ФЕДЕРАЛЬНОЕ АГЕНТСТВО ПО ОБРАЗОВАНИЮ
НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Физический факультет

Кафедра радиофизики

Изучение среды программирования LabWindows/CVI

Методические указания к лабораторной работе №1
практикума ТСАНИ

Новосибирск, 2016

УДК 621.3.01
ББК з-84я73-4
П691
ISBN-978-5-94356-736-0

Лабораторная работа посвящена изучению среды программирования LabWindows/CVI. В данном пособии изложена структура данной среды программирования, приведены сведения, необходимые для начала работы. Так-же в приложении приведены справочные данные по компонентам, рекомендуемых к использованию в данной работе и во всем практикуме.

Для выполнения данной работы студенту нужно будет написать программу – виртуальный прибор, выводящую на график сигнала синусоидальной формы с параметрами, задаваемыми на лицевой панели данного прибора.

Составитель Середняков С.С.

Рецензент Батраков А.М.

Ответственная за выпуск Тенекеджи О. А.

Методическое пособие подготовлено в рамках реализации
Программы развития НИУ-НГУ на 2009-2018 гг.

Новосибирский государственный
университет, 2016

1.Введение

Среда программирования LabWindows/CVI была разработана фирмой National Instruments в середине 80-х годов прошлого века. В отличие от ряда широкоизвестных средств разработки (Borland C++, Delphi, MS Visual Studio и т.д.), LabWindows/CVI ориентирована в первую очередь на разработку программного обеспечения, управляющего различными научными экспериментами, технологическими либо измерительными процессами. Эта направленность и определила содержание и функциональность данного программного продукта.

Среда LabWindows/CVI включает в себя интегрированную среду разработки (IDE) на основе языка ANSI C, и обширную библиотеку компонентов, которая состоит из следующих частей:

1. Набор компонентов для создания пользовательского интерфейса – создание и вывод на экран панелей, вывод графиков, создание регуляторов, индикаторов, кнопок и т.д (**User Interface Library**).
2. Набор библиотек для управления измерительными и управляющими устройствами(**VISA Library, VXI Library, RS-232 Library, GPIB/GPIB 488.2 Library и др**).
3. Средства для удаленного управления приложениями через локальную сеть или Интернет(**Network Variable Library**).
4. Средства обработки полученных данных – фильтрация, математическая обработка полученных данных, генерация сигналов различной формы (**Analysis Library, Advanced Analysis Library**).
5. Стандартная библиотека **ANSI C**.

Также в библиотеку входят средства для использования технологий DDE, ActiveX, различных сетевых протоколов, технологий .NET, создания многопоточных приложений и др.

Имея набор таких средств в среде LabWindows/CVI с одной стороны можно легко автоматизировать простые «настольные» эксперименты. С другой стороны мощь перечисленных выше библиотек позволяет разрабатывать распределенные системы управления, управляющие сложными научно-технологическими процессами с большим количеством разнообразной аппаратуры.

2. Использование LabWindows CVI.

2.1. Интегрированная среда разработки (IDE).

Среда LabWindows/CVI во многом похожа на широко известные интегрированные среды разработки. На рисунке 1 приведено рабочее окно среды LabWindows/CVI.

