CHAPTER 3

Section 3.1

SALIENT FEATURES OF VISUAL BASIC

Visual Basic is a high level programming language which evolved from the earlier DOS version called BASIC. BASIC means Beginners' All-purpose Symbolic Instruction Code. Visual Basic (VB) was developed from the BASIC programming language. In the 1970s, Microsoft started developing ROM-based interpreted BASIC for the early microprocessor-based computers. Later on, Microsoft Corporation created the enhanced version of BASIC called Visual Basic for Windows. It is a very easy programming language to learn. The code looks a lot like English Language. Different software companies produced different versions of Basic, such as Microsoft QBASIC, QUICKBASIC, GWBASIC, IBM BASICA and so on. However, people prefer to use Microsoft Visual Basic today, as it is a well developed programming language and supporting resources are available everywhere. Now, there are many versions of VB exist in the market, the most popular one and still widely used by many VB programmers is none other than Visual Basic 6. We also have VB.net, VB2005, VB2008 and the latest VB2010. Both VB2008 and VB2010 are fully object oriented programming (OOP) language.

Visual Basic 6 is an ideal programming language for developing sophisticated professional applications for Microsoft windows. It makes use of Graphical User Interface for creating robust and powerful applications. The Graphical User Interface as the name suggests, uses illustrations for text, which enable users to interact with an application. This feature makes it easier to comprehend things in a quicker and easier way. Coding in GUI environment is quite a transition to traditional, linear programming methods where the user
is guided through a linear path of execution and is limited to small set of operations. In GUI environment, the number of options open to the user is much greater, allowing more freedom to the user and developer. Features such as easier comprehension, user-friendliness, faster application development and many other aspects such as introduction to ActiveX technology and Internet features make Visual Basic an interesting tool to work with.

Visual Basic is a visual and events driven programming language. This is called because programming is done in a graphical environment unlike the previous version of BASIC where programming is done in a text only environment and executed sequentially in order to control the user interface. Visual Basic enables the user to design the user interface quickly by drawing and arranging the user elements. Due to this spent time is saved for the repetitive task. However, In VB, you just need to drag and drop any graphical object anywhere on the form, and you can change its color any time using the properties windows.

On the other hand, because the user may click on certain object randomly, so each object has to be programmed independently to be able to response to those actions (events). Therefore, a VB Program is made up of many subprograms, each has its own program code, and each can be executed independently and at the same time each can be linked together in one way or another.

The salient features of Visual Basic 6 are

- Full set of objects to draw the application
- Lots of icons and pictures for use
- Response to mouse and keyboard actions
CHAPTER 3

- Clipboard and printer access
- Fully array of mathematical, string handling and graphics functions
- Can handle fixed and dynamic variable and control arrays
- Sequential and random access file support
- Useful debugger and error-handling facilities
- Powerful database access tools
- Package and development wizard makes distributing the application simple

Handling some of the common controls in Visual Basic 6:

The Text Box:

The text box is the standard control for accepting input from the user as well as to display the output. It can handle string (text) and numeric data but not images or pictures. String in a text box can be converted to a numeric data by using the function `Val(text)`. The Text Box is like a Label but, it is used to input data into the program. The data typed in is in the Text property of the control. When the program is Run, only the controls that can be manipulated will be activated. For example, if the form contains 3 Labels, 3 Text Boxes and 3 Buttons, when it is Run, the cursor will not stop at the labels. When the user hits the Tab key, the cursor will go to the first Text Box or Button - not necessarily the first one on the form but, the first one that was created. That is called the Tab order and you have to specify it.

On the form there is only one control at any given time that has the cursor on it - it is said to have Focus. If you type data, the control with Focus will receive it. You change the Focus with Tab or by clicking on a different control.
Text1.SelStart = Len(Text1.Text)

Text1.SelText = StringToBeAdded

One of the typical operations you could find yourself performing with these properties is selecting the entire contents of a TextBox control. You often do it when the caret enters the field so that the user can quickly override the existing value with a new one, or start editing it by pressing any arrow key:

Private Sub Text1_GotFocus()
    Text1.SelStart = 0

    ' A very high value always does the trick.
    Text1.SelLength = 9999
End Sub

The Label:

The label is a very useful control for Visual Basic, as it is not only used to provide instructions and guides to the users, it can also be used to display outputs. One of its most important properties is Caption. Using the syntax label.Caption, it can display text and numeric data. You can change its caption in the properties window and also at runtime.

