実技課題E-45B

実技課題

「パソコンを用いた計測制御システムの製作（Visual C#）」

|  |
| --- |
| １　作業時間  　　 240分（4時間：休憩を除く）  ２　配付資料  問題用紙，解答用紙  ３　課題作成、提出方法  ・各個人で作業すること  ・ソースファイルによる提出 |

１．課題名：パソコンを用いた計測制御システム製作

２．課題内容（イメージ）

　以下の仕様に基づいた計測制御システムのプログラムを作成しなさい。

　システムの使用はパソコンから入力対象となる機器へ制御データを入力し、制御対象から得られたデータを出力対象となる機器からパソコンに出力し、その制御内容をGUIで管理する。ただし実習環境によってはパソコンのみで行なって良いものとする。

**パソコン（プログラム）**

・通信（RS-232C, GPIB, USB, Ethernet等）

・拡張ボード制御（DIO、A/D、D/A等）

・GUI画面（操作、グラフィック表示等）

・ファイル処理（制御データ、ログデータ等）

**出力対象**

・計測器（オシロ、DMM等）

・モータ（回転数制御等）

**入力対象**

・入力信号（FG、電源等）

・モータ（回転量）



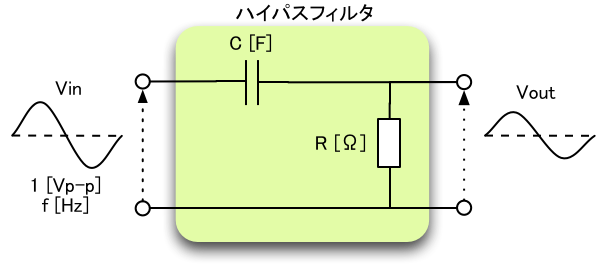
**制御・測定対象**

・アナログ回路（増幅・変調）

・制御回路(ON/OFF, PWM)

