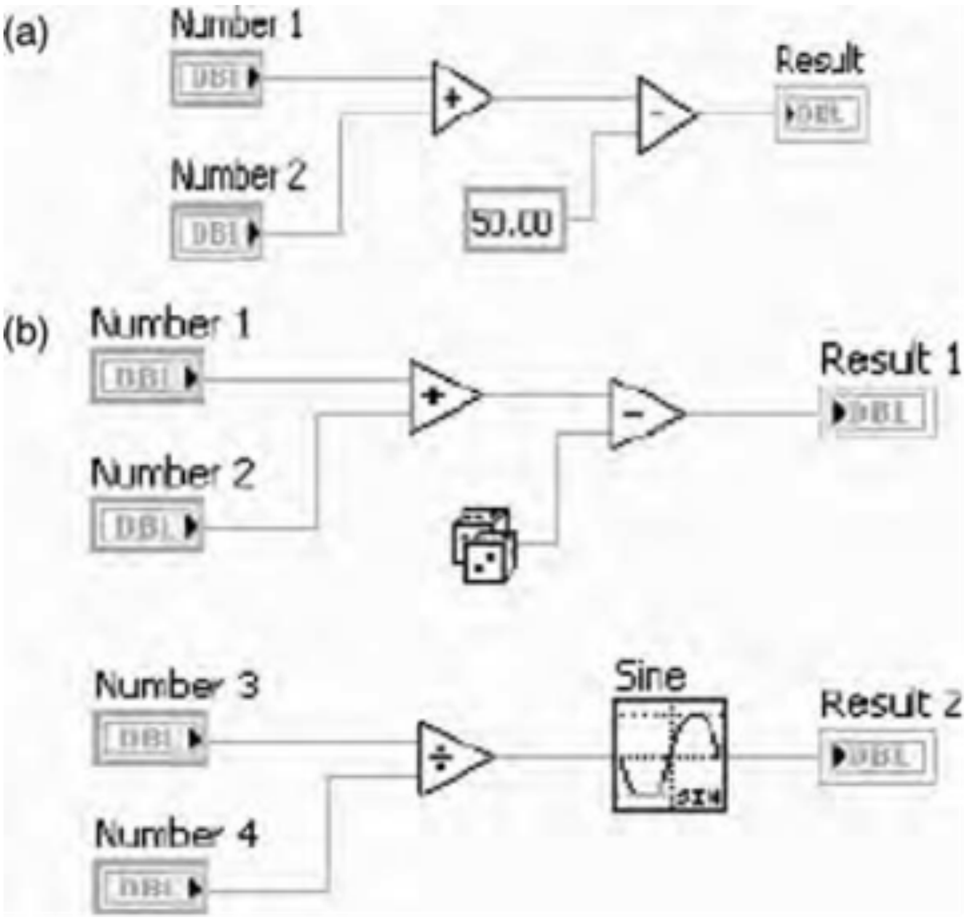


Fig. 2. Dataflow programming example

‘G’ Programming

The ‘G’ sequence structure is used to control execution order when natural data dependency does not exist. The user also can create an artificial data dependency in which the receiving node does not actually use the data received. Instead, the receiving node uses the arrival of data to trigger its execution. The programming language used in LabVIEW, called “G”, is a dataflow language. Execution is determined by the structure of a graphical block diagram (the LV-source code) on which the programmer connects different function-nodes by drawing wires. These wires propagate variables and any node can execute as soon as all its input data become available. Since this might be the case for multiple nodes simultaneously, “G” is inherently capable of parallel execution. Multiprocessing and multi-threading hardware is automatically exploited by the built-in scheduler, which multiplexes multiple OS threads over the nodes ready for execution. Programmers with a background in conventional programming often show a certain reluctance to adopt the LabVIEWdataflow scheme, claiming that LabVIEW is prone to race conditions. In reality, this stems from a misunderstanding of the data-flow paradigm. The afore-mentioned data-flow (which can be “forced,” typically by linking inputs and outputs of nodes) completelydefines the execution sequence, and that can be fully controlled by the programmer. The graphical approach also allows nonprogrammers to build programs by simply dragging and dropping virtual representations of the lab equipment with which they are already familiar. The LabVIEW programming environment, with the included examples and the documentation, makes iteasy to create small applications. This is a benefit on one side but there is also a certain danger of underestimating the expertise needed for good quality “G” programming. For complex algorithms or large-scale code it is important that the programmer understands the special LabVIEW syntax and the topology of its memory management well. The most advanced Lab-VIEW development systems offer the possibility of building standalone applications.

Virtual Instruments

Virtual Instruments are front panel and block diagram. The front panel oruser interface is built with controls and indicators. Controls are knobs, pushbuttons, dials, and other input devices. Indicators are graphs, LEDs, and otherdisplays.

Front Panel

The front panelis the window through which the user interacts with the program. The input data to the executing program is fed through the front panel and the output can also be viewed on the front panel, thus making it indispensable.

Front Panel Toolbar
Controls and Indicators

The front panel is primarily a combination of 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.

Fig 3: Front panel tool bars

S. No	Icon	Name	Meaning
1		**Run**	Used to run a VI. The VI runs if the **Run** button appears as a solid white arrow as shown at the left. The solid white arrow also indicates the VI can be used as a subVI if a connector pane for the VI is created.
2		**Run**	While the VI runs, the **Run** button appears as shown at left if the VI is a top-level VI, meaning it has no callers and therefore is not a subVI.
3		**Run**	If the VI that is running is a subVI, the **Run** button appears as shown at left.
4		**Run**	The **Run** button appears broken, shown at left, when the VI that is created and edited contains errors. If the **Run** button still appears broken after wiring of the block diagram is completed, then the VI is broken and cannot run. On clicking this button, an **Error list** window is displayed, which lists all errors and warnings.
5		**Run Continuously**	The **Run Continuously** button, shown at left, is used to run the VI until one abortion or pause execution.
6		**Abort Execution**	While the VI runs, the **Abort Execution** button, shown at left, appears. Click this button to stop the VI immediately if there is no other way to stop the VI. If more than one running top-level VI uses the VI, the button is dimmed.
7		**Pause**	Click the **Pause** button, shown at left, to pause a running VI. Upon clicking the **Pause** button, LabVIEW highlights on the block diagram the location where the execution was paused, and the **Pause** button appears red. Click the button again to continue running the VI.

Fig 4: Front Panel tool bars

Controls and Indicators :

Controls and Indicators are generally not interchangeable; the differenceshould be clear among the user. The user can "drop" controls and indicatorsonto the front panel by selecting them from a subpalette of the floating **Controls**palette window shown in Fig 5.

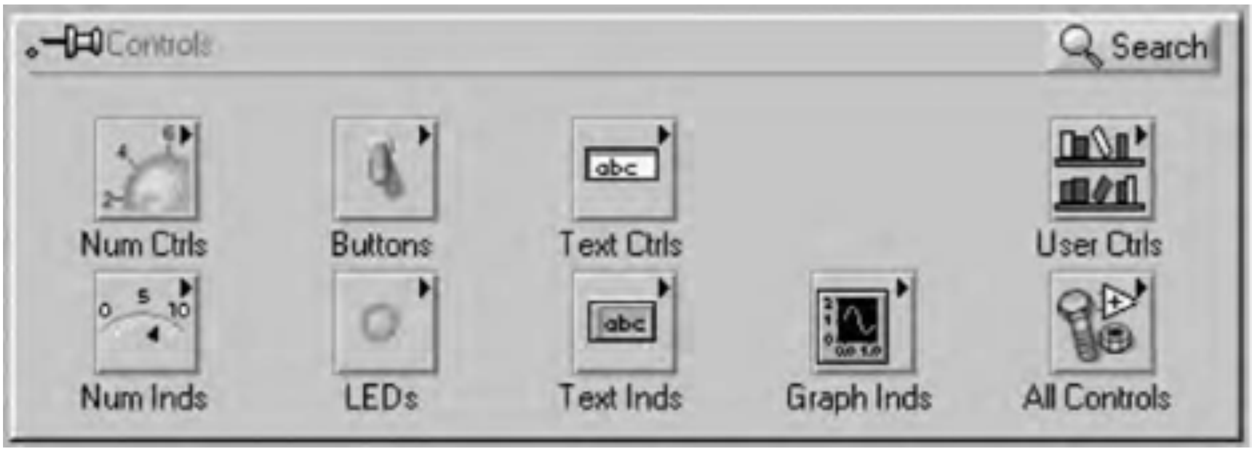
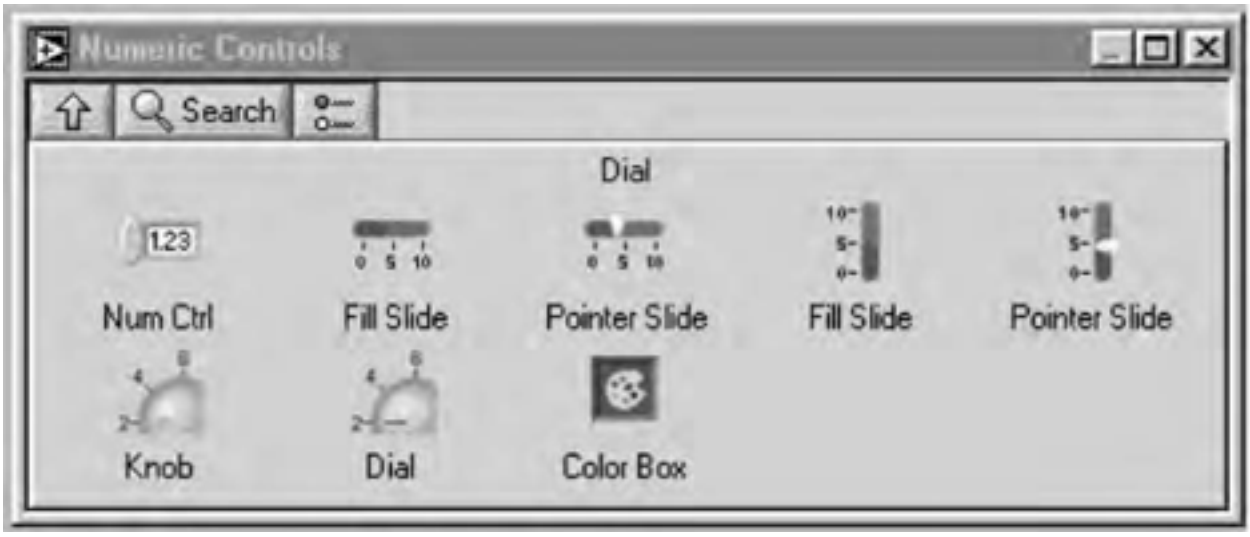


Fig5: Controls and Indicators

Once an object is on the front panel, the user can easily adjust its size, shape,position, color, and other attributes.Controls and indicators can be broadly classified as:
1. Numeric controls and indicators
2. Boolean controls and indicators
Numeric Controls and Indicators
The two most commonly used numeric objects are the numeric control and the numericindicator, as shown in Fig6. The values in a numeric control can be entered or changed by selecting the increment and decrement buttons with the Operating tool or double-clicking the number with either the Labeling tool or the Operating tool.
Boolean Controls and Indicators
The Boolean controls and indicators (Fig) are used to enter and display Boolean (True or false) values. Boolean objects simulate switches, push buttons, and LEDs. The most common Boolean objects are the vertical toggle switch and the round LED.



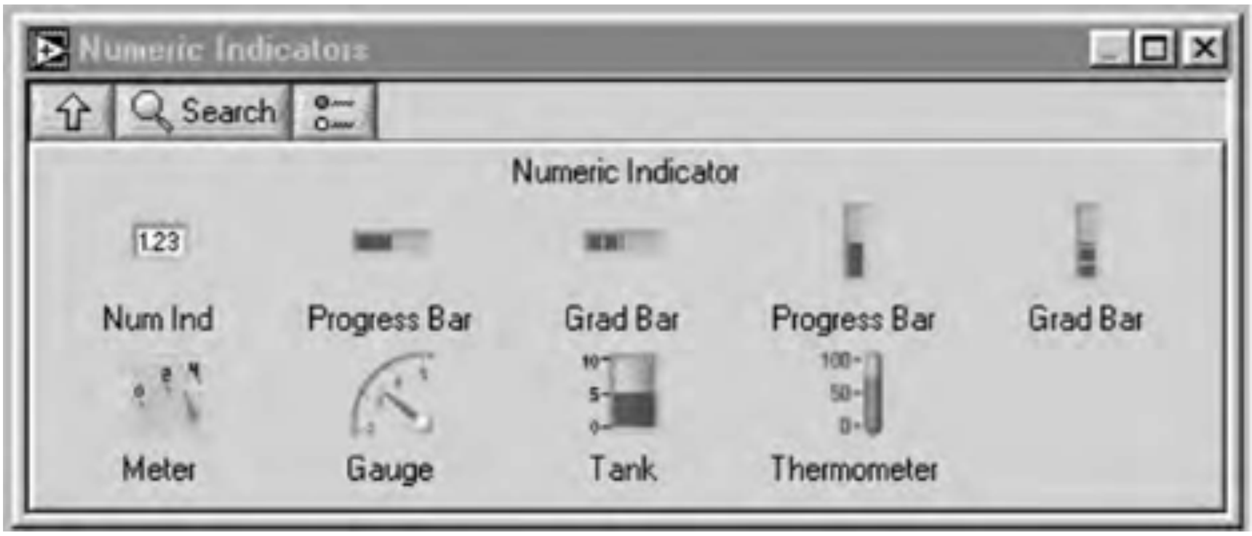


Fig 6: Numeric Control and Indicator

Block Diagram

The block diagram window holds the graphical source code of a LabVIEW'sblock diagram corresponds to the lines of text found in a more conventional language like C or BASIC – it is the actual executable code. The block diagram can be constructed with the basic blocks such as: terminals, nodes, and wires.

Terminals

Terminals are entry and exit ports that exchange information between the front panel and block diagram. Terminals are analogous to parameters and constants in text-based programming languages.

Types of terminals include control or indicator terminals and node terminals. Control and indicator terminals belong to front panel controls and indicators. When a control or indicator is placed on the front panel, LabVIEWautomatically creates a corresponding terminal on the block diagram. The terminals represent the data type of the control or indicator. The user cannot delete a block diagram terminal that belongs to a control or indicator. The terminal disappears when its corresponding control or indicator is deleted on

the front panel. The front panel controls or indicators can be configured to appear as icon

or data type terminals on the block diagram. By default, front panel objects appear as icon terminals.
Control terminals have thick borders, while indicator terminal borders are thin. It is very important to distinguish between the two since they are not functionally equivalent i.e., control=input, indicator=output, and so they are not interchangeable.

Nodes:

Nodes are analogous to statements, operators, functions, and subroutines in standardprogramming languages. The AND and XOR functions represent one type of node. A structure is another type of node. Structures can execute code repeatedly or conditionally, similar to loops and Case statements in traditional programming languages. LabVIEW also has special nodes, called Formula Nodes, which are useful for evaluating mathematical formulas or expressions.

Wires

Wires connecting nodes and terminals hold a LabVIEW VI together. Wiresare the data paths between source and destination terminals, they deliverdata from one source terminal to one or more destination terminals. Wiresare analogous to variables in text-based programming languages. If more thanone source or no source at all is connected to a wire, LabVIEW disagreesand the wire will appear broken. This principle of wires connecting sourceand destination terminals explains why controls and indicators are not interchangeable;controls are source terminals, whereas indicators are destinations,or sinks. Each wire has a different style or color, depending on the data typethat flows through the wire. The below Table shows a few wires and their correspondingtypes.By default, basic wire styles are used in block diagrams. To avoid confusionamong the data types, the colors and styles are simply matched.

Block Diagram Toolbar

When a VI is run, buttons appear on the block diagram toolbar that can beused to debug the VI.

Table : Data type and representation

	Scalar	1D array	2D array	Color
Floating point number				Orange
Integer number				Blue
Boolean				Green
String				Pink

Table 1 :Dialogue box menu

S No	Icon	Name	Meaning
1		New	The **New** button is used to create a new VI. The arrow on the **New** button is used to open a blank VI or to open the **New** dialog box.
2		Open	The **Open** button is used to open an existing VI. The arrow on the **Open** button is used to open recent files.
3		Configure	The **Configure** button is used to configure the data acquisition devices. The arrow on the **Configure** button is used to configure LabVIEW.
4		Help	The **Help** button is used to launch the *LabVIEW Help*. The arrow on the **Help** button is used for other Help options, including the NI Example Finder.

Startup Menu

The menus at the top of a VI window contain items common to other applications,such as Open, Save, Copy, and Paste, and other items specific toLabVIEW. Some menu items also list shortcut key combinations (Mac OS).The menus appear at the top of the screen (Windows and UNIX). Themenus display only the most recently used items by default. The arrows atthe bottom of a menu are used to display all items. All menu items can bedisplayed by default by selecting Tools->Options

Palletes

LabVIEW has graphical; floating palettes to help the user to create and runVIs.LabVIEW has three often-used floating palettes that can be placed in aconvenient spot on the screen: the Tools palette, the Controls palette, andthe Functions palette.Controls and Functions Palettes.The Controls palette will often be used, since the controls and indicatorsthat are required on the front panel are available. The user will probably usethe Functions palette even more often, since it contains the functions andstructures used to build a VI. The Controls and Functions palettes are uniquein several ways. Most importantly the Controls palette is only visible when thefront panel window is active, and the Functions palette is only visible whenthe block diagram window is active. Both palettes have subpalettes containingthe objects to be accessed. As the cursor is passed over each subpalette buttonin the Controls and Functions palettes, the subpalette's name appears at thetop of the window.If a button is clicked, the associated subpalette appears and replaces theprevious active palette. To select an object in the subpalette, the mouse button
is clicked over the object, and then clicked on the front panel or block diagramto place it wherever desired. Like palette button names, subpalette objectname appear when the cursor is run over them.

To return to the previous("owning") palette, the top-left arrow on each palette is selected. Clicking onthe spyglass icon the user can search for a specific item in a palette, and thenthe user can edit palettes by clicking the options buttons.There is another way to navigate palettes that some people find a littleeasier. Instead of having each subpalette replace the current palette, the user can pass through subpalettes in a hierarchical manner without them replacingtheir parent palettes.

Controls Palette

The Controls palette can be displayed by selecting Window->Show ControlsPalette or right-clicking the front panel workspace. The Controlspalette can also be tacked down by clicking the thumbtack on the top leftcorner of the palette. By default, the Controls palette starts in the Expressview. The Express palette view includes subpalettes on the top level of theControls and Functions palettes. The All Controls and All Functionssubpalettes contain the complete set of built-in controls, indicators, VIs, andfunctions. The Advanced palette view includes subpalettes on the top level ofthe Controls and Functions palettes that contain the complete set of builtincontrols, indicators, VIs, and functions. The Express subpalettes containExpress VIs and other objects required to build common measurement applications.Click the Options button on the Controls or Functions palette tochange to another palette view or format.

Functions Palette

The Functions palette is available only on the block diagram. The Functionspalette contains the VIs and functions used to build the block diagram. TheFunctions Palette can be displayed by selecting the Windows->Show orright-clicking the block diagram workspace to display the Functions palette.

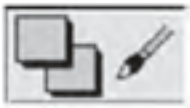
Fig : 7 Function Palette

Tools Palette

A tool is a special operating mode of the mouse cursor. Tools are used toperform specific diting and operation functions, similar to that used in a standard paint program.

Tool Palette

Table 2 : Tool Palatte Menu

S. No	Icon	Name	Meaning
1		Operating Tool	The Operating Tool is used to change values of front panel controls and indicators. Used to operate knobs, switches, and other objects with the Operating tool – hence the name. It is the only front panel tool available when the VI is running or in run mode.
2		Positioning tool	The Positioning tool selects, moves, and resizes objects.
3		Labeling tool	The Labeling tool creates and edits text labels.
4		Wiring tool	The Wiring tool wires objects together on the block diagram. It is also used to assign controls and indicators on the front panel to terminals on the VI's connector.
5		Color tool	The Color tool brightens objects and background by allowing the user to choose from a multitude of hues. Both foreground and background colors can be selected by clicking on the appropriate color area in the Tools palette. If an object is popped up with the Color tool, a hue from the color palette appears and the required color can be chosen.

S. No	Icon	Name	Meaning
6		Pop-up tool	The Pop-up tool opens an object's Pop-up menu when the user clicks on the object with it. This tool can be used to access Pop-up menus instead of the standard method for popping up (right-clicking under Windows and Unix and <command>-clicking on MacOS).
7		Scroll tool	The Scroll tool lets the user to scroll in the active window.
8		Breakpoint tool	The Breakpoint tool sets breakpoints on VI diagrams to help in debugging the code. It causes execution to suspend so that the user can see what is going on and change input values if required.
9		Probe tool	The Probe tool creates probes on wires so that the user can view the data traveling through them while the VI is running.
10		Color Copy tool	Use the Color Copy tool to pick up a color from an existing object, and then use the Color tool to paste that color onto other objects. This technique is very useful if the user needs to duplicate an exact shade but can't remember which one it was. The user can also access the Color Copy tool when the Color tool is active by holding down the <control> key on Windows, <option> on MacOS, <meta> on Sun, and <alt> on Linux and HP-UX.
11		Automatic Tool Selection	The **Automatic Tool Selection** button on the **Tools** palette can be selected to disable automatic tool selection. Press the <Shift-Tab> keys or click the **Automatic Tool Selection** button to enable automatic tool selection again. Press the <Tab> or <Shift-Tab> keys or clicks the **Automatic Tool Selection** button on the **Tools** palette to enable automatic tool selection again. If automatic tool selection is disabled, the user can press the spacebar to switch to the next most useful tool.

16BEBME702A VIRTUAL INSTRUMENTATION DESIGN FOR MEDICAL SYSTEMS

UNIT II PROGRAMMING

TECHNIQUES

VIS and sub-VIS, loops & charts, arrays, clusters, graphs, case & sequence structures, formula modes, local and global variable, string & file input.

VIS and sub-VIS:

Loops & Charts:

LabVIEW offers two loop structures namely, the For Loop and While Loop to control repetitive operation in a VI. A For Loop executes a specific number of times; a While Loop executes until a specified condition is no longer true.

The For Loop

A For Loop executes the code inside its borders, called its sub diagram, for total of count times, where the count equals the value contained in the countterminal. The count can be set by wiring a value from outside the loop of the count terminal. If '0' is wired to the count terminal, the loop does not execute.

The iteration terminal contains the current number of completed loop iterations; 0 during the first iteration, 1 during the second, and so on, up to N-1 (where N is the number of times the loop executes).

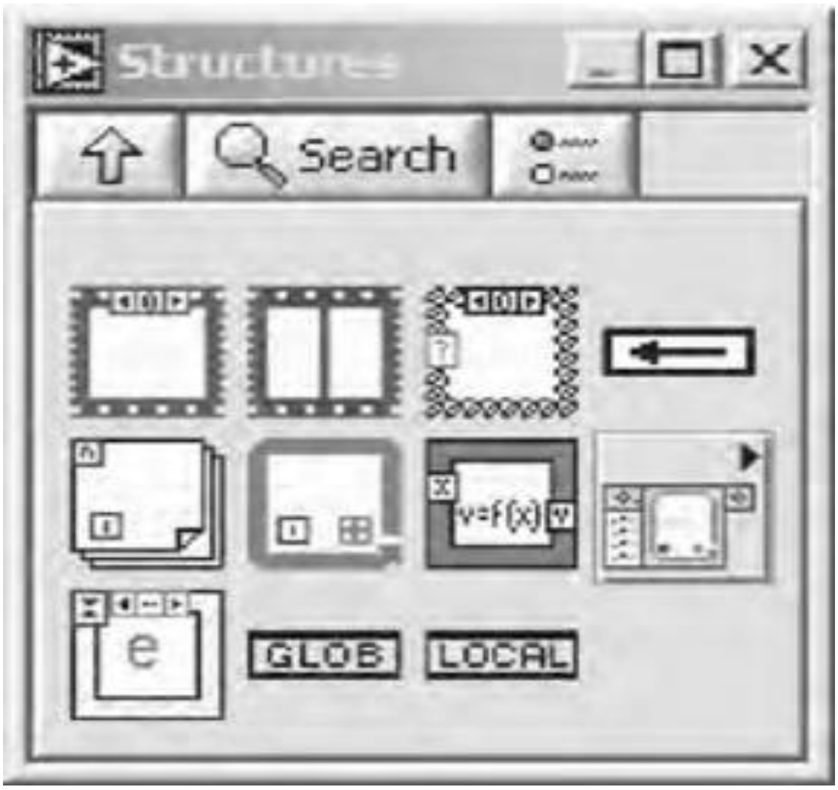


Fig 1: Structure Palette

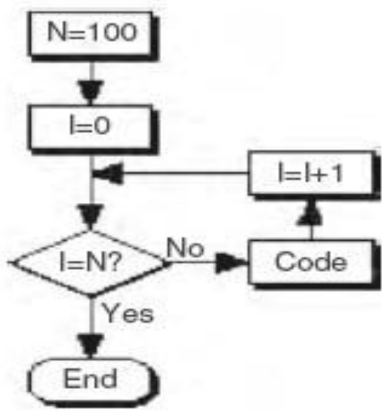


Fig 2: Flow chart of For loop

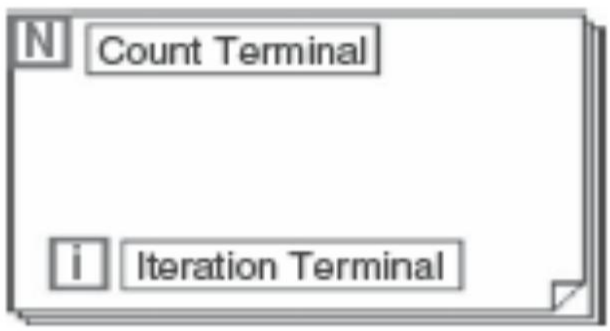


Fig 3: For Loop

- The For Loop is located on the **Functions**->**All Functions**->**Structures**
- The value in the count terminal (an input terminal), indicates how many times to repeat the sub diagram.
- The iteration terminal (an output terminal), contains the number of iterations completed. The iteration count always starts at zero.During the first iteration, the iteration terminal returns 0.

The While Loop

The While Loop executes the subdiagram inside its borders until the Boolean value wired to its conditional terminal is FALSE. LabVIEW checks the conditional terminal value at the end of iteration. If the value is TRUE, the iteration repeats. The default value of the conditional terminal is FALSE, so if left unwired, the loop iterates only once.

The While Loop is equivalent to the following pseudocode:

Do
Execute sub diagram

While condition is TRUE

We can also change the state that the conditional terminal of the While Loop checks, so that instead looping while true, we can have it loop unless it’strue. To do this, we pop-up on the conditional terminal, and select **“Stop ifTrue.”**

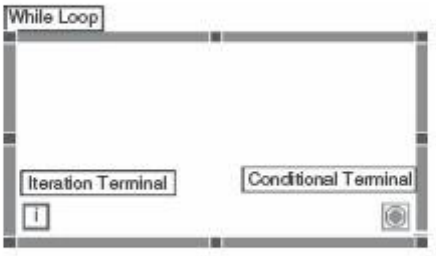


Fig 4: While loop terminals

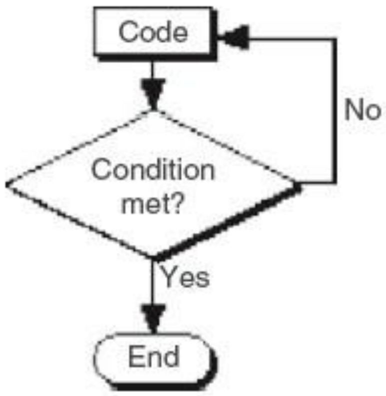


Fig 5: Flowchart of while loop

Do
Execute subdiagram
While condition is NOT TRUE

- The While Loop is located on the **Functions**>>**Execution Control** palette
- The section of code to be added inside the While loop is dragged or while loop encloses the area of code to be executed conditionally.

Condition Terminal
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**. When a conditional terminal is **Stop If True**, the While Loop executes its subdiagram until the conditional terminal receives a True value.

Iteration Terminal
The iteration terminal, an output terminal, contains the number of completed iterations. The iteration count always starts at zero. During the first iteration,

the iteration terminal returns 0.

Example 1

Problem statement: To find the sum of first 10 natural numbers using For Loop.

Block diagram construction:

– The For Loop is added from the structures sub palette on the functions Palette located on the block diagram.

– The count terminal is wired with the value 10 for 10 natural numbers.

– The iteration terminal is wired the greater than or equal to node from the Comparison subpalette from the Functions palette.

– The Boolean Output of the greater than or equal to is wired to the conditional terminal as in Fig.

Front panel construction:

– The control and indicator are added from the controls palette of the front panel. Using the labeling tool the Owned label is changed as input and output, respectively.

– The front panel with the result is shown in Fig. .

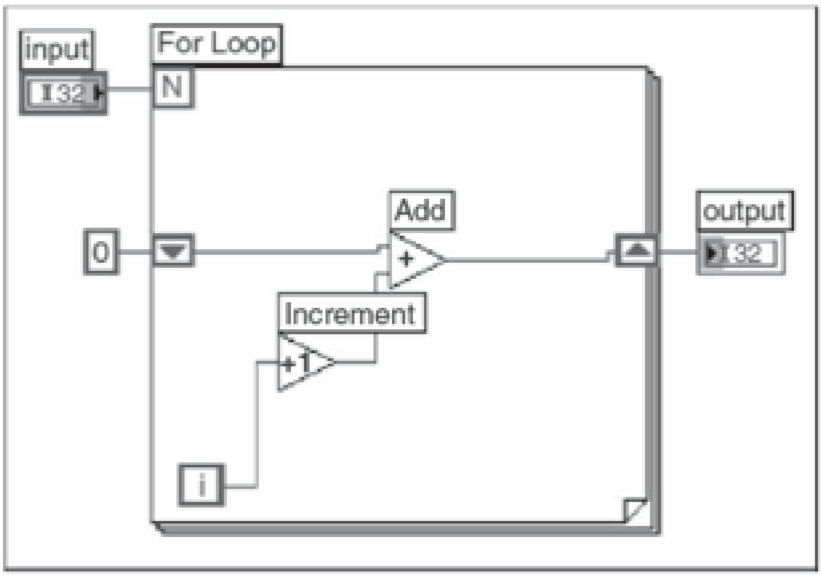


Fig 7: Block diagram of for loop example

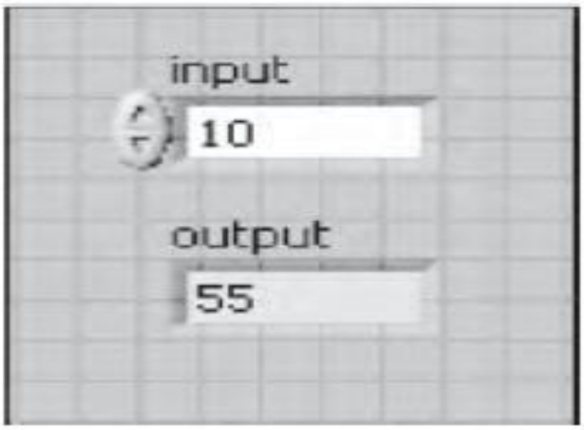


Fig 8: Front panel of For loop example

Example 2

Problem statement: To find the sum of first 10 natural numbers using while loop

Block diagram construction:

– The While Loop is added from the structures subpalette on the functions palette located on the block diagram.
– The count terminal is wired with the value 10 for 10 natural numbers.
– The iteration terminal is wired the greater than or equal to node from the comparison subpalette from the Functions palette.
– The Boolean Output of the greater than or equal to is wired to the conditional terminal of the While Loop as in Fig.

Front panel construction:

– The control and indicator are added from the controls palette of the front panel as in Fig. Using the labeling tool the Owned label is changed as input and output, respectively.

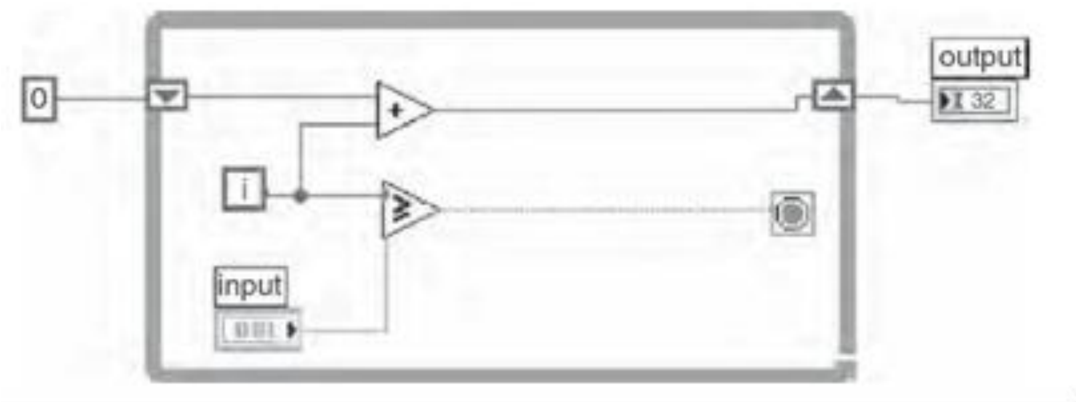


Fig 9: Block diagram of while loop for the above example

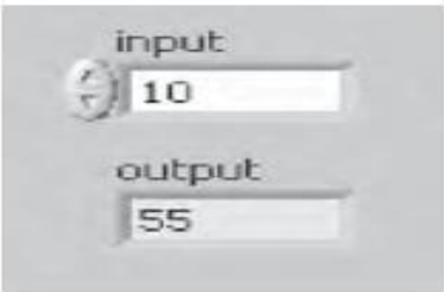


Fig 10: Front panel

Arrays:

A LabVIEW array is a collection of data elements that are all the same type similar to traditional programming languages. An array can have one or more dimensions, and has elements 2^{31}per dimension (memory permitting, of course). An array element can have any type except another array, a chart, or a graph.

Single and Multidimensional Arrays:
Single Dimensional Array:
Array elements are accessed by their indices; each element's index is in range 0 to $N - 1$, where N is the total number of elements in the array.
The *first* element has index 0, the *second* element has index 1, and so on. Generally, waveforms are often stored in arrays, with each point in the waveform comprising an element of the array. Arrays are also useful for storing data generated in loops, where each loop iteration generates one element of the array.

Two steps are involved to make the controls and indicators for compound data types such as arrays and clusters.

Step 1. The array control or indicator known as the Array shell is created from the **Array** subpalette of the **Controls** palette available in the front panel

Step 2. The array shell is combined with a data object, which can be numeric, Boolean, or string (or cluster).
Using the above two steps the created structure resembles

Index	0	1	2	3	4	5	6	7	8	9
10-element array	12	32	82	8.0	4.8	5.1	6.0	1.0	2.5	10

Fig11: Single Dimensional Array

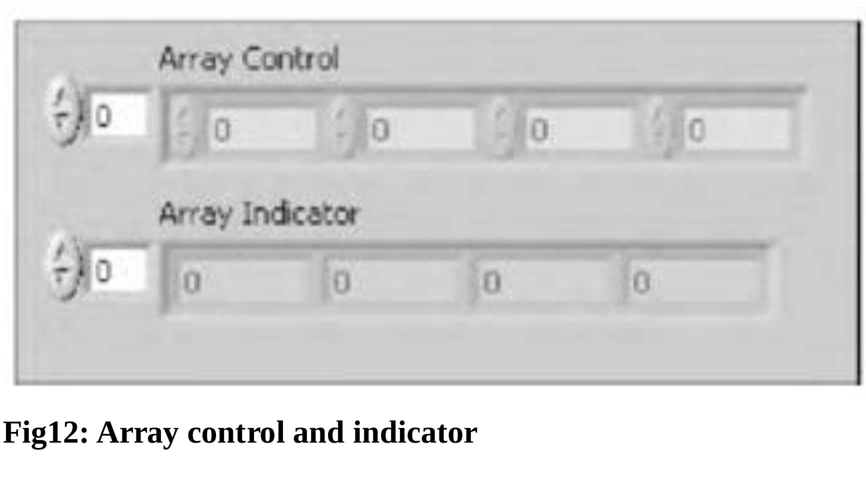


Fig12: Array control and indicator

The user can also create array constants on the block diagram just like creating numeric, Boolean, or string constants. **Array Constant** can be chosen from the **Array** subpalette of the **Functions** palette. Then the data is placed in an appropriate data type (usually another constant) similar to that created on the front panel. This feature is useful when the user needs to initialize shift registers or provide a data type to a file or network functions. To clear an array control, indicator, or constant of data, the user can pop-on the index display.

Two-Dimensional Arrays
A two-dimensional, or 2D, array stores elements in a gridline fashion. It requires two indices to locate an element: a column index and a row index, both of which are zero-based in LabVIEW

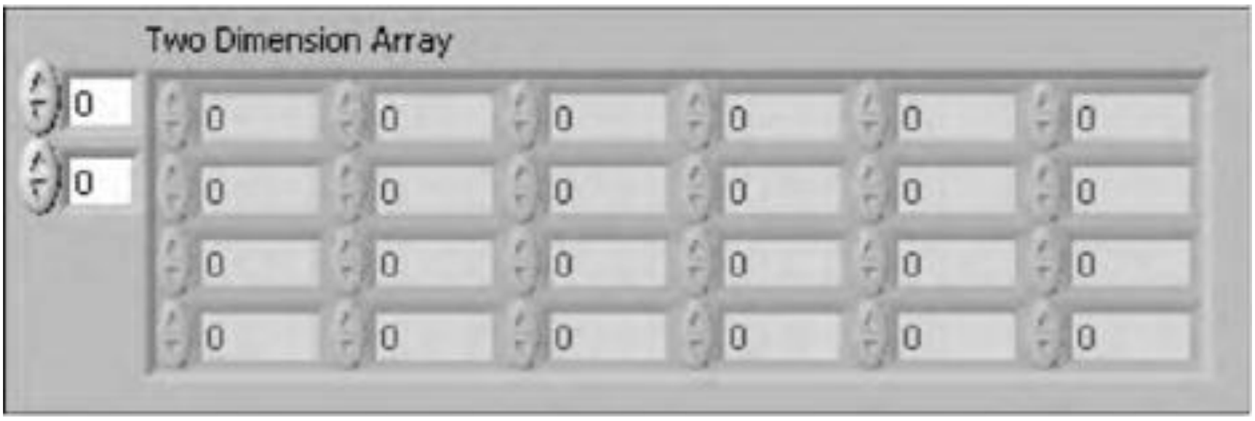


Fig13: 6*4 2D array

The user can add dimensions to an array control or indicator by popping up on its index display (not on the element display) and choosing **Add Dimension** from the Pop-up menu. In a 2D array of digital controls the user will have two indices to specify each element. The grid cursor of the Positioning tool can be used to expand the element display in two dimensions so that more elements can be seen. Unwanted dimensions can be removed by selecting **Remove Dimension** from the index display's Pop-up menu.

Creating Two-Dimensional Array
Two For Loops, one inside the other, can be used to create a 2D array on the front panel. The inner For Loop creates a row, and the outer For Loop "stacks" these rows to fill in the columns of the matrix. In two For Loops creating a 2D array of random numbers using auto indexing

Autoindexing
The For Loop and the While Loop can index and accumulate arrays at their boundaries automatically, adding one new element for each loop iteration. This capability is called ***autoindexing***. One important thing to remember is that autoindexing enabled by default on For Loops but disabled by default on While Loops. The For Loop autoindexes, an array at its boundary. Each iteration creates the next array element. After the loop completes, the arraypasses out of the loop to the indicator; none of the array data are available until the Loop finishes. Notice that the wire becomes thicker as it changes to an array wire type at the Loop border.