This is the first control in a Visual Basic Form. It is used to display static text, titles and screen output from operations. The important properties to remember:

1. Caption - the text that is displayed in the label
2. BackColor and ForeColor - colors of the background and the text
3. BackStyle - opaque or transparent - whether the background is visible or not
4. Font - font and size of text
5. Alignment - text centered, left or right
6. Multiline- True or False - if True, you can have several lines of text, delimited by <CR> in the label - by default, it is set to False.

The Visual Basic Control for Labels is defined by the following code segment.

//---------------Label Control for Visual Basic Properties ------------------------//

Private Sub MakeTransparent()
    Label1.BackStyle = vbTransparent
End Sub

Private Sub MakeTransparent()
    ' Let controls behind the label show through.
    Label1.SendToBack()
    ' Make the portion of controls behind the label transparent
    Label1.BringToFront()
End Sub

The Command Button:

Using Command Button controls is trivial. In most cases, just draw the control on the form's surface, set its caption property to a suitable string (adding character to associate a hot key with the control if you so choose), and you're finished, at least with user-interface issues. To make the button functional, you write code in its Click event procedure, as in this fragment:

// ---------------------- Command Button Properties ------------------------//

Private Sub Command1_Click ()
    ' Save data, then unload the current form.
Call SaveDataToDisk
Unload Me
End Sub

The command button is one of the most important controls as it is used to execute commands. It displays an illusion that the **End Sub** button is pressed when the user clicks on it. The most common event associated with the command button is the **Click** event, and the syntax for the procedure is:

```vbnet
Private Sub Command1_Click()
Statements
End Sub
```

**The PictureBox:**

**The PictureBox Control:**

PictureBox controls are among the most powerful and complex items in the Visual Basic Toolbox window. In a sense, these controls are more similar to forms than to other controls. For example, PictureBox controls support all the properties related to graphic output, including AutoRedraw, ClipControls, HasDC, FontTransparent, CurrentX, CurrentY, and all the Draw, Fill, and Scale properties. PictureBox controls also support all graphic methods, such as Cls, PSet, Point, Line, and Circle and conversion methods, such as ScaleX, ScaleY, TextWidth, and TextHeight. In other words, all the techniques that are described for forms can also be used for PictureBox controls.

**The Combo Box:**

The function of the Combo Box is also to present a list of items where the user can click and select the items from the list. However, the user needs to click on the small arrowhead on the right of the combo box to see the items which are presented in a drop-down list. In
order to add items to the list, you can also use the AddItem method. For example, if you wish to add a number of items to Combo box 1, you can key in the following statements.

Example

Private Sub Form_Load()
    Combo1.AddItem "Item1"
    Combo1.AddItem "Item2"
    Combo1.AddItem "Item3"
    Combo1.AddItem "Item4"
End Sub

The Check Box:

The Check Box control lets the user selects or unselects an option. When the Check Box is checked, its value is set to 1 and when it is unchecked, the value is set to 0. You can include the statements Check1.Value=1 to mark the Check Box and Check1.Value=0 to unmark the Check Box, as well as use them to initiate certain actions. For example, the program will change the background color of the form to red when the check box is unchecked and it will change to blue when the check box is checked. You will learn about the conditional statement If…Then….Elesif in later lesson. VbRed and vbBlue are color constants and BackColor is the background color property of the form.

Example

Private Sub Command1_Click()
    If Check1.Value = 1 And Check2.Value = 0 Then
        MsgBox "Apple is selected"
    End If
ElseIf Check2.Value = 1 And Check1.Value = 0 Then

MsgBox "Orange is selected"

Else

MsgBox "All are selected"

End If

End Sub
Section 3.2

SOFTWARE DEVELOPMENT

The software for the design and development of computer based system for the measurement of ultrasonic velocity and absorption in liquids is developed using Visual Basic 6.