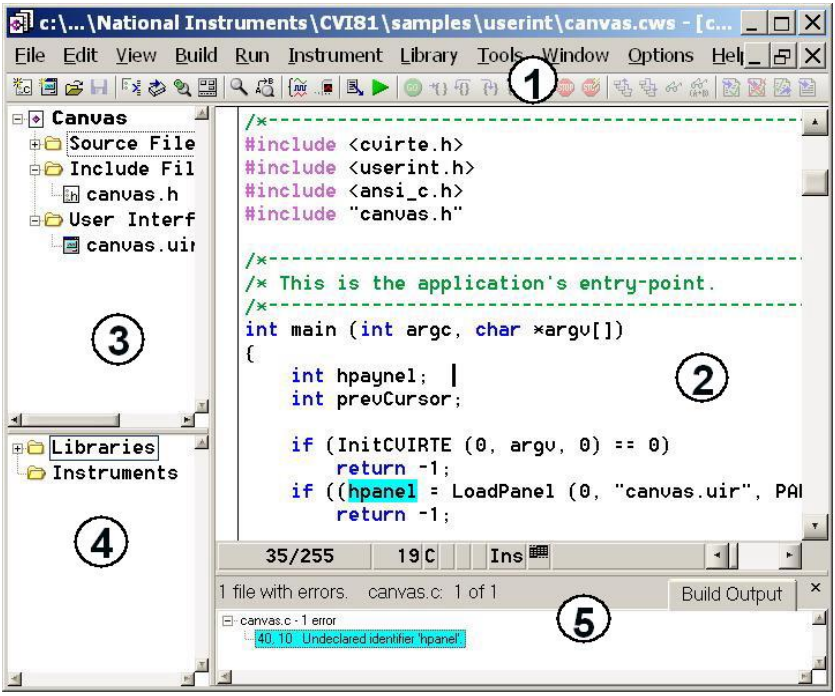

Рисунок 1. Интерфейс среды разработки LabWindows/CVI

Как видно, главное окно состоит из следующих основных частей:

1. Строка меню, панель инструментов.
2. Окно для редактирования исходного кода программы, либо панелей интерфейса пользователя (окон может быть несколько).
3. Дерево проекта – содержит список файлов данного проекта и позволяет управлять файлами проекта – открывать в окне 2, удалять из проекта и т.д. При двойном щелчке левой кнопкой мыши на файле, он открывается в окне для редактирования справа.

4. Дерево библиотек – содержит структурированный список всех функций, содержащихся в библиотеках LabWindows/CVI. Найдя нужную функцию, можно получить о ней справочную информацию, или вставить вызов этой функции в текст программы, задав необходимые аргументы.

5. Область выходных данных – может содержать информацию об отладке программы, ошибки компиляции, исполнения, окна для наблюдения значений переменных(**Watch, Variables**) а также результаты поиска в тексте программы.

2.2. Структура проекта приложения в LabWindows/CVI.

Для начала работы над программой нужно создать новый проект, либо открыть существующий. Чтобы открыть уже существующий проект нужно выполнить команду **File – Open - Project(*.prj)….**

Для того чтобы создать новый проект нужно выполнить команду меню **File->New->Project from Template.** Далее вам будет предложено задать тип проекта, его имя и расположение (путь) на жестком диске. Проекты в среде LabWindows/CVI могут быть двух типов:

1. **Command-line Application** – проект консольного приложения без графики и пользовательского интерфейса. Можно использовать для ввода и вывода числовых значений.
2. **User Interface Application** - проект приложения с окном для пользовательского интерфейса. Пользователь может добавлять, удалять и редактировать различные элементы интерфейса (кнопки, поля ввода-вывода, графики …).

Допустим, мы создали проект типа **User Interface Application** с именем **FirstProject**. Проект данного типа обычно содержит файлы следующих расширений:

1. **.PRJ** – файл созданного проекта.
2. **.C , .H** - файлы исходных текстов программы на языке С.
3. **.UIR** - Содержит данные о созданных окнах приложения и элементах пользовательского интерфейса на них. При выборе данного файла в дереве проекта открывается редактор интерфейса пользователя в окне редактирования.
4. **.CWS** – файл содержащий информацию о временных настройках проекта(положение и состояние окон, даты изменения файлов и т.д.)
5. **.FP** – файлы панелей функций для драйверов устройств (во вновь созданном проекте отсутствуют).

После создания проекта вышеуказанным способом дерево проекта должно содержать следующие файлы: **FirstProject.c, FirstProject.uir**. В файле **FirstProject.uir** находится одна панель, которая является главным

окном программы. В файле **FirstProject.c** находится заготовка исходного текста программы – подключение нужных библиотек (**#include** <>) и функция **main()** с некоторым исходным кодом, который необходим для старта программы.

2.3. Структура программы в среде LabWindows/CVI.

Если дважды щелкнуть левой кнопкой мыши на файле **FirstProject.c**, его содержимое откроется в окне редактирования. Если убрать сгенерированные средой комментарии, у вас должно появиться рабочее окно сходное с показанным на рисунке 2.

