**Приложение для наглядного ознакомления с ДПФ Фурье и его представлением**

*(Приложение разработано в учебных целях)*

Оглавление

[Постановка и спецификация задачи 3](#_Toc73890038)

[Описание программного интерфейса(Use Case) 4](#_Toc73890039)

[Диаграмма процесса использования программы 4](#_Toc73890040)

[Определение списка объектов пользовательского интерфейса 5](#_Toc73890041)

[Определение действий, выполняемых в пользовательском интерфейсе и элементы управления 7](#_Toc73890042)

[Диаграмма классов 8](#_Toc73890043)

[Диаграмма потоков данных(Data Flow Diagram) 9](#_Toc73890044)

[Код проекта с комментариями 9](#_Toc73890045)

[Proj\_letiDlg.cpp 9](#_Toc73890046)

[Proj\_leti.cpp 21](#_Toc73890047)

[PaintPic.cpp 23](#_Toc73890048)

[PaintPic2.cpp 24](#_Toc73890049)

[Cproj\_letiCalc.cpp 26](#_Toc73890050)

[Cproj\_letiCalcDPF.cpp 27](#_Toc73890051)

[stdafx.cpp 29](#_Toc73890052)

[Proj\_letiDlg.h 29](#_Toc73890053)

[Proj\_leti.h 30](#_Toc73890054)

[PaintPic.h 30](#_Toc73890055)

[PaintPic2.h 31](#_Toc73890056)

[Cproj\_letiCalc.h 31](#_Toc73890057)

[Cproj\_letiCalcDPF.h 32](#_Toc73890058)

[Stdafx.h 33](#_Toc73890059)

[Выбор и обоснование собственных типов,переменных,классов 34](#_Toc73890060)

[Вводимые классы 34](#_Toc73890061)

[Вводимые типы и структуры 34](#_Toc73890062)

[Примеры работы программы и контрольный пример 35](#_Toc73890063)

[Примеры работы программы 35](#_Toc73890064)

[Контрольный пример 39](#_Toc73890065)

[Выводы и особенности программы 41](#_Toc73890066)

# Постановка и спецификация задачи

Разработать GUI приложение с использованием библиотеки MFC,которое для сигнала заданного вида вычисляет дискретное преобразование Фурье (ДПФ),рисует график сигнальной функции и график модуля ее ДПФ.

Формулировка задания:

**“**Для сигнала заданного вида вычислить дискретное преобразование Фурье (ДПФ), нарисовать график сигнальной функции и график модуля ее ДПФ, предоставить возможность изменять масштаб графика по Х и по У, выбор линейного или логарифмического масштаба по У для графика ДПФ, интерактивный режим изменения параметров сигнала, запись в файл графиков сигнала и ДПФ в формате BMP.**”**

Техническое задание:

**Сигнал x(t)=a\*sin(2\*PI\*(f +m\*sin(2\*PI\*Fm\*t)\*t))** – сигнал с непрерывной частотной модуляцией. N=100 … 1000; Fд=10 МГц; f=[0.1\*Fд…0.4\*Fд); Fm=[0.01 Fд…0.09\*Fд); m=[0…f/10];

**2.1. Требования к расчету и программе из задания**

2.1.1 Имеется возможность построения графика сигнальной функции

2.1.2 Имеется возможность построения графика модуля ДПФ сигнальной функции

2.1.3 Имеется возможность интерактивно изменять размер графика с помощью ввода аргумента

2.1.4 Имеется возможность сохранения графиков сигнальной функции и ее модуля ДПФ в формате BMP,PNG,GIF,JPG.

2.1.5 Имеется возможность изменять масштаб графика по Х и по У, выбор линейного или логарифмического масштаба по У для графика ДПФ

2.1.6 Имеется дополнительное вызываемое меню с функциями(пунктами меню*”помощь” и “выход”(Примечание:*активация пункта “Помощь” выводит подсказки для пользователя на главном окне приложения)

**2.2. Требования к отчету**

* + 1. Отчёт должен соответствовать ГОСТу 19.701-90 единой системы программной документации
    2. В отчёт необходимо включить описание программного интерфейса, диаграмму классов и диаграмму потоков данных, выбор и обоснование переменных, собственных типов и классов, код с комментариями, пример работы программы и контрольный пример
    3. Контрольный пример должен быть представлен в виде расчётов и графиков в программе Mathcad или других интернет-ресурсов, подтверждающих правильность результата

**2.3. Требования к пользовательскому интерфейсу программы**

2.3.1 Имеются сведения о программе, об авторских правах, об авторе

2.3.2 Имеется название и иконка приложения

2.3.3 Имеются окна для вывода графиков

2.3.4 Используются кнопки управления приложением

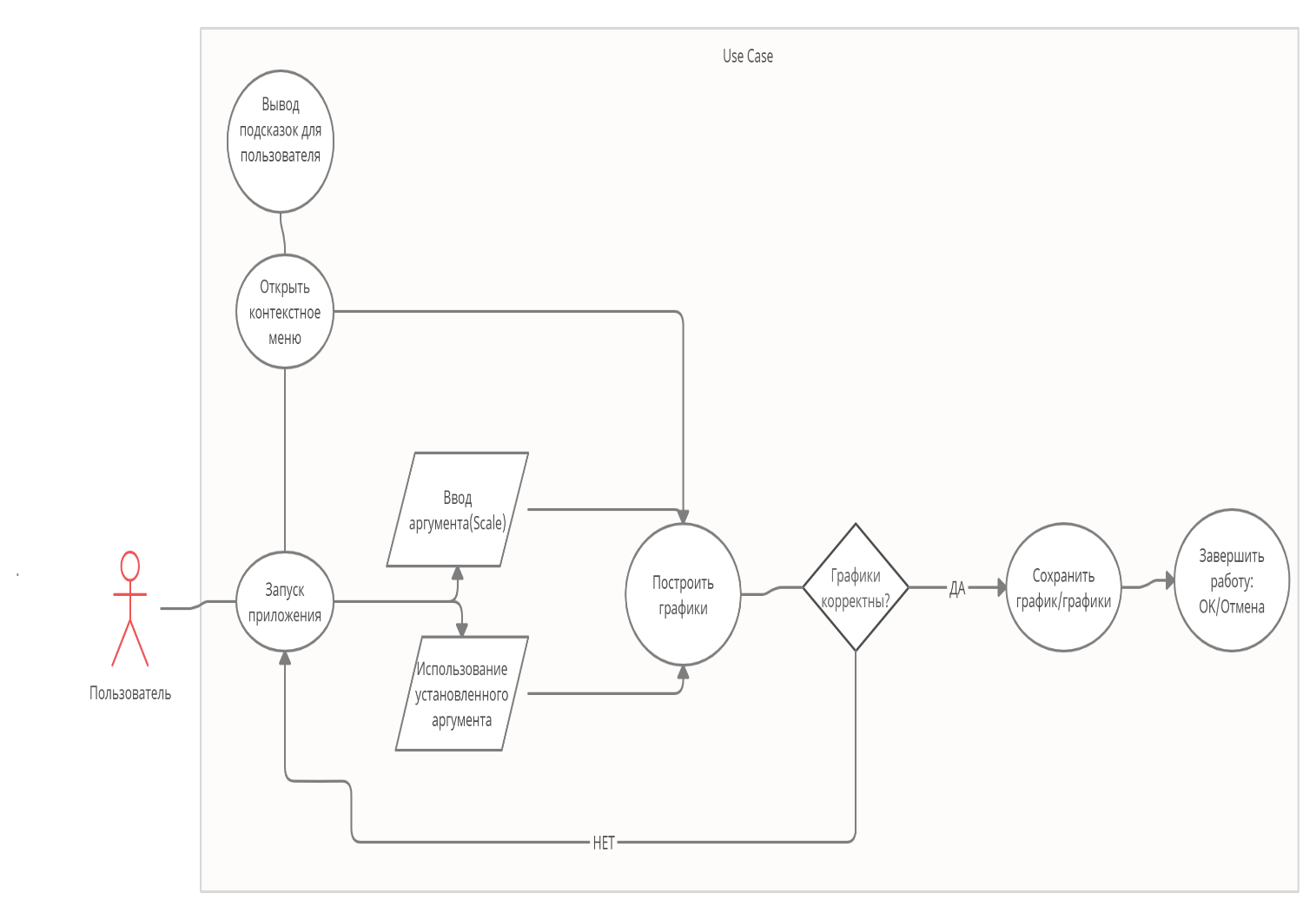
2.3.5 Используются поля ввода данных для расчетов

2.3.6 Имеется соответствие понятия "дружественный интерфейс"

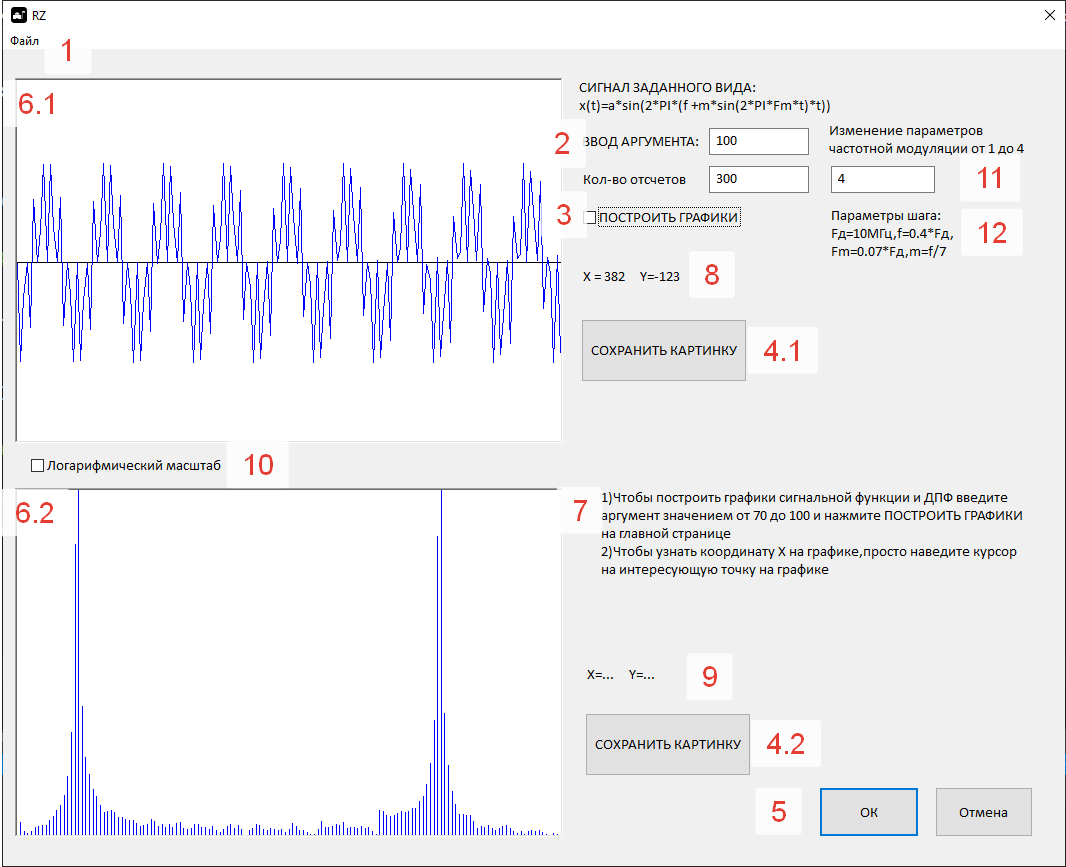
2.3.7 Используется русский язык

# Описание программного интерфейса(Use Case)

### Диаграмма процесса использования программы



### Определение списка объектов пользовательского интерфейса



Для каждого следующего объекта ПИ(Здесь и далее ПИ-Программный интерфейс):

№.Функция-тип-ID

1. Вызов контекстного меню - тип Menu – IDR\_MENU\_DOP
2. Ввод аргумента(Arg) в интервале от 70 до 100 - тип Edit Control – IDC\_EDIT1
3. Построить графики - тип Check Box с 2-мя режимами работы – IDC\_CHECK3
4. Сохранить картинки графиков-тип Button - IDC\_SAVE\_PIC(Для 6.1),IDC\_SAVE\_PICDPF(Для 6.2)
5. Кнопки ОК/Отмена для завершения работы приложения - тип BUTTON – IDOK/IDCANCEL
6. Окна в которых происходит отрисовка графиков – тип Animation Control-IDC\_ANIMATE1(6.1),IDC\_ANIMATE2(6.2)
7. Статический текст вызываемый из контекстного меню- тип Static Text-IDC\_HELP
8. Статический текст с координатами для IDC\_ANIMATE1-тип Static Text – IDC\_ST\_SH
9. Статический текст с координатами для IDC\_ANIMATE2-тип Static Text – IDC\_ST\_SH\_DPF
10. .Чек-бокс для логарифмического увеличения графика-Check Box с 2-мя режимами работы -IDC\_CHECK\_LOG
11. Ввод шага параметров-Edit Control-IDC\_EDIT\_STEP
12. Статичный текст с выводом параметров шага-Static Text-IDC\_STATIC\_STEP