**制御**

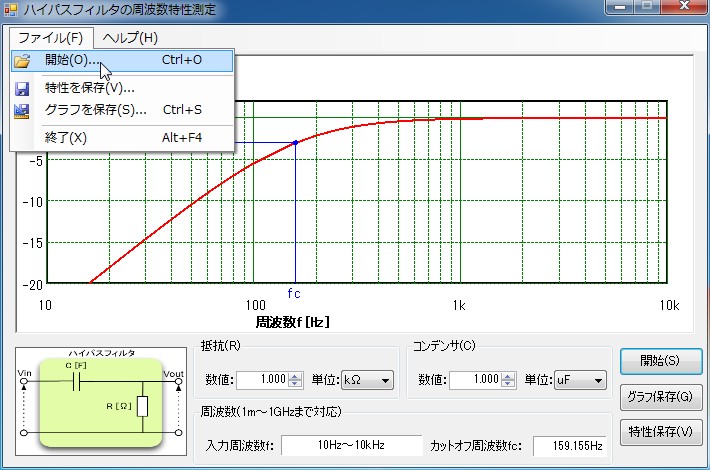
**取得**

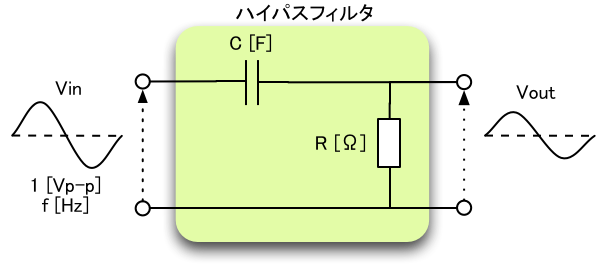


1. 各機器の電源を投入し、プログラムを起動する。
2. プログラム起動後、「開始」のボタンをクリックすると入力対象からデータを取得する。
3. 対象がオシロの場合、描画中の波形のデータをRS-232CやGPIBといった計測向けのインタフェース(USBやEthernetも可)で取得する。
4. 計測対象の状態をON/OFF信号（回転数等）、あるいはアナログ信号取得する。
5. 機材の都合上、入力対象機器が用意できない場合は、入力機器の信号を想定したログ等をCSVファイル形式で用意する。
6. データを取得後、目的の制御に応じた結果を出力対象に反映させる。
7. 対象がFGの場合、アプリケーションで指定した波形をFGに出力させる信号を送信する。
8. 入力対象の信号から演算した結果を出力対象へ出力する。A/D、D/A変換をしている場合はディジタル値とアナログ信号の換算を行なうこと。
9. 機材の都合上、入力対象機器が用意できない場合は、入力された信号（もしくはファイル）を時系列もしくは特性等（例：周波数等）でまとめたものをグラフで描画し、グラフの保存機能を追加する。

**仕様例（フィルタ回路のカットオフ周波数fcの評価）**

　下記の回路図は抵抗RとコンデンサCによって構成されるハイパスフィルタの回路です。この回路の抵抗値RとコンデンサCを設定し、フィルタの性能を表す周波数特性の表示と及びカットオフ周波数の算出を行なうプログラムを作成します。

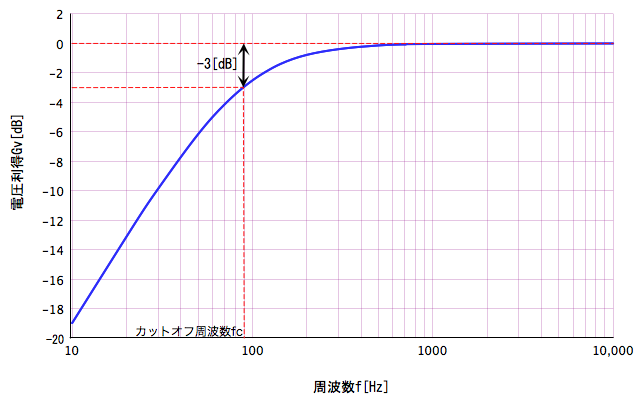




ファイルで周波数fを読み込む。

ファイルに電圧利得Gvを書き込む。

電圧利得：



制御対象回路（ハイパスフィルタ）の電圧利得の周波数（f-Gv）特性

※（X軸は周波数を対数軸、Y軸は電圧利得を線形軸で描画）

この回路に1[Vp-p](固定)の周波数f[Hz](可変)の電圧Vinを入力し、出力電圧Voutから周波数特性とカットオフ周波数fc（電圧利得Gv[dB]が-3[dB]低下した周波数）を算出し描画するプログラムを作成しなさい。作成条件は以下のとおりとします。

10

20

30

50

70

100

200

300

500

700

1000

2000

…

1. パソコン上で抵抗RとコンデンサCの値を指定し、「開始」のボタンをクリックすると周波数特性を測定する。測定する周波数の範囲は右図のようなテキストファイルを選択し読み込むものとする。

なおカットオフ周波数fcの理論値は指定された抵抗RとコンデンサCを用いて以下の式から算出できる。



1. フィルタの電圧利得Gv[dB]は以下の式を用いて算出する。

1. 周波数fを変化させ、周波数特性を表すグラフを描画する。グラフのX軸は周波数[Hz]（対数軸）、Y軸は電圧利得[dB]とする。グラフの描画は、提供するGraphクラスを使用しても良いものとする。
2. 周波数特性からカットオフ周波の測定値fcを求め表示する。
3. 「保存」のボタンをクリックすると、周波数特性のグラフを「名前を付けて保存」のダイアログを表示して画像ファイル(BMP、JPEG、PNG等のパソコンで表示可能なファイル形式)を任意の場所に保存できるようにすること。
4. 独自の機能を追加してもよい。以下は例である。
5. アプリケーションの操作方法や動作についてテキストファイル等で第3者が理解できるようなドキュメントを作成すること（簡易的なもので可）。独自の機能を追加し合は追加した機能をドキュメントに記載すること。

