Практическая работа №7

Построение и оцифровка аналогового сигнала
на основе графических средств языка Visual Basic

Цель работы

Цель практической работы – разработать программу для графической визуализации аналогового сигнала (например, электрического тока или напряжения), заданного простейшей функцией , где *А ‑*  амплитуда сигнала, *ν* ‑ частота сигнала в герцах, *t –* момент времени, *ϕ* ‑ фазовый сдвиг. Программа должна позволять моделировать параметры сигнала и визуализировать соответствующие изменения в графике функции..

В ходе выполнения работы студент должен освоить приемы программирования графических изображений средствами языка Visual Basic, использовать элементы объектно-ориентрованного программирования, а также закрепить знания и навыки алгоритмизации задач.

Практическая работа рассчитана на 2 занятия, т.к. предполагает большой объем самостоятельного программирования.

I. Работа с графикой стандартными средствами Visual Basic

Для выполнения практической работы рассмотрим некоторые графические возможности языка Visual Basic (VB), необходимые для проектирования поставленной задачи. Для VB основой любого приложения являются формы. Элементы управления, практически, те же, что и для VBA. Рассмотрим, как включить в форму изображения, и какими операторами их создавать.

Для графического оформления в форме можно использовать элементы управления *Image* и *PictureBox* (рис. 4.1)

Для обоих элементов *Image* и *PictureBox* ключевое значение имеет свойство *Picture*, которое позволяет задать графический файл, откуда берется изображение.

Одним из важных свойств Picture является его ***масштаб***, который определяется свойством *ScaleMode*. При построении графиков функций наиболее удобно задавать *произвольный масштаб* методом *Scale.* Для задания такого масштаба необходимо, чтобы метод *ScaleMode* имел значение *vbUser* (или значение 0). Например, чтобы нарисовать синусоиду, можно ввести масштаб от - (2π) до 2π по оси *x*, от -1 до 1 по оси *y* следующим образом (переменная pi равна π):

Picture1.Scale (-2\*pi, 1)-(2\*pi, -1)

Рассмотрим ***методы вывода*** графики. В квадратных скобках – необязательные параметры.



Элемент управления PictureBox

Элемент управления Image

PictureBox в форме

Image в форме

Кнопка с троеточием, открывающая окно загрузки рисунка

Рис. 4.1. Элементы управления *Image* и *PictureBox* в форме

Метод **PSet** [*Step*] (*x, y*), [*color*] рисует точку с заданными координатами (*x, y*) и цветом *color*.

Метод может использоваться следующим образом:

*PSet Step*(*x, y*) ‑ рисует точку, расположенную на расстоянии *x* по горизонтали и *y* по вертикали от предыдущей точки. Например:

Picture1.Pset (10, 10) ' рисуем точку с координатами 10, 10

Picture1.Pset Step (20, 10) ' рисуем точку с координатами 30, 20 т.е. отстоящую от предыдущей нарисованной точки на расстоянии 20, 10.

Метод **Line** [*Step*] (*x1, y1*) [*Step*] (*x2, y2*), [*color*], [*B*][*F*] рисует линию или прямоугольник с заданными координатами координатами вершин (*x1, y1*) и (*x2, y2*) и цветом *color*

Метод может использоваться следующим образом:

*Line Step* (*x1*, *y1*) - (*x2*,*y2*) - рисует линию, где *x1*, *y1* - расстояние от предыдущей точки

*Line* (*x1*, *y1*) - Step (*x2*, *y2)* - рисует линию из точки (*x1*, *y1*) в точку, отстоящую от нее на расстояние *x2*, *y2*

*Line* - (*x*, *y*) - рисует линию из предыдущей точки в точку с координатами (x, y)

*Line* (*x1*, *y1*) - (*x2*, *y2*), , *B* - рисует прямоугольник

*Line* (*x1*, *y1*) - (*x2*, *y2*), , *BF* - рисует прямоугольник закрашенный цветом линий.

Например:

Picture1.Line (10, 10) - (20, 20) ' рисуем линию из точки (10, 10) в точку (20, 20)

Picture1.Line Step (10, 20) - (40, 50) ' рисуем линию из точки (30, 40) в точку (40, 50)

Picture1.Line (50, 50) - Step (20, 30) ' рисуем линию из точки (50, 50) в точку (70, 80)

Picture1.Line (90, 90) ' рисуем линию из точки (70, 80) в точку (90, 90)

Метод **Circle** [*Step*] (*x*, *y*), *radius*, [*color*, *start*, *end*, *aspect*] рисует окружность, дугу или эллипс. Здесь:

