実技課題　　解答及び解説

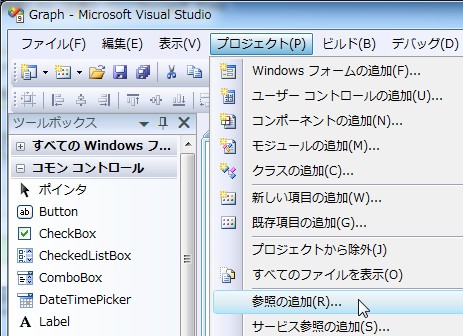
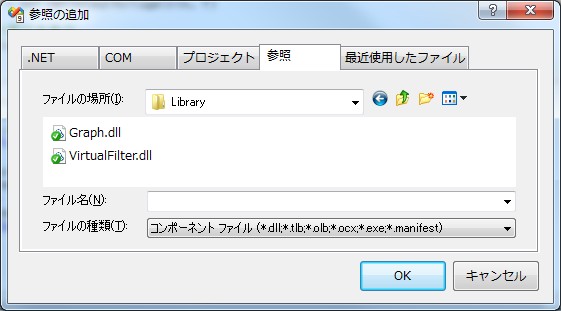
実技課題　解答及び解説

「パソコンを用いた計測制御システムの製作（Visual Basic）」

　解答例を以下に示す。

（１）事前の設定(別途ファイルに格納)

1. プロジェクトを開いたあと、グラフ描画用のDLL(Graph.dll)をメニューバーの「プロジェクト」→「参照の追加」から以下のダイアログを表示し、「参照」のタブで、配布した"Graph.dll"を選択する。

1. "Polytech.Windows.Form"名前空間でGraphクラスを定義しているため、Importsステートメントで"Polytech.Windows.Form"名前空間を追加すること。

'名前空間のインポート

Imports System.Drawing

Imports System.Drawing.Drawing2D

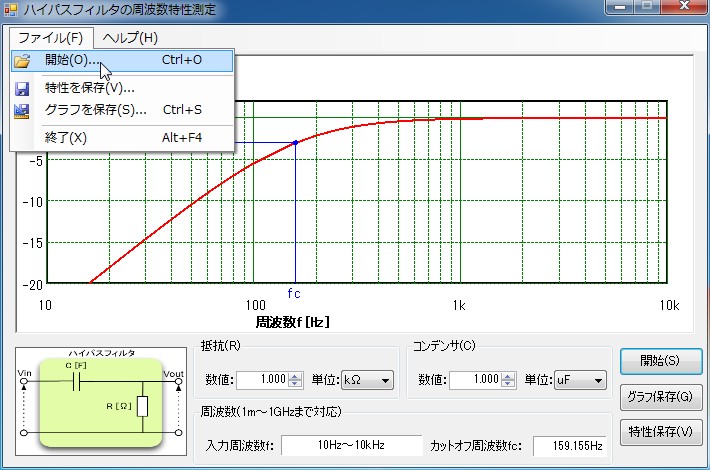
Imports System.Drawing.Imaging

'グラフ用

Imports Polytech.Windows.Form

1. コントロールと各機能

|  |  |  |
| --- | --- | --- |
| コントロール | オブジェクト名 | 備考 |
| Form | Form1 | Graphクラスの初期化と軸の描画、初期描画の登録 |
| Button | GraphInit | 初期描画状態を呼び出す |
| SaveCSV | 周波数特性をCSVファイル等で保存 |
| SaveGraph | グラフを画像ファイルとして保存 |
| NumericUpdown | InputNumberR | 抵抗値の数値を入力 |
| InputNumberC | 静電容量の数値を入力 |
| ListBox | ListPrefixR | 抵抗値の単位を入力 |
| ListPrefixC | 静電容量の単位を入力 |
| Label | SimlateShowFreq | カットオフ周波数の理論値を表示 |
| MeasureShowFreq | カットオフ周波数の測定値を表示 |
| DifferenceFreqPercemt | 理論値と測定値の誤差を%表示 |
| PictureBox | GraphDisplay | グラフの描画対象 |