３．作業時間：240分（4時間）

・機材の準備時間等は含まないものとする。

・これまでの訓練で使用したソースファイル等を流用して良いものとする。

４．課題仕様

（１）課題に必要な機器を用意する。実機が用意できない場合はプログラムのみの課題でも良いものとする。

|  |  |  |
| --- | --- | --- |
| 機器 | 実機による例 | プログラムのみによる例  （本資料に記載） |
| パソコン(PC) | Windows XP/Vista/7/8 | Windows 7 |
| 開発環境 | Visual Studio 2005/2008/2010/2012 | Visual Studio 2005以降 |
| 入力・出力I/F | RS-232C/USB/Ethernet/GP-IB | なし |
| 入力機器 | ファンクションジェネレータ | CSVファイルの読み込み |
| 出力機器 | オシロスコープ | CSVファイルの書き込み |
| 制御・負荷回路 | フィルタ回路、モータ制御回路 | フィルタ回路の計算 |

　（２）動作仕様

1. 各種機器の接続（電源、入出力信号、通信ケーブル等）を行なう。
2. 機器の電源を投入し、動作可能な状態とする。パソコン側は作成したアプリケーションを起動する。
3. アプリケーションにパラメータ（測定時間、測定用数値）を入力し「開始」のボタンをクリックすると計測を開始する。
4. 計測を開始し制御機器の特性と、その結果を表示する。

　グラフ描画には別途配布するDLLを開発環境に組み込んで作成すること。また使用例については後述のソースファイルを参照すること。

　（３）作業内容

1. Visual Studioを起動しプロジェクトをWindowsフォームアプリケーションで作成する。言語の仕様やバージョン等については講師の指示に従うこと。
2. 提出物

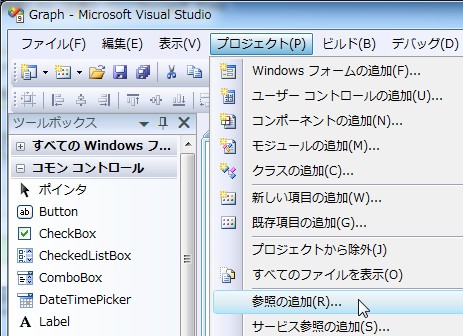
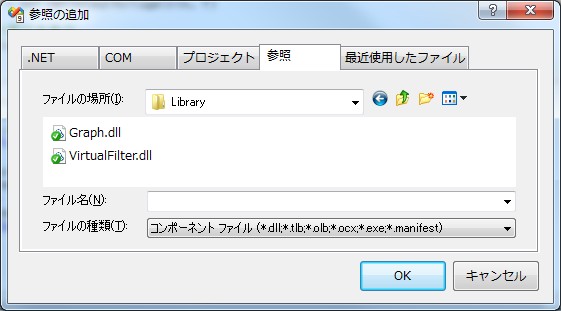
|  |  |  |
| --- | --- | --- |
|  | 機能 | 備考 |
| １ | 作成したプログラムのファイル一式 |  |
| ２ | プログラムのドキュメント（操作手順、動作、追加機能等を記載したもの） | テキストファイル  (形式は講師で指示) |

５．参考資料

　参考として、グラフを描画するGraphクラスのサンプルを以下に示す。

（１）Graphクラスの使用方法(GraphSample.zipに格納)

1. プロジェクト作成後、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。

'独自の名前空間

Using Polytech.Windows.Form; 'Graphクラス

'既定の名前空間

Using System.Text;

1. 例として以下の式をxが対数軸で描画するプログラムのサンプルを示す。xの範囲は1～10000とする。

|  |  |  |
| --- | --- | --- |
| 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 |
| DrawLine1 | 式1のグラフを描画 |
| DrawLine2 | 式2のグラフを描画 |
| SaveGraph | グラフを画像ファイルとして保存 |
| PictureBox | GraphDisplay | グラフの描画対象 |

