МИНИСТЕРСТВО ВЫСШЕГО И СРЕДНЕГО СПЕЦИАЛЬНОГО ОБРАЗОВАНИЯ РЕСПУБЛИКИ УЗБЕКИСТАН

ТАШКЕНТСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ им. АБУ РАЙХАНА БЕРУНИ

На правах рукописи.

УДК 519.876.5

Факультет: «Электроника и автоматика»
Кафедра: «Электроника и микроэлектроника»

УСАРОВ ОРИФЖОН ОЛИМОВИЧ

“Создание виртуальных приборов по электронике в среде LabVIEW”

Специальность: 5А310801 - «Промышленная электроника»

Д И С С Е Р Т А Ц И Я

на соискание академической степени магистра

Научный руководитель: **д.ф-м.н., проф. Х. М. Илиев**

Ташкент – 2013

СОДЕРЖАНИЕ

Аннотация к диссертационной работе магистра Усарова О.О.
Тема: «Создание виртуальных приборов по электронике в среде Lab VIEW »

В данной диссертационной работе описывается программный язык Lab VIEW (Laboratory Virtual Instrument Engineering Workbench), который разработан фирмой «National Instruments», а также приведены примