

# PC CONNECTIVITY USING LabVIEW

## Project Report

Submitted by

**Vanitha Seshadri  
R. Vasanth  
V. Tamil Selvan  
A. Subbiah**

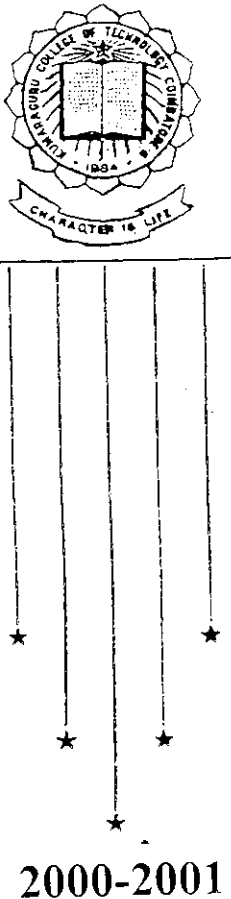
Guided by

**Dr. K.A. Palaniswamy, M.Sc. (Engg.), Ph.D.,  
MISTE, C.Eng. (I), F.I.E.  
Prof. and Head,  
EEE Department.**

Sponsored by

**Premier Polytronics Ltd., Coimbatore**

In partial fulfilment of the requirements  
for the award of the Degree of  
**BACHELOR OF ENGINEERING** in  
**Electrical and Electronics Engineering**  
of the Bharathiar University.



P-520

**Department of Electrical and Electronics Engineering**

**Kumaraguru College of Technology**

Coimbatore – 641 006.

**Department of Electrical and Electronics  
Engineering  
Kumaraguru College of Technology  
Coimbatore – 641 006.**

**CERTIFICATE**

This is to certify that the Report entitled  
**PC CONNECTIVITY USING LabVIEW**  
Has been submitted by

<b>Ms. Vanitha Seshadri</b>	<b>97EEE58</b>
<b>Mr. R. Vasanth</b>	<b>97EEE59</b>
<b>Mr. V. Tamil Selvan</b>	<b>97EEE55</b>
<b>Mr. A. Subbiah</b>	<b>97EEE52</b>

In partial fulfilment of the requirements for the award of  
Bachelor of Engineering  
in the Electrical and Electronics Engineering  
Branch of the Bharathiar University, Coimbatore – 641 046  
during the academic year 2000-2001.

.....  
Guide

.....  
Professor  
Department of Electrical and Electronics Engineering  
**Head of the Department**  
Kumaraguru College of Technology  
Coimbatore - 641 006

Certified that the candidates with University Register Numbers .....  
were examined in project work viva-voce Examination held on .....

.....  
Internal Examiner

.....  
External Examiner

# **PREMIER**

## **CERTIFICATE**

This is to certify that the following students of KUMARAGURU COLLEGE OF TECHNOLOGY, Coimbatore, of branch Electrical & Electronics Engineering

Vanitha Seshadri

R Vasanth

A Subbiah

V Tamilselvan

had undertaken their project entitled "PC CONNECTIVITY USING LabVIEW", from June 2000 to March 2001 at our Industry and have successfully completed it.

Their performance during that period was found to be good. We wish them all success.

P-520

Place : Coimbatore

Date : March 9, 2001

### **Industry Seal**

Premier Polytronics Limited  
304, TRICHY ROAD  
SINGANAILLUR POST  
COIMBATORE-641 005.

  
**K K VENKATARAMAN**  
Vice President - R&D

# ACKNOWLEDGEMENT

The successful completion of a project depends upon team-work and co-operation of various people involved directly or indirectly . We take this opportunity to express our sincere gratitude to all those who have put in a helping hand in our project .

First of all we thank our Principal Dr. K.K. Padmanabhan . B.Sc(Engg) , M.Tech., Ph.D., for providing ample facilities in the college .

We are highly indebted to our project guide Dr.K.A.Palaniswamy , M.Sc(Engg) , Ph.D., FIE , C.Eng(I).. MISTE . Professor & Head of the Department , Electrical and Electronics Engineering , for his guidance and invaluable suggestions , the keen and constant interest evinced by him through out the various phases of the project .

We thank Mr. L. Balraj , Manager-HR & Admn , Premier Polytronics Ltd. , for granting us permission to do the project in the industry .

Our sincere thanks to Mr. K.K.Venkatraman ,Vice President .  
Premier Polytronics Ltd. , for providing us with the resources needed  
to complete the project .

We are highly grateful to our external guide Mr.S. Venkatraman.  
Junior Engineer , R&D Department , Premier Polytronics Ltd. ,  
for the constant help rendered to us in the fine tuning of the project .  
We also thank all the staff of PPL who assisted us during the  
course of the project .

# ABOUT THE COMPANY

The company was established in 1983 with the technical and financial support from *ZELLWEGER USTER*. It belongs to the Premier Mills group which has currently about 1000 spinning mills as its customers. Since 1996, Premier has been independently manufacturing and marketing products worldwide. More than 30% of Premier's sales turnover comes from exports. The company still continues to manufacture splicers with Mesdan, Italy. The complete range of products are:

- Fibre Testing
- Yarn Testing
- Online Monitoring
- Winding

## ACHIEVEMENTS :

Just to give an idea about the company's track record, it has ISO 9001 certification under its belt. It is CE certified for most of

its products . The company has also bagged **THE R&D** awards for the years 1996-97 , 1997-98 . Further more the company has achieved the **EEPC** award for export excellence in the calendar year 1998-99 . The company also boasts of a **FITEI** special export award in the year 1998-99 . It has also won the **Govt . of INDIA Department of Electronics** award for its excellence in the field of electronics .

# SYNOPSIS

Quality of Sliver-eye determines largely the quality of yarn produced . As any meaningful correction of quality deviations being practically possible only in the drawframe Sliver-eye stage , it is imperative that quality control is strictly exercised at this stage : otherwise , tracking and correcting quality deviations in the subsequent processes such as roving and spinning is practically time consuming and it only adds to the production cost . The uncorrected quality deviation will ultimately end up in poor fabric appearances and will show as fabric faults such as streaks and stripes etc .

With the use of high speed Modern Autolevelier Drawframes , even a few minutes production of defective quality would end up in the production of kilometers of yarn . Frequent machine stops will only add to changes in quality levels .



Decisive to the yarn quality characteristics, thus, are the Sliver-eye quality characteristics as mass variations of higher counts, lengths, evenness characteristics, periodic faults and thick and thin places in Sliver materials. These Sliver qualities directly influence count CV %, evenness and seldom occurring faults of yarns.

Online monitoring of Sliver quality thus assumes significance, otherwise much material will be wasted before off-line inspections point out any deviations.

Off-line inspections are only sample-based and hence 100 % detections of deviations are impossible to achieve. Sliver-eye monitors online the quality and production parameters for preparatory machinery such as drawframes and combers. The central idea of the project will be to display data from all Sliver-eye units in a single PC using **NATIONAL INSTRUMENTS** software **LabVIEW** as the front end. The test results are also given in this report.

# CONTENTS

CERTIFICATE	i
ACKNOWLEDGEMENT	iii
ABOUT THE COMPANY	v
SYNOPSIS	vii
CONTENTS	ix
<b>CHAPTERS</b>	
<b>1. SLIVER EYE 5000</b>	i
1.1 AN OVERVIEW	1
1.2 PARTS OF SLIVER-EYE UNIT	2
1.2.1 CONDENSER UNIT	2
1.2.2 PROCESSING AND DISPLAY UNIT	3
1.2.3 POWER SUPPLY UNIT	4
<b>2. INTRODUCTION TO LabVIEW</b>	7
2.1 INTRODUCTION	7
2.2 PLANNING AND DESIGNING PROCESS	8
<b>3. VIRTUAL INSTRUMENTS</b>	11
3.1 FRONT PANEL	11
3.2 BLOCK DIAGRAM	12
3.3 ICON/CONNECTOR	12
<b>4. MENUS</b>	14
4.1 POP-UP MENUS	14
4.2 PULL DOWN MENUS	14
<b>5. ABOUT PALETTES</b>	17
5.1 TOOLS PALETTE	17
5.2 CONTROLS AND FUNCTIONS PALETTE	18
5.2.1 CONTROLS PALETTE	18
5.2.2 FUNCTIONS PALETTE	19

<b>6. VI LIBRARIES</b>	20
6.1 CREATING AND EDITING A VI	21
6.1.1 CREATING A SUB VI	22
6.1.2 EDITING A SUB VI	22
6.2 STRUCTURES	23
6.3 ARRAYS AND GRAPHS	24
6.4 CASE AND SEQUENCE STRUCTURES	24
6.5 CLUSTERS	25
<b>7. SERIAL PORT COMMUNICATION</b>	26
<b>8. SLIVER-EYE CONNECTIVITY</b>	29
<b>9. CONCLUSION</b>	32
<b>10. REFERENCES</b>	33
<b>11. APPENDIX</b>	34

# Chapter – 1

## SLIVER-EYE 5000

### 1.1 AN OVERVIEW

Sliver-eye 5000 , an online quality and production monitoring systems for preparatory machinery such as drawframes and combers . measures and displays the important Sliver-eye quality characteristics such as :

1.  $U_m$  % and  $CV_m$  % : evenness characteristics
2.  $CV$  % [ 0.5m , 1m , 3m & 5m cut lengths ] : cut length variations
3.  $P_m$  % : mass variations
4. Spectrogram : periodic faults
5. Thick & thin places : short term mass variation and helps producing better and consistent Sliver quality .

The production parameters include :

1. length produced
2. machine stops
3. efficiency , etc .

are also being monitored and displayed .

## 1.2 PARTS OF SLIVER-EYE UNIT :

The different parts of the Sliver-eye unit are :

1. Condenser unit
2. Processing & Display unit
3. Power supply & Display unit

### 1.2.1 CONDENSER UNIT :

It basically consists of a strain gauge sensor which is embedded inside a condenser , specifically designed for this purpose . As the Sliver passes through the condenser , the material gets compressed in the D-profile and exerts a force on the sensor which is directly proportional to the volume of the Sliver .

The construction of the condenser is specific to every machine . Inorder to accommodate the various count of the material , the condenser unit is made with various sizes .

As the Sliver material passes through the condenser , fluff and other micro-dust gets deposited near the sensor . Inorder to remove this dust , the sensor is continuously cleaned with a mild blow of compressed air . To facilitate this and increase the efficiency of the system an oil filter with a regulator is provided . The sensor PCB circuit is shown in fig 1.1 .