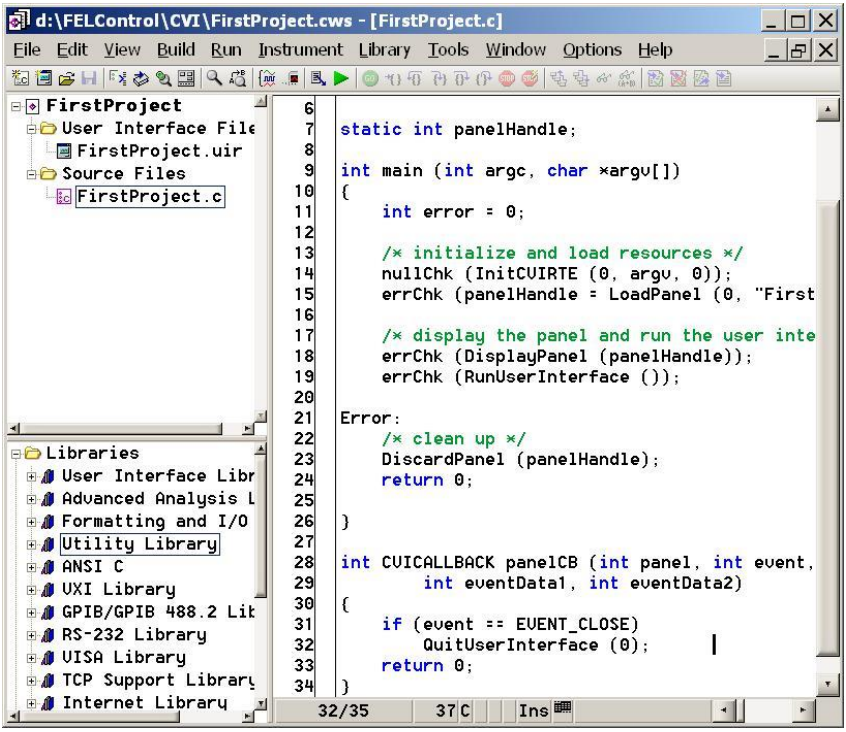

Рисунок 2. Рабочее окно после создания проекта.

Рассмотрим содержимое функции **main**:

После входа в функцию **main**, происходит следующий вызов (строка 14):

```
nullChk (InitCVIRTE (0, argv, 0));
```

который необходим для инициализации библиотек LabWindows/CVI. Далее (строка 15) происходит загрузка панели, находящейся в файле **FirstProject.uir** в память:

```
errChk(panelHandle=LoadPanel(0,"FirstProject.uir",PANEL));
```

и отображение этой самой панели на рабочем столе:

```
errChk (DisplayPanel (panelHandle));
```

После этого, вызовом следующей функции:

```
errChk (RunUserInterface ());
```

программа переходит в режим ожидания и обработки внешних событий (нажатие кнопок, обновление значений, и т.д.). Выход их этой функции происходит только после того, как пользователь захочет выйти из программы. Тогда программа перейдет к следующей строке:

```
DiscardPanel (panelHandle);
```

после чего выполнение программы будет завершено.

Данный порядок вызова функций является необходимым – нельзя удалять, либо менять местами данные конструкции !

Исходя из вышеизложенного порядка начального запуска главного окна программы, можно выделить несколько областей в функции **main()**, для добавления своего кода:

- Область перед инициализацией библиотек (строки 11-13). Можно использовать для начальной инициализации переменных, массивов, ввода данных с клавиатуры, чтения из файлов и т.д.
- Область после загрузки главного окна в память (строки 16-17). Здесь можно выполнить все начальные действия с открывшимся окном: задавать свойства элементов управления, масштабы графиков, начальные положения регуляторов, а также вывести в них некие начальные данные (нарисовать график, вывести числовое значение на экран).
- Область перед завершением программы (после выполнения строки **DiscardPanel (panelHandle);**). Можно использовать для сохранения данных на диск, закрытия открытых файлов, отключения от устройств.

2.4. Разработка программ в среде LabWindows/CVI.