　……（式1）

　……（式2）

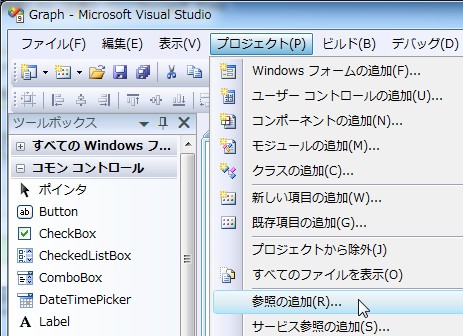
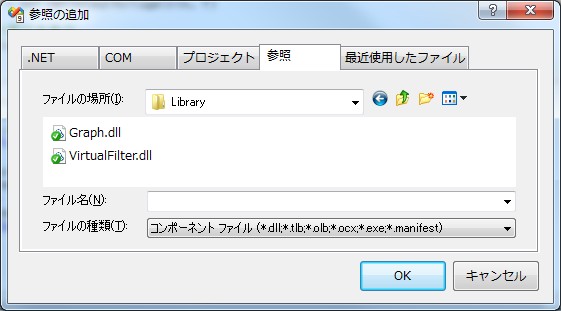


Graphクラスを用いたグラフの描画サンプル

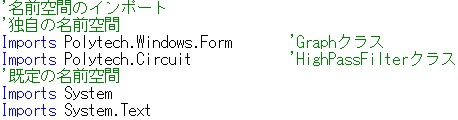
|  |  |
| --- | --- |
| 行 | プログラム |
|  | using System; |
|  | using System.Collections.Generic; |
|  | using System.ComponentModel; |
|  | using System.Data; |
|  | using System.Drawing; |
|  | using System.Linq; |
|  | using System.Text; |
|  | using System.Windows.Forms; |
|  | using System.Drawing.Drawing2D; |
|  | using Polytech.Windows.Form; |
|  |  |
|  | namespace GraphSample |
|  | { |
|  | public partial class Form1 : Form |
|  | { |
|  |  |
|  | private Graph myGraph; |
|  |  |
|  | public Form1() |
|  | { |
|  | InitializeComponent(); |
|  | } |
|  |  |
|  | private void Form1\_Load(object sender, EventArgs e) |
|  | { |
|  | float x, y;//数式用 |
|  | int i; //ループ変数 |
|  |  |
|  | //PictureBox1に描画座標を左下(1,0)、右上(10000, 10), 左右の余白20pixel， |
|  | //上下の余白30pixel, X軸を対数軸, Y軸を線形軸でグラフを描画 |
|  | myGraph = new Graph(GraphDisplay, 1, 0, 10000, 10, 30, 20,  Graph.AxisStyle.XlogYlinear); |
|  |  |
|  | myGraph.BackColor = Color.White; //背景色を白 |
|  |  |
|  | //フォントの設定(フォント:ＭＳ ゴシック",サイズ:8pt,スタイル:太字,色:黒) |
|  | myGraph.FontName = "ＭＳ ゴシック"; |
|  | myGraph.FontSize = 8; |
|  | myGraph.FontStyle = FontStyle.Bold; |
|  | myGraph.FontColor = Color.Black; |
|  |  |
|  | //グラフのタイトルを(10,11)へ描画 |
|  | myGraph.DrawText(10, 11, "Graphクラスサンプル"); |
|  |  |
|  | //フォント一括設定(フォント:ＭＳ ゴシック",サイズ:7pt,スタイル:標準,色:青) |
|  | myGraph.FontSet("MS ゴシック", 7, FontStyle.Regular, Color.Blue); |
|  |  |
|  | //線のスタイルを実線、太さを2、色を黒 |
|  | myGraph.LineStyle = DashStyle.Solid; |
|  | myGraph.LineWidth = 2; |
|  | myGraph.LineColor = Color.Black; |
|  |  |
|  | //X軸とY軸を描画 |
|  | myGraph.DrawLine(1, 0, 10000, 0); //(1,0)から(10000,0)へ直線描画 |
|  | myGraph.DrawLine(1, 0, 1, 10); //(1,0)から(1,10)へ直線描画 |
|  |  |
|  | //線のスタイルを破線、太さを1、色を緑 |
|  | myGraph.LineStyle = DashStyle.Solid; |
|  | myGraph.LineWidth = 1; |
|  | myGraph.LineColor = Color.Green; |
|  |  |
|  | //x軸の目盛線と目盛の数値の描画 |
|  | for (i = 0; i < 5; i++) { |
|  | x = (float)Math.Pow(10 , i); |
|  | myGraph.DrawLine(x, 0, x, 10); |
|  | if ((3 <= i) && (i < 6)) { //10の3乗以上6未満で補助単位k(キロ) |
|  | myGraph.DrawText(x, 0, (x / 1000).ToString("###k")); |
|  | } |
|  | else { //上記以外のとき補助単位なし |
|  | myGraph.DrawText(x, 0, x.ToString()); |
|  | } |
|  | } |
|  |  |
|  | //y軸の目盛線と目盛の数値の描画 |
|  | for (i = 0;i <= 10; i += 2) { |
|  | y = i; |
|  | myGraph.DrawLine(1, y, 10000, y); |
|  | myGraph.DrawText(0.6f, y + 0.5f, y.ToString().PadLeft(4)); |
|  | } |
|  | //描画したイメージを記憶する |
|  | myGraph.SaveInitialGraph(); |
|  | } |
|  |  |
|  | private void GraphInit\_Click(object sender, EventArgs e) |
|  | { |
|  | myGraph.LoadInitialGraph(); //初期 |
|  | } |
|  |  |
|  | private void DrawLine1\_Click(object sender, EventArgs e) |
|  | { |