解答例のソースコード(フォーム)

|  |  |
| --- | --- |
| 行 | プログラム |
|  | '名前空間のインポート |
|  | Imports System |
|  | Imports System.Text |
|  | Imports System.IO |
|  | Imports System.Windows.Forms |
|  | Imports System.Drawing |
|  | Imports System.Drawing.Drawing2D |
|  | Imports System.Drawing.Imaging |
|  | Imports System.Math |
|  | 'グラフ用 |
|  | Imports Polytech.Windows.Form |
|  |  |
|  | Public Class GraphForm |
|  |  |
|  | 'フォーム内で使用するメンバー |
|  | Private myGraph As Graph 'グラフ描画用オブジェクト |
|  | Private PartR, PartC As Decimal '抵抗値, 静電容量 |
|  | Private CutoffPoint As PointF 'カットオフ周波数の座標 |
|  | Private ReadOnly GvRangeMin As Single = -20 'Y軸最小値 |
|  | Private ReadOnly GvRangeMax As Single = 2 'Y軸最大値 |
|  | Private ReadOnly FreqRangeMin = 10 ^ -3 'X軸最小値 |
|  | Private ReadOnly FreqRangeMax = 10 ^ 9 'X軸最大値 |
|  | Private ReadOnly Vin As Decimal = 1 '入力電圧 |
|  | Private Series As New List(Of PointF) 'グラフの系列 |
|  | Private hpf As HighPassFilter '測定対象のハイパスフィルタ |
|  |  |
|  | ''' <summary>ウィンドウを閉じたときに実行されるイベントです。</summary> |
|  | Private Sub GraphForm\_FormClosing(ByVal sender As Object, ByVal e As  System.Windows.Forms.FormClosingEventArgs) Handles Me.FormClosing |
|  |  |
|  | Dim result As Windows.Forms.DialogResult 'MessageBoxの選択結果 |
|  |  |
|  | result = MessageBox.Show("プログラムを終了しますか?", "確認",  MessageBoxButtons.YesNo, MessageBoxIcon.Question) |
|  | If result = Windows.Forms.DialogResult.No Then |
|  | e.Cancel = True 'このイベントをキャンセル |
|  | End If |
|  |  |
|  | End Sub |
|  |  |
|  | ''' <summary>起動時に実行されるイベントです。</summary> |
|  | Private Sub GraphForm\_Load(ByVal sender As System.Object, ByVal e As  System.EventArgs) Handles MyBase.Load |
|  |  |
|  | 'ピクチャーボックスの外形線を立体表示 |
|  | GraphDisplay.BorderStyle = BorderStyle.Fixed3D |
|  |  |
|  | '抵抗の単位のリスト追加 |
|  | ListPrefixR.Items.Add("Ω") |
|  | ListPrefixR.Items.Add("kΩ") |
|  | ListPrefixR.Items.Add("MΩ") |
|  | ListPrefixR.SelectedIndex = 1 |
|  | '抵抗値の入力範囲の設定 |
|  | InputNumberR.Value = 1 |
|  | InputNumberR.Maximum = 10 ^ 5 |
|  | InputNumberR.Minimum = 10 ^ -3 |
|  | InputNumberR.DecimalPlaces = 3 |
|  | InputNumberR.Increment = 0.01 |
|  |  |
|  | '静電容量の単位のリスト追加 |
|  | ListPrefixC.Items.Add("pF") |
|  | ListPrefixC.Items.Add("nF") |
|  | ListPrefixC.Items.Add("uF") |
|  | ListPrefixC.SelectedIndex = 2 |
|  | '静電容量の入力範囲の設定 |
|  | InputNumberC.Value = 1 |
|  | InputNumberC.Maximum = 10 ^ 5 |
|  | InputNumberC.Minimum = 10 ^ -3 |
|  | InputNumberC.DecimalPlaces = 3 |
|  | InputNumberC.Increment = 0.01 |
|  |  |
|  | 'イベントの関連付け(上記の設定変更でイベントが動作しないようにするため) |
|  | AddHandler InputNumberR.ValueChanged, AddressOf InputParameter |
|  | AddHandler InputNumberC.ValueChanged, AddressOf InputParameter |
|  | AddHandler ListPrefixR.SelectedValueChanged, AddressOf InputParameter |
|  | AddHandler ListPrefixC.SelectedValueChanged, AddressOf InputParameter |
|  |  |
|  | 'パラメータ入力のイベントを呼び出す |