The main role of the software in the present study is to test and control the following activities.

1. To move the receiver transducer (Rx) precisely by means of a stepper motor by sending the appropriate signals in sequence.
2. To fix the position of the receiver transducer corresponding to the minimum.
3. To make the data acquisition system to convert the analog information into digital information.
4. To compute and display the ultrasonic velocity and absorption in given liquids.
5. To make different functional units of the system to work in a systematic and sequential manner.
6. To indicate the hardware defects if any.

The necessary software package is developed on these lines for effective functioning of the system. Graphical user friendly Visual Basic language is chosen for this purpose. This software is developed in visual studio 6 front end is designed with the help of tool provided in visual basic 6. As Visual Basic is one of the event driven programming language which requires to define any event on associated with controls.
Various control used in software are textbox, command button, combo box, sliders, labels, picture box, etc. are listed below with purpose.

1) This is pictorial representation of temp at which ultrasonic velocity is calculated.

2) Text box: - this control is used to display obtained value or to enter any value for calculation in text form which is internally converted in to desired type, in code.

a. This text is used to display temperature at which ultrasonic velocity is calculated

\[25 \, ^\circ\text{C}\]

b. This displays initial start time.

Start Time (T1)

\[
\begin{array}{c}
\text{Start Time (T1)} \\
\end{array}
\]

c. This displays calculated end time value.

End Time (T2)

\[
\begin{array}{c}
\text{End Time (T2)} \\
\end{array}
\]

d. This displays total time taken to move from distance \(X_1\) to distance \(X_2\)

\[
\text{Transit Time (T = T2 - T1)}
\]

e. This textbox is used to enter distance move by the user in mm

Dist Moved (mm)

\[
\begin{array}{c}
\text{Dist Moved (mm)} \\
\end{array}
\]

f. This text box is used to enter distance D1 moved by the user to calculate attenuation.

Dist. Moved D1

\[
\begin{array}{c}
\text{Dist. Moved D1 (M)} \\
\end{array}
\]
g. This text box is used to enter distance D2 moved from distance D1 by the user to calculate attenuation.

\[
\text{Dist. Moved D2 (M)}
\]

h. This text box displays initial peak value of signal in terms of voltage at distance D1

\[
\text{Voltage 1}
\]

i. This text box displays final peak value of signal in terms of voltage at distance D2

\[
\text{Voltage 2}
\]

j. This text box is used to display calculated alpha value

\[
\text{Alpha (Np/m)}
\]

k. This text box is used to display calculated attenuated value.

\[
\text{Np/MHz2}
\]

3) Combo box: This control is used to make choice from the list of options provided in combobox, it drops down various content in list form to make choice.

\[
\text{Sample}
\]

Benzene
Dist. Water
Carbon Tetrachloride

4) Command Button: This control is used to perform action on associated event with control. Action on event is defined at back end, code is provided in code section for particular command button.

a. This command button is used to calculate ultrasonic velocity for chosen sample

\[
\text{Get Velocity}
\]
b. This command button is used to calculate ultrasonic absorption for chosen sample.

5) This command button is designed to save calculated ultrasonic velocity and absorption values.

6) This command button is designed to clear all calculated value to reenter values for calculation.

7) This command button is designed to start stepper motor.

8) This command button is designed to stop stepper motor.

9) This command button is designed to move stepper motor forward.

10) This command button is designed to move stepper motor backward direction.

11) This command button used to move stepper motor in provided index.

The flow chart diagram for the measurement of ultrasonic velocity and absorption in liquids is shown in Fig.3.1. The detailed program for the measurement of ultrasonic velocity and absorption in liquids is followed later.
Fig. 3.1: Flow chart for the measurement of ultrasonic velocity and absorption in liquids & liquid mixtures.
Section 3.3

PROGRAM IN DETAIL

The software developed in the present study is a graphical user-friendly program and easy to operate⁵. The software package is written in Visual Basic. The execution of the program gives the measurement of ultrasonic velocity and absorption in liquids. The program details are given below.

