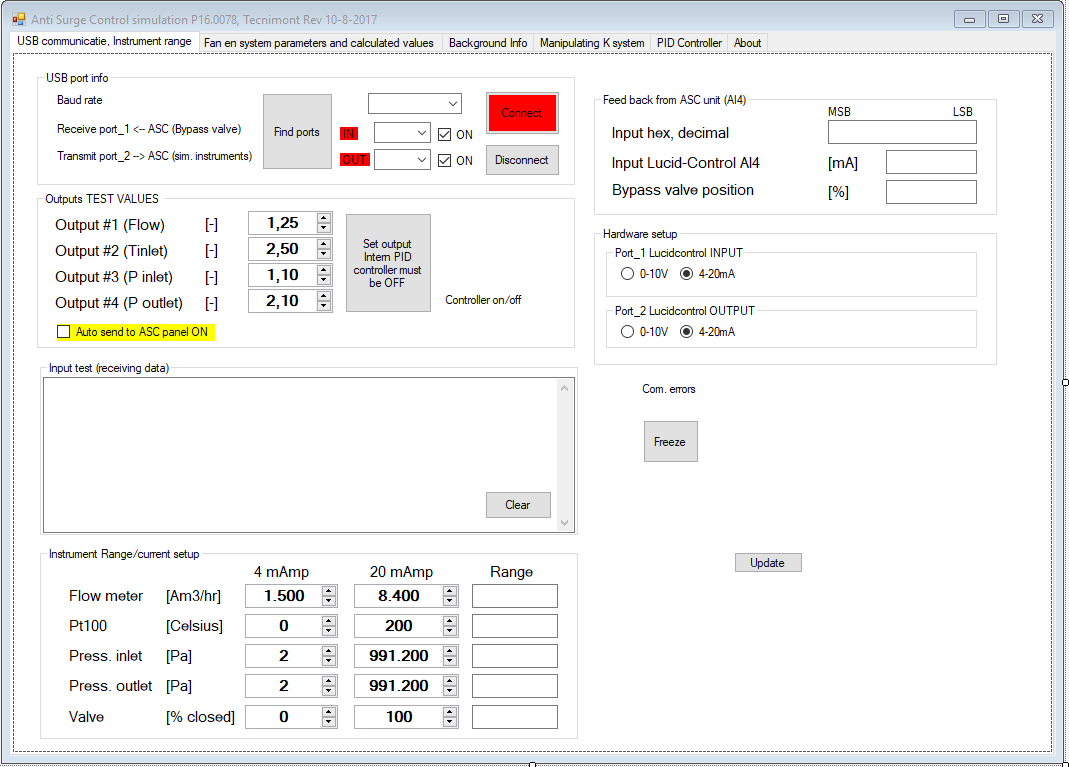
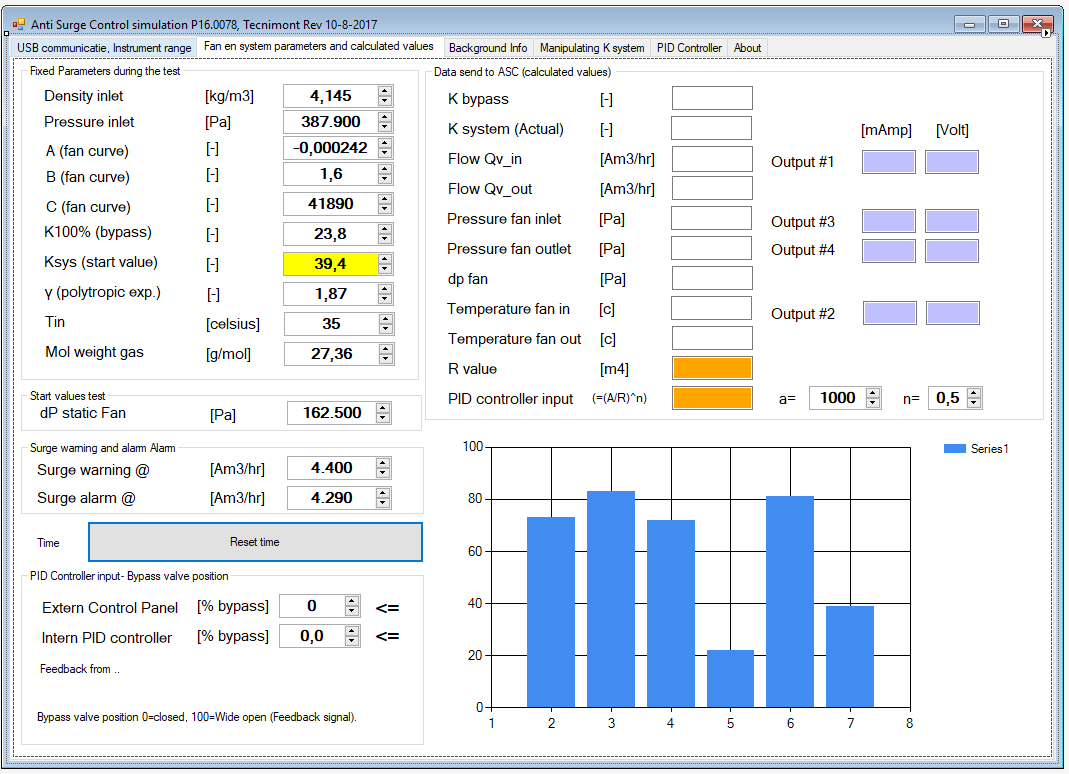
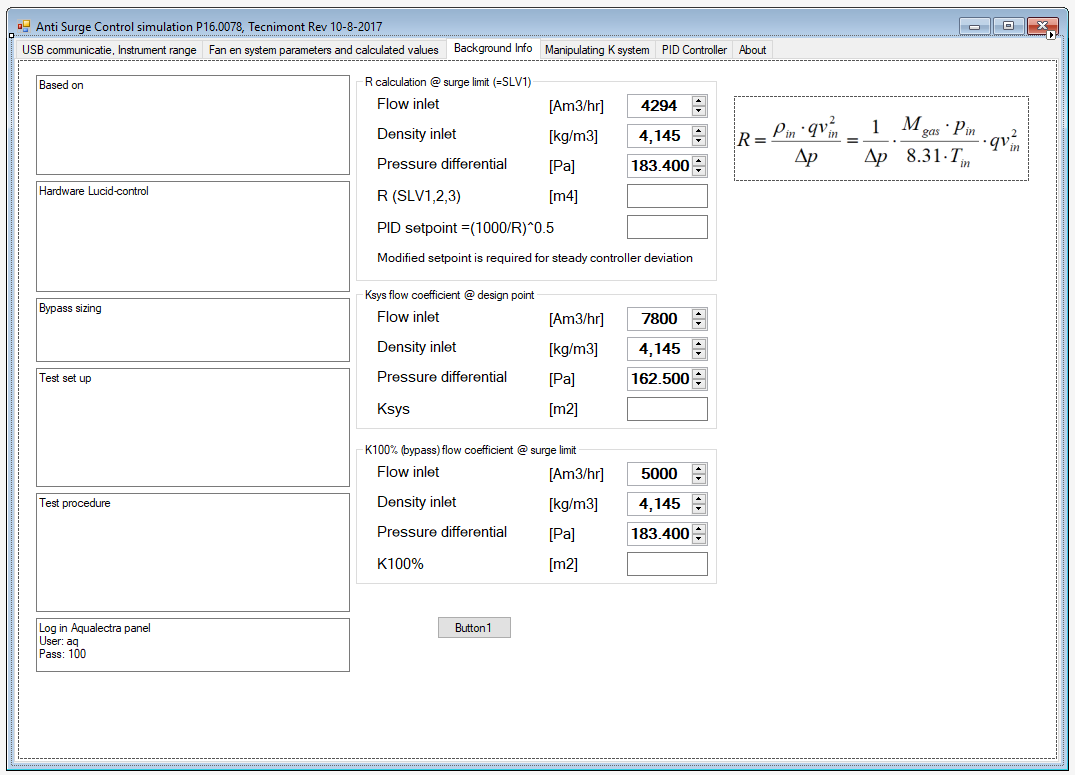
Source listing Anti Surge Simulation Program.