|  | InputParameter(sender, e) |
|  |  |
|  | End Sub |
|  |  |
|  | ''' <summary>周波数特性の実測値の描画とカットオフ周波数を開始するイベントです。  </summary> |
|  | Private Sub drawGraph\_Click(ByVal sender As System.Object, ByVal e As  System.EventArgs) Handles DrawGraph.Click, MenuFileOpen.Click |
|  |  |
|  | Dim i As Integer |
|  | Dim freqMin, freqMax As Single |
|  | Dim freqList As New List(Of Single) 'ファイルから読み込んだ周波数 |
|  | Dim graphFreqMin, graphFreqMax As Single'グラフに適用する周波数の最小と最大値 |
|  |  |
|  | '①ファイル読み込み最小、最大周波数と周波数のListの取得 |
|  | If OpenCSV(freqMin, freqMax, freqList) = False Then Exit Sub |
|  |  |
|  | '②測定対象のハイパスフィルタのインスタンス生成 |
|  | hpf = New HighPassFilter(PartR, PartC) '抵抗とコンデンサのパラメータ |
|  | CutoffPoint.X = hpf.GetCutoffFrequency() 'カットオフ周波数の取得 |
|  | CutoffPoint.Y = hpf.GetGain(Vin, CutoffPoint.X) '利得の計算 |
|  | CutoffFreqLabel.Text = AddPrefixUnit(CutoffPoint.X, "Hz") |
|  |  |
|  | '③周波数の範囲を表示 |
|  | FreqRangeLabel.Text = AddPrefixUnit(freqMin, "Hz") & "～" &  AddPrefixUnit(freqMax, "Hz") |
|  | '周波数の範囲を10の乗数になるように変換してからグラフを初期化 |
|  | graphFreqMin = Convert.ToSingle(10 ^ Floor(Log10(freqMin))) |
|  | graphFreqMax = Convert.ToSingle(10 ^ Ceiling(Log10(freqMax))) |
|  | DrawSheet(graphFreqMin, graphFreqMax) |
|  |  |
|  | '④描画データの生成 |
|  | Series.Clear() 'グラフの座標のリストをクリア |
|  | '電圧利得を算出して系列データに代入 |
|  | For Each freq As Single In freqList |
|  | Series.Add(New PointF(freq, hpf.GetGain(Vin, freq))) |
|  | Next |
|  |  |
|  | '⑤系列データ間を線分で描画 |
|  | myGraph.LineSet(DashStyle.Solid, 2, Color.Red) |
|  | For i = 0 To Series.Count - 2 |
|  | If GvRangeMin <= Series(i).Y And Series(i).Y <= GvRangeMax Then |
|  | myGraph.DrawLine(Series(i), Series(i + 1)) |
|  | End If |
|  | Next |
|  |  |
|  | '⑥カットオフ周波数の描画 |
|  | myGraph.MarkerSet(Graph.MarkerShapeStyle.Circle, 5, Color.Blue) |
|  | myGraph.DrawMarker(CutoffPoint) |
|  | myGraph.LineSet(DashStyle.Solid, 1, Color.Blue) |
|  | myGraph.DrawLine(CutoffPoint.X, GvRangeMin, CutoffPoint.X, CutoffPoint.Y) |
|  | myGraph.DrawLine(graphFreqMin, CutoffPoint.Y, CutoffPoint.X,  CutoffPoint.Y) |
|  | myGraph.FontColor = Color.Blue |
|  | myGraph.DrawText(CutoffPoint.X \* 0.9, GvRangeMin - 0.5, "fc") |
|  | myGraph.DrawText(graphFreqMin \* 0.7, -3 + 0.5, AddPrefixUnit(-3).PadLeft(4)) |
|  |  |
|  | End Sub |
|  |  |
|  | ''' <summary>グラフの目盛等を描画するメソッドです。</summary> |
|  | '''<param name="frequencyMin" >描画する周波数の最小値を指定します。</param> |
|  | ''' <param name="frequencyMax">描画する周波数の最大値を指定します。</param> |
|  | Private Sub DrawSheet(ByVal frequencyMin As Single, ByVal frequencyMax As Single) |
|  |  |
|  | Dim digitMin, digitMax As Integer |
|  |  |
|  | digitMin = Convert.ToInt32(Math.Log10(frequencyMin)) |
|  | digitMax = Convert.ToInt32(Math.Log10(frequencyMax)) |
|  | '範囲を左下(最小周波数, 最小利得-20dB)と右上(最大周波数, 最大利得2dB),余白の上下を30pixelと左右20pixel,X軸線形軸,Y軸対数軸でグラフを描画 |