VISUAL BASIC PROGRAM:

************************************************************************
GLOBAL DECLARATION OF VARIABLES
************************************************************************

Public Declare Function MoveToEx Lib "gdi32" (ByVal hdc As Long, ByVal x As Long,
ByVal y As Long, lpPoint As POINTAPI) As Long

Public Declare Function LineTo Lib "gdi32" (ByVal hdc As Long, ByVal x As Long,
ByVal y As Long) As Long

Public Declare Function GetPixel Lib "gdi32" (ByVal hdc As Long, ByVal x As Long,
ByVal y As Long) As Long

Public Declare Function SetPixel Lib "gdi32" (ByVal hdc As Long, ByVal x As Long,
ByVal y As Long, ByVal crColor As Long) As Long

Declare Function GetPortVal Lib "WinIo.dll" (ByVal PortAddr As Integer, ByRef PortVal
As Long, ByVal bSize As Byte) As Boolean

Declare Function SetPortVal Lib "WinIo.dll" (ByVal PortAddr As Integer, ByVal PortVal
As Long, ByVal bSize As Byte) As Boolean
Declare Function InitializeWinIo Lib "WinIo.dll" () As Boolean

Declare Function ShutdownWinIo Lib "WinIo.dll" () As Boolean

Public Declare Function StretchBlt Lib "gdi32" (ByVal hdc As Long, ByVal x As Long, ByVal y As Long, ByVal nWidth As Long, ByVal nHeight As Long, ByVal hSrcDC As Long, ByVal xSrc As Long, ByVal ySrc As Long, ByVal nSrcWidth As Long, ByVal nSrcHeight As Long, ByVal dwRop As Long) As Long

Public Declare Function BitBlt Lib "gdi32" (ByVal hDestDC As Long, ByVal x As Long, ByVal y As Long, ByVal nWidth As Long, ByVal nHeight As Long, ByVal hSrcDC As Long, ByVal xSrc As Long, ByVal ySrc As Long, ByVal dwRop As Long) As Long

Public Const SRCCOPY = &HCC0020 ' (DWORD) dest = source

Public Type POINTAPI
  x As Long
  y As Long
End Type

Type FILETIME
  lLowDateTime As Long
  lHighDateTime As Long
End Type

Public Declare Function FindWindow Lib "user32" Alias "FindWindowA" (ByVal lpClassName As String, ByVal lpWindowName As String) As Long

Public Declare Function GetWindowThreadProcessId Lib "user32" (ByVal hwnd As Long, lpdwProcessId As Long) As Long
Public Declare Function OpenProcessToken Lib "advapi32.dll" (ByVal ProcessHandle As Long, ByVal DesiredAccess As Long, TokenHandle As Long) As Long

Public Declare Function OpenProcess Lib "kernel32" (ByVal dwDesiredAccess As Long, ByVal bInheritHandle As Long, ByVal dwProcessId As Long) As Long

Public Declare Function TerminateProcess Lib "kernel32" (ByVal hProcess As Long, ByVal uExitCode As Long) As Long

Public Declare Function CloseHandle Lib "kernel32" (ByVal hObject As Long) As Long