### Определение действий, выполняемых в пользовательском интерфейсе и элементы управления

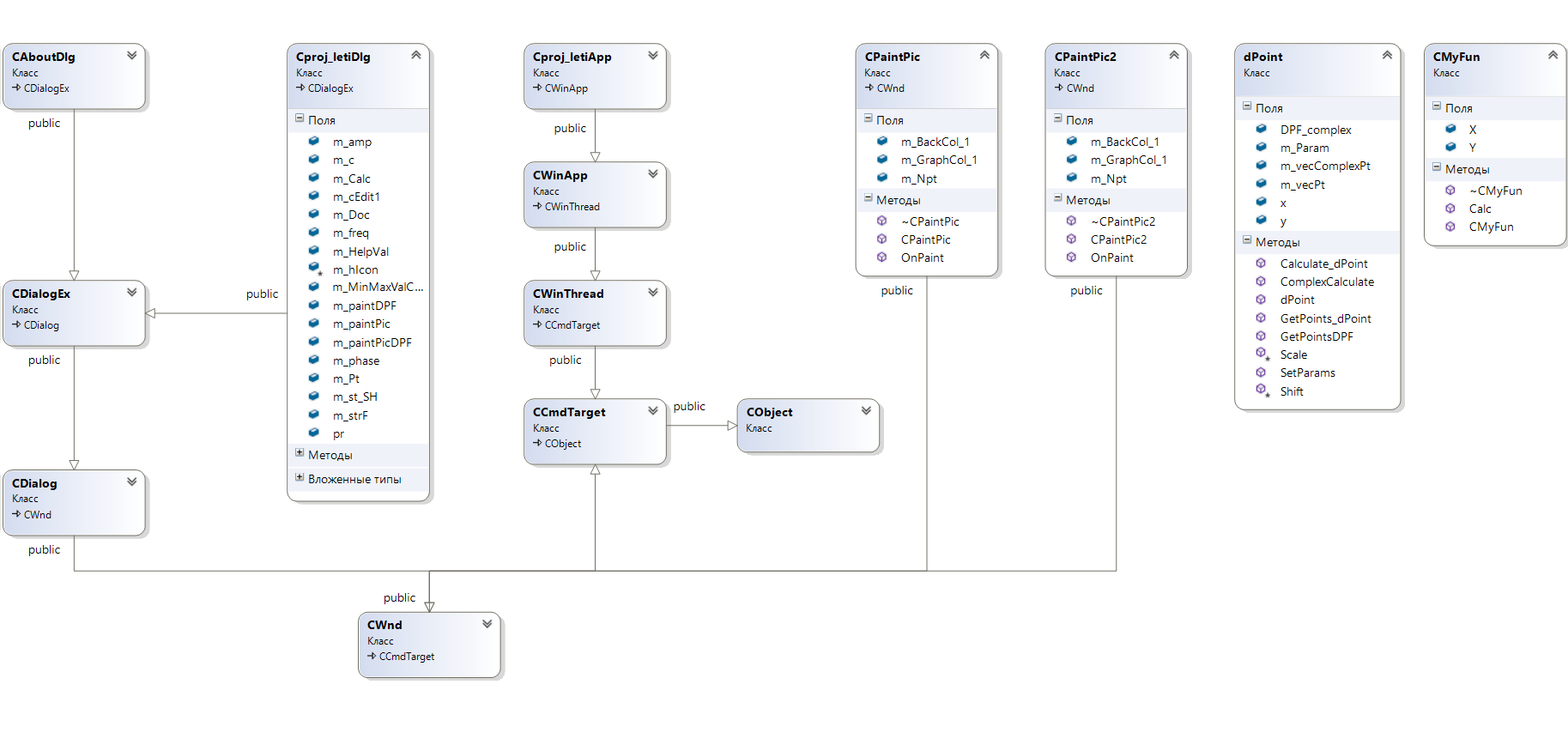
1 Дополнительное(контекстное) меню : по нажатию открывает список возможных действий(Помощь,Выход)

2 Ввод аргумента(Scale) для масштабирования размеров графиков

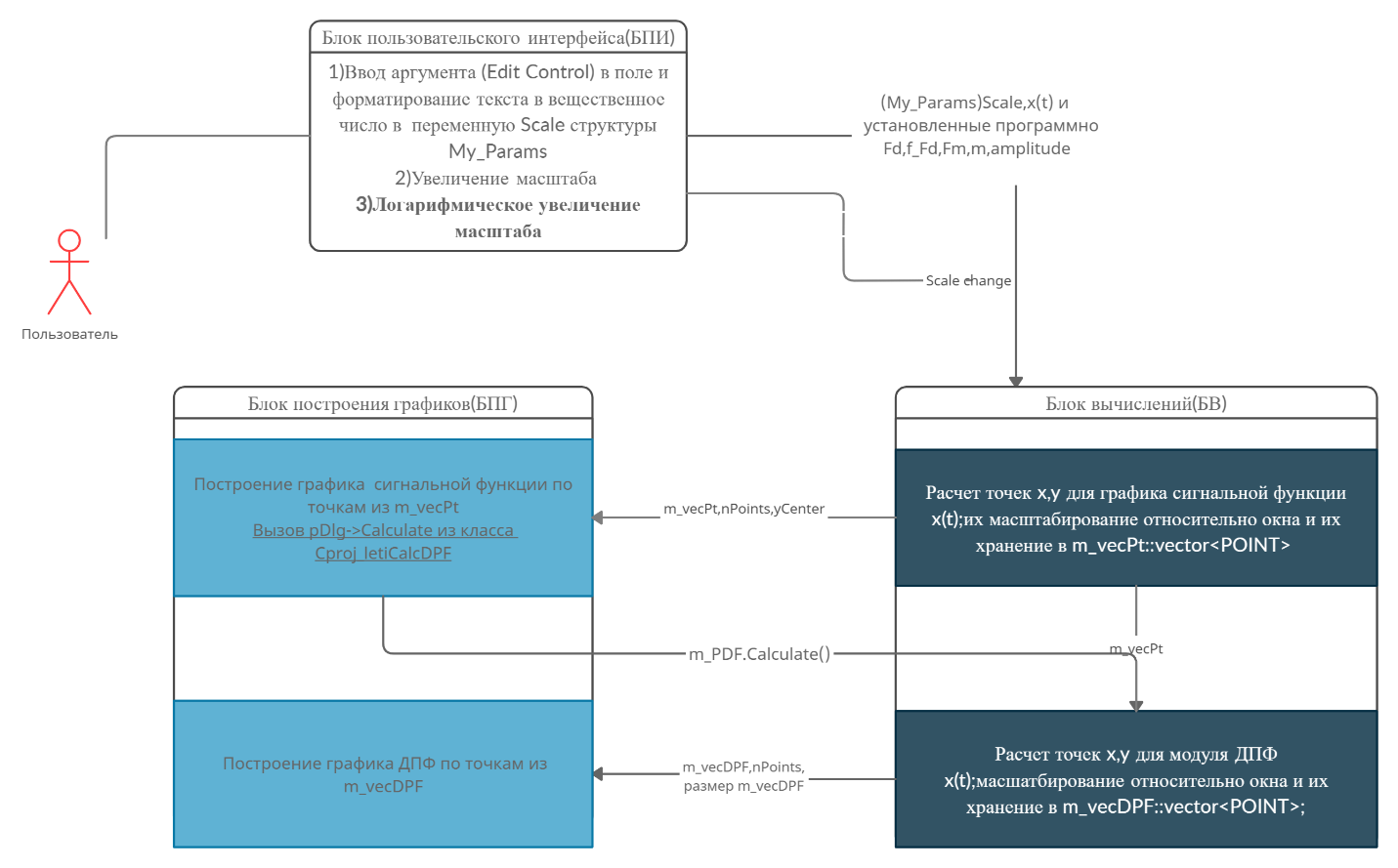
Ввод номера шага параметров

1. Построение графиков по нажатию
2. Перерасчет количества точек(установка нового количества)
3. Логарифмическое увеличение графика путем нажатия чек-бокса или увеличение графика с помощью колеса мышки по наведению на окно графика
4. Сохранение графиков в форматах:BMP,JPG,GIF,PNG по нажатию
5. Завершение работы программы нажатием кнопки OK с сохранением аргумента(Scale) посредством записи значения Scale в текущей сессии работы программы в реестр Windows
6. Завершение работы программы/Выход из программы –“Отмена”

# Диаграмма классов



# Диаграмма потоков данных(Data Flow Diagram)



# Код проекта с комментариями

## Proj\_letiDlg.cpp

// proj\_letiDlg.cpp : файл реализации

//Подключаем заголовочные файлы и мат.библиотеки

#include "stdafx.h"

#include "proj\_leti.h"

#include "proj\_letiDlg.h"

#include "afxdialogex.h"

#include "Cproj\_letiCalc.h"

#include "PaintPic.h"

#include <cstdlib>

#include <initguid.h>

#include<vector>

#include "math.h"

#define \_USE\_MATH\_DEFINES

#ifdef \_DEBUG

#define new DEBUG\_NEW

#endif

//define

//Выставляем ID дял каждого таймера

#define MY\_TIMER1 1111

#define MY\_TIMER2 2222

#define PI 3.14

//конфигурация дял сохранения Image

DEFINE\_GUID(ImageFormatBMP, 0xb96b3cab, 0x0728, 0x11d3, 0x9d, 0x7b, 0x00, 0x00, 0xf8, 0x1e, 0xf3, 0x2e);

DEFINE\_GUID(ImageFormatEMF, 0xb96b3cac, 0x0728, 0x11d3, 0x9d, 0x7b, 0x00, 0x00, 0xf8, 0x1e, 0xf3, 0x2e);

DEFINE\_GUID(ImageFormatWMF, 0xb96b3cad, 0x0728, 0x11d3, 0x9d, 0x7b, 0x00, 0x00, 0xf8, 0x1e, 0xf3, 0x2e);

DEFINE\_GUID(ImageFormatJPEG, 0xb96b3cae, 0x0728, 0x11d3, 0x9d, 0x7b, 0x00, 0x00, 0xf8, 0x1e, 0xf3, 0x2e);

DEFINE\_GUID(ImageFormatPNG, 0xb96b3caf, 0x0728, 0x11d3, 0x9d, 0x7b, 0x00, 0x00, 0xf8, 0x1e, 0xf3, 0x2e);

DEFINE\_GUID(ImageFormatGIF, 0xb96b3cb0, 0x0728, 0x11d3, 0x9d, 0x7b, 0x00, 0x00, 0xf8, 0x1e, 0xf3, 0x2e);

DEFINE\_GUID(ImageFormatTIFF, 0xb96b3cb1, 0x0728, 0x11d3, 0x9d, 0x7b, 0x00, 0x00, 0xf8, 0x1e, 0xf3, 0x2e);

// Диалоговое окно CAboutDlg используется для описания сведений о приложении

double arg; // ввод аргумента в IDC EDIT 1 - m\_Calc переменная привязанная к m\_Calc

class CAboutDlg : public CDialogEx

{

public:

CAboutDlg();

// Данные диалогового окна

#ifdef AFX\_DESIGN\_TIME

enum { IDD = IDD\_ABOUTBOX };

#endif

protected:

virtual void DoDataExchange(CDataExchange\* pDX); // поддержка DDX/DDV

// Реализация

protected:

DECLARE\_MESSAGE\_MAP()

public:

afx\_msg void OnIdrMenuDop();

};

CAboutDlg::CAboutDlg() : CDialogEx(IDD\_ABOUTBOX)

{

}

void CAboutDlg::DoDataExchange(CDataExchange\* pDX)

{

CDialogEx::DoDataExchange(pDX);

}

//Подключаем обработчики схемы сообщений

BEGIN\_MESSAGE\_MAP(CAboutDlg, CDialogEx)

ON\_COMMAND(IDR\_MENU\_DOP, &CAboutDlg::OnIdrMenuDop)

ON\_WM\_RBUTTONUP()

END\_MESSAGE\_MAP()

// диалоговое окно Cproj\_letiDlg

Cproj\_letiDlg::Cproj\_letiDlg(CWnd\* pParent /\*=NULL\*/)

: CDialogEx(IDD\_PROJ\_LETI\_DIALOG, pParent)

, m\_paintPic()

, m\_strF(\_T(""))

, m\_HelpVal(\_T(""))

, m\_MinMaxValCheck(0)

,m\_paintDPF()

, m\_MinMaxStepCheck(0)

, m\_MinMaxCheckPoints(0)

{

m\_hIcon = AfxGetApp()->LoadIcon(IDR\_MAINFRAME);

}

void Cproj\_letiDlg::DoDataExchange(CDataExchange\* pDX)