|  | myGraph = New Graph(GraphDisplay, frequencyMin, GvRangeMin - 4,  frequencyMax, GvRangeMax + 3, 30, 20, Graph.AxisStyle.XlogYlinear) |
|  |  |
|  | '背景色は白 |
|  | myGraph.BackColor = Color.White |
|  |  |
|  | 'フォントの変更 |
|  | myGraph.FontSet("ＭＳ ゴシック", 10, FontStyle.Regular, Color.Black) |
|  |  |
|  | '線を実線、幅2、黒に変更 |
|  | myGraph.LineSet(DashStyle.Solid, 2, Color.Black) |
|  | myGraph.DrawRectangle(frequencyMin, GvRangeMin, frequencyMax, GvRangeMax) |
|  |  |
|  | 'X軸の目盛線を追加 |
|  | For i = digitMin To digitMax Step 1 |
|  | For j = 1 To 9 Step 1 |
|  | If j = 1 Then |
|  | myGraph.LineSet(DashStyle.Solid, 1, Color.Green) |
|  | myGraph.DrawText(j \* 10 ^ i \* 0.9, GvRangeMin - 2,  AddPrefixUnit(j \* 10 ^ i)) |
|  | Else |
|  | myGraph.LineSet(DashStyle.Dash, 1, Color.Green) |
|  | End If |
|  | myGraph.DrawLine(j \* 10 ^ i, GvRangeMin, j \* 10 ^ i, GvRangeMax) |
|  | Next |
|  | Next |
|  | 'X軸のタイトル表示 |
|  | myGraph.FontStyle = FontStyle.Bold |
|  | myGraph.DrawText(10 ^ (digitMax - digitMin - 1), GvRangeMin - 4, "周波数f[Hz]") |
|  | myGraph.FontStyle = FontStyle.Regular |
|  |  |
|  | 'Y軸の目盛線を追加 |
|  | For i = GvRangeMin To GvRangeMax Step 5 |
|  | If i = 0 Then |
|  | myGraph.LineSet(DashStyle.Solid, 1, Color.Green) |
|  | Else |
|  | myGraph.LineSet(DashStyle.Dash, 1, Color.Green) |
|  | End If |
|  | myGraph.DrawLine(frequencyMin, i, frequencyMax, i) |
|  | myGraph.DrawText(frequencyMin \* 0.7, i + 0.5,  AddPrefixUnit(i).PadLeft(4)) |
|  | Next |
|  | 'Y軸のタイトル表示 |
|  | myGraph.FontStyle = FontStyle.Bold |
|  | myGraph.DrawText(frequencyMin \* 0.8, GvRangeMax + 2, "電圧利得Gv[dB]") |
|  | myGraph.FontStyle = FontStyle.Regular |
|  |  |
|  | Series.Clear() ' 系列データのクリア |
|  |  |
|  | End Sub |
|  |  |
|  | ''' <summary>抵抗RとコンデンサCの入力された数値と単位からパラメータを演算する  メソッドです。</summary> |
|  | Private Sub InputParameter(ByVal sender As System.Object,  ByVal e As System.EventArgs) |
|  |  |
|  | '入力されたRとCが0以下の時は以下を実行しない |
|  | If (InputNumberR.Value <= 0) OrElse (InputNumberC.Value <= 0) Then Exit Sub |
|  |  |
|  | Select Case ListPrefixR.Text |
|  | Case "MΩ" |
|  | PartR = InputNumberR.Value \* 10 ^ 6 |
|  | Case "kΩ" |
|  | PartR = InputNumberR.Value \* 10 ^ 3 |
|  | Case Else |
|  | PartR = InputNumberR.Value |
|  | End Select |
|  |  |
|  | Select Case ListPrefixC.Text |
|  | Case "pF" |
|  | PartC = InputNumberC.Value \* 10 ^ -12 |
|  | Case "nF" |
|  | PartC = InputNumberC.Value \* 10 ^ -9 |
|  | Case "uF" |
|  | PartC = InputNumberC.Value \* 10 ^ -6 |
|  | Case Else |
|  | PartC = InputNumberC.Value |
|  | End Select |
|  |  |
|  | End Sub |
|  |  |
|  | ''' <summary>数値にSI接頭辞と単位を付加して文字列を返すメソッドです。</summary> |
|  | ''' <param name="number">数値を指定します。</param> |
|  | ''' <param name="unit">単位を指定します。省略した場合は単位は付加されません。</param> |
|  | ''' <returns>小数点以下第3位までの数値にSI接頭辞を文字列を返します。</returns> |
|  | Private Function AddPrefixUnit(ByVal number As Single,  Optional ByVal unit As String = "") As String |