*x*, *y* - координаты центра,

*radius* - радиус окружности,

*color* - цвет окружности,

*start*, *end* - начало и конец дуги в радианах (по умолчанию *start*= 0, *end*= 6.28)

*aspect* определяет степень сжатия эллипса. По умолчанию *aspect*= 1. Если *aspect*> 1, эллипс будет вытянут по вертикали, если *aspect*< 1 - по горизонтали. При этом максимальный диаметр эллипса будет равен 2 × *radius*. Например:

Picture1.Circle (20, 20), 10 ' рисуем окружность с координатами центра (20, 20) и радиусом 10

pi=3.14

Circle (20, 20), 10, , pi/2, 3\*pi/2 ' рисуем дугу от угла 90 градусов до 270 градусов

Circle (20, 20), 10, , , , 2 ' рисуем эллипс с высотой 10 и шириной 5

Для ***вывода текста*** существует метод **Print**. Этот метод может быть определен и для формы и для элемента *PictureBox.* Положение текста определяется свойствами *CurrentX* и *CurrentY*. Например:

CurrentX = 10

CurrentY = 10

Print "text" ' Печатаем текст на форме, левый верхний угол которого соответствует координатам (10, 10)

Picture1.Print "Печать в PictureBox"

Теперь рассмотрим, каким образом можно задать ***цвета***  встроенным на форму объектам. Цвет состоит из трех составляющих - красная, зеленая и синяя. Каждая составляющая может изменяться от 0 до 255. Цвет кодируется следующим образом:

red + 256\*green + 65536\*blue или red + 28\*green + 216\*blue

Эту запись позволяет упростить *функция* *RGB* (red, green, blue). Например, желтый цвет = *RGB* (255, 255, 0).

В идеальном случае можно вывести 2563 = 16777216 цветов (24 бита), некоторые из этих цветов представлены в табл. 4.1. В реальном случае количество цветов зависит от экранных настроек. В таблице 4.2 представлены цвета для параметров функции *RGB*.

Таблица 4.1. Некоторые параметры функции RGB

|  |  |  |  |
| --- | --- | --- | --- |
| RGB(255,255,255)  | белый  | RGB(0,255,255)  | бирюзовый  |
| RGB(255,0,0)  | красный  | RGB(255,0,255)  | пурпурный  |
| RGB(0,255,0)  | зеленый  | RGB(255,255,0)  | желтый  |
| RGB(0,0,255)  | синий  | RGB(192,192,192)  | серый |

Помимо функции RGB есть *функция* *QBColor*(*color*), где *color* = 0÷15, которая возвращает 16 наиболее часто используемых цветов (табл. 4.2).

Таблица 4.2. Параметры аргумента *Color.*

|  |  |  |  |
| --- | --- | --- | --- |
| **Число** | **Цвет** | **Число** | **Цвет** |
| 0 | Черный | 8 | Серый |
| 1 | Синий | 9 | Светло-синий |
| 2 | Зеленый | 10 | Светло-зеленый |
| 3 | Голубой | 11 | Светло-голубой |
| 4 | Красный | 12 | Светло-красный |
| 5 | Пурпурный | 13 | Светло-пурпурный |
| 6 | Желтый | 14 | Светло-желтый |
| 7 | Белый | 15 | Ярко-белый |

Помимо обычных цветов существуют системные цвета, т.е. цвета, установленные по умолчанию системой для кнопок, текстовых полей и т.д. Чтобы не спутать эти цвета с обычными они кодируются отрицательными числами. Чтобы получить коды этих цветов можно использовать *предопределенные константы* (например, *vbButtonFace*).

Пример:

Private Sub Command1\_Click()

CurrentX = 10: CurrentY = -20: Picture1.Print vbButtonFace’ на объекте *PictureBox1* будет напечатано отрицательное значение цвета кнопки Command1

End Sub

Предопределенные константы (*vbBlack ‑* черный, *vbRed ‑* красный, *vbGreen ‑* зеленый, *vbYellow ‑* желтый, *vbBlue ‑* синий, *vbMagenta ‑* пурпурный, *vbCyan ‑* голубой, *vbWhite –* белый) могут быть использованы и для задания цвета любых объектов:

Picture1.Line (0, 0)-(11, 0), vbRed’ линия красного цвета

***Цвет фона графического окна*** определяется свойством *BackColor*. ***Цвет линий*** ‑ свойством *ForeColor*. Цвет, заданный этим свойством, является по умолчанию цветом для графических методов.