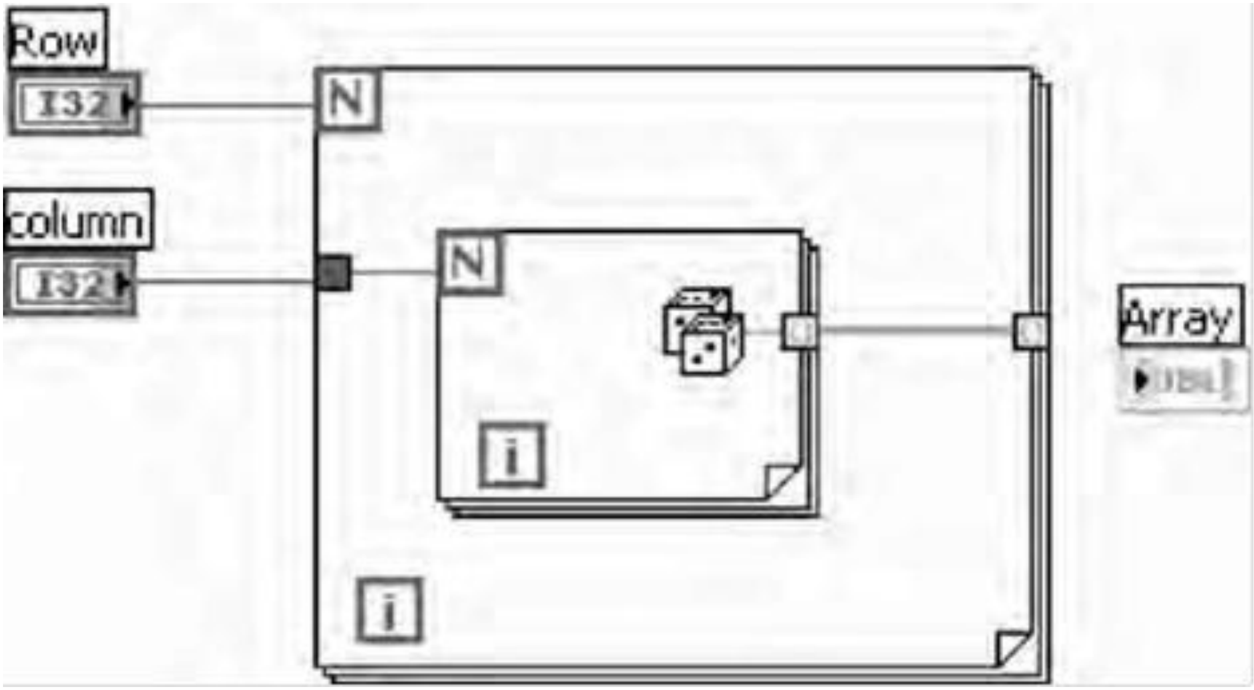


Fig14: For loop for creating 2 D array

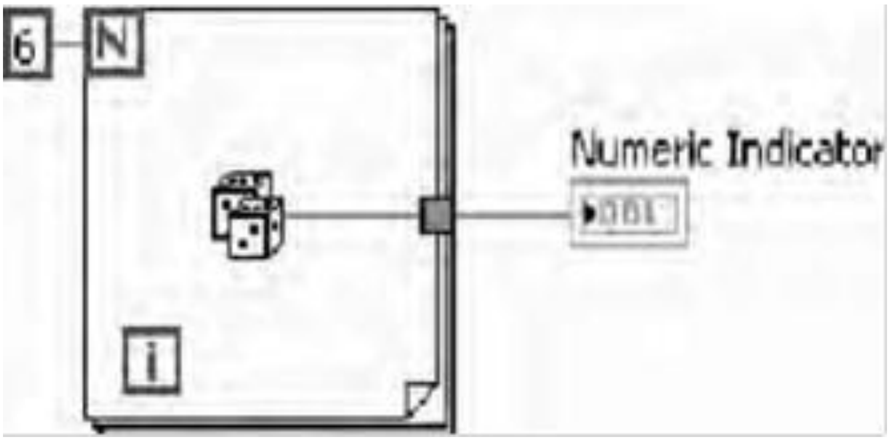


Fig15: Auto indexing disabled

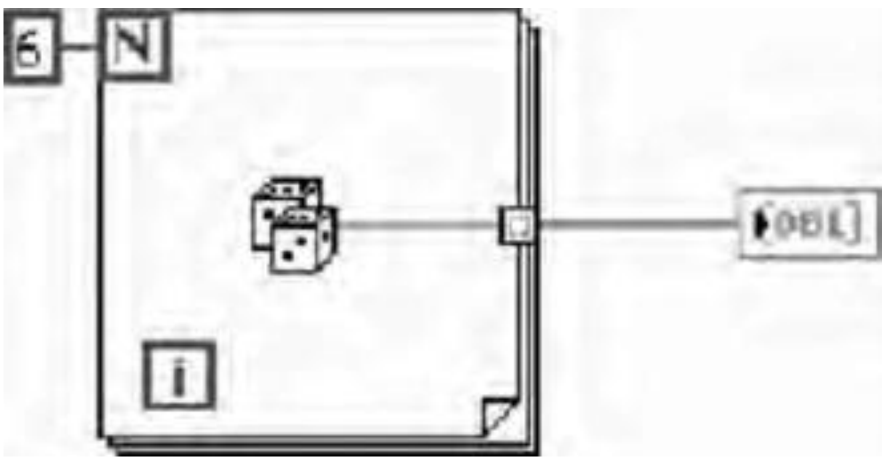


Fig16: Auto indexing enabled

To wire a scalar value out of a For Loop without creating an array, autoindexing must be disabled. This can be done by popping up on the tunnel (the square with the [] symbol) and choosing **Disable Indexing** from the tunnel's Pop-up menu. Since autoindexing is disabled by default, to wire array data out of a While Loop, the user must pop-up on the tunnel and select**Enable Indexing**. When autoindexing is disabled, only the last value returned from the **Random Number (0–1)** function passes out of the loop. Notice that the wire remains the same size after it leaves the loop. Pay attention to this wire size, because autoindexing is a common source of problems among beginners. Autoindexing also applies when the user is wiring arrays into loops. If indexing is enabled, the loop will index off one element from the array each time it iterates (note how the wire becomes thinner as it enters the loop). If indexing is disabled, the entire array passes in to the loop at once. Because For Loops are often used to process arrays, LabVIEW enables autoindexing by default when an array is wired into or out of them.

Clusters :

clusteris a data structure that groups data. However, unlike an array, a cluster can group data of different types (i.e., numeric, Boolean, etc.); it is analogous to a structin C or the data members of a class in C++ or Java.

Cluster elements can be accessed by *unbundling* them all at once or by indexing one at a time, depending on the function chosen; each method has its place. Unbundling a cluster is similar as unwrapping a telephone cable and having access to the different-colored wires. Unlike arrays, which can change size dramatically, clusters have a fixed size, or a fixed number of wires in them. The Unbundled By Name function is used to access specific cluster elements. Cluster terminals can be connected with a wire only if they have exactly the same type; in other words, both clusters must have the same number of elements, and corresponding elements must match in both data type and order. The principle of polymorphism applies to clusters as well as arrays, as long as the data types match.

Clusters are often used in error handling. The error clusters, **Error In.ctl** and**Error Out.ctl**, are used by LabVIEW to pass a record of errors between multiple VIs in a block diagram

Creating Cluster Controls and Indicators

A cluster is created by placing a **Cluster** shell (**Array & Cluster** subpalette of the **Controls** palette) on the front panel. Like arrays, objects can be added directly inside when they are pulled out from the **Controls** palette, or the user can drag an existing object into a cluster. **Objects inside a cluster mustbe all controls or all indicators**.

A cluster cannot have a combination of both controls and indicators; this is based on the status of the first object one place inside it. The cluster can be resized with the Positioning tool if necessary. The cluster can conform exactly to the size of the objects inside it, by popping up on the border(not inside the cluster) and choosing an option in the **Auto sizing** menu.

Creating Cluster Constants

If a cluster control or indicator is available on the front panel and if the user wants to create a cluster constant containing the same elements on the block diagram, then the user can either drag that cluster from the front panel to the block diagram or right-click the cluster on the front panel and select **Create**>>**Constant** from the shortcut menu.

Fig17: Cluster palette

Bundling Data

The **Bundle** function (**Cluster** palette) assembles individual components into a new cluster or allows us to replace elements in an existing cluster. The function appears as the icon at the left when one places it in the diagram window. Dragging a corner of the function with positioning tool can increase the number of inputs. When wired on each input terminal, a symbol representing the data type of the wired element appears on the empty terminal. The order of the resultant cluster will be the order of inputs to the **Bundle**. To create a new cluster the user need not wire an input to the center **cluster** input of the **Bundle** function. This needs to be wired only if elements are replaced in the cluster.

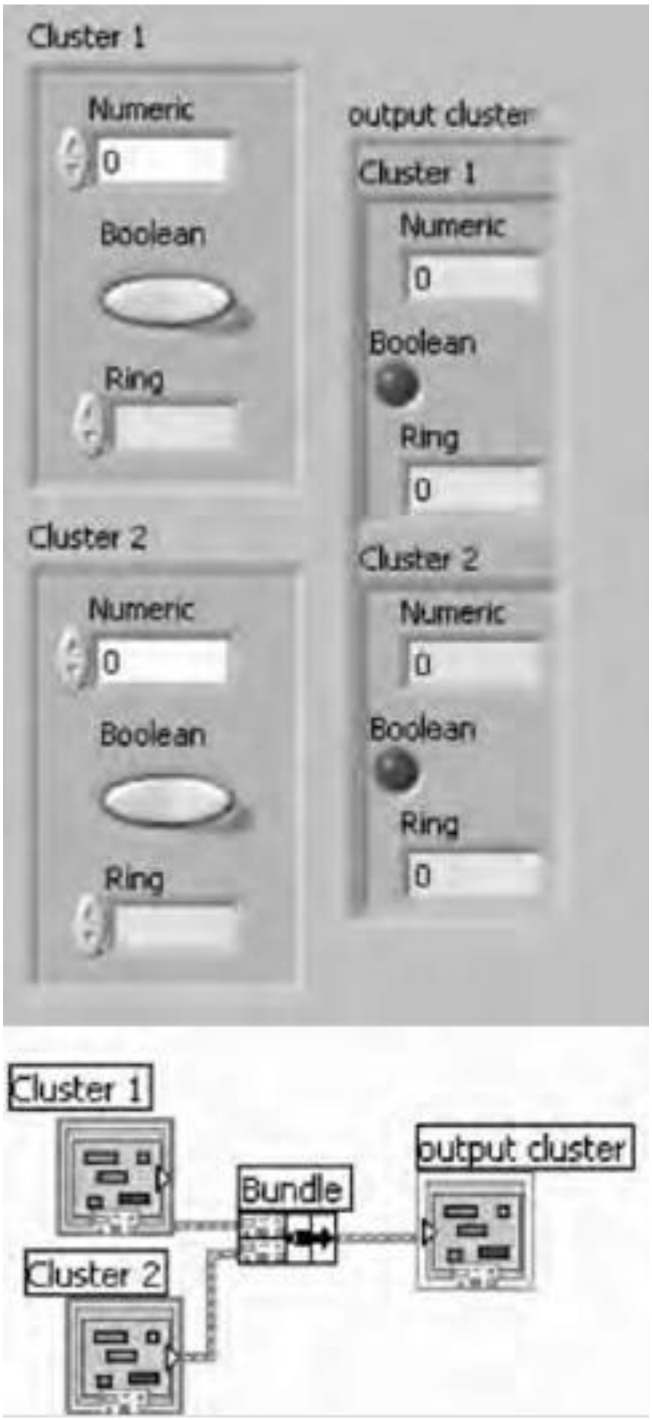


Fig18: Bundle Function

Unbundling Clusters

The **unbundled** function (**Cluster** palette) in Fig. 19 splits a cluster into each of its individual components. The output components are arranged fromtop to bottom in the same order they have in the cluster. If they have the same data type, the elements' order in the cluster is the only way to distinguish among them. Dragging a corner in the function with the Positioning tool can increase the number of outputs. The **Unbundled** function must be sized to contain the same number of outputs as there are elements in the input cluster, or it will produce bad wires. When an input cluster is wired to the correctly sized **Unbundled**, the previously blank output terminals will assume the symbols of the data types in the cluster example has been shown in Fig. LabVIEW does have a way to bundle and unbundled clusters using element names.

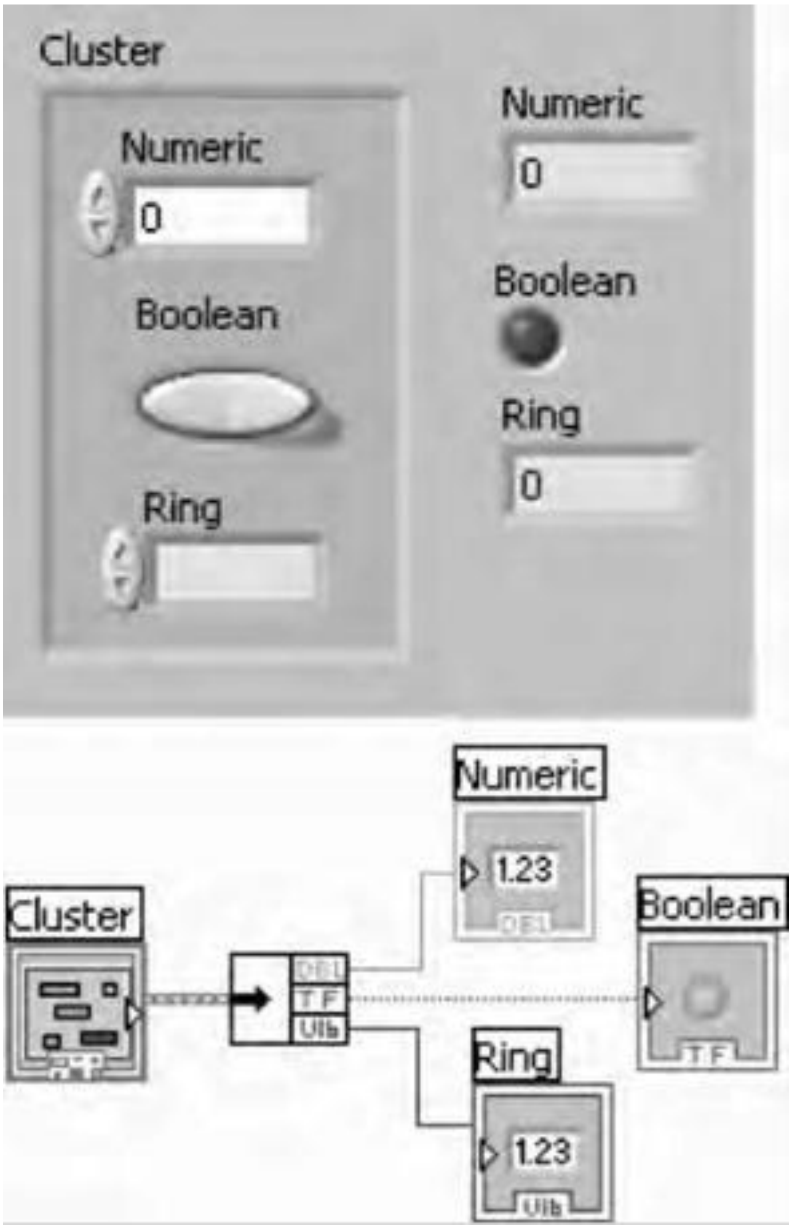


Fig19: Unbundle function

Bundling and Unbundling by Name
Sometimes there is no need to assemble or disassemble an entire cluster – the user just needs to operate on an element or two. This is accomplished using **Bundle By Name** and **Unbundle By Name** functions. **Unbundle By Name**, also located in the **Cluster** palette, returns elements whose name(s) are specified. There is no need to consider cluster order to correct **Unbundle** function size. The unbundle function is illustrated in Fig.

Bundle by Name, found in the **Cluster** palette, references elements by name instead of by position (as **Bundle** does). Unlike **Bundle**, we can access only the elements that are required. However, **Bundle by Name** cannot create new clusters; it can only replace an element in an existing cluster. Unlike **Bundle**, **Bundle by Name's** middle input terminal should be wired to allow the function know which element in the cluster has to be to replaced. This function is illustrated in Fig. All cluster elements should have owned labels when the **By Name** functions are used. As soon as the cluster input of **Bundle By Name** or **Unbundled By Name** is wired, the name of the first element in the cluster appears in the name input or output.

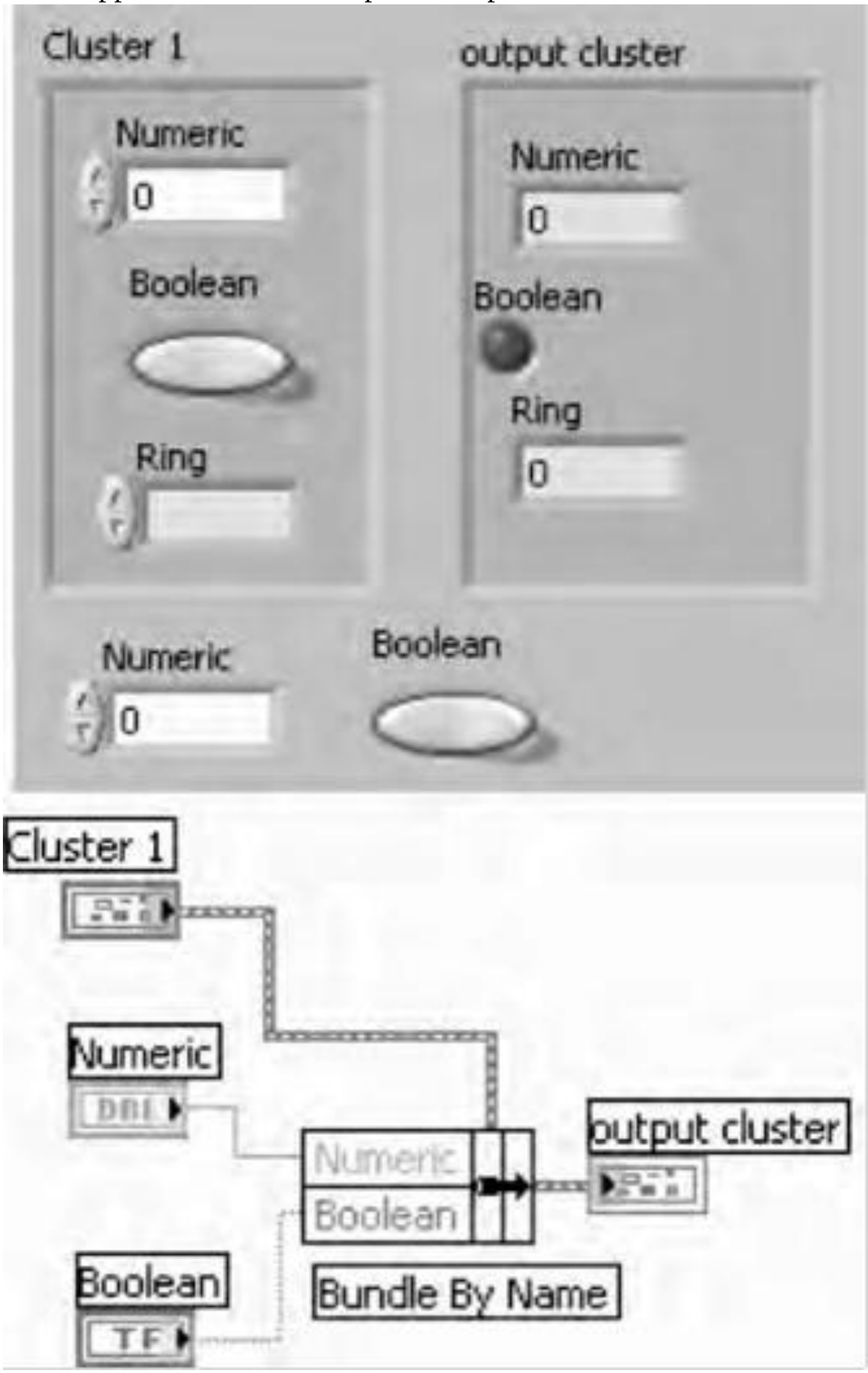


Fig20: Bundle by name

!

!

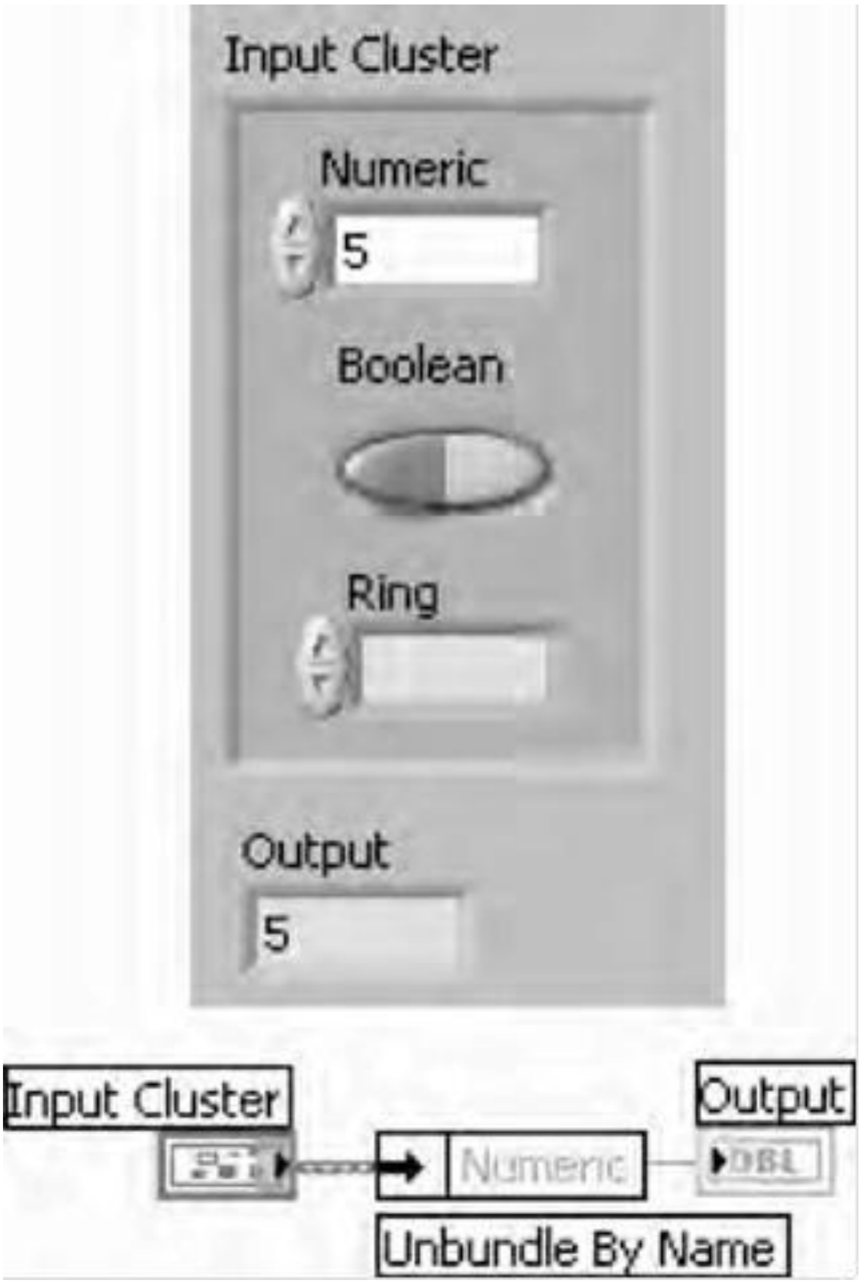


Fig21: Unbundle by name

Graphs:

!

!

The graphs located on the **Controls_Graph Indicators** palette include the waveform graph and XY graph. The waveform graph plots onlysingle-valued functions, as in *y* = f(*x*), with points evenly distributed along the x-axis, such as acquired time-varying waveforms.

!

!

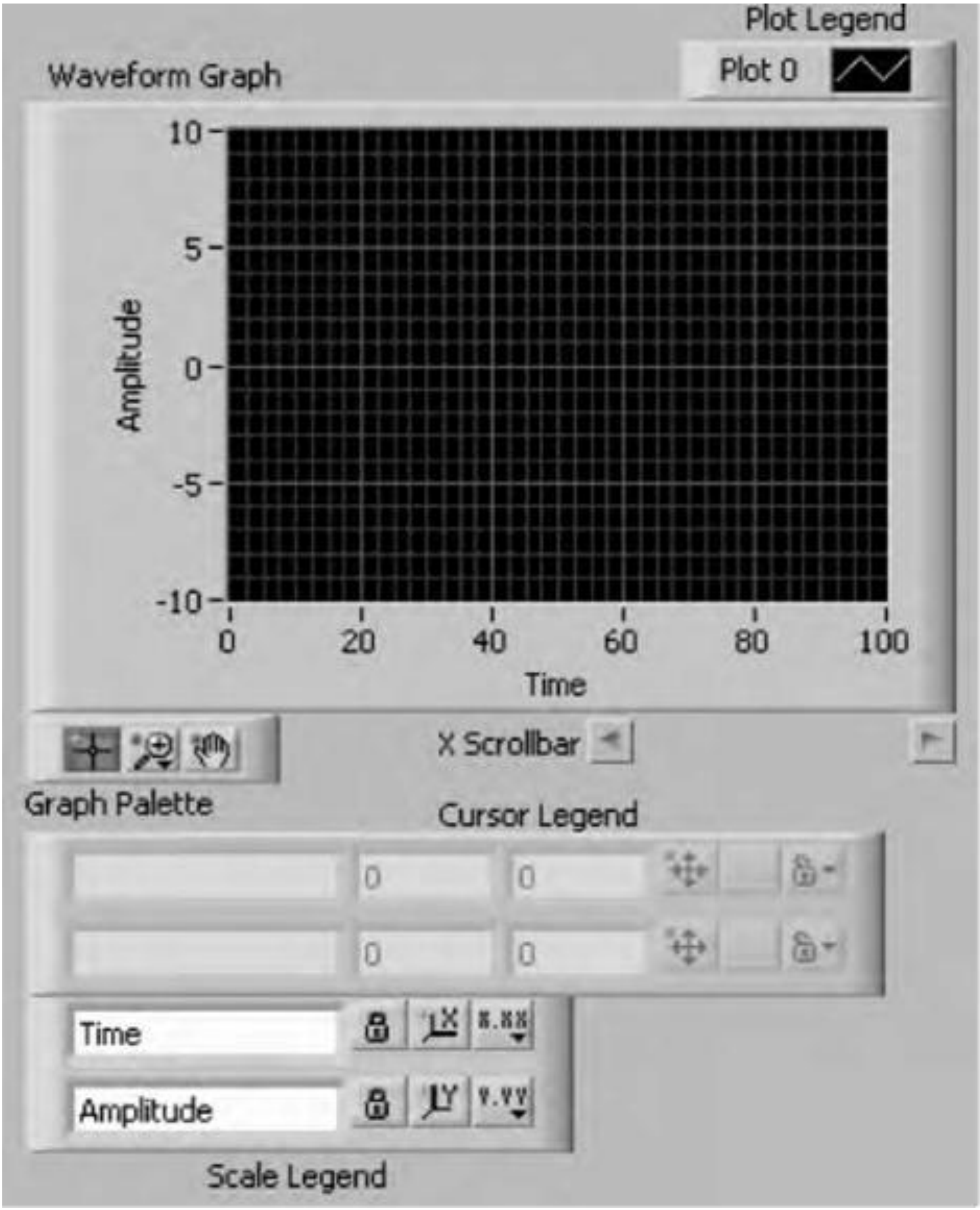


Fig 22: waveform graph

Single-Plot Waveform Graphs

The waveform graph accepts a single array of values and interprets the data as points on the graph and increments the *x* index by one starting at $x = 0$. The graph also accepts a cluster of an initial *x* value, and an array of *y* data.

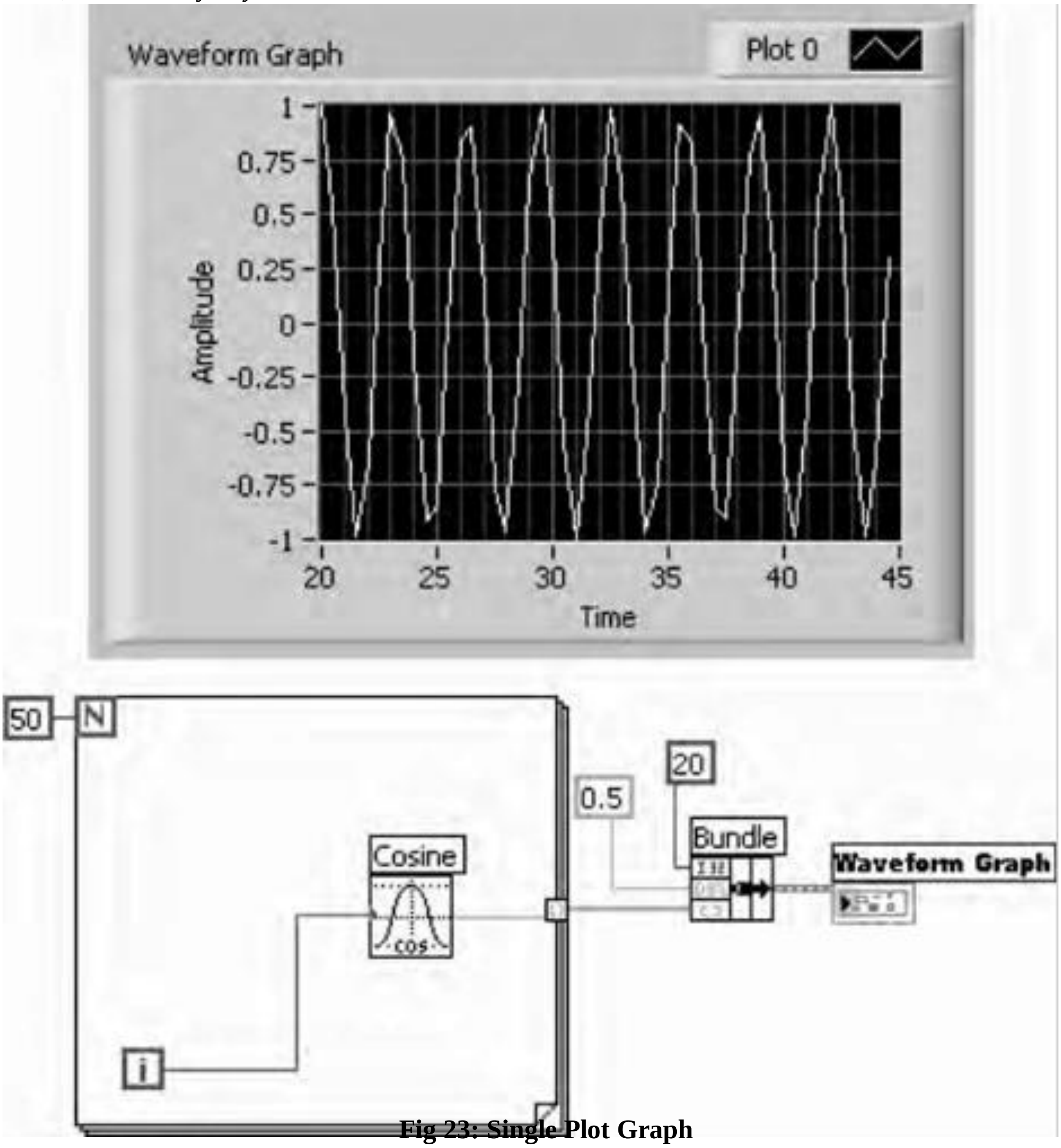


Fig 23: Single Plot Graph

Multiple-Plot Waveform Graphs

A multi plot waveform graph accepts a 2D array of values, where each row of the array is a single plot. The graph interprets the data as points on the graph and increments the *x* index by one, starting at $x = 0$. Wire a 2D array data type to the graph, right-click the graph, and select **Transpose Array** from the shortcut menu to handle each column of the array as a plot.

!

!

XY Graphs

XY graphs display any set of points, evenly sampled or not. Resize the plot legend to display multiple plots. Use multiple plots to save space on the front panel and to make comparisons between plots. XY and waveform graphs automatically adapt to multiple plots.

Single Plot XY Graphs

The single-plot XY graph accepts a cluster that contains an *x* array and a *y* array. The XY graph also accepts an array of points, where a point is a clusterthat contains an *x* value and a *y* value.

Multiplot XY Graphs

The multiplot XY graph accepts an array of plots, where a plot is a cluster that contains an *x* array and a *y* array. The multi plot XY graph also accepts an array of clusters of plots, where a plot is an array of points.

Waveform Charts

A plot is simply a graphical display of *X* versus *Y* values. Often, *Y* values in a plot represent the data value, while *X* values represent time. The waveform chart, located in the **Graph** subpalette of the **Controls** palette, is a special numeric indicator that can display one or more plots of data.

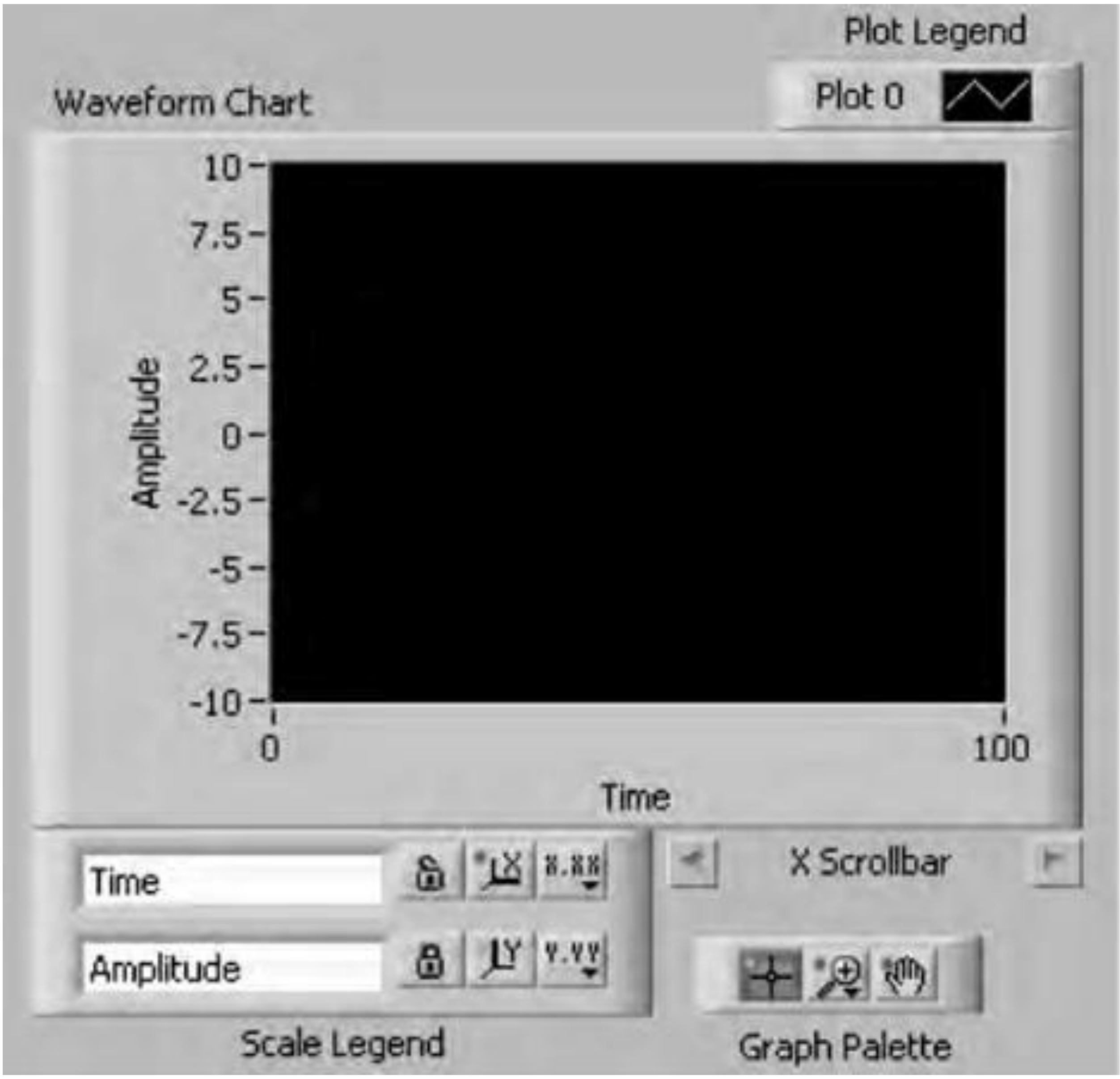


Fig24: Waveform chart and its component

Chart Update Modes

The waveform chart has three update modes – strip chart mode, scope chart mode, and sweep chart mode. The update mode can be changed by poppingup on the waveform chart and choosing one of the options from the **Advanced**>>**update Mode**>>menu. Modes can be changed while the VI isrunning by selecting **Update Mode** from the chart's runtime Pop-up menu.The default mode is **Strip Chart**. The strip chart has a scrolling display similar to a paper strip chart . The scope chart and the sweep chart have retracing displays similar to that of an oscilloscope.

Strip Chart

A strip chart shows running data continuously scrolling from left to right across the chart.

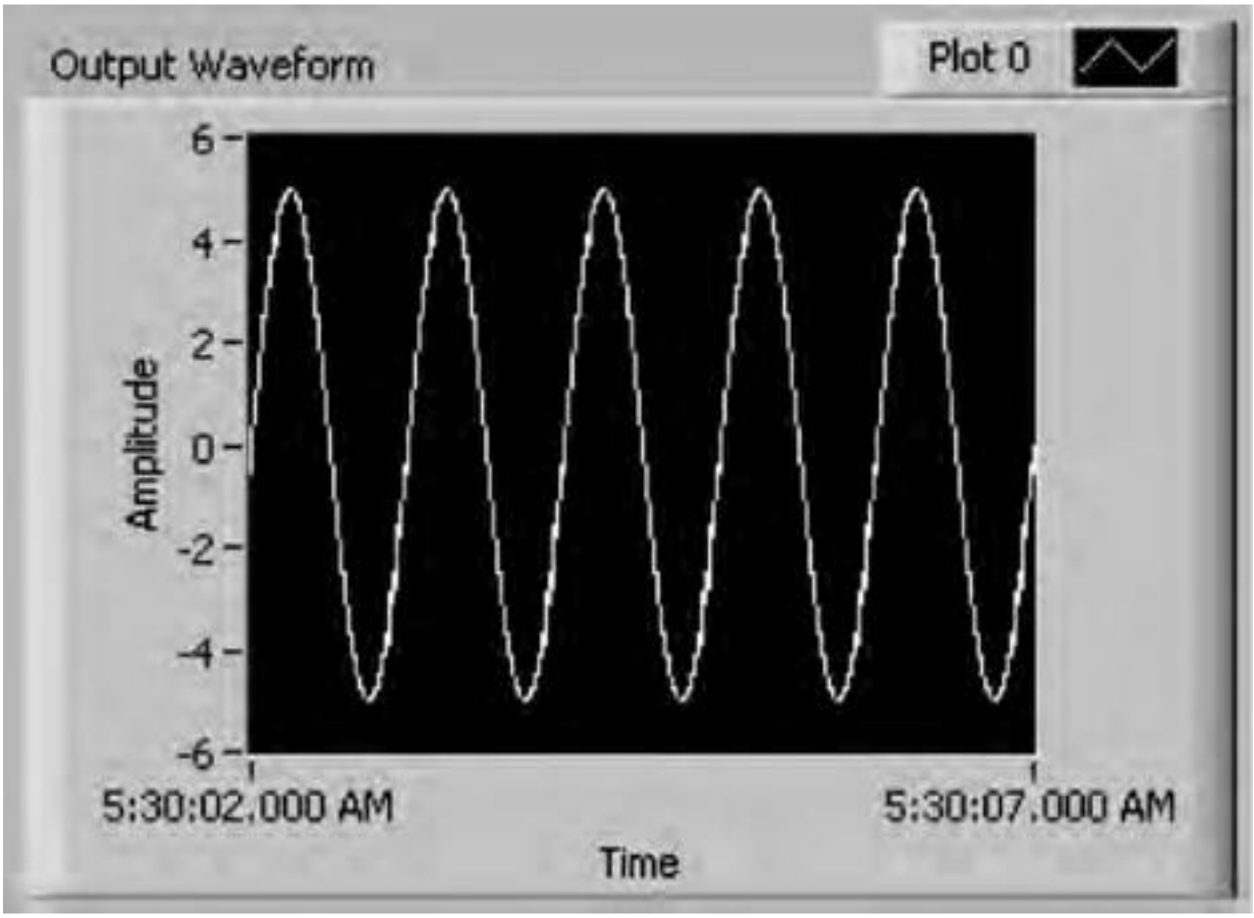


Fig:25 Strip chart

Scope chart:

A scope chart shows one item of data, such as a pulse or wave, scrolling partway across the chart

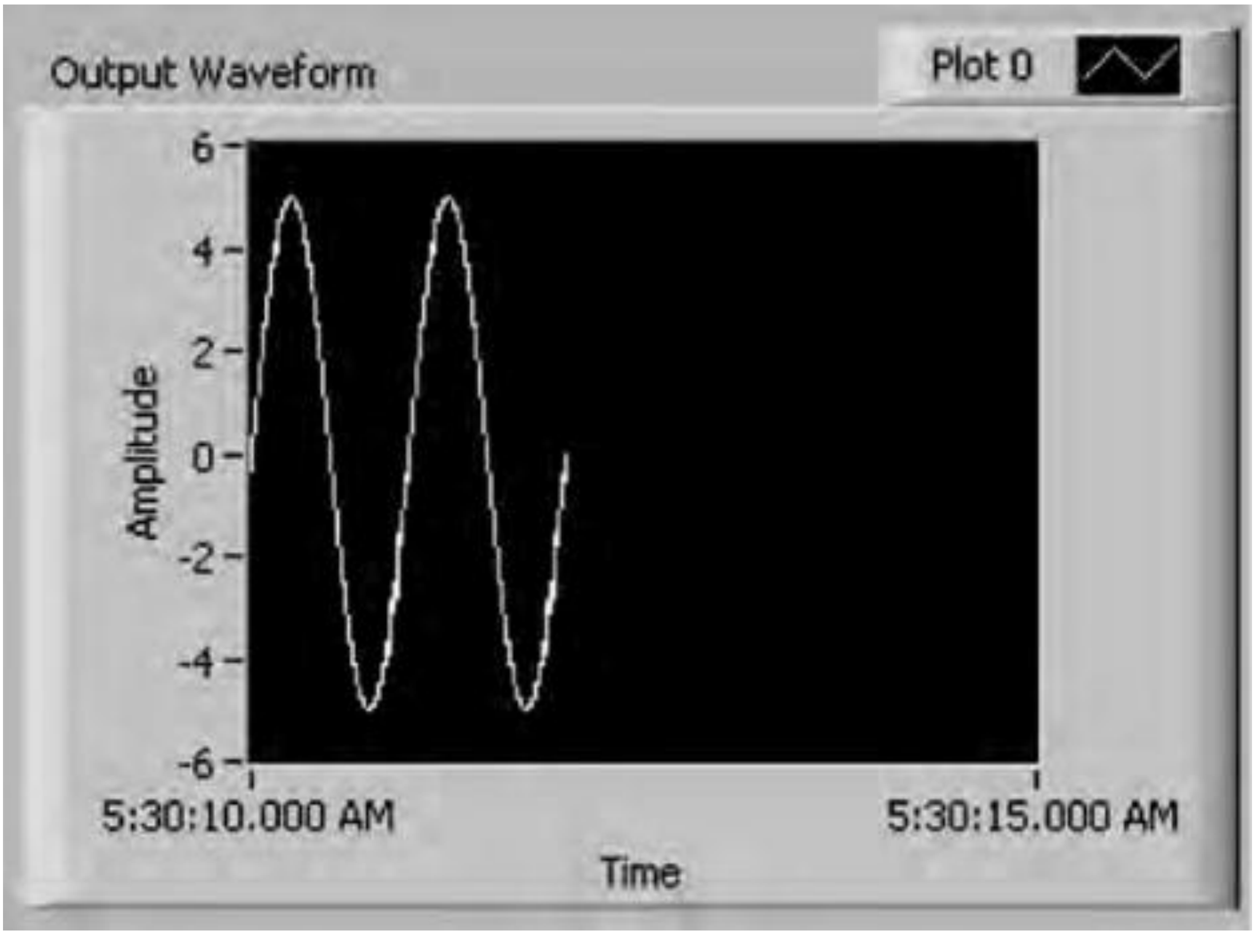


Fig: 26 Scope chart

Sweep Chart:

A sweep chart is similar to an EKG display. A sweep chart works similarly to a scope except it shows the older data on the right and the newer data on the left separated by a vertical line. The scope chart and sweep chart have retracing displays similar to an oscilloscope. Because there is less overhead in retracing a plot, the scope chart and the sweep chart display plots significantly faster than the strip chart.