{

CDialogEx::DoDataExchange(pDX);

DDX\_Control(pDX, IDC\_EDIT1, m\_Calc);

DDX\_Text(pDX, IDC\_ANIMATE1, m\_strF);

DDX\_Control(pDX, IDC\_ST\_SH, m\_st\_SH);

DDX\_Text(pDX, IDC\_HELP, m\_HelpVal);

DDX\_Control(pDX, IDC\_ANIMATE2, m\_paintDPF);

DDX\_Text(pDX, IDC\_EDIT1, m\_MinMaxValCheck);

DDV\_MinMaxFloat(pDX, m\_MinMaxValCheck, 70, 100);

DDX\_Text(pDX, IDC\_EDIT4, m\_MinMaxStepCheck);

DDV\_MinMaxFloat(pDX, m\_MinMaxStepCheck, 1, 4);

DDX\_Text(pDX, IDC\_EDIT\_CT, m\_MinMaxCheckPoints);

DDV\_MinMaxFloat(pDX, m\_MinMaxCheckPoints, 100, 1000);

}

BEGIN\_MESSAGE\_MAP(Cproj\_letiDlg, CDialogEx)

//Обработчики схемы сообщений

ON\_WM\_SYSCOMMAND()

ON\_WM\_PAINT()

ON\_WM\_QUERYDRAGICON()

ON\_STN\_CLICKED(IDC\_EDIT1, &Cproj\_letiDlg::OnEnChangeEdit1)

ON\_WM\_TIMER()

ON\_BN\_CLICKED(IDC\_CHECK3, &Cproj\_letiDlg::OnBnClickedCheck3)

ON\_WM\_TIMER()

ON\_BN\_CLICKED(IDC\_SAVE\_PIC, &Cproj\_letiDlg::OnBnClickedSavePic)

ON\_WM\_RBUTTONUP()

ON\_WM\_MOUSEMOVE()

ON\_BN\_CLICKED(IDOK, &Cproj\_letiDlg::OnBnClickedOk)

ON\_COMMAND(ID\_32773, &Cproj\_letiDlg::On32773)

ON\_COMMAND(ID\_32772, &Cproj\_letiDlg::On32772)

ON\_BN\_CLICKED(IDC\_SAVE\_PICDPF, &Cproj\_letiDlg::OnBnClickedSavePicdpf)

ON\_WM\_MOUSEWHEEL()

ON\_EN\_CHANGE(IDC\_EDIT\_CT, &Cproj\_letiDlg::OnEnChangeEditCt)

ON\_EN\_CHANGE(IDC\_EDIT4, &Cproj\_letiDlg::OnEnChangeEdit4)

ON\_BN\_CLICKED(IDC\_BUTTON\_LOG1, &Cproj\_letiDlg::OnBnClickedButtonLog1)

ON\_STN\_CLICKED(IDC\_ST\_SH\_DPF, &Cproj\_letiDlg::OnStnClickedStShDpf)

ON\_STN\_CLICKED(IDC\_STATIC\_LGSCALE, &Cproj\_letiDlg::OnStnClickedStaticLgscale)

ON\_STN\_CLICKED(IDC\_STATIC\_STEP, &Cproj\_letiDlg::OnStnClickedStaticStep)

END\_MESSAGE\_MAP()

// обработчики сообщений Cproj\_letiDlg

BOOL Cproj\_letiDlg::OnInitDialog()

{

CDialogEx::OnInitDialog();

// Добавление пункта "О программе..." в системное меню.

// IDM\_ABOUTBOX должен быть в пределах системной команды.

ASSERT((IDM\_ABOUTBOX & 0xFFF0) == IDM\_ABOUTBOX);

ASSERT(IDM\_ABOUTBOX < 0xF000);

CMenu\* pSysMenu = GetSystemMenu(FALSE);

if (pSysMenu != NULL)

{

BOOL bNameValid;

CString strAboutMenu;

bNameValid = strAboutMenu.LoadString(IDS\_ABOUTBOX);

ASSERT(bNameValid);

if (!strAboutMenu.IsEmpty())

{

pSysMenu->AppendMenu(MF\_SEPARATOR);

pSysMenu->AppendMenu(MF\_STRING, IDM\_ABOUTBOX, strAboutMenu);

}

}

// Задает значок для этого диалогового окна. Среда делает это автоматически,

// если главное окно приложения не является диалоговым

SetIcon(m\_hIcon, TRUE); // Крупный значок

SetIcon(m\_hIcon, FALSE); // Мелкий значок

// TODO: добавьте дополнительную инициализацию

SetWindowText(L"RZ");

m\_paintPic.SubclassDlgItem(IDC\_ANIMATE1, this);//получаем указатель на IDC\_ANIMATE1

m\_Pt.m\_Param.Scale = AfxGetApp()->GetProfileInt(\_T("Settings"),\_T("Arg"),45);//Устанавливаем в реестр значение Scale

CString strA;

m\_Pt.m\_Param.nPoints = AfxGetApp()->GetProfileInt(\_T("Settings"), \_T("Points"), 300);//Устанавливаем в реестр значение Scale

strA.Format(\_T("%.0f"), m\_Pt.m\_Param.Scale);//В strA кладем Scale приведенную к строке

CWnd\* pE = GetDlgItem(IDC\_EDIT1);//Получаем указатель CWnd на IDC\_EDIT1

if (pE) {

pE->SetWindowText(strA);//В окно ввода аргумента подставляем значение Scale

}

return TRUE;

}

void Cproj\_letiDlg::OnSysCommand(UINT nID, LPARAM lParam)

{

if ((nID & 0xFFF0) == IDM\_ABOUTBOX)

{

CAboutDlg dlgAbout;

dlgAbout.DoModal();

}

else

{

CDialogEx::OnSysCommand(nID, lParam);

}

}

// При добавлении кнопки свертывания в диалоговое окно нужно воспользоваться приведенным ниже кодом,

// чтобы нарисовать значок. Для приложений MFC, использующих модель документов или представлений,

// это автоматически выполняется рабочей областью.

//Обработчик рисования графика

void Cproj\_letiDlg::OnPaint()

{

CRect rc;

m\_paintDPF.GetClientRect(&rc);

m\_DPF.m\_Param.Height = rc.Height();

m\_DPF.m\_Param.Width = rc.Width();

if (IsIconic())

{

CWnd \*pWn = GetDlgItem(IDC\_ANIMATE1);

CPaintDC dc(pWn); // контекст устройства для рисования

SendMessage(WM\_ICONERASEBKGND, reinterpret\_cast<WPARAM>(dc.GetSafeHdc()), 0);

// Выравнивание значка по центру клиентского прямоугольника

int cxIcon = GetSystemMetrics(SM\_CXICON);

int cyIcon = GetSystemMetrics(SM\_CYICON);

CRect rect;

GetClientRect(&rect);

int x = (rect.Width() - cxIcon + 1) / 2;

int y = (rect.Height() - cyIcon + 1) / 2;

// Нарисуйте значок

dc.DrawIcon(x, y, m\_hIcon);

}

else

{

CDialogEx::OnPaint();

}

}

// Система вызывает эту функцию для получения отображения курсора при перемещении

// свернутого окна.

HCURSOR Cproj\_letiDlg::OnQueryDragIcon()

{

return static\_cast<HCURSOR>(m\_hIcon);

}

CString sA;

int ArgChangeCopy;

void Cproj\_letiDlg::OnEnChangeEdit1(){//Обработчик для функции ввода аргумента

CWnd\* pE = GetDlgItem(IDC\_EDIT1);//Получаем указатель CWnd на окно первого графика

if (pE)

{

pE->GetWindowText(sA);//Иницилизуруем sA введенным значением

//Устанавливаем параметры MyParams\_t

MyParams\_t par = m\_Pt.m\_Param;

par.Scale = \_wtof(sA);

m\_Pt.SetParams(par);

//Обновляем графики сигнальной функции и ДПФ

m\_paintPic.Invalidate();

m\_paintDPF.Invalidate();

}

}

void Cproj\_letiDlg::OnTimer(UINT\_PTR nIDEvent)//Обработчик сиситемного таймера

{

if (nIDEvent == MY\_TIMER1) {

OnEnChangeEdit1();

//Увеличиваем кол-во рисуемых точек на 1 и обновляем график

m\_paintPic.m\_Npt++;

m\_paintPic.Invalidate();

}

if (nIDEvent == MY\_TIMER2) {

OnEnChangeEdit1();

//Увеличиваем кол-во рисуемых точек на 1 и обновляем график

m\_paintDPF.m\_Npt++;

m\_paintDPF.Invalidate();

}

CDialogEx::OnTimer(nIDEvent);

}

void Cproj\_letiDlg::OnBnClickedCheck3()//Обработчик нажатия Check\_control для IDC\_CHECK3

{

CButton \*pW = (CButton \*)GetDlgItem(IDC\_CHECK3);

if (pW) {

if (pW->GetCheck()) {//Если клавиша нажата то устанавилваем таймеры

m\_Calc.EnableWindow(FALSE);

SetTimer(MY\_TIMER1, 1, NULL);

SetTimer(MY\_TIMER2, 1, NULL);

}

else {//Иначе останавливаем отрисовку графиков

m\_Calc.EnableWindow(TRUE);

KillTimer(MY\_TIMER1);

KillTimer(MY\_TIMER2);

}

}

}

void CAboutDlg::OnIdrMenuDop()//функция обработчика нового меню::OnIdrMenuDop()

{

//Создаем меню и загружаем его под ID IDR\_MENU\_DOP

CMenu mn\_up;

mn\_up.LoadMenuW(IDR\_MENU\_DOP);

}

void Cproj\_letiDlg::OnRButtonUp(UINT nFlags, CPoint point)

{

CMenu mn;

mn.LoadMenuW(IDR\_MENU\_DOP); // загружаем из ресурсов меню

CPoint pt = point; //берем точку нажатия

ClientToScreen(&pt); // переводим в координаты экрана из коорд. диалога

if (mn.GetSubMenu(0)) {// берем меню первое

mn.GetSubMenu(0)->TrackPopupMenu(TPM\_LEFTALIGN | TPM\_TOPALIGN | TPM\_LEFTBUTTON, pt.x, pt.y, this);// вызываем на экран меню и ждем реакцию пользователя

}

CDialogEx::OnRButtonUp(nFlags, point);

}

void Cproj\_letiDlg::OnMouseMove(UINT nFlags, CPoint point)//функция обработчика WM\_MOUSE\_MOVE

{

CPoint pt;

GetCursorPos(&pt);

m\_paintPic.ScreenToClient(&pt);

CRect rc;

CStatic \*pW = (CStatic \*)GetDlgItem(IDC\_ST\_SH);//IDC\_ST\_SH-id статика в котором выводим значение "x"

ASSERT(pW);

m\_paintPic.GetClientRect(&rc);

if (rc.PtInRect(pt))

{

m\_strF.Format(\_T("X = %d Y=%d"), pt.x,pt.y ? -(pt.y-190):pt.y-190);//m\_strF-переменная CString для формата координаты "x"

pW->SetWindowText(m\_strF);

}

else {

pW->SetWindowText(\_T("X=... Y=..."));

}

//////////

const int HeightY = 344;

GetCursorPos(&pt);

m\_paintDPF.ScreenToClient(&pt);

CStatic \*pWDPF = (CStatic \*)GetDlgItem(IDC\_ST\_SH\_DPF);//IDC\_ST\_SH-id статика в котором выводим значение "x"

ASSERT(pWDPF);

m\_paintDPF.GetClientRect(&rc);

if (rc.PtInRect(pt))

{

m\_strF.Format(\_T("X = %d Y=%d"), pt.x, HeightY-pt.y);//m\_strF-переменная CString для формата координаты "x"

pWDPF->SetWindowText(m\_strF);

}

else {

pWDPF->SetWindowText(\_T("X=... Y=..."));

}

CDialogEx::OnMouseMove(nFlags, point);

}

void Cproj\_letiDlg::OnBnClickedOk()//Кнопка ОК

{

//Сохраняем в реестр значение Scale

AfxGetApp()->WriteProfileInt(\_T("Settings"), \_T("Arg"),m\_Pt.m\_Param.Scale);

AfxGetApp()->WriteProfileInt(\_T("Settings"), \_T("Points"), m\_Pt.m\_Param.nPoints);

CDialogEx::OnOK();

}

void Cproj\_letiDlg::On32773()//Пункт Выход в доп.меню

{

ASSERT(AfxGetApp()->m\_pMainWnd != NULL);

AfxGetApp()->m\_pMainWnd->PostMessage(WM\_CLOSE, 0, 0);

}

void Cproj\_letiDlg::On32772()//Пункт Помощь в доп.меню

{

CStatic \*pW = (CStatic \*)GetDlgItem(IDC\_HELP);//Получаю указатель на Статик окно в котором выведу текст