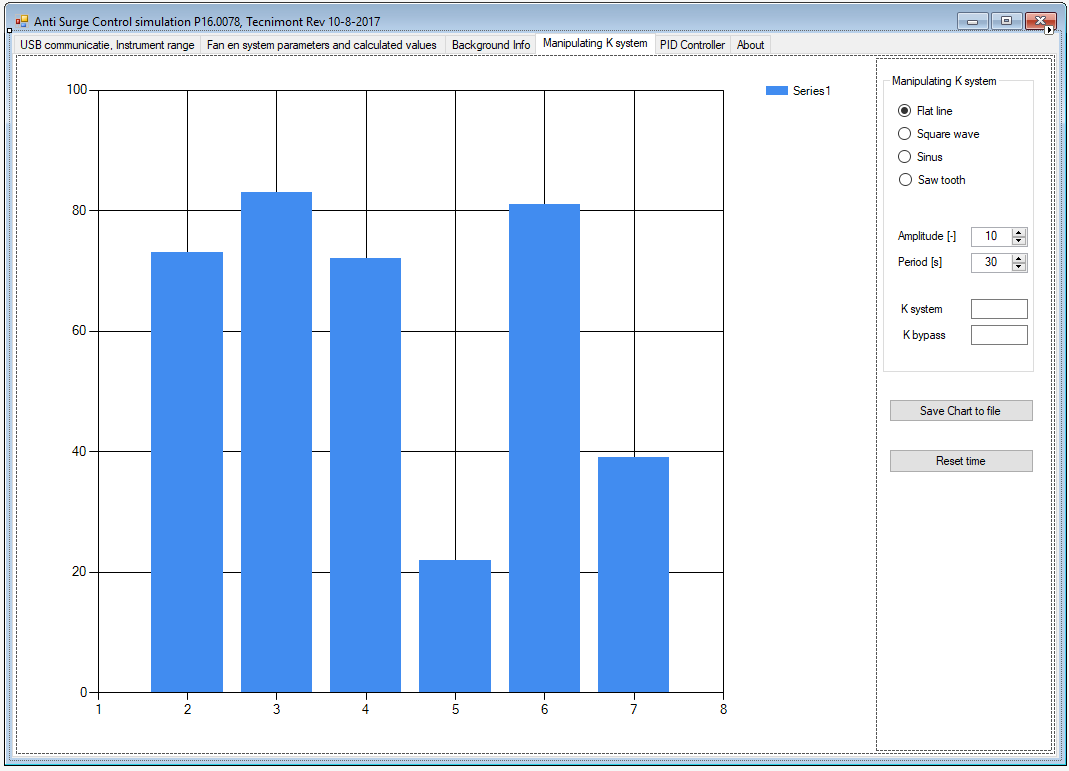
G.Pathuis

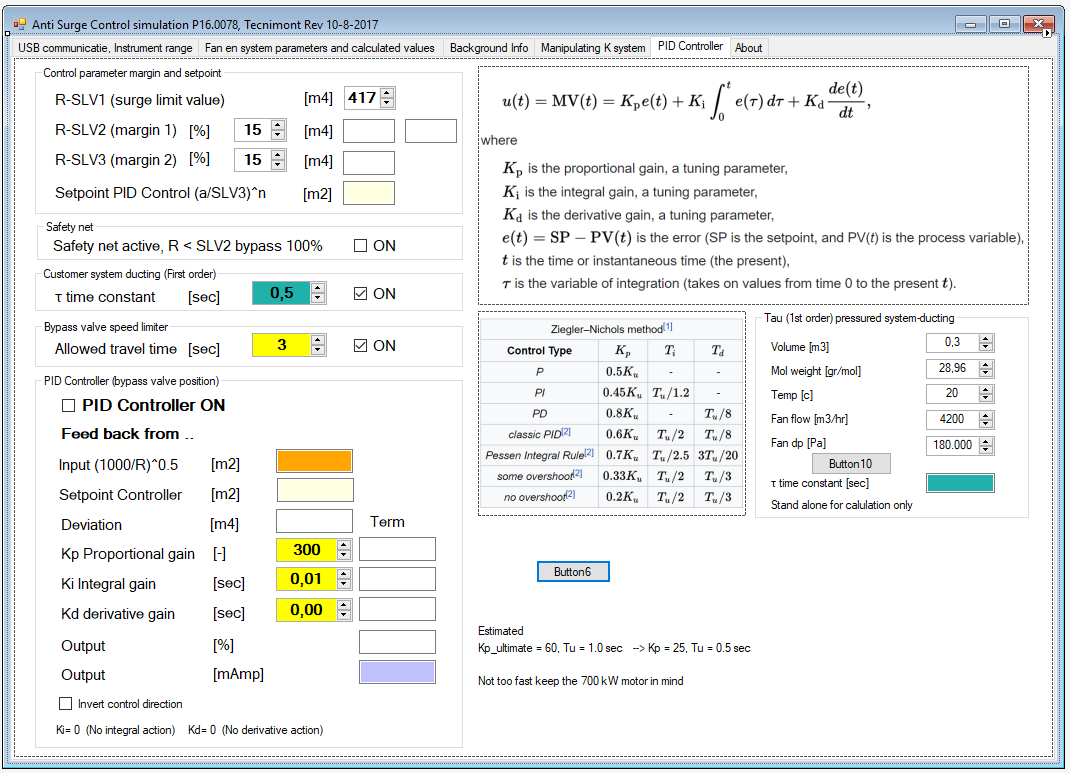
Dd 17-07-2017

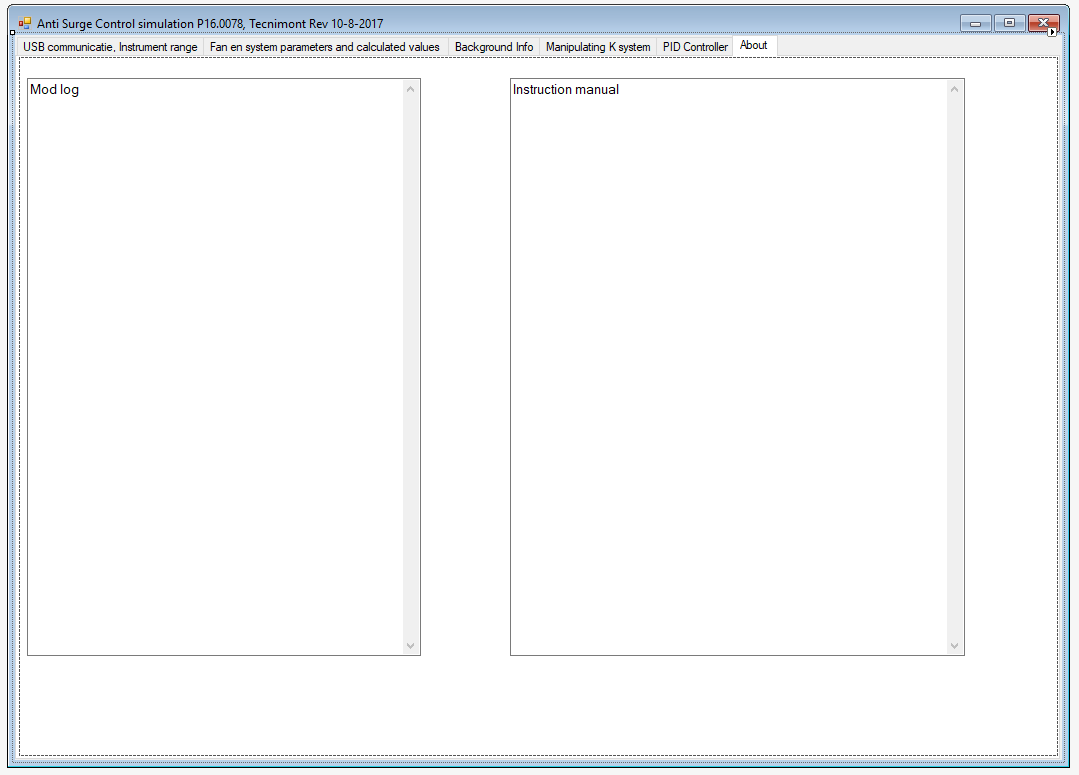












'======================================================

'Update via N:\Engineering\VBasic\Anti\_Surge\_Control\

'======================================================

Imports System.Globalization

Imports System.IO

Imports System.IO.Ports

Imports System.Math

Imports System.Text

Imports System.Threading

Public Class Form1

Dim time As Double

Dim pv(4) As Double 'Process values 0,1,2,3,4

Dim Cout(4) As Double 'Current Outputs

Dim Vout(4) As Double 'Voltage Outputs

Dim \_bypass\_pos As Double = 0 'Bypass valve position (0-100%)

Dim \_bypass\_ma As Double = 0 'Bypass valve position (mAmp)

Dim \_last\_deviation As Double 'PID control

Dim \_last\_output As Double 'PID control

Dim \_Pterm, \_Iterm, \_Dterm As Double

Dim \_counter As Integer = 0

Dim \_yold(9) As Double 'Used in first order system ducting

Dim myPort As Array 'COM Ports detected

Dim comOpen As Boolean

Dim \_PID\_output\_ma As Double 'Interne PID controller mAmp output

' Private Property ConnectionOK As Boolean

Private Sub Form1\_Load(sender As Object, e As EventArgs) Handles MyBase.Load

Dim i As Integer

Thread.CurrentThread.CurrentCulture = New CultureInfo("en-US")