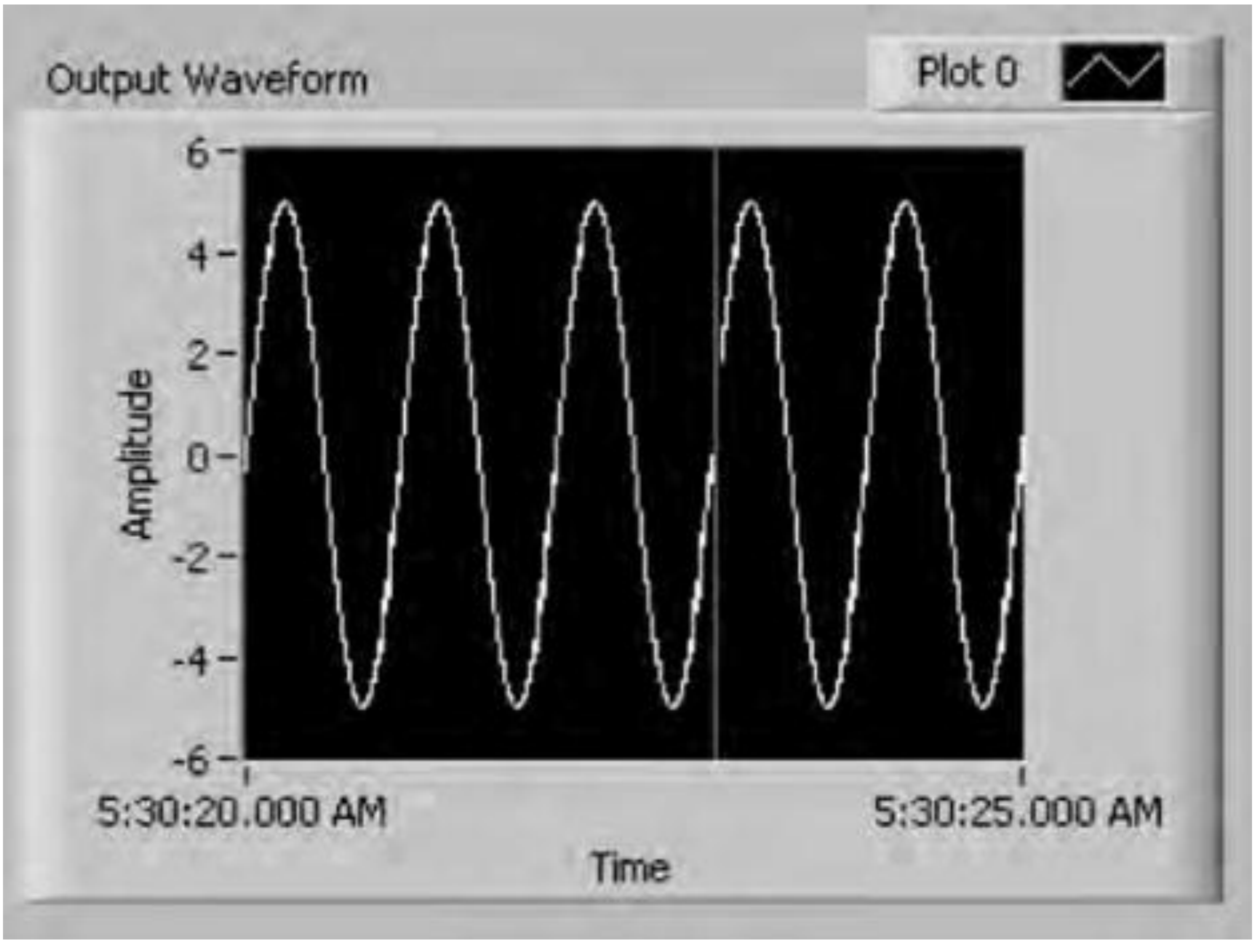


Fig:27 Sweep Chart

Wiring Charts

A scalar output can be wired directly to a waveform chart.

Strings

A string is a sequential collection of displayable or nondisplayable ASCIIcharacters. Strings provide a platform-independent format for informationand data. Often, strings may be used for displaying simple text message.

Some of the more common applications of strings include the following:

– Creating simple text messages.
– Passing numeric data as character strings to instruments and then converting the strings to numeric values.
– Storing numeric data to disk. To store numeric values in an ASCII file, the numeric values must be first converted to strings before writing the numeric values to a disk file.
– Instructing or prompting the user with dialog boxes. On the front panel, strings appear as tables, text entry boxes, and labels.

Creating String Controls and Indicators

The string control and indicator located on the Controls_Text Controlsand Controls_Text Indicators palettes are used to simulate text entryboxes and labels. Using the Operating toolor labeling tool text data can betyped or edited in a string control. The Positioning tool is used to resize a frontpanel string object. The space occupied by a string object can be minimized byright-clicking the object and selecting the Visible Items_Scrollbar optionfrom the shortcut menu. The display types can be selected by right-clickinga string control or indicator on the front panel

String Functions

String Length returns the number of characters in a given string asshown in Fig 28.

Concatenate Strings concatenates all input strings into a single outputstring as shown in Fig 29.The function appears as the icon shown in block diagram in Fig 29.The function can be resized with the Positioning tool to increase the numberof inputs. In addition to simple strings, the user can also wire a 1D array ofstrings as input; the output will be a single string containing a concatenationof strings in the array.

String Subset accesses a particular section of a string. It returns the substringbeginning at offset and containing length which indicates the number

Fig:28string length

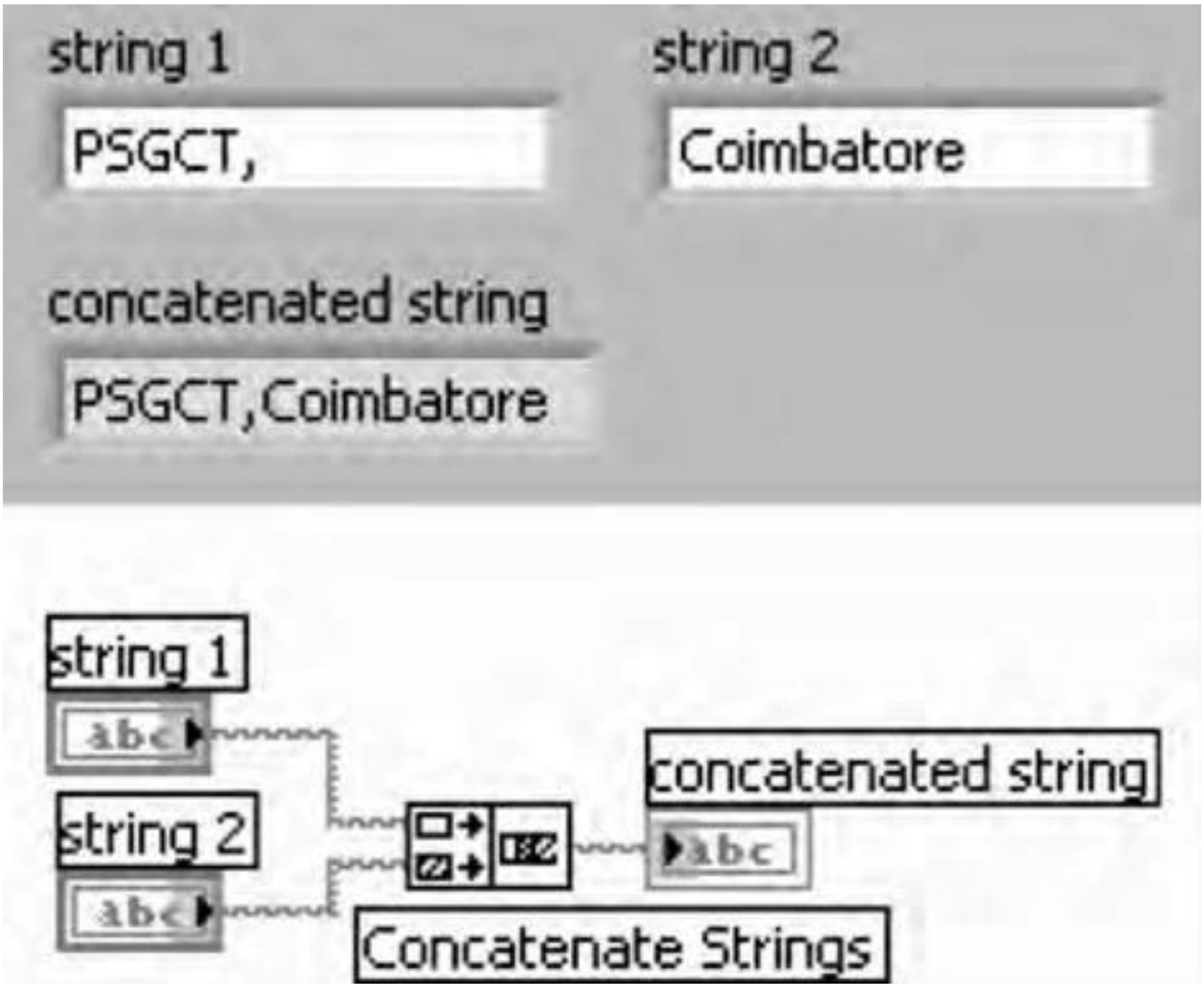


Fig 29: Concatenate Strings

Sequence Structures (Flat and Stacked Structures)

Determining the execution order of a program by arranging its elements ina certain sequence is called control flow. Visual Basic, C, and most other**Fig. 30.**

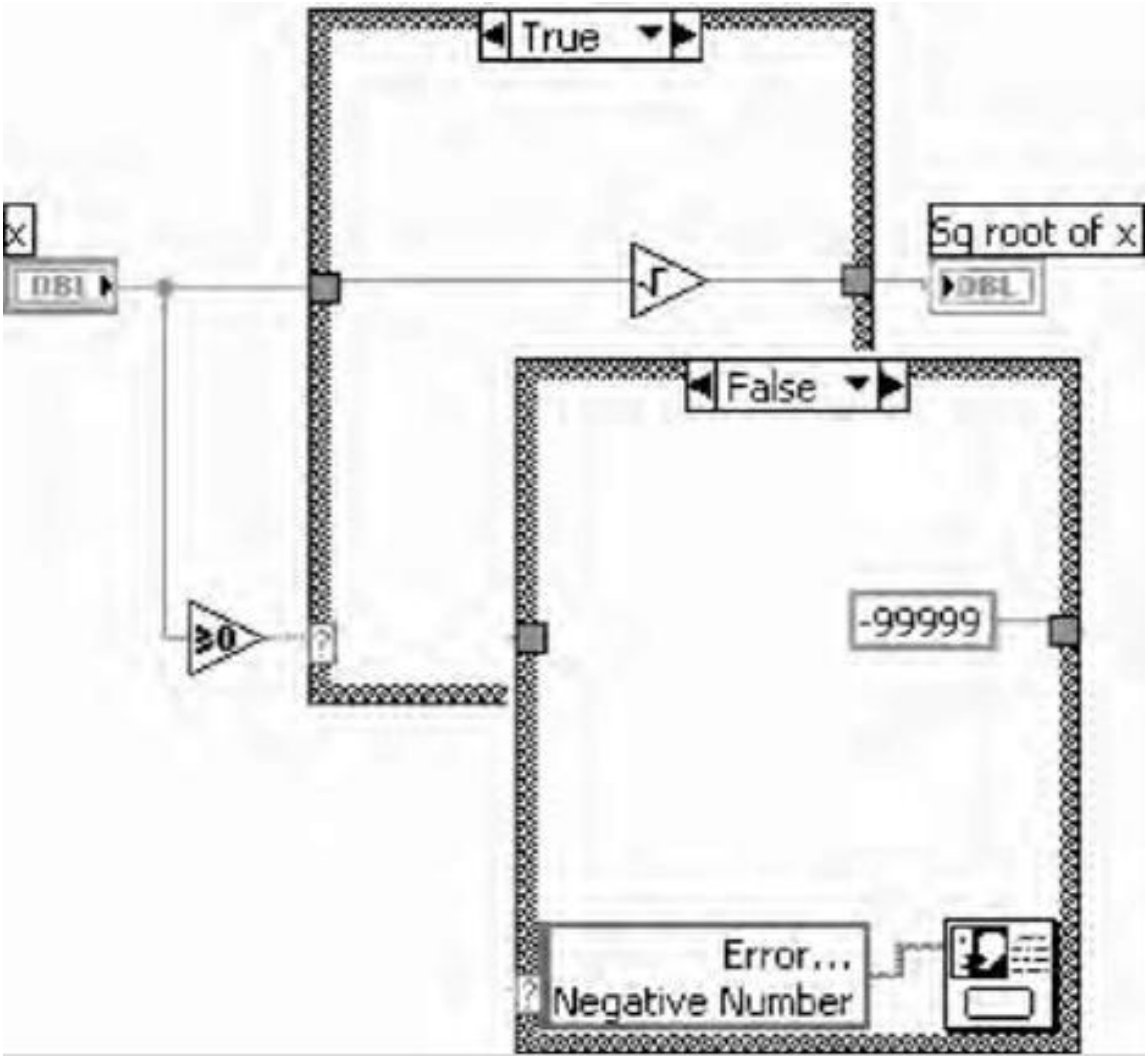


Fig30: Block diagram

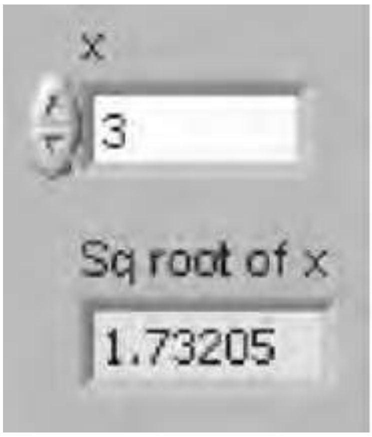


Fig 31: Front panel

procedural programming languages have inherent control flow because statementsexecutes in the order in which they appear in the program. LabVIEWuses the *Sequence Structure* to obtain control flow within a dataflow framework.A Sequence Structure executes frame 0, followed by frame 1, then frame2, until the last frame executes. Only when the last frame completes dataleaves the structure.The Sequence Structure as shown in Fig. 30, looks like a frame of film.It can be found in the **Structures** subpalette of the **Functions** palette.Like the Case Structure, only one frame is visible at a time – the arrowsat the top of the structure can be selected to see other frames; or the topdisplay can be clicked to obtain a listing of existing frames, or the user canpop-up on the structure border and choose **Show Frame**. When a SequenceStructure is first dropped on the block diagram, it has only one frame; thus,it has no arrows or numbers at the top of the structure to designate whichframe is showing. New frames can be created by popping up on the structureborder and selecting **Add Frame After** or **Add Frame Before** as shownin Fig. 31.The Sequence Structure is used to control the order of execution of nodesthat are not data dependent on each other. Within each frame, as in the rest ofthe block diagram, data dependency determines the execution order of nodes.

The Formula Node

The *Formula Node* is a resizable box that is used to enter algebraic formulasdirectly into the block diagram. The Formula Node is a convenient text-basednode used to perform mathematical operations on the block diagram. FormulaNodes are useful for equations that have many variables or are otherwisecomplicated and for using existing text-based code. The existing text-basedcode can be copied and pasted into a Formula Node rather than recreatingit graphically on the block diagram. This feature is found to be extremelyuseful when the user has a long formula to solve. For example, consider thefairly simple equation $y = x{*}{*}3 + sin(x)$. Even for this simple formula, whenimplemented equation using regular LabVIEW arithmetic functions, the blockdiagram is a little bit harder to follow than the text equationsThe same equation can be implemented using formula node. With theFormula Node, the user can directly enter a formula or formulas, in lieu ofcreating complex block diagram subsections. The task is very simple; the usercan simply enter the formula inside the box. The input and output terminalsof the Formula Node can be created by popping up on the border of the nodeand choosing **Add Input** or **Add Output** from the Pop-up menu as illustrate
in Fig32. Then enter variable names into the input and output boxes.Names are case sensitive, and each formula statement must terminate witha semicolon (;). The equation is typed in the structure. Formula Nodes canalso be used for decision-making similar to case structure and select functiondiscussed in the previous sections.

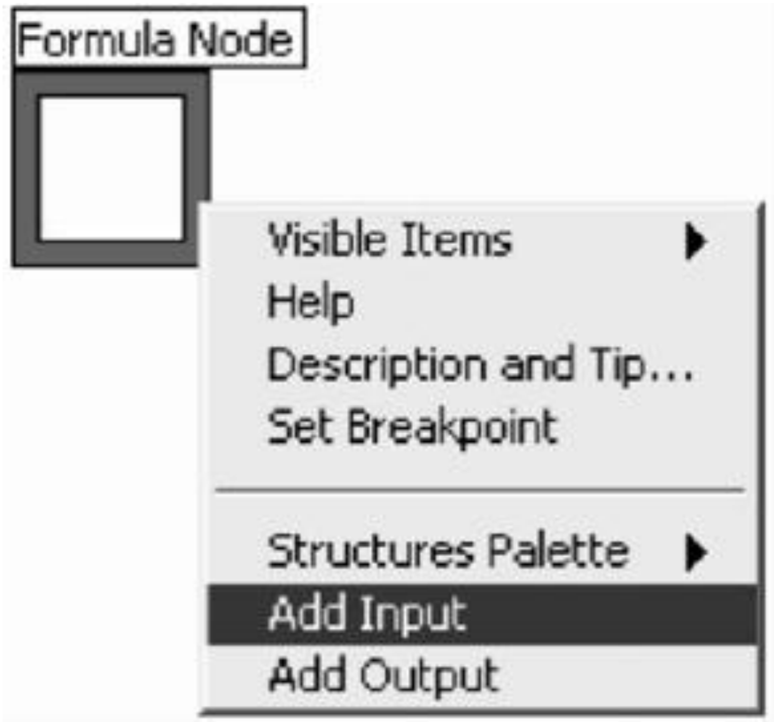


Fig32: Formula mode

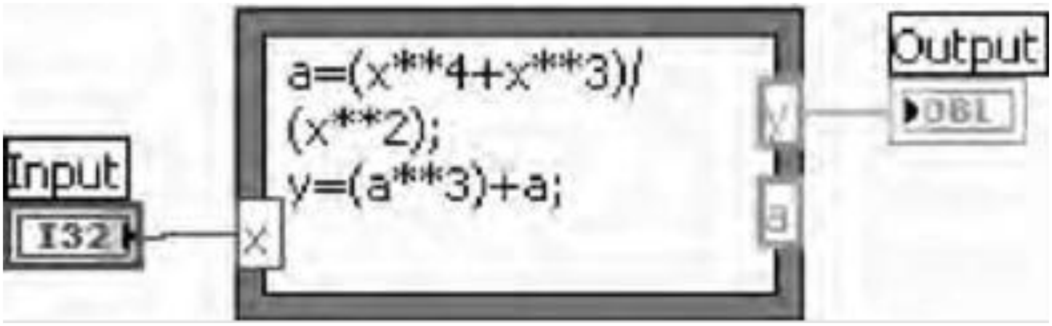


Fig33: Formula mode example

The inputis ‘x’ which is given as input to the Formula Node. The node computes thegiven function and returns the output in terms of ‘y’

Case Structures

The case structureis LabVIEW’s method of executing conditional text, sort of like an“if–then–else”statement. It is located in the **Structures** subpalette of the**Functions** palette. The Case Structure, has two or more subdiagrams, orcases; only one of them executes, depending on the value of the Boolean,numeric, or string value wired to the *selector terminal*.If a Boolean value is wired to the selector terminal, the structure has twocases, FALSE and TRUE. If a numeric or string data type is wired to theselector, the structure can have from zero to almost unlimited cases. Initiallyonly two cases are available, but number of cases can be easily added. Morethan one value can be specified for a case, separated by commas. In addition,the user can always select a “Default” case that will execute if the value wiredto the selector terminal doesn’t match any of the other cases.

When a casestructure is first placed on the panel, the Case Structure appears in its Booleanform; it assumes numeric values as soon as a numeric data type is wired toits selector terminal.Case Structures can have multiple subdiagrams, but the user can see onlyone case at a time, sort of like a stacked deck of cards. Clicking on the decrement(left) or increment (right) arrow at the top of the structure displaysthe previous or next subdiagram, respectively. The user can also click on thedisplay at the top of the structure for a pull-down menu listing all cases, andthen pop-up on the structure border and select **Show Case**. If a floating-pointnumber is wired to the selector, LabVIEW rounds that number to the nearestinteger value. LabVIEW coerces negative numbers to 0 and reduces any valuehigher than the highest-numbered case to equal the number of that case.The selector terminal can be positioned anywhere along the left border. Ifthe data type wired to the selector is changed from a numeric to a Boolean,cases 0 to 1 change to FALSE and TRUE. If other cases exist (2 to n), Lab-VIEW does not discard them, in case the change in data type is accidental.However, these extra cases must be deleted before the structure can execute.For string data types wired to case selectors, the user should always specifythe case values as strings between quotes.

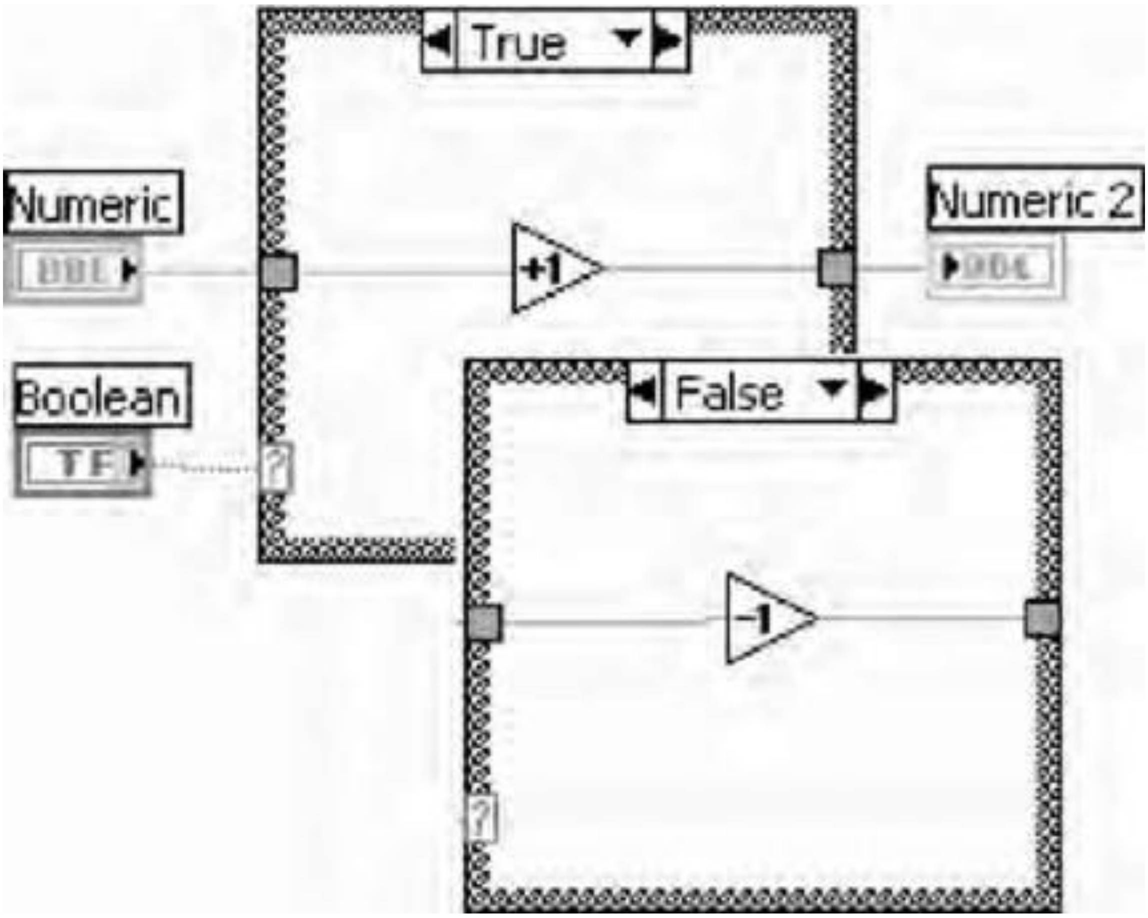


Fig 34: Boolean case structure

Boolean Case Structure

The following example is a Boolean Case structure shown in Fig34. Thecases are shown overlapped to simplify the illustration. If the Boolean controlwired to the selector terminal is True, the VI increments the numeric value.Otherwise, the VI decrements the numeric value.

Integer Case Structure

The following example is an integer Case structure shown in Fig35. **Integer**is a text ring control located on the **Controls_Text Controls** palette thatassociates numeric values with text items. If the text ring control wired to theselector terminal is 0 (add), the VI decrements the numeric values. If the valueis 1 (subtract), the VI increments the numeric values. If the text ring control isany other value than 0 (add) or 1 (subtract), the VI adds the numeric values,because that is the default case.

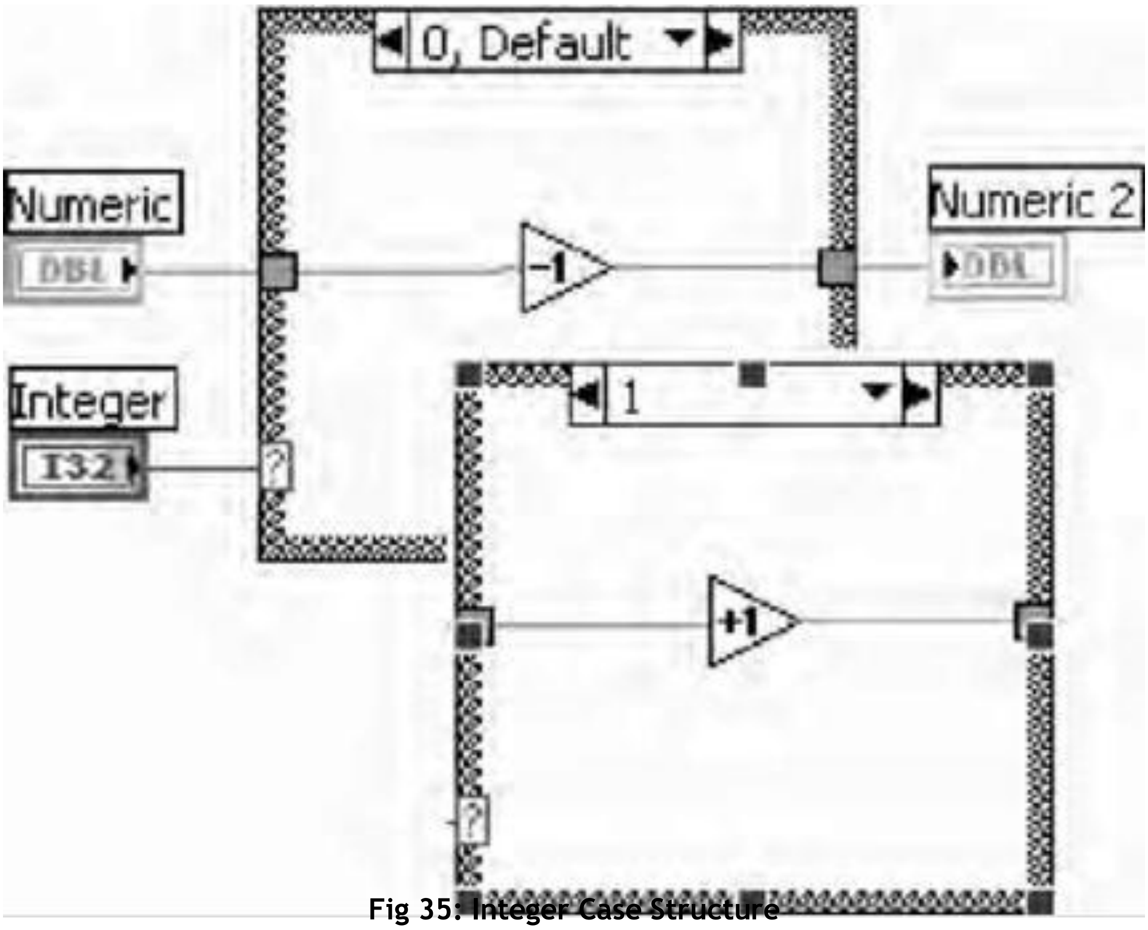


Fig 35: Integer Case Structure

String Case Structure

The following example is a string Case structure as shown in Fig36. If**String** is “Increment,” the VI increments the numeric values. If **String** is “Decrement,” the VI decrements the numeric values.

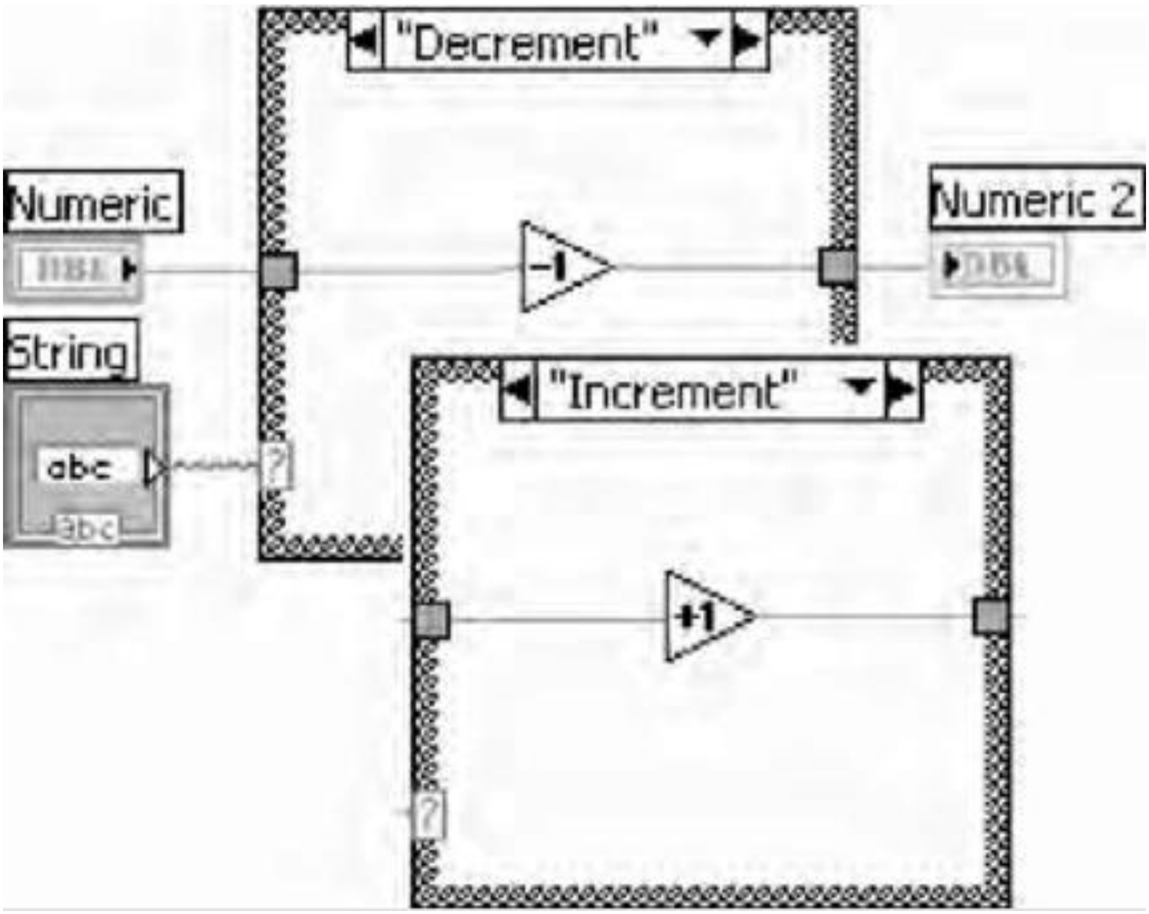


Fig 36: string case

Enumerated Case Structure

The following example is an enumerated Case structure as shown in Fig.37 An enumerated control gives users a list of items from which to select.

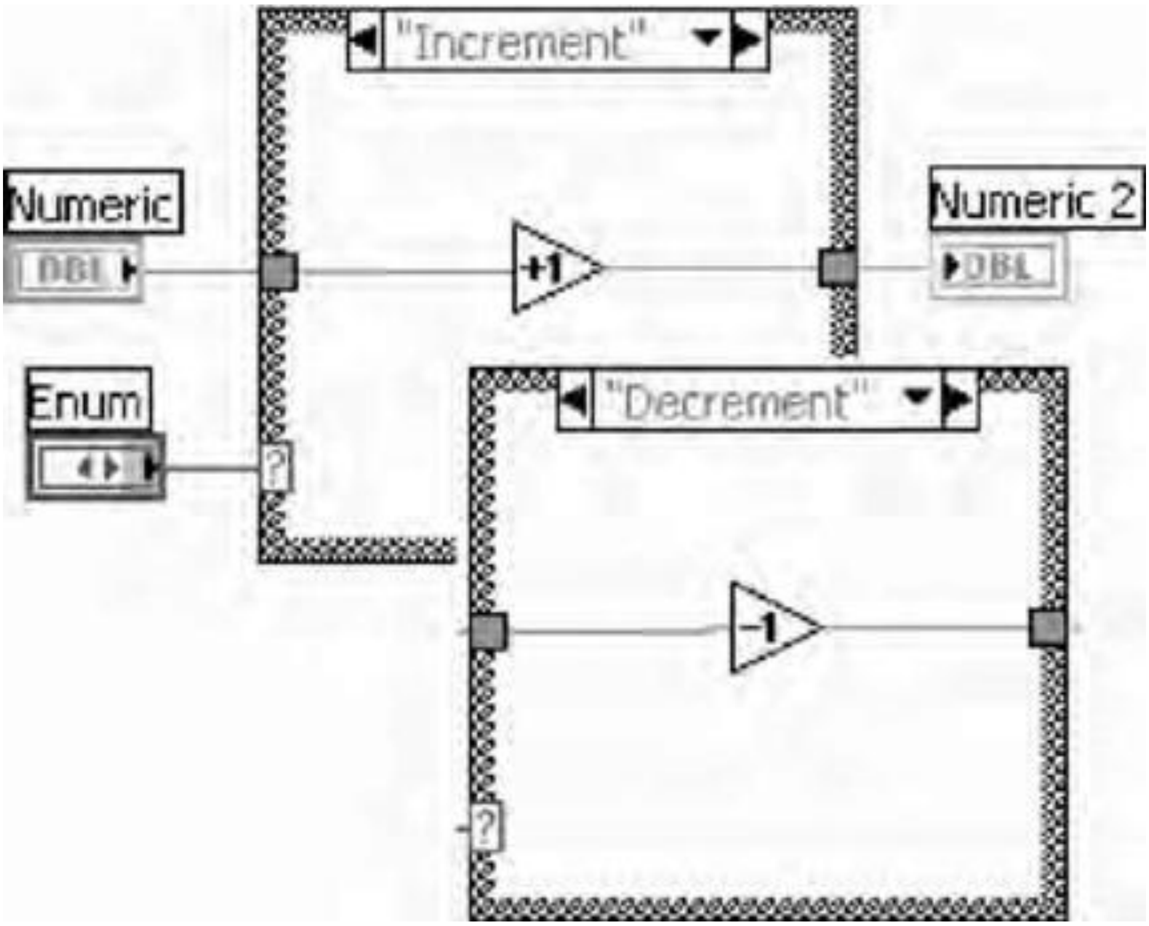


Fig:37 Enumerated case

The data type of an enumerated control includes information about the numericvalues and string labels in the control. When an enumerated control iswired to the selector terminal of a Case structure, the case selector displays acase for each item in the enumerated control. The Case structure executes theappropriate case subdiagram based on the current item in the enumeratedcontrol. If **Enum**is "Increment," the VI increments the numeric values. If**Enum**is "Decrement," the VI decrements the numeric values.

Error Case Structure

Whenan error cluster is wired to the selector terminal of a Case structure, the caseselector label displays two cases, Error and No Error, and the border of theCase structure changes color – red for Error and green for No Error.The Case structure executes the appropriate case subdiagram based onthe error state. When an error cluster is wired to the selection terminal, theCase structure recognizes only the **status** Boolean of the cluster.

SEQUENCE STRUCTURES

A sequence structure contains one or more subdiagrams, or frames, that execute in sequentialorder. Within each frame of a sequence structure, as in the rest of the block diagram, data dependencydetermines the execution order of nodes

There are two types of sequence structure the Flat Sequence structure and the Stacked Sequence structureThe Flat Sequence structure, shown as follows, displays all the frames atonce and executes the frames from left to right and when all data valueswired to a frame are available, until the last frame executes. The data valuesleave each frame as the frame finishes executing.Use the Flat Sequence structure to avoid using sequence locals and tobetter document the block diagram. When you add or delete frames in aFlat Sequence structure, the structure resizes automatically.To convert a Flat Sequence structure to a Stacked Sequence structure, right-click the FlatSequence structure and select Replace with Stacked Sequence from the shortcut menu. If youchange a Flat Sequence to a Stacked Sequence and then back to a Flat Sequence, LabVIEW movesall input terminals to the first frame of the sequence. The final Flat Sequence should operate thesame as the Stacked Sequence. After you change the Stacked Sequence to a Flat Sequence with allinput terminals on the first frame, you can move wires to where they were located in the originalFlat Sequence.