ASSERT(pW);

pW->SetWindowTextW(L"1)Чтобы построить графики сигнальной функции и ДПФ введите аргумент значением от 70 до 100 и нажмите ПОСТРОИТЬ ГРАФИКИ на главной странице\n2)Чтобы узнать координату X на графике,просто наведите курсор на интересующую точку на графике");

}

void Cproj\_letiDlg::OnBnClickedSavePicdpf()//Обработчик функции сохранения изображения графика ДПФ

{

CWnd \*pWn = GetDlgItem(IDC\_ANIMATE2);

if (!pWn)

return;

CWindowDC winDC(pWn);

CRect rc;

pWn->GetClientRect(&rc);

CDC memDC;

memDC.CreateCompatibleDC(&winDC);

CBitmap bitMap;

bitMap.CreateCompatibleBitmap(&winDC, rc.Width(), rc.Height());

HGDIOBJ pOld = memDC.SelectObject(&bitMap);

memDC.FillSolidRect(&rc, RGB(100, 243, 255));

memDC.BitBlt(0, 0, rc.Width(), rc.Height(), &winDC, 0, 0, SRCCOPY);

memDC.SelectObject(pOld);

static TCHAR szFilter[] = \_T("BMP Files (\*.bmp)|\*.bmp|")

\_T("PNG Files (\*.png)|\*.png|GIF Files (\*.gif)|\*.gif|")

\_T("JPG Files (\*.jpg)|\*.jpg|All Files (\*.\*)|\*.\*||");

CFileDialog dlg(FALSE, \_T(".bmp"), NULL, 6UL, szFilter);

if (IDOK == dlg.DoModal())

{

CImage image;

image.Attach(HBITMAP(bitMap));

CString strFull = dlg.GetOFN().lpstrFile;

HRESULT hr;

if (-1 != strFull.Find(\_T(".png")))

hr = image.Save(strFull, ImageFormatPNG);

else if (-1 != strFull.Find(\_T(".jpg")))

hr = image.Save(strFull, ImageFormatJPEG);

else if (-1 != strFull.Find(\_T(".gif")))

hr = image.Save(strFull, ImageFormatGIF);

else if (-1 != strFull.Find(\_T(".bmp")))

hr = image.Save(strFull, ImageFormatBMP);

else

{

strFull += \_T(".bmp");

hr = image.Save(strFull, ImageFormatBMP);

}

if (FAILED(hr))

{

CString strErr;

strErr.Format(L" Couldn't Save File: %s, %x ", (LPCTSTR)strFull, hr);

AfxMessageBox(strErr, MB\_OK | MB\_ICONERROR);

}

}

}

void Cproj\_letiDlg::OnBnClickedSavePic()//Обработчик функции сохранения изображения графика сигнальной функции

{

CWnd \*pWn = GetDlgItem(IDC\_ANIMATE1);

if (!pWn)

return;

CWindowDC winDC(pWn);

CRect rc;

pWn->GetClientRect(&rc);

CDC memDC;

memDC.CreateCompatibleDC(&winDC);

CBitmap bitMap;

bitMap.CreateCompatibleBitmap(&winDC, rc.Width(), rc.Height());

HGDIOBJ pOld = memDC.SelectObject(&bitMap);

memDC.FillSolidRect(&rc, RGB(100, 243, 255));

memDC.BitBlt(0, 0, rc.Width(), rc.Height(), &winDC, 0, 0, SRCCOPY);

memDC.SelectObject(pOld);

static TCHAR szFilter[] = \_T("BMP Files (\*.bmp)|\*.bmp|")

\_T("PNG Files (\*.png)|\*.png|GIF Files (\*.gif)|\*.gif|")

\_T("JPG Files (\*.jpg)|\*.jpg|All Files (\*.\*)|\*.\*||");

CFileDialog dlg(FALSE, \_T(".bmp"), NULL, 6UL, szFilter);

if (IDOK == dlg.DoModal())

{

CImage image;

image.Attach(HBITMAP(bitMap));

CString strFull = dlg.GetOFN().lpstrFile;

HRESULT hr;

if (-1 != strFull.Find(\_T(".png")))

hr = image.Save(strFull, ImageFormatPNG);

else if (-1 != strFull.Find(\_T(".jpg")))

hr = image.Save(strFull, ImageFormatJPEG);

else if (-1 != strFull.Find(\_T(".gif")))

hr = image.Save(strFull, ImageFormatGIF);

else if (-1 != strFull.Find(\_T(".bmp")))

hr = image.Save(strFull, ImageFormatBMP);

else

{

strFull += \_T(".bmp");

hr = image.Save(strFull, ImageFormatBMP);

}

if (FAILED(hr))

{

CString strErr;

strErr.Format(L" Couldn't Save File: %s, %x ", (LPCTSTR)strFull, hr);

AfxMessageBox(strErr, MB\_OK | MB\_ICONERROR);

}

}

}

BOOL Cproj\_letiDlg::OnMouseWheel(UINT nFlags, short zDelta, CPoint pt)//Обработчик увеличения графика

{

GetCursorPos(&pt);//Получаем точку в которйо находится курсор

//Создаем прямоугольное окно и получаем его как фокус с данной точкой pt

CRect rc;

m\_paintPic.ScreenToClient(&pt);

m\_paintPic.GetClientRect(&rc);

if (rc.PtInRect(pt)) { //проверяем попадает ли точка в окно графика

//Изменяем параметры MyParams\_t

MyParams\_t par = m\_Pt.m\_Param;

par.Scale = par.Scale + zDelta / 50.;

m\_Pt.SetParams(par);//Устанавливаем новые параметры

m\_paintPic.Invalidate();//Обновляем график

}

return CDialogEx::OnMouseWheel(nFlags, zDelta, pt);

}

void Cproj\_letiDlg::OnEnChangeEditCt()

{

CString sCt;

CWnd\* pE = GetDlgItem(IDC\_EDIT\_CT);

if (pE)

{

pE->GetWindowText(sCt);//Иницилизуруем sA введенным значением

//Устанавливаем параметры MyParams\_t

MyParams\_t par = m\_Pt.m\_Param;

par.nPoints = \_wtof(sCt);

m\_Pt.SetParams(par);

MyParams\_DPF par1 = m\_DPF.m\_Param;

par1.nPoints = \_wtof(sCt);

m\_DPF.SetParams(par1);

//Обновляем графики сигнальной функции и ДПФ

m\_paintPic.Invalidate();

m\_paintDPF.Invalidate();

}

}

void Cproj\_letiDlg::OnEnChangeEdit4()

{

CString sL;

CWnd\* pE = GetDlgItem(IDC\_EDIT4);

if (pE)

{

CStatic \*pW = (CStatic \*)GetDlgItem(IDC\_STATIC\_STEP);

pE->GetWindowText(sL);//Иницилизуруем sA введенным значением

//Устанавливаем параметры MyParams\_t

int checkL;

checkL= \_wtof(sL);

switch (checkL) {

case 1:

m\_Pt.m\_Param.Fm = 0.01\*m\_Pt.m\_Param.Fd;

m\_Pt.m\_Param.f\_Fd = 0.1\*m\_Pt.m\_Param.Fd;

m\_Pt.m\_Param.m = m\_Pt.m\_Param.f\_Fd/1;

pW->SetWindowText(\_T("Параметры шага:\nFд=10МГц,f=0.1\*Fд,\nFm=0.01\*Fд,m=f/1"));

break;

case 2:

m\_Pt.m\_Param.Fm = 0.02\*m\_Pt.m\_Param.Fd;

m\_Pt.m\_Param.f\_Fd = 0.2\*m\_Pt.m\_Param.Fd;

m\_Pt.m\_Param.m = m\_Pt.m\_Param.f\_Fd / 3;

pW->SetWindowText(\_T("Параметры шага:\nFд=10МГц,f=0.2\*Fд,\nFm=0.02\*Fд,m=f/3"));

break;

case 3:

m\_Pt.m\_Param.Fm = 0.05\*m\_Pt.m\_Param.Fd;

m\_Pt.m\_Param.f\_Fd = 0.3\*m\_Pt.m\_Param.Fd;

m\_Pt.m\_Param.m = m\_Pt.m\_Param.f\_Fd / 5;

pW->SetWindowText(\_T("Параметры шага:\nFд=10МГц,f=0.3\*Fд,\nFm=0.05\*Fд,m=f/5"));

break;

case 4:

m\_Pt.m\_Param.Fm = 0.07\*m\_Pt.m\_Param.Fd;

m\_Pt.m\_Param.f\_Fd = 0.4\*m\_Pt.m\_Param.Fd;

m\_Pt.m\_Param.m = m\_Pt.m\_Param.f\_Fd / 7;

pW->SetWindowText(\_T("Параметры шага:\nFд=10МГц,f=0.4\*Fд,\nFm=0.07\*Fд,m=f/7"));

break;

default:

pW->SetWindowText(\_T("Неккоректный ввод"));

break;

}

//Обновляем графики сигнальной функции и ДПФ

m\_paintPic.Invalidate();

m\_paintDPF.Invalidate();

}

}

void Cproj\_letiDlg::OnBnClickedButtonLog1()

{

CButton \*pW = (CButton \*)GetDlgItem(IDC\_BUTTON\_LOG1);

if (pW) {

if (pW->GetCheck()) {

check\_log = true;

OnStnClickedStaticLgscale();

pW->SetWindowText(\_T("Логарифмический масштаб"));

MyParams\_t par = m\_Pt.m\_Param;

par.Scale = std::log10(m\_Pt.m\_Param.Scale)\*10;

m\_Pt.SetParams(par);

m\_paintPic.Invalidate();

m\_paintDPF.Invalidate();

}

else {

check\_log = false;

OnStnClickedStaticLgscale();

pW->SetWindowText(\_T("Стандартный масштаб"));

MyParams\_t par = m\_Pt.m\_Param;

par.Scale = 100;

m\_Pt.SetParams(par);

m\_paintPic.Invalidate();

m\_paintDPF.Invalidate();

}

}

}

void Cproj\_letiDlg::OnStnClickedStShDpf()

{

// TODO: добавьте свой код обработчика уведомлений

}

void Cproj\_letiDlg::OnStnClickedStaticLgscale()

{

CStatic \*pW = (CStatic \*)GetDlgItem(IDC\_STATIC\_LGSCALE);

if (check\_log) {

pW->SetWindowText(\_T("log10(Scale) \* 10"));

}

else {

pW->SetWindowText(\_T(""));

}

}

void Cproj\_letiDlg::OnStnClickedStaticStep()

{

}

#### 

## Proj\_leti.cpp

#include "stdafx.h"

#include "proj\_leti.h"

#include "proj\_letiDlg.h"

#define \_USE\_MATH\_DEFINES

#ifdef \_DEBUG

#define new DEBUG\_NEW

#endif

// Cproj\_letiApp

BEGIN\_MESSAGE\_MAP(Cproj\_letiApp, CWinApp)

ON\_COMMAND(ID\_HELP, &CWinApp::OnHelp)

END\_MESSAGE\_MAP()

// создание Cproj\_letiApp

Cproj\_letiApp::Cproj\_letiApp()

{

// поддержка диспетчера перезагрузки

m\_dwRestartManagerSupportFlags = AFX\_RESTART\_MANAGER\_SUPPORT\_RESTART;

// TODO: добавьте код создания,

// Размещает весь важный код инициализации в InitInstance

}

// Единственный объект Cproj\_letiApp

Cproj\_letiApp theApp;

// инициализация Cproj\_letiApp

BOOL Cproj\_letiApp::InitInstance()

{

// InitCommonControlsEx() требуется для Windows XP, если манифест

// приложения использует ComCtl32.dll версии 6 или более поздней версии для включения

// стилей отображения. В противном случае будет возникать сбой при создании любого окна.

INITCOMMONCONTROLSEX InitCtrls;

InitCtrls.dwSize = sizeof(InitCtrls);

// Выберите этот параметр для включения всех общих классов управления, которые необходимо использовать

// в вашем приложении.

InitCtrls.dwICC = ICC\_WIN95\_CLASSES;

InitCommonControlsEx(&InitCtrls);

CWinApp::InitInstance();

AfxEnableControlContainer();

// Создать диспетчер оболочки, в случае, если диалоговое окно содержит

// представление дерева оболочки или какие-либо его элементы управления.