Thread.CurrentThread.CurrentUICulture = New CultureInfo("en-US")

TextBox16.Text =

"Based on " & vbCrLf &

"Anti Surge Control Test Procedure, Guus van Gemert 2017" & vbCrLf &

"KIMA, Ammonia & Urea Fertilizer Project, " & vbCrLf &

"Start-up blower HD2 407/1235/T16B, 3000 rpm, 700 kW" & vbCrLf &

"VTK project P16.0078" & vbCrLf &

"Bypass valve size is 6 inch, valve speed Is 3 seconds"

TextBox17.Text =

"Hardware " & vbCrLf &

"4 channel Analog Input (4-20mAmp) USB Module, LucidControl AI4" & vbCrLf &

"4 channel Analog Output (4-20mAmp) USB Module, LucidControl AO4" & vbCrLf &

"Install a Windows drive, see Lucid-Control.com For download"

TextBox18.Text =

"Bypass valve sizing" & vbCrLf &

"Size To handle 50% Or more Of the maximum flow" & vbCrLf &

"Valve speed 30 seconds" & vbCrLf &

" "

TextBox19.Text =

"Test set-up" & vbCrLf &

"The Laptop is connect to a AO4 and AAI4 LucidControl unit." & vbCrLf &

"The Laptop program generates 4 signals and receives 1 each are 4-20mAmp" & vbCrLf &

"The send signals represent Flow inlet/outlet [Am3/hr], Temp fan inlet [c], " & vbCrLf &

"Pressure [Pa] fan inlet and delta pressure over the fan [Pa]" & vbCrLf &

"The receives signals represent the position of the bypass valve" & vbCrLf &

" "

TextBox20.Text =

"Test procedure" & vbCrLf &

"Start situation is stable, sitting on Fan Curve" & vbCrLf &

"The system flow Coefficient Ksys is changed resulting" & vbCrLf &

"in moving to another spot on the fan Curve." & vbCrLf &

"When we near the Surge-area the connected ASC must react by" & vbCrLf &

"opening the bybass valve and returning to a save spot on the" & vbCrLf &

"fan-Curve."

TextBox39.Text =

"Operator instructions" & vbCrLf &

"Setup the instrument ranges, e.g. PT100, 0-200c equals 4-20mA" & vbCrLf &

"Select Lucid controls 0-10V or 4-20mA " & vbCrLf &

"Select Surge Limit Margins @ 15%" & vbCrLf &

" " & vbCrLf &

" \*\*\*Determine Fan system Characteristics\*\*\* " & vbCrLf &

"ASC panel, set PID settings Kp=0.1, Ki= 0.0" & vbCrLf &

"ASC panel switch to manual, open the bypass 100%" & vbCrLf &

"Start the fan against closed valve, open slow when at speed" & vbCrLf &

"Close bypass valve slowly" & vbCrLf &

"When system stable switch PID controller to automatic " & vbCrLf &

"Now increase PID controller Kp value, until system becomes unstable " & vbCrLf &

"Write down Kp value, and system cycle time" & vbCrLf &

"Use Ziegler-Nichols rules" & vbCrLf &

"Enter Kp and Ti into the PID controller" & vbCrLf &

"If system too slow increase Kp" & vbCrLf &

"If system unstable reduce Kp" & vbCrLf &

" " & vbCrLf &

"There are 2 different operator modes" & vbCrLf &

"1) ==Testing Anti Surge Control panel==" & vbCrLf &

"Connect the Lucid control modules to the control panel" & vbCrLf &

"Connect the USB cables to the laptop" & vbCrLf &

"Find the ports, connect the ports" & vbCrLf &

"Note with swapped USB ports, the program will not work properly" & vbCrLf &

"Checkbox PID controller OFF" & vbCrLf &

"Checkbox AUTO send ON " & vbCrLf &

"Select sinus/flat-line/etc " & vbCrLf &

"The ASC panel sees now fluctuating instrument input signals" & vbCrLf &

"The physical process values must identical on the laptop and panel" & vbCrLf &

"Reduce Ksys now the Bypass valve must start opening " & vbCrLf &

" " & vbCrLf &

"2) ==Fan system Characteristics are known, test the PID settings== " & vbCrLf &

"Checkbox PID controller ON" & vbCrLf &

"Checkbox AUTO send OFF " & vbCrLf &

"Set Ducting First order time constant (say 20 sec)" & vbCrLf &

"Enter Kp and Ti into the PID controller" & vbCrLf &

"Play with the parameters to find the optimum solution " & vbCrLf &

" " & vbCrLf &

" "

TextBox50.Text =

"Program modification history" & vbCrLf &

"dd 17-07-2017" & vbCrLf &

"Extern/Intern feedback depends on checkbox PID controller On/Off" & vbCrLf &

"PID Invert Control direction flipped" & vbCrLf &

"PID settings changed to Kp=25, Ki= 0.5" & vbCrLf &

"dd 18-07-2017" & vbCrLf &

"Ksys, K100% calc. added at the background tab" & vbCrLf &

"Operator instructions added" & vbCrLf &

" " & vbCrLf &

" "

For i = 0 To 3

pv(i) = 1 'Initial value

Next

Reset()

Update\_calc\_screen()

If RadioButton8.Checked Then 'Set outputs to V or mA

Output\_set\_to\_V()

Else

Output\_set\_to\_mA()

End If

End Sub

Private Sub Reset()

Init\_Chart1()

Init\_Chart2()

Timer1.Interval = 300 'Berekeningsinterval 300 msec

time = 0

Timer1.Enabled = True

TextBox31.Text = "50" 'PID controller output 50%

End Sub

Private Sub Init\_Chart1()

Dim i As Integer

Try

Chart1.Series.Clear()

Chart1.ChartAreas.Clear()

Chart1.Titles.Clear()

Chart1.ChartAreas.Add("ChartArea0")

For i = 0 To 4

Chart1.Series.Add(i.ToString)

Chart1.Series(i.ToString).ChartArea = "ChartArea0"

Chart1.Series(i.ToString).ChartType = DataVisualization.Charting.SeriesChartType.Line

Chart1.Series(i.ToString).BorderWidth = 3

Next

Chart1.Titles.Add("ASC testing")

Chart1.Titles(0).Font = New Font("Arial", 12, System.Drawing.FontStyle.Bold)

Chart1.Series(0).Name = "Flow inlet"

Chart1.Series(1).Name = "Pressure in"

Chart1.Series(2).Name = "delta P"

Chart1.Series(3).Name = "Temp in"

Chart1.Series(4).Name = "Bypass valve"

Chart1.Series(0).Color = Color.Black

Chart1.ChartAreas("ChartArea0").AxisX.Title = "[sec]"

Chart1.ChartAreas("ChartArea0").AxisY.Title = "mAmp"

Chart1.ChartAreas("ChartArea0").AxisY.Minimum = 4

Chart1.ChartAreas("ChartArea0").AxisY.Maximum = 20

Chart1.ChartAreas("ChartArea0").AxisX.MajorTickMark.Size = 2

Chart1.ChartAreas("ChartArea0").AlignmentOrientation = DataVisualization.Charting.AreaAlignmentOrientations.Vertical

Catch ex As Exception

MessageBox.Show("Init Chart1 failed")

End Try

End Sub

Private Sub Init\_Chart2()

Dim i As Integer

Try

Chart2.Series.Clear()

Chart2.ChartAreas.Clear()

Chart2.Titles.Clear()

Chart2.ChartAreas.Add("ChartArea1")

For i = 0 To 4

Chart2.Series.Add(i.ToString)

Chart2.Series(i.ToString).ChartArea = "ChartArea1"

Chart2.Series(i.ToString).ChartType = DataVisualization.Charting.SeriesChartType.Line

Chart2.Series(i.ToString).BorderWidth = 3

Next

Chart2.Titles.Add("ASC testing")

Chart2.Titles(0).Font = New Font("Arial", 12, System.Drawing.FontStyle.Bold)

Chart2.Series(0).Name = "Flow inlet"

Chart2.Series(1).Name = "Pressure in"

Chart2.Series(2).Name = "delta P"

Chart2.Series(3).Name = "Temp in"

Chart2.Series(4).Name = "Bypass valve"

Chart2.Series(0).Color = Color.Black

Chart2.ChartAreas("ChartArea1").AxisX.Title = "[sec]"

Chart2.ChartAreas("ChartArea1").AxisY.Title = "mAmp"

Chart2.ChartAreas("ChartArea1").AxisY.Minimum = 4

Chart2.ChartAreas("ChartArea1").AxisY.Maximum = 20

Chart2.ChartAreas("ChartArea1").AxisX.MajorTickMark.Size = 2

Chart2.ChartAreas("ChartArea1").AlignmentOrientation = DataVisualization.Charting.AreaAlignmentOrientations.Vertical