|  |  |
|  | Dim addString As String = String.Empty |
|  |  |
|  | Select Case number |
|  | Case Is > 10 ^ 12 |
|  | addString = (number / (10 ^ 12)).ToString("0.###") & "T" |
|  | Case Is >= 10 ^ 9 |
|  | addString = (number / (10 ^ 9)).ToString("0.###") & "G" |
|  | Case Is >= 10 ^ 6 |
|  | addString = (number / (10 ^ 6)).ToString("0.###") & "M" |
|  | Case Is >= 10 ^ 3 |
|  | addString = (number / (10 ^ 3)).ToString("0.###") & "k" |
|  | Case Is >= 10 ^ 0 |
|  | addString = number.ToString("0.###") |
|  | Case Is >= 10 ^ -3 |
|  | addString = (number \* (10 ^ 3)).ToString("0.###") & "m" |
|  | Case Is >= 10 ^ -6 |
|  | addString = (number \* (10 ^ 6)).ToString("0.###") & "u" |
|  | Case Is >= 10 ^ -9 |
|  | addString = (number \* (10 ^ 9)).ToString("0.###") & "n" |
|  | Case Is >= 10 ^ -12 |
|  | addString = (number \* (10 ^ 12)).ToString("0.###") & "p" |
|  | Case Else |
|  | addString = number.ToString("0.###") |
|  | End Select |
|  |  |
|  | addString = addString & unit |
|  |  |
|  | Return addString |
|  |  |
|  | End Function |
|  |  |
|  | ''' <summary>ダイアログを表示して周波数のファイルを読み込みます。</summary> |
|  | ''' <param name="fMin">最小周波数を参照渡しで取得します。</param> |
|  | ''' <param name="fMax">最大周波数を参照渡しで取得します。</param> |
|  | ''' <param name="fList">周波数をListクラスで取得します。</param> |
|  | ''' <returns>ファイルを正常に読み込めた場合はTrueを、  読み込めない場合はFalseを返します。</returns> |
|  | Private Function OpenCSV(ByRef fMin As Single, ByRef fMax As Single,  ByRef fList As List(Of Single)) As Boolean |
|  |  |
|  | Dim openDialog As New OpenFileDialog 'ファイルから読み込むためのダイアログ |
|  |  |
|  | fList = New List(Of Single) 'ファイルから読み込んだ周波数 |
|  |  |
|  | 'ダイアログのタイトル |
|  | openDialog.Title = "入力信号用のファイルを読み込んでください。" |
|  | 'デフォルトのファイル名指定なし |
|  | openDialog.FileName = "" |
|  | 'デスクトップをデフォルトのフォルダに指定 |
|  | openDialog.InitialDirectory =  My.Computer.FileSystem.SpecialDirectories.Desktop |
|  | 'ファイルの種類はCSVとText,すべてのファイル |
|  | openDialog.Filter =  "CSVファイル|\*.csv|テキストファイル|\*.txt|すべてのファイル|\*.\*" |
|  | 'ダイアログの表示がOK以外の時は以下の処理を実行しない |
|  | If openDialog.ShowDialog() <> Windows.Forms.DialogResult.OK Then  Return False |
|  |  |
|  | 'ファイルを読み込む処理 |
|  | Using sr As New StreamReader(openDialog.FileName) |
|  | Do Until sr.EndOfStream 'ファイルを末尾まで読み込む |
|  | Dim f As Single 'ファイルから読み込む数値 |
|  | If Single.TryParse(sr.ReadLine(), f) Then  '変換可能な文字列なら変換して格納 |
|  | fList.Add(f) '周波数の値を追加 |
|  | End If |
|  | Loop |
|  | End Using |
|  |  |
|  | If fList.Count < 2 Then |
|  | MessageBox.Show("読み込んだファイルのデータに不足、または異常があります。",  "警告", MessageBoxButtons.OK, MessageBoxIcon.Exclamation) |
|  | Return False |
|  | End If |
|  |  |
|  | fList.Sort() '周波数のリストを昇順でソート |
|  |  |
|  | '周波数の最小値と最大値を格納 |
|  | fMin = fList(0) |
|  | fMax = fList(fList.Count - 1) |
|  |  |
|  | '周波数の測定範囲をチェック |
|  | If fMin < FreqRangeMin OrElse FreqRangeMax < fMax Then |