Процесс разработки приложения в LabWindows/CVI можно разделить на 2 части:

1. **Редактирование окон пользовательских интерфейсов, добавление элементов управления, редактирование их свойств.**
2. **Написание исходного текста программы.**

Редактирование пользовательского интерфейса начинается с главной панели, которая открывается, если дважды щелкнуть мышкой на файле **.uir** в дереве проекта. Открывшаяся панель имеет набор свойств, доступ к которым можно получить, если дважды на ней щелкнуть левой кнопкой мыши. При этом откроется окно свойств данного элемента (см. рис.3).

Каждый элемент управления имеет индивидуальный набор свойств, который отображается в аналогичном окне, но есть свойства присущие всем элементам управления. Это, прежде всего уникальное имя элемента (отображается в поле с названием **Constant Name**) и имя функции-обработчика событий от данного элемента (поле **CallBack Function**).

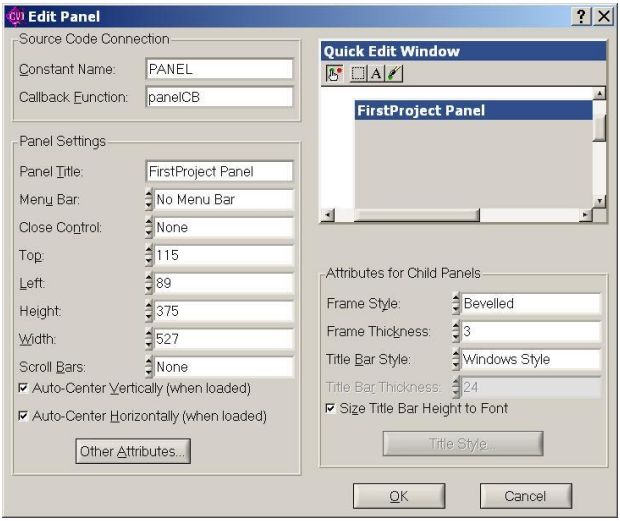

Рисунок 3. Окно свойств элемента

Для того чтобы добавить элемент управления на данную панель, нужно щелкнуть правой кнопкой мыши на свободном месте панели. При этом появится список элементов управления сгруппированных по типам (см.рис.

4), зайдя в которые можно выбрать интересующий элемент. Ниже приводятся наиболее часто используемые типы элементов управления из всего списка:

- **Numeric** – элементы управления для ввода-вывода числовых данных.
- **Command Button** - обычная кнопка типа «**OK**».
- **Toggle Button** – кнопки, имеющие 2 состояния (нажатое и ненажатое).
- **Led** – «лампочки» отображающие 2 состояния (on и off).
- **Graph –** элементы для построения графиков.
- **Timer** - элемент для обеспечения таймирования программы – периодического выполнения некоторых действий.

Полный список элементов управления и функции, которые рекомендуется использоваться в этом практикуме, а также их свойства приведены в главе 2 методического пособия «Справочные материалы по практикуму ТСАНИ».

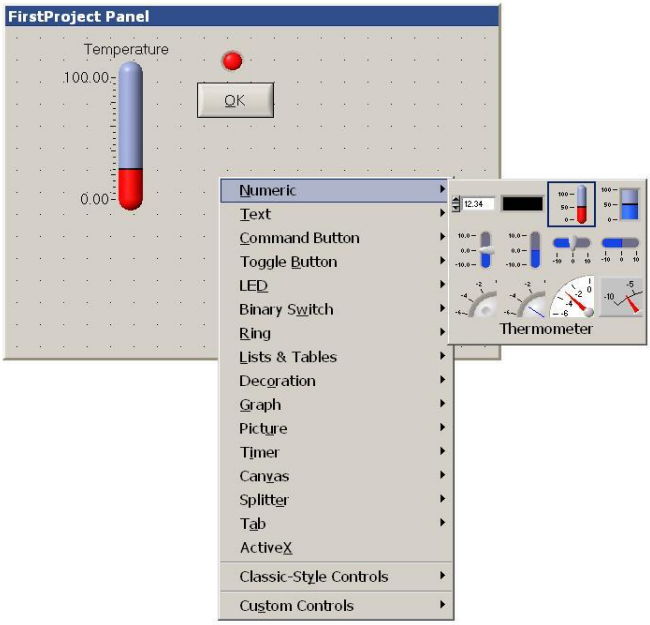

Рисунок 4. Выбор элемента управления из списка.