Catch ex As Exception

MessageBox.Show("Init Chart2 failed")

End Try

End Sub

Private Sub Draw\_Chart1()

Try

Chart1.Series(0).Points.AddXY(time, Cout(1)) 'Flow in

Chart1.Series(1).Points.AddXY(time, Cout(2)) 'Pressure in

Chart1.Series(2).Points.AddXY(time, Cout(3)) 'dP

Chart1.Series(3).Points.AddXY(time, Cout(4)) 'Temp in

If CheckBox5.Checked Then 'Intern or extern

Chart1.Series(4).Points.AddXY(time, \_PID\_output\_ma)

Else

Chart1.Series(4).Points.AddXY(time, \_bypass\_ma)

End If

Chart2.Series(0).Points.AddXY(time, Cout(1)) 'Flow in

Chart2.Series(1).Points.AddXY(time, Cout(2)) 'Pressure in

Chart2.Series(2).Points.AddXY(time, Cout(3)) 'dP

Chart2.Series(3).Points.AddXY(time, Cout(4)) 'Temp in

If CheckBox5.Checked Then 'Intern or extern

Chart2.Series(4).Points.AddXY(time, \_PID\_output\_ma)

Else

Chart2.Series(4).Points.AddXY(time, \_bypass\_ma)

End If

Catch ex As Exception

MessageBox.Show("AddXY failed")

End Try

End Sub

Private Sub Button2\_Click(sender As Object, e As EventArgs) Handles Button2.Click

SetOut()

End Sub

Private Sub SetOut()

Dim SetIoGroup(4) As Byte 'Set-Voltage or Current, see Page 22 en 26

Dim SetIoG As String = String.Empty

Dim str\_hex1 As String = String.Empty

Dim str\_hex2 As String = String.Empty

Dim str\_hex3 As String = String.Empty

Dim str\_hex4 As String = String.Empty

Dim message\_length As Integer = 0

Dim bb() As Byte

' Dim ret As String

SetIoGroup(1) = &H42 'OPC= SetIoGroup !!!!

SetIoGroup(2) = &HF 'Channel 1...4 (0000.1111==0x0F)

If RadioButton8.Checked Then

SetIoGroup(3) = &H1D 'Volt (0 to 1,000,000 MicroVolt)

Else

SetIoGroup(3) = &H23 'Current (0 to 1,000,000 MicroAmp)

End If

SetIoGroup(4) = &H10 'Len (4 x 4=16 bytes)

'------ make Command string of the Command byte array---

SetIoG = System.Text.Encoding.Default.GetString(SetIoGroup)

'---------- now convert to hex-------

SetIoG = String\_ascii\_to\_Hex\_ascii(SetIoG)

'-----Voltage/Current output, channel #1...4

str\_hex1 = Hex(CDec(NumericUpDown5.Value \* 10 ^ 6))

str\_hex2 = Hex(CDec(NumericUpDown10.Value \* 10 ^ 6))

str\_hex3 = Hex(CDec(NumericUpDown14.Value \* 10 ^ 6))

str\_hex4 = Hex(CDec(NumericUpDown15.Value \* 10 ^ 6))

'------ convert to Big endian and ------

'------ adding all string-sections to one string

SetIoG &= To\_big\_endian(str\_hex1)

SetIoG &= To\_big\_endian(str\_hex2)

SetIoG &= To\_big\_endian(str\_hex3)

SetIoG &= To\_big\_endian(str\_hex4)

'------ convert to bytes and write to port-----

bb = HexStringToByteArray(SetIoG)

If SerialPort2.IsOpen Then

TextBox26.Text &= "SetIoG= " & SetIoG & vbCrLf

Try

SerialPort2.Write(bb, 1, 20)

Catch generatedExceptionName As TimeoutException

End Try

Else

TextBox26.Text &= "SerialPort2 is closed" & vbCrLf

End If

End Sub

Private Function To\_big\_endian(str\_num As String) As String

Dim return\_val As String = String.Empty

Dim bytes() As Byte = {&H0, &H0, &H0, &H0}

Dim bytes\_big() As Byte = {&H0, &H0, &H0, &H0}

Dim b As Byte

Dim byte\_0x00() As Byte = {&H0, &H0, &H0, &H0}

Dim byte\_sum() As Byte = {&H0, &H0, &H0, &H0}

Dim value, no\_bytes As Integer

value = Convert.ToInt32(str\_num, 16)

bytes = BitConverter.GetBytes(value)

'------- determine number significant bytes

If value > 2 ^ 32 Then MessageBox.Show("Problem in To\_big\_endian")

Select Case value

Case Is >= CInt(2 ^ 24)

no\_bytes = 4

Case Is < CInt(2 ^ 24)

no\_bytes = 3

Case Is < CInt(2 ^ 16)

no\_bytes = 2

Case Is < CInt(2 ^ 8)

no\_bytes = 1

End Select

'---------- just for testing----------------

'MessageBox.Show("str\_num=" & str\_num & " No\_bytes= " & no\_bytes.ToString)

'MessageBox.Show(" bytes(0)= " & Conversion.Hex(bytes(0)))

'MessageBox.Show(" bytes(1)= " & Conversion.Hex(bytes(1)))

'MessageBox.Show(" bytes(2)= " & Conversion.Hex(bytes(2)))

'MessageBox.Show(" bytes(3)= " & Conversion.Hex(bytes(3)))

Select Case no\_bytes

Case 1 '[1 bytes] move right and add zero

bytes\_big(0) = &H0

bytes\_big(1) = &H0

bytes\_big(2) = &H0

bytes\_big(3) = bytes(0)

Case 2 '[2 bytes] move right and add zero

bytes\_big(0) = &H0

bytes\_big(1) = &H0

bytes\_big(2) = bytes(1)

bytes\_big(3) = bytes(0)

Case 3 '[3 bytes] move right and add zero (This one works!!)

bytes\_big(0) = &H0

bytes\_big(1) = bytes(2)

bytes\_big(2) = bytes(1)

bytes\_big(3) = bytes(0)

Case 4

bytes\_big(0) = bytes(3)

bytes\_big(1) = bytes(2)

bytes\_big(2) = bytes(1)

bytes\_big(3) = bytes(0)

End Select

Array.Reverse(bytes\_big) 'Now reverse order

'--------- make the string-------

For Each b In bytes\_big

return\_val += b.ToString("X2")

Next

' MessageBox.Show("Little Endian=" & str\_num.ToString & " To\_big\_endian= " & return\_val.ToString)

Return return\_val

End Function

Private Sub Timer1\_Tick(sender As Object, e As EventArgs) Handles Timer1.Tick

Dim flow1, Press\_in, p2, t1 As Double

Dim SLV1, SLV2, SLV3, setpoint As Double

Dim pv As Double

TextBox36.Text = \_bypass\_pos.ToString 'bypass % open

'Send result calculations to the outputs

If CheckBox3.Checked Then

Double.TryParse(TextBox1.Text, flow1) 'Flow

Double.TryParse(TextBox2.Text, Press\_in) 'P\_inlet

Double.TryParse(TextBox3.Text, p2) 'P\_outlet

Double.TryParse(TextBox23.Text, t1) 'Tinlet

'keeps things with the selected output range-----

If flow1 > NumericUpDown5.Maximum Then flow1 = NumericUpDown5.Maximum

If flow1 < NumericUpDown5.Minimum Then flow1 = NumericUpDown5.Minimum

If Press\_in > NumericUpDown10.Maximum Then Press\_in = NumericUpDown10.Maximum

If Press\_in < NumericUpDown10.Minimum Then Press\_in = NumericUpDown10.Minimum

If p2 > NumericUpDown14.Maximum Then p2 = NumericUpDown14.Maximum

If p2 < NumericUpDown14.Minimum Then p2 = NumericUpDown14.Minimum

If t1 > NumericUpDown15.Maximum Then t1 = NumericUpDown15.Maximum

If t1 < NumericUpDown15.Minimum Then t1 = NumericUpDown15.Minimum

NumericUpDown5.Value = CDec(flow1) 'Flow

NumericUpDown14.Value = CDec(Press\_in) 'P\_inlet

NumericUpDown15.Value = CDec(p2) 'P\_outlet

NumericUpDown10.Value = CDec(t1) 'Tinlet

End If

GetIO() 'Get the feedback value

If CheckBox3.Checked Then SetOut() 'Set the output values

Update\_calc\_screen()

Draw\_Chart1()

'------------- setpoint ---------

SLV1 = NumericUpDown35.Value

SLV2 = SLV1 \* (1 + NumericUpDown39.Value / 100)

SLV3 = SLV2 \* (1 + NumericUpDown40.Value / 100)

setpoint = Convert\_R(SLV3)

'------ pocess value----

TextBox27.Text = TextBox44.Text

Double.TryParse(TextBox44.Text, pv) 'Process Variable actual

PID\_controller(setpoint, pv)