|  | float x, y; //数式用 |
|  | int i, j, k; //ループ変数 |
|  | PointF[] series = new PointF[37]; //座標格納用用配列 |
|  |  |
|  | //1～10000まで対数目盛間隔で配列に代入 |
|  | //1,2,3,...,9,10,20,30,...,90,100,200 |
|  | k = 0; |
|  | for (i = 0; i < 5; i++) { //10のi乗 |
|  | for(j = 1; j < 10; j++) { //1～9の値 |
|  |  |
|  | y = x / 1000; //yの値 |
|  | series[k] = new PointF(x, y); //配列に格納 |
|  | k = k + 1; //次の配列の要素番号 |
|  | if( x >= 10000) break; //10000まで格納後このステートメントを抜ける |
|  | } |
|  |  |
|  | //線のスタイルを実線、太さを1、色を青 |
|  | myGraph.LineStyle = DashStyle.Solid; |
|  | myGraph.LineWidth = 1; |
|  | myGraph.LineColor = Color.Blue; |
|  |  |
|  | //マーカーのスタイルを円、幅を5px、色を青 |
|  | myGraph.MarkerStyle = Graph.MarkerShapeStyle.Circle; |
|  | myGraph.MarkerWidth = 5; |
|  | myGraph.MarkerColor = Color.Blue; |
|  |  |
|  | //最初の点のマーカーを描画 |
|  | myGraph.DrawMarker(series[0]); |
|  |  |
|  | //配列内の座標の2点間を直線で描画、終点にはマーカー描画 |
|  | for (i = 0; i < series.Length - 1; i ++) { |
|  | myGraph.DrawMarker(series[i + 1]); |
|  | myGraph.DrawLine(series[i], series[i + 1]); |
|  | } |
|  | } |
|  |  |
|  | private void DrawLine2\_Click(object sender, EventArgs e) |
|  | { |
|  | PointF formula = new PointF(); //数式用 |
|  | int i, j; //ループ変数 |
|  | List<PointF> series = new List<PointF>();  //PointF構造体でListクラスのインスタンス生成 |
|  |  |
|  | //1～10000まで対数目盛間隔  (1,2,3,...,9,10,20,30,...,90,100,200...)でリストへ追加 |
|  | for (i = 0; i < 5; i++) { //10のi乗 |
|  | for (j = 1; j < 10; j++) { //1～9の値 |
|  | formula.X = (float)(j \* Math.Pow(10, i)); //xの値 |
|  | formula.Y = (float)(Math.Log10(formula.X)); //yの値 |
|  | series.Add(formula); //座標を追加 |
|  | if (formula.X >= 10000) break;  //10000まで格納後このステートメントを抜ける |
|  | } |
|  | } |
|  |  |
|  | //線のスタイルを実線、太さを1、色を赤 |
|  | myGraph.LineSet(DashStyle.Solid, 1, Color.Red); |
|  |  |
|  | //マーカーの形状を三角、幅を7px、色を赤 |
|  | myGraph.MarkerSet(Graph.MarkerShapeStyle.Triangle, 7, Color.Red); |
|  |  |
|  | //配列内の座標の2点間を直線で描画、終点にはマーカー描画 |
|  | for (i = 0; i < series.Count; i++) |
|  | { |
|  | if (i == 0) |
|  | { //最初の点のマーカーのみ描画 |
|  | myGraph.DrawMarker(series[0]); |
|  | } |
|  | else |
|  | { //以降は前に描画した座標を始点として直線を描画 |
|  | myGraph.DrawLine(series[i]); |
|  | myGraph.DrawMarker(series[i]); |
|  | } |
|  | } |
|  | } |
|  |  |
|  | private void SaveGraph\_Click(object sender, EventArgs e) |
|  | { |
|  | String filePath; //画像ファイルの保存先のパス |
|  |  |
|  | filePath = "C:\\Temp\\Graph.jpg"; //保存先の代入  (画像形式はBMP, JPEG, GIF. PNG, TIFF等) |
|  |  |
|  | //Exportメソッドで描画しているPictureBoxのグラフィックを保存 |
|  | if (myGraph.Export(filePath)) { |
|  | //保存成功時はTrue |
|  | MessageBox.Show(filePath + "に保存しました。", "完了", MessageBoxButtons.OK, MessageBoxIcon.Information); |
|  | } |
|  | else { |
|  | //失敗時はFalse |
|  | MessageBox.Show(filePath + "に保存できません。", "失敗",  MessageBoxButtons.OK, MessageBoxIcon.Exclamation); |
|  | } |
|  | } |
|  | } |
|  | } |