Public P As POINTAPI

Public Base_Add As Long = &H800

Public Ctr1 As Long = Base_Add + &H3

Public Ctr2 As Long = Base_Add + &H13

Public Port_A1 As Long = Base_Add + &H0

Public Port_B1 As Long = Base_Add + &H1

Public Port_C1 As Long = Base_Add + &H2

Public Port_A2 As Long = Base_Add + &H10

Public Port_B2 As Long = Base_Add + &H11

Public Port_C2 As Long = Base_Add + &H12

Public ActAdcVal As Long

Public ActiveColor, InactiveColor As Long

Public Analysis As Boolean
CHAPTER 3

PROGRAM FOR CONFIGURATION AND START OF MODULE

Sub Main ()
Dim t As String
Open App.Path & "\System.add" For Input As #5
Line Input #5, t
Base_Add = Val(t)
Close #5
If InitializeWinIo = False Then
MsgBox "Unable to Initialize System Software! Please Restart Windows and try again.", vbCritical,
"System Initialize"
'Exit Sub
End If
/*Configuring PCI Card*/
Port_A1 = Base_Add + &H0
Port_B1 = Base_Add + &H1
Port_C1 = Base_Add + &H2
Ctr1 = Base_Add + &H3
Port_A2 = Base_Add + &H4 ' &H10
Port_B2 = Base_Add + &H5 ' &H11
Port_C2 = Base_Add + &H6 ' &H12
Ctr2 = Base_Add + &H7 ' &H13
open_file App.Path & "\parameter\default.prm"
CurrentFile = App.Path & "\Parameter\default.prm"
SignalFrm.Show
End Sub
Private Sub read_FastADC()
    Dim Temp As Long
    Temp = 0
    For i = 0 To 511
        /* send read clock, data will be latched*/
        SetPortVal Port_B2, &H56, 1
        /* send byte address*/
        SetPortVal Port_B2, &H57, 1
        GetPortVal Port_C1, FADCData(i), 1 ' get data
        FADCData(i) = FADCData(i) And &HFF
        If (i > gatePosition1 And i < gatePosition2) Then
            If Temp < FADCData(i) Then
                Temp = FADCData(i)
            End If
        End If
    Next
    AvgTemp = AvgTemp + Temp
    TempCount = TempCount + 1
Text8.Text = Temp
If TempCount > 50 Then
AvgTemp = AvgTemp / TempCount
AvgFilter = AvgFilter + AvgTemp
AvgTemp = 0
TempCount = 0
AvgFilterCount = AvgFilterCount + 1
End If
If AvgFilterCount > 2 Then
AvgFilter = AvgFilter / AvgFilterCount
Text7.Text = AvgFilter
If AmpFetch = True And AmpRdCount < 15 Then
AmpDif(AmpRdCount) = (AvgFilter * (5 / 255))
Command9.Enabled = True
AmpFetched = True
AmpFetch = False
End If
AvgFilter = 0
AvgFilterCount = 0
TempCount = 0
AvgTemp = 0
End If
If AmpFetched = False
AmPXcur = ((Val(Text23.Text) / 5) + 1) * 40
AmPYcur = (Picture11.height - Int(((NewAmp) * 200) / 5))
Picture11.ForeColor = vbYellow
Picture11.DrawWidth = 5
Picture11.Line (AmPXcur, AmPYcur)-(AmPXcur, AmPYcur)
Picture11.ForeColor = vbRed
Picture11.DrawWidth = 1
Picture11.Line (AmPXcur, Picture11.ScaleHeight)-(AmPXcur, AmPYcur)
Picture11.ForeColor = vbGreen
Picture11.DrawWidth = 1
Picture11.Line (AmPXprev, AmPYprev)-(AmPXcur, AmPYcur)
AmPXprev = AmPXcur
AmPYprev = AmPYcur
Picture11.Refresh
AmpFetched = False
/* clear control signal*/
ControlByte = (ControlByte Or &H18) ' 0001 1000
End Sub
Private Sub Read_DualSADC()
Dim Val1 As Long
Dim sts As Long
Dim Mask As Byte
Dim InvalidCounts As Long
Dim Databyte As Long
read_FastADC
Picture5.Cls
plot_FADC
End Sub

*************************************************************************/

PROGRAM FOR PLOTTING OF TRANSDUCER SIGNAL
*************************************************************************/

Private Sub Draw_Signal()
    /*draw white graph of RF waveform*/
    Picture5.ForeColor = vbWhite
    Xprev = 1
    Xcur = Xprev + StepSize
    MoveT0Ex Picture5.hdc, 0, 255 - (FADCData(0)), P
    For i = 1 To 512 'RFSamples ' Step 2
        FADCData(i) = FADCData(i) And &HFF
        LineTo Picture5.hdc, Xcur, 255 - (FADCData(i))
        Xcur = Xcur + StepSize
        Text6.Text = (FADCData(i))
    Next
    End Sub
PROGRAM FOR READING TRANSDUCER SIGNAL AND MEASUREMENT OF
ULTRASONIC ABSORPTION IN LIQUIDS & LIQUID MIXTURES