Например:

ForeColor = vbBlue

Line (10, 10) - (20, 10), vbRed ' рисуем красную линию

Line (20, 10) - (20, 20) ' рисуем синюю линию

***Толщина линий*** определяется свойством *DrawWidth*.

Если *DrawWidth* = 1, то метод *Pset*, если в нем не задать цвет, выведет точку черным цветом независимо от свойства *ForeColor* (работает только для VB5 и более ранних версий).

Свойство *DrawStyle* определяет ***тип линии***. Если для свойства *DrawWidth* установлено значение больше 1, то свойство *DrawStyle*, независимо от записанного параметра(1 до 4), всегда выводит сплошную линию, значение *DrawStyle*при этом не изменяется. Если *DrawWidth* имеет значение 1, то *DrawStyle*создает эффект, описанный для каждого параметра в таблице 3.

Например:

Picture1.DrawWidth = 2

Picture1.DrawStyle = vbDash’ нарисуется сплошная линия

Picture1.DrawWidth = 1

Picture1.DrawStyle = vbDash’ нарисуется пунктирная линия

Таблица 3. Значения констант, допустимые для свойства *DrawStyle*.

|  |  |  |
| --- | --- | --- |
| **Константа** | **Значение** | **Описание** |
| **vbSolid** | 0 | Сплошная (по умолчанию) |
| **vbDash** | 1 | Тире |
| **vbDot** | 2 | Точка |
| **vbDashDot** | 3 | Тире - точка |
| **vbDashDotDot** | 4 | Тире (- кнопку с многоточием |

II. Объект ScrollBar и его свойства

Почти любая программа под Windows имеет полосы прокрутки, ползунки, позволяющие пользователю перемешаться по данным вверх-вниз или влево-вправо. Этот объект называется *ScrollBar*. Сам объект получает имя по умолчанию *VScroll1*, если это вертикальный ползунок, или*HScroll1*, если - горизонтальный. Свойства и события у обоих одинаковые:

*.Value* - это главное значение, оно меняется, когда мы двигаем ползунок ScrollBar ' а*.Max ‑* задает максимальное значение, до какого будет меняться значение

*.Min*  ‑ задает минимальное значение *.Value*. Здесь тоже ограничения до -32768.

 Главный недостаток объекта *Scroll* в том, что значение *Max*, *Min,*  а, стало быть, и *Value* не может превышать 32767 (215-1). Это всегда надо учитывать при работе с этим объектом.

*.SmallChange* ‑ задает, насколько меняется *.Value*, если щелчок мыши делается по стрелочкам вверху или внизу объекта *ScrollBar*

*.LargeChange ‑* задает, насколько меняется *.Value*, если щелкнуть мышью по полю *ScrollBar*, а не по стрелочке.

Чтобы посмотреть, как работает этот объект, создадим новый проект и положим на форму *Text1* и *VScroll1*. Затем дважды щелкнем мышью по объекту *VScroll1*. У нас откроется окно кода и создастся процедура *VScroll1\_Change()***.**

Впишем в нее такую строчку кода: *Text1.Text = VScroll1.Value* и запустим программу. Как только мы начинаем двигать ползунок, в текстовом поле отражается значение *VScroll1.Value* от 0 до 32767.

Остановим программу и создадим процедуру *Form\_Load()***,** выбрав в верхнем левом окне кода значение *Form* и щелкнув по нему. Впишем в эту процедуру значения свойств, о которых мы только что говорили. Должно получиться следующее:

Option Explicit 1’Обязательное описание всех переменных

Private Sub Form\_Load()

VScroll1.Value = 0

VScroll1.Max = 32767

VScroll1.Min = -32768

VScroll1.LargeChange = 100

VScroll1.SmallChange = 1

End Sub

Private Sub VScroll1\_Change()

Text1.Text = VScroll1.Value

End Sub

Чтобы ползунок не мигал, поставим в окне свойств *VScroll1*свойство *TabStop* в значение *False*. Запустим программу. Значение *.Value* мы задали 0, а поскольку интервал у нас от 32767 до -32768, то ползунок стоит посередине. При щелканье мыши по стрелочкам вверху и внизу, значение *.Value* изменяется на 1, а при щелканье по полю *VScroll1* - на 100. Рис. 4.2 иллюстрирует описанные величины.

 

«клик» -1

«клик» 1

0

«клик» -100

«клик» 100

Рис. 4.2 Соотношение числового значения с положением и скоростью перемещения бегунка на полосе прокрутки.

III. Разработка проекта для построения графика аналогового сигнала

1. Построение элементов формы

Прежде чем разрабатывать проект, введем обозначения для переменных (мнемонические имена), описывающих гармонический сигнал:

*А ‑*  амплитуда сигнала – *“a”*, *ν* ‑ частота сигнала в герцах – *“nu”*, *t –* момент времени – “t”, *ϕ* ‑ фазовый сдвиг – “phi”.