The sensor PCB consists of:

1. strain gauge
2. precision instrumentation amplifier

### **1.2.2 PROCESSING & DISPLAY UNIT :**

The PDU consists of:

1. Analog conditioning & signal processing unit
2. Production monitoring & storage unit
3. Liquid crystal display (LCD)
4. Keyboard

The input to this unit is got through the sensor cables in the form of analog signals . The 80196 microcontroller is used for the analog conditioning and signal processing . It is used to sense the signals . The 80C320 microcontroller is used for production monitoring and storage .It has 8 bit address line with 40 pins . It helps in interfacing the keyboard and LCD . The LCD is used to display the results of analysis . The results are displayed only if the keyboard is locked .

### **1.2.3 POWER SUPPLY UNIT :**

The power supply unit produces all the required regulated d.c supplies for the functioning of the PDU .

The block diagram of the Sliver-eye unit is shown in fig 1.2 .

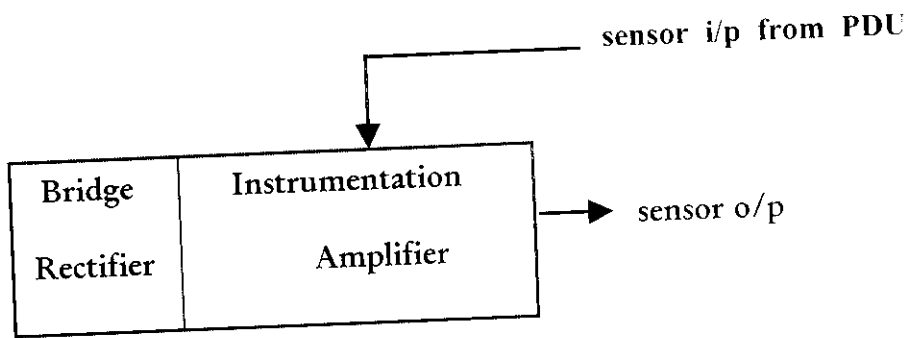


Fig 1.1 – Sensor circuit



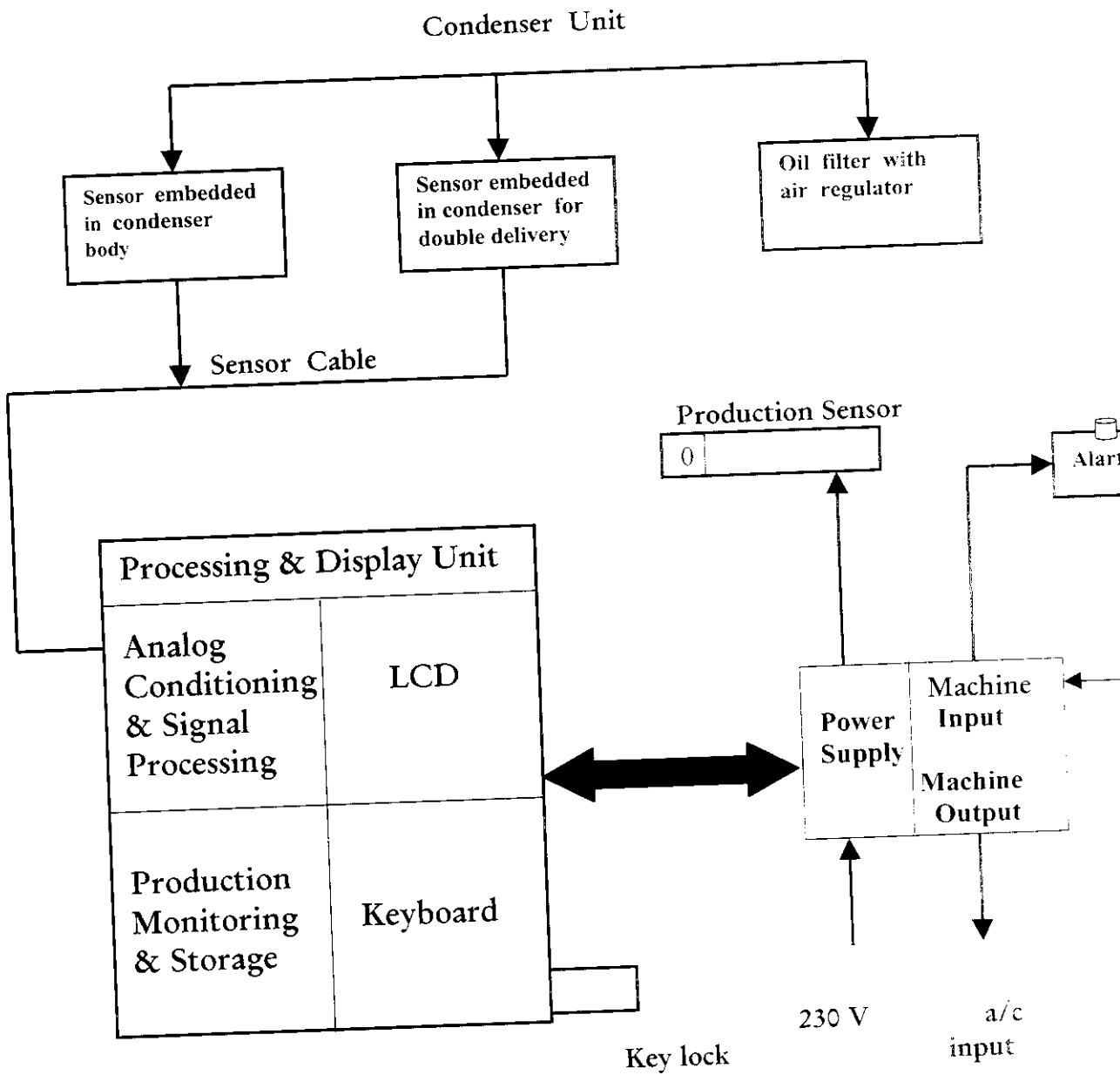
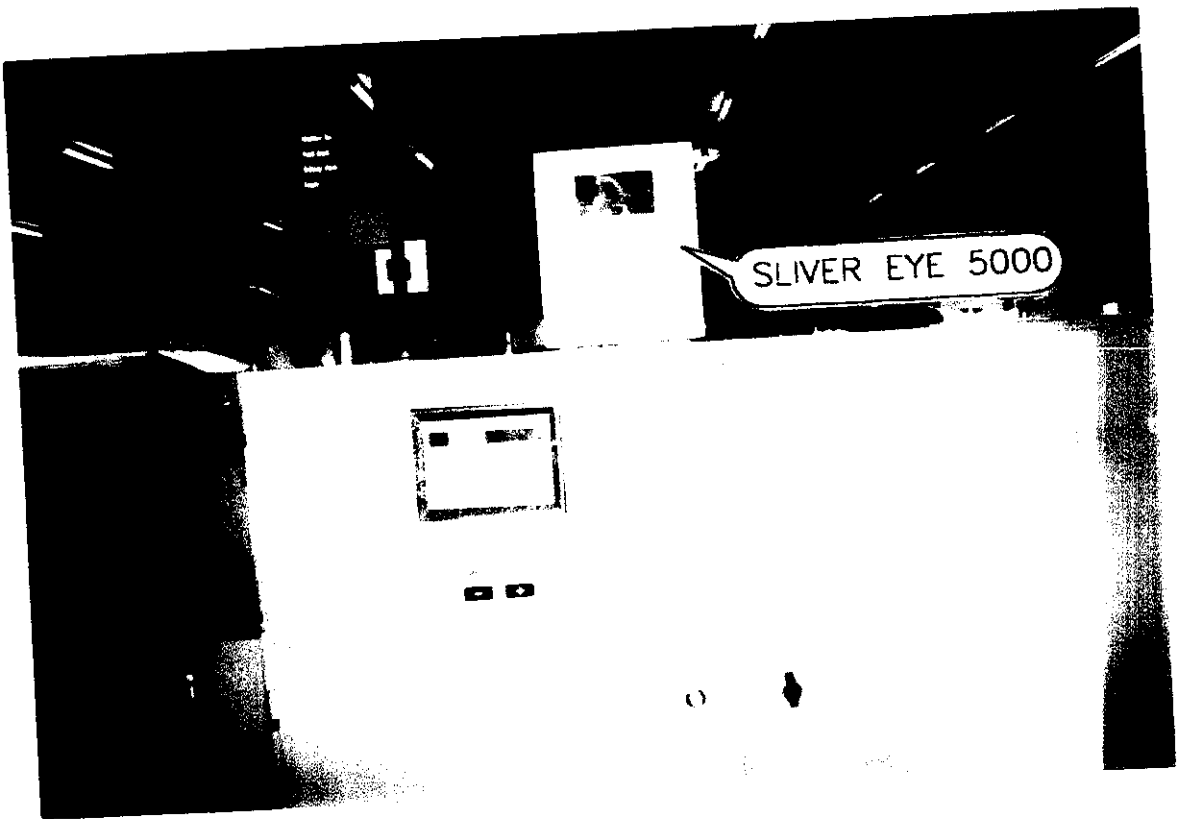


Fig 1.2 – Block diagram of Sliver-eye unit



**Fig. 1.3 SLIVER EYE UNIT**

## Chapter – 2

# INTRODUCTION TO LabVIEW

### 2.1 INTRODUCTION :

LabVIEW ( Laboratory Virtual Instrumentation Engineering Workbench ) is a powerful graphical programming language for virtual instrumentation and analysis . It can be used with the following OS : MICROSOFT , Windows , concurrent Power Max , HP-UX work stations, etc . LabVIEW departs from the sequential nature of traditional programming languages and features a graphical programming environment and all the tools required for data acquisition analysis and presentation . With this graphical programming language called “ G ”, one can program in a block diagram notation , the natural design notation of scientists and engineers . After the creation of the block diagram program , LabVIEW compiles it into a machine code .

LabVIEW integrates data acquisition , analysis and presentation in one system . For acquiring data and controlling instruments , LabVIEW

supports RS-232 / 422 , IEEE 488 and VXI , including Virtual Instrument Software Architecture ( VISA ) functions , as well as plug-in data acquisition ( DAQ ) boards . An instrument library with drives of hundreds of instruments simplifies instrument control applications . For analyzing data , the extensive analysis library contains functions for signal generation , signal processing , filters , windows, statistics, regression , linear algebra and array arithmetic .