CShellManager \*pShellManager = new CShellManager;

// Активация визуального диспетчера "Классический Windows" для включения элементов управления MFC

CMFCVisualManager::SetDefaultManager(RUNTIME\_CLASS(CMFCVisualManagerWindows));

// Стандартная инициализация

// Если эти возможности не используются и необходимо уменьшить размер

// конечного исполняемого файла, необходимо удалить из следующих

// конкретных процедур инициализации, которые не требуются

// Измените раздел реестра, в котором хранятся параметры

// TODO: следует изменить эту строку на что-нибудь подходящее,

// например на название организации

SetRegistryKey(\_T("RZ"));

Cproj\_letiDlg dlg;

m\_pMainWnd = &dlg;

INT\_PTR nResponse = dlg.DoModal();

if (nResponse == IDOK)

{

// TODO: Введите код для обработки закрытия диалогового окна

// с помощью кнопки "ОК"

}

else if (nResponse == IDCANCEL)

{

// TODO: Введите код для обработки закрытия диалогового окна

// с помощью кнопки "Отмена"

}

else if (nResponse == -1)

{

TRACE(traceAppMsg, 0, "Предупреждение. Не удалось создать диалоговое окно, поэтому работа приложения неожиданно завершена.\n");

TRACE(traceAppMsg, 0, "Предупреждение. При использовании элементов управления MFC для диалогового окна невозможно #define \_AFX\_NO\_MFC\_CONTROLS\_IN\_DIALOGS.\n");

}

// Удалить диспетчер оболочки, созданный выше.

if (pShellManager != NULL)

{

delete pShellManager;

}

#ifndef \_AFXDLL

ControlBarCleanUp();

#endif

// Поскольку диалоговое окно закрыто, возвратите значение FALSE, чтобы можно было выйти из

// приложения вместо запуска генератора сообщений приложения.

return FALSE;

}

## PaintPic.cpp

//Подключаем заголовочные файлы

#include "stdafx.h"

#include "proj\_letiDlg.h"

#include "proj\_leti.h"

#include "PaintPic.h"

#include "Cproj\_letiCalc.h"

#include <string>

#define \_USE\_MATH\_DEFINES

#ifdef \_DEBUG

#define new DEBUG\_NEW

#endif

IMPLEMENT\_DYNAMIC(CPaintPic, CWnd)

CPaintPic::CPaintPic()//конструктор CPaintPic()

{

//Инициализация m\_BackCol\_1,m\_Npt,m\_GraphCol\_1

m\_BackCol\_1 = RGB(255, 255, 255);

m\_GraphCol\_1 = RGB(0, 0, 255);

m\_Npt = 0;

}

CPaintPic::~CPaintPic()//Деструктор CpaintPic()

{

}

// обработчики сообщений CPaintPic

BEGIN\_MESSAGE\_MAP(CPaintPic, CWnd)

ON\_WM\_PAINT()

ON\_WM\_MOUSEMOVE()

ON\_WM\_MOUSEWHEEL()

ON\_WM\_MOUSEHWHEEL()

END\_MESSAGE\_MAP()

void CPaintPic::OnPaint()

{

CPaintDC dc(this); //объявляем объект рисования с указателем this на главное окно

CRect rc;//создаем прямоугольную область(холст)

GetClientRect(&rc);//получаем данный холст

CRgn rgn;//создаем клип область

rgn.CreateRectRgn(rc.left, rc.top, rc.right, rc.bottom);//обозначаем границы для rgn

dc.SelectClipRgn(&rgn);//выбираем данный клип для dc

dc.FillSolidRect(rc, m\_BackCol\_1);//делаем заливку фона RGB из переменной m\_backCol\_1

CPoint pt;

//создаем объект карандаша и определяем параметры

CPen pAxes(PS\_SOLID, 2, RGB(0,0,0));

HGDIOBJ old\_Axes = dc.SelectObject(pAxes);

Cproj\_letiDlg \*pDlg = (Cproj\_letiDlg \*)AfxGetMainWnd();

if (!pDlg) {

return;

}

//Рисование осей//

dc.MoveTo(-1,182);

dc.LineTo(rc.right, 182);

dc.SelectObject(old\_Axes);

/////////////////Переопредление карандаша под график

CPen pV(PS\_SOLID, 1.5, m\_GraphCol\_1);

HGDIOBJ old = dc.SelectObject(pV);

/////////////////

std::vector<POINT> \*vec = pDlg->m\_Pt.GetPoints\_dPoint();//В vec кладем по указателю функцией Get\_Points\_d\_Point рассчитанные значения точек x,y

if (vec->size())//Проверяем не пустой ли вектор точек

{

std::vector<POINT>::iterator it = vec->begin();//создаем итератор-указатель на каждую пару точек x,y из vec

dc.MoveTo(\*it);//устанавливаем кисть на координаты на первую точку

size\_t nCur = 0;//иницилизируем переменную для проверки кол-ва точек

for (it++; it != vec->end(); it++)//циклом перебираем точки vec используя итератор it(обращаясь по указателю)

{

dc.LineTo(\*it);//Рисуем отрезок от предыдущей точки до точки из vec используя итератор

if (nCur++ >= m\_Npt) {//проверяем выход из вектора(проверка количества точек)

break;//если условие истинно то выходим из цикла

}

}

}

pDlg->m\_DPF.Calculate();//Вызываем функцию Calcualate() из класса proj\_letiCalcDPF

pDlg->m\_paintDPF.Invalidate();//Обновление окна IDC\_ANIMATE2

dc.SelectObject(old);

}

## PaintPic2.cpp

// PaintPic2.cpp: файл реализации

//

//Подключаем заголовочные файлы

#include "stdafx.h"

#include "proj\_leti.h"

#include "PaintPic2.h"

#include "stdafx.h"

#include "proj\_letiDlg.h"

#include "Cproj\_letiCalc.h"

#include <string>

#define \_USE\_MATH\_DEFINES

#ifdef \_DEBUG

#define new DEBUG\_NEW

#endif

// CPaintPic2

IMPLEMENT\_DYNAMIC(CPaintPic2, CWnd)

CPaintPic2::CPaintPic2()//конструктор CPaintPic()

{

//Инициализация m\_BackCol\_1,m\_Npt,m\_GraphCol\_1

m\_BackCol\_1 = RGB(255, 255, 255);

m\_GraphCol\_1 = RGB(0, 0, 255);

m\_Npt = 0;

}

CPaintPic2::~CPaintPic2()//Деструктор CpaintPic2()

{

}

// обработчики сообщений CPaintPic

BEGIN\_MESSAGE\_MAP(CPaintPic2, CWnd)

ON\_WM\_PAINT()

ON\_WM\_MOUSEMOVE()

END\_MESSAGE\_MAP()

void CPaintPic2::OnPaint()

{

CPaintDC dc(this);//объявляем объект рисования с указателем this на главное окно

CRect rc;//создаем прямоугольную область(холст)

GetClientRect(&rc);

CRgn rgn;//создаем клип область

rgn.CreateRectRgn(rc.left, rc.top, rc.right, rc.bottom);//обозначаем границы для rgn

dc.SelectClipRgn(&rgn);//выбираем данный клип для dc

dc.FillSolidRect(rc, m\_BackCol\_1);//делаем заливку фона RGB из переменной m\_backCol\_1

CPoint pt;

CPen pAxes(PS\_SOLID, 2, RGB(0, 0, 0));//создаем объект карандаша и определяем параметры и выбираем его

HGDIOBJ old\_Axes = dc.SelectObject(pAxes);

Cproj\_letiDlg \*pDlg= (Cproj\_letiDlg \*)AfxGetMainWnd();

if (!pDlg) {

return;

}

//Рисование осей//

dc.MoveTo(-1, 182);

dc.LineTo(rc.right, 182);

dc.SelectObject(old\_Axes);

/////////////////Переопредление карандаша под график

CPen pV(PS\_SOLID, 1.5, m\_GraphCol\_1);

HGDIOBJ old = dc.SelectObject(pV);

/////////////////

std::vector<POINT> \*vec = pDlg->m\_DPF.GetPoints\_DPF();

if (vec->size())//Проверка на наличие точек в векторе

{

std::vector<POINT>::iterator it = vec->begin();

dc.MoveTo(\*it);

size\_t nCur = 0;

const int HeightY = 344;//Иницилизируем высоту окна графика

for (it++; it != vec->end();it++)//циклом проходим по вектору vec

{

dc.MoveTo(it->x,HeightY);

dc.LineTo(\*it);

if (nCur++ >= m\_Npt)

break;

}

if (m\_Npt >= vec->size())

AfxGetMainWnd()->SendMessage(WM\_USER + 1, m\_Npt, vec->size());

}

dc.SelectObject(old);

}

## Cproj\_letiCalc.cpp

//Подключаем заголовочные файлы

#include "stdafx.h"

#include "proj\_leti.h"

#include "Cproj\_letiCalc.h"

#include "proj\_letiDlg.h"

#include "cmath"

#define \_USE\_MATH\_DEFINES

#include "math.h"

dPoint::dPoint()//Класс dPoint с заданными параметрами

{

m\_Param.Scale = 100;

m\_Param.xMax = 4\*M\_PI;

m\_Param.nPoints = 240;

m\_Param.yCenter = 182;

m\_Param.dXRad = 0;

}

bool dPoint::Calculate\_dPoint() {

m\_vecPt.clear();//Очищаем вектор перед заполнением

double DeltaArg =m\_Param.xMax / m\_Param.nPoints;

for (size\_t ni = 0; ni < m\_Param.nPoints; ni++)//Проходим циклом вызывая функции nPoints раз,заполняя массив

{

double Arg = ni \* DeltaArg;

CPoint pt;

if (Scale(pt, m\_Param.Scale, m\_Param.Scale, Arg, m\_Param.dXRad, m\_Param.f\_Fd, m\_Param.Fm, m\_Param.m, m\_Param.amplitude))//вызываем функцию масштабирования

{

if (Shift(pt))

m\_vecPt.push\_back(POINT(pt));//Добавляем в конец вектора m\_vecPt точку pt

}

}

return m\_vecPt.size() > 0;

}

std::vector<POINT> \*dPoint::GetPoints\_dPoint()//функция возвращающая вектор с рассчитанными точками

{

return &m\_vecPt;

}

bool dPoint::Scale(CPoint & pt, double ScaleX, double ScaleY, double Arg, double Param, double f\_Fd, double Fm, double m, double amplitude)//функция масштабирования

{

CMyFun fn;

if (fn.Calc(Arg, Param, f\_Fd, Fm, m, amplitude))

{

pt.x = (int)floor(ScaleX \* fn.X);//масштабируем точки относительно окна и приводим их к int

pt.y = -(int)floor(ScaleY \* fn.Y);

return true;

}

return false;

}

bool dPoint::Shift(CPoint & pt)

{

pt.Offset(m\_Param.xCenter, m\_Param.yCenter);

return true;

}

bool dPoint::SetParams(MyParams\_t &par)

{

//Устанавливаем параметры для m\_Param и вызываем функцию для расчета точек

m\_Param = par;

Calculate\_dPoint();

return true;

}

bool CMyFun::Calc(double f, double k, double f\_Fd, double Fm, double m, double amplitude)

{

//Расчет точек сигнальной дял графика сигнальной функции

X = f;

//a\*sin(2\*PI\*(f +m\*sin(2\*PI\*Fm\*t)\*t)-график сигнальной функции

Y = amplitude \* sin(2 \* M\_PI\*(f\_Fd + m \* sin(2 \* M\_PI\*Fm \* f)\*f));

return true;

## }Cproj\_letiCalcDPF.cpp

/Подключаем заголовочные файлы

#include "stdafx.h"

#include "Cproj\_letiCalcDPF.h"

#define \_USE\_MATH\_DEFINES

#include "math.h"

#include "proj\_letiDlg.h"

//Конструктор

Cproj\_letiCalcDPF::Cproj\_letiCalcDPF()

{

}

//Деструктор

Cproj\_letiCalcDPF::~Cproj\_letiCalcDPF()

{

}

bool Cproj\_letiCalcDPF::Calc(std::vector<POINT> \*vec, size\_t k, double yShift)//Функция вычислений точек

{

double A = 0.;

double B = 0.;

size\_t N = vec->size();

X = 2.\*k / double(N - 1);

int Xn = 0;

for (size\_t n = 0; n < N; n++) {

Xn = (vec->begin() + n)->y - yShift;

A += Xn\*cos(2 \* M\_PI\*n\*k / N);

B += Xn\*sin(2 \* M\_PI\*n\*k / N);

}

Y = sqrt(A\*A + B\*B) / N;

return true;

}

bool Cproj\_letiCalcDPF::Shift(CPoint &pt)

{

pt.Offset(0, m\_Param.Height);

return true;

}

bool Cproj\_letiCalcDPF::SetParams(MyParams\_DPF &par)

{

//Устанавливаем параметры для m\_Param и вызываем функцию для расчета точек

m\_Param = par;

Calculate();

return true;

}

bool Cproj\_letiCalcDPF::Scale(CPoint & pt, double ScaleX, double ScaleY, double yShift, std::vector<POINT> \*vec, size\_t k)//Функция масштабирования

{

Cproj\_letiCalcDPF fn;

if (fn.Calc(vec, k, yShift))

{

//Приводим рассчитанные точки к int

pt.x = (int)floor(ScaleX \* fn.X);

pt.y = -(int)floor(ScaleY \* fn.Y);

return true;

}

return false;

}

int Cproj\_letiCalcDPF::GetNumPoints() // функция возвращающая кол-во точек

{

return m\_vecDPF.size();

}

std::vector<POINT> \*Cproj\_letiCalcDPF::GetPoints\_DPF()//функция возвращающая вектор рассчитанных точек