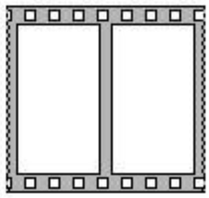


Fig:38 Flat sequence

Stacked Sequence Structure

The Stacked Sequence structure, shown as follows, stacks each frame soyou see only one frame at a time and executes frame 0, then frame 1, and soon until the last frame executes.The Stacked Sequence structure returns data only after the last frameexecutes. Use the Stacked Sequence structure if you want to conserve space

on the block diagram. To convert a Stacked Sequence structure to a FlatSequence structure, right-click the Stacked Sequence structure and selectReplace»Replace with Flat Sequence from the shortcut menu. The sequence selector identifier, shown as follows, at the top of the Stacked Sequence structure contains the

current frame number and range of frames.Use the sequence selector identifier to navigate through the available frames and rearrangeframes. The frame label in a Stacked Sequence structure is similar to the case selector label of theCase structure. The frame label contains the frame number in the center and decrement and incrementarrows on each side. Click the decrement and increment arrows to scroll through the availableframes. You also can click the down arrow next to the frame number and select a frame from thepull-down menu. Right-click the border of a frame, selectMake This Frame, and select a framenumber from the shortcut menu to rearrange the order of a Stacked Sequence structure. Unlike thecase selector label, you cannot enter values in the frame label. When you add, delete, or rearrangeframes in a Stacked Sequence structure, LabVIEW automatically adjusts the numbers in the framelabels.To pass data from one frame to any subsequent frame of a Stacked Sequence structure, use asequence local terminal shown .An outward-pointing arrow appears in the sequence local terminal of the frame that containsthe data source. The terminal in subsequent frames contains an inward-pointing arrow, indicatingthat the terminal is a data source for that frame. You cannot use the sequence local terminal inframes that precede the first frame where you wired the sequence local.

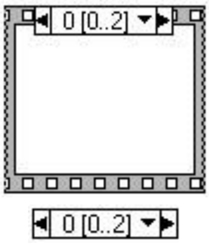


Fig 39: Stacked Structure

BASICS OF FILE INPUT/OUTPUT

File I/O records or reads data in a file. File I/O operations pass data to and from files. Use the fileI/O VIs and functions located on the Functions->All Functions»File I/O palette to handle all aspectsof file I/O, including the following:

●Opening and closing data files

●Reading data from and writing data to files

●Reading from and writing to spreadsheet-formatted files

●Moving and renaming files and directories

●Changing file characteristics

●Creating, modifying and reading configuration files

LabVIEW can use or create the following file formats: Binary, ASCII, LVM, and TDM.

●**Binary**—Binary files are the underlying file format of all other file formats.

●**ASCII**—An ASCII file is a specific type of binary file that is a standard used by mostprograms. It consists of a series of ASCII codes. ASCII files are also called text files.

●**LVM**—The LabVIEW measurement data file (.lvm) is a tab-delimited text file you canopen with a spreadsheet application or a text-editing application. The .lvm file includesinformation about the data, such as the date and time the data was generated. This fileformat is a specific type of ASCII file created for LabVIEW.

●**TDM**—This file format is a specific type of binary file created for National Instrumentsproducts. It actually consists of two separate files: an XML section contains the dataattributes and a binary file for the waveform.

Use of Text Files

Use text format files for your data to make it available to other users or applications, if disk spaceand file I/O speed are not crucial, if you do not need to perform random access reads or writes, andif numeric precision is not important. Text files are the easiest format to use and to share. Almostany computer can read from or write to a text file.

Use of Binary Files

Storing binary data, such as an integer, uses a fixed number of bytes on disk. For example, storingany number from 0 to 4 billion in binary format, such as 1, 1,000, or 1,000,000, takes up 4 bytesfor each number. Use binary files to save numeric data and to access specific numbers from a fileor randomly access numbers from a file. Binary files are machine readable only, unlike text fileswhich are human readable. Binary files are the most compact and fastest format for storing data.

Use of Datalog Files

Use datalog files to access and manipulate data only in LabVIEW and to store complex datastructures quickly and easily. A datalog file stores data as a sequence of identically structured records, similar to a spreadsheet, where each row represents a record. Each record in a datalog filemust have the same data types associated with it. LabVIEW writes each record to the file as acluster containing the data to store.

A typical file I/O operation involves the following process,
1. Create or open a file. Indicate where an existing file resides or where you want to createa new file by specifying a path or responding to a dialog box to direct LabVIEW to thefile location. After the file opens, a refnum represents the file.
2. Read from or write to the file.
3. Close the file.
File I/O VIs and some File I/O functions, such as the Read from Text File and Write to Text File functions, can perform all three steps for common file I/O operations. The VIs and functions designed for multiple operations might not be as efficient as the functions configured or designed for individual operations.

LOCAL VARIABLES

- Local variables transfer data within a single VI and allow data to be passed between parallel loops
- They also break the dataflow programming paradigm.

Two ways to create
a local variable are right-click on an object’s terminal and select Create->Local Variable.

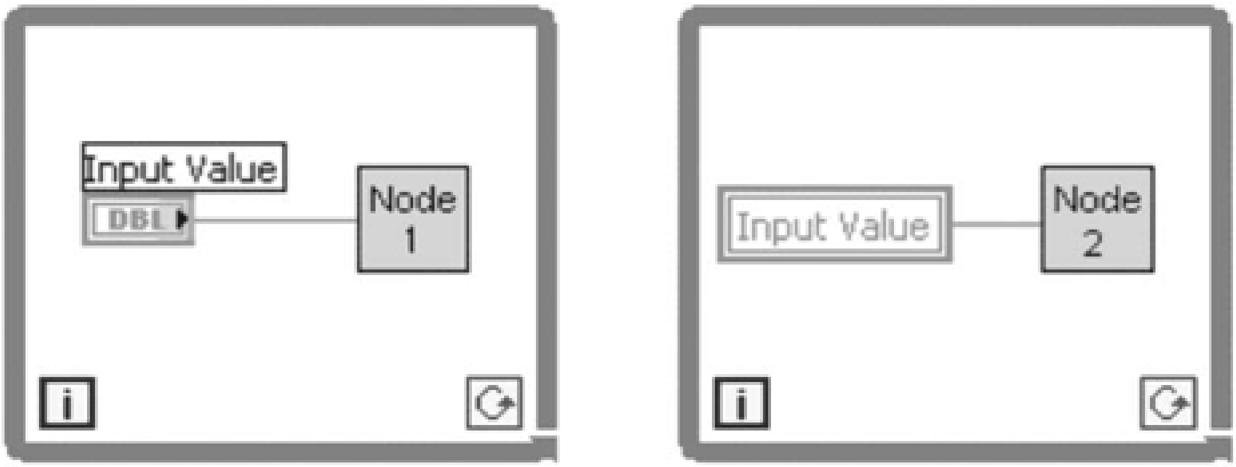


Fig 40: Local Variable

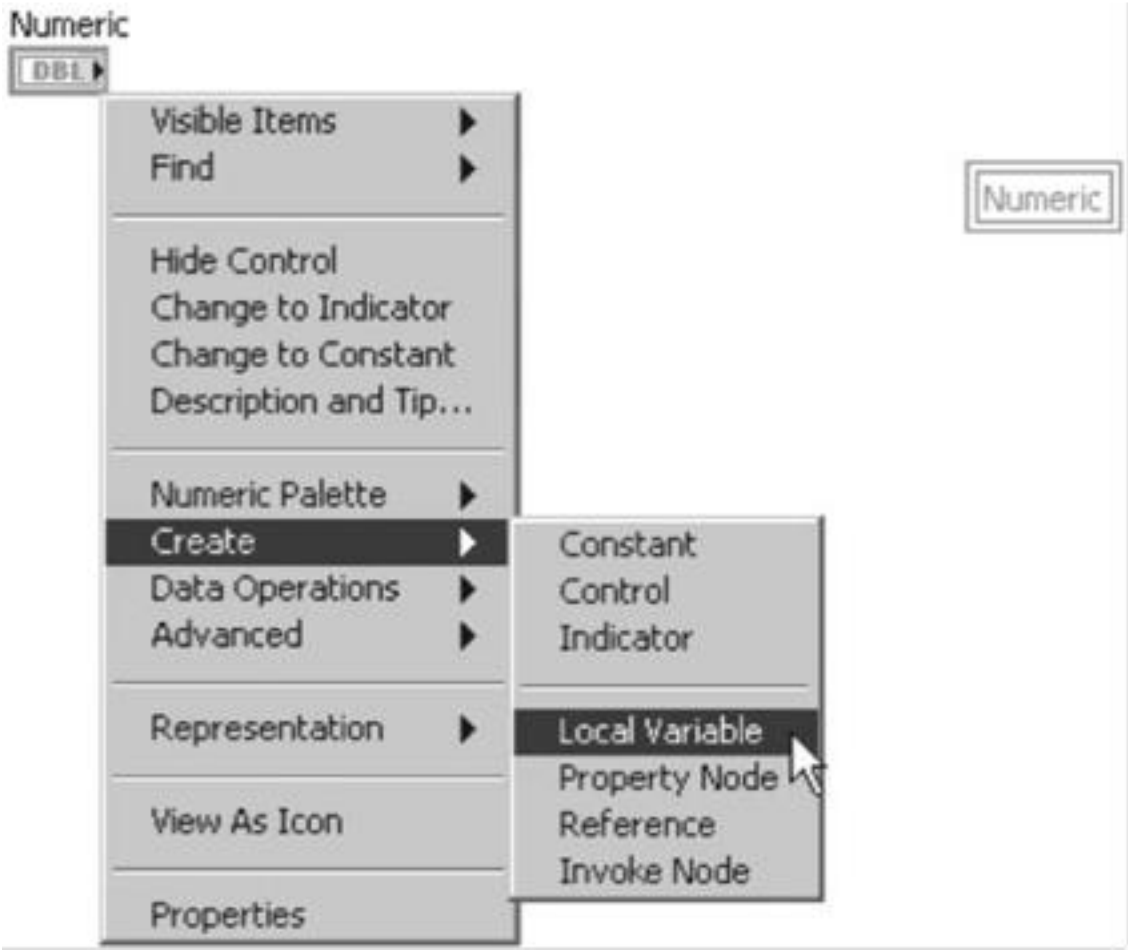


Fig41: Creating local variable

Another way is to select the Local Variable from the Structures palette. Create the frontpanel and select a local variable from the Functions palette and place it on the block diagram. Thelocal variable node, shown as follows, is not yet associated with a control or indicator. To associatea local variable with a control orindicator, right-click the local variable node and select Select Itemfrom the shortcut menu.

GLOBAL VARIABLES

- Global variables are built-in LabVIEW objects.
- use variables to access and pass data among several VIs that run simultaneously
- A local variable shares data within a VI; a global variable also shares data, but it shares data with multiple VIs.

For example, suppose you havetwo VIs running simultaneously. Each VI contains a While Loop and writes data points to awaveform chart. The first VI contains a Boolean control to terminate both VIs. You can use a

global variable to terminate both loops with a single Boolean control as shown in Figure . Ifboth loops were on a single block diagram within the same VI, you could use a local variable toterminate the loops.

When you 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 theglobal VI to define the data types of the global variables. Select a global variable as shown inFigure42 .from the *Functions* palette and place it on the block diagram. Double-click the globalvariable node to display the front panel of the global VI. Place controls and indicators on this frontpanel the same way you do on a standard front panel. LabVIEW uses owned labels to identifyglobal variables, so label the front panel controls and indicators with descriptive owned labels.

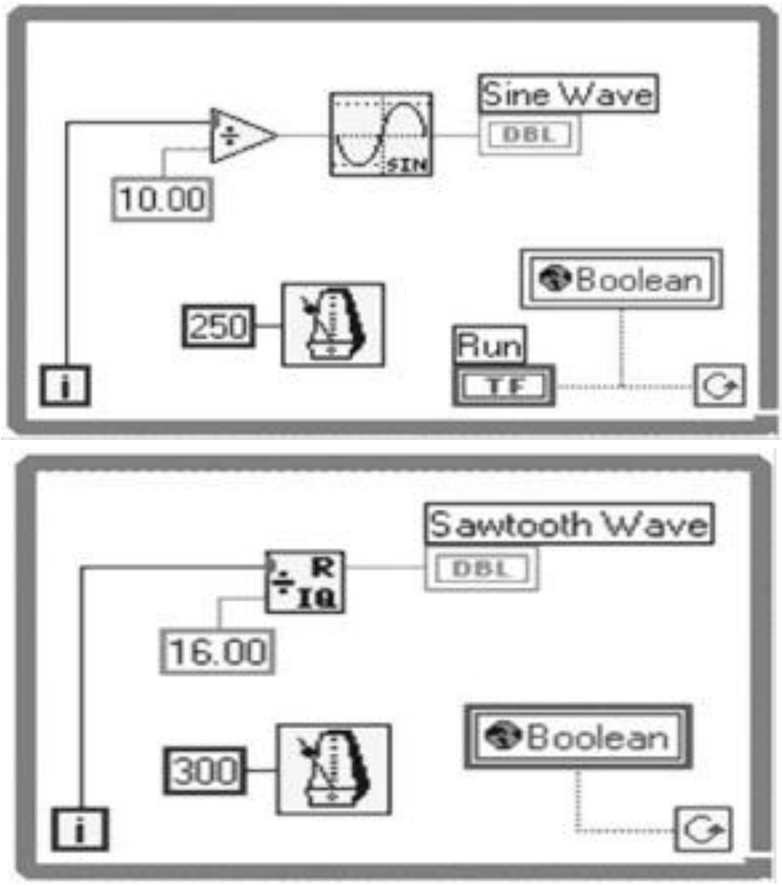


Fig:42 Global Variable

UNIT III
DATAACQUISITION

Introduction to data acquisition on PC

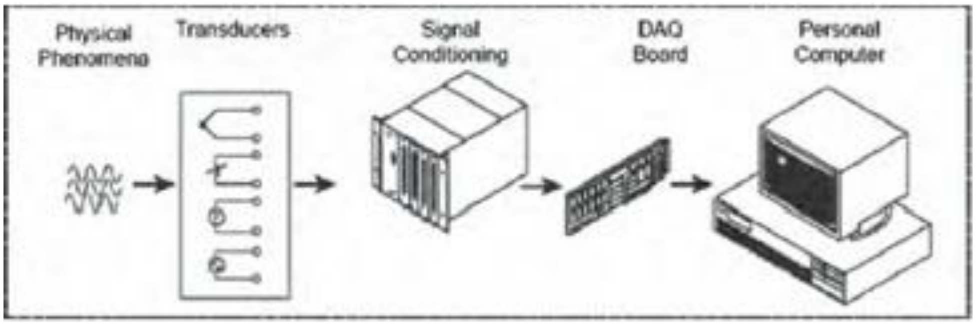


Fig 1:A general PC-based DAQ system with signal conditioning

PC-based data acquisition (DAQ) systems and plug-in boards are used in awide range of applications in the laboratory, in the field, and on the manufacturingplant floor. Typically, DAQ plug-in boards are general-purpose dataacquisition instruments that are well suited for measuring voltage signals.However, many real-world sensors and transducers output signals that mustbe conditioned before a DAQ board can effectively and accurately acquire thesignal.This front-end preprocessing, which is generally referred to as *signal conditioning*,includes functions such as signal amplification, filtering, electricalisolation, and multiplexing. In addition, many transducers require excitationcurrents or voltages, bridge completion, linearization, or high amplificationfor proper and accurate operation. Therefore, most PC-based DAQ systemsinclude some form of signal conditioning in addition to the plug-in DAQ boardand personal computer, as shown in Fig. 1.An instrument may be defined as a device or a system, which is designed tomaintain a functional relationship between prescribed properties of physicalvariables and must include ways and means of communication to a humanobserver. The functional relationship remains valid only as long as the staticcalibration of system remains constant. On the other hand, the performanceof a measurement system can be described in terms of static and dynamiccharacteristics.It is possible and desirable to describe the operation of a measuringinstrument or a system in a generalized manner without resorting to intricatedetails of the physical aspects of a specific instrument or a system. Thewhole operation can be described in terms of functional elements. Most ofthe measurement systems contain three main functional elements such as theprimary sensing element, variable conversion element, and data presentationelement. Each functional element is made up of a distinct component or groupsof components, which perform the required and definite steps in the measurement.These may be taken as basic elements, whose scope is determined bytheir functioning rather than their construction.Primary sensing element. The quantity under measurement makes its firstcontact with the primary sensing element of a

measurement system. In otherwords the measurand is first detected by primary sensor. This act is thenimmediately followed by the conversion of measurand into an analogous electricalsignal performed by a transducer. A transducer in general, is defined
as a device which converts energy from one form to another. But in Electricalmeasurement systems, this definition is limited in scope. A transducer isdefined as a device which converts a physical quantity into an electrical quantity.The physical quantity to be measured, in the first place is sensed anddetected by an element which gives the output in a different analogous form.This output is then converted into an electrical signal by a transducer. Thisis true of most of the cases but is not true for all. In many cases the physicalquantity is directly converted into an electrical quantity by a transducer. Thefirst stage of a measurement system is known as a detector transducer stage.Variable conversion element. The output of the primary sensing elementmay be electrical signal such as a voltage, a frequency or some other electricalparameter. Sometimes this output is not suited to the system. For theinstrument to perform the desired function, it may be necessary to convert thisoutput to some other suitable form while preserving the information contentof the original signal. For example, suppose output is in analog form and thenext stage of the system accepts input signals only in digital form and therefore,an A/D converter will have to be used for converting signals from A/Dform for them to be acceptable for the next stage of the system.Variable manipulation element. The function of this element is to manipulatethe signal presented to it preserving the original nature of the signal.Manipulation here means only a change in numerical value of the signal. Forexample, an electronic amplifier accepts a small voltage signal as input andproduces an output signal which is also voltage but of greater magnitude. Thusvoltage amplifier acts as a variable manipulation element. It is not necessarythat a variable manipulation element should follow the variable conversionelement as shown in Fig. 2. It may precede the variable conversion elementin many cases. In case, the voltage is too high, attenuators are used whichlower the voltage or power for the subsequent stages of the system.As discussed earlier, the output of transducer contains information neededfor further processing by the system and the output signal is usually a voltageor some other form of electrical signal. The two most important propertiesof voltage are its magnitude and frequency though polarity may be a considerationin some cases. Many transducers develop low voltages of the order

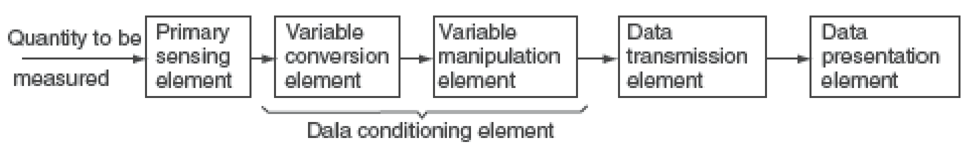


Fig 2:Functional elements of an instrumentation system

of mV and some even µV. A fundamental problem is to prevent this signalbeing contaminated by unwanted signals like noise due to an extraneous sourcewhich may interfere with the original output signal. Mother problem is that aweak signal may be distorted by processing equipment. The signal after beingsensed cannot transmitted to the next stage without removing the interferingsources, as otherwise highly distorted results may be obtained which alefar from true. Many a times it becomes necessary to perform certain operationson the signal before it is transmitted further. These processes may belinear like amplification, attenuation, integration, differentiation, addition andsubtraction. Some nonlinear processes like modulation, detection, sampling,filtering, chopping and clipping, etc. are also performed on the signal to bringit to the desired form to be accepted by the next stage of measurement system.This process of conversion is called Signal Conditioning. The term signalconditioning includes many other functions in addition to variable conversionand variable manipulation. In fact the element that follows the primarysensing element in any instrument or measurement system is called SignalConditioning Element.When the elements of an instrument are actually physically separated, itbecomes necessary to transmit data from one to another. The element thatperforms this function is called a Data Transmission Element. For example,space-crafts are physically separated from the earth where the control stationsguiding their movements are located. Therefore control signals are sent fromthese stations to space-crafts by complicated telemetry systems using radiosignals. The signal conditioning and transmission stage is commonly knownas Intermediate Stage.

In the signal conditioning stage, the signals are passed through a slew ofsteps like:
– Amplification
– Rectification
– Filtering
– Scaling
Data presentation element.The information about the quantity under measurementhas to be conveyed to the personnel handling the instrument orthe system for monitoring, control, or analysis purposes. The informationconveyed must be in a form intelligible to the personnel or to the intelligentinstrumentation system. This function is done by data presentation element.In case data is to be monitored, visual display devices are needed. Thesedevices may be analog or digital indicating instruments like ammeters, voltmeters,etc. In case the data is to be recorded, recorders like magnetic tapes,high speed camera and TV equipment, storage type C.R.T, printers, analogand digital computers, or microprocessors may be used. For control andanalysis purpose microprocessors or computers may be used.The final stage in a measurement system is known as terminating stage.As an example of a measurement system, consider the simple bourdon tube

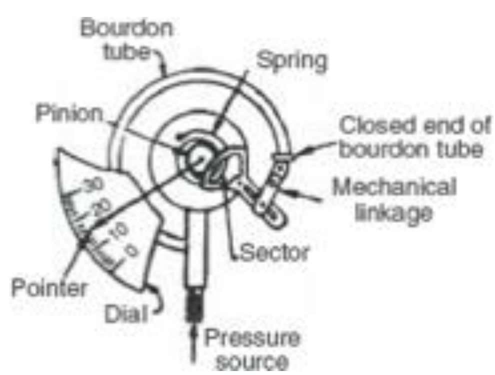


Fig 3:Bourdon tube pressure gauge

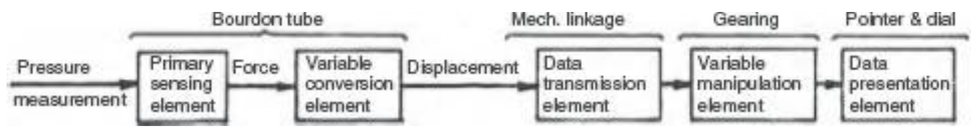


Fig 4:Schematic diagram of a Bourdon tube pressure gauge

pressure gauge as shown in Fig. 3. This gauge offers a good example of a measurementsystem. In this case the bourdon tube acts as the primary sensingelement and a variable conversion element. It senses the input quantity (pressurein this case). On account of the pressure the closed end of the bourdontube is displaced. Thus the pressure is converted into a small displacement.The closed end of the bourdon tube is connected through mechanical linkageto a gearing arrangement. The gearing arrangement amplifies the smalldisplacement and makes the pointer to rotate through a large angle. Themechanical linkage thus acts as a data transmission element while the gearingarrangement acts as a data manipulation element.The final data presentation stage consists of the pointer and dial arrangement;which when calibrated with known pressure inputs, gives an indicationof the pressure signal applied to the bourdon tube. The schematic diagram ofthis measurement system is given in Fig. 4.When a control device is used for the final measurement stage, it is necessaryto apply some feedback to the input signal to accomplish the controlobjectives. The control stage compares the signal representing the measuredvariable with a reference signal of the same form. This reference signal has avalue the measured signal should have and is presented to a controller. If themeasured signal agrees with the reference value, the controller does nothing.However, if there is a difference between the measured value and the referencevalue, an error signal is generated. Thus the controller sends a signal to adevice which acts to alter the value of the measured signal. Suppose the measuredvariable is flow of a liquid, then the control device is a motorized valveplaced in the flow system. In case the measured flow rate is too low than the preset flow rate, then the controller would cause the valve to open, therebyincreasing the flow rate. If on the other hand, the flow rate were too high, thevalves are closed. The operation of closing or opening of valve will cease whenthe output flow rate is equal to preset value of flow rate. The physical phenomenon,which is the signal from external world faces signal abnormalities,so these have to be rectified in order to make the signal more effective.

Sampling fundamentals

A fundamental rule of sampled data systems is that the input signal must besampled at a rate greater than twice the highest frequency component in thesignal. This is known as the Nyquistcriterion. Stated as a formula, it requiresthat fs/2 >fa, where fsis the sampling frequency and fais the signal beingsampled. Violating the Nyquist criteria is called under sampling, and resultsin aliasing.

Filtering and Averaging

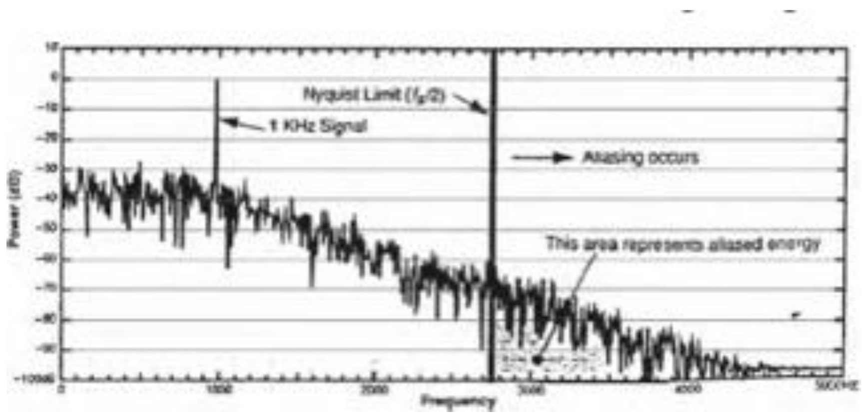


Fig 5:Power spectrum of a 1 kHz sine wave with low pass-filtered noise added

To get rid of aliasing, a type of low-pass filter has to be used, known asan anti-aliasing filter. Analog filters are absolutely mandatory, regardless of the sampling rate, unless the signal's frequency characteristics are known.In order to limit the bandwidth of the raw signal to *fs/*2 an analog filter isused. The analog filter can be in the transducer, the signal conditioner, on theA/D board, or in all three sections. The filter is usually made up of resistors,capacitors, and sometimes, operational amplifiers, in which case it is called anactive filter. One problem with analog filters is that they are very complex andexpensive. If the desired signal is fairly close to the Nyquist limit, the filterneeds to cut off very quickly, which requires lots of stages, which depends onthe order of the filter.Digital filters can augment, but cannot replace, analog filters. Digital filterVIs is included with the LabVIEW analysis VIs, and they are functionallyequivalent to analog filters. The simplest type of digital filter is a movingaverager, which has the advantage of being usable in real-time on a sampleby-sample basis. One way to simplify the anti-aliasing filter problem is to oversample the input. If the A/D hardware is fast enough, the sampling rate isfirst increased and then a digital filter is used to eliminate the higher frequenciesthat are of no interest. This makes the analog filtering design problemsimpler because the Nyquist frequency has been raised much higher, so theanalog filter need not be sharp. The signal has to be sampled at a rate highenough to avoid significant aliasing with a modest analog filter, but samplingat too high rate may not be practical because the hardware is too expensiveand extra data may overload the CPU. Figure 4.9 illustrates the occurrenceof aliasing in a 1 kHz sine wave with low pass filtered noise in it.A potential problem with averaging comes up while handling nonlineardata. The process of averaging is defined to be the summation of severalvalues, divided by the number.If the main concern is rejecting 60 Hz line frequency interference, an oldtrick is to grab an array of samples over one line period (16.66 ms). Thisshould be done for every channel. Using plug-in boards with LabVIEW's dataacquisition drivers, the user can adjust the sampling interval with high precision,making this a reasonable option. First, a simple experiment is set to

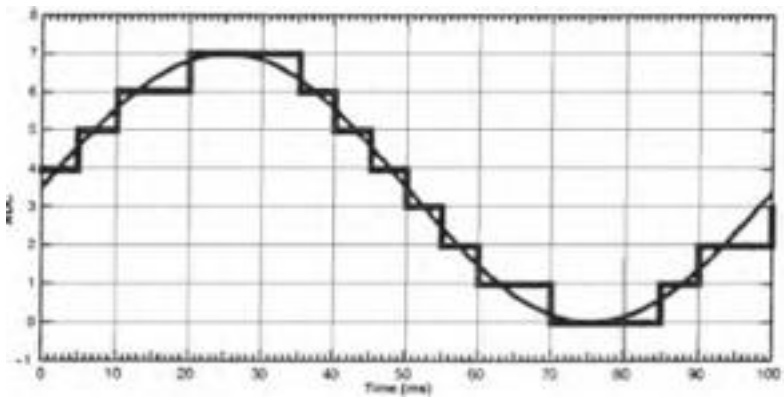


Fig 6:A sine wave and its representation by a 3-bit ADC sampling every 5 ms

acquire and average data from a noisy input. The sampling period is variedand checked for a null noise at each 16.66 ms multiple.If there is an attempt to average recurrent waveforms to reduce noise, thearrays of data that are acquired must be perfectly in-phase. If a phase shiftoccurs during acquisition, then the waveforms will partially cancel each otheror distort in some other way. Triggered data acquisition is the normal solutionbecause it helps to guarantee that each buffer of data is acquired at the samepart of the signal's cycle.Some other aspects of filtering that may be important to some of theapplications are impulse response and phase response. For ordinary datalogging, these factors are generally ignored. But if the user is doing dynamictesting like vibration analysis, acoustics, or seismology, impulse and phaseresponse can be very important. As a rule of thumb, when filters becomevery complex (high-order), they cutoff more sharply, have more radical phaseshifts around the cutoff frequency, and (depending on the filter type) exhibitringing on transients.The best way to analyze filtering needs is to use a spectrum analyzer sincewe are able to know exactly what signals are present and what has to befiltered out. Some common methods used for filtering are:– A dedicated spectrum analyzer instrument (very expensive)– A digital oscilloscope with FFT capability (LabVIEW computes the powerspectrum) – Even a multifunction I/O board running as fast as possible with LabVIEW doing the power spectrum.Looking at the power spectrum in Fig. 4.10, some of the noise power is abovethe floor for the ADC and is also at frequencies above 2.75 kHz. This littletriangle represents aliased energy and gives the user a qualitative feel for howmuch contamination the user can expect. It depends on the nature of theout-of-band noise.

Analog-to-Digital Control (ADC)

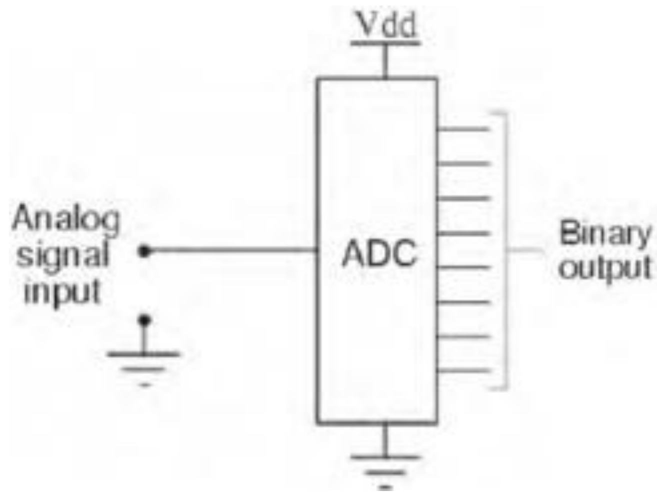


Fig 7:Analog-to-digital converter

Connecting digital circuitry to sensor devices is simple if the sensor devices areinherently digital in nature. witches, relays, and encoders are easily interfacedwith gate circuits due to the on/off nature of their signals. However, whenanalog devices are involved, interfacing becomes much more complex. Somemechanism is needed to electronically translate analog signals into digital(binary) quantities, and visa-versa. An analog-to-digital converter, or ADC,performs the former task while a digital-to-analog converter, or DAC, performsthe latter. An ADC inputs an analog electrical signal such as voltage or currentand outputs a binary number. In block diagram form, it can be representedas shown in Fig. 7.

Understanding Integrating ADCs:

Integrating ADCs provide high resolution A/D conversions, with good noiserejection.TheseADCsare ideal fordigitizing lowbandwidthsignals,andare usedin applications such as digitalmultimeters and panel meters. They often includeLCD or LED drivers and can be used stand alone without a microcontrollerhost. The following section explains how integrating ADCs work. Discussionsinclude single-, dual- and multislope conversions. Also, an in-depth analysis ofthe integrating architecture will be discussed.Finally a comparison against other ADC architectures will aid in theunderstanding and selection of integrating ADCs. Integrating analog-to-digitalconverters (ADCs) provide high resolution and can provide good line frequencyand noise rejection. Having started with the ubiquitous 7106, theseconverters have been around for quite some time. The integrating architectureprovides a novel and straightforward approach to converting a low bandwidth analog signal into its digital representation. These types of converters ofteninclude built-in drivers for LCD or LED displays and are found in manyportable instrument applications, including digital panel meters and digitalmultimeters.

Single-Slope ADC Architecture

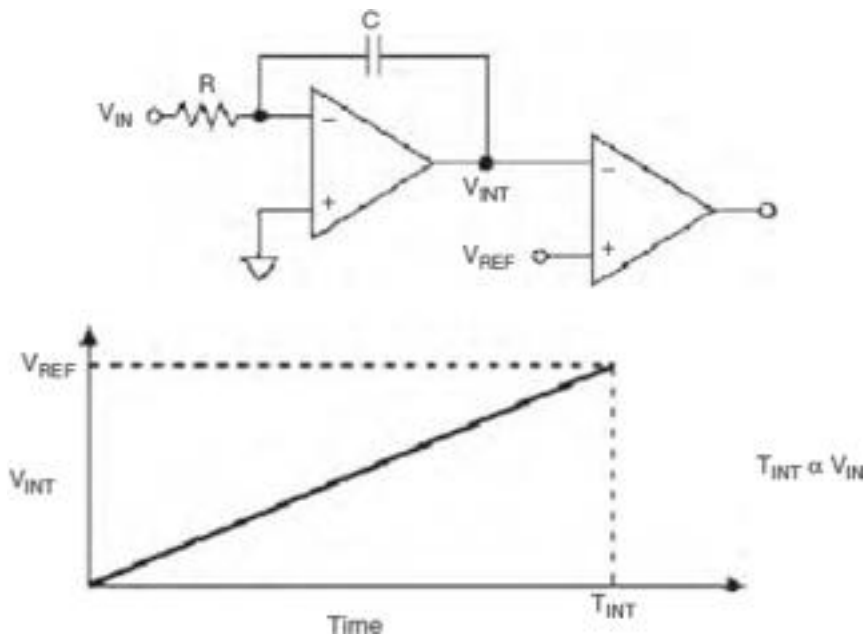


Fig 8:(a) Single-slope ADC circuit (b) Response of single slope ADC

The simplest form of an integrating ADC uses single-slope architecture(Figs. 8a, b). Here, an unknown input voltage is integrated and the value iscompared against a known reference value. The time it takes for the integratorto trip the comparator is proportional to the unknown voltage (VINT/VIN).In this case, the known reference voltage must be stable and accurate to guaranteethe accuracy of the measurement.One drawback to this approach is that the accuracy is also dependenton the tolerances of the integrator's R and C values. Thus in a productionenvironment, slight differences in each component's value change the conversionresult and make measurement repeatability quite difficult to attain. Toovercome this sensitivity to the component values, the dual-slope integratingarchitecture is used.

Dual-Slope ADC Architecture

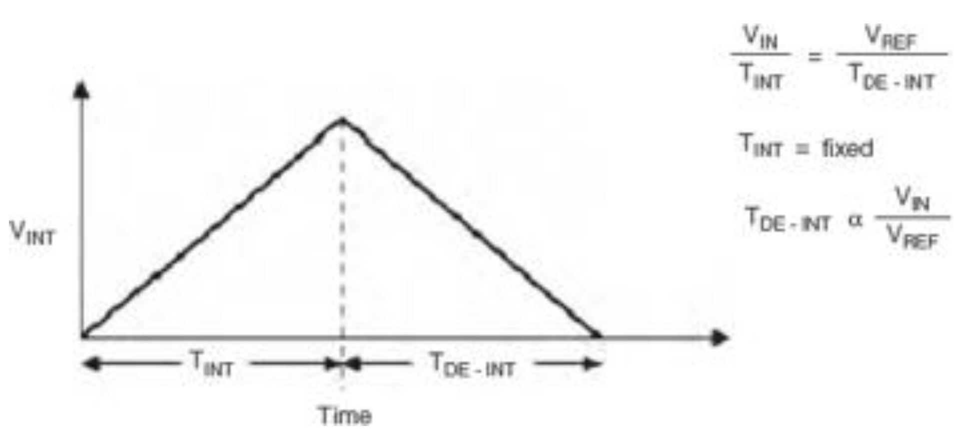


Fig 9:Dual slope integration

A dual-slope ADC (DS-ADC) integrates an unknown input voltage (VIN) fora fixed amount of time (TINT), then “disintegrates” (TDEINT) using a knownreference voltage (VREF) for a variable amount of time (Fig. 9).
The key advantage of this architecture over the single-slope is that thefinal conversion result is insensitive to errors in the component values. Thatis, any error introduced by a component value during the integrate cycle willbe canceled out during the de-integrate phase. In above equation form:

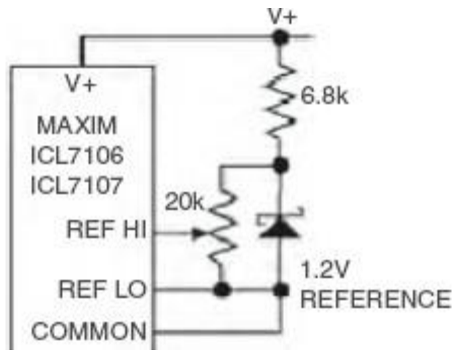


Fig 10:Dual Slope Converter

VIN∗TINT = VREF∗TDEINT or TDEINT = TINT(VIN/VREF)

From this equation, it is found that the de-integrate time is proportional tothe ratio of VIN/VREF. A complete circuit diagram of a dual-slope converteris shown in Fig. 10.As an example, to obtain 10-bit resolution, integration is performed for1024 (210) clock cycles, then disintegrated for 1024 clock cycles (giving amaximum conversion of two 210 cycles). For more resolution, increase thenumber of clock cycles. This tradeoff between conversion time and resolutionis inherent in this implementation. It is possible to speed up the conversiontime for a given resolution with moderate circuit changes. Unfortunately, allimprovements shift some of the accuracy to matching, external components,charge injection, etc. In other words, all speed-up techniques have largererror budgets. Even in the simple converter in Fig. 4.1, there are manypotential error sources to consider such as power supply rejection (PSR),common-mode rejection (CMR), finite gain, over-voltage concerns, integratorsaturation, comparator speed, comparator oscillation, “rollover,” dielectricabsorption, capacitor leakage current, parasitic capacitance, charge injection,etc.

SAR ADC:

Successive-approximation-register (SAR) analog-to-digital converters (ADCs)represent the majority of the ADC market for medium to high resolutionADCs. SAR ADCs provide up to 5 Msps (mega samples per second) samplingrate with resolution from 8 to 18 bits. The SAR architecture allows for highperformance, low power ADCs to be packaged in small form factors for today’sdemanding applications. It also provides an explanation for the heart of theSAR ADC, the capacitive DAC and also the high-speed comparator.Finally, the article will contrast the SAR architecture against pipeline,flash ADCs. SAR ADCs are frequently the architecture of choice for mediumto-high-resolution applications with sample rates under 5 Msps. SAR ADCsmost commonly range in resolution from 8 to 16 bits and provide low powerconsumption as well as a small form factor. This combination makes themideal for a wide variety of applications, such as portable/battery-poweredinstruments, pen digitizers, industrial controls, and data/signal acquisition.As the name implies, the SAR ADC basically implements a binary searchalgorithm. Therefore, while the internal circuitry may be running at severalmegahertz (MHz), the ADC sample rate is a fraction of that number due tothe successive-approximation algorithm.

Architecture:

Although there are many variations in the implementation of an SAR ADC,the basic architecture is quite simple (Fig. 11). The analog input voltage(VIN) is held on a track/hold. To implement the binary search algorithm, theN-bit register is first set to mid scale (that is, 100 00, where the MSB isset to “1”). This forces the DAC output (VDAC) to be VREF/2, where VREF isthe reference voltage provided to the ADC. A comparison is then performedto determine if VIN is less than or greater than VDAC.If VIN is greater than VDAC, the comparator output is at logic high or “1”and the MSB of the N-bit register remains at “1.” Conversely, if VIN is lessthan VDAC, the comparator output is logic low and the MSB of the register iscleared to logic “0.” The SAR control logic then moves to the next bit down,forces that bit high, and does another comparison. The sequence continues allthe way down to the LSB. Once this is done, the conversion is complete, andthe N-bit digital word is available in the register.The y-axis (and the bold line in Fig. 4.33) represents the DAC outputvoltage. In the example, the first comparison shows that VIN <VDAC. Thus,bit 3 is set to “0.” The DAC is then set to 01002 and the second comparisonis performed. As VIN >VDAC, bit 2 remains at “1.”

The DAC is then set to 01102, and the third comparison is performed. Bit 1 is set to “0,” and theDAC is then set to 01012 for the final comparison. Finally, bit 0 remains at“1” because VIN >VDAC.