Since , LabVIEW is graphical in nature , it is inherently a data presentation package . Using LabVIEW , one can generate charts , graphs and customized user defined graphics .

## **2.2 PLANNING & DESIGN PROCESS :**

To design larger projects using LabVIEW , a top-down approach is usually preferred . The step by step procedure involved in solving the problem is explained by means of a flow diagram . The diagram for the entire process is shown in fig 2.1 .

LabVIEW project development process defines the general characteristics and specifications of the project . We begin by developing sub VI's , ie sub programs which are eventually assembled to form the complete project . This stage represents “ **Bottom-up** ” development period , customer feedback helps in determining new features for next version .

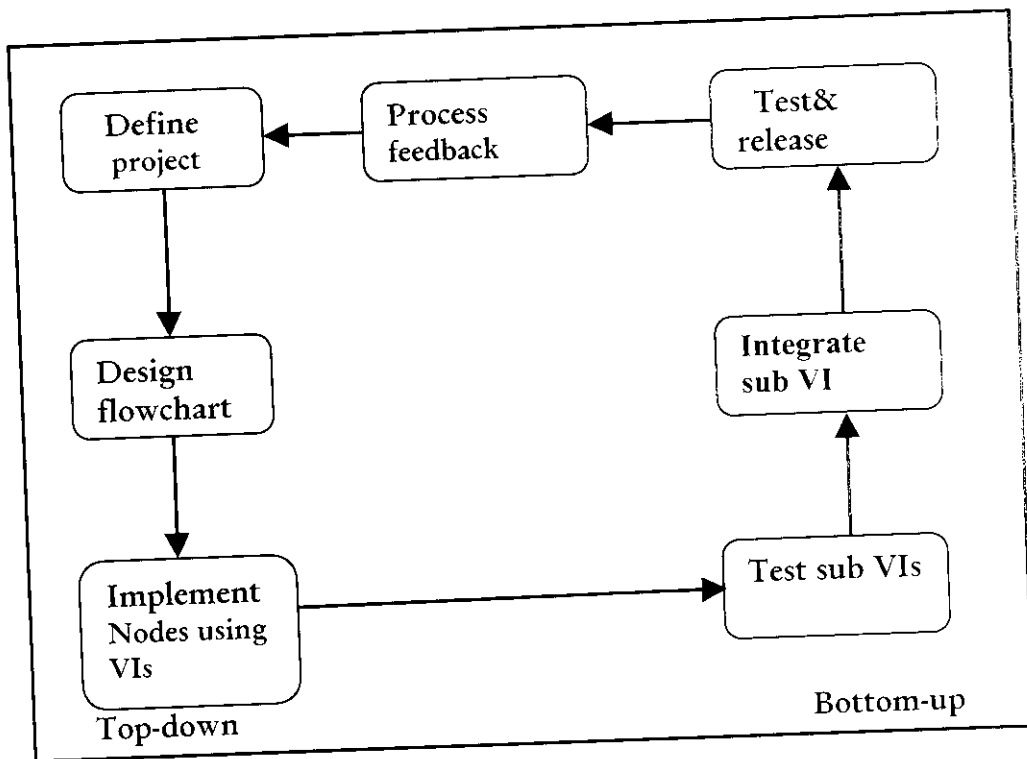


Fig 2.1 – Flow diagram for planning and design process

## Chapter – 3

# VIRTUAL INSTRUMENTS

LabVIEW programs are called **Virtual Instruments** (VI's). VI's have three main parts . They are :

1. Front panel
2. Block diagram
3. Icon / connector

### 3.1 FRONT PANEL :

The front panel is a means of setting input values and viewing outputs from the VI block diagram . As the front panel is analogous to a front panel of a real instrument , the input's are to as **Controls** and the outputs are referred to as **Indicators** . There are variety of controls and indicators , such as knobs , charts . switches . buttons , graphs and so on to make the front panel easily identifiable and understandable .

### **3.2 BLOCK DIAGRAM :**

Each front panel has an accompanying block diagram, which is the VI program. The block diagram is built using the graphical programming language called "G". The block diagram acts as a source code. The components of the block diagram represents nodes, for example: For loops, Case structures, arithmetic functions. They also have constants, functions, sub VI's, structures and wires that carry data from one object to another. The components are wired together to define the flow of data within the block diagram.

### **3.3 ICON / CONNECTOR :**

The icon / connector is used to turn a VI into an object (sub VI). This can be used as a subroutine in the block diagram of other VI's. The connector terminals determine where the inputs and outputs are wired on the icon. The terminals are analogous to



parameters passed to a subroutine . They correspond to the controls and indicators on the VI front panel .

The power of LabVIEW lies in the hierarchical nature of the VI . After a VI is created , it can be used as a sub VI in the block diagram of a higher level VI . There is no limit on the number of layers in the hierarchy . This feature makes the block diagram modular and easy to debug , understand and maintain .

Both the panel and the block diagram windows contain a toolbar of command buttons and status indicators that can be used to control a VI . The toolbar appearing on the block diagram is quite familiar to toolbar appearing on the front panel except for a few additional debugging features .

# Chapter – 4

## MENUS

The LabVIEW has both pop-up and pull down menus .

### 4.1 POP-UP MENUS :

Most commonly used one is the **pop-up menu** . All the objects used to build the VI's are provided with pop-up menus for selection and modification . The process of accessing the pop-up menu is known as **popping up** .

### 4.2 PULL DOWN MENUS :

The menu bar at the top of the LabVIEW screen contains several pull down menus . The pull down menus contains options common to most applications , such as Open , Save , Copy and Paste as well as many others particular to LabVIEW . The pull down menu is shown in fig 4.1 .

The options in the **File** menu are primarily used to open . save , close and print VI's .

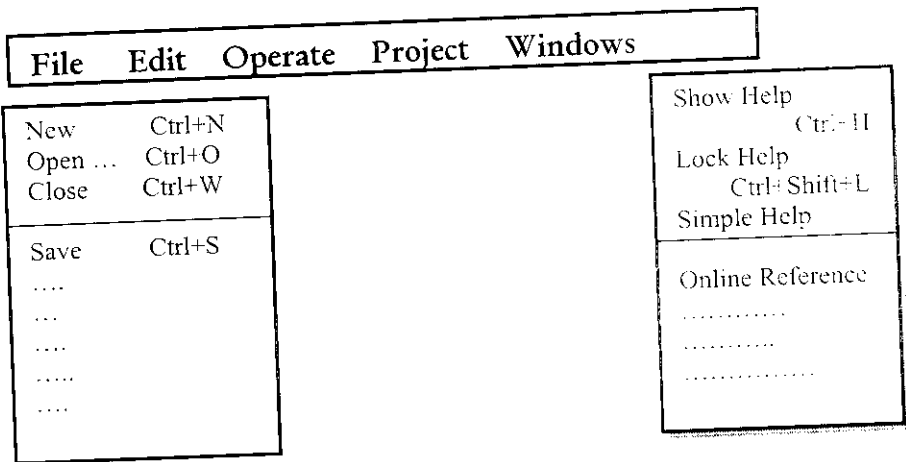
The options in the **Edit** menu are used to modify the front panel and the block diagram objects of a VI. These options are used to manipulate and arrange LabVIEW components to a specific taste.

The commands in the **Operate** menu are used to execute the VI.

The **Project** menu is to obtain additional information regarding the VI, its sub VI's and windows.

The **Windows** menu is used to locate opened windows quickly and to open windows of sub VIs and calling VIs.

The **Help** menu is to view information about the panel or diagram objects, to activate the online reference utilities and to view information about the LabVIEW version number and computer memory.



**Fig 4.1 – Pull down menu**

# Chapter – 5

## ABOUT PALETTES

LabVIEW has both graphical , floating palettes to aid in creating and operating the VI's . The three palettes include the following :

1. Tools palette
2. Controls palette
3. Functions palette

### 5.1 TOOLS PALETTE :

The tools in this palette are used to create , modify and debug VI's . If the tools palette is not visible , select **Show Tools Palette** from the **Windows** menu to display the palette . Once the tool is selected , the mouse cursor takes its shape . The information regarding the tool can be got from the **Help** menu , by placing the cursor on the icon .

## 5.2 CONTROLS & FUNCTIONS PALETTE :

The controls and functions palette consists of top-level icons representing subpalettes, giving access to a full range of available objects that are used in creating a VI. The subpalettes can also be accessed by clicking on the top-level icons. It can be converted to a floating palette that remains on the screen by tacking down the thumbtack on the top left corner.

### 5.2.1 CONTROLS PALETTE :

Controls and indicators needed in the front panel can be added via the **Controls Palette**. Each option in the palette displays a subpalette of available controls and indicators for that selection. If the **Controls Palette** is not visible, select the **Show Controls Palette** from the **Windows** menu. The palette can also be accessed by popping up on an open area in the Panel window. Then it can be tacked into a floating palette by clicking on the pushpin on the top left corner of the palette.

*This palette is available only when the Panel window is active.*

## 5.2.2 FUNCTIONS PALETTE :

The block diagram is built using the **Functions Palette** . Each option in the palette displays a subpalette of top-level icons . If the **Functions Palette** is not visible , the palette is opened by selecting **Show Functions Palette** from the **Windows** menu . Similar to the **Controls Palette** , the **Functions Palette** can be popped up on an open area in the Diagram window . It is then converted into a floating palette by clicking on the pushpin .

*This palette is available only when the Diagram window is active .*

# Chapter – 6

## VI LIBRARIES

The VI's can be loaded and saved to/from a special file called **VI library**. Some of the *advantages* of using the VI library are :

- # 255 characters can be used to name VI's, including “.vi” extension .

- # VI's are compressed to save disk space .

- # Because multiple VI's are in a single file , it is easier to transfer VIs between computers .

The other characteristics of the VI library include :

- # They are not hierarchical in nature , ie . it cannot create VI library within another VI library .

- # Saving and loading VIs to and from the file system is faster than to and from VI libraries .

- # VI libraries can be transferred between platforms .

*Certain operating system-specific VI's are portable between platforms .*



## 6.1 CREATING & EDITING A VI :

The VI is created using the controls and indicators . The block diagram is made up of nodes , terminals and wires . The control terminals have a thicker border than indicator terminals in the block diagram . To change a control to an indicator , pop up on the terminal or object and select **Change to Indicator ( Control )** . A node executes only when data is available at all its input terminals ; the node supplies data to all of its input terminals when it finishes executing and the data passes immediately from the source to destination terminals . Using execution highlighting , single stepping and break points and probes helps debug the VIs easily by tracing the flow of data . LabVIEW single stepping features include :

**Step Into Button :** single steps into sub VIs , loops and so on for debugging .

**Step Over Button :** enables single-stepping mode , bypasses single stepping through node and pauses at the next node in the main VI .