{

return &m\_vecDPF;

}

bool Cproj\_letiCalcDPF::Calculate()//Функция заполнения рассчитанными точками вектора m\_vecDPF

{

m\_vecDPF.clear();

Cproj\_letiDlg \*pDlg = (Cproj\_letiDlg \*)AfxGetMainWnd();//Получаем указатель на текущий диалог

if (!pDlg) {

return false;

}

std::vector<POINT> \*vec = pDlg->m\_Pt.GetPoints\_dPoint();//Получаем рассчитанные точки сигнальной функции

size\_t N = vec->size();

for (size\_t k = 0; k <= N / 2; k++)

{

double xScale = m\_Param.Width;

CPoint pt;

if (Scale(pt, xScale, m\_Param.Scale, m\_Param.yShift, vec, k))//Вызываем функции Scale,Shift для масштабирования относительно окна

{

if (Shift(pt))

m\_vecDPF.push\_back(POINT(pt));

}

}

return m\_vecDPF.size() > 0;

}

## stdafx.cpp

// stdafx.cpp: исходный файл, содержащий стандарт, включающий

// proj\_leti.pch будет предварительно откомпилированным заголовком

// stdafx.obj будет содержать предварительно откомпилированные сведения о типе

#include "stdafx.h"

## Proj\_letiDlg.h

// proj\_letiDlg.h : файл заголовка

//

#pragma once

//Подключаем заголовочные файлы других модулей и элементы ст.библиотеки

#include "afxwin.h"

#include "PaintPic.h"

#include "afxcmn.h"

#include<vector>

#include "Cproj\_letiCalc.h"

#include "PaintPic2.h"

#include "Cproj\_letiCalcDPF.h"

using namespace std;//используем стандартное пространство имен

// диалоговое окно Cproj\_letiDlg

class Cproj\_letiDlg : public CDialogEx

{

// Создание

public:

Cproj\_letiDlg(CWnd\* pParent = NULL); // стандартный конструктор

// Данные диалогового окна

#ifdef AFX\_DESIGN\_TIME

enum { IDD = IDD\_PROJ\_LETI\_DIALOG };

#endif

protected:

virtual void DoDataExchange(CDataExchange\* pDX); // поддержка DDX/DDV

// Реализация

protected:

HICON m\_hIcon;

// Созданные обработчики схемы сообщений

virtual BOOL OnInitDialog();

afx\_msg void OnSysCommand(UINT nID, LPARAM lParam);

afx\_msg void OnPaint();

afx\_msg void OnPaint2();

afx\_msg void OnBnClickedOk();

afx\_msg HCURSOR OnQueryDragIcon();

DECLARE\_MESSAGE\_MAP()

public:

afx\_msg void OnEnChangeEdit1();//обработчик ввода Scale и установки значений My\_Params

// static drawing

CPaintPic m\_paintPic;//Переменная контроля для класса PaintPic

CEdit m\_Calc;

afx\_msg void OnBnClickedCheck3();

afx\_msg void OnTimer(UINT\_PTR nIDEvent);//обработчик для системного таймера

afx\_msg void OnBnClickedSavePic();//обработчик сохранения изображения для IDC\_ANIMATE1

//переменная класса PaintPic control

dPoint m\_Pt;

//Обработчики стандартного и логарифмического увеличения графика

afx\_msg void OnRButtonUp(UINT nFlags, CPoint point);

afx\_msg void OnMouseMove(UINT nFlags, CPoint point);

// Переменная управления граф.окном для WM\_MOUSE\_MOVE(расчет координат)

CString m\_strF;

CStatic m\_st\_SH;

//

CString m\_HelpVal;

afx\_msg void On32773();//функция доп.меню "Выход"

afx\_msg void On32772();//функция доп.меню "Помощь"

// Переменная control для анимации второго окна

// Переменная для контроля ввода аргумента в значених 30-70

float m\_MinMaxValCheck;

CPaintPic2 m\_paintDPF;//переменная класса CPaintPic2 для контроля

afx\_msg void OnDeltaposSpin1(NMHDR \*pNMHDR, LRESULT \*pResult);

CSpinButtonCtrl m\_SpinPicSignal;//переменная контроля для IDC\_SPIN1

afx\_msg void OnBnClickedSavePicdpf();//обработчик сохранения изображения для IDC\_ANIMATE2

Cproj\_letiCalcDPF m\_DPF;//переменная контроля класса Cproj\_letiCalcDPF

afx\_msg BOOL OnMouseWheel(UINT nFlags, short zDelta, CPoint pt);//обработчик движения колесика мышки

bool check\_log=false;

CSpinButtonCtrl m\_spinF;

double argF;

afx\_msg void OnStnClickedStaticStep();

};

## Proj\_leti.h

// proj\_leti.h : главный файл заголовка для приложения PROJECT\_NAME

//

#pragma once

#ifndef \_\_AFXWIN\_H\_\_

#error "включить stdafx.h до включения этого файла в PCH"

#endif

#include "resource.h" // основные символы

// Cproj\_letiApp:

// О реализации данного класса см. proj\_leti.cpp

//

class Cproj\_letiApp : public CWinApp

{

public:

Cproj\_letiApp();

// Переопределение

public:

virtual BOOL InitInstance();

// Реализация

DECLARE\_MESSAGE\_MAP()

};

## PaintPic.h

#pragma once

// класс CPaintPic наследник CWnd

class CPaintPic : public CWnd

{

DECLARE\_DYNAMIC(CPaintPic)

public:

//Объявляем конструктор и виртуальный деструктор

CPaintPic();

virtual ~CPaintPic();

protected:

DECLARE\_MESSAGE\_MAP()

public:

afx\_msg void OnPaint();//объявляем обработчик ст.сообщений

//Переменная для передачи кол-ва точек

size\_t m\_Npt;

//Заливка фона для 1 графика

COLORREF m\_BackCol\_1;

//Цвет карандаша для 1 графика

COLORREF m\_GraphCol\_1;

};

## PaintPic2.h

#pragma once

// класс CPaintPic2 наследник CWnd

class CPaintPic2 : public CWnd

{

DECLARE\_DYNAMIC(CPaintPic2)

public:

//Объявляем конструктор и виртуальный деструктор

CPaintPic2();

virtual ~CPaintPic2();

protected:

DECLARE\_MESSAGE\_MAP()

public:

afx\_msg void OnPaint();//объявляем обработчик ст.сообщений

//Переменная для передачи кол-ва точек

size\_t m\_Npt;

//Заливка фона для 2 графика

COLORREF m\_BackCol\_1;

//Цвет карандаша для 2 графика

COLORREF m\_GraphCol\_1;

};

## Cproj\_letiCalc.h

#pragma once

#include <vector>

typedef struct

{

public:

//Объявляем переменные масштабирования,кол-ва точек для построения,центр отностиельно графика,частные переменные частотной модуляции

double Scale;

double xMax;

size\_t nPoints=360;

int yCenter;

int xCenter;

double dXRad;

double t;

double Fd = 10000000;

double f\_Fd=0.1\*Fd;

double Fm = 0.01\*Fd;

double m=f\_Fd/1;

double amplitude=1;

}MyParams\_t;//Объявляем структуру MyParams\_t

//Объявляем класс dPoint для расчета точек сигнального графика

class dPoint {

public:

dPoint();//Объявляем конструктор

public:

MyParams\_t m\_Param;

double x;

double y;

bool Calculate\_dPoint();//функция вычислений точек сигнальной функции

std::vector<POINT> m\_vecPt;//вектор дял рассчитанных точек

std::vector<POINT> \*GetPoints\_dPoint();//функция возвращающая вектор точек

bool SetParams(MyParams\_t &par);//функция установки параметров

//Переменные для расчета точек x,y для каждого элемента <POINT>m\_vecPt

double X;

double Y;

public:

//функции масштабирования

bool Scale(CPoint & pt, double ScaleX, double ScaleY, double Arg, double Param, double f\_Fd, double Fm, double m, double amplitude);

bool Shift(CPoint & pt);

};

//объявляем класс CMyFun для доп вычислений(разбиваем код на сотсавляющие)

class CMyFun

{

public:

CMyFun() {};

~CMyFun() {};

public:

double X;

double Y;

public:

bool Calc(double f, double k, double f\_Fd, double Fm, double m, double amplitude);

};

## Cproj\_letiCalcDPF.h

#pragma once

#include <vector>

typedef struct

{

//Иницилизируем размеры окна,размер Scale и переменную дял масштабирования yShift,переменную для передачи кол-ва точек

double Scale = 8.;

size\_t nPoints=360;

int Width;

int Height;

double yShift;

}MyParams\_DPF;//Объявляем структуру MyParams\_DPF

class Cproj\_letiCalcDPF

{

public:

//Объявляем перенменные X,Y для расчета точки(разбиваем код на составляющие)

double X;

double Y;

bool Calc(std::vector<POINT> \*vec, size\_t k, double yShift);//объявляем функцию вычисления с передачей по указателю вектора vec,k и yShift

public:

//Объявляем конструктор и деструктор

Cproj\_letiCalcDPF();

~Cproj\_letiCalcDPF();

protected:

//Функции масштабирования

bool Shift(CPoint &pt);

bool Scale(CPoint & pt, double ScaleX, double ScaleY, double yShift, std::vector<POINT> \*vec, size\_t k);

public:

bool Calculate();//функция заполнения массива по переданным расччитанным точкам из m\_vecDPF

MyParams\_DPF m\_Param;//Объявляем объект структуры My\_Params\_D

bool SetParams(MyParams\_DPF &par);//объявляем функцию установки параметров

int GetNumPoints();//объявляем функцию передачи кол-ва рассчитанных точек

std::vector<POINT>\* GetPoints\_DPF();//объявляем функцию возвращающую вектор рассчитанных точек m\_vecDPF

std::vector<POINT> m\_vecDPF;//вектор типа POINT для рассчитанных точек

};

## Stdafx.h

// stdafx.h: включите файл для добавления стандартных системных файлов

//или конкретных файлов проектов, часто используемых,

// но редко изменяемых

#pragma once

#ifndef VC\_EXTRALEAN

#define VC\_EXTRALEAN // Исключите редко используемые компоненты из заголовков Windows

#endif

#include "targetver.h"

#define \_ATL\_CSTRING\_EXPLICIT\_CONSTRUCTORS // некоторые конструкторы CString будут явными

// отключает функцию скрытия некоторых общих и часто пропускаемых предупреждений MFC

#define \_AFX\_ALL\_WARNINGS

#include <afxwin.h> // основные и стандартные компоненты MFC

#include <afxext.h> // расширения MFC

#include <afxdisp.h> // классы автоматизации MFC

#ifndef \_AFX\_NO\_OLE\_SUPPORT

#include <afxdtctl.h> // поддержка MFC для типовых элементов управления Internet Explorer 4

#endif

#ifndef \_AFX\_NO\_AFXCMN\_SUPPORT

#include <afxcmn.h> // поддержка MFC для типовых элементов управления Windows

#endif // \_AFX\_NO\_AFXCMN\_SUPPORT

#include <afxcontrolbars.h> // поддержка MFC для лент и панелей управления

#ifdef \_UNICODE

#if defined \_M\_IX86

#pragma comment(linker,"/manifestdependency:\"type='win32' name='Microsoft.Windows.Common-Controls' version='6.0.0.0' processorArchitecture='x86' publicKeyToken='6595b64144ccf1df' language='\*'\"")

#elif defined \_M\_X64

#pragma comment(linker,"/manifestdependency:\"type='win32' name='Microsoft.Windows.Common-Controls' version='6.0.0.0' processorArchitecture='amd64' publicKeyToken='6595b64144ccf1df' language='\*'\"")

#else

#pragma comment(linker,"/manifestdependency:\"type='win32' name='Microsoft.Windows.Common-Controls' version='6.0.0.0' processorArchitecture='\*' publicKeyToken='6595b64144ccf1df' language='\*'\"")

#endif

#endif

# Выбор и обоснование собственных типов,переменных,классов

## Вводимые классы

1.Cdpoint-класс для расчетов точек графика сигнальной функции и их масштабирования относительно окна

2.CMyFun-класс входящий в dPoint;также используется для расчетов,введен для повышения читаемости кода;

3.CPaintPic-класс рисования для графика сигнальной функции

4.CPaintPic2-класс рисования для графика ДПФ

5.Cproj\_letiCalcDPF-класс для расчетов точек ДПФ для сигнальной функции и их масштабирования относительно окна

## Вводимые типы и структуры

1.Структура MyParams\_t для хранения параметров для построения графиков и расчетов точек сигнальной функции

Параметры структуры MyParams\_t:

*Иницилизированные:*

size\_t nPoints=360

double Fd = 10000000

double f\_Fd=0.1\*Fd

double Fm = 0.01\*Fd

double m=f\_Fd/1

double amplitude=1

*Объявленные:*

double Scale;

double xMax;

int yCenter;

int xCenter;

double dXRad;

double t;