После появления элемента можно аналогичным способом открыть окно его свойств. Также можно открыть контекстное меню данного элемента, если щелкнуть на нем правой кнопкой мыши. При этом появится меню, аналогичное изображенному на рисунке 5. Доступные команды меню имеют следующее назначение:

1. **Generate Control Callback** – создать функцию-обработчик элемента в тексте программы с именем, заданным в окне свойств.
2. **View Control Callback** - перейти к функции обработчику в тексте программы.
3. **Control Help** - вызов справки по данному элементу управления.
4. **Edit Control** - редактировать свойства элемента. При этом будет вызвано окно свойств элемента (см. рис. 3).

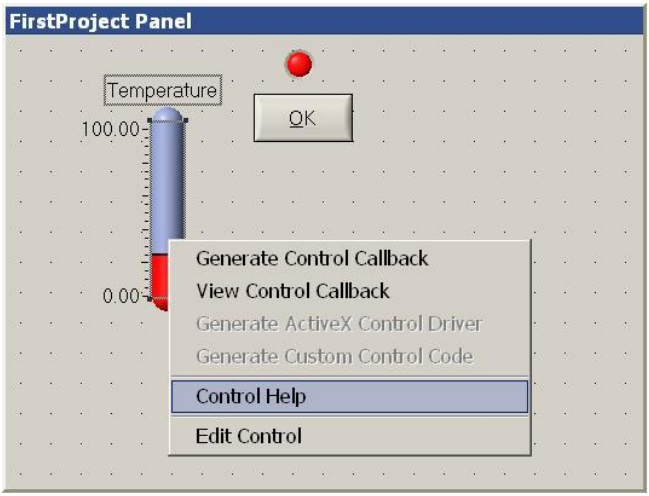

Рисунок 5. Вызов контекстного меню элемента.

За исключением необходимой начальной инициализации, описанной в конце пункта 3, написание исходного текста программы в LabWindows/CVI обычно сводится к созданию и заполнению кодом функций-обработчиков событий от элементов пользовательского интерфейса(т.н. Callback-функций).

Как и любое Windows-приложение, программа, разработанная в LabWindows/CVI, управляется с помощью внешних событий (нажатие кнопкой мыши, нажатие клавиши, изменение содержимого текстового поля и т.д.). Чтобы обеспечить обработку этих событий, в среде LabWindows/CVI предусмотрен механизм Callback-функций – функций,

которые выполняются в ответ на событие, пришедшее от системы соответствующему элементу управления.

Данные Callback-функции генерируются автоматически по команде пользователя и имеют строго определенный набор аргументов. Имя этой функции задается в панели свойств (поле Callback Function:), а сама функция в тексте программы создается по команде контекстного меню **Generate Control Callback.** На рисунке 6 представлен пример одной из Callback-функций: функция с именем **OKButtonCB**. Как видно из текста, функция имеет следующие параметры:

- **panel** – идентификатор панели. Переменная, которая возвращается после выполнения функции **LoadPanel** (см.выше).
- **control** – имя элемента управления.
- **event** – идентификатор события.

Остальные аргументы являются дополнительными данными, которые можно передать в callback-функцию, и зависят от типа элемента управления. Как видно, основным параметром, определяющим ход выполнения callback-функции, является параметр **event** – имя события. С помощью механизма выбора(**switch**) по значению этого параметра, приведенная ниже callback-функция будет реагировать на следующие события:

1. **EVENT_COMMIT** - общее событие, которое возникает, если элемент выполняет основное, требуемое от него действие. В случае кнопки **OK** это просто нажатие на кнопку.
2. **EVENT_LEFT_CLICK** – событие, возникающее при нажатии левой кнопки мыши.
3. **EVENT_RIGHT_CLICK** - событие, возникающее при нажатии правой кнопки мыши.
4. **EVENT_GOT_FOCUS** – данный элемент управления стал активным.

```
int CVICALLBACK OKButtonCB (int panel, int control, int event,
        void *callbackData, int eventData1, int eventData2)
{
    switch (event)
    {
        case EVENT_COMMIT:

            MessagePopup ("Event-Commit","Event-Commit!!");

            break;
        case EVENT_LEFT_CLICK:

            MessagePopup ("Event-LeftClick","Event-LeftClick!!");

            break;
        case EVENT_RIGHT_CLICK:

            MessagePopup ("Event-RightClick","Event-RightClick!!");

            break;
        case EVENT_GOT_FOCUS:

            MessagePopup ("Event-GotFocus","Event-GotFocus!!");

            break;
    }
    return 0;
}
```