**Step Out Button :** stops single stepping through a node and returns to the main VI .

### 6.1.1 CREATING A SUB VI :

A VI that is used as a sub VI needs an icon to represent it in the block diagram of a calling VI . The sub VI must have a connector with definite number of terminals to pass data to and from the higher level VI . The **Icon Editor** creates icons for the VI . The **Required , Recommended , Optional** classifications assigned to the input terminals make it easier to prevent common wiring mistakes . **Show VI Info** allows to document and maintain pertinent comments about the VI .

### 6.1.2 EDITING A SUB VI :

Using the various tools in the tool palette , the sub VI can be selected , moved , deleted and labeled . For this purpose the positioning tool , wiring tool , labeling tool , etc ., are used .The sub VI can also be resized , font changed , aligning and distribution can also be done .

## 6.2 STRUCTURES :

This controls the flow of data in a VI. LabVIEW has 4 structures to control program flow :- While loop , Case structure , For loop and Sequence structures . The **While** loop repeats a part of the block diagram code multiple times until the condition is true . The **For** loop repeats the block diagram code for a predetermined number of times . The **Waveform** chart is a special numeric indicator that displays one or more charts . It has 3 update modes :

# **Strip chart** : scrolling display

# **Scope chart** : plots data until it reaches the right border , erases plot and retraces the plot from left .

# **Sweep chart** : retracing display with moving vertical line between old and new data .

Shift registers are used to remember stored values from one iteration of a loop to the next . Pop-up on a waveform chart is used to set attributes . Coercion dots appear when LabVIEW is forced to convert a numeric representation of one terminal to match the numeric representation of another terminal .

### 6.3 ARRAYS & GRAPHS :

An array is a collection of data elements of the same type . The data elements can be of any type – numeric , string , Boolean . By selecting an array shell in the window and adding the desired control or indicator to the shell the array can be created . **Polymorphism** is the ability of a function to adjust to input data of different data structures . Waveform graphs and XY graphs display data from arrays . Graphs have many unique features used to customize the plot displays . More than one plot on a graph can be displayed , this becomes a multiplot graph when the array of outputs are wired to the terminal .

### 6.4 CASE & SEQUENCE STRUCTURES :

It is used to control the data flow . LabVIEW depicts both structures like a deck of cards . The **Case** structure is used to branch different diagrams depending on the input to the selection terminal of the case structure . The sub diagrams are placed inside the border of each case structure . The case structure may be Boolean , string or numeric .

The **Sequence** structure is used to execute the diagram in a specific order in the frame. The sequence locals pass values to the frames and they are available in the frames in which they are created. With the formula node, the formulae are directly entered into the block diagram. This is useful when the function equation has many variables or it is complicated. The variable names are case sensitive and each formula statement must end with a “;”.

## **6.5 CLUSTERS :**

A cluster is a data structure that groups data, even of different types. The various front panel controls and indicators needed to be associated with the terminals are grouped under clusters. Objects under clusters must all be controls or all indicators. Using the cluster shell the clusters can be created, while working with these order of components is critical. Error clusters are a powerful method of error handling. These pass information from one VI to another. An error handler at the end of data flow receives the error cluster and displays information on the dialog box.

## Chapter – 7

# SERIAL PORT COMMUNICATION

Serial communications is a popular means of transmitting data between a computer and a peripheral device such as programmable instrument or even another computer . It uses a transmitter and receiver to send and receive data bits at a time . It is popular because computer has large number of serial ports and so no external hardware other than a cable is needed to connect the instrument to the computer . Serial communication requires to specify 4 parameters :

- # Baud rate : it is a measure of how fast data moves between instruments .
- # Data bits : it encodes a character . It is transmitted upside down and backwards .
- # Parity bits : it follows the data bits in the character frame . It is used for error checking .
- # Stop bits : it is the last part of the character frame having 1 , 1.5 , 2 bits .

RS - 232 , RS - 422 , RS - 423 are the different types of serial port communication . In this project RS - 232 are made use of . It is developed by the *Electronics Industries Association* ( EIA ) and other parties specifying the serial interface between the Data Terminal Equipment ( DTE ) and Data Communications Equipment ( DCE ) . It includes electrical signal characteristics ( voltage levels ) , interface mechanical characteristics (connectors ) , functional description of interchange circuits ( functions of each electrical signal ) and some kind of recipes for terminal-to-modem connections . Most frequently encountered version is called **RS - 232C** with high degrees of fidelity . The PC to PC connection is shown in fig 7.1 .

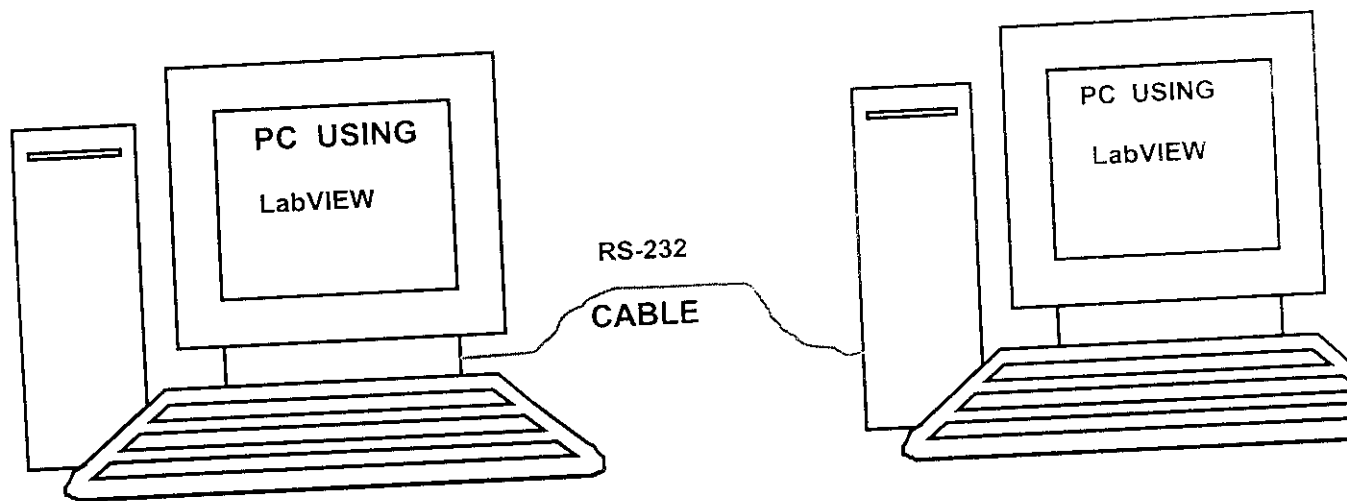


Fig 7.1 – PC to PC connectivity using RS-232



## Chapter – 8

# SLIVER – EYE CONNECTIVITY

Having knowledge about the various topics and the additional features of LabVIEW, the front panel and the various block diagrams associated with the Sliver connectivity are designed as shown in fig 8.1 and 8.2. The software written in LabVIEW consists of a front panel, which displays all the quality and production parameters. For the purpose of testing, a simulator, which simulates the Sliver-eye data, has also been written in LabVIEW as shown fig 8.3 and 8.4.

The simulator program is run on a PC, which is interconnected by RS-232 protocol to another PC from where the user monitors the quality and production parameters. A request pattern is made to send, if it matches the prewritten request format, data is sent to the user's terminal and then the data can be conveniently monitored using the tabular columns shown in the program sheets. The program is written to monitor the data from two Sliver-eye units.

The Quality data table consists of the following

parameters :

- U %
- CV %
- CV % ( 0.5m , 1m , 3m , 5m )
- PM %
- Doff Start Time
- Doff End Time

The Production data table consists of the following

parameters :

- Shift number
- Hank ( for delivery 1 & 2 )
- Hank unit
- Length
- Kilogram ( for delivery 1 & 2 )
- Efficiency
- No. of stops
- Stop duration
- No. of doff

- Doff duration
- Monitored time

The front panel and the block diagrams are shown in fig 8.1 – 8.4 .

The program is an integration of various sub VI's like,

Serial Port Init . vi

Serial Port Write . vi

Bytes at Serial Port . vi

Serial Port Read . vi



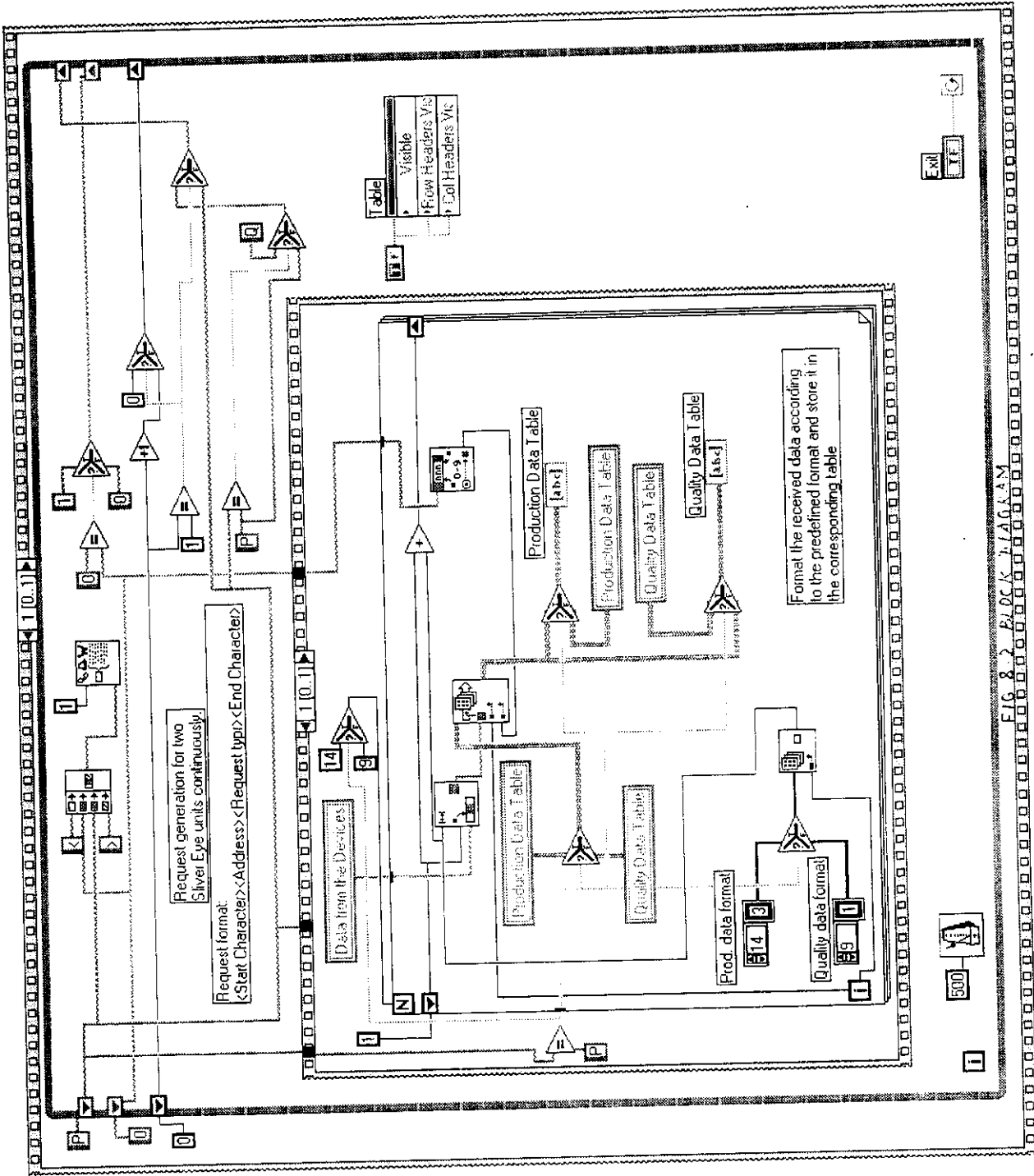
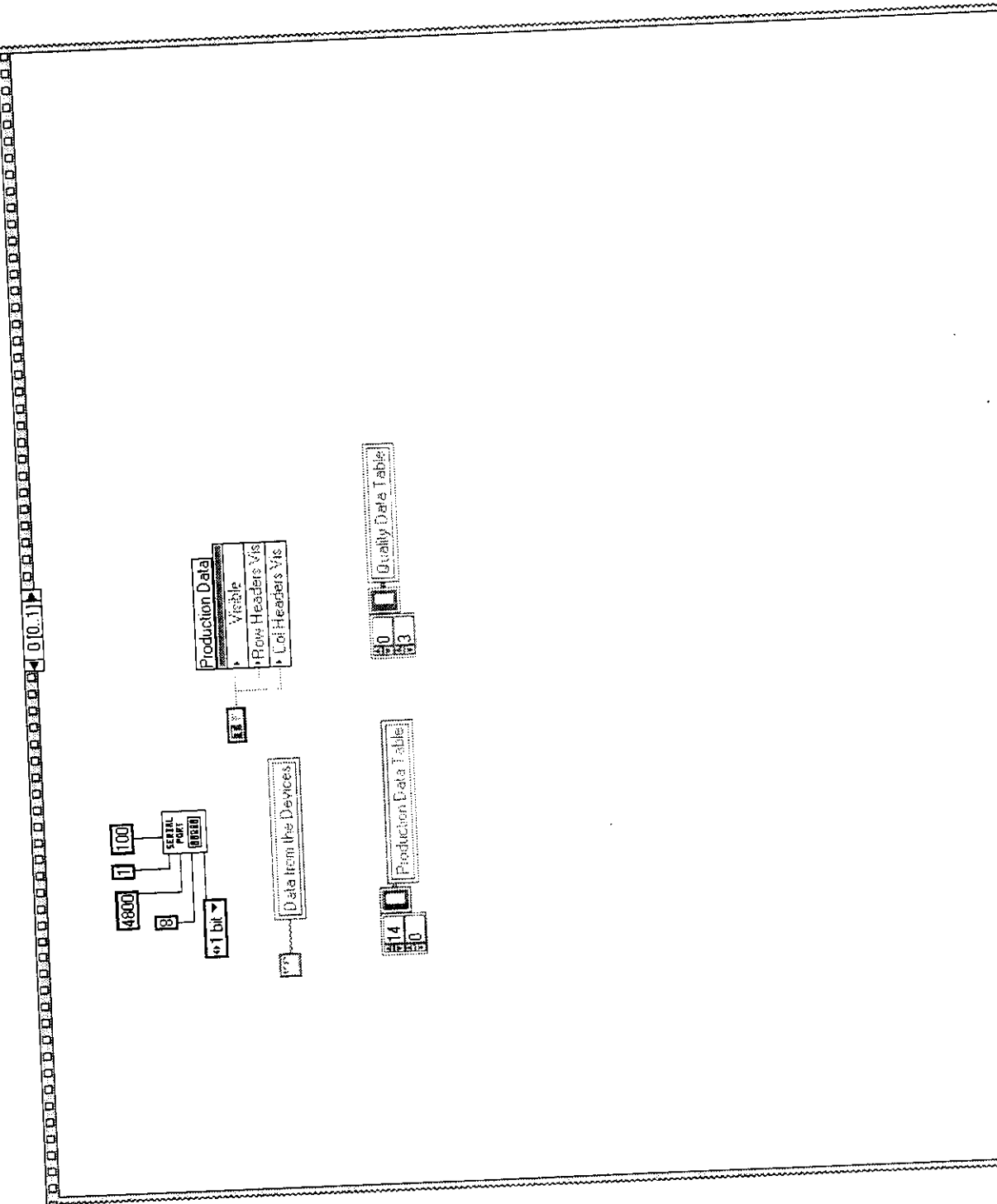
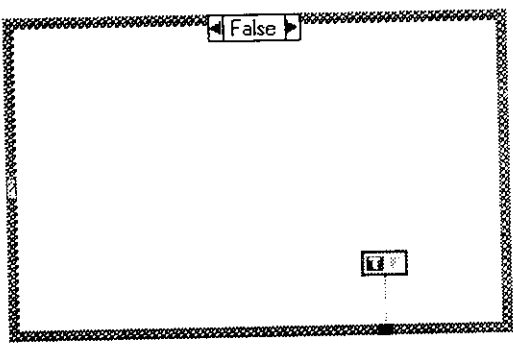
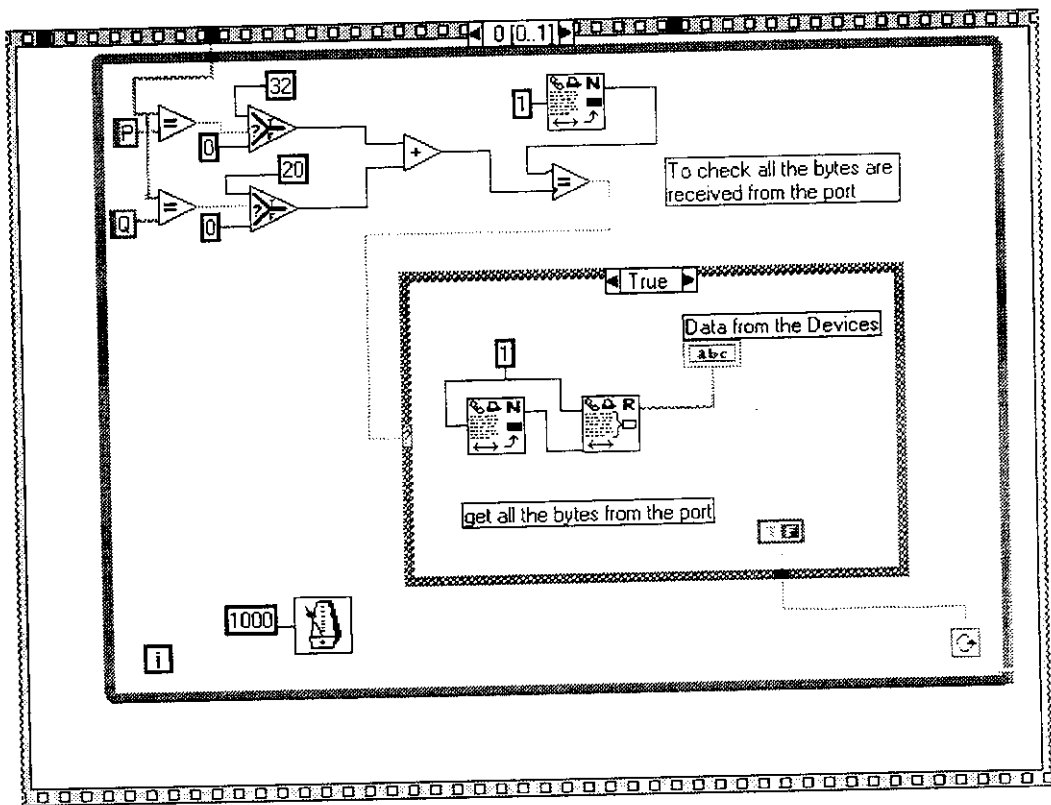