2.Структура MyParams\_DPF для хранения параметров для построения графиков и расчетов точек ДПФ

Параметры структуры MyParams\_DPF:

*Иницилизированные:*

double Scale = 20.

size\_t nPoints=360

*Объявленные:*

int Width

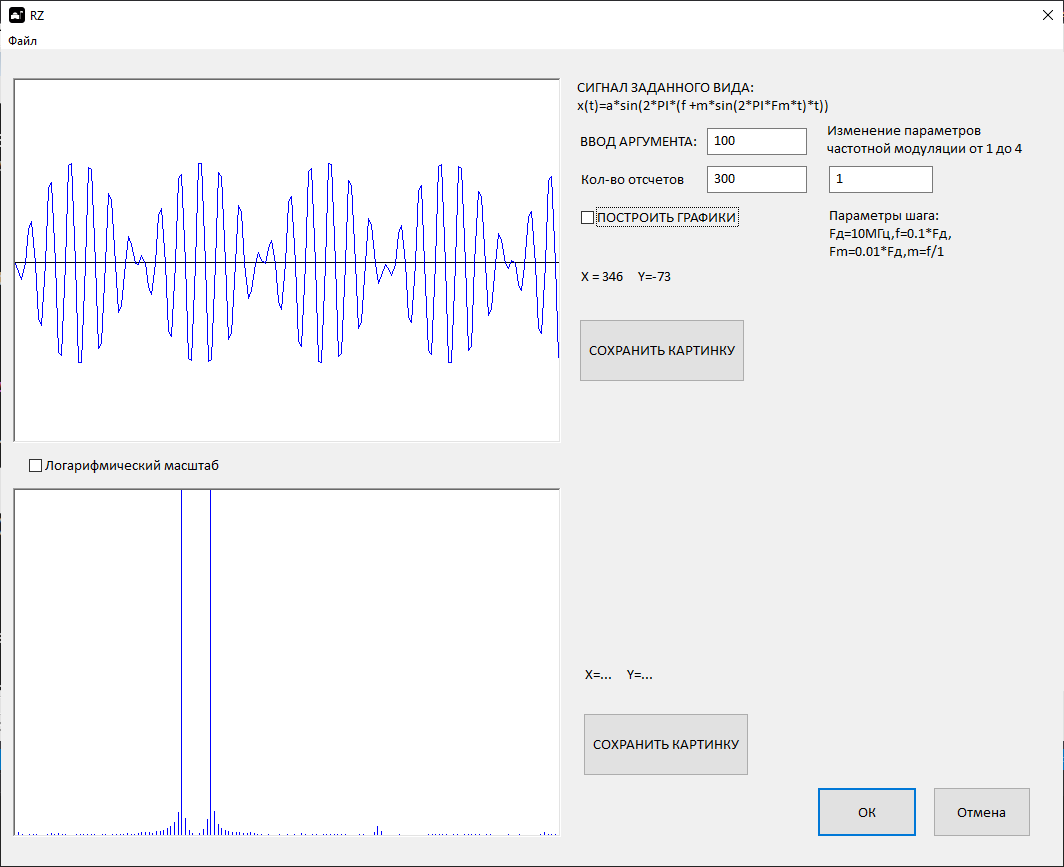
int Height

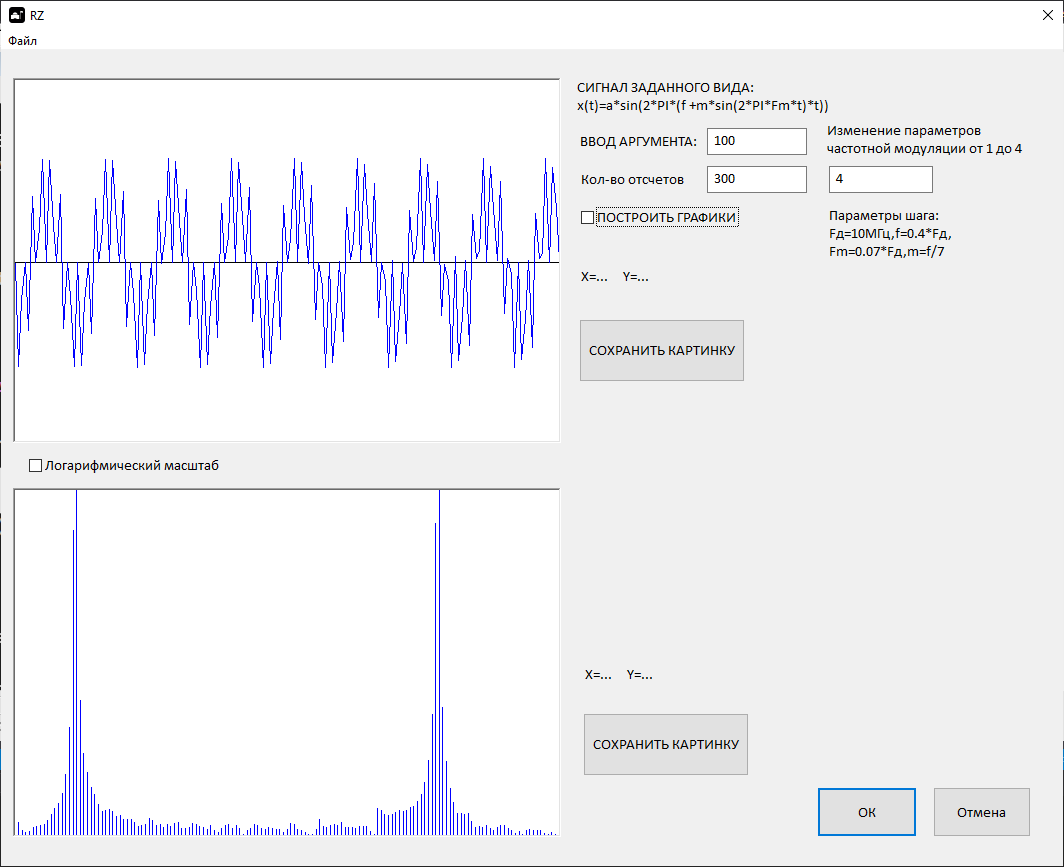
double yShift

# Примеры работы программы и контрольный пример

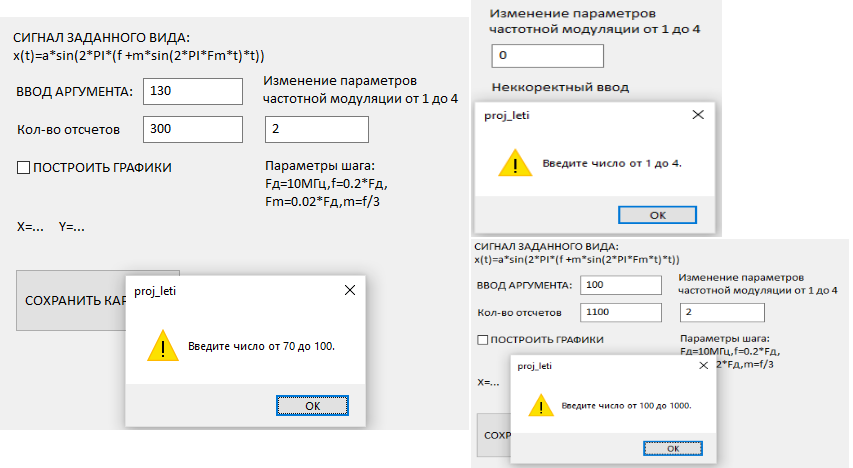
## Примеры работы программы

1.Построение графиков с аргументом 100 с отображением координаты на графике по наведению курсора с различными шагами(разные заданные параметры частотной модуляции)

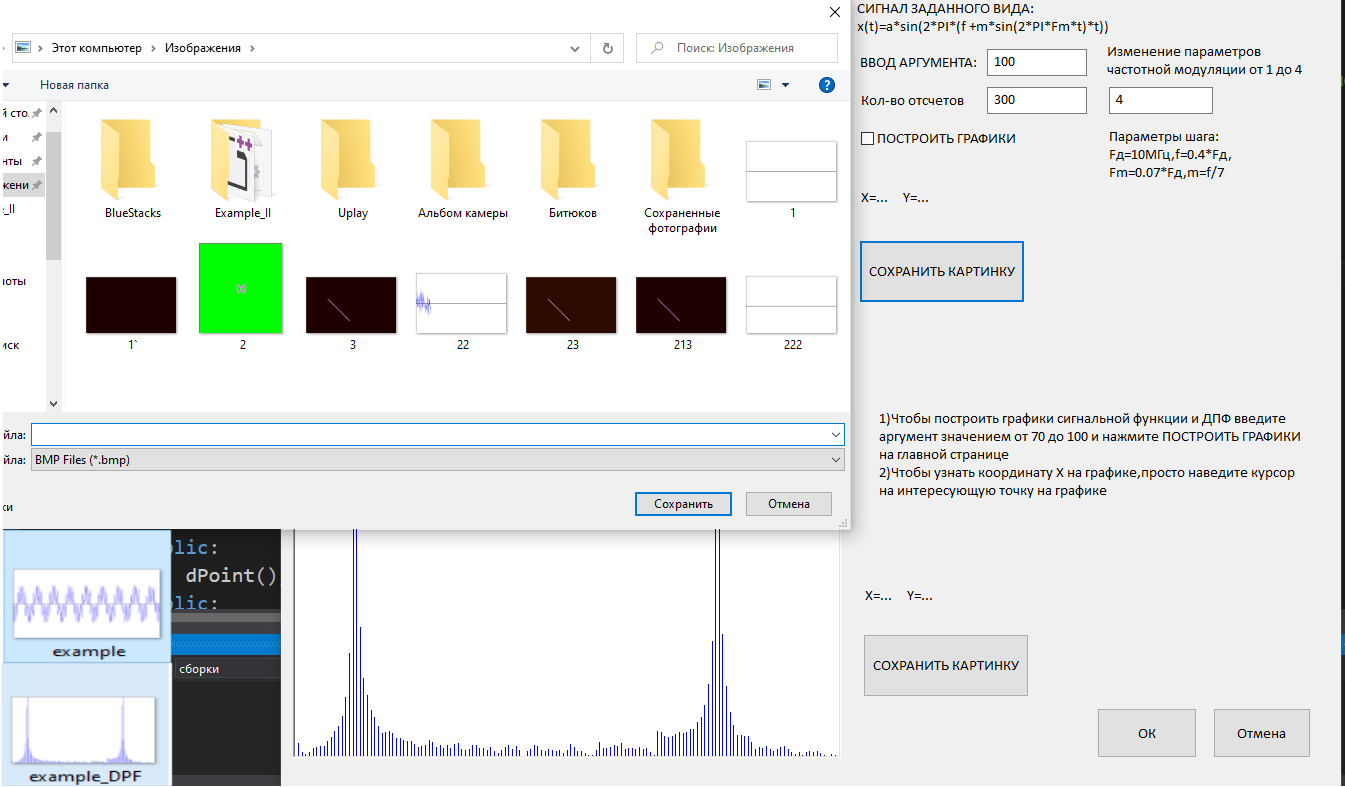




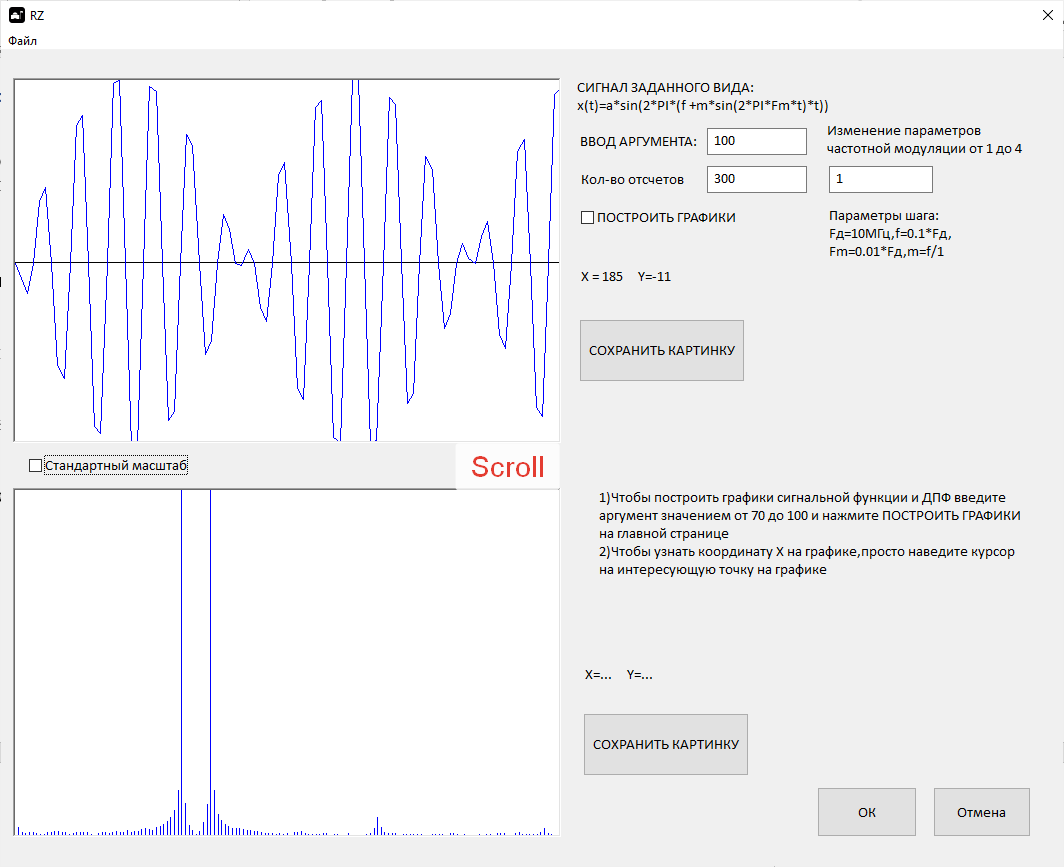
2.Попытка ввести неккоректные значения в поле:ввода аргумента(Scale),количества отсчетов,изменения параметров частотной модуляции.