Private Sub ReadSignal ()
Dim Temp As Long
Temp = 0
For i = 0 To 511
GetPortVal Port_C1, FADCData(i), 1
FADCData(i) = FADCData(i) And &HFF
If (i > gatePosition1 And i < gatePosition2) Then
If Temp < FADCData(i) Then
Temp = FADCData(i)
End If
End If
Next
AvgTemp = AvgTemp + Temp
TempCount = TempCount + 1
Text8.Text = Temp
If TempCount > 50 Then
AvgTemp = AvgTemp / TempCount
AvgFilter = AvgFilter + AvgTemp
AvgTemp = 0
TempCount = 0
AvgFilterCount = AvgFilterCount + 1
End If
If AvgFilterCount > 2 Then
AvgFilter = AvgFilter / AvgFilterCount
Text7.Text = AvgFilter
If AmpFetch = True And AmpRdCount < 15 Then
AmpDif(AmpRdCount) = (AvgFilter * (5 / 255))
Command9.Enabled = True
Text20.Text = Round(AmpDif(AmpRdCount), 5)
Text21.Text = AvgFilterCount
AmpFetched = True
AmpFetch = False
End If
AvgFilter = 0
AvgFilterCount = 0
TempCount = 0
AvgTemp = 0
End If
If AmpRdCount >= 9 Then Exit Sub
If AmpFetched = True And AmpRdCount < 15 Then
If AmpDif(AmpRdCount) > 127 And AmpDif(AmpRdCount) < 150 Then
MsgBox "There Is No Signal Please Check The Connection"

Exit Sub

End If

If AmpRdCount = 0 Then

AmPXcur = 40

AmPXprev = 40

AmPYcur = (Picture11.Height - Int(((AmpDif(0)) * 200) / 5))

AmPYprev = (Picture11.Height - Int(((AmpDif(0)) * 200) / 5))

MoveToEx Picture11.hdc, Xcur, Ycur, P

Picture11.ForeColor = vbYellow

Picture11.DrawWidth = 3

Picture11.Line (AmPXcur, AmPYcur)-(AmPXcur, AmPYcur)

Picture11.ForeColor = vbYellow

AmPXprev = AmPXcur

AmPYprev = AmPYcur

Picture11.Refresh

AmpFetched = False

Txt_Voltage1.Text = Round(AmpDif(0), 3)

Text20.Text = AmpDif(0)

Else

If AmpFetched = True Then

DeltaDistance = (((Val(Text23.Text) - Val(Text22.Text)) / 1000))

NewAmp = AmpDif(AmpRdCount)
If (NewAmp * (255 / 5)) > 127 And (NewAmp * (255 / 5)) < 140 Then

MsgBox "please check signal in the gate"

AmpRdCount = AmpRdCount - 1

AmpFetched = False

Exit Sub

End If

Private Sub Attenution_calcu()

Amplitude = (((AmpDif(AmpRdCount - 1)) / (AmpDif(AmpRdCount))))

Amplitude = Log(Amplitude)

Alpha = (Amplitude) / (DeltaDistance)

Attenution = Round(Alpha / 4, 5) * 1000 & " * 10^-15"

Txt_Voltage1.Text = Round((AmpDif(AmpRdCount - 1)), 3)

Txt_voltage2.Text = Round((AmpDif(AmpRdCount)), 3)

Txt_Alpha.Text = Round(Alpha, 4)

Txt_Attenution.Text = Round(Alpha / 4, 5) * 1000 & " * 10^-15"

Lbl_unit.Caption = "    Np/M Hz2"

End sub

End If

If AmpFetched = False

AmPXcur = ((Val(Text23.Text) / 5) + 1) * 40

AmPYcur = (Picture11.Height - Int(((NewAmp) * 200) / 5))

Picture11.ForeColor = vbYellow

Picture11.DrawWidth = 5
Picture11.Line (AmPXcur, AmPYcur)-(AmPXcur, AmPYcur)

Picture11.ForeColor = vbRed

Picture11.DrawWidth = 1

Picture11.Line (AmPXcur, Picture11.ScaleHeight)-(AmPXcur, AmPYcur)