FIG. 8 - BLOCK DIAGRAM





## List of SubVIs



**Serial Port Init.vi**

D:\NATIONAL INSTRUMENTS\LabVIEW\vi.lib\Instr\Serial.lib\Serial Port Init.vi



**Serial Port Write.vi**

D:\NATIONAL INSTRUMENTS\LabVIEW\vi.lib\Instr\Serial.lib\Serial Port Write.vi



**Bytes At Serial Port.vi**

D:\NATIONAL INSTRUMENTS\LabVIEW\vi.lib\Instr\Serial.lib\Bytes At Serial Port.vi



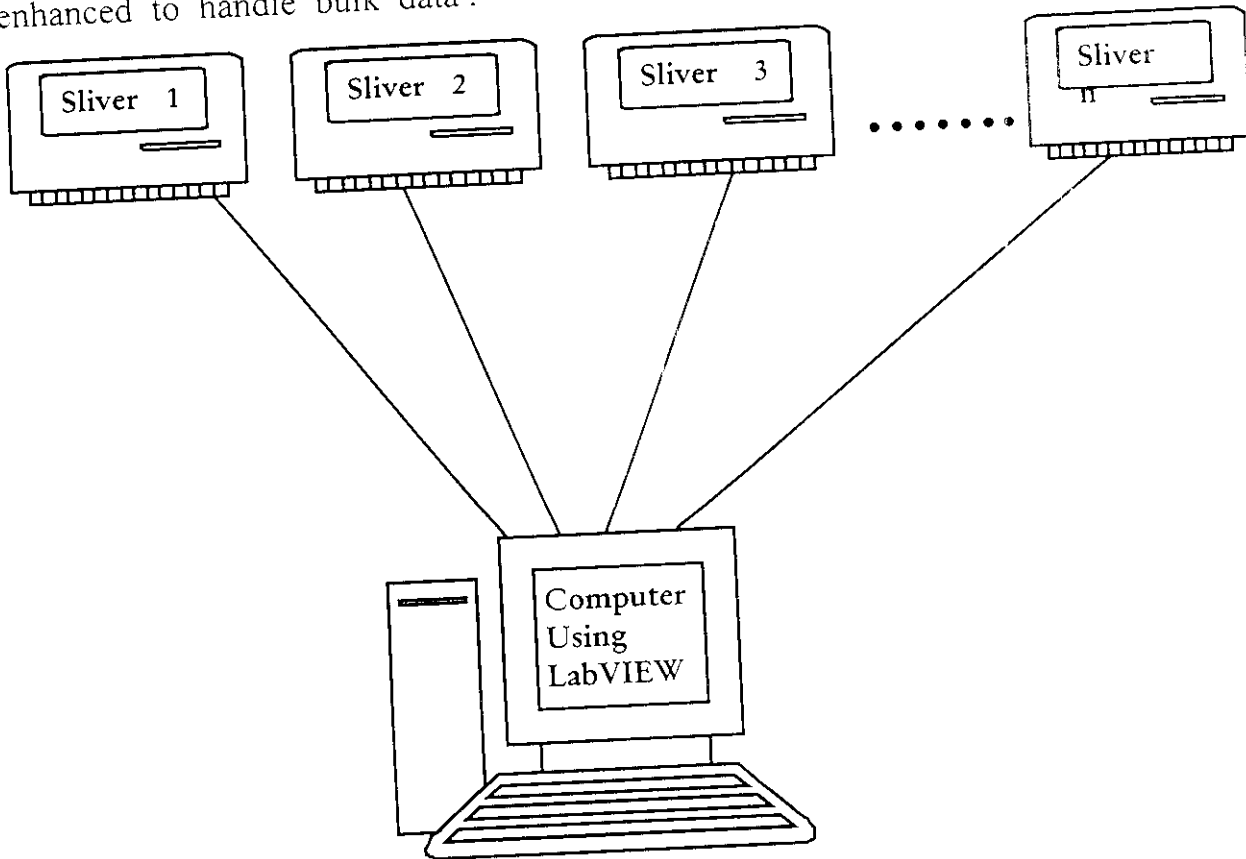
**Serial Port Read.vi**

D:\NATIONAL INSTRUMENTS\LabVIEW\vi.lib\Instr\Serial.lib\Serial Port Read.vi

# Chapter – 9

## CONCLUSION

The online quality monitoring of yarn was designed and tested using LabVIEW. The prime advantage of online quality and production monitoring is that it brings down the production cost considerably. Also the quality of yarn can be monitored from any remote location. As a future expansion of this project many Sliver-eye units can be monitored using a single PC with the same software. Further long time reports can also be generated. The database can be enhanced to handle bulk data.





# Chapter – 10

## REFERENCES

1. LabVIEW Basics – 1 , NATIONAL INSTRUMENTS , TEXAS .
2. LabVIEW Basics – 2 , NATIONAL INSTRUMENTS , TEXAS .
3. On the Internet , website is [www.NI.com](http://www.NI.com) .

You use the Help menu to view information about panel or diagram objects, to activate the online reference utilities, and to view information about your LabVIEW version number and computer memory.

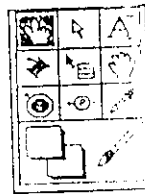
Help		
Show Help	Ctrl+H	Activates the Help window.
Lock Help	Ctrl+Shift+L	Locks the Help window screen on one subject.
✓ Simple Help		Enables the simple diagram view in the Help window.
-----		
Online Reference...	Ctrl+?	Opens the LabVIEW Online Reference utility.
Online Help for Untitled ?		Displays online help for the current VI.
-----		
Explain Error...		Shows error cluster values and message for debugging.
Internet Links	▶	Opens direct Internet links to National Instruments.
Online Tutorial...		Launches the interactive online tutorial.
Search Examples...		Opens a utility that searches through all example VIs.
Technical Support Form...		Calls the technical support form to log problems.
lvdaq1...		
-----		
About LabVIEW...		Shows you: LabVIEW version and memory information.



## Palettes


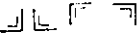









LabVIEW has graphical, floating palettes to aid in creating and operating VIs. The three palettes include the Tools, Controls, and Functions palettes.

## Tools Palette

You can create, modify, and debug VIs using the tools located in the floating Tools palette. If the Tools palette is not visible, select **Show Tools Palette** from the Windows menu to display the palette. After you select a tool from this menu, the mouse cursor takes its shape. (*Windows*—You also can access the Tools palette by pressing <shift> and the right mouse button. *Macintosh*—Access the Tools palette by pressing <command-shift> and the mouse button.) Place any tool found in the Tools palette over a subVI or function icon to display information pertaining to that subVI or function in the Help window. You first must select Show Help from the Help menu.



-  Operating tool. Use the Operating tool to manipulate the values of front panel controls and indicators.
-  The tool changes to the icon shown at left when it passes over a text-based control, such as a digital or string control.

-  Positioning tool. Use the Positioning tool to select, move, or resize objects.
-  The tool changes to the icon shown at left, when it passes over a corner of a resizable object.
-  Labeling tool. Use the Labeling tool to enter text into labels.
-  The Labeling tool changes to the icon shown at left when you create free labels.
-  Wiring tool. Use the Wiring tool to wire objects together on the block diagram. Place the Wiring tool over a wire to display the data type of the wire in the Help window. You first must select Show Help from the Help menu.
-  Object pop-up menu tool. Use the object pop-up menu tool to pop up on an object's pop-up menu with the left mouse button.
-  Scrolling tool. Use the Scrolling tool to scroll through windows without using scrollbars.
-  Breakpoint tool. Use the Breakpoint tool to set breakpoints on VIs, functions, and structures.
-  Probe tool. Use the Probe tool to create probes on wires in the block diagram.
-  Color Copy tool. Use the Color Copy tool to copy colors for pasting with the Coloring tool.
-  Coloring tool. Use the Coloring tool to color an object. It also displays the foreground and background of the object.

## Controls and Functions Palettes

The Controls and Functions palettes consist of top-level icons representing subpalettes, giving access to a full range of available objects that you can use in creating a VI. You can access the subpalettes by clicking on the top-level icon. You also can convert the subpalette to a floating palette that remains on your screen by tacking down the thumbtack at top left corner of the subpalette.

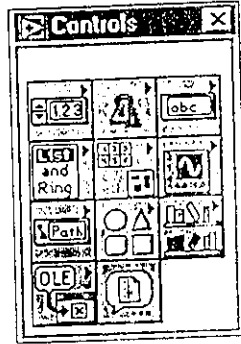
## Controls Palette


You add controls and indicators to the front panel via the Controls palette. Each option in the palette displays a subpalette of available controls and indicators for that selection. If the Controls palette is not visible, you can open the palette by selecting **Show Controls Palette** from the Windows


menu. You also can access the Controls palette by popping up on an open area in the Panel window. Then you can tack down the Controls palette into a floating palette by clicking on the pushpin on the top left corner of the palette.


Note


*The Controls palette is available only when the Panel window is active.*





- 


**Numeric subpalette.** Consists of controls and indicators for numeric data.
- 


**Boolean subpalette.** Consists of controls and indicators for Boolean values.
- 

**String subpalette.** Consists of controls and indicators for strings and tables.
- 


**List & Ring subpalette.** Consists of controls and indicators for menu rings and listboxes.
- 


**Array & Cluster subpalette.** Consists of controls and indicators that group sets of data types.
- 


**Graph subpalette.** Consists of indicators to plot data in graphs or real-time charts.
- 

**Path & Refnum subpalette.** Consists of controls and indicators for file paths and refnums.
- 

**Decorations subpalette.** Consists of graphical objects for customizing front panel displays.

 User Controls subpalette. Location for placing users' controls.

 ActiveX subpalette. Consists of controls and indicators that allow ActiveX Container capability (PC only).

 Select a Control subpalette. Displays a dialog box to load custom controls.

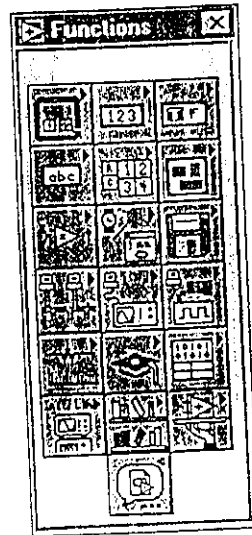
## Functions Palette


You build the block diagram with the Functions palette. Each option in the palette displays a subpalette of top-level icons. If the **Functions palette** is not visible, you can open the palette by selecting **Show Functions Palette** from the **Windows** menu. You access the **Functions palette** by popping up on an open area in the Diagram window. Then you can convert the **Functions palette** to a floating palette by clicking on the pushpin.

















Note




*The Functions palette is available only when the Diagram window is active.*





 Structures subpalette. Consists of program control structures such as For Loops.


 Numeric subpalette. Consists of arithmetic, trigonometric, logarithmic, and numeric functions.

-  Boolean subpalette. Consists of logical and Boolean functions.
-  String subpalette. Consists of functions to manipulate strings.
-  Array subpalette. Consists of functions to process arrays.
-  Cluster subpalette. Consists of functions to process clusters.
-  Comparison subpalette. Consists of functions to compare numbers, Booleans, and strings.
-  Time & Dialog subpalette. Consists of functions for dialog windows, timing, and error handling.
-  File I/O subpalette. Consists of functions and VIs for File I/O.
-  Communication subpalette. Consists of networking VIs for TCP, DDE, Apple Events, and OLE.
-  Instrument I/O subpalette. Consists of VIs for GPIB, serial, and VISA instrument control.
-  Data Acquisition subpalette. Consists of VIs for plug-in data acquisition boards.
-  Analysis subpalette. Consists of data analysis VIs.
-  Tutorial subpalette. Consists of VIs used in the LabVIEW tutorial.
-  Advanced subpalette. Consists of miscellaneous functions such as the call library function, memory functions, data manipulation, and so on.