'----- present-------

TextBox46.Text = SLV2.ToString("0")

TextBox55.Text = Convert\_R(SLV2).ToString("0.00")

TextBox47.Text = SLV3.ToString("0")

TextBox48.Text = setpoint.ToString("0.00")

TextBox54.Text = setpoint.ToString("0.00")

End Sub

Private Sub GetIO()

Dim GetIo(4) As Byte 'Get-Voltage, see Page 22 en 23

'--------- update time on sceen-------

time += Timer1.Interval \* 0.001 '[msec]--->[sec]

Label1.Text = time.ToString("000.0")

If SerialPort1.IsOpen Then

'--------- prepare request to Lucid Control------

GetIo(1) = &H48 'OPC= GetIoGroup

GetIo(2) = &H1 'Channel 1

If RadioButton5.Checked Then

GetIo(3) = &H1D 'Voltage range 0-100,000,000 mV (4Bytes)

Label102.Text = "[Volt]"

Else

GetIo(3) = &H23 'Amp range 0-1,000,000 mAmp (4Bytes)

Label102.Text = "[mAmp]"

End If

GetIo(4) = &H0 'LEN

'-------LucidControl AI4, Input module -------------

Try

SerialPort1.Write(GetIo, 1, 4)

Thread.Sleep(5)

Catch generatedExceptionName As TimeoutException

End Try

End If

End Sub

Private Sub Button9\_Click(sender As Object, e As EventArgs) Handles Button9.Click

'Find ports----------

combo\_Port1.SelectedIndex = -1 'To instruments in ACS panel

combo\_Port1.Items.Clear()

combo\_Port2.SelectedIndex = -1 'From Bypass valve

combo\_Port2.Items.Clear()

Serial\_setup()

End Sub

Private Sub Serial\_setup() 'Serial ports setup

combo\_Baud.Items.Clear()

If (SerialPort1.IsOpen = True) Then 'Write to Instruments

SerialPort1.DiscardInBuffer() 'Preventing exceptions

SerialPort1.Close()

End If

If (SerialPort2.IsOpen = True) Then 'Read from Bypass Valve

SerialPort2.DiscardInBuffer() 'Preventing exceptions

SerialPort2.Close()

End If

Try

myPort = SerialPort.GetPortNames() 'Get all com ports available

For Each port In myPort

combo\_Port1.Items.Add(port)

combo\_Port2.Items.Add(port)

Next port

combo\_Port1.Text = CType(combo\_Port1.Items.Item(0), String) 'Set cmbPort text to the first COM port detected

combo\_Port2.Text = CType(combo\_Port2.Items.Item(0), String) 'Set cmbPort text to the first COM port detected

Catch ex As Exception

MsgBox("No COM ports detected")

End Try

combo\_Baud.Items.Add(9600) 'Populate the cmbBaud Combo box to common baud rates used

combo\_Baud.Items.Add(19200)

combo\_Baud.Items.Add(38400)

combo\_Baud.Items.Add(57600)

combo\_Baud.Items.Add(115200)

combo\_Baud.SelectedIndex = 0 'Set cmbBaud text to 9600 Baud

End Sub

Private Sub BtnConnect\_Click(sender As System.Object, e As System.EventArgs) Handles Button12.Click

'Connect

If combo\_Port1.Text.Length = 0 Then

MsgBox("Sorry, did not find any connected Lucid Controllers")

Else

SerialPort1.PortName = combo\_Port1.Text 'Set SerialPort1 to the selected COM port at startup

SerialPort1.BaudRate = CInt(combo\_Baud.Text) 'Set Baud rate to the selected value on

SerialPort1.Parity = Parity.None

SerialPort1.StopBits = StopBits.One

SerialPort1.Handshake = Handshake.None

SerialPort1.DataBits = 8

SerialPort1.Encoding = Encoding.GetEncoding(28591) 'important otherwise it will not work

SerialPort2.PortName = combo\_Port2.Text 'Set SerialPort2 to the selected COM port at startup

SerialPort2.BaudRate = CInt(combo\_Baud.Text) 'Set Baud rate to the selected value on

SerialPort2.Parity = Parity.None

SerialPort2.StopBits = StopBits.One

SerialPort2.Handshake = Handshake.None

SerialPort2.DataBits = 8 'Open our serial port

SerialPort2.Encoding = Encoding.GetEncoding(28591) 'important otherwise it will not work

Try

If CheckBox2.Checked Then SerialPort1.Open()

If CheckBox4.Checked Then SerialPort2.Open()

Button12.Enabled = False 'Disable Connect button

Button12.BackColor = Color.Yellow

Button12.Text = "OK connected"

btnDisconnect.Enabled = True 'and Enable Disconnect button

Catch ex As Exception

MsgBox("Error 654 Open: " & ex.Message)

End Try

Label94.BackColor = CType(IIf(SerialPort1.IsOpen, Color.Yellow, Color.Red), Color) 'Port1

Label95.BackColor = CType(IIf(SerialPort2.IsOpen, Color.Yellow, Color.Red), Color) 'Port2

End If

End Sub

Private Sub BtnDisconnect\_Click(sender As System.Object, e As System.EventArgs) Handles btnDisconnect.Click

'Disconnect ports

Try

SerialPort1.DiscardInBuffer()

SerialPort1.Close() 'Close our Serial Port

SerialPort1.Dispose()

SerialPort2.DiscardInBuffer()

SerialPort2.Close() 'Close our Serial Port

SerialPort2.Dispose()

Button12.Enabled = True

Button12.BackColor = Color.Red

Label94.BackColor = Color.White 'Port 1

Label95.BackColor = Color.White 'Port 2

Button12.Text = "Connect"

btnDisconnect.Enabled = False

Catch ex As Exception

MsgBox("Error 104 Open: " & ex.Message)

End Try

End Sub

Private Sub Button4\_Click(sender As Object, e As EventArgs) Handles Button4.Click

Reset()

End Sub

Private Sub SerialPort1\_DataReceived(sender As System.Object, e As SerialDataReceivedEventArgs) Handles SerialPort1.DataReceived

Dim intext\_hex As String = String.Empty

Dim intext As String = String.Empty

Dim status\_code As String = String.Empty

Dim bigE As String = String.Empty

Dim status\_OK As String = "00"

Dim Value\_channel\_0\_hex As String 'Lucid-Control AI4, 10V module

Dim Value\_channel\_0\_dec As Double 'Lucid-Control AI4, 10V module

Try

intext\_hex = SerialPort1.ReadExisting 'Read the data

intext = String\_ascii\_to\_Hex\_ascii(intext\_hex) 'Convert data to hex

Catch generatedExceptionName As TimeoutException

End Try

'--------- Status Communication-------

If intext.Length > 2 Then status\_code = intext.Substring(0, 2)

'---------- instring OK then continue------

If String.Equals(status\_code, status\_OK) And (intext.Length = 12) Then

'---------- Test value -----------

'intext = "0004" & "D0121300" 'Test value +1.2500 Volt/mA

'intext = "0004" & "A0252600" 'Test value +2.5000 Volt/mA

'intext = "0004" & "404b4c00" 'Test value +5.000 Volt/mA

'intext = "0004" & "40548900" 'Test value +9.000 Volt/mA

'intext = "0004" & "002D3101" 'Test value +20.000 Volt/mA

'intext = "0004" & "C0B4B3FF" 'Test value -5.000 Volt/mA

'intext = "0004" & "39FFFFFF" 'Test value 0.000199 Volt/mA

Value\_channel\_0\_hex = intext.Substring(4, 8) 'Skip the 4 status bytes

'---- The received value is little-Endian (now reverse order)-----

bigE = Value\_channel\_0\_hex.Substring(6, 2)

bigE &= Value\_channel\_0\_hex.Substring(4, 2)

bigE &= Value\_channel\_0\_hex.Substring(2, 2)

bigE &= Value\_channel\_0\_hex.Substring(0, 2)

'---------- calc the value---------

Value\_channel\_0\_dec = Convert.ToInt32(bigE, 16) '[micro(V/A)Volt] Channel 0

Value\_channel\_0\_dec /= 10 ^ 6 '[micro(V/A)-->Volt]

'--------- Present data--------------

Try

Invoke(Sub() TextBox38.Text = bigE) 'Hex 4 Bytes valueTextBox36

Invoke(Sub() TextBox37.Text = Value\_channel\_0\_dec.ToString("0.000")) 'Value

Invoke(Sub() TextBox26.Text &= intext & " ")

'----------- bypass valve position 0-100%-----------

If RadioButton5.Checked Then

\_bypass\_pos = CInt(Value\_channel\_0\_dec / 10 \* 100) 'Volt input

Else

\_bypass\_pos = CInt((Value\_channel\_0\_dec - 4) / 16 \* 100) 'Amp input

End If

If \_bypass\_pos > 100 Then \_bypass\_pos = 100 'max 100% open

If \_bypass\_pos < 0 Then \_bypass\_pos = 0 'min 0% open

\_bypass\_ma = \_bypass\_pos / 100 \* 16 + 4 '[%]--> [4-20 mA]

Catch ex As Exception

End Try

Else

\_counter += 1

Invoke(Sub() Label121.Text = " Error count" & \_counter.ToString)

SerialPort1.DiscardInBuffer() 'empty inbuffer

End If

End Sub

Private Sub SerialPort2\_DataReceived(sender As Object, e As SerialDataReceivedEventArgs) Handles SerialPort2.DataReceived

'-------- Keep the buffer empty----------

Dim intext\_hex2 As String = String.Empty

Try

intext\_hex2 = SerialPort2.ReadExisting 'Read the data

Catch generatedExceptionName As TimeoutException

End Try

End Sub

Public Function String\_ascii\_to\_Hex\_ascii(str As String) As String

Dim byteArray() As Byte

Dim hexNumbers As Text.StringBuilder = New Text.StringBuilder

byteArray = System.Text.Encoding.BigEndianUnicode.GetBytes(str)

For i As Integer = 1 To byteArray.Length - 1 Step 2

hexNumbers.Append(byteArray(i).ToString("x2"))

Next

Return (hexNumbers.ToString())

End Function

Public Function String\_Hex\_to\_ascii(Data As String) As String

Dim com As String = String.Empty

'see http://stackoverflow.com/questions/14017007/how-to-convert-a-hexadecimal-value-to-ascii

'Data = "484558" 'Example string Ascii HEX= 0x48 0x45 0x58 = 72 69 88

For x = 0 To Data.Length - 1 Step 2

com &= ChrW(CInt("&H" & Data.Substring(x, 2)))

Next

Return com

End Function

Public Function HexStringToByteArray(hexString As String) As Byte()

Dim com As String = String.Empty

'see http://www.vbforums.com/showthread.php?643593-Hex-String-to-Byte-Array

'hexString= "01050001FFFF8FFB" 'Example string

Dim length As Integer = hexString.Length

Dim upperBound As Integer = length \ 2

Dim bytes(upperBound) As Byte

If length Mod 2 = 0 Then

upperBound -= 1

Else

hexString = "0" & hexString

End If

For i As Integer = 0 To upperBound

bytes(i) = Convert.ToByte(hexString.Substring(i \* 2, 2), 16)

Next

Return bytes

End Function

Private Sub Update\_calc\_screen()

Dim Range(4) As String

Dim K\_sys, K\_bypass, k\_sum, K100, valve\_open, dp, ro As Double

Dim A, B, C, Qv\_in, Qv\_out, A1 As Double

Dim Pin, Pout As Double 'fan inlet and outlet

Dim Tin, Tout As Double 'fan inlet and outlet

Dim γ As Double

Dim p\_time, period, amplitude As Double

Dim Qv\_a, Qv\_b As Double

Dim R\_control, PID\_input As Double

'Range is required for converting the signal to and from 4-20 mAmp

Range(0) = CType(NumericUpDown28.Value - NumericUpDown27.Value, String) 'Flow

Range(1) = CType(NumericUpDown29.Value - NumericUpDown30.Value, String) 'Temp

Range(2) = CType(NumericUpDown31.Value - NumericUpDown32.Value, String) 'Pressure in

Range(3) = CType(NumericUpDown36.Value - NumericUpDown37.Value, String) 'Pressure out

Range(4) = CType(NumericUpDown13.Value - NumericUpDown34.Value, String) 'Valve position

NumericUpDown33.Value = CDec(\_bypass\_pos) 'Valve postion from extern Control panel

ro = NumericUpDown19.Value 'Density [kg/Am3]

A = NumericUpDown17.Value 'Fan Curve [-]

B = NumericUpDown16.Value 'Fan Curve [-]

C = NumericUpDown20.Value 'Fan Curve [-]

K100 = NumericUpDown21.Value 'K-value at 100% open [-]

Pin = NumericUpDown18.Value 'Pressure inlet fan [Pa]

Tin = NumericUpDown23.Value 'Temp inlet fan [c]

γ = NumericUpDown22.Value 'Poly tropic exponent γ

'----------- Feedback Extern or Intern--------

If CheckBox5.Checked Then

valve\_open = NumericUpDown38.Value / 100 'Position bypass valve [%] (Intern)

Else

valve\_open = NumericUpDown33.Value / 100 'Position bypass valve [%] (Extern)

End If

If ro > 0 Then 'to prevent exceptions

'----- step 1 determin the K values----

K\_bypass = K100 \* valve\_open

period = NumericUpDown7.Value

amplitude = NumericUpDown2.Value

p\_time = time Mod period

Select Case True

Case RadioButton1.Checked 'Feedback from ASC controller

K\_sys = NumericUpDown25.Value

Case RadioButton2.Checked 'Square wave

If (p\_time > (period / 2)) Then

K\_sys = NumericUpDown25.Value + amplitude / 2

Else

K\_sys = NumericUpDown25.Value - amplitude / 2

End If

Case RadioButton3.Checked 'Sine

K\_sys = NumericUpDown25.Value + (amplitude / 2) \* Sin(p\_time / period \* 2 \* PI)

Case RadioButton4.Checked 'Saw tooth

K\_sys = NumericUpDown25.Value - (amplitude / 2) + amplitude \* p\_time / period

End Select

TextBox5.Text = K\_sys.ToString("0.0")

TextBox6.Text = K\_bypass.ToString("0.0")

'----- step 2 determine qv---

'------ ABC formula ----------

k\_sum = K\_sys + K\_bypass

A1 = A - 1 / k\_sum ^ 2

Qv\_a = (-B + (Sqrt(B ^ 2 - 4 \* A1 \* C))) / (2 \* A1)

Qv\_b = (-B - (Sqrt(B ^ 2 - 4 \* A1 \* C))) / (2 \* A1)

Qv\_in = CDbl(IIf(Qv\_a > 0, Qv\_a, Qv\_b))

If CheckBox6.Checked Then Qv\_in = First\_order(Qv\_in, NumericUpDown47.Value, Timer1.Interval \* 0.001, 0)

'----- step 3 determine new dp---

dp = ro \* (A \* Qv\_in ^ 2 + B \* Qv\_in + C) 'Fan curve

'----- step 4 determine Temp outlet fan ---

Tout = Tin \* (1 + dp / Pin) ^ ((γ - 1) / γ)

'----- step 5 determine Pressure outlet fan ---

Pout = Pin + dp

'----- step 6 determine Discharge flow fan ---

Qv\_out = Qv\_in \* (Pin / Pout) ^ (1 / γ)

'------ calc R\_controller input--------

R\_control = ro \* Qv\_in ^ 2 / dp

'----- present the data ----

TextBox7.Text = K\_bypass.ToString("0.00") 'Resistance Bypass valve

TextBox8.Text = Qv\_out.ToString("0")

TextBox10.Text = Tin.ToString("0")

TextBox11.Text = Range(0).ToString 'Range

TextBox12.Text = Range(1).ToString 'Range

TextBox13.Text = Range(2).ToString 'Range press inlet flange

TextBox24.Text = Range(3).ToString 'Range Press outlet flange

TextBox32.Text = Range(4).ToString 'Range Valve

TextBox14.Text = K\_sys.ToString("0.00") 'Resistance Total system

TextBox15.Text = Qv\_in.ToString("0")

TextBox21.Text = Pin.ToString("0") 'Pressure inlet flange

TextBox9.Text = Pout.ToString("0") 'Pressure outlet flange

TextBox29.Text = dp.ToString("0") 'dp

TextBox25.Text = Tout.ToString("0.0") 'Outlet temperature

TextBox30.Text = R\_control.ToString("0") 'R value

PID\_input = Convert\_R(R\_control) 'PID input

TextBox44.Text = PID\_input.ToString("0.000") 'R (controller input)

End If

'-------- Surge warning-Alarm-----------

'wanrning number must be bigger then alarm number

If NumericUpDown46.Value < NumericUpDown1.Value + 200 Then

NumericUpDown46.Value = NumericUpDown1.Value + 200

End If

Select Case True

Case Qv\_in > NumericUpDown1.Value And Qv\_in < NumericUpDown46.Value

TextBox15.BackColor = Color.Orange

NumericUpDown46.BackColor = Color.Orange

Case Qv\_in < NumericUpDown1.Value

TextBox15.BackColor = Color.Red

NumericUpDown1.BackColor = Color.Red

Case Else

TextBox15.BackColor = Color.LightGreen

NumericUpDown46.BackColor = Color.White

NumericUpDown1.BackColor = Color.White

End Select

'---------- calc output currents--------------

Cout(1) = Convert\_Units\_to\_mAmp("Flow", Qv\_in)

Cout(2) = Convert\_Units\_to\_mAmp("Pressure\_in", Pin)

Cout(3) = Convert\_Units\_to\_mAmp("Pressure\_out", Pout)

Cout(4) = Convert\_Units\_to\_mAmp("Temperature", Tin)

'--------present [4-20 mAmp]-------------

TextBox1.Text = Cout(1).ToString("0.0") 'Flow inlet/out Actual [Am3/hr]

TextBox2.Text = Cout(2).ToString("0.0") 'Pressure in [Pa]

TextBox3.Text = Cout(3).ToString("0.0") 'Pressure out [Pa]

TextBox23.Text = Cout(4).ToString("0.0") 'Temp fan in [c]

'------------[0-10V]----------------

Vout(1) = Convert\_mAmp\_to\_V(Cout(1))

Vout(2) = Convert\_mAmp\_to\_V(Cout(2))

Vout(3) = Convert\_mAmp\_to\_V(Cout(3))

Vout(4) = Convert\_mAmp\_to\_V(Cout(4))

'--------present [0-10 Volt]-------------

TextBox40.Text = Vout(1).ToString("0.0") 'Flow inlet/out Actual [Am3/hr]

TextBox41.Text = Vout(2).ToString("0.0") 'Pressure in [Pa]

TextBox42.Text = Vout(3).ToString("0.0") 'Delta P [Pa]

TextBox43.Text = Vout(4).ToString("0.0") 'Temp fan in [c]

End Sub

'------- Convert fysical units ----> mAmp's

Private Function Convert\_Units\_to\_mAmp(outType As String, value As Double) As Double

Dim results, range, value\_4ma As Double

Select Case outType

Case "Flow"

value\_4ma = NumericUpDown27.Value

Double.TryParse(TextBox11.Text, range)

results = (value - value\_4ma) / range \* 16.0 + 4.0

Case "Temperature"

value\_4ma = NumericUpDown30.Value

Double.TryParse(TextBox12.Text, range)

results = (value - value\_4ma) / range \* 16.0 + 4.0

Case "Pressure\_in"

value\_4ma = NumericUpDown32.Value

Double.TryParse(TextBox13.Text, range)

results = (value - value\_4ma) / range \* 16.0 + 4.0

Case "Pressure\_out"

value\_4ma = NumericUpDown37.Value

Double.TryParse(TextBox13.Text, range)

results = (value - value\_4ma) / range \* 16.0 + 4.0

Case "Valve-positioner"

value\_4ma = NumericUpDown34.Value

Double.TryParse(TextBox32.Text, range)

' MessageBox.Show(range.ToString)

results = (value - value\_4ma) / range \* 16.0 + 4.0

Case Else

MessageBox.Show("Oops error in Convert\_Units\_to\_mAmp function")

End Select

Return (results)

End Function

'------- Convert [4-20 mA] ----> [0-10 V]

Private Function Convert\_mAmp\_to\_V(ma\_value As Double) As Double

Dim result As Double

result = (ma\_value - 4) / 16 \* 10

Return (result)

End Function

'------- Convert from mAmp's ----> fysical units

Private Function Convert\_mAmp\_to\_Units(outType As String, value As Double) As Double

Dim results, range, value\_4ma As Double

Select Case outType

Case "Flow"

value\_4ma = NumericUpDown27.Value

Double.TryParse(TextBox11.Text, range)

results = (value - 4) / 16 \* range + value\_4ma

Case "Temperature"

value\_4ma = NumericUpDown30.Value

Double.TryParse(TextBox12.Text, range)

results = (value - 4) / 16 \* range + value\_4ma

Case "Pressure\_in"

value\_4ma = NumericUpDown32.Value

Double.TryParse(TextBox13.Text, range)

results = (value - 4) / 16 \* range + value\_4ma

Case "Pressure\_out"

value\_4ma = NumericUpDown37.Value

Double.TryParse(TextBox24.Text, range)

results = (value - 4) / 16 \* range + value\_4ma

Case "Valve-positioner"

value\_4ma = NumericUpDown34.Value

Double.TryParse(TextBox32.Text, range)

results = (value - 4) / 16 \* range + value\_4ma

Case Else

MessageBox.Show("Oops error in Calc\_in function")

End Select

Return (results)

End Function

Public Function First\_order(kx As Double, τ As Double, dt As Double, id As Integer) As Double

'τ.(dy/dt) + y = Kx

'Hieruit volgt dy= (Kx-y) \* dt/τ

'kx = input, tau= time constant, dt= time step, ident = identification number

'y is de output

Dim dy, y As Double 'output

dy = (kx - \_yold(id)) \* dt / τ

y = \_yold(id) + dy

\_yold(id) = y

Return (y)

End Function

Private Sub Button1\_Click(sender As Object, e As EventArgs) Handles Button1.Click, NumericUpDown6.ValueChanged, NumericUpDown4.ValueChanged, NumericUpDown3.ValueChanged, NumericUpDown42.ValueChanged, NumericUpDown41.ValueChanged, NumericUpDown11.ValueChanged, NumericUpDown45.ValueChanged, NumericUpDown44.ValueChanged, NumericUpDown43.ValueChanged

Dim K\_ro, k\_dp, K\_flow, Ks As Double

Dim K100\_ro, k100\_dp, K100\_flow, K100 As Double

Dim R\_ro, R\_dp, R\_flow, R\_surge, R\_alt As Double

'Calculate Ksys @ work point

K\_flow = NumericUpDown6.Value '[Am3/hr]

k\_dp = NumericUpDown3.Value '[Pa]

K\_ro = NumericUpDown4.Value '[kg/m3]

Ks = Calc\_K(k\_dp, K\_ro, K\_flow) '[m2]

TextBox4.Text = Ks.ToString("0.0")

'Calculate K100% (bypass) @ ... point

K100\_flow = NumericUpDown43.Value '[Am3/hr]

k100\_dp = NumericUpDown44.Value '[Pa]

K100\_ro = NumericUpDown45.Value '[kg/m3]

K100 = Calc\_K(k100\_dp, K100\_ro, K100\_flow) '[m2]

TextBox52.Text = K100.ToString("0.0")

'Calculate R @ surge point point (=SLV1)

R\_dp = NumericUpDown41.Value '[Pa]

R\_ro = NumericUpDown42.Value '[kg/m3]

R\_flow = NumericUpDown11.Value '[Am3/hr]

R\_surge = Calc\_R(R\_dp, R\_ro, R\_flow) '[m4]

TextBox51.Text = R\_surge.ToString("0.0")

R\_alt = Convert\_R(R\_surge) '[m2]

TextBox53.Text = R\_alt.ToString("0.00")

End Sub

Private Function Calc\_R(dp As Double, ro As Double, flow As Double) As Double

Dim R\_value As Double

'Calculate the R value, dp [Pa], ro [kg/m3], flow [Am3/hr]

R\_value = ro \* flow ^ 2 / dp

Return (R\_value)

End Function

Private Function Calc\_K(dp As Double, ro As Double, flow As Double) As Double

Dim k\_value As Double

'Calculate the K value, dp [Pa], ro [kg/m3], flow [Am3/hr]

'By defenition K=R^0.5

k\_value = flow \* Sqrt(ro / dp)

Return (k\_value)

End Function

Private Function Convert\_R(rs As Double) As Double

Dim a, n As Double

a = NumericUpDown54.Value 'Factor

n = NumericUpDown55.Value 'Power

'Required to have a more stabe deviation in the OID controller

Return ((a / rs) ^ n)

End Function

Private Sub Button5\_Click(sender As Object, e As EventArgs) Handles Button5.Click

Safe\_to\_file()

End Sub

Private Sub Button8\_Click(sender As Object, e As EventArgs) Handles Button8.Click

TextBox26.Clear()

End Sub

Private Sub Output\_set\_to\_V()

GroupBox5.Text = "Outputs test values 0-5 Volt"

NumericUpDown5.Minimum = 0

NumericUpDown5.Maximum = 10

NumericUpDown10.Minimum = 0

NumericUpDown10.Maximum = 10

NumericUpDown14.Minimum = 0

NumericUpDown14.Maximum = 10

NumericUpDown15.Minimum = 0

NumericUpDown15.Maximum = 10

Check\_output\_limits() 'stay inside the limits

Label19.Text = "[V]"

Label23.Text = "[V]"

Label109.Text = "[V]"

Label110.Text = "[V]"

End Sub

Private Sub Output\_set\_to\_mA()

GroupBox5.Text = "Outputs test values 4-20 mAmp"

'---- max and min

NumericUpDown5.Minimum = 4

NumericUpDown5.Maximum = 20

NumericUpDown10.Minimum = 4

NumericUpDown10.Maximum = 20

NumericUpDown14.Minimum = 4

NumericUpDown14.Maximum = 20

NumericUpDown15.Minimum = 4

NumericUpDown15.Maximum = 20

Check\_output\_limits() 'stay inside the limits

Label19.Text = "[mA]"