Рисунок 6. Callback-функция кнопки OK

Кроме вышеперечисленных вам, возможно, понадобятся следующие события:

- **EVENT_CLOSE –** событие возникающее, если пользователь хочет закрыть панель.
- **EVENT_TIMER_TICK** – событие, возникающее при срабатывании таймера(см. Приложение 1).
- **EVENT_PANEL_SIZE** – появляется при изменении размеров окна.

В среде LabWindows/CVI для каждого элемента управления можно создать отдельную Callback-функцию. Однако есть возможность задать одну Callback-функцию для разных элементов управления. Для этого в свойствах этих элементов нужно указать одинаковое имя Callback-функции, а в самой функции реализовать выбор исполняемого кода с помощью конструкций **switch/case** или **if/else** по аргументу функции **control**, который содержит имя элемента управления, который совершил вызов(см. пример ниже):

```
int CVICALLBACK CommonCB(int panel, int control,
            int event,void *callbackData, int eventData1, int eventData2)
{
  switch (event)
   {
      case EVENT_COMMIT:

      switch(control)
       {
          case  PANEL_NUMERIC_VAL1:
        // код для обработки событий от элемента NUMERIC_VAL1
          break;
          case  PANEL_NUMERIC_VAL2:
        // код для обработки событий от элемента NUMERIC_VAL2
          break;
       }
      break;
   }
      return 0;
}
```

Типы событий, которые будут сразу включены в callback-функцию можно настроить. Для этого нужно, находясь в окне редактирования пользовательского интерфейса, выполнить команду меню **Code->Preferences->Default Panel Events** для элементов управления, либо **Code->Preferences->Default Panel Events** для панелей, и выбрать нужные события из списка, проставив напротив них галочки.

Запуск на выполнение программы осуществляется командой **Run->Debug FileName.exe** (**Shift+F5)**. Также в меню **Run** находятся средства отладки приложения, приведенные в главе 2 методического пособия «Справочные материалы по практикуму ТСАНИ».

.

2.5. Справочная система.

В среде LabWindows/CVI к услугам разработчиков предоставляется обширная и удобная в использовании справочная система. Вызвать ее можно следующими способами:

1. Из текста программы, если навести курсор на вызов интересующей функции и нажать **F1**. При этом вызовется справочная система, сразу же открытая на справочной странице выбранной вами функции.

2. Выполнить команду меню **Help->Contents**.
3. Выбрать команду **Control Help** в контекстном меню элемента управления.

Сама справочная система организованна древовидным способом. Структура справки находится во вкладке **Содержание**. Из всего каталога справочной информации, наиболее полезной является вкладка **Library Reference,** в которой представлена вся библиотека функций LabWindows/CVI, сгруппированная по разделам. Примерный состав библиотеки приводился во введении. Группировка функций по области применения позволяет искать подходящую функцию, не зная всего содержания библиотеки. Это может быть очень полезным, например, для математической обработки полученных данных.

3. Практические задания.

Задание 1

Написать программу, которая бы вычисляла и выводила на экран дискриминант и корни квадратного уравнения, а также координаты вершины параболы по введенным коэффициентам (**A,B,C**). Программа должна реализовывать следующую функциональность:

- Коэффициенты квадратного уравнения (**A,B,C**) программа должна предлагать ввести при старте программы. Для этого удобно пользоваться функциями **printf, scanf.**
- Само вычисление корней должно выполняться внутри отдельной функции, которая должна иметь следующий прототип:

```
void CalculateRoots (float A, float B, float C, float *pD, float *pValue1,
float *pValue2);
```

Функция должна принимать коэффициенты квадратного уравнения (`A,B,C`) и выдавать после своего выполнения значение вычисленного дискриминанта через указатель `pD`. Через указатели `pValue1` и `pValue2` должны передаваться значения корней `X1` и `X2` в случае положительного дискриминанта, или значения реальной и мнимой части комплексно-сопряженных корней в случае, когда `D`<0.