|  | MessageBox.Show("周波数が描画可能が範囲を超えています。", "警告",  MessageBoxButtons.OK, MessageBoxIcon.Exclamation) |
|  | Return False |
|  | Else |
|  | Return True |
|  | End If |
|  |  |
|  | End Function |
|  |  |
|  | ''' <summary>パラメータと系列データをCSVファイルに保存するイベントです。</summary> |
|  | Private Sub SaveCSV\_Click(ByVal sender As System.Object,  ByVal e As System.EventArgs) Handles SaveCSV.Click, MenuFileSaveCSV.Click |
|  |  |
|  | Dim saveDialog As New SaveFileDialog |
|  | 'ダイアログのタイトル |
|  | saveDialog.Title = "周波数特性の保存先を指定してください。" |
|  | 'デフォルトのファイル名は"年月日\_時分秒.csv" |
|  | saveDialog.FileName = Now.ToString("yyyyMMdd\_HHmmss") & ".csv" |
|  | 'デスクトップをデフォルトの保存先に指定 |
|  | saveDialog.InitialDirectory =  My.Computer.FileSystem.SpecialDirectories.Desktop |
|  | 'ファイルの種類はCSVとText,すべてのファイル |
|  | saveDialog.Filter =  "CSVファイル|\*.csv|テキストファイル|\*.txt|すべてのファイル|\*.\*" |
|  | 'ダイアログの表示がOK以外の時は以下の処理を実行しない |
|  | If saveDialog.ShowDialog() <> Windows.Forms.DialogResult.OK Then Exit Sub |
|  |  |
|  | Using sw As New StreamWriter(saveDialog.FileName) |
|  | 'ファイルの先頭にパラメータを記述 |
|  | sw.WriteLine("R=" & PartR.ToString()) |
|  | sw.WriteLine("C=" & PartC.ToString()) |
|  | '特性をファイルへ書き込む |
|  | For Each p As PointF In Series |
|  | sw.Write(p.X.ToString()) |
|  | sw.Write(",") |
|  | sw.WriteLine(p.Y.ToString()) |
|  | Next |
|  | End Using |
|  |  |
|  | End Sub |
|  |  |
|  | ''' <summary>グラフを画像ファイルとして保存するメソッドです。</summary> |
|  | Private Sub SaveGraph\_Click(ByVal sender As System.Object,  ByVal e As System.EventArgs) Handles SaveGraph.Click, MenuFileSaveGraph.Click |
|  |  |
|  | Dim saveDialog As New SaveFileDialog |
|  | 'ダイアログのタイトル |
|  | saveDialog.Title = "グラフの保存先を指定してください。" |
|  | 'デフォルトのファイル名は"年月日\_時分秒.bmp" |
|  | saveDialog.FileName = Now.ToString("yyyyMMdd\_HHmmss") & ".bmp" |
|  | 'デスクトップをデフォルトの保存先に指定 |
|  | saveDialog.InitialDirectory =  My.Computer.FileSystem.SpecialDirectories.Desktop |
|  | 'ファイルの種類はBMP, JPEG, PNG |
|  | saveDialog.Filter = "BMPファイル(\*.bmp)|\*.bmp|JPEGファイル  (\*.jpg;\*jpeg)|\*.jpg;\*jpeg|PNGファイル(\*.png)|\*.png" |
|  | 'ダイアログの表示がOK以外の時は以下の処理を実行しない |
|  | If saveDialog.ShowDialog() <> Windows.Forms.DialogResult.OK Then Exit Sub |
|  |  |
|  | If myGraph.Export(saveDialog.FileName) = False Then |
|  | MessageBox.Show("保存するファイル形式、保存先が正しくありません。") |
|  | End If |
|  |  |
|  | End Sub |
|  |  |
|  | ''' <summary>メニューバーの「終了」を選択時に実行されるイベントです。</summary> |
|  | Private Sub MenuExit\_Click(ByVal sender As System.Object,  ByVal e As System.EventArgs) Handles MenuExit.Click |
|  |  |
|  | Application.Exit() '終了 |
|  |  |
|  | End Sub |
|  |  |
|  | ''' <summary>メニューバーの「回路図」をクリックしたときに実行される  イベントです。</summary> |
|  | Private Sub MenuCircuitView\_Click(ByVal sender As System.Object,  ByVal e As System.EventArgs) Handles MenuCircuitView.Click |
|  |  |
|  | 'ダイアログに回路図が表示 |
|  | AboutBox.ShowDialog() |
|  |  |
|  | End Sub |
|  |  |
|  | End Class |