Label23.Text = "[mA]"

Label109.Text = "[mA]"

Label110.Text = "[mA]"

End Sub

Private Sub Check\_output\_limits()

'---- minimum

If NumericUpDown5.Value < NumericUpDown5.Minimum Then NumericUpDown5.Value = NumericUpDown5.Minimum

If NumericUpDown10.Value < NumericUpDown10.Minimum Then NumericUpDown10.Value = NumericUpDown10.Minimum

If NumericUpDown14.Value < NumericUpDown14.Minimum Then NumericUpDown14.Value = NumericUpDown14.Minimum

If NumericUpDown15.Value < NumericUpDown15.Minimum Then NumericUpDown15.Value = NumericUpDown15.Minimum

'---- maximum

If NumericUpDown5.Value > NumericUpDown5.Maximum Then NumericUpDown5.Value = NumericUpDown5.Maximum

If NumericUpDown10.Value > NumericUpDown10.Maximum Then NumericUpDown10.Value = NumericUpDown10.Maximum

If NumericUpDown14.Value > NumericUpDown14.Maximum Then NumericUpDown14.Value = NumericUpDown14.Maximum

If NumericUpDown15.Value > NumericUpDown15.Maximum Then NumericUpDown15.Value = NumericUpDown15.Maximum

End Sub

Private Sub Button11\_Click(sender As Object, e As EventArgs) Handles Button11.Click

If Timer1.Enabled Then

Timer1.Stop() 'Freeze

Button11.Text = "Thaw"

Else

Timer1.Start() 'Freeze

Button11.Text = "Freeze"

End If

End Sub

Private Sub Button3\_Click(sender As Object, e As EventArgs) Handles Button3.Click

Reset()

End Sub

Private Sub NumericUpDown15\_ValueChanged(sender As Object, e As EventArgs) Handles NumericUpDown5.ValueChanged, NumericUpDown15.ValueChanged, NumericUpDown14.ValueChanged, NumericUpDown10.ValueChanged

SetOut()

End Sub

Private Sub Button7\_Click(sender As Object, e As EventArgs) Handles Button7.Click, RadioButton8.CheckedChanged

If RadioButton8.Checked Then

Output\_set\_to\_V()

Else

Output\_set\_to\_mA()

End If

End Sub

Private Sub Button10\_Click(sender As Object, e As EventArgs) Handles NumericUpDown53.ValueChanged, NumericUpDown52.ValueChanged, NumericUpDown51.ValueChanged, NumericUpDown50.ValueChanged, NumericUpDown49.ValueChanged

Dim volume, mol, temp, fan\_flow, fan\_dp, τ As Double

volume = NumericUpDown53.Value '[m3] system volume

mol = NumericUpDown52.Value / 1000 '[kg/mol]

temp = NumericUpDown51.Value '[celsius]

fan\_flow = NumericUpDown49.Value / 3600 '[Am3/sec]

fan\_dp = NumericUpDown50.Value '[Pa]

τ = Calc\_tau(volume, mol, temp, fan\_flow, fan\_dp)

TextBox49.Text = τ.ToString("0.0")

End Sub

Private Sub PID\_controller(setpoint As Double, pv As Double)

Dim deviation, PID\_output\_pro, dt As Double

Dim Kp, Ki, Kd As Double 'Setting PID controller

Dim ddev As Double

Dim output\_dev As Double

'------ Setting PID controller --------

Kp = NumericUpDown9.Value

Ki = NumericUpDown8.Value

Kd = NumericUpDown12.Value

'------ time interval-----

dt = Timer1.Interval \* 0.001 '[sec]

deviation = (pv - setpoint) '[m4]

If CheckBox1.Checked Then deviation \*= -1 'reverse direction

If deviation > 10000 Then deviation = 0.001 'for startup

If deviation < -10000 Then deviation = 0.001 'for startup

'-----------Intern PID control ON--------------

If CheckBox5.Checked Then

Label133.Visible = True

Label132.Visible = False

Label64.Text = "Intern PID controller feed back to built-in simulation"

Label66.Text = "Intern PID controller feed back to built-in simulation"

Label162.Text = "Intern PID contr. is ON"

Button2.Enabled = False

NumericUpDown5.Enabled = False

NumericUpDown10.Enabled = False

NumericUpDown14.Enabled = False

NumericUpDown15.Enabled = False

Else

Label132.Visible = True

Label133.Visible = False

Label64.Text = "Extern panel feedback to built-in simulation"

Label66.Text = "Extern panel feedback to built-in simulation"

Label162.Text = "Intern PID contr. is OFF"

Button2.Enabled = True

NumericUpDown5.Enabled = True

NumericUpDown10.Enabled = True

NumericUpDown14.Enabled = True

NumericUpDown15.Enabled = True

End If

'----------- start calculating---------------

If pv > 0 And CheckBox5.Checked Then

ddev = deviation - \_last\_output 'change in PID output

\_last\_deviation = deviation

'=========== Calculate PID controller==========

Double.TryParse(TextBox31.Text, PID\_output\_pro)

\_Pterm = Kp \* deviation 'P action

If Ki > 0 Then 'I-action

\_Iterm = \_Iterm + Ki \* deviation \* dt

Else

\_Iterm = 0

End If

If \_Iterm > 100 Then \_Iterm = 100 'anti-Windup

If \_Iterm < 0 Then \_Iterm = 0 'anti-Windup

\_Dterm = Kd \* ddev / dt 'D action

PID\_output\_pro = \_Pterm + \_Iterm + \_Dterm

'--------- limit the output----------

If PID\_output\_pro > 100 Then PID\_output\_pro = 100

If PID\_output\_pro < 0 Then PID\_output\_pro = 0

'--- safety net vased on SLV2--------

Dim slv2 As Double

If CheckBox8.Checked Then 'Safet net is ON

Double.TryParse(TextBox46.Text, slv2)

If pv > Convert\_R(slv2) Then

PID\_output\_pro = 100

Label125.BackColor = Color.AliceBlue

Else

Label125.BackColor = Color.White

End If

End If

'------- bypass valve speed limiter -------

Dim travelt As Double

Dim maxc As Double

If CheckBox7.Checked Then 'Speed limitter on

travelt = NumericUpDown48.Value

maxc = 100 \* dt / travelt

output\_dev = PID\_output\_pro - \_last\_output 'change in output

If output\_dev > maxc Then PID\_output\_pro = \_last\_output + maxc

If output\_dev < -maxc Then PID\_output\_pro = \_last\_output - maxc

End If

\_last\_output = PID\_output\_pro

'=============================================

\_PID\_output\_ma = Convert\_Units\_to\_mAmp("Valve-positioner", PID\_output\_pro)

NumericUpDown38.Value = CDec(PID\_output\_pro)

'---------- present results ------------

TextBox28.Text = deviation.ToString("0.00")

TextBox31.Text = PID\_output\_pro.ToString("0.00") 'output [%]

TextBox22.Text = \_PID\_output\_ma.ToString("0.00") 'output [mAmp]

TextBox33.Text = \_Pterm.ToString("0.00")

TextBox34.Text = \_Iterm.ToString("0.00")

TextBox35.Text = \_Dterm.ToString("0.00")

End If

End Sub

Private Sub Safe\_to\_file()

Dim file\_name As String

Dim dirpath\_Home As String = "C:\Temp\"

file\_name = dirpath\_Home & "ASC-Chart\_" & DateTime.Now.ToString("yyyy-MM-dd\_HH\_mm\_ss") & ".jpeg"

If Directory.Exists(dirpath\_Home) Then

Chart1.SaveImage(file\_name, Imaging.ImageFormat.Jpeg)

Else

MessageBox.Show("File is NOT saved" & vbCrLf & "Directory doen not exist" & vbCrLf & "Please create " & dirpath\_Home)

End If

End Sub

Private Function Calc\_tau(volume As Double, mol As Double, Temp As Double, fan\_flow As Double, fan\_dp As Double) As Double

'Calculates time constant filling the ducting

'See page 64-65, Regeltechniek, 7e druk

'de klep is vervangen door een fan

'de fan wordt lineair geacht

'R\_fan = dp/Volume\_Flow zie formule (4.3)

'Capaciteit tank-system-duct zie formule (4.5)

Dim R, τ As Double

Dim cap\_duct As Double

Dim R\_fan As Double

R = 8.314 / mol 'Specific gas constant

Temp += 273 '[Kelvin]

cap\_duct = volume / (R \* Temp) '[..] capacity of the ducting

R\_fan = fan\_dp / fan\_flow '[..] fan resistance

τ = R\_fan \* cap\_duct '[s]

Return (τ)

End Function

End Class