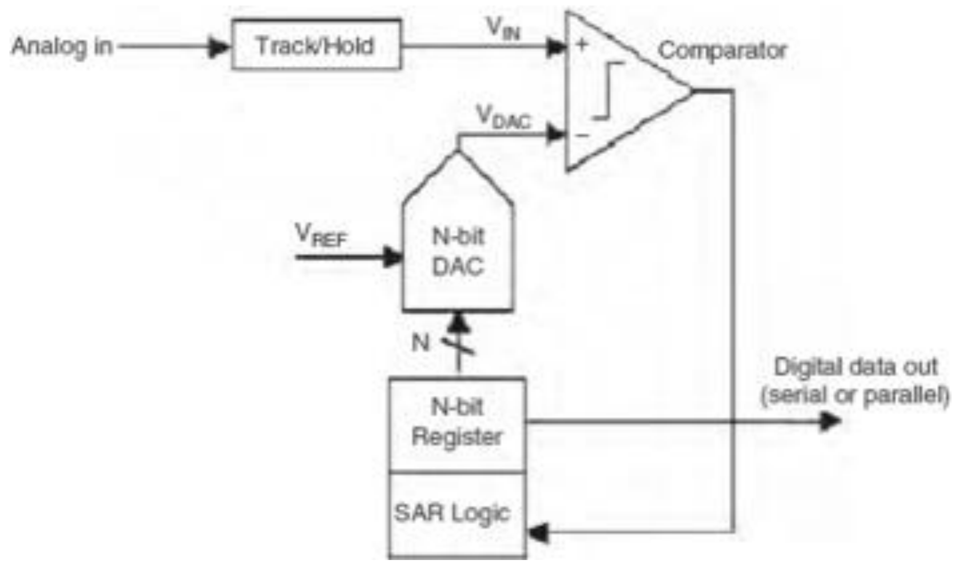


Fig 11:Simplified N-bit SAR ADC architecture

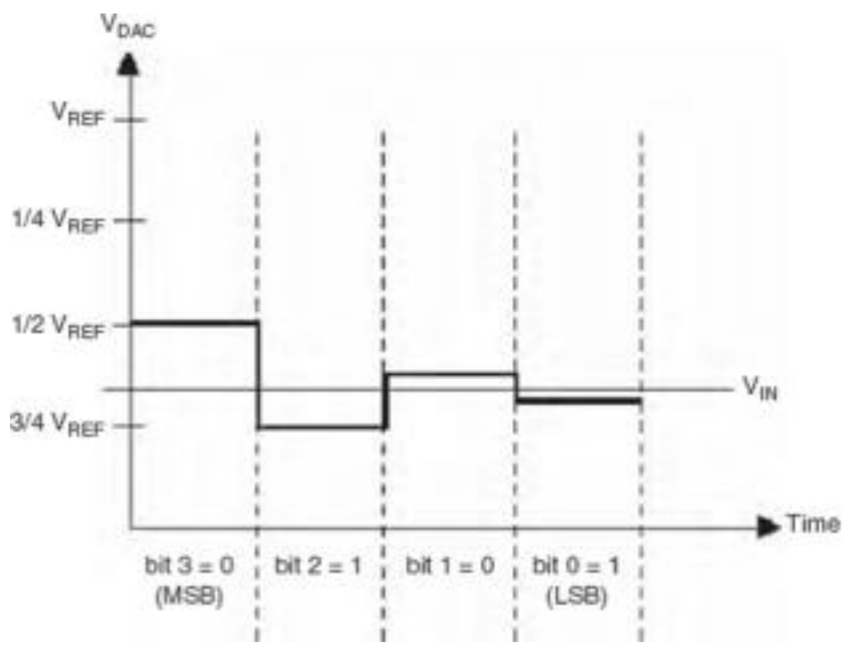


Fig 12: SAR operation

Four comparison periods are required for a 4-bit ADC. Generally speaking,an N-bit SAR ADC will require N comparison periods and will not be readyfor the next conversion until the current one is complete. Thus, these typesof ADCs are power and space-efficient. Some of the smallest ADCs availableon the market are based on the SAR architecture.The MAX1115-MAX1118 series of 8-bit ADCs as well as their higherresolutioncounterparts, the MAX1086 and the MAX1286 (10 and 12 bits, respectively), fit in tiny SOT23 packages measuring 3mm by 3 mm. Anotherfeature of SAR ADCs is that power dissipation scales with the sample rate,unlike flash or pipelined ADCs, which usually have constant power dissipationversus sample rate. This is especially useful in low-power applications orapplications where the data acquisition is not continuous as in the case ofPDA Digitizers.

DAC

A digital to analog converter (DAC) converts a digital signal to an analog voltage or current output

Types of DACs

- Many types of DACs available.
- Usually switches, resistors, and op-amps used to implement conversion
- Two Types:
    - Binary Weighted Resistor
        - R-2R Ladder

Binary Weighted Resistor

Utilizes a summing op-amp circuit Weighted resistors are used to distinguish each bit from the most significant to the least significant Transistors are used to switch between Vref and ground (bit high or low)

- Assume Ideal Op-amp
- No current into op-amp
- Virtual ground at inverting input
- $V_{out} = -IR_f$

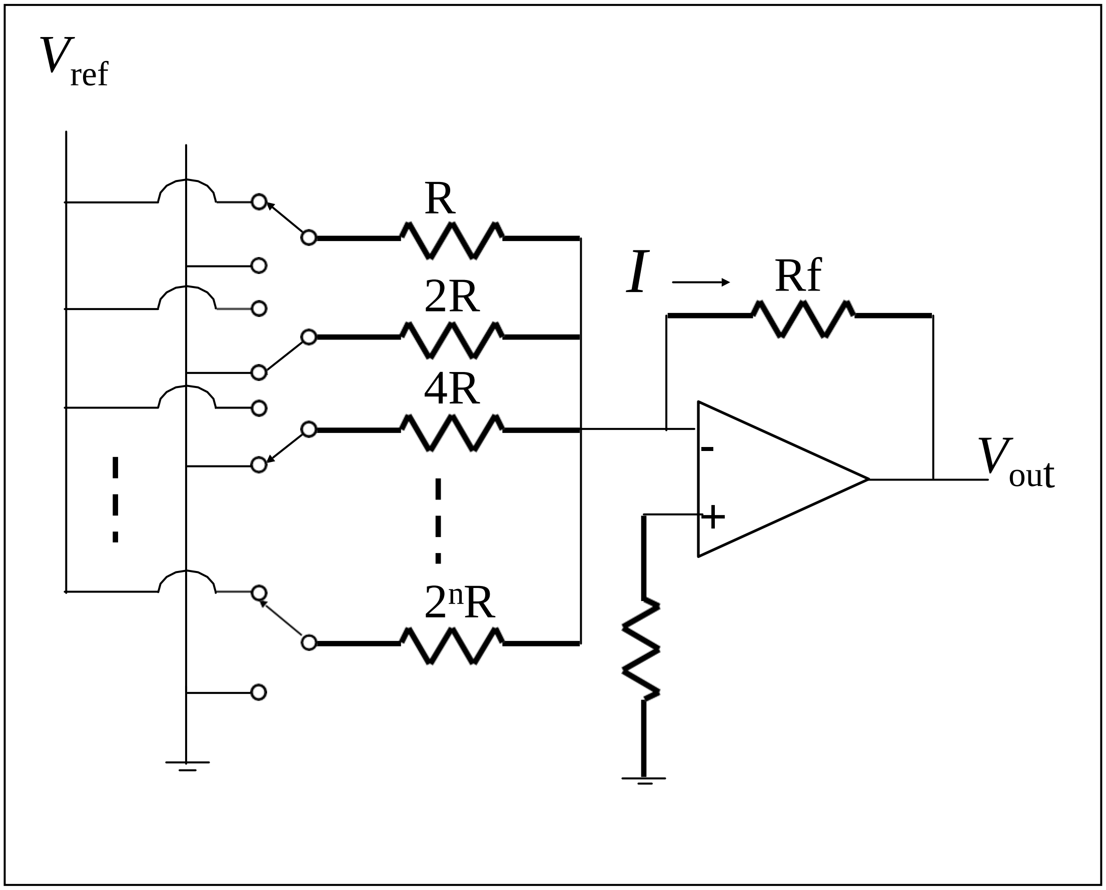


Fig 13: Binary Weighted Resistor

R-2R Ladder

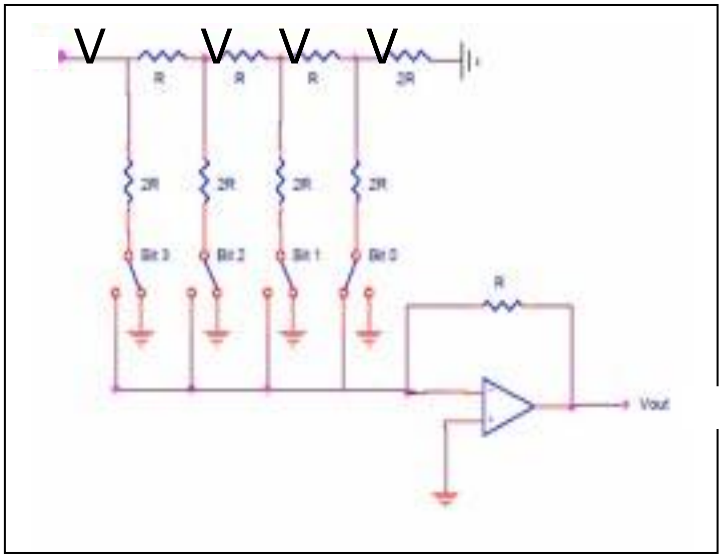


Fig 14: R-2R Ladder

For a 4-Bit R-2R Ladder

$$V_{out} = -V_{ref}\left(b_3\frac{1}{2} + b_2\frac{1}{4} + b_1\frac{1}{8} + b_0\frac{1}{16}\right)$$

Advantages

- Only two resistor values (R and 2R)
- Does not require high precision resistors

Disadvantage

- Lower conversion speed than binary
- weighted DAC

Specifications of DACs

- Resolution
- Speed
- Linearity
- Settling Time
- Reference Voltages
- Errors

Resolution

- Smallest analog increment corresponding to 1 LSB change
- An N-bit resolution can resolve 2^N distinct analog levels

Common DAC has a 8-16 bit resolution

Speed

- Rate of conversion of a single digital input to its analog equivalent
- Conversion rate depends on
    - clock speed of input signal
    - settling time of converter
- When the input changes rapidly, the DAC conversion speed must be high.

Linearity

- The difference between the desired analog output and the actual output over the full range of expected values

Applications

- Digital Motor Control
- Computer Printers
- Sound Equipment (e.g. CD/MP3 Players, etc.)
- Electronic Cruise Control

Counters and Timers

A counter is a digital timing device. Typical applications of counters areevent counting, frequency measurement, period measurement, position measurement,and pulse generation. A counter contains the following four maincomponents:

– **Count Register**. Stores the current count of the counter. The user canquery the count register with software.

– **Source**. An input signal that can change the current count stored in thecount register. The counter looks for rising or falling edges on the sourcesignal. Whether a rising or falling edge changes the count is softwareselectable. The type of edge selected is referred to as the active edge ofthe signal. When an active edge is received on the source signal, the countchanges. Whether an active edge increments or decrements the current count is also software selectable.

– **Gate**. An input signal that determines if an active edge on the sourcewill change the count. Counting can occur when the gate is high, low, orbetween various combinations of rising and falling edges. Gate settingsare made in software.

– **Output**. An output signal that generates pulses or a series of pulses,otherwise known as a pulse train.When a counter is configured for simpleevent counting, the counter increments when an active edge is receivedon the source. In order for the counter to increment on an active edge,the counter must be started. A counter has a fixed number it can count to as determined by the resolution of the counter. For example, a 24-bitcounter can count to:

2(Counter Resolution) − 1 = 224 − 1 = 16, 777, 215

When a 24-bit counter reaches the value of 16,777,215, it has reached theterminal count. The next active edge will force the counter to roll overand start at 0.

DMA

D*irect* **m***emory* **a***ccess*, a technique for transferring datafrom main memory to a device without passing it through the CPU.

Computers that have DMA channels can transfer data to and from devices much more quickly than computers without a DMA channel can. This is useful for making quick backups and for *real-time*applications.Some expansion boards, such as CD-ROM cards, are capable of accessing the computer's DMA channel.

Modes of operation

Burst mode

An entire block of data is transferred in one contiguous sequence. Once the DMA controller is granted access to the system bus by the CPU, it transfers all bytes of data in the data block before releasing control of the system buses back to the CPU, but renders the CPU inactive for relatively long periods of time. The mode is also called "Block Transfer Mode". It is also used to stop unnecessary data.

Cycle stealing mode

The cycle stealing mode is used in systems in which the CPU should not be disabled for the length of time needed for burst transfer modes. In the cycle stealing mode, the DMA controller obtains access to the system bus the same way as in burst mode, using **BR (Bus Request) and BG (Bus Grant) signals**, which are the two signals controlling the interface between the CPU and the DMA controller. However, in cycle stealing mode, after one byte of data transfer, the control of the system bus is deserted to the CPU via BG. It is then continually requested again via BR, transferring one byte of data per request, until the entire block of data has been transferred. By continually obtaining and releasing the control of the system bus, the DMA controller essentially interleaves instruction and data transfers. The CPU processes an instruction, then the DMA controller transfers one data value, and so on. On the one hand, the data block is not transferred as quickly in cycle stealing mode as in burst mode, but on the other hand the CPU is not idled for as long as in burst mode. Cycle stealing mode is useful for controllers that monitor data in real time.

Transparent mode

The transparent mode takes the most time to transfer a block of data, yet it is also the most efficient mode in terms of overall system performance. The DMA controller only transfers data when the CPU is performing operations that do not use the system buses. It is the primary advantage of the transparent mode that the CPU never stops executing its programs and the DMA transfer is free in terms of time. The disadvantage of the transparent mode is that the hardware needs to determine when the CPU is not using the system buses, which can be complex.

Digital Calibration

The MAX1200, MAX1201, and the MAX1205 have ADCs with sampling ratesof 16-bit, 14-bit, and 2-bit Msps, respectively. The MAX family employs digitalcalibration to ensure its excellent accuracy and dynamic performance. TheMAX1200 family is a CMOS pipelined ADC with four 4-bit stages (with 1-bitoverlap) and a 5-bit flash ADC at the end, giving a total of 3+3+3+3+5 = 17. The extra 1–3 bits are required by the digitalcalibration to quantize the error terms to greater accuracy than the ADCitself and are discarded to give either 14 bits or 16 bits overall.Calibration starts from the multiplying digital-to-analog converter(MDAC) in the third stage; beyond the third stage the MDAC error terms are small enough that calibration is not needed. The third-stage output is digitizedby the remaining pipelined ADC, and the error terms are stored in on-chipRAM. Once the third MDAC is calibrated, it can be used to calibrate the secondMDAC in a similar fashion. Likewise, once the second and third MDACare calibrated, they are used to calibrate the first MDAC. Averaging is used(especially in the first and second MDAC) to ensure that the calibration isnoise-free. During normal conversions, those error terms are recalled from theRAM and are used to adjust the outputs from the digital error correction logic.

Resolution

Resolution is important. The number of bits used to represent an analog signal determinesthe resolution of the ADC. The resolution on a DAQ device is similar to the marks on a ruler. Themore marks a ruler has, the more precise the measurements are. The higher the resolution is on aDAQ device, the higher the number of divisions into which a system can break down the ADCrange, and therefore, the smaller the detectable change.A 3-bit ADC divides the range into 23 or eight divisions. A binary or digital code between000 and 111 represents each division. The ADC translates each measurement of the analog signalto one of the digital divisions. Figure 11.19 shows a 5-kHz sine wave digital image obtained by a3-bit ADC. The digital signal does not represent the original signal adequately because the converterhas too few digital divisions to represent the varying voltages of the analog signal. However,increasing the resolution to 16 bits to increase the ADC number of divisions from eight (23) to65,536 (2^{16}) allows the 16-bit ADCto obtain an extremely accurate representation of the analogsignal.

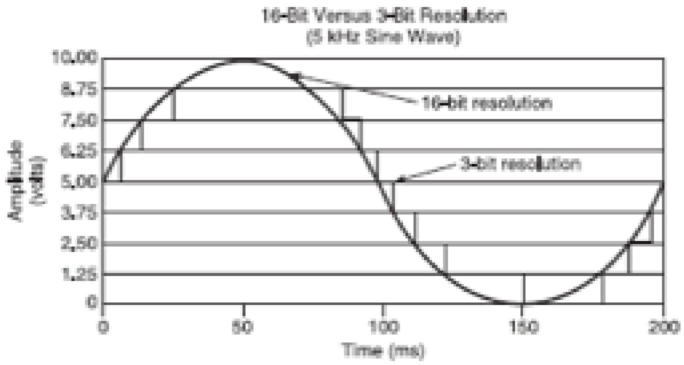


Fig : 3 bit and 16 bit resolution

Range refers to the minimum and maximum analog signal levels that the ADC can digitize.Many DAQ devices feature selectable ranges (typically 0 to 10 V or –10 to 10 V), so you can matchthe ADC range to that of the signal to take best advantage of the available resolution to accuratelymeasure the signal. For example, in Figure 11.20, the 3-bit ADC in chart 1 has eight digital divisions in the range from 0 to 10 V, which is a unipolar range. If you select a range of –10 to 10 V, whichis a bipolar range, as shown in chart 2, the same ADC separates a 20 V range into eight divisions.The smallest detectable voltage increases from 1.25 to 2.50 V, and the right chart is a much lessaccurate representation of the signal.

Amplification or attenuation of a signal can occur before the signal is digitized to improvethe representation of the signal. By amplifying or attenuating a signal, you can effectively decreasethe input range of an ADC and thus allow the ADC to use as many of the available digital divisionsas possible to represent the signal. For example, the below fig shows the effects of applyingamplification to a signal that fluctuates between 0 and 5 V using a 3-bit ADC and a range of 0 to 10
V. With no amplification, or gain = 1, the ADC uses only four of the eight divisions in the conversion.By amplifying the signal by two before digitizing, the ADC uses all eight digital divisions, and thedigital representation is much more accurate. Effectively, the device has an allowable input rangeof 0 to 5 V because any signal above 5 V when amplified by a factor of two makes the input to the
ADC greater than 10 V. The range, resolution, and amplification available on a DAQ devicedetermine the smallest detectable change in the input voltage. This change in voltage representsone least significant bit (LSB) of the digital value and is also called the code width.

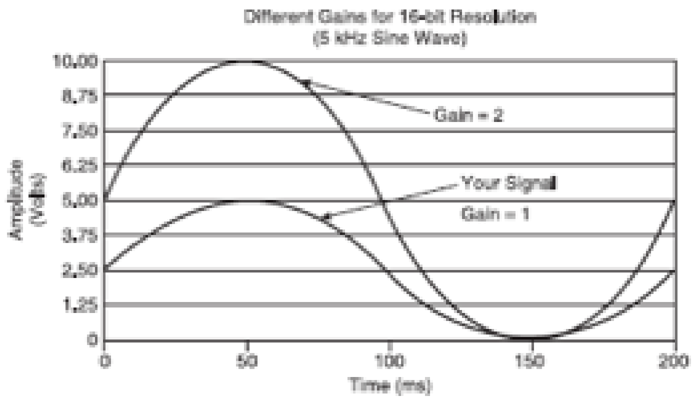


Fig : Effects of applying amplification in resolutionusing the following formula

$$C = D\left(\frac{1}{2^R}\right)$$

where*C* is code width,
D is device input range,
and*R* is bits of resolution.
Device input range is a combination of the gain applied to the signal and the input range ofthe ADC. For example, if the ADC input range is –10 to +10 V peak to peak and the gain is 2, thedevice input range is –5 to +5 V peak to peak, or 20 V. The smaller the code width, the moreaccurately a device can represent the signal. The formula confirms what you have already learnedin the discussion on resolution, range and gain:

- Larger resolution = smaller code width = more accurate representation of the signal
- Larger amplification = smaller code width = more accurate representation of the signal
- Larger range = larger code width = less accurate representation of the signal

Hardware installation of DAQ

Windows Configuration Manager

The Windows Configuration Manager keeps track of all the hardware installedin the computer, including National Instruments DAQ devices. If a Plug &Play (PnP) device, such as an E Series MIO device is available, the WindowsConfiguration Manager automatically detects and configures the device. Ifthe device is a non-PnP device, or legacy device, the user must configure thedevice manually using the Add New Hardware option in the Control Panel. LabVIEW installs Measurement & Automation Explorer (MAX), whichestablishes all device and channel configuration parameters. After installinga DAQ device in the computer, the user must run this configuration utility.MAX reads the information the Device Manager records in the WindowsRegistry and assigns a logical device number to each DAQ device.

Channel and Task Configuration

Using Traditional NI-DAQ a set of *virtual channels* can be configured.A collection of property settings that include a physical channel, the typeof measurement or generation specified in the channel name, and scalinginformation are also configured using this utility. In Traditional NI-

DAQ andearlier versions, virtual channels are a simple method to remember which channelsare used for different measurements. NI-DAQmx*channels* are similar tothe virtual channels of Traditional NI-DAQ. NI-DAQmx also includes *tasks*that are integral to the API. A *task* is a collection of one or more channelsand the timing, triggering, and other properties that apply to the task itself.

Hardware Triggering
There are two ways to begin a DAQ operation:
1. Through a software trigger.
2. Through a hardware trigger.
With software triggering, the DAQ operation begins when the softwarefunction that initiates the acquisition executes. For example, the “Easy” DACVIs use software triggering.

Analog Input
Analog input is used to perform A/D conversions.

Digital Output
Measuring and generating digital values is used in a variety of applications,including controlling relays and monitoring alarm states.

DAQ Software
DAQ software is specialized software that has been developed specifically foruse with data acquisition boards, cards and systems. This software can rangefrom simple software used for one application such as data logging, or it can beextremely robust, offering a wide range of uses for measurement and controland laboratory applications. DAQ software in conjunction with a personalcomputer is used to control the data acquisition board’s functions.

High-Level Functions
More elaborate functions provided by DAQ software includes acquiring, convertingto engineering units, and displaying data graphically on a PC monitor;analyzing a complex analog input signal, providing linearization and coldjunctioncompensation calculations for various types of thermocouple signals;or simultaneously sampling multiple channels of analog input data, comparingthem, and plotting them in a three-dimensional presentation.

High-Level Mathematical Functions
FFT and frequency analysis, signal generation, mathematics, curve fitting andinterpolation, along with time and frequency domain analysis, allow users toderive meaningful information from data.

Application Development
Whether the user is taking temperature measurements with an A/D board,analyzing signals using a stand-alone oscilloscope board, or measuring pressurewith a sophisticated signal conditioner; using specialized application developmentDAQ software with the appropriate driver is ideal for

the application.From data acquisition to instrument control, and image acquisition to motioncontrol, application development software provides the tools to rapidlydevelop the acquisition system.

UNIT IV
INSTRUMENT INTERFACES

4–20mA Current Loop

Current-mode data transmission is the preferred technique in many environments, particularly in industrial applications. Most systems employ the familiar 2-wire, 4–20mA current loop, in which a single twisted-pair cable supplies power to the module and carries the output signal as well.
The 3-wire interface is less common but allows the delivery of more power to the module electronics. A 2-wire system provides only 4mA at the line voltage (the remaining 16mA carries the signal). Current loops offer several advantages over voltage-mode output transducers:
– They do not require a precise or stable supply voltage.
– Their insensitivity to IR drops makes them suitable for long distances.
– A 2-wire twisted-pair cable offers very good noise immunity.
– The 4mA of current required for information transfer serves two purposes:
it can furnish power to a remote module, and it provides a distinctionbetween zero (4 mA) and no information (no current flow).In a 2-wire, 4–20mA current loop, supply current for the sensor electronicsmust not exceed the maximum available, which is 4mA (the remaining 16mAcarries the signal). Because a 3-wire current loop is easily derived from the2-wire version, the following discussion focuses on the 2-wire version.

Need for a Current Loop

The 4–20mA current loop shown in Fig. 1 is a common method of transmittingsensor information in many industrial process-monitoring applications. A sensor is a device used to measure physical parameters such as temperature, pressure, speed, liquid flow rates, etc. Transmitting sensory information through a current loop is particularly useful when theinformationhas to be sent to a remote location over long distances (1,000 ft, ormore). The loop's operation is straightforward: a sensor's output voltage is first converted to a proportional current, with 4mA normally representing the sensor's zero-level output, and 20mA representing the sensor's full-scale output. Then, a receiver at the remote end converts the 4–20mA current back into a voltage which in turn can be further processed by a computer or display module. However, transmitting a sensor's output as a voltage over long
distances has several drawbacks. Unless very high input-impedance devices are used, transmitting voltages over long distances produces correspondingly lower voltages at the receiving end due wiring and interconnect resistances.However, high-impedance instruments can be sensitive to noise pickup since the lengthy signal-carrying wires often run in close proximity to other electrically noisy system wiring. Shielded wires can be used to minimize noise pickup, but their high cost may be prohibitive when long distances areinvolved. Sending a current over long distances produces voltage losses proportionalto the wiring's length. However, thesevoltage

losses also known as loop drops” do not reduce the 4–20mA current as long as the transmitter anloop supply can compensate for these drops. The magnitude of the current inthe loop is not affected by voltage drops in the system wiring since all of thecurrent (i.e., electrons) originating at the negative (−) terminal of the loop power supply has to return back to its positive (+) terminal

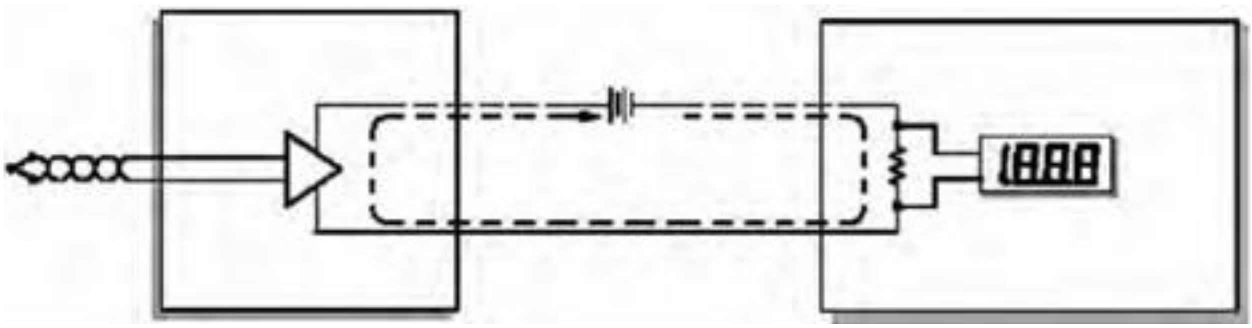


Fig1: Typical components in a loop

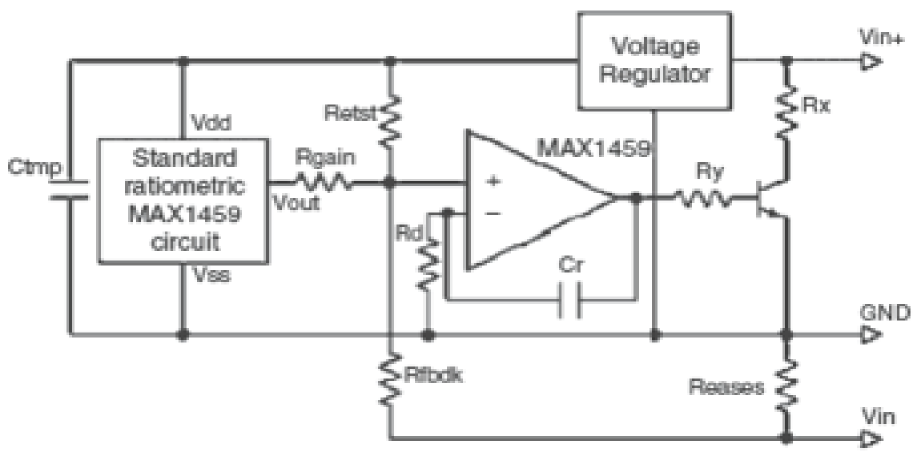


Fig2: 4-20 mA transmitter
Basic 2-wire Circuit
A voltage output should be converted to current when configuring a ratio metric 4–20mA circuit, because current-mode applications require a 4mA offset and 16mA span. This section presents the circuit details and results obtained from a current-loop configuration based on the MAX1459 sensorsignal conditioner. In principle, a voltage regulator is to be added, which converts the 10–32V loop voltage to a fixed 5V for operating the MAX1459. Figure shows the circuitry required for implementing a standard ratio metric version of the MAX1459 circuit. The voltage regulator can be any low-cost device whose quiescent current is sufficiently low for the 4mA budget.

Advantages of 4–20mA Current Loop
Current loops offer four major advantages such as:

– Long-distance transmission without amplitude loss
– Detection of offline sensors, broken transmission lines, and other failures
– Inexpensive 2-wire cables
– Lower EMI sensitivity

RS 232C/RS 485

The RS-232/485 port sequentially sends and receives bytes of information onebit at a time. Although this method is slower than parallel communication,which allows the transmission of an entire byte at once, it is simpler and canbe used over longer distances because of lower power consumption.For example, the IEEE 488 standard for parallel communication states thatthe cable length between the equipments should not exceed 20m total, witha maximum distance of 2m between any two devices. RS-232/485 cabling,however, can extend 1,200m or greater. Typically, RS-232/485 is used totransmit American Standard Code for Information Interchange (ASCII) data.Although National Instruments serial hardware can transmit 7-bit as wellas 8-bit data, most applications use only 7-bit data. Seven-bit ASCII canrepresent the English alphabet, decimal numbers, and common punctuationmarks. It is a standard protocol that virtually all hardware and software understand.Serial communication is mainly using the three transmission lines:
(1) ground, (2) transmit, and (3) receive. Because RS-232/485 communicationis asynchronous, the serial port can transmit data on one line while receivingdata on another. Other lines such as the handshaking lines are not required.The important serial characteristics are baud rate, data bits, stop bits, andparity. To communicate between a serial instrument and a serial port on computer,these parameters must match.The RS-232 port, or ANSI/EIA-232 port, is the serial connection foundon IBM-compatible PCs. It is used for many purposes, such as connecting amouse, printer, or modem, as well as industrial instrumentation. The RS-232protocol can have only one device connected to each port. The RS-485 (EIA-485 Standard) protocol can have 32 devices connected to each port. Withthis enhanced multidrop capability the user can create networks of devicesconnected to a single RS-485 serial port. Noise immunity and multi dropcapability make RS-485, the choice in industrial applications requiring manydistributed devices networked to a PC or other controller for data collection.USB was designed primarily to connect peripheral devices to PCs, includingkeyboards, scanners, and disk drives.RS-232 (Recommended standard-232) is a standard interface approved bythe Electronic Industries Association (EIA) for connecting serial devices. Inother words, RS-232 is a long established standard that describes the physicalinterface and protocol for relatively low-speed serial data

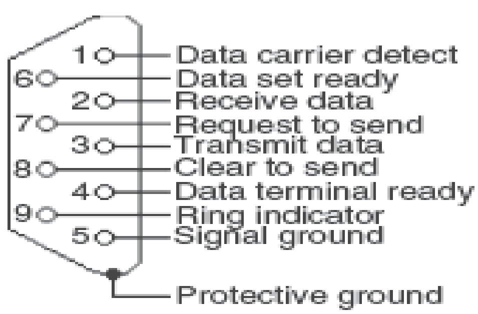


Fig 3: Pin diagram

Table: pin description

Pin Number	Signal	Description
1	DCD	Data carrier detect
2	RxD	Receive data
3	TxD	Transmit data
4	DTR	Data terminal ready
5	GND	Signal ground
6	DSR	Data set ready
7	RTS	Ready to send
8	CTS	Clear to send
9	RI	Ring indicator

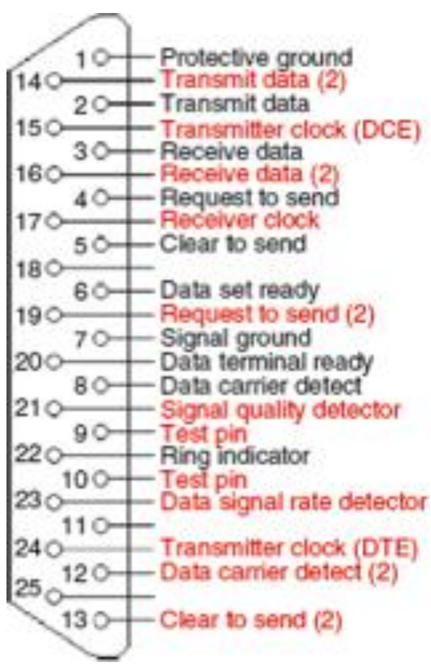


Fig 4: RS232c DB 25 pinout

Signal Descriptions

– TxD – This pin carries data from the computer to the serial device

– RXD – This pin carries data from the serial device to the computer

– DTR signals – DTR is used by the computer to signal that it is readyto communicate with the serial device like modem. In other words, DTRindicates the modem that the DTE (computer) is ON.
– DSR – Similarly to DTR, Data set ready (DSR) is an indication from themodem that it is ON.
– DCD – Data carrier detect (DCD) indicates that carrier for the transmitdata is ON.
– RTS – This pin is used to request clearance to send data to a modem.
– CTS – This pin is used by the serial device to acknowledge the computersRTS signal. In most situations, RTS and CTS are constantly onthroughout the communication session.
– Clock signals (TC, RC, and XTC) – The clock signals are only used forsynchronous communications. The modem or DSU extracts the clock fromthe data stream and provides a steady clock signal to the DTE. Thetransmit and receive clock signals need not have to be the same, or evenat the same baud rate.
– *CD – CD stands for Carrier detect*. Carrier detect is used by a modemto signal that it has a made a connection with another modem, or hasdetected a carrier tone. In other words, this is used by the modem tosignal that a carrier signal has been received from a remote modem.
RI – RI stands for ring indicator. A modem toggles (keystroke) the stateof this line when an incoming call rings in the phone. In other words,this is used by an autoanswer modem to signal the receipt of a telephonering signal. The carrier detect (CD) and the ring indicator (RI) lines areonly available in connections to a modem. Because most modems transmitstatus information to a PC when either a carrier signal is detected

RS485 Serial Communication

RS485 is an EIA standard for multipoint communications. It supports severaltypes of connectors, including DB-9 and DB-37. RS-485 is similar to RS-422 but can support more nodes per line. RS485 meets the requirements for atruly multipoint communications network, and the standard specifies up to32 drivers and 32 receivers on a single (2-wire) bus. With the introductionof "automatic" repeaters and high-impedance drivers/receivers this "limitation"can be extended to hundreds or even thousands of nodes on a network.The RS-485 and RS-422 standards have much in common, and are often confusedfor that reason. RS-485, which specifies bi-directional, half-duplex datatransmission, is the only EIA standard that allows multiple receivers anddrivers in "bus" configurations. RS-422, on the other hand, specifies a single,unidirectional driver with multiple receivers.

GPIB

The purpose of this section is to provide guidance and understanding of theGeneral Purpose Interface Bus (GPIB) bus to new GPIB bus users or toprovide more information on using the GPIB bus's features.GPIB Data Acquisition and Control Module provides analog and digitalsignals for controlling virtually any kind of a device and the capability toread back analog voltages, digital signals, and temperatures. The 4867 DataAcquisition and Control module is an IEEE-488.2 compatible device and has aStandard Commands for Programmable Instruments (SCPI) command parserthat accepts SCPI and short form commands for ease of programming. Applicationsinclude device control, GPIB interfacing and data logging. The 4867is housed in a small 7 in.×7 in.

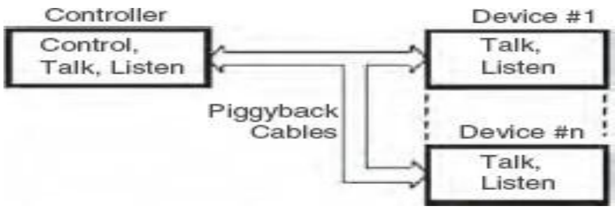


Fig 5: Block Diagram

Controllershave the ability to send commands, to talk data onto the bus and to listento data from devices. Devices can have talk and listen capability. Control canbe passed from the active controller (Controller-in-charge) to any device withcontroller capability. One controller in the system is defined as the SystemController and it is the initial controller-in-charge (CIC).Devices are normally addressable and have a way to set their address. Eachdevice has a primary address between 0 and 30. Address 31 is the unlisten oruntalk address.

Types of GPIB Messages

GPIB devices communicate with other GPIB devices by sending devicedependentmessages and interface messages through the interface system.**Device-dependent messages**, often called data or data messages, containdevice-specific information, such as programming instructions, measurementresults, machine status, and data files.**Interface messages** manage the bus. Usually called commands or commandmessages, interface messages perform such functions as initializing thebus, addressing and unaddressing devices, and setting device modes for remoteor local programming.The term “command” as used here should not be confused with somedevice instructions that are also called commands. Such device-specific commandsare actually data messages as far as the GPIB interface system itselfis concerned.

Physical Bus Structure

GPIB is a 24 conductor as shown in Fig 6Physically, the GPIB interfacesystem consists of 16 low-true signal lines and eight ground-return or shielddrainlines. The 16 signal lines, discussed later, are grouped into data lines(eight), handshake lines (three), and interface management lines (five).

Data Lines

The eight data lines, DIO1 through DIO8, carry both data and commandmessages. The state of the Attention (ATN) line determines whether theinformation is data or commands. All commands and most data use the 7-bitASCII or ISO code set, in which case the eighth bit, DIO8, is either unusedor used for parity.

Handshake Lines

Three lines asynchronously control the transfer of message bytes betweendevices. The process is called a 3-wire interlocked handshake. It guaranteesthat message bytes on the data lines are sent and received without transmissionerror.

NRFD (not ready for data) – Indicates when a device is ready or not readyto receive a message byte. The line is driven by all devices when receivingcommands by listeners when receiving data messages, and by the talkerwhen enabling the HS488 protocol.

– NDAC (not data accepted) – Indicates when a device has or has notaccepted a message byte. The line is driven by all devices when receivingcommands, and by Listeners when receiving data messages.

– DAV (data valid) – Tells when the signals on the data lines are stable(valid) and can be accepted safely by devices. The Controller drives DAVwhen sending commands, and the Talker drives DAV when sending datamessages.Three of the lines are handshake lines, NRFD, NDAC, and DAV, whichtransfer data from the talker to all devices who are addressed to listen. The talker drives the DAV line; the listeners drive the NDAC and NRFD lines.The remaining five lines are used to control the bus's operation.

– ATN (attention) is set true by the controller-in-charge while it is sendinginterface messages or device addresses. ATN is false when the bus istransmitting data.

– EOI (end or identify) can be asserted to mark the last character of amessage or asserted with the ATN signal to conduct a parallel poll.

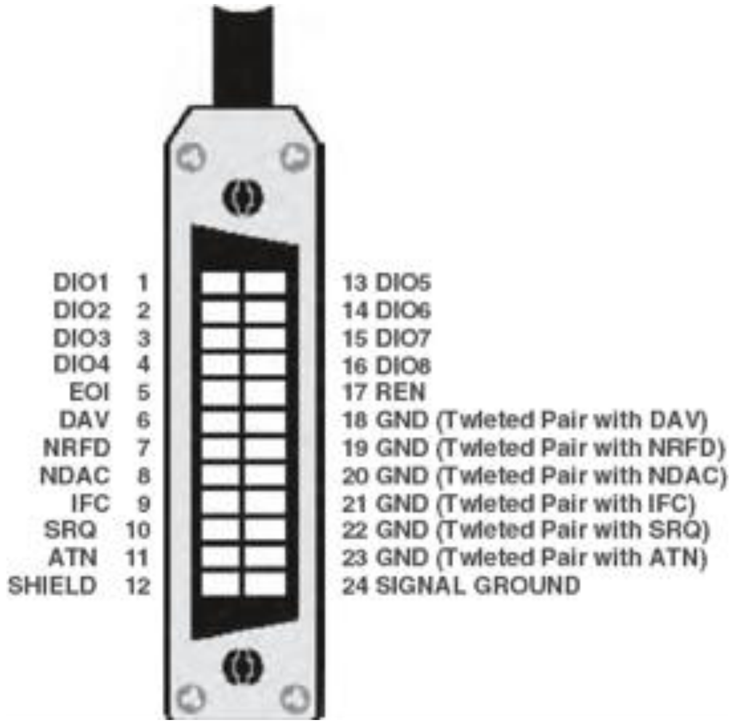


Fig :6 Bus structure of GPIB

USB

The USB is a medium-speed serial data bus designed to carry relatively largeamounts of data over relatively short cables: up to about 5m long. It cansupport data rates of up to 12Mbs−1 (megabits per second), which is fastenough for most PC peripherals such as scanners, printers, keyboards, mice,joysticks, graphics tablets, low-res digital cameras, modems, digital speakers,low speed CD-ROM and CD-writer drives, external Zip disk drives, and soon. The USB is an addressable bus system, with a 7-bit address code so it cansupport up to 127 different devices or nodes. However, it can have only onehost. So a PC with its peripherals connected through the USB forms a starLocal Area Network (LAN).On the other hand any device connected to the USB can have a numberof other nodes connected to it in daisy-chain fashion, so it can also form the

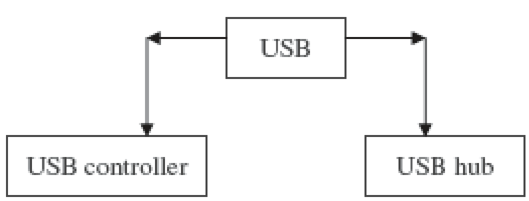


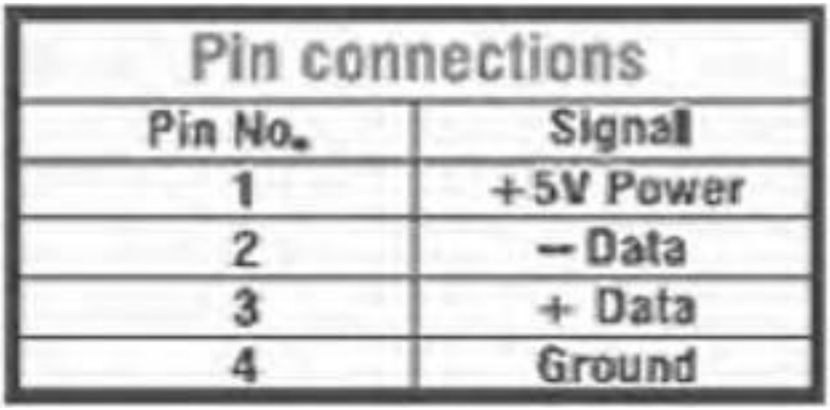
Fig 7: USB Structure

Most hubs provide either four or seven downstream ports, or less. Anotherimportant feature of the USB is that it is designed to allow hot swapping.Devices can be plugged into and unplugged from the bus without having toturn the power off and on again, re-boot the PC or even, manually starta driver program. A new device can simply be connected to the USB, andthe PCs operating system should recognize it and automatically set up thenecessary driver to service it.