Picture11.ForeColor = vbGreen

Picture11.DrawWidth = 1

Picture11.Line (AmPXprev, AmPYprev)-(AmPXcur, AmPYcur)

AmPXprev = AmPXcur

AmPYprev = AmPYcur

Picture11.Refresh

AmpFetched = False

End If

End If

End Sub
Private Sub Velocity Click() /* Calculation of velocity*/

/*Checking For All inputs Whether it is NULL of incorrect format if So Promting for Its Correction*/
Check = IsNumeric(Text23Text)
If  Text10.Text = "" Or (Not Check) Then
MsgBox "Please D1 Either It Is Null Or Alpha Numeric"
Text10.SetFocus
Exit Sub
End If

/*Checking For All inputs Whether it is NULL of incorrect format if So Promting For Its Correction */
If Combo1.Text = "" Then
MsgBox " Sample Is Not Selected Please Select Sample"
Exit Sub
End If

/*Checking Between End Time and Start Time And Positive Difference Is Calculated*/
If Mode = True And Val(Etime) > Val(Stime) Then
tskip = Abs(MainSkip2 - MainSkip1)
/* End Time Is Greater Then Skip Value Is Added To Difference of End Time And Start

time*/

b = Abs(tskip + (Val(Etime) - Val(Stime)))

Text11.Text = Round(((Val(Text10.Text) / (b)) * 1000), 4)

End If

/*Checking Between End Time and Start Time And Positive Difference Is Calculated*/

If Mode = True And Val(Etime) < Val(Stime) Then

tskip = Abs(MAinSkip2 - MainSkip1)

/*End Time Is Greater Then Skip Value Is Subtracted To Difference of End Time And Start
time*/

b = Abs(tskip - (Val(Stime) - Val(Etime)))

Text11.Text = Round(((Val(Text10.Text) / (b)) * 1000), 4)

End If

End Sub
**PROGRAM FOR STEPPER MOTOR CODE**

Private Sub Stepper_Start_Click()

/*this acived through setting TTI Voltage in assigned port and Latching data value in to the register*/

SetPortVal Port_A1, &HEA, 1

/* Setting some delay before latching*/

For i = 0 To 1000

Next

SetPortVal Port_A1, &HFF, 1

End Sub

Private Sub Stepper_Stop()

SetPortVal Port_A1, &HE8, 1

SetPortVal Port_A1, &HFF, 1

End Sub

Private Sub BackWord_Click()

SetPortVal Port_A1, &HE, 1

End Sub

Private Sub Forward_Click()

SetPortVal Port_A1, &H7, 1

SetPortVal Port_A1, &HFF, 1

End Sub

Private Sub Index_Click()

Yindex = Val(Text31.Text) * 400    ' 1 mm = 400 Pulses
/*Y motor Step Pulses Dumping*/

" SetPortVal Port_C2, &HFF, 1 'aking D4 bit high for dumping

'SetPortVal Port_C2, &HFF, 1
LsbData = Yindex Mod 256
SetPortVal Port_C2, &HFF, 1
SetPortVal Port_B1, LsbData, 1 ' lsb data
SetPortVal Port_C2, &HFF, 1
SetPortVal Port_A1, &HE3, 1 ' lsb address
SetPortVal Port_A1, &HFF, 1
SetPortVal Port_C2, &HFF, 1
SetPortVal Port_B1, MsbData, 1 ' msb data
SetPortVal Port_C2, &HFF, 1
SetPortVal Port_A1, &HE4, 1 ' msb address
SetPortVal Port_A1, &HFF, 1
MsbData = (Yindex And &HFF00) / 256
SetPortVal Port_C2, &HFF, 1
MsbData = MsbData + 1
SetPortVal Port_B1, MsbData, 1 ' msb data
SetPortVal Port_A1, &HFF, 1
SetPortVal Port_C2, &HFF, 1
SetPortVal Port_A1, &HE8, 1
For i = 0 To 1000
Next
SetPortVal Port_A1, &HFF, 1
End Sub
Fig. 3.2: Front end of Visual Basic for ultrasonic measurements
REFERENCES

2. Dr. Dobb, Software tools for the professional programmer, volume 32, M&T Publication (2007).