-  Application Control subpalette. Consists of functions and VIs for LabVIEW server capability, programmatic printing, changing LabVIEW menus, showing the Help window, and stopping or exiting LabVIEW.
-  Select a VI... subpalette. Consists of a dialog box for inserting subVIs into the current VI.
-  User Libraries subpalette. Location for placing users' VIs. Any VIs or VI libraries you place into the user .lib directory will appear in this subpalette after you relaunch LabVIEW.

 Note

The Basics Course subpalette  (User Libraries » Basics Course) consists of VIs used in the LabVIEW Basics I Course."

-  Instrument Drivers subpalette. Location for placing instrument driver VIs. Any VIs or VI libraries you place in to the instr.lib directory will appear in this subpalette after you relaunch LabVIEW.

## VI Libraries

You can load and save VIs to/from a special file called a *VI library* (normally a file with the .llb extension). The `BASICS.LLB` library is an example of a VI library. Advantages to using VI libraries include:

- With VI libraries, you can use up to 255 characters to name your VIs, including the .vi extension.
- VI libraries compress VIs to save disk space (they are decompressed at load time).
- Because multiple VIs are in a single file, it is easier to transfer VIs between computers.

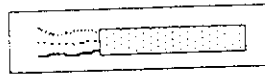
Other VI library characteristics include:

- VI libraries are not hierarchal in nature. That is, you cannot create a VI library within another VI library.
- Saving and loading VIs to and from the file system is faster than to and from VI libraries.

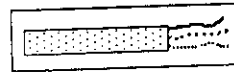
## A. Clusters

A cluster is a data structure that combines one or more data components into a new data type. The components that form a cluster may even have different data types. For example, you can combine Boolean, string, and integer data types into a new data type. A cluster is analogous to a *record* in Pascal or a *struct* in C.

On the block diagram, a wire that carries the data stored in a cluster may be thought of as a bundle of smaller wires, much like a telephone cable. Each wire in the cable represents a different component of the cluster. Because a cluster constitutes only one “wire” in the block diagram, clusters reduce wire clutter and the number of connector terminals that subVIs need.

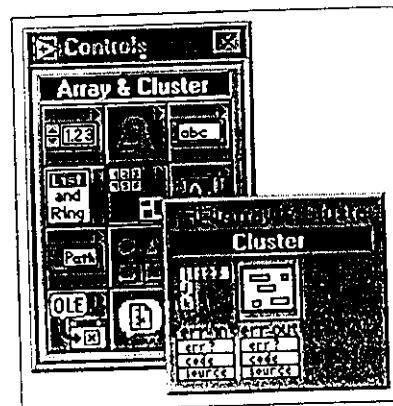


You “unbundle” the cluster in the diagram to access its components. You may think of unbundling a cluster as unwrapping a telephone cable and accessing the individual wires within the cable.



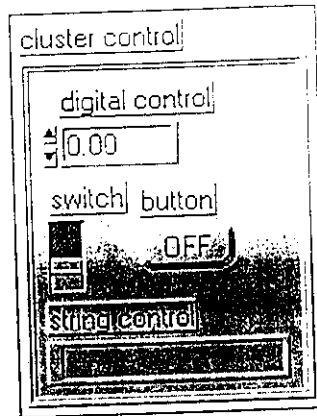
### Creating Cluster Controls and Indicators On the Front Panel

You create cluster controls and indicators on a VI’s front panel by placing a *cluster shell* in the Panel window. To place an empty cluster shell on the panel, choose **Array & Cluster » Cluster** from the **Controls** palette. Then, click in the Panel window to place the cluster. You can adjust the size of the cluster shell by holding down the mouse button and dragging the cursor when placing the cluster shell on the panel.



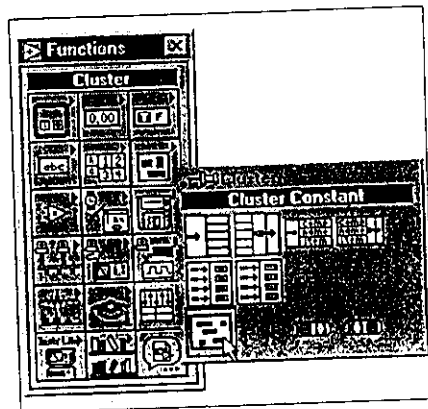


You can place any objects inside the cluster that you normally place in the Panel window. You can deposit objects directly inside the cluster by popping up inside the cluster, or you can drag an object into a cluster. *Objects inside a cluster must be all controls or all indicators.* You cannot combine both controls and indicators inside the same cluster. For example, if you drop an indicator into a cluster containing controls, the indicator changes to a control. The cluster assumes the data direction (control or indicator) of the first object you place inside the cluster. A cluster with four controls is shown below.



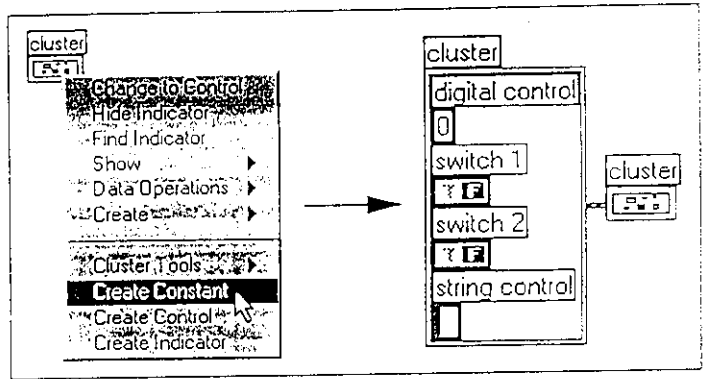
## Creating Cluster Constants on the Block Diagram

To create a cluster constant on the block diagram, you can use the same technique as you used on the front panel. From the Diagram window, choose **Cluster » Cluster Constant** from the **Functions** palette to create the cluster shell. Then, place other constants of the appropriate data type within the cluster shell.



If you have a cluster control or indicator on the front panel, and would like to create a cluster constant containing the same components in the diagram,

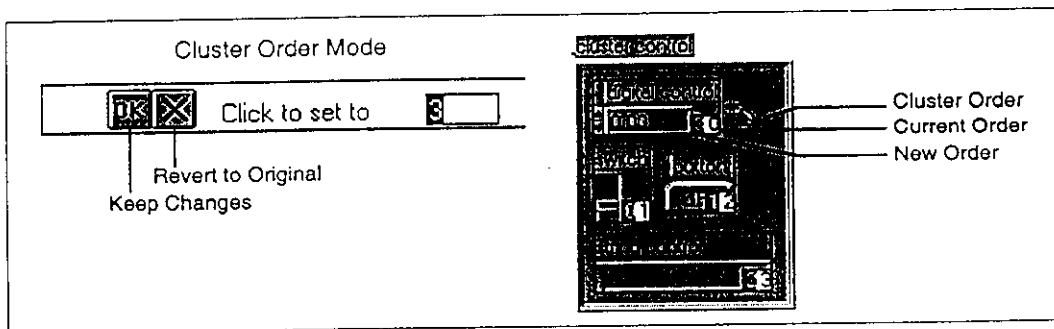
you can pop up on its terminal and select **Create Constant** from the pop-up menu. This technique is a great time-saver. If you use this method on a cluster indicator, the constant is wired to the indicator automatically.



## Cluster Order

When LabVIEW manipulates clusters of data, not only are the data types of the individual components within the cluster important, but also the order of the components in the cluster. Cluster components have a logical order unrelated to their position within the shell. The first object placed in the cluster shell is component 0, the second is component 1, and so on. If you delete a component, the order adjusts automatically.

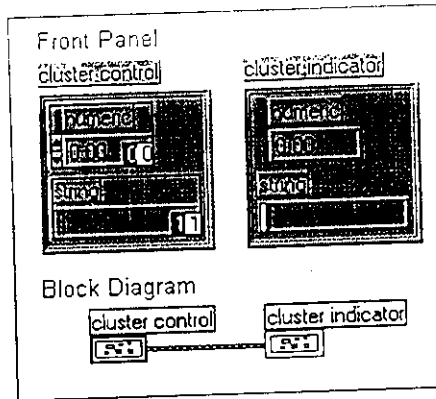
You can change the order of the objects within the cluster by popping up on the cluster *border* and choosing **Cluster Order...** from the pop-up menu. A new set of buttons replaces the toolbar, and the cluster appearance changes as shown below. The white box on each component shows its current place in the cluster order. The black box shows a component's new place in the order.



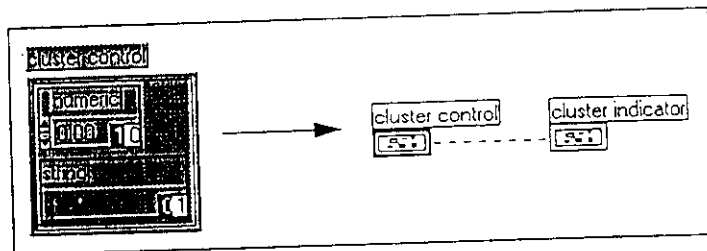


To set a cluster component to a particular index in the cluster order, first type the desired order number into the "Click to set to" field. Then click on the desired component. You will notice that the component's cluster order index changes. You will also notice that the cluster order indices of the other components adjust automatically. To save your changes, click on the OK button in the palette. To revert to the original settings, click on the Revert to Original button. You use this technique to set the cluster order for constants on both the front panel and the block diagram.

The example shown below illustrates the importance of cluster order. The front panel contains two simple clusters. In the first cluster, component 0 is a numeric control, and component 1 is a string control. In the second cluster, component 0 is a numeric indicator, and component 1 is a string indicator. The cluster control wires to the cluster indicator on the block diagram.

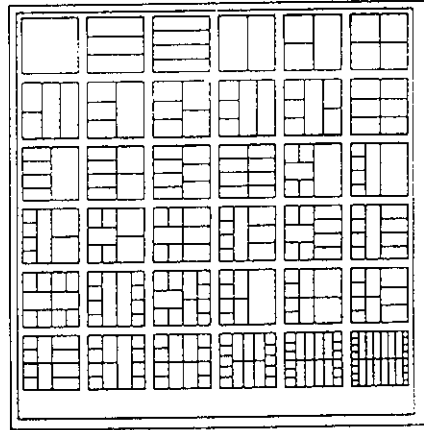


However, if you change the cluster order of the indicator so the string indicator is component 0 and the numeric is component 1, the wire connecting the control to the indicator is broken. If you try to run the VI, you get an error message stating that there is a type conflict because the data types do not match.