（１）HighPassFilterクラスの使用方法(FilterSample.zipに格納)

1. プロジェクト作成後、グラフ描画用のDLL(VirtualFilter.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"VirtualFilter.dll"を選択する。

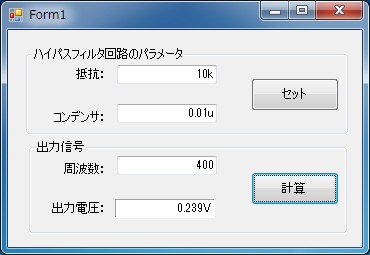
　

1. 名前空間"Polytech.Circuit"でHighPassFilterクラスを定義しているため、Importsステートメントで"Polytech.Circuit"名前空間を追加すること。



1. 例として仮想ハイパスフィルタ（HighPassFilterクラス）を用いて抵抗R、コンデンサC、周波数f、入力電圧vin=1Vのときの出力電圧voutを算出するプログラムを作成する。

|  |  |  |
| --- | --- | --- |
| コントロール | オブジェクト名 | 備考 |
| Form | Form1 | スタートアップフォーム |
| TextBox | InputResistance | 抵抗値を入力(例:10[kΩ]なら"10k") |
| InputCapacitance | 静電容量を入力(例:0.1[uF]なら"0.1u") |
| InputFrequency | 周波数を入力(例:100[Hz]なら"100") |
| Button | CalculateVout | ハイパスフィルタの出力電圧を算出 |
| Label | DisplayVout | 出力電圧Voutを表示 |