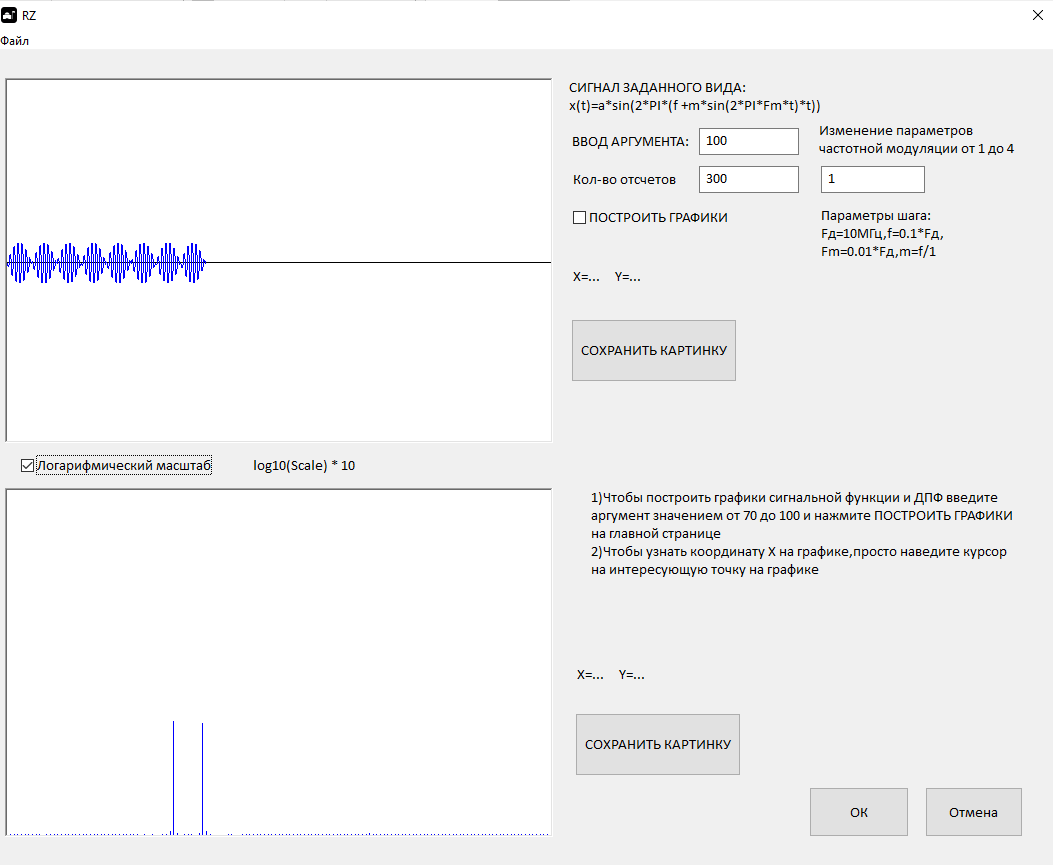


3.Сохранение графиков

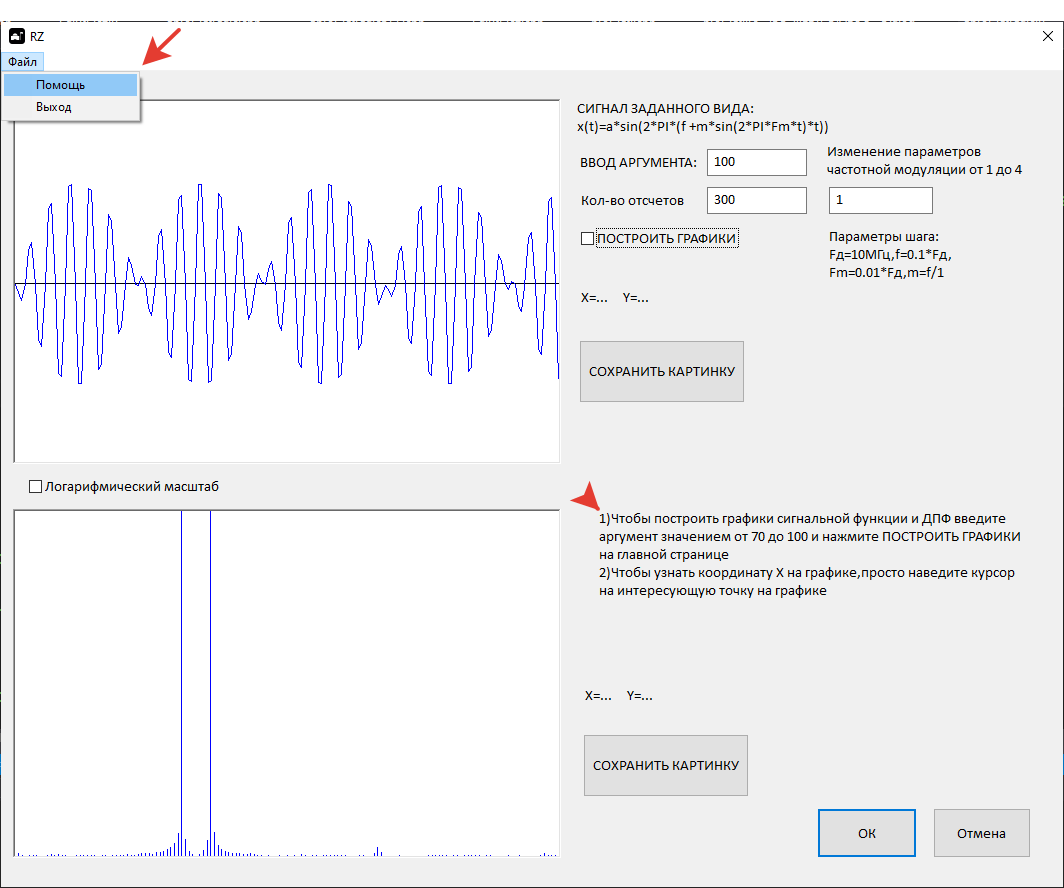


4.Логарифмическое увеличение графиков;увеличение графиков с помощью колеса мыши



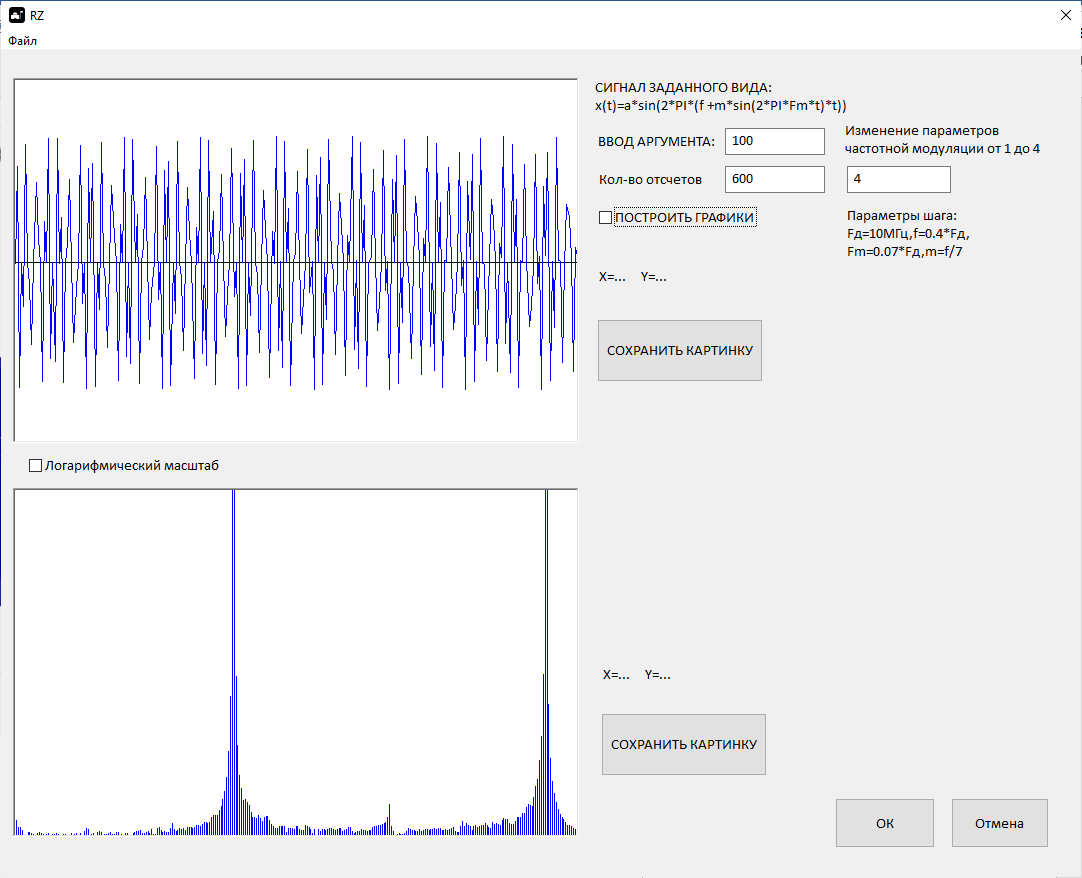


5.Вызов дополнительного меню и использование его функций

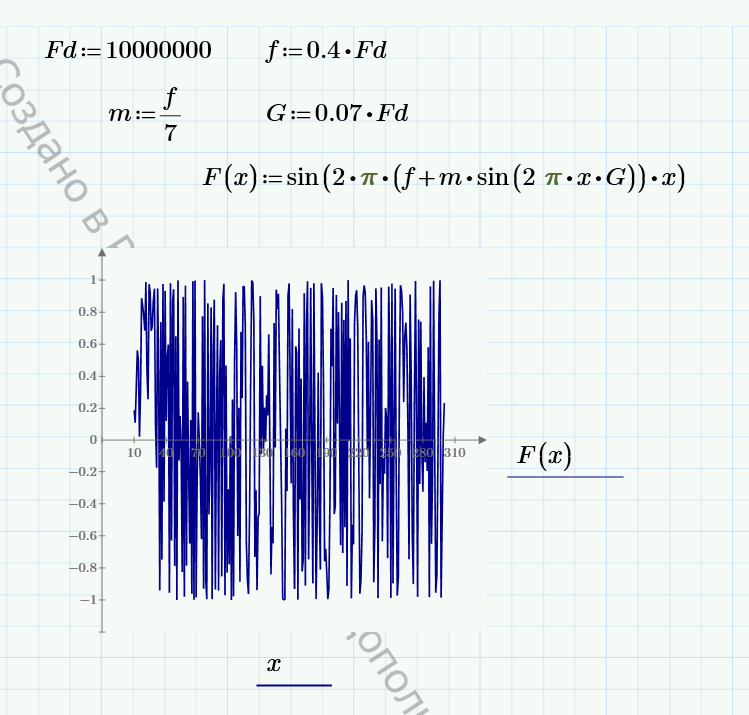


## Контрольный пример

Пример графика сигнальной функции



Контрольный пример из MathCad(Express Prime 6.0)



# Особенности программы

*Особенности*

1.Программа имеет Dynamic Layout по горизонтали ;возможно растяжение окна по горизонтали->увеличение холста графиков по горизонтали ;все элементы ПИ также сдвигаются и не накладываются друг на друга.

2.Программа записывает значение аргумента(Scale) в текущей сессии при нажатии ОК в реестр Windows и при следующем запуске программы автоматически устанавливает его для аргумента(Scale).

3.В программе есть вызываемое дополнительно меню с функциями :”Помощь” и “Выход”.

4.Программа имеет проверку на значение вводимого аргумента(Scale),кол-ва точек(отсчетов),номера шага параметров частотной модуляции.