Need for USB

The USB is host controlled. There can only be one host per bus. The specificationin itself does not support any form of multi master arrangement. Thisis aimed at and limited to single point to point connections such as a mobilephone and personal organizer and not multiple hub, multiple device desktopconfigurations. The USB host is responsible for undertaking all transactionsand scheduling bandwidth. Data can be sent by various transaction methodsusing a token-based protocol. One of the original intentions of USB was to reduce the amount of cabling at the back of PC. The idea came from theApple Desktop Bus, where the keyboard, mouse and some other peripheralscould be connected together (daisy chained) using the one cable.However, USB uses a tiered star topology, similar to that of 10BaseTEthernet. This imposes the use of a hub somewhere, which adds to greaterexpense, more boxes on desktop and more cables. However it is not as bad as itmay seem. Many devices have USB hubs integrated into them. For example,keyboard may contain a hub which is connected to computer. Mouse andother devices such as digital camera can be plugged easily into the back ofkeyboard. Monitors are just another peripheral on a long list which commonlyhas in-built hubs.This tiered star topology, rather than simply daisy chaining devicestogether has some benefits. First power to each device can be monitored andeven switched-off if an over current condition occurs without disrupting otherUSB devices. High, full and low speed devices can be supported, with the hubfiltering out high speed and full speed transactions so lower speed devices donot receive them. Up to 127 devices can be connected to any one USB bus atany one given time. To extent devices simply add another port/ host. Whileearlier USB hosts had two ports, most manufacturers have seen this as limitingand are starting to introduce 4 and 5 port host cards with an internal portfor hard disks etc. The early hosts had one USB controller and thus both portsshared the same available USB bandwidth. As bandwidth requirements haveincreased, multiport cards with two or more controllers allowing individualchannels are used.

Table: Pin connections



Pin connections	
Pin No.	Signal
1	+5V Power
2	– Data
3	+ Data
4	Ground

USB cables use two different types of connectors:
– *Type A*. Plugs for the upstream end.
– *Type B*. Plugs for the downstream end.

PCMCIA

Personal Computer Memory Card International Association (PCMCIA) isan international standards body and trade association founded in 1989developed a standard for small, credit card-sized devices, called PC Cards.Originally designed for adding memory to portable computers, the PCMCIA standard has been expanded several times and is now suitable for many typesof devices. The inclusion of PCMCIA technology in PCs delivers a varietyof benefits. Besides providing an industry standard interface for third-partycards (PC Cards), PCMCIA allows users to easily swap cards in and out of aPC as needed, without having to deal with the allocation of system resourcesfor those devices. These useful features hot swapping and automatic configuration,as well as card slot power management and other PCMCIA capabilities –are supported by a variety of software components on the PCMCIA-based PC.In most cases, the software aspect of PCMCIA remains relatively transparentto the user. As the demand for notebook and laptop computers began skyrocketingin the late 1980s, users realized that their expansion options werefairly limited. Mobile machines were not designed to accept the wide array ofavailable expansion cards that their desktop counterparts could enjoy

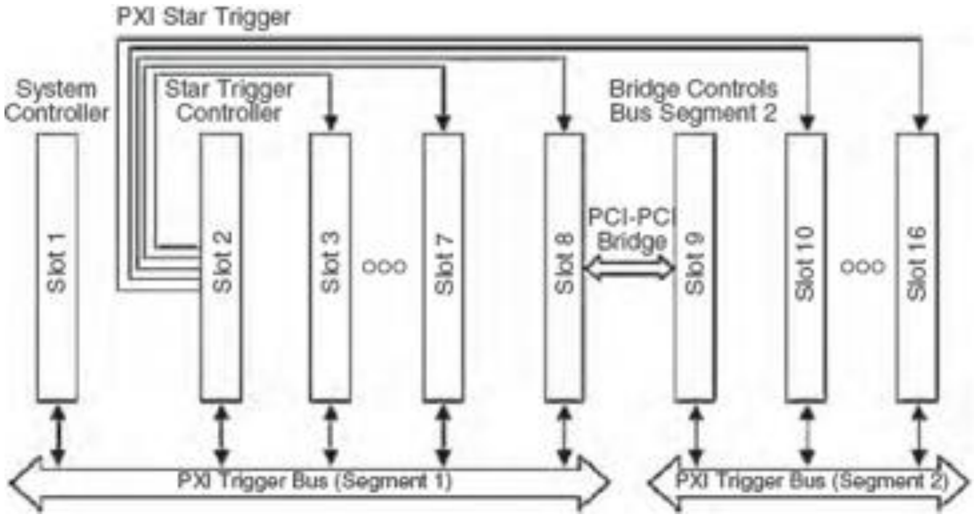


Fig 8: Triggering Architecture

Features of PCMCIA

– One rear slot, access from rear of PC
– Accept Type I/II/III Cards
– Comply with PCI Local Bus Specification Rev.2.2
– Comply with 1995 PC Card Standard
– Extra compatible registers are mapped in memory
– Use T/I 1410 PCMCIA controller
– Support PC Card with Hot Insertion and Removal
– Support 5V or 5/3.3V 16-bit PC Cards
– Support Burst Transfers to Maximize Data Throughput on both PCIBuses
– Supports Distributed DMA and PC/PCI DMA

Utilities of PCMCIA Card in the Networking Category

Under the networking category, typically PCMCIA slot supports 4 types of cards such as:

– LAN card
– Wireless LAN card
– Modem card
– ATA flash disk card

VISA

Virtual Instrumentation Software Architecture (VISA) is a standard I/O languagefor instrumentation programming. VISA by itself does not provideinstrumentation programming capability. VISA is a high-level API that callsinto lower level drivers. VISA is capable of controlling VXI, GPIB, or Serial instruments and makes the appropriate driver calls depending on the type of instrument being used. Whendebugging VISA problems, it is important to keep in mind that this hierarchyexists.

Fig9: VISA Architecture

The terminology related with VISA are explained as follows:
Resources. The most important objects in the VISA language are known asresources.*Operations*. In object-oriented terminology, the functions that can be usedwith an object are known as operations *Attributes*. In addition to the operations that can be used with an object, theobject has variables associated with it that contain information related tothe object. In VISA, these variables are known as attributes. There is a default resource manager at the top of the VISA hierarchythat can search for available resources or open sessions to them. Resourcescan be GPIB, serial message based VXI, or register-based VXI. The mostcommon operations for message based instruments are read and write.

Waveform generator

Stand-alone traditional instruments such as oscilloscopes and waveform generatorsare very powerful, expensive, and designed to perform one or morespecific tasks defined by the vendor. However, the user generally cannot extendor customize them. The knobs and buttons on the instrument, the built-incircuitry, and the functions available to the user, are all specific to the natureof the instrument. In addition, special technology and costly components mustbe developed to build these instruments, making them very expensive and slowto adapt.

UNIT V
APPLICATION OF VI

Fourier Transforms

LabVIEW and its analysis VI library provide a complete set of tools to perform Fourier and spectral analysis. The Fast Fourier Transform (FFT) and Power Spectrum VIs are optimized, and their outputs adhere to the standard DSP format.

FFT is a powerful signal analysis tool, applicable to a wide variety of fields including spectral analysis, digital filtering, applied mechanics, acoustics, medical imaging, modal analysis, numerical analysis, seismography, instrumentation, and communications.

The LabVIEW analysis VIs, located on the **Signal Processing** palette, maximize analysis throughput in FFT-related applications. This document discusses FFT properties, how to interpret and display FFT results, and how to manipulate FFT and power spectrum results to extract useful frequency information.

FFT Properties

The fast Fourier transform maps time-domain functions into frequency-domain representations. FFT is derived from the Fourier transform equation, which is

$$X(f) = \boldsymbol{F}\{x(t)\} = \int_{-\infty}^{\infty} x(t)\, e^{-j2\pi ft}\, dt \ ,$$

where x(t) is the time domain signal, X(f) is the FFT, and ft is the frequency to analyze.Similarly, the discrete Fourier transform (DFT) maps discrete-time sequences into discrete-frequency representations. DFT is given by the following equation

$$X_k = \sum_{i=0}^{n-1} x_i\, e^{-j2\pi ik/n} \qquad \text{for } k = 0, 1, 2, \dots, n-1$$

where x is the input sequence, X is the DFT, and n is the number of samples in both the discrete-time and the discrete-frequency domains. Direct implementation of the DFT, as shown in equation 2, requires approximately n complex operations. However, computationally efficient algorithms can require as little as n log2(n) operations.

$$\boldsymbol{F}\{x\} = X = X_{Re} + j\, X_{Im} = \mathrm{Re}\{X\} + j\, \mathrm{Im}\{X\}$$

An inherent DFT property is the following:

$$X_{n-i} = X_{-i}$$

FFT Standard Output:

The output format of the FFT VI can now be described with Xn-i. If the total number of samples, n, of the input sequence, x, is even, the format of the complex output sequence, X, and the interpretation of the results are shown in the below table.

Table 1: Output Format for Even n, k = n ÷ 2

Array Element	**Interpretation**
X[0]	DC component
X[1]	1st harmonic or fundamental
X[2]	2nd harmonic
.	.
.	.
.	.
X[k-2]	(k-2)th harmonic
X[k-1]	(k-1)th harmonic
X[k] = X[-k]	Nyquist harmonic
X[k+1]= X[n-(k-1)] = X[-(k-1)]	(k-1)th harmonic
.	
.	
.	
X[n-3]	-3rd harmonic
X[n-2]	-2nd harmonic
X[n-1]	-1st harmonic

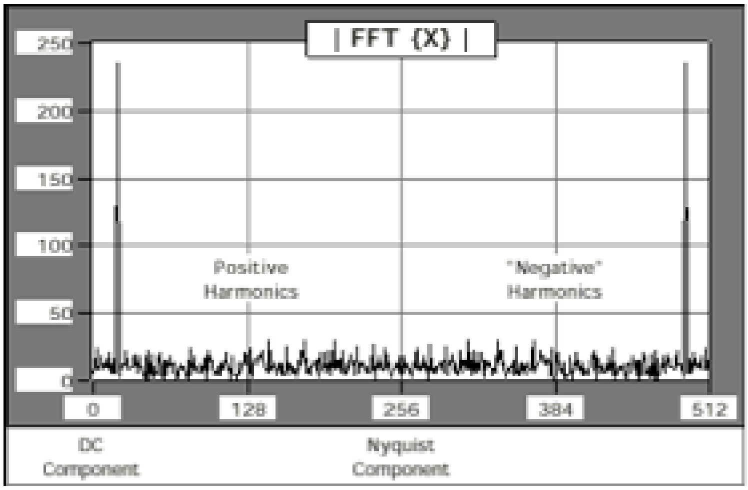


Fig 1 : Graph of table

If the total number of samples, n, of the input sequence, x, is odd, the format of the complex output sequence, X, and the interpretation of the results are shown in the below table

Table:2Output Format for Odd n, k = (n-1) ÷ 2

Array Element	**Interpretation**
X[0]	DC component
X[1]	1st harmonic or fundamental
X[2]	2nd harmonic
.	.
.	.
.	.
X[k-1]	kth -1harmonic
X[k]	kth harmonic
X[k+1]= X[n-k] = X[-k] kth	kth harmonic
. X[k+2]= X[n-(k-1)] = X[-(k-1)]	- (k-1)th harmonic
.	
.	
X[n-3]	-3rd harmonic
X[n-2]	-2nd harmonic

X[n-1]	-1st harmonic

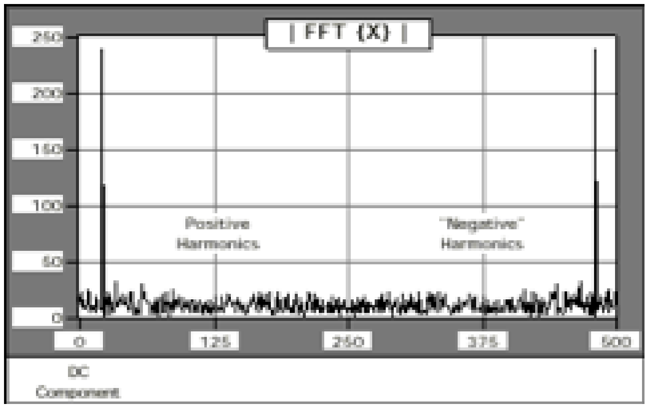


Fig 2 : Graph of table 2

Standard Output

Standard output is the format for even and odd-sized discrete-time sequences, described in Tables 1 and 2 of this document. This format is convenient because it does not require any further data manipulation.To graphically display the results of the FFT, wire the output arrays to the waveform graph Fig : 2 Graph of table2. The FFT output is complex and requires two graphs to display all the information.

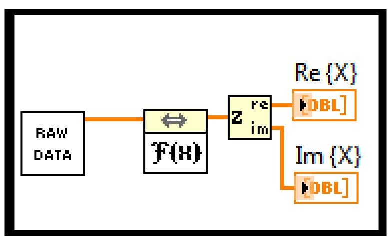


Fig 3 : Standard output by using VI

Power spectrum and Correlation:

The power spectrum reveals the existence, or the absence, of repetitive patterns and correlation structures in a signal process. These structural patterns are important in a wide range of applications such as data forecasting, signal coding, signal detection, radar, pattern ecognition, and decision-making systems. The most common method of spectral estimation is based on the fast Fourier transform (FFT).

In particular, the total power is given by

$$P_x(\omega) = \sum_{k=-\infty}^{\infty} r_x[k]\, e^{-jk\omega}$$

One can show that Px(w) is real, even and positive. The auto-correlation can be recovered with the inverse Fourier transform

$$r_x[k] = \frac{1}{2\pi} \int_{-\pi}^{\pi} d\omega \, P_x(\omega) \, e^{jk\omega}$$

Power spectrum – properties:

In particular, the total power is given by

$$r_x[0] = \frac{1}{N} \sum_{n=1}^{N} |x[n]|^2 = \frac{1}{2\pi} \int_{-\pi}^{\pi} d\omega \, P_x(\omega)$$

The power spectrum is sometimes called **spectral density** because it is *positive* and the signal power can always be*normalized* to $r(0) = (2p)^{-1}$.

Example: Uncorrelated noise has a constant power spectrum

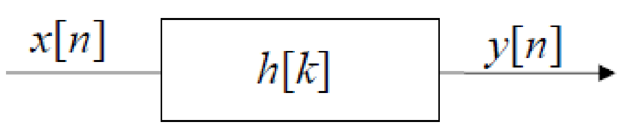
$$r[k] = \sigma^2 \delta(k)$$

$$P_x(\omega) = \sum_{k=-\infty}^{\infty} \sigma^2 \delta(k) e^{-jk\omega} = \sigma^2$$

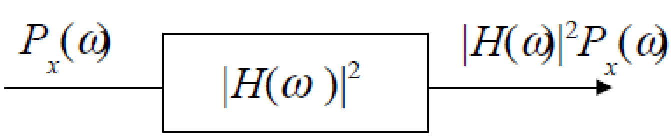
Hence it is also called **white noise**.

Effect of Filtering on Power Spectrum

A linear system with impulse response $h[k]$



Transforms the power spectrum as



Spectral Content

The power spectrum gives the spectral content of the data. To see that consider the power of a signal after filtering with a narrow bandpass filter around w 0.

$$\begin{aligned} E[|y[n]|^2] &= \frac{1}{2\pi}\int_{-\pi}^{\pi} d\omega P_y(\omega) \\ &= \frac{1}{2\pi}\int_{-\pi}^{\pi} d\omega |H(\omega)|^2 P_x(\omega) \\ &= \frac{1}{2\pi}\int_{\omega_0-\Delta\omega/2}^{\omega_0+\Delta\omega/2} d\omega P_x(\omega) \\ &\approx \frac{\Delta\omega}{2\pi} P_x(\omega_0) \end{aligned}$$

Power spectrum - properties

The power spectrum captures the spectral content of the sequence. It can be estimate directly from the Fourier transform of the data.

$$\hat{P}_x(\omega) = \frac{1}{N}|X(\omega)|^2$$

$$X(\omega) = \sum_{k=0}^{N-1} x[k] e^{-j\omega k}$$

Correlation methods

Correlation is a statistical technique that can show whether and how strongly pairs of variables are related.

Correlation Coefficient

The main result of a correlation is called the **correlation coefficient** (or "r"). It ranges from -1.0 to +1.0. The closer r is to +1 or -1, the more closely the two variables are related.

Cross and Auto-correlation

The cross-correlation is defined as

$$r_{yx}(k) = \sum_{n=-\infty}^{\infty} y^*[n]x[n+k]$$

Note that correlation is a convolution with opposite sign. It can be computed with the Fourier transform.

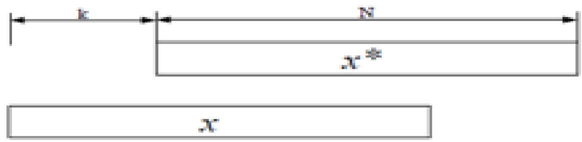
$$R_{xy}(\omega) = Y^*(\omega)X(\omega)$$

The auto-correlation is defined as

$$r_x(k) = \sum_{n=-\infty}^{\infty} x^*[n]x[n+k]$$

For a sample of finite length *N* this is typically normalized. We call this the sample auto-correlation

$$\hat{r}_x[k] = \frac{1}{N-k}\sum_{n=1}^{N-k} x^*[n]x[n+k]$$



Use xcorr() to compute cross of auto-correlation.

Auto-correlation properties

The auto-correlation is symmetric.

$$r_x[k] = r_x^*[-k]$$

The zero lag gives the total power of the signal

The auto-correlation has the power as an upper-bound

$$r_x[0] \geq |r_x[k]| \qquad r_x[0] = \sum_n |x[n]|^2$$

Windowing & flittering

Windowing is the process of taking a small subset of a larger dataset, for processing and analysis. A naive approach, the rectangular window, involves simply truncating the dataset before and after the window, while not modifying the contents of the window at all. However this is a poor method of windowing and causes power leakage.

Windowing of a simple waveform like cos ωt causes its Fourier transform to develop non-zero values (commonly called spectral leakage) at frequencies other than ω. The leakage tends to be worst (highest) near ω and least at frequencies farthest from ω.

If the waveform under analysis comprises two sinusoids of different frequencies, leakage can interfere with the ability to distinguish them spectrally. If their frequencies are dissimilar and one component is weaker, then leakage from the stronger component can obscure the weaker one's presence. But if the frequencies are similar, leakage can render them unresolvable even when the sinusoids are of equal strength. The rectangular window has excellent resolution characteristics for sinusoids of comparable strength, but it is a poor choice for sinusoids of disparate amplitudes. This characteristic is sometimes described as low-dynamic-range.

At the other extreme of dynamic range are the windows with the poorest resolution and sensitivity, which is the ability to reveal relatively weak sinusoids in the presence of additive random noise. That is because the noise produces a stronger response with high-dynamic-range windows than with high-resolution windows. Therefore, high-dynamic-range windows are most often justified in wideband applications, where the spectrum being analyzed is expected to contain many different components of various amplitudes.

In between the extremes are moderate windows, such as Hamming and Hann. They are commonly used in narrowband applications, such as the spectrum of a telephone channel. In summary, spectral analysis involves a tradeoff between resolving comparable strength components with similar frequencies and resolving disparate strength components with dissimilar frequencies. That tradeoff occurs when the window function is chosen.

Application in Process Control projects

Software for Remote ON/OFF Control Experiment

The software for the Remote ON/OFF controller lab experiment is developed using LabVIEW.
There are two distinct parts in the software:

1. ON/OFF controller server program and logic
2. Internet communication using DataSocket protocol

These parts will be explained in the following sections

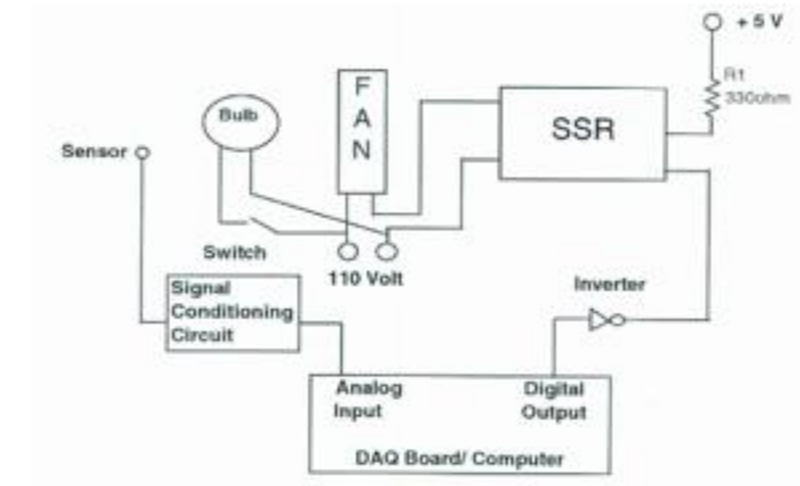


Fig 4: ON-OFF Temperature Control

ON/OFF Controller Server Program and Logic

The LabVIEW program on the server first reads the voltage from the analog input channel of the DAQ board and converts it to a temperature using the equation

$$Tout = 6\ Vin + 60$$

where Tout is the process temperature, and Vin is the input voltage from the DAQ board. The desired Set-point temperature value of the process is obtained from the client computer, and the Error is determined using the equation

$$Error = Set\text{-}point\ .\ Tout.$$

Based on the error and neutral zone High Limit and Low Limit, an ON/OFF controller logic is implemented using LabVIEW. In this logic, Digital Output Channel is the output logic value sent to the DAQ board which controls the fan through the SSR, Cooling Fan Indicators are the LED ON/OFF indicators on the front panel of the LabVIEW VI that show the state of the fan, and Within Limits Indicator shows if the process is operating within the neutral zone. The Cooling Fan ON Indicator is also defined as a local read only variable which is used in the logic implementation.

The LabVIEW front panel of the server program VI

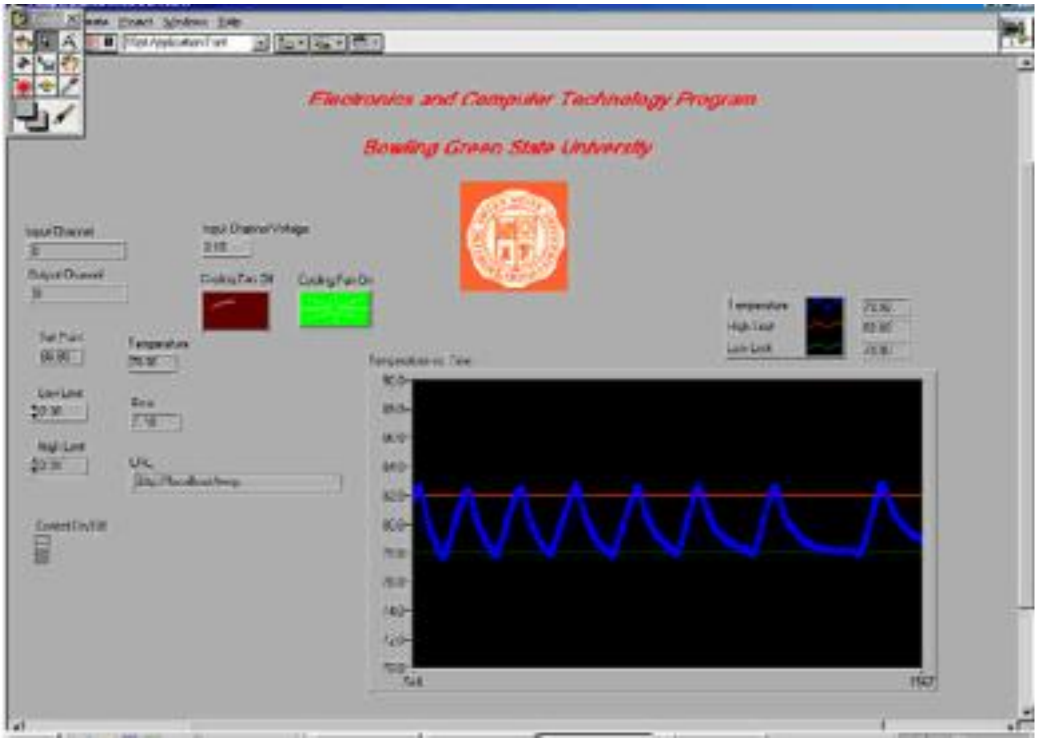


Fig 5: Server front panel

Temperature data acquisition system

The hardware for server workstation consists of a PC with Pentium III, 550 MHz processor, 128 Mb RAM, Network Interface Card (NIC), National Instruments DAQ board. The server software includes Windows 98, NI-DAQ driver, LabVIEW 5.1 and NI-DataSocket Manager. The designed experiment is connected to the DAQ board. The server is assigned static IP address. The clients could be any PC.s with NIC that can run a LabVIEW program. The objective of the experiment is to maintain the temperature inside a wooden box at some desired set-point value, within neutral zone limits, using a two-state-controller mode. The wooden box is heated with a light bulb. The temperature is measured using LM335 solid-state The hardware for server workstation consists of a PC with Pentium III, 550 MHz processor, 128 Mb RAM, Network Interface Card (NIC), National Instruments DAQ board. The server software includes Windows 98, NI-DAQ driver, LabVIEW 5.1 and NI-DataSocket Manager. The designed experiment is connected to the DAQ board. The server is assigned static IP address. The clients could be any PC.s with NIC that can run a LabVIEW program. The objective of the experiment is to maintain the temperature inside a wooden box at some desired set-point value, within neutral zone limits, using a two-state-controller mode. The wooden box is heated with a light bulb. The temperature is measured using LM335 solid-state

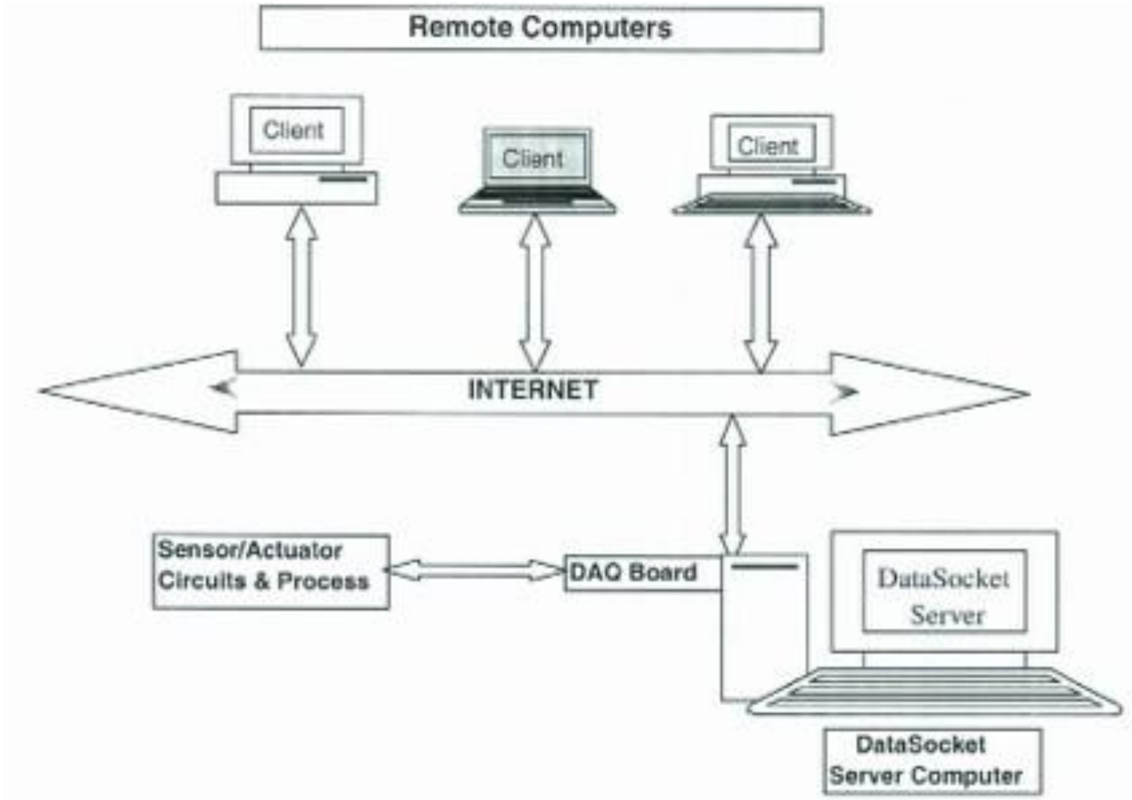


Fig 6: Temperature data Acquisition

Oscilloscope

Open Windows oscilloscopes from the 5000, 6000, 7000, and 8000 series are Windows-based and contain scope-specific software for acquisition, connectivity, and control. By default, the scope user interface runs Windows-based application software that performs the following scope-specific tasks:

• Presents the scope user interface
• Configures scope hardware based on commands from the user and signal content
• Displays acquired signals and derivations of acquired signals

You can use LabVIEW to extend the feature list of your scope to include the following tasks:

• Custom analysis and signal processing of acquired signals
• Automation of scope-related tasks and sequences of tasks
• A unique and customizable user interface
• Custom reports, including reports that you can publish live over the Internet
• Remote, Internet-based control and monitoring

Example:

Connect the function generator to the scope and put up the sample waveform. Increase the amplitude to 2v and pull down the VISA box and click on GPIB for the oscilloscope. The display of the wave is shown in the fig 7.

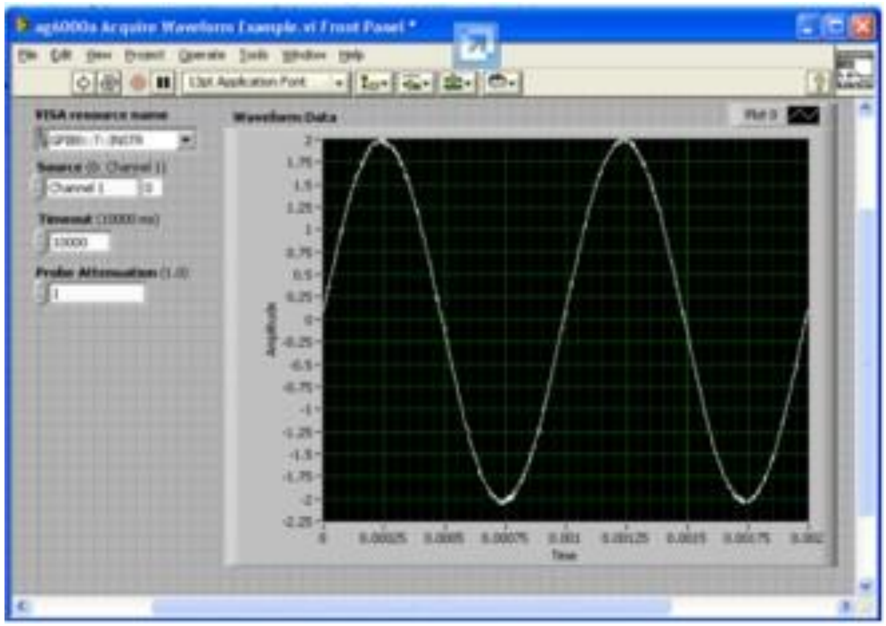


Fig 7: waveform acquired by oscilloscope

Digital Multimeter

Digital Multimeters (DMMs) are specialized in takingflexible, high resolution measurements

• Low to medium acquisition speeds

• High resolution

• Measures current or resistance in addition to voltage.

• Different form factors: PXI, PCIe, PCI, USBSoftware Support in NI LabVIEW and SignalExpressThe NI 407x series of DMMs offer unique capabilities

− Isolated Digitizer Mode at up to 1.8 MS/s

− Industry Leading Accuracy

− 2-year Calibration Cycle

• USB-4065 & PCMCIA-4050 for portable measurements

Example:

Fig 8 : Front Panel

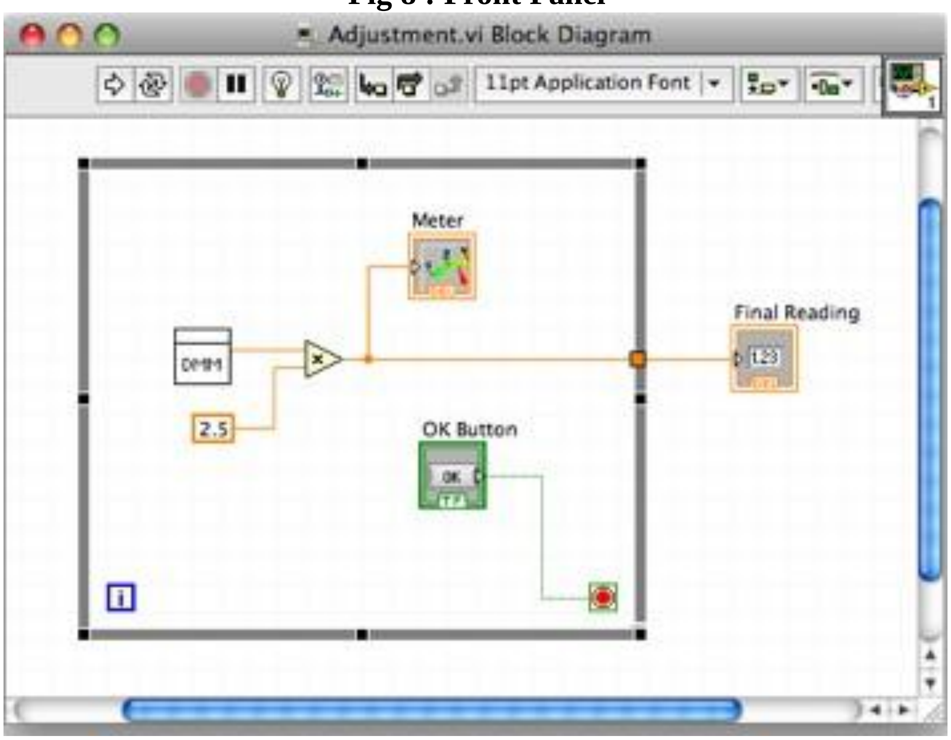


Fig 9: Block diagram of multimeter

Motion control employing stepper motor

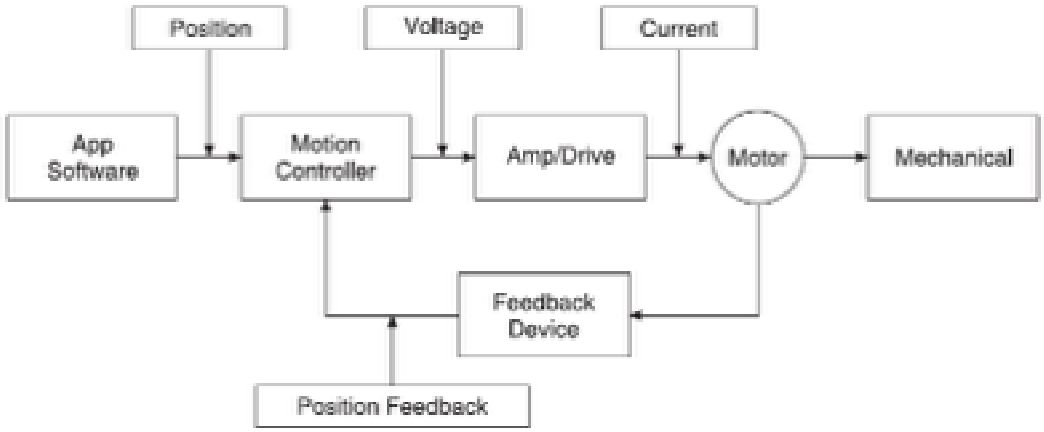


Fig 10: Components of motion control

Motion controller: The motion controller acts as brain of the system by taking the desiredtarget positions and motion profiles and creating the trajectories for the motors to follow, butoutputting a ±10 V signal for servomotors, or a step and direction pulses for stepper motors.

Amplifier or drive: Amplifiers (also called drives) take the commands from the controller andgenerate the current required to drive or turn the motor.

Motor: Motors turn electrical energy into mechanical energy and produce the torque required tomove to the desired target position.

Mechanical elements: Motors are designed to provide torque to some mechanics. These includelinear slides, robotic arms and special actuators.

Feedback device or position sensor: A position feedback device is not required for some motioncontrol applications (such as controlling stepper motors), but is vital for servomotors. The feedbackdevice, usually a quadrature encoder, senses the motor position and reports the result to the controller.

Image Acquisition and Processing

Image analysis combines techniques that compute statistics and measurements based on the graylevel intensities of the image pixels. Image processing contains information about lookup tables, convolution kernels, spatial filters and grayscale morphology. The lookup table (LUT) transformations are basic image-processing functions that highlight details in areas containing significant information at the expense of otherareas. These functions include histogram equalization, gamma corrections logarithmic corrections,and exponential corrections. NI-IMAQ is a complete and robust API for image acquisition. Whether you are using LabVIEW, Measurement Studio, Visual Basic, or Visual C++, NI-IMAQ gives you high-level control of NationalInstruments image acquisition devices. NI-IMAQ performs all of the computer- and board-specifictasks, allowing straightforward image acquisition without register-level

programming. NI-IMAQis compatible with NI-DAQ and all other National Instruments driver software for easily integrating
an imaging application into any National Instruments solution. NI-IMAQ is included with yourhardware at no charge that you can call from your application programming environment. These functions include routines for video configuration, image acquisition (continuous and single-shot), memory buffer allocation, triggercontrol and board configuration. NI-IMAQ performs all functionality required to acquire and saveimages. For image analysis functionality, refer to the IMAQ Vision software analysis librarieswhich are discussed later in this course. NI-IMAQ resolves many of the complex issues betweenthe computer and IMAQ hardware internally, such as programming interrupts and DMA controllers.

Fig 11: NI IMAQ functions

NI-IMAQ and IMAQ Vision use five categories of functions to acquire and display images:
● **Utility functions**—Allocate and free memory used for storing images; begin and endimage acquisition sessions
● **Single buffer acquisition functions**—Acquire images into a single buffer using the snapand grab functions
● **Multiple buffer acquisition functions**—Acquire continuous images into multiple buffersusing the ring and sequence functions

● **Display controls**—Display images for processing

● **Trigger functions**—Link a vision function to an event external to the computer, such asreceiving a pulse to indicate the position of an item on an assembly line

Snaps and grabs are the most basic types of acquisitions. A snap is simply a snapshot, inwhich you acquire a single image from the camera. A grab is more like a video, in which youacquire every image that comes from the camera. The images in a grab are displayed successively,producing a full-motion video, consisting of around 25 to 30 frames per second. IMAQ Snap,IMAQ Grab Setup and IMAQ Grab Acquire are used to snap and grab images.

IMAGE PROCESSING TOOLS AND FUNCTIONS IN IMAQ VISION

Utility functions include VIs for image management and manipulation, file management, calibrationand region of interest processing. Image processing functions include VIs for analysis, colorprocessing, frequency processing, filtering, morphology, operations, and processing, includingIMAQ Histogram, IMAQ Threshold and IMAQ Morphology. Machine vision VIs are used forcommon inspection tasks such as checking for the presence or absence of parts in an image ormeasuring dimensions in comparison to specifications. Some examples of the machine vision Visare the caliper and coordinate system VIs.

Image processing is a time-consuming process, both in computer processor time anddevelopment time. National Instruments has developed an application to accelerate the designtime of a machine vision application.. IMAQVision Assistant allows even the first-time vision developer to learn image processing techniques

and test inspection strategies. In addition, more experienced developers can develop and explorevision algorithms faster with less programming.