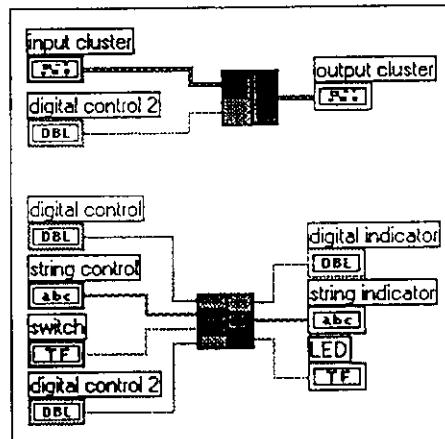


## Using Clusters to Pass Data to and from SubVIs

As shown in the figure below, the connector pane of a VI can have a maximum of 28 terminals. When you use a connector pane that has a large number of terminals, the terminals are very small. With such small terminals, wiring errors are more likely.



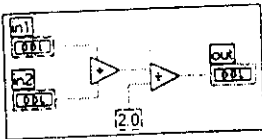
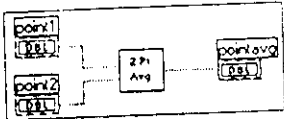
By bundling a number of controls into a cluster, and passing the cluster to the subVI, you can overcome the 28-terminal limit of the connector pane. One cluster control uses one terminal on the connector pane, but that cluster can contain several controls. Similarly, one terminal assigned to a cluster indicator can pass several outputs from the subVI. Because your subVI uses clusters containing several items each, you can use fewer, and therefore larger, terminals on the connector pane. This will make for cleaner wiring on the diagram.



## A. Basic Ideas

The key to creating LabVIEW applications is understanding and using the hierarchical nature of the VI. That is, after you create a VI, you can use it as a subVI in the block diagram of a higher-level VI. If a block diagram has a large number of icons, you can group them into a lower-level VI to maintain the simplicity of the block diagram. This modular approach makes applications easy to debug, understand, and maintain. You can learn more about application development in the LabVIEW Basics II course.

SubVIs are similar to functions or subroutines in a conventional programming language. The following pseudo-code and block diagram demonstrate the analogy between subVIs and subroutines.

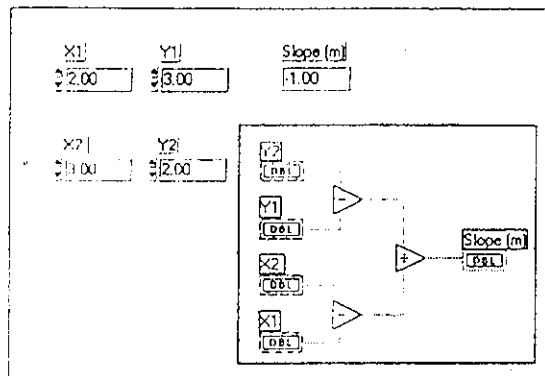
<p style="text-align: center;"><b>Function Code</b></p> <pre>function average (in1, in2, out) {   out = (in1 + in2) / 2.0; }</pre>	<p style="text-align: center;"><b>Calling Program Code</b></p> <pre>main {   average (point1, point2,           pointavg); }</pre>
<p style="text-align: center;"><b>SubVI Block Diagram</b></p> 	<p style="text-align: center;"><b>Calling VI Block Diagram</b></p> 

## B. Creating the Icon and Connector

A VI that you use as a subVI needs an icon to represent it in the block diagram of a calling VI. The subVI also must have a connector with terminals to pass data to and from the higher-level VI.

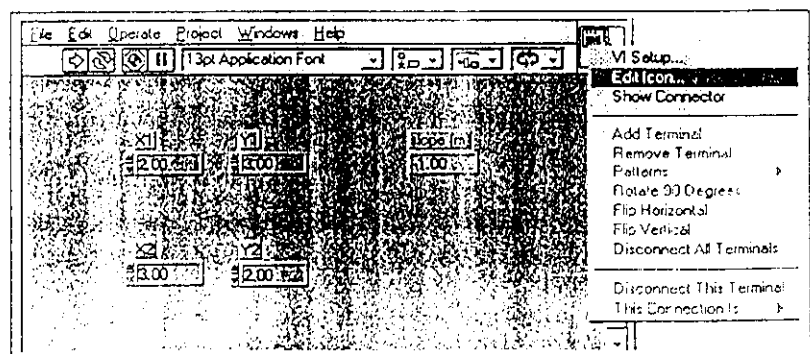
### Example—Slope Function

As an example, consider a VI that calculates the slope of two coordinates. The front panel and the block diagram for the VI are shown below. To use this VI as a subVI, you must create an icon and a connector for it.



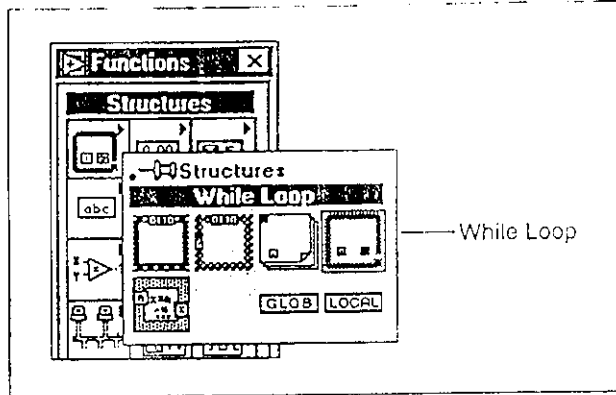
### Icon

Every VI has a default icon displayed in the upper-right corner of the Panel and Diagram windows. For VIs, the default icon is a picture of the LabVIEW logo and a number indicating how many new VIs you have opened since launching LabVIEW. You use the Icon Editor to customize the icon by turning individual pixels on and off. To activate the Icon Editor, pop up on the default icon in the top right corner of the Panel or Diagram window and select **Edit Icon** as shown below.

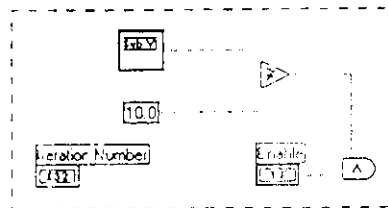


## A. While Loop

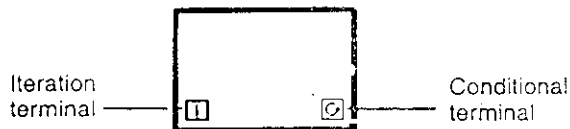
A *While Loop* repeats part of your block diagram code multiple times. You place a While Loop in the block diagram by first selecting it from the Structures subpalette of the Functions palette.



Then use the mouse cursor to click-and-drag a selection area around the code you want to repeat. When you release the mouse button, a While Loop boundary *encloses* the code you have selected as shown below.



The completed While Loop is a resizable box. You can add additional block diagram elements to the While Loop by dragging and dropping them inside the boundary with the mouse.



  
Conditional  
terminal

  
Iteration  
terminal

The VI repeats the code inside the While Loop until the Boolean value passed to the *conditional terminal* (an input terminal) is **FALSE**. The VI checks the conditional terminal at the *end* of each iteration; therefore, the **While Loop always executes once**. The *iteration terminal* is a numeric output terminal that contains the number of times the loop has executed, starting at zero. (That is, during the first execution of the loop, the iteration terminal contains the number zero.)

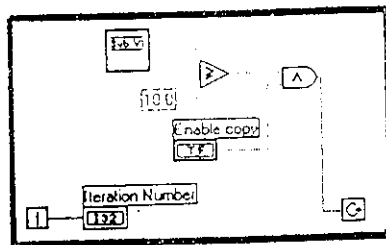
A While Loop is equivalent to the following pseudo-code:

Do

Execute Diagram Inside the Loop (which sets the condition)

While the condition is TRUE

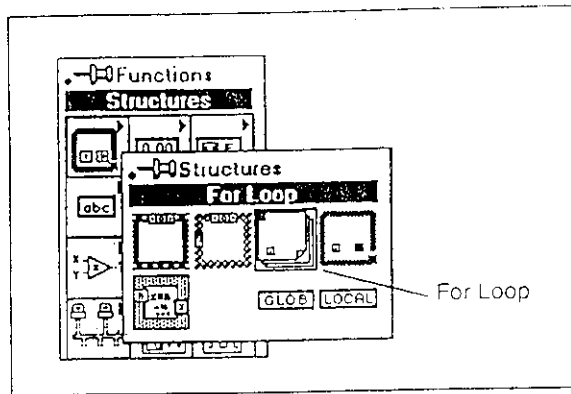
In the example below, the While Loop executes until the value output from the subVI is less than 10 or the Enable Boolean is FALSE. (The **And** function outputs a TRUE only if both inputs are TRUE; otherwise, it outputs a FALSE.)





## D. For Loop

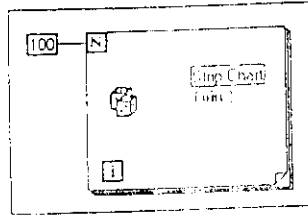
A For Loop repeats part of your block diagram code a predetermined number of times. You select a For Loop from the Structures subpalette of the Functions palette, and then enclose the code you want to repeat in the For Loop boundary. A For Loop (shown below) is a resizable box. The For Loop has two terminals: the count terminal (an input terminal) and the iteration terminal (an output terminal). The count terminal specifies the number of times to execute the loop. The iteration terminal contains the number of times the loop has executed.



The difference between the For Loop and the While Loop is that the For Loop executes a *predetermined* number of times. A While Loop stops repeating the code it encloses only if the value at the conditional terminal becomes FALSE. The For Loop is equivalent to the following pseudo-code:

```
For i = 0 to N-1
  Execute Diagram Inside The Loop
```

The example below shows a For Loop that generates 100 random numbers and displays the points on a waveform chart.



## Numeric Conversion

Until now, all the numeric controls and indicators you have used are double-precision floating-point numbers. LabVIEW, however, can represent numerics as integers (byte, word, or long) or floating-point numbers (single, double, or extended precision). If you wire together two terminals that are of different data types, LabVIEW will convert one of the terminals to the same representation as the other terminal. As a reminder, LabVIEW places a dot, called a coercion dot, on the terminal where the conversion takes place.

**N**  
For Loop  
count terminal

For example, consider the For Loop count terminal. The terminal representation is long integer. If you wire a double-precision floating-point number to the count terminal, LabVIEW converts the number to a long integer. Notice the gray dot in the count terminal of the first For Loop.