Создадим форму (рис. 3), на основе, которой будет создан проект построения и оцифровки гармонического сигнала.



Рис. 3 Форма с элементами для разработки проекта построения изображения гармонического сигнала

На форме встроены объекты *PictureBox,* семь текстовых полей, три горизонтальных прокрутки, три кнопки и соответствующие надписи.

1. Разработка кода основных элементов формы

Создать код для встроенных элементов согласно следующим текстам программ. Изучить значение переменных по комментариям программы.

а) шапка проекта с обязательным описанием всех используемых переменных

Option Explicit

Dim nu As Single ‘ частота

Dim phi As Single ‘ фаза

Dim T As Single, dt As Single, tt As Integer, ttr As Single’ период, шаг по времени, вспомогательные переменные для времени

Dim dpx As Byte, tscale As Integer, tmax As Single, sppx As Single

Public a As Single, amax As Single

Dim x As Integer, y As Single ' координаты рисунка

Dim cmnd1\_click As Boolean ' переменная, ответственная за

Dim nu\_r As Single, pi As Single

Dim vbColor As Byte

Dim n As Byte

Dim da, dtd As Single

б) параметры объектов, встроенных на форму

Private Sub Form\_Load()

cmnd1\_click = False’ логическая переменная для изображения изменяемого параметра в поле text6

pi = Atn(1) \* 4

tmax = 1 \* 2 \* pi ' левая граница экрана в радианах

amax = 5 ' max y – максимальное значение по амплитуде

tscale = 500 ' число пикселей по оси x – соответствует времени 1 секунда

' 1 px=2e-3 s (sspx). We take step in pixel: dpx=1

sppx = tmax / tscale ' значение пиксела в радианах

dt = 1 ' шаг по времени в пикселах

Form1.Caption = "PАЗВЕРТКА СИГНАЛА ПО ВРЕМЕНИ t" ' это заголовок формы (вместо Form1)

Rem Параметры для второй полосы прокрутки‑ изменение частоты

HScroll2.Value = 51

HScroll2.Min = 1

HScroll2.LargeChange = 10

HScroll2.SmallChange = 1

HScroll2.Max = 150

nu = 1

Text3.Text = Str(nu)

T = 1 / nu ' период в секундах

Rem Параметры для третьей полосы прокрутки‑ изменение фазы

HScroll3.Value = 0

HScroll3.Min = 0

HScroll3.LargeChange = 2

HScroll3.SmallChange = 1

HScroll3.Max = 10

phi = 0

Text4.Text = Str(phi)

Text5.Text = "1с"

Text6.Text = Str(amax)

Text7.Text = Str(T)

vbColor = 0

End Sub

в) Программа для построения осей графика

Sub frame()

'Параметры графика

Picture1.ScaleMode = vbUser ' 0

Picture1.Scale (0 - 10, amax)-(tscale + 10, -amax) ' размер экрана

Picture1.BackColor = vbWhite 'белый фон

Picture1.DrawWidth = 1

Picture1.ForeColor = vbBlack ' QBColor(0)

Picture1.Line (0, 0)-(tscale, 0) ' x-axis

Picture1.Line (0, amax)-(0, -amax) 'y-axis

Label5.Visible = False

Text1.Visible = False

End Sub

г) Программа для построения графика гармонического сигнала

Sub signal()

Picture1.ForeColor = vbBlue ' QBColor(1 + 8)

Picture1.DrawWidth = 2 'line width

Picture1.PSet (0, 0)

For tt = 0 To tscale Step dt 'в пикселах

ttr = tt \* sppx ' в радианах

y = a \* Sin(nu \* ttr + phi)

x = tt

Picture1.Line -(x, y), QBColor(1 + vbColor)

Next tt

If cmnd1\_click Then Label5.Visible = True: Text1.Visible = True

End Sub

д) Ниже представлены программы для события ‑ изменение значение полосы прокрутки *HScroll2* и *HScroll3* и вызываемых ими функций. Внимательно разобрать алгоритм обеих программ и сопровождающих функций.

Private Sub HScroll2\_Change()

vbColor = IIf(vbColor > 13, 1, vbColor + 1) ’ выбор цвета графика

nu = f\_nu(HScroll2.Value) 'определяем значение частоты по значению бегунка

Text3.Text = Str(nu)

T = 1 / nu ' период в секундах

Text7.Text = Str(T)

If cmnd1\_click Then signal ' автоматическое воспроизводство графика после изменения параметра

Text1.Text = "nu" ’ в поле показана изменяемая переменная

End Sub

Здесь частота *nu* определяется функцией *f\_nu* от значения положения бегунка. Формально, если бегунок стоит в середине полосы прокрутки (*Value=51*), то частота равно 1 Гц (1 колебание в секунду). Если бегунок сдвигается влево (Value <51), то частота возрастает (2, 3,…100 Гц), если вправо – то частота убывает (1/2 Гц, 1/3 Гц,…,1/100Гц).