解答例のソースコード(ハイパスフィルタのクラス)

|  |  |
| --- | --- |
| 行 | プログラム |
|  | 'オプション指定 |
|  | Option Explicit On |
|  | Option Strict On |
|  | Option Infer Off |
|  | Option Compare Binary |
|  |  |
|  | Imports System.Math |
|  |  |
|  | ''' <summary>抵抗とコンデンサの一次フィルタ用のクラスです。</summary> |
|  | ''' <remarks>各種パラメータ及びプロパティはDecimal型を適用しています。</remarks> |
|  | Public Class HighPassFilter |
|  | 'プロパティ及びメソッド用メンバー |
|  | Private rValue As Decimal '抵抗値 |
|  | Private cValue As Decimal '静電容量 |
|  |  |
|  | ''' <summary>ハイパスフィルタのインスタンスを生成します。</summary> |
|  | ''' <param name="r">抵抗値を指定します。</param> |
|  | ''' <param name="c">コンデンサの静電容量を指定します。</param> |
|  | ''' <remarks>インスタンス生成時に抵抗とコンデンサの値でフィルタの特性が設定されます。</remarks> |
|  | Public Sub New(ByVal r As Decimal, ByVal c As Decimal) |
|  |  |
|  | '抵抗とコンデンサの値とする。 |
|  | rValue = r |
|  | cValue = c |
|  |  |
|  | End Sub |
|  |  |
|  | ''' <summary>フィルタの入力信号（電圧と周波数を指定）から  電圧利得を取得するメソッドです。</summary> |
|  | ''' <param name="inputVoltage">入力電圧を指定します。</param> |
|  | ''' <param name="inputFrequency">入力周波数(Hz)を指定します。</param> |
|  | ''' <returns>電圧利得を返します。</returns> |
|  | ''' <remarks>抵抗とコンデンサの値はインスタンス生成時の値が適用されます。</remarks> |
|  | Public Function GetGain(ByVal inputVoltage As Decimal,  ByVal inputFrequency As Decimal) As Decimal |
|  |  |
|  | Dim vout, gv, w, wcr As Decimal '出力電圧と角周波数ωとωCR算出用の変数 |
|  |  |
|  | w = 2 \* Convert.ToDecimal(PI) \* inputFrequency '角周波数ω算出 |
|  | wcr = w \* cValue \* rValue 'ωCR算出 |
|  | vout = Convert.ToDecimal(Sqrt(Pow(wcr, 2) / (1 + Pow(wcr, 2))) \* inputVoltage) |
|  | gv = 20 \* Convert.ToDecimal(Log10(vout / inputVoltage)) '電圧利得の算出 |
|  | Return gv '電圧利得を返す |
|  |  |
|  | End Function |
|  |  |
|  | ''' <summary>フィルタのカットオフ周波数を取得するメソッドです。</summary> |
|  | ''' <returns>カットオフ周波数を返します。</returns> |
|  | ''' <remarks>抵抗とコンデンサの値はインスタンス生成時の値が適用されます。</remarks> |
|  | Public Function GetCutoffFrequency() As Decimal |
|  |  |
|  | Dim fc As Decimal |
|  |  |
|  | fc = Convert.ToDecimal(1 / (2 \* PI \* rValue \* cValue)) |
|  |  |
|  | Return fc |
|  |  |
|  | End Function |
|  |  |
|  | ''' <summary>このフィルタの抵抗値を取得します。</summary> |
|  | ''' <returns>抵抗値を単位[Ω]で返します。</returns> |
|  | ''' <remarks>このプロパティは確認用です。</remarks> |
|  | Public ReadOnly Property Rohm() As Decimal |
|  | Get |
|  | Return rValue |
|  | End Get |
|  | End Property |
|  |  |
|  | ''' <summary>このフィルタの静電容量を取得します。</summary> |
|  | ''' <returns>静電容量を単位[F]で返します。</returns> |
|  | ''' <remarks>このプロパティは確認用です。</remarks> |
|  | Public ReadOnly Property Cfarad() As Decimal |
|  | Get |
|  | Return cValue |
|  | End Get |
|  | End Property |
|  |  |
|  | End Class |