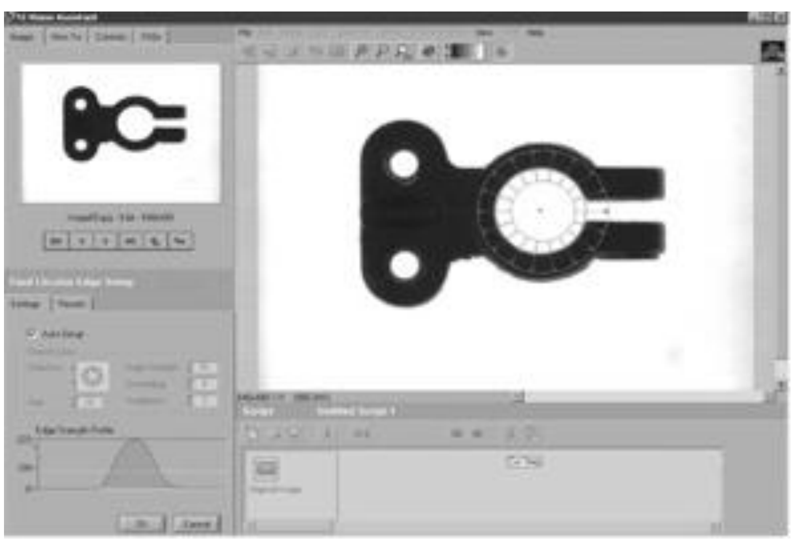


Fig 12: IMAQ Assistant

UNIT IV
INSTRUMENT INTERFACES

4–20mA Current Loop

Current-mode data transmission is the preferred technique in many environments, particularly in industrial applications. Most systems employ the familiar 2-wire, 4–20mA current loop, in which a single twisted-pair cable supplies power to the module and carries the output signal as well.
The 3-wire interface is less common but allows the delivery of more power to the module electronics. A 2-wire system provides only 4mA at the line voltage (the remaining 16mA carries the signal). Current loops offer several advantages over voltage-mode output transducers:
– They do not require a precise or stable supply voltage.
– Their insensitivity to IR drops makes them suitable for long distances.
– A 2-wire twisted-pair cable offers very good noise immunity.
– The 4mA of current required for information transfer serves two purposes:
it can furnish power to a remote module, and it provides a distinctionbetween zero (4 mA) and no information (no current flow).In a 2-wire, 4–20mA current loop, supply current for the sensor electronicsmust not exceed the maximum available, which is 4mA (the remaining 16mAcarries the signal). Because a 3-wire current loop is easily derived from the2-wire version, the following discussion focuses on the 2-wire version.

Need for a Current Loop

The 4–20mA current loop shown in Fig. 1 is a common method of transmittingsensor information in many industrial process-monitoring applications. A sensor is a device used to measure physical parameters such as temperature, pressure, speed, liquid flow rates, etc. Transmitting sensory information through a current loop is particularly useful when theinformationhas to be sent to a remote location over long distances (1,000 ft, ormore). The loop's operation is straightforward: a sensor's output voltage is first converted to a proportional current, with 4mA normally representing the sensor's zero-level output, and 20mA representing the sensor's full-scale output. Then, a receiver at the remote end converts the 4–20mA current back into a voltage which in turn can be further processed by a computer or display module. However, transmitting a sensor's output as a voltage over long
distances has several drawbacks. Unless very high input-impedance devices are used, transmitting voltages over long distances produces correspondingly lower voltages at the receiving end due wiring and interconnect resistances.However, high-impedance instruments can be sensitive to noise pickup since the lengthy signal-carrying wires often run in close proximity to other electrically noisy system wiring. Shielded wires can be used to minimize noise pickup, but their high cost may be prohibitive when long distances areinvolved. Sending a current over long distances produces voltage losses proportionalto the wiring's length. However, these voltage losses also known as loop drops" do not reduce the 4–20mA current as long as the transmitter anloop supply can compensate for these drops. The magnitude of the current inthe loop is not affected by voltage drops in the system wiring since all of thecurrent (i.e., electrons) originating at the negative (−) terminal of the loop power supply has to return back to its positive (+) terminal

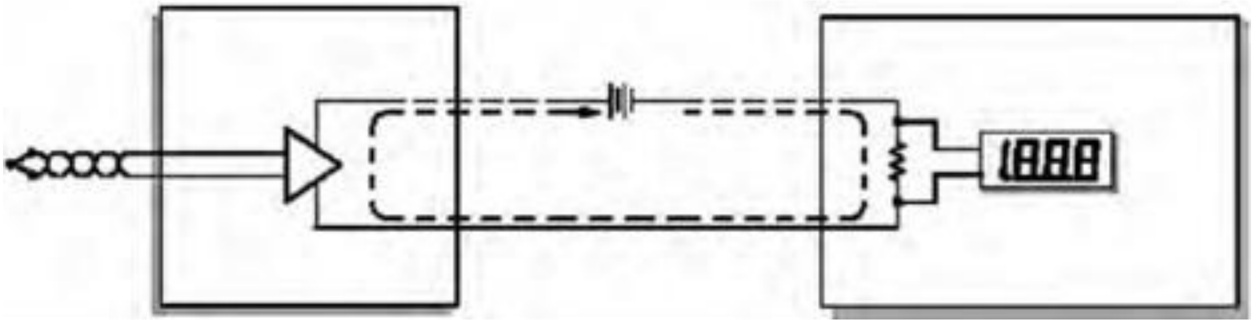


Fig1: Typical components in a loop

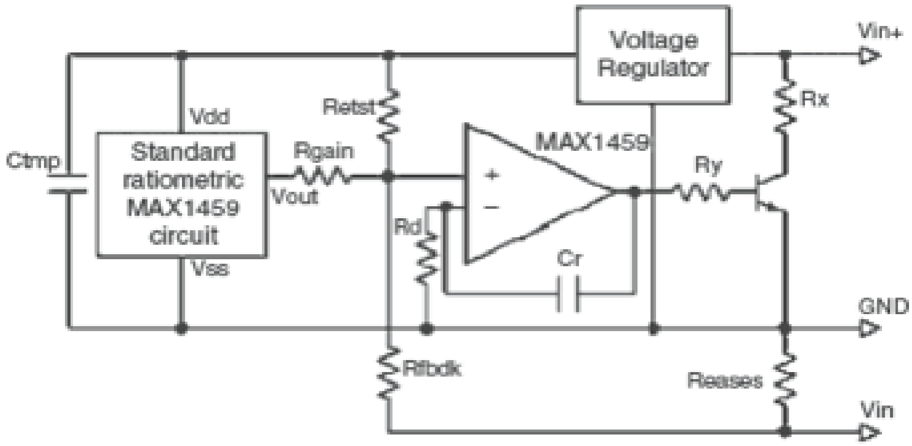


Fig2: 4-20 mA transmitter

Basic 2-wire Circuit

A voltage output should be converted to current when configuring a ratio metric 4–20mA circuit, because current-mode applications require a 4mA offset and 16mA span. This section presents the circuit details and results obtained from a current-loop configuration based on the MAX1459 sensorsignal conditioner. In principle, a voltage regulator is to be added, which converts the 10–32V loop voltage to a fixed 5V for operating the MAX1459. Figure shows the circuitry required for implementing a standard ratio metric version of the MAX1459 circuit. The voltage regulator can be any low-cost device whose quiescent current is sufficiently low for the 4mA budget.

Advantages of 4–20mA Current Loop

Current loops offer four major advantages such as:

– Long-distance transmission without amplitude loss
– Detection of offline sensors, broken transmission lines, and other failures
– Inexpensive 2-wire cables
– Lower EMI sensitivity

RS 232C/RS 485

The RS-232/485 port sequentially sends and receives bytes of information onebit at a time. Although this method is slower than parallel communication,which allows the transmission of an entire byte at once, it is simpler and canbe used over longer distances because of lower power consumption.For example, the IEEE 488 standard for parallel communication states thatthe cable length between the equipments should not exceed 20m total, witha maximum distance of 2m

between any two devices. RS-232/485 cabling,however, can extend 1,200m or greater. Typically, RS-232/485 is used totransmit American Standard Code for Information Interchange (ASCII) data.Although National Instruments serial hardware can transmit 7-bit as wellas 8-bit data, most applications use only 7-bit data. Seven-bit ASCII canrepresent the English alphabet, decimal numbers, and common punctuationmarks. It is a standard protocol that virtually all hardware and software understand.Serial communication is mainly using the three transmission lines:
(1) ground, (2) transmit, and (3) receive. Because RS-232/485 communicationis asynchronous, the serial port can transmit data on one line while receivingdata on another. Other lines such as the handshaking lines are not required.The important serial characteristics are baud rate, data bits, stop bits, andparity. To communicate between a serial instrument and a serial port on computer,these parameters must match.The RS-232 port, or ANSI/EIA-232 port, is the serial connection foundon IBM-compatible PCs. It is used for many purposes, such as connecting amouse, printer, or modem, as well as industrial instrumentation. The RS-232protocol can have only one device connected to each port. The RS-485 (EIA-485 Standard) protocol can have 32 devices connected to each port. Withthis enhanced multidrop capability the user can create networks of devicesconnected to a single RS-485 serial port. Noise immunity and multi dropcapability make RS-485, the choice in industrial applications requiring manydistributed devices networked to a PC or other controller for data collection.USB was designed primarily to connect peripheral devices to PCs, includingkeyboards, scanners, and disk drives.RS-232 (Recommended standard-232) is a standard interface approved bythe Electronic Industries Association (EIA) for connecting serial devices. Inother words, RS-232 is a long established standard that describes the physicalinterface and protocol for relatively low-speed serial data

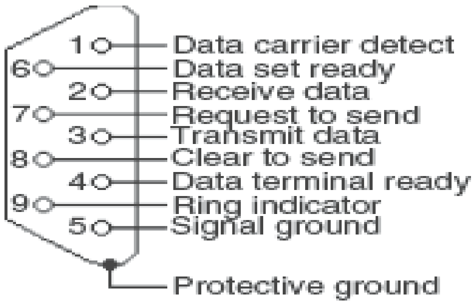


Fig 3: Pin diagram

Table: pin description

Pin Number	Signal	Description
1	DCD	Data carrier detect
2	RxD	Receive data
3	TxD	Transmit data
4	DTR	Data terminal ready
5	GND	Signal ground
6	DSR	Data set ready
7	RTS	Ready to send
8	CTS	Clear to send
9	RI	Ring indicator

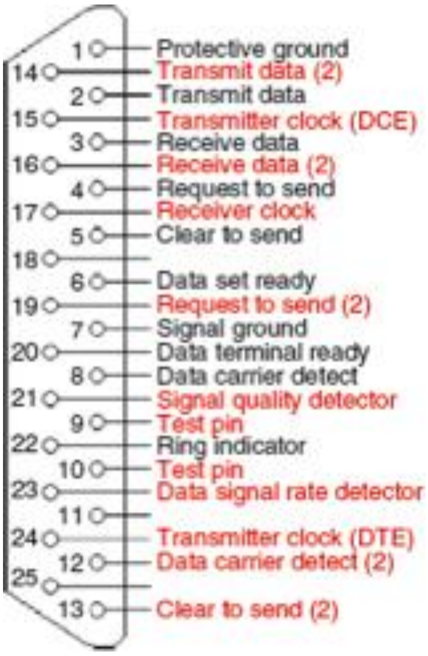


Fig 4: RS232c DB 25 pinout

Signal Descriptions

– TxD – This pin carries data from the computer to the serial device

– RXD – This pin carries data from the serial device to the computer

– DTR signals – DTR is used by the computer to signal that it is readyto communicate with the serial device like modem. In other words, DTRindicates the modem that the DTE (computer) is ON.

– DSR – Similarly to DTR, Data set ready (DSR) is an indication from themodem that it is ON.

– DCD – Data carrier detect (DCD) indicates that carrier for the transmitdata is ON.

– RTS – This pin is used to request clearance to send data to a modem.

– CTS – This pin is used by the serial device to acknowledge the computersRTS signal. In most situations, RTS and CTS are constantly onthroughout the communication session.

– Clock signals (TC, RC, and XTC) – The clock signals are only used forsynchronous communications. The modem or DSU extracts the clock fromthe data stream and provides a steady clock signal to the DTE. Thetransmit and receive clock signals need not have to be the same, or evenat the same baud rate.

– *CD – CD stands for Carrier detect*. Carrier detect is used by a modemto signal that it has a made a connection with another modem, or hasdetected a carrier tone. In other words, this is used by the modem tosignal that a carrier signal has been received from a remote modem.

RI – RI stands for ring indicator. A modem toggles (keystroke) the stateof this line when an incoming call rings in the phone. In other words,this is used by an autoanswer modem to signal the receipt of a telephonering signal. The carrier detect (CD) and the ring indicator (RI) lines areonly available in connections to a modem. Because most modems transmitstatus information to a PC when either a carrier signal is detected

RS485 Serial Communication

RS485 is an EIA standard for multipoint communications. It supports severaltypes of connectors, including DB-9 and DB-37. RS-485 is similar to RS-422 but can support more nodes per line. RS485 meets the requirements for atruly multipoint communications network, and the standard specifies up to32 drivers and 32 receivers on a single (2-wire) bus. With the introductionof "automatic" repeaters and high-impedance drivers/receivers this "limitation"can be extended to hundreds or even thousands of nodes on a network.The RS-485 and RS-422 standards have much in common, and are often confusedfor that reason. RS-485, which specifies bi-directional, half-duplex datatransmission, is the only EIA standard that allows multiple receivers anddrivers in "bus" configurations. RS-422, on the other hand, specifies a single,unidirectional driver with multiple receivers.

GPIB

The purpose of this section is to provide guidance and understanding of theGeneral Purpose Interface Bus (GPIB) bus to new GPIB bus users or toprovide more information on using the GPIB bus's features.GPIB Data Acquisition and Control Module provides analog and digitalsignals for controlling virtually any kind of a device and the capability toread back analog voltages, digital signals, and temperatures. The 4867 DataAcquisition and Control module is an IEEE-488.2 compatible device and has aStandard Commands for Programmable Instruments (SCPI) command parserthat accepts SCPI and short form commands for ease of programming. Applicationsinclude device control, GPIB interfacing and data logging. The 4867is housed in a small 7 in.×7 in.

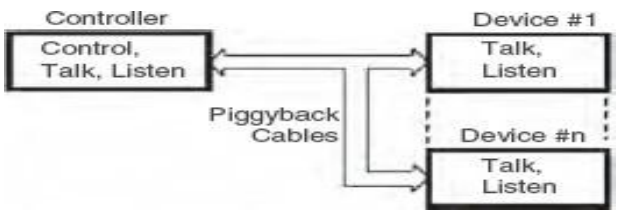


Fig 5: Block Diagram

Controllershave the ability to send commands, to talk data onto the bus and to listento data from devices. Devices can have talk and listen capability. Control canbe passed from the active controller (Controller-in-charge) to any device withcontroller capability. One controller in the system is defined as the SystemController and it is the initial controller-in-charge (CIC).Devices are normally addressable and have a way to set their address. Eachdevice has a primary address between 0 and 30. Address 31 is the unlisten oruntalk address.

Types of GPIB Messages

GPIB devices communicate with other GPIB devices by sending devicedependentmessages and interface messages through the interface system.**Device-dependent messages**, often called data or data messages, containdevice-specific information, such as programming instructions, measurementresults, machine status, and data files.**Interface messages** manage the bus. Usually called commands or commandmessages, interface messages perform such functions as initializing thebus, addressing and unaddressing devices, and setting device modes for remoteor local programming.The term “command” as used here should not be confused with somedevice instructions that are also called commands. Such device-specific commandsare actually data messages as far as the GPIB interface system itselfis concerned.

Physical Bus Structure

GPIB is a 24 conductor as shown in Fig 6Physically, the GPIB interfacesystem consists of 16 low-true signal lines and eight ground-return or shielddrainlines. The 16 signal lines, discussed later, are grouped into data lines(eight), handshake lines (three), and interface management lines (five).

Data Lines

The eight data lines, DIO1 through DIO8, carry both data and commandmessages. The state of the Attention (ATN) line determines whether theinformation is data or commands. All commands and most data use the 7-bitASCII or ISO code set, in which case the eighth bit, DIO8, is either unusedor used for parity.

Handshake Lines

Three lines asynchronously control the transfer of message bytes betweendevices. The process is called a 3-wire interlocked handshake. It guaranteesthat message bytes on the data lines are sent and received without transmissionerror.

NRFD (not ready for data) – Indicates when a device is ready or not readyto receive a message byte. The line is driven by all devices when receivingcommands by listeners when receiving data messages, and by the talkerwhen enabling the HS488 protocol.

– NDAC (not data accepted) – Indicates when a device has or has notaccepted a message byte. The line is driven by all devices when receivingcommands, and by Listeners when receiving data messages.

– DAV (data valid) – Tells when the signals on the data lines are stable(valid) and can be accepted safely by devices. The Controller drives DAVwhen sending commands, and the Talker drives DAV when sending datamessages.Three of the lines are handshake lines, NRFD, NDAC, and DAV, whichtransfer data from the talker to all devices who are addressed to listen. The
talker drives the DAV line; the listeners drive the NDAC and NRFD lines.The remaining five lines are used to control the bus’s operation.

– ATN (attention) is set true by the controller-in-charge while it is sendinginterface messages or device addresses. ATN is false when the bus istransmitting data.

– EOI (end or identify) can be asserted to mark the last character of amessage or asserted with the ATN signal to conduct a parallel poll.

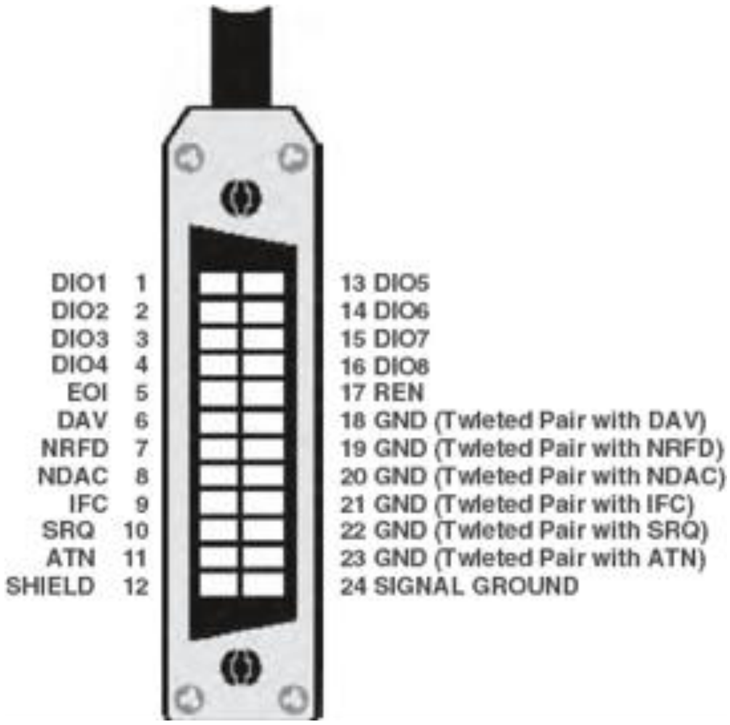


Fig :6 Bus structure of GPIB

USB

The USB is a medium-speed serial data bus designed to carry relatively largeamounts of data over relatively short cables: up to about 5m long. It cansupport data rates of up to 12Mbs−1 (megabits per second), which is fastenough for most PC peripherals such as scanners, printers, keyboards, mice,joysticks, graphics tablets, low-res digital cameras, modems, digital speakers,low speed CD-ROM and CD-writer drives, external Zip disk drives, and soon. The USB is an addressable bus system, with a 7-bit address code so it cansupport up to 127 different devices or nodes. However, it can have only onehost. So a PC with its peripherals connected through the USB forms a starLocal Area Network (LAN).On the other hand any device connected to the USB can have a numberof other nodes connected to it in daisy-chain fashion, so it can also form the

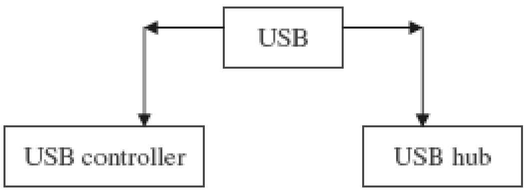


Fig 7: USB Structure

Most hubs provide either four or seven downstream ports, or less. Anotherimportant feature of the USB is that it is designed to allow hot swapping.Devices can be plugged into and unplugged from the bus without having toturn the power off and on again, re-boot the PC or even, manually starta driver program. A new device can simply be connected to the USB, andthe PCs operating system should recognize it and automatically set up thenecessary driver to service it.

Need for USB

The USB is host controlled. There can only be one host per bus. The specificationin itself does not support any form of multi master arrangement. Thisis aimed at and limited to single point to point connections such as a mobilephone and personal organizer and not multiple hub, multiple device desktopconfigurations. The USB host is responsible for undertaking all transactionsand scheduling bandwidth. Data can be sent by various transaction methodsusing a token-based protocol. One of the original intentions of USB was to reduce the amount of cabling at the back of PC. The idea came from theApple Desktop Bus, where the keyboard, mouse and some other peripheralscould be connected together (daisy chained) using the one cable.However, USB uses a tiered star topology, similar to that of 10BaseTEthernet. This imposes the use of a hub somewhere, which adds to greaterexpense, more boxes on desktop and more cables. However it is not as bad as itmay seem. Many devices have USB hubs integrated into them. For example,keyboard may contain a hub which is connected to computer. Mouse andother devices such as digital camera can be plugged easily into the back ofkeyboard. Monitors are just another peripheral on a long list which commonlyhas in-built hubs.This tiered star topology, rather than simply daisy chaining devicestogether has some benefits. First power to each device can be monitored andeven switched-off if an over current condition occurs without disrupting otherUSB devices. High, full and low speed devices can be supported, with the hubfiltering out high speed and full speed transactions so lower speed devices donot receive them. Up to 127 devices can be connected to any one USB bus atany one given time. To extent devices simply add another port/host. Whileearlier USB hosts had two ports, most manufacturers have seen this as limitingand are starting to introduce 4 and 5 port host cards with an internal portfor hard disks etc. The early hosts had one USB controller and thus both portsshared the same available USB bandwidth. As bandwidth requirements haveincreased, multiport cards with two or more controllers allowing individualchannels are used.

Table: Pin connections

Pin connections	
Pin No.	Signal
1	+5V Power
2	− Data
3	+ Data
4	Ground

USB cables use two different types of connectors:
– *Type A*. Plugs for the upstream end.
– *Type B*. Plugs for the downstream end.

PCMCIA

Personal Computer Memory Card International Association (PCMCIA) isan international standards body and trade association founded in 1989developed a standard for small, credit card-sized devices, called PC Cards.Originally designed for adding memory to portable computers, the PCMCIA standard has been expanded several times and is now suitable for many typesof devices. The inclusion of PCMCIA technology in PCs delivers a varietyof benefits. Besides providing an industry standard interface for third-partycards (PC Cards), PCMCIA allows users to easily swap cards in and out of aPC as needed, without having to deal with the allocation of system resourcesfor those devices. These useful features hot swapping and automatic configuration,as well as card slot power management and other PCMCIA capabilities –are supported by a variety of software components on the PCMCIA-based PC.In most cases, the software aspect of PCMCIA remains relatively transparentto the user. As the demand for notebook and laptop computers began skyrocketingin the late 1980s, users realized that their expansion options werefairly limited. Mobile machines were not designed to accept the wide array ofavailable expansion cards that their desktop counterparts could enjoy

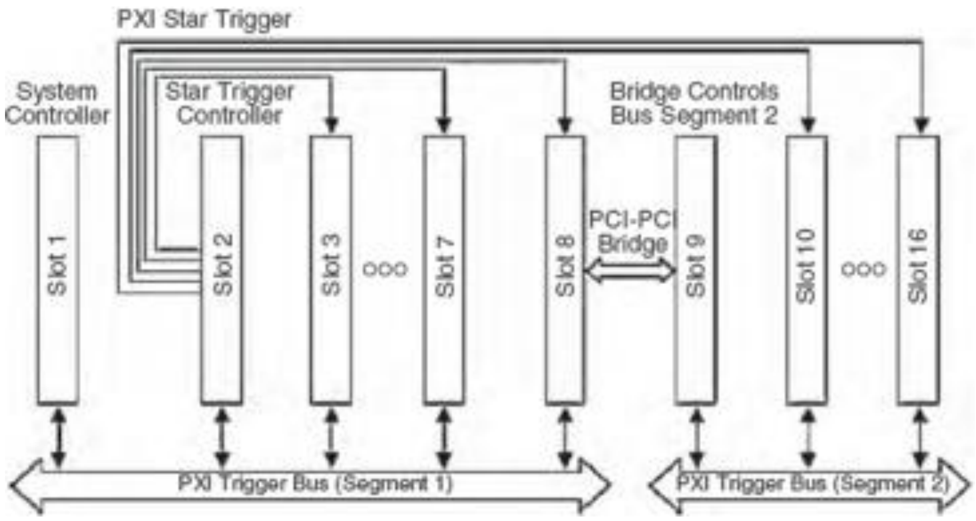


Fig 8: Triggering Architecture

Features of PCMCIA

– One rear slot, access from rear of PC
– Accept Type I/II/III Cards
– Comply with PCI Local Bus Specification Rev.2.2
– Comply with 1995 PC Card Standard
– Extra compatible registers are mapped in memory
– Use T/I 1410 PCMCIA controller
– Support PC Card with Hot Insertion and Removal
– Support 5V or 5/3.3V 16-bit PC Cards
– Support Burst Transfers to Maximize Data Throughput on both PCIBuses
– Supports Distributed DMA and PC/PCI DMA

Utilities of PCMCIA Card in the Networking Category

Under the networking category, typically PCMCIA slot supports 4 types of
cards such as:
– LAN card
– Wireless LAN card
– Modem card
– ATA flash disk card

VISA

Virtual Instrumentation Software Architecture (VISA) is a standard I/O languagefor instrumentation programming. VISA by itself does not provideinstrumentation programming capability. VISA is a high-level API that callsinto lower level drivers. VISA
is capable of controlling VXI, GPIB, or Serial instruments and makes the appropriate
driver calls depending on the type of instrument being used. Whendebugging VISA problems, it is important to keep in mind that this hierarchyexists.

Fig9: VISA Architecture

The terminology related with VISA are explained as follows:
Resources. The most important objects in the VISA language are known asresources.*Operations*. In object-oriented terminology, the functions that can be usedwith an object are known as operations *Attributes*. In addition to the operations that can be used with an object, theobject has variables associated with it that contain information related tothe object. In VISA, these variables are known as attributes. There is a default resource manager at the top of the VISA hierarchythat can search for available resources or open sessions to them. Resourcescan be GPIB, serial message based VXI, or register-based VXI. The mostcommon operations for message based instruments are read and write.

Waveform generator

Stand-alone traditional instruments such as oscilloscopes and waveform generatorsare very powerful, expensive, and designed to perform one or morespecific tasks defined by the vendor. However, the user generally cannot extendor customize them. The knobs and buttons on the instrument, the built-incircuitry, and the functions available to the user, are all specific to the natureof the instrument. In addition, special technology and costly components mustbe developed to build these instruments, making them very expensive and slowto adapt.

UNIT V
APPLICATION OF VI

Fourier Transforms

LabVIEW and its analysis VI library provide a complete set of tools to perform Fourier and spectral analysis. The Fast Fourier Transform (FFT) and Power Spectrum VIs are optimized, and their outputs adhere to the standard DSP format.

FFT is a powerful signal analysis tool, applicable to a wide variety of fields including spectral analysis, digital filtering, applied mechanics, acoustics, medical imaging, modal analysis, numerical analysis, seismography, instrumentation, and communications.

The LabVIEW analysis VIs, located on the **Signal Processing** palette, maximize analysis throughput in FFT-related applications. This document discusses FFT properties, how to interpret and display FFT results, and how to manipulate FFT and power spectrum results to extract useful frequency information.

FFT Properties

The fast Fourier transform maps time-domain functions into frequency-domain representations.
FFT is derived from the Fourier transform equation, which is

$$X(f) = \boldsymbol{F}\{x(t)\} = \int_{-\infty}^{\infty} x(t)\, e^{-j2\pi ft}\, dt \;,$$

where x(t) is the time domain signal, X(f) is the FFT, and ft is the frequency to analyze.Similarly, the discrete Fourier transform (DFT) maps discrete-time sequences into discrete-frequency representations. DFT is given by the following equation

$$X_k = \sum_{i=0}^{n-1} x_i\, e^{-j2\pi ik/n} \qquad \text{for } k = 0, 1, 2, \ldots, n-1$$

where x is the input sequence, X is the DFT, and n is the number of samples in both the discrete-time and the discrete-frequency domains. Direct implementation of the DFT, as shown in equation 2, requires approximately n complex operations. However, computationally efficient algorithms can require as little as n log2(n) operations.

$$\boldsymbol{F}\{x\} = X = X_{Re} + j\,X_{Im} = Re\{X\} + j\,Im\{X\}$$

An inherent DFT property is the following:

$$X_{n-i} = X_{-i}$$

FFT Standard Output:

The output format of the FFT VI can now be described with Xn-i. If the total number of samples, n, of the input sequence, x, is even, the format of the complex output sequence, X, and the interpretation of the results are shown in the below table.

Table 1: Output Format for Even n, k = n ÷ 2

Array Element	**Interpretation**
X[0]	DC component
X[1]	1st harmonic or fundamental
X[2]	2nd harmonic
.	.
.	.
.	.
X[k-2]	(k-2)th harmonic
X[k-1]	(k-1)th harmonic
X[k] = X[-k]	Nyquist harmonic
X[k+1]= X[n-(k-1)] = X[-(k-1)]	(k-1)th harmonic
.	
.	
.	
X[n-3]	-3rd harmonic
X[n-2]	-2nd harmonic
X[n-1]	-1st harmonic

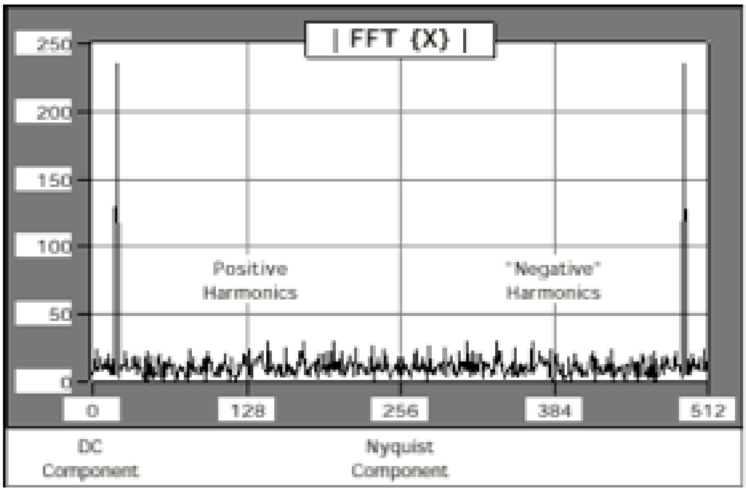


Fig 1 : Graph of table

If the total number of samples, n, of the input sequence, x, is odd, the format of the complex output sequence, X, and the interpretation of the results are shown in the below table

Table:2Output Format for Odd n, k = (n-1) ÷ 2

Array Element	Interpretation
X[0]	DC component
X[1]	1st harmonic or fundamental
X[2]	2nd harmonic
.	.
.	.
.	.
X[k-1]	kth -1harmonic
X[k]	kth harmonic
X[k+1]= X[n-k] = X[-k] kth	kth harmonic
. X[k+2]= X[n-(k-1)] = X[-(k-1)]	- (k-1)th harmonic
.	
.	
X[n-3]	-3rd harmonic
X[n-2]	-2nd harmonic
X[n-1]	-1st harmonic

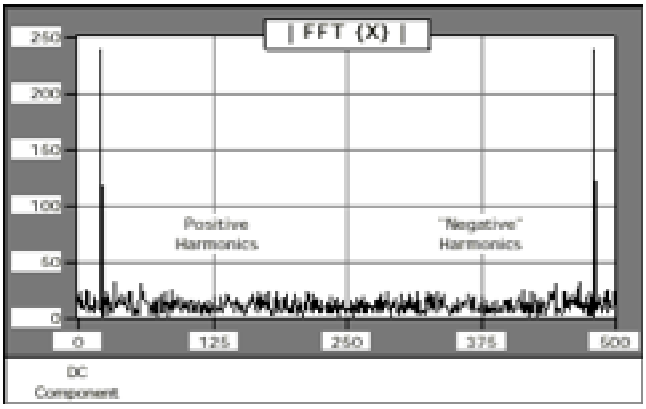


Fig 2 : Graph of table 2

Standard Output

Standard output is the format for even and odd-sized discrete-time sequences, described in Tables 1 and 2 of this document. This format is convenient because it does not require any further data manipulation.To graphically display the results of the FFT, wire the output arrays to the waveform graph Fig : 2 Graph of table2. The FFT output is complex and requires two graphs to display all the information.

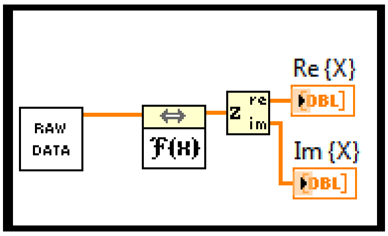


Fig 3 : Standard output by using VI

Power spectrum and Correlation:

The power spectrum reveals the existence, or the absence, of repetitive patterns and correlation structures in a signal process. These structural patterns are important in a wide range of applications such as data forecasting, signal coding, signal detection, radar, pattern ecognition, and decision-making systems. The most common method of spectral estimation is based on the fast Fourier transform (FFT).

In particular, the total power is given by

$$P_x(\omega)=\sum_{k=-\infty}^{\infty} r_x[k]\,e^{-jk\omega}$$

One can show that Px(w) is real, even and positive. The auto-correlation can be recovered with the inverse Fourier transform

$$r_x[k]=\frac{1}{2\pi}\int_{-\pi}^{\pi} d\omega\,P_x(\omega)\,e^{jk\omega}$$

Power spectrum – properties:

In particular, the total power is given by

$$r_x[0]=\frac{1}{N}\sum_{n=1}^{N}|x[n]|^2=\frac{1}{2\pi}\int_{-\pi}^{\pi} d\omega\,P_x(\omega)$$

The power spectrum is sometimes called **spectral density** because it is *positive* and the signal power can always be*normalized* to *r*(0) =(2p)-1 .

Example: Uncorrelated noise has a constant power spectrum

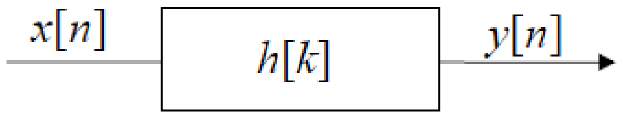
$$r[k]=\sigma^2\delta(k)$$

$$P_x(\omega)=\sum_{k=-\infty}^{\infty}\sigma^2\delta(k)e^{-jk\omega}=\sigma^2$$

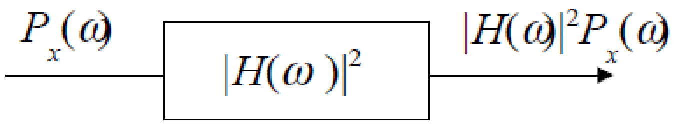
Hence it is also called **white noise**.

Effect of Filtering on Power Spectrum

A linear system with impulse response $h[k]$



Transforms the power spectrum as



Spectral Content

The power spectrum gives the spectral content of the data. To see that consider the power of a signal after filtering with a narrow bandpass filter around w 0.

$$\begin{aligned} E[|y[n]|^2] &= \frac{1}{2\pi}\int_{-\pi}^{\pi} d\omega\, P_y(\omega) \\ &= \frac{1}{2\pi}\int_{-\pi}^{\pi} d\omega\, |H(\omega)|^2 P_x(\omega) \\ &= \frac{1}{2\pi}\int_{\omega_0-\Delta\omega/2}^{\omega_0+\Delta\omega/2} d\omega\, P_x(\omega) \\ &\approx \frac{\Delta\omega}{2\pi} P_x(\omega_0) \end{aligned}$$

Power spectrum - properties

The power spectrum captures the spectral content of the sequence. It can be estimate directly from the Fourier transform of the data.

$$\hat{P}_x(\omega) = \frac{1}{N}|X(\omega)|^2$$

$$X(\omega) = \sum_{k=0}^{N-1} x[k] e^{-j\omega k}$$

Correlation methods

Correlation is a statistical technique that can show whether and how strongly pairs of variables are related.

Correlation Coefficient

The main result of a correlation is called the **correlation coefficient** (or "r"). It ranges from -1.0 to +1.0. The closer r is to +1 or -1, the more closely the two variables are related.

Cross and Auto-correlation

The cross-correlation is defined as

$$r_{yx}(k) = \sum_{n=-\infty}^{\infty} y^*[n]x[n+k]$$

Note that correlation is a convolution with opposite sign. It can be computed with the Fourier transform.

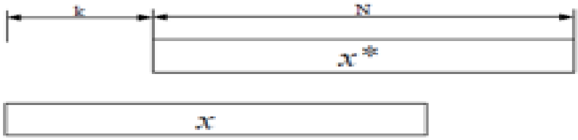
$$R_{xy}(\omega) = Y^*(\omega)X(\omega)$$

The auto-correlation is defined as

$$r_x(k) = \sum_{n=-\infty}^{\infty} x^*[n]x[n+k]$$

For a sample of finite length N this is typically normalized. We call this the sample auto-correlation

$$\hat{r}_x[k] = \frac{1}{N-k} \sum_{n=1}^{N-k} x^*[n]x[n+k]$$



Use xcorr() to compute cross of auto-correlation.

Auto-correlation properties

The auto-correlation is symmetric.

$$r_x[k] = r_x^*[-k]$$

The zero lag gives the total power of the signal

The auto-correlation has the power as an upper-bound

$$r_x[0] \geq |r_x[k]| \quad r_x[0] = \sum_n |x[n]|^2$$

Windowing & flittering

Windowing is the process of taking a small subset of a larger dataset, for processing and analysis. A naive approach, the rectangular window, involves simply truncating the dataset before and after the window, while not modifying the contents of the window at all. However this is a poor method of windowing and causes power leakage.

Windowing of a simple waveform like cos ωt causes its Fourier transform to develop non-zero values (commonly called spectral leakage) at frequencies other than ω. The leakage tends to be worst (highest) near ω and least at frequencies farthest from ω.

If the waveform under analysis comprises two sinusoids of different frequencies, leakage can interfere with the ability to distinguish them spectrally. If their frequencies are dissimilar and one component is weaker, then leakage from the stronger component can obscure the weaker one's presence. But if the frequencies are similar, leakage can render them unresolvable even when the sinusoids are of equal strength. The rectangular window has excellent resolution characteristics for sinusoids of comparable strength, but it is a poor choice for sinusoids of disparate amplitudes. This characteristic is sometimes described as low-dynamic-range.

At the other extreme of dynamic range are the windows with the poorest resolution and sensitivity, which is the ability to reveal relatively weak sinusoids in the presence of additive random noise. That is because the noise produces a stronger response with high-dynamic-range windows than with high-resolution windows. Therefore, high-dynamic-range windows are most often justified in wideband applications, where the spectrum being analyzed is expected to contain many different components of various amplitudes.

In between the extremes are moderate windows, such as Hamming and Hann. They are commonly used in narrowband applications, such as the spectrum of a telephone channel. In summary, spectral analysis involves a tradeoff between resolving comparable strength components with similar frequencies and resolving disparate strength components with dissimilar frequencies. That tradeoff occurs when the window function is chosen.

Application in Process Control projects

Software for Remote ON/OFF Control Experiment

The software for the Remote ON/OFF controller lab experiment is developed using LabVIEW.
There are two distinct parts in the software:

1. ON/OFF controller server program and logic
2. Internet communication using DataSocket protocol

These parts will be explained in the following sections

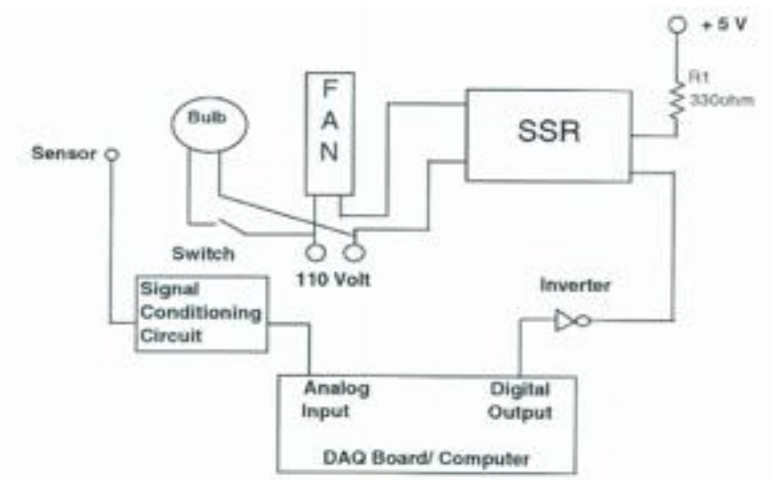


Fig 4: ON-OFF Temperature Control

ON/OFF Controller Server Program and Logic

The LabVIEW program on the server first reads the voltage from the analog input channel of the DAQ board and converts it to a temperature using the equation

$$Tout = 6\ Vin + 60$$

where Tout is the process temperature, and Vin is the input voltage from the DAQ board. The desired Set-point temperature value of the process is obtained from the client computer, and the Error is determined using the equation

Error = Set-point . Tout.