Function f\_nu(n)

If n <= 50 Then

f\_nu = 100 - (n - 1) \* 2

ElseIf n > 51 Then

f\_nu = 1 / (n - 50)

Else: f\_nu = 1

End If

End Function

Фаза *phi* определяется функцией *f\_ phi* от значения положения бегунка. Формально, изменение положения бегунка на 1 позицию производит изменение фазы на 1/10 текущего периода *Т* (в радианах).

Private Sub HScroll3\_Change()

vbColor = IIf(vbColor > 13, 1, vbColor + 1) ’ выбор цвета графика

n = HScroll3.Value

phi = f\_phi(n) 'Записываем текущее значение HSvroll1 в переменную phi.

Text4.Text = Str(n / 10)

If cmnd1\_click Then signal ' автоматическое воспроизводство графика после изменения параметра

Text1.Text = "phi"

End Sub

Function f\_phi(n)

f\_phi = T / 10 \* n \* 2 \* pi

End Function

1. Самостоятельная разработка кода для элементов формы

После внесения кодов основных элементов записываем программы для трёх кнопок.

а) Кнопка «Вывести оси координат» вызывает программу *frame.*

б) Кнопка «Очистить экран» обновляет поле рисунка, используя единственную команду Picture1.Picture = LoadPicture().

в) Кнопка «Задать сигнал» вызывает программу построения сигнала *signal* и меняет значение переменной cmnd1\_click на истину.

***Проверить работоспособность кнопок и полос прокрутки*** для изменения частоты и фазы сигнала.

г) ***Разработать код для изменения положения бегунка*** на полосе прокрутки HScroll1\_Change(), ответственного за изменение амплитуды *а* гармонического сигнала*.*

Для этого вначале в программе *Form\_Load*() необходимо описать параметры первой полосы прокрутки *HScrool1.* Максимальные и минимальные значения согласовать со значением *amax.* Свойство *Value* и, соответственно, величина *a* должны иметь значение 1.

Программа, отслеживающая изменение бегунка на полосе прокрутки, должна выполнять следующие действия:

1. изменять значение переменной *vbColor* для задания нового цвета при построения графика с измененной амплитудой;

2. присваивать амплитуде *а* значение Value, связанное с положением бегунка;

3. в поле *Text2* выводить текущее числовое значение *а*;

4. если переменная cmnd1\_click истинна, то вызвать построение графика сигнала с новой амплитудой;

5. в поле *Text1* выводить символ амплитуды «А».

д) В пустом пространстве формы, справа от объекта *PictureBox,* ***вывести формулу***  (рис. 4). Для этого необходимо сделать следующее:

1. набрать в Microsoft Equation формулу и сохранить её изображение в графическом файле, используя, например, редактор Paint;

2. создать объект Imagine (рисунок) слева от *PictureBox;*

3. в код программы *Form\_Load* ввести следующий оператор:

Image1.Picture = LoadPicture("полный адрес файла с рисунком формулы")

е) Создать кнопку программного завершения работы проекта. Перед тем, как выполнить команду завершения *Form1.Hide,* желательно вставить операторы, предупреждающие о завершении проекта и подтверждении выхода из программы.

Результаты работы программы с изменением параметров могут выглядеть, например, так, как показано на рис. 4.



Поработать с построенным проектом, проверить работоспособность всех кнопок. Показать результаты работы преподавателю.

\* \* \*

Контрольные вопросы

1. Перечислить операторы создания изображения точки, линии и окружности на форме VBA.
2. Какие цвета соответствует функциям RGB(0,255,0) и RGB(0,0,0)?
3. Какой цвет соответствует функции *QBColor*(*13*)?
4. Указаны следующие параметры полосы прокрутки:

VScroll1.Value = 50

VScroll1.Max = 100

VScroll1.Min = 0

Это вертикальная или горизонтальная полоса? Где будет находиться бегунок при визуализации полосы?

1. Допустимы ли следующие параметры полосы прокрутки?

VScroll1.Value = 0

VScroll1.Max = 100000

VScroll1.Min = -150000

1. Что делает оператор Picture1.ForeColor = vbBlue
2. Поясните значение переменной vbColor в программе signal. В какой программе контролируется её значение?