HighPassFilterクラスを用いたグラフの描画サンプル

|  |  |
| --- | --- |
| 行 | プログラム |
|  | Imports Polytech.Circuit 'HighPassFilterクラス |
|  |  |
|  | Public Class Form1 |
|  |  |
|  | 'ハイパスフィルタのオブジェクト |
|  | Private hpf As HighPassFilter |
|  |  |
|  | ''' <summary>抵抗RとコンデンサCでフィルタ回路を生成する</summary> |
|  | Private Sub SetParameter\_Click(ByVal sender As System.Object,  ByVal e As System.EventArgs) Handles SetParameter.Click |
|  |  |
|  | Dim r, c As Decimal '抵抗、コンデンサ |
|  | 'テキストボックスに入力された文字列を数値に変換して代入 |
|  | r = SIPrefixToDecimal(InputResistance.Text) '抵抗値[Ω] |
|  | c = SIPrefixToDecimal(InputCapacitance.Text) 'コンデンサ[F] |
|  |  |
|  | 'ハイパスフィルタのインスタンス生成(R, C) |
|  | 'インスタンスを生成すると内部で許容差を含めた値の回路になる |
|  | hpf = New HighPassFilter(r, c) |
|  |  |
|  | End Sub |
|  |  |
|  | ''' <summary>出力電圧を算出して表示</summary> |
|  | Private Sub CalculateVout\_Click(ByVal sender As System.Object,  ByVal e As System.EventArgs) Handles CalculateVout.Click |
|  |  |
|  | Dim vin, vout, f As Decimal '入力電圧、出力電圧、周波数 |
|  |  |
|  | 'テキストボックスに入力された文字列を数値に変換して代入 |
|  | f = SIPrefixToDecimal(InputFrequency.Text) '周波数[Hz] |
|  | vin = 1 '入力電圧を1V |
|  | 'ハイパスフィルタに電圧vinと周波数fを入力して出力電圧を取得 |
|  | vout = hpf.GetOutputVoltage(vin, f) |
|  |  |
|  | '出力電圧を表示 |
|  | DisplayVout.Text = vout.ToString("0.###V") |
|  |  |
|  | End Sub |
|  |  |
|  | ''' <summary>SI接頭辞が含まれる文字列を数値に換算するメソッド</summary> |
|  | ''' <param name="numberWithSIPrefix">SI接頭辞が末尾にある数値の文字列</param> |
|  | ''' <returns>SI接頭辞がない数値</returns> |
|  | Private Function SIPrefixToDecimal(ByVal numberWithSIPrefix As String)  As Decimal |
|  |  |
|  | Dim siPrefix As Char '末尾の文字 |
|  | Dim number As Decimal 'SI接頭辞がない数値 |
|  | '末尾の1文字を取得 |
|  | siPrefix = numberWithSIPrefix.Chars(numberWithSIPrefix.Length - 1) |
|  |  |
|  | '末尾の1文字を評価 |
|  | If Char.IsDigit(siPrefix) = True Then '末尾が数値のとき |
|  | number = Convert.ToDecimal(numberWithSIPrefix) |
|  | Else |
|  | number = Convert.ToDecimal(numberWithSIPrefix.TrimEnd(siPrefix)) |
|  | Select Case siPrefix |
|  | Case "u" |
|  | number = number \* (10 ^ -6) '"uを換算" |
|  | Case "m" |
|  | number = number \* (10 ^ -3) '"mを換算" |
|  | Case "k" |
|  | number = number \* (10 ^ 3) '"kを換算" |
|  | Case Else 'それ以外の文字は例外とする |
|  | Throw New FormatException("入力された文字が適切ではありません。") |
|  | End Select |
|  | End If |
|  |  |
|  | Return number 'SI接頭辞がない数値を渡して復帰 |
|  |  |
|  | End Function |
|  |  |
|  | End Class |