Based on the error and neutral zone High Limit and Low Limit, an ON/OFF controller logic is implemented using LabVIEW. In this logic, Digital Output Channel is the output logic value sent to the DAQ board which controls the fan through the SSR, Cooling Fan Indicators are the LED ON/OFF indicators on the front panel of the LabVIEW VI that show the state of the fan, and Within Limits Indicator shows if the process is operating within the neutral zone. The Cooling Fan ON Indicator is also defined as a local read only variable which is used in the logic implementation.

The LabVIEW front panel of the server program VI

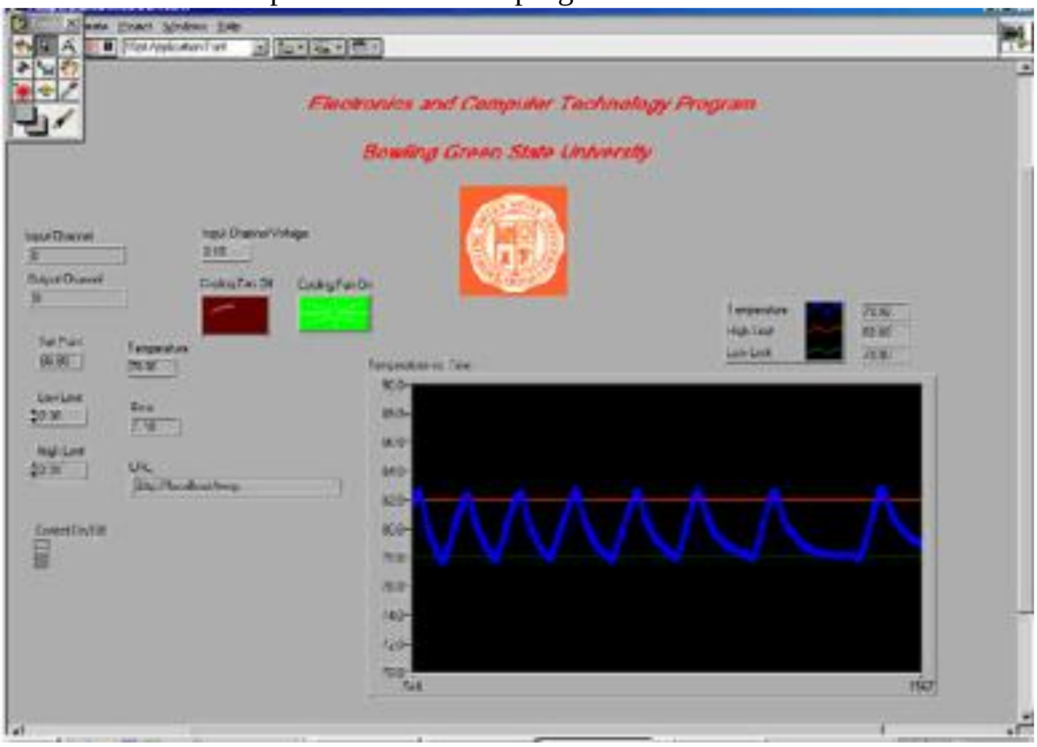


Fig 5: Server front panel

Temperature data acquisition system

The hardware for server workstation consists of a PC with Pentium III, 550 MHz processor, 128 Mb RAM, Network Interface Card (NIC), National Instruments DAQ board. The server software includes Windows 98, NI-DAQ driver, LabVIEW 5.1 and NI-DataSocket Manager. The designed experiment is connected to the DAQ board. The server is assigned static IP address. The clients could be any PC.s with NIC that can run a LabVIEW program. The objective of the experiment is to maintain the temperature inside a wooden box at some desired set-point value, within neutral zone limits, using a two-state-controller mode. The wooden box is heated with a light bulb. The temperature is measured using LM335 solid-state The hardware for server workstation consists of a PC with Pentium III, 550 MHz processor, 128 Mb RAM, Network Interface Card (NIC), National Instruments DAQ board. The server software includes Windows 98, NI-DAQ driver, LabVIEW 5.1 and NI-DataSocket Manager. The designed experiment is connected to the DAQ board. The server is assigned static IP address. The clients could be any PC.s with NIC that can run a LabVIEW program. The objective of the experiment is to maintain the temperature inside a wooden box at some desired set-point value, within neutral zone limits, using a two-state-controller mode. The wooden box is heated with a light bulb. The temperature is measured using LM335 solid-state

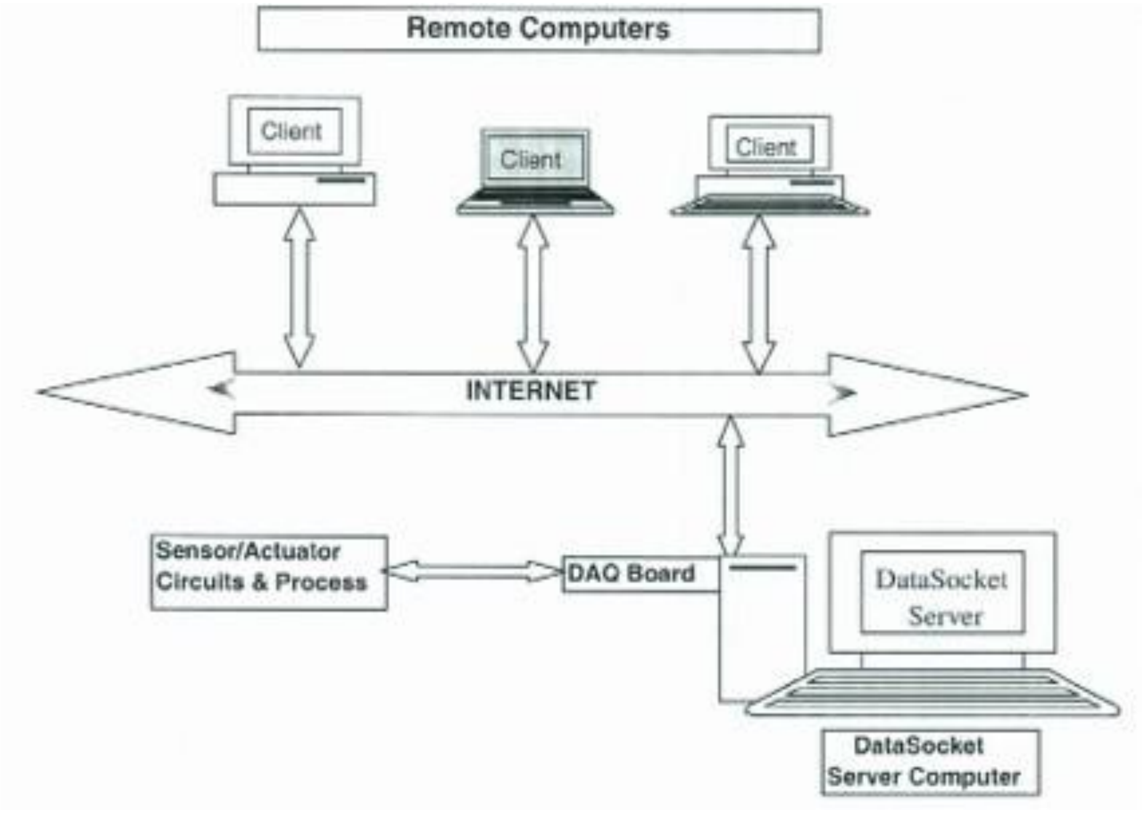


Fig 6: Temperature data Acquisition

Oscilloscope

Open Windows oscilloscopes from the 5000, 6000, 7000, and 8000 series are Windows-based and contain scope-specific software for acquisition, connectivity, and control. By default, the scope user interface runs Windows-based application software that performs the following
scope-specific tasks:

• Presents the scope user interface
• Configures scope hardware based on commands from the user and
signal content
• Displays acquired signals and derivations of acquired signals

You can use LabVIEW to extend the feature list of your scope to include
the following tasks:

• Custom analysis and signal processing of acquired signals
• Automation of scope-related tasks and sequences of tasks
• A unique and customizable user interface
• Custom reports, including reports that you can publish live over the Internet
• Remote, Internet-based control and monitoring

Example:
Connect the function generator to the scope and put up the sample waveform. Increase the amplitude to 2v and pull down the VISA box and click on GPIB for the oscilloscope. The display of the wave is shown in the fig 7.

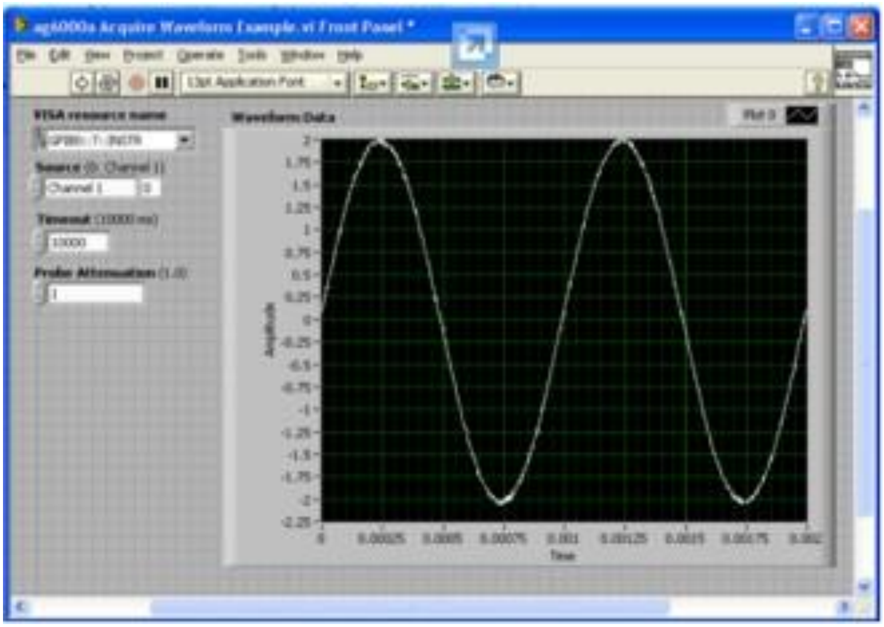


Fig 7: waveform acquired by oscilloscope

Digital Multimeter
Digital Multimeters (DMMs) are specialized in takingflexible, high resolution measurements
• Low to medium acquisition speeds
• High resolution
• Measures current or resistance in addition to voltage.
• Different form factors: PXI, PCIe, PCI, USBSoftware Support in NI LabVIEW and SignalExpressThe NI 407x series of DMMs offer unique capabilities
− Isolated Digitizer Mode at up to 1.8 MS/s
− Industry Leading Accuracy
− 2-year Calibration Cycle
• USB-4065 & PCMCIA-4050 for portable measurements

Example:

Fig 8 : Front Panel

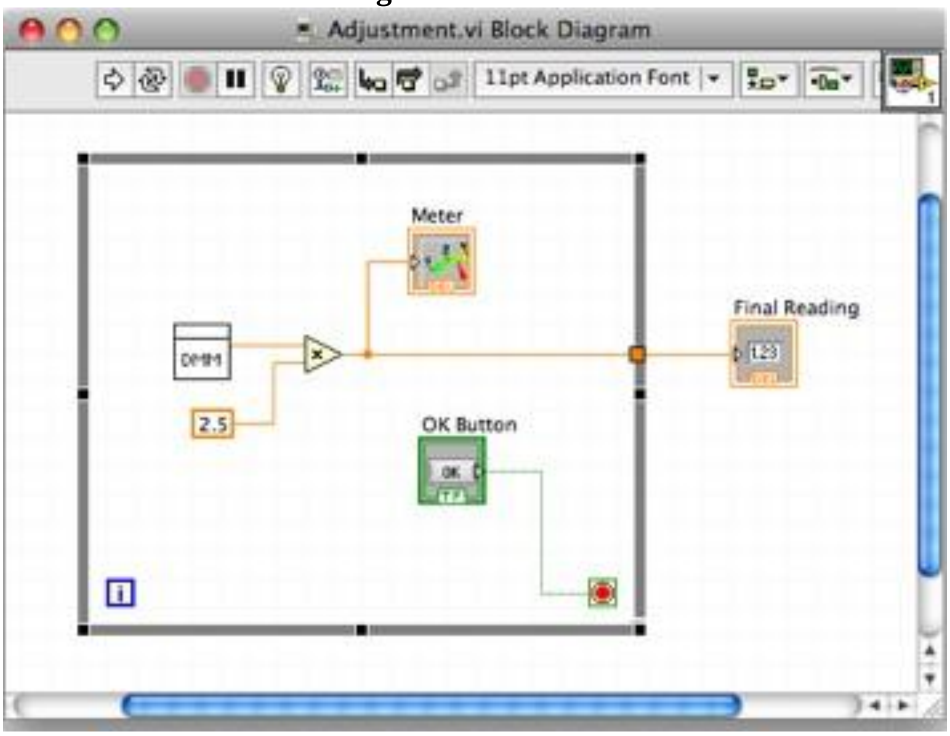


Fig 9: Block diagram of multimeter

Motion control employing stepper motor

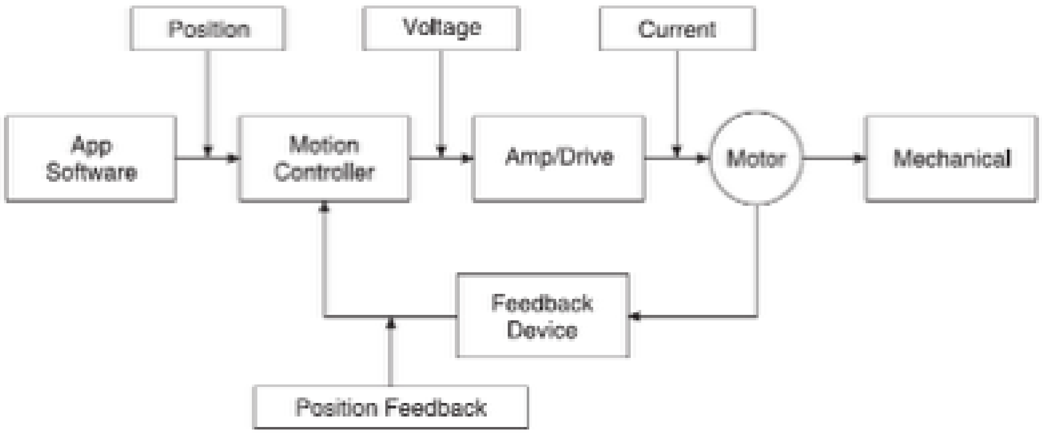


Fig 10: Components of motion control

Motion controller: The motion controller acts as brain of the system by taking the desiredtarget positions and motion profiles and creating the trajectories for the motors to follow, butoutputting a ±10 V signal for servomotors, or a step and direction pulses for stepper motors.

Amplifier or drive: Amplifiers (also called drives) take the commands from the controller andgenerate the current required to drive or turn the motor.

Motor: Motors turn electrical energy into mechanical energy and produce the torque required tomove to the desired target position.

Mechanical elements: Motors are designed to provide torque to some mechanics. These includelinear slides, robotic arms and special actuators.

Feedback device or position sensor: A position feedback device is not required for some motioncontrol applications (such as controlling stepper motors), but is vital for servomotors. The feedbackdevice, usually a quadrature encoder, senses the motor position and reports the result to the controller.

Image Acquisition and Processing

Image analysis combines techniques that compute statistics and measurements based on the graylevel intensities of the image pixels. Image processing contains information about lookup tables, convolution kernels, spatial filters and grayscale morphology. The lookup table (LUT) transformations are basic image-processing functions that highlight details in areas containing significant information at the expense of otherareas. These functions include histogram equalization, gamma corrections logarithmic corrections,and exponential corrections. NI-IMAQ is a complete and robust API for image acquisition. Whether you are using LabVIEW,
Measurement Studio, Visual Basic, or Visual C++, NI-IMAQ gives you high-level control of NationalInstruments image acquisition devices. NI-IMAQ performs all of the computer- and board-specifictasks, allowing straightforward image acquisition without register-level programming. NI-IMAQis compatible with NI-DAQ and all other National Instruments driver software for easily integrating
an imaging application into any National Instruments solution. NI-IMAQ is included with yourhardware at no charge that you can call from your application programming environment. These functions include routines for video configuration, image acquisition (continuous and single-shot), memory buffer allocation, triggercontrol and board configuration. NI-IMAQ performs all

functionality required to acquire and saveimages. For image analysis functionality, refer to the IMAQ Vision software analysis librarieswhich are discussed later in this course. NI-IMAQ resolves many of the complex issues betweenthe computer and IMAQ hardware internally, such as programming interrupts and DMA controllers.

Fig 11: NI IMAQ functions

NI-IMAQ and IMAQ Vision use five categories of functions to acquire and display images:

- **Utility functions**—Allocate and free memory used for storing images; begin and endimage acquisition sessions
- **Single buffer acquisition functions**—Acquire images into a single buffer using the snapand grab functions
- **Multiple buffer acquisition functions**—Acquire continuous images into multiple buffersusing the ring and sequence functions
- **Display controls**—Display images for processing
- **Trigger functions**—Link a vision function to an event external to the computer, such asreceiving a pulse to indicate the position of an item on an assembly line

Snaps and grabs are the most basic types of acquisitions. A snap is simply a snapshot, inwhich you acquire a single image from the camera. A grab is more like a video, in which youacquire every image that comes from the camera. The images in a grab are displayed successively,producing a full-motion video, consisting of around 25 to 30 frames per second. IMAQ Snap,IMAQ Grab Setup and IMAQ Grab Acquire are used to snap and grab images.

IMAGE PROCESSING TOOLS AND FUNCTIONS IN IMAQ VISION

Utility functions include VIs for image management and manipulation, file management, calibrationand region of interest processing. Image processing functions include VIs for analysis, colorprocessing, frequency processing, filtering, morphology, operations, and processing, includingIMAQ Histogram, IMAQ Threshold and IMAQ Morphology. Machine vision VIs are used forcommon inspection tasks such as checking for the presence or absence of parts in an image ormeasuring dimensions in comparison to specifications. Some examples of the machine vision Visare the caliper and coordinate system VIs.

Image processing is a time-consuming process, both in computer processor time anddevelopment time. National Instruments has developed an application to accelerate the designtime of a machine vision application.. IMAQVision Assistant allows even the first-time vision developer to learn image processing techniques

and test inspection strategies. In addition, more experienced developers can develop and explorevision algorithms faster with less programming.

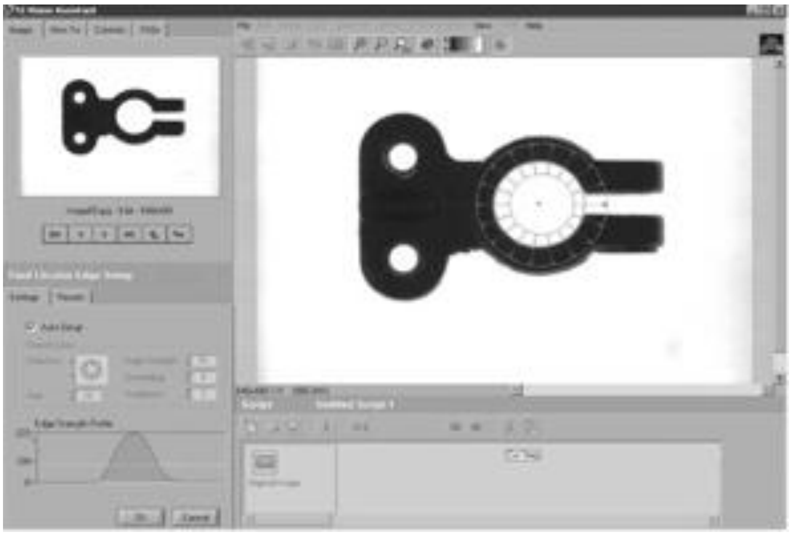


Fig 12: IMAQ Assistant

Questions	opt1	opt2	opt3	opt4	Answer
The ________ is the user interface of a VI and specifies the input and output of the VI	Front panel	block diagram	indicator	control	front panel
By the use of the _________we place the controls and indicators on the front panel	Function I	control palette	tool	interface	control palette
With the __________we can place VIs and functions on the block diagram	Function Palette	control palette	tool	interface	functions palette
As the interest of the computer grew user duplicated the _________ functions of the existing instruments through the computer interface	interface	front panel	processor	tool	front panel
VI lies in progressively moving the intelligence of the instrument into_____	software	hardware	tool	interface	software
In VI all the signal handling and control action is done through___________	hardware	external device	software	processor	external device
The primary difference between hardware instrumentation and virtual instrumentation is__________	software is used to replace hardware	hardware is used to replace	both aand b	none of the above	software is used to replace hardware
_________is the most important component of a virtual instrument.	software	hardware	processing module	sensor	software
an important advantage that software provides is	modularity	linearity	both a and b	none of the above	modularity
A person controls a software application hardware device is known as_________	virtual interface	user interface	sensor interface	none of the above	user interface
The user interface are classified into how many types	3	4	5	6	3
The LabVIEW is a _____________software	GUI	MDI	CLI	none of the above	GUI
In LabVIEW we build a user interface by use of_____________	controls	indicators	both a and b	none of the above	both a and b
Graph is a type of____________	indicator	data	control	none of the above	indicator
The block diagram window contains_________	graphical functioning modules	controls and indicators	both a and b	none of the above	graphical source
the objects on the front diagram window appears as _________ on the block diagram	terminals	wires	instructions	data	terminals
________are the objects on the block diagram that have inputs/outputs and perform operations	nodes	controls	indicators	terminals	nodes
Terminals are entry and exit ports that exchange information between the __________ & _________	controls and indicators	front panel and block diagram	sensors and computer	nodes and wires	front panel and block diagram
The interface is put between _______ & ________	data and c	sensor anc	wires and	none of the above	sensor and computer
The _________ detects the physical signal and transmit into electrical form.	database i	sensor mo	processing	computing	sensor module
through which the sensor module communicate with the computer.	database i	sensor inte	computing	none of th	sensor interface
sensor module performs__________	signal con	signal mar	signal con	none of the above	signal conditioning
the sensor module consist of how many parts	3	4	5	6	3
The A/D convertor changes the detected and conditional voltage into ______	digital valı	analog value	amplified signal	both b and c	digital value
GPIB stands for	general pu	general pr	graphical	none of the above	general purpose interface bus
LabVIEW is a __________ environment.	graphical	assembly	object orie	none of the above	graphical programming
In LabVIEW ______that compiles into machine code and eliminates a lot of the syntectial details.	graphical l	graphical i	programm	none of the above	graphical block diagram
Labview offers more _________than standard laboratory	flexibility	linearity	performan	none of the above	flexibility
Labview programs are also called________	VIRTUAI	OOPS	GUI	none of the above	VIRTUAL INSTRUMENTS

The _______accompany the front panel in a VI.	palettes	tools	block diag	none of the above	block diagram
The function of block biagram is used to control__________	nodes	front pane	coding	none of th	front panel objects
The ________is the actual executable program.	front pane	icons	block diag	nodes	block diagram
The lower level VI's,built in functions,constants and program execution control structures are the components of________	block diag	front pane	both a and	none	block diagram
_________have analogous terminals on the block diagram.	tools	front pane	block diag	wires	front panel objects
________is the easiest most powerful tool for acquiring,analysing and presenting real world data.	C++	LabVIEW	java	C	LabVIEW
An icon is a graphical representation of ________	VI	instrument	both a and	none of th	VI
how many types of palettes are available in lab view ?	9	3	4	5	3
The tool palettes are available on ______	BLOCK D	FRONT P	BOTH A .	none of th	both a and b
A tool is a special operating mode of the ___________	coding	icons	mouse cur	none of th	mouse cursor
The cursor corresponds to the icon of the tools selected in the _______	tools palet	control pa	Function p	none of th	Tools palettes
Shortcut Key to open block diagram of a VI is________	cntrl+alt	cntrl+E	cntrl+B	none	cntrl+E
Syntax is must for _________ programming	Graphical	Text-base	both	none	Text-based
Data/program flow can be visualized in _________ programming	Graphical	Text-base	both	none	Graphical
Shortcut key to clear all broken wires is _______	cntrl+alt	cntrl+E	cntrl+B	none	cntrl+B
An error object can be viewed in Labview through ___________ option	place erro	show erro	exhibit err	none	show error
Can a error VI be run in LAbVIEW	yes	no	run but no	none	no
Connector pane is visible in the right corner of _________	front pane	icons	block diag	nodes	front panel
_______________ is non interactive	LabVIEW	Simulink	Python	both a and	Python
An example for text based programing is ______________	LabVIEW	Simulink	Python	both a and	Python
Graphical programming are chosen as it is ______________	accurate	less expen	easy to us	all the abo	all the above

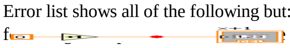
UNIT II

Questions	opt1	opt2	opt3	opt4	Answer
The integer and real is a type of_______	boolean d	numeric data type	string data types	none of the above	numeric data type
________objects simulate switches,push buttons and LEDs.	boolean	numeric data type	string data types	none of the above	boolean
String data types is a sequence of________	hexadecir	EBCIDIC characters	decimal	ASCII characters	ASCII characters
_______is used to receive text from the user	string con	string indicator	boolean control	numeric control	string control
The ____________ is used to control repetitive operation	data types	loops	control	indicator	loops
The FOR LOOP is located on the _________	structure p	numeric palette	control palette	none of the above	structure palette
Is IF loop there in LabVIEW	yes	no	in other name	none of the above	no
The value in the __________ terminal of for loop how many times to repeat the sub diagram.	count term	input terminal	both a and b	none of the above	both a and b
The input terminal of for loop is also known as _________	iteration te	count term	both a and	none of the above	count terminal
The ________terminal gives result for the number of completed iteration	input term	output terminal	count terminal	none of the above	output
The iteration count always starts at______	0	1	2	3	0
the count and iteration terminals are ________ bit integers.	16	32	40	26	32
While loop has how many built in functions	1	2	3	4	1
While loop executes only when the input condition is________	TRUE	FALSE	both a and b	none of the above	TRUE
The block diagram window contains_________	graphical functioning modules	controls and indicators	both a and b	none of the above	graphical source
______ loop executes a sub diagram until a condition met.	For	While	both	none of the above	terminals
__________variables transfer data within a single VI.	local	global	both	none	both

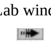
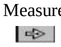
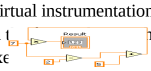
Terminals are entry and exit ports that exchange information between the __________ & _________	controls and indicators	front panel and block diagram	sensors and computer	nodes and wires	front panel and block diagram
The interface is put between _______ & ________	data and c	sensor anc	wires and	none of the above	sensor and computer
The local variables are located on the ______ palette.	structures	Controls p	numeric pa	tools palet	structures palette
Local variable can be placed for a control/constant by _________	Left click	Right clicl	Shortcut n	none of th	Shortcut menu
We can design indicator / control for a terminal by selecting ________ option in short cut menu	Create	replace	both	none of the above	Create
Global variables are built-in _________objects	block diagram	labview	front panel	none of the above	labview
We can use ____________ to access and pass data among several VI that run simultaneously	varibles	data types	constants	both b and c	varibles
Use of ________________avoids unnecessary wires in performing math operations	formula nc	sub vi	Local vari	none of the above	formula node
Use of _______ saves time for creating complex Vis	formula nc	sub vi	Local vari	none of the above	sub vi
__________ can be included in a new VI	Sub VI	Express V	both a and	none of the above	both a and b
Labview offers more _________than standard laboratory	flexibility	linearity	performan	none of the above	flexibility
Labview enables auto indexing in ___________loop	FOR LOC	WHILE LOOP	both	none of the above	FOR LOOP
Labview disable auto indexing in ________ loop	palettes	tools	block diag	none of the above	block diagram
Array is a ________	varibles	data Type	constants	both b and c	data type
The array consist of _______	elements	dimension	both	none	block diagram
Clusters groups data elements of ______ types	different t	similar typ	mixed typ	none	mixed type
The ______ function assembles a cluster from individual elements	array	cluster	bundle	none of th	bundle
The _______ is used to separate data based on their name	Bundle	unbundle	unbundle l	bundle by	unbundle by name
The ________ function assembles data by its name	Bundle	unbundle	unbundle l	bundle by	bundle by name
The size of component in a cluster is __________	variable	constant	fixed	none	fixed
The tool palettes are available on ______	BLOCK D	FRONT P	BOTH A	none of th	both a and b
The cluster functions are located on the _______	cluster pal	control pa	numeric pa	none of th	cluster palettes
The _______is a convenient text_based node you can use to perform mathematical operation on the block diagram.	formula nc	block diag	front pane	none of tl	formula node
For executing ________ express VI can also be used	Structure	Case struc	Formula V	Formula n	Formula VI
___can also perform automatic type conversion	control pa	formula nc	data types	none	formula nodes
Time Delay is an example for ___________	SubVI	Express V	Object	None	Express VI
The formula node syntax is similar to the _____used in text based programming languages.	syntax	do loop	both	none	syntax
Formula in expess VI should end with ______	!	:	;	"	;
The ___________ cluster reports the status,code and source of the error	error clust	disassemb	both	none	error cluster
The functions are located on the _______	numeric pa	control pa	Functions	none	Functions palette
Time Delay VI can be obtained form ________	block diag	Front pane	both	none	block diagram
Flat case structure executes the cases __________	In sequenc	randomly	assigned c	anyway	In sequence
How many cases can be set to a case structure in LabVIEW	5	4	8	n	n

UNIT III

Questions	opt1	opt2	opt3	opt4	opt5	Answer
A virtual plant can be used for ________Tests.	HMI	HIL	MMI	LCD		HIL
The following data communication standard is used for prototyping pl	Ethernet	Profi bus	CAN bus	None of these		Ethernet
Which object of the LabVIEW panel is an input to the code?	Control	Indicator	Display	Monitor		Control
The block diagram executes from	Top to bot	Left to righ	Right to le	Random order		Left to right
FOR loop is located on	Control pa	Indicator pa	Structure p	Sub VI		Structure palette
Which variables can share data with multiple Vis?	Local vari	Global vari	Both	Register variables		Global variable
Which one has the fundamental task of measuring or generating the re	DOS	DAQ	DCS	All of these		DAQ
The representation of signal conditioning device is	SCS	SXS	SCIX	SCXI		SCXI
Give any one application of VI related to signal processing?	Fourier tra	Windowing	power Spe	All of these		All of these
Which toolkit is available for developing control applications?	Bio toolki	PID control	Fuzzy tool	Neural toolkit		PID control toolkit
________loop contains conditional and iteration terminal.	if	while	for	do while		while
______ is similar to record or struct in text based language.	array	string	cluster	file		cluster

_______ is the latest NI-DAQ driver.	DAQ	NI DAQm	NI DAQ	NI DAQdx	NI DAQmx
Energy of the signal is measured by _______express vi.	spectral m	spectral an	spectral tr	signal analysis	spectral measurements
VI finishes execution when____________ button becomes dimmed.	stop	run	step up	step over	run
Shortcut key to clear all broken wires is ________.	cntrl+alt	cntrl+E	cntrl+B	none	cntrl+B
RTSI is________.	real-time s	rest-time s	real-time s	rest-time system interf	real-time system intergration bus
Biosignals can be acquired using __________.	sensors	transducer	both a and	none of the above	both a and b
Which of the following display options are available for strings on the	'\' Codes	Password	Hex	All of the above	All of the above
Which of the following is NOT a traditional debugging feature used tc	Highlight	single Ste	Breakpoin	Stop Values	stop Values
Graphical system design model is used in .	Design pl	Prototypir	Deployme	All the above three p	All the above three phases
Consider the given statements and choose the correct option i) Virtual	(i) & (ii) a	(ii) & (iii) ar	(iii) & (iv)	(i) & (iv) are correct	(iii) & (iv) are correct
Execution of a program in a VI is from______.	right to lef	left to righ	top to bott	bottom to top	right to left
The symbol means	The VI is	VI is runnin	VI runnin	Not executable VI	VI running is a Sub VI
A Block Diagram has a special palette called____.	Functions	Controls pa	Tools pale	Colouring palette	Functions palette
Auto indexing is enabled by default in______.	FOR loop	WHILE loo	Case struc	Formula node	FOR loop
A graph ideal for plotting data with varying time bases	waveform	XY graph	intensity g	digital waveform graph	XY graph
_________ converts a string of valid numeric characters to numeric dat	Format int	Scan from	Concatena	String subset	Scan from string
To write and read data as Text Files end of line character to separate r	\r	\p	\d	\e	\r
Which of the following is NOT a component of an Error Cluster?	Status.	Code.	Source.	VI name	VI name
VISA is given information about physical connections using the	Resource	VISA reso	either a or	None of the above	either a or b
_______ is not a type of signal conditioning.	amplificat	transducer	isolation	data acquisition	data acquisition
In image processing applications HSL stands for	High Strin	Hue Satura	Hue Satur	High String Lightness	Hue Saturation Lightness
Connector pane is visible in the right corner of __________	front pane	icons	block diag	nodes	front panel
A potentiometer is a ______feedback device.	velocity	voltage	current	frequency	voltage
MAX is _________.	Measurem	Measurem	Multi acti	Measurements access	Measurements and automation explorer
Primary module of DAQ system is __________.	signals an	signal cond	API softw	digitizer	signals and sensors
Shortcut key to clear all broken wires is ________.	cntrl+alt	cntrl+E	cntrl+B	none	cntrl+B
The Error list shows all of the following but:	Items with	Errors and	Details abo	Error Codes	Error Codes
The f  following:	Causes the	Causes the	Causes the	Causes the VI to sing	Causes the VI to pause

UNIT IV

Questions	opt1	opt2	opt3	opt4	opt5	Answer
Use of computers (software and hardware) for solving problems related to science and engineering is	Design	Prototype	Scientific	Deploy		Scientific computing
Which is not an Application Software?	LabVIEW	Lab windo	Measurem	Distributed IO		Distributed IO
During the execution of SubVI the run button appears as						
Identify the display style of string that is not available in LabVIEW	Password	Backslash	Normal di	Hex Display		Backslash display
Code is a _________that identifies the error numerically	32-bit sigr	32-bit unsi	16-bit sigr	64-bit signed integer		32-bit signed integer
The GPIB is a digital,______ number of conductor parallel bus	8	16	12	24		24
Identify palettes contains all the functions necessary for your data acquisition Program.	Mathema	FPGA	Signal proc	Measurement I/O		Measurement I/O
The RT module of LabVIEW was originally designed to support	Distribute	Real time	Both a and	Only array operations		Real time applications
Which one is not included in the three step process of scientific computing applications	Data acqu	Data analy	Data visu	Data software		Data software
__________requires predefined hardware components	Data acqu	Data analy	Data visu	Data software		Data acquisition
If you wire 4.51 to count terminal of FOR loop then the loop Iterates _______ times.	4	5	3	does not get executed		4
Unlike graphs, which display an entire waveform that ______ the data already displayed, charts update periodically and _______the data previously displayed.	Erases/up	Erases/upc	Displays,	Displays, maintains the		Erases/updates, maintains the history
_________gives vital information about the measured analog signal since analog signals can take on any value.	Axis	Level	Time	Data		Level
A _______motor converts electronic pulses into proportionate mechanical movement.	DC	Actuator	Stepper	Pulse		Stepper
IMAQ is_________	Image acq	Impulse ac	Informatic	Input acquisition		Image acquisition
Which of the following is not a serial bus?	RS232	USB	GPIB	RS422		GPIB
An example for text based programing is _______________	LabVIEW	Simulink	Python	both a and b		Python
VISA is a ________ API that calls low-level drivers.	Mid level	Low level	High level	Very high level		High level
Motion control is a system used to control ___________.	Physical n	Mechanica	Vertical m	Horizontal movements		Mechanical movements
Virtual instrument is to __________.	repeat th	repeat the f	imitate th	repeat the function of		repeat the function of any instrument
Virtual instrumentation __________.	runs in spe	runs in all	is itself ar	runs without	hardv	runs in all operating system
In t  hat will Result equal when this calculation is exe	55	70	65	Indeterminate		70
To switch between the front panel and block diagram, select Window >> Show Block Diagram/Front Panel, or simply press.	Ctrl-E	Ctrl-Z	Ctrl-Shift-	Ctrl-B		Ctrl-E
The LabVIEW cluster is analogous to a ______in C programming language	record	Subroutine	Statement	Structure		Structure
______type of file format can be used for frequent accessibility	Binary	ASCCI	LVM	String		Binary
VISA stands for	Virtual In	Virtual Inst	Virtual I	Virtual Instrument Soft		Virtual Instrumentation Software Architecture
GPIB is a ________type of communication bus.	Serial	Parallel	Serial & I	USB		Parallel
Which operating systems does not NI-DAQmx base 3.7 support?	Windows	Windows X	Windows	Windows Vista (32-bit		Windows XP (64-bit)
In motion control system application software is divided into _______categories.	2	5	3	4		2
__________ is the most important component of a virtual instrument.	applicatio	application	applicatio	application specific		application development software

In LabVIEW _______ code is translated into executable machine code by interpreting the syntax and by compilation	graphical	textual	visual	audiovisual		graphical
________ is used to extract a floating point number from an input string.	save from	scan from	set from s	none of the above		scan from string
For Loops and while loops can index and accumulate arrays at their boundaries is known as ________ .	auto index	auto aliasi	array inde	array aliasing		auto indexing
GPIB devices communicate with other GPIB devices by sending _______through the interface system.	signals	commands	controls	impulses		commands
_________ update periodically and maintain a history of the data previously stored.	waveform	waveform	intense ch	intense graph		waveform chart
Which of the following does not conform to data flow programming paradigm?	Shift Regi	Tunnels	SubVIs	Local Variables		Local Variables
Which VI memory components are ALWAYS resident for a SubVI?	Data Spac	Front Pan	Block Dia	All the above		Data Space
RTSI is________.	real-time s	rest-time sy	real-time sy	rest-time system interf		real-time system intergration bus
Motion control is a system used to control __________.	Physical	Mechanica	Vertical m	Horizontal moveme		Mechanical movements

UNIT V

Questions	opt1	opt2	opt3	opt4	opt5	Answer
Which of the following does not conform to dataflow programming pa	Shift Regi	Tunnels	SubVIs	Local variables		Local variables
The merit of graphical programming is________.	Syntax ne	Front pane	Logical er	All the above		All the above
The statements which is false is ________.	A SubVI	The color	SubVI m	A SubVI icon can be		A SubVI icon can be edited from the functions palett
A coercion dot indicates that_______ .	The data t	A polymo	A data bu	Data values are being		A data buffer is created to handle data conversion
While loop follows post-test mode and hence ________ .	Both Stop	It execute	A conditio	Auto-indexing is not e		It executes atleast once
The functions assembles cluster elements by their owned labels is ___	Unbundle	Unbundle	Bundle b	Bundle		Bundle by Name
The DAQ that allows measurement of high voltages have_________.	Amplifier	Attenuator	Filter	Isolator		Attenuator
In a general purpose data acquisition system, the module which contro	Analog in	Analog out	Digital I/C	Timing I/O		Digital I/O
In a typical multichannel DAQ system, simultaneous sampling is made	Before A	Before PGA	Before M	Before Amplifier		Before MUX
Among the following communication standards, the one which offers	RS-232	USB	RS-485	GPIB		GPIB
The terminal that returns the number of times the loop has executed is	iteration t	control ter	loop tunn	iteration input termina		iteration terminal
The IF…Else statement of text based programming is done in La	case struct	formula no	both a and	none of the above		both a and b
________ displays an entire waveform that overwrites the data alread	waveform	waveform	intense ch	intense graph		waveform graph
Global variables shares data with ________ VI.	between l	single	multiple	both (B) and (C)		both (B) and (C)
The resolution of 8-bit ADC operating in 10V range is____	0.03906	0.039093	0.03606	0.036093		0.03906
Primary module of DAQ system is _________.	signals an	signal cond	API softw	digitizer		signals and sensors
The baud rate is a measure of how fast data moves between ins	serial com	parallel co	both a and	none of the above		serial communication
What is the output of the Initialize Array function after the following	1-D Array	1-D Array c	1-D Array	1-D Array of {4, 3}		1-D Array of {3, 3, 3, 3}
In what instance would you use the Probe tool rather than Highlight E	To see the	To see the	To look in	To slowdown the VI an		To slowdown the VI and show data values in wires

e