- **!!! Внутри самой функции CalculateRoots не должно быть функций ввода-вывода (printf, scanf и т.д.). Весь ввод и вывод производить в главной вызывающей функции (main), до, либо после вызова данной функции.**
- Функция должна корректно отрабатывать все предельные случаи (**a=0, b=0,c=0** и т.д.).
- Вычисление координат вершин параболы должны выполняться до либо после функции **CalculateRoots.**
- Программа должна работать циклическим способом – после вычисления переходить в режим ожидания, а после нажатия какой-либо клавиши на клавиатуре, переходить на начало выполнения (ввод коэффициентов **A,B,C**, вычисление….) и так до нажатия некой клавиши «Выход» (например **“q”**). Для этого можно воспользоваться функциями **GetKey()** и **KeyHit().**

Для выполнения этого задания нужно создать проект типа **Command-line Application.**

Задание 2

1. Выведите на экран график синусоиды с заданными амплитудой и частотой.

Для этого нужно создать новый проект типа с пользовательским интерфейсом (**команда File – New – Project from Template** – тип проекта – **User Interface Application**). Далее, с помощью редактора пользовательских интерфейсов нужно создать на главной панели элемент управления «График» (**Graph->Graph)**.

Для правильной организации программы нужно создать функцию (например **CreateSignal()**), которая генерировала бы сигнал синусоиды, с заданными параметрами и сохраняла его в массив. Массив должен быть глобальной переменной и иметь размерность 1000 значений, что соответствовало бы времени 1000 миллисекунд. Количество периодов синусоиды в сгенерированном массиве должно соответствовать заданной частоте при генерации сигнала.

Для вывода на экран нужно также сделать отдельную функцию **DrawGraph()** которая задавала бы фиксированные масштабы по обеим осям графика и выводила массив построенного сигнала на экран.

После этого нужно вставить вызовы этих функций в функцию **main()** в то место, где уже создана главная панель программы(см. п. 2.4). График

должен иметь следующие оси: по X – время (T), диапазон - 0…1000.0 ms(миллисекунды), по Y – напряжение(U), диапазон -15.0…15.0 V(вольты).

Для этого лучше пользоваться функциями: **DeleteGraphPlot, SetAxisScalingMode, PlotY** (см главу 2 методического пособия «Справочные материалы по практикуму ТСАНИ»

2. Сделать так, чтобы график обновлялся в реальном времени с частотой 2 Гц. Для этого создать элемент управления «Таймер» (**Timer->Timer)**.

Также добавить поля для ввода значений амплитуды сигнала (в вольтах), частоты (в герцах) и начальной фазы (в радианах). Для этого можно использовать элементы управления **Numeric->Numeric**. Для считывания значений из полей ввода нужно воспользоваться функцией **GetCtrlVal.**

График синусоиды должен непрерывно перерисовываться с вновь веденными значениями.

3. Добавить шум к синусоиде. Для этого можно использовать функцию **Random(Rmin,Rmax).** Добавить регулятор уровня шума (элемент **Numeric->Knob**).

4. Добавить команду сохранения последнего сигнала в файл. Для этого нужно создать элемент «кнопка» (например **Command Button->Square Command Button**), и в её callback-функцию вставить сохранение массива сигнала. Для работы с файлами пользоваться функциями **fopen, fprintf, fclose** (см. п. 7.2. главы 1 «Справочного руководства по практикуму ТСАНИ»). Имя файла для сохранения может быть постоянным.

5. Добавить вычисление и вывод на экран мощности полученного сигнала. Для этого нужно создать в главном окне ещё один элемент типа «График».

Для удобства можно создать еще одну функцию, например **CalculatePower()** в которую вынести конфигурацию и очистку массива мощности сигнала, собственно вычисление и вывод этого массива на график. Далее, вставить вызов этой функции в callback-функцию таймера для непрерывного исполнения.

4. Литература

National Instruments LabWindows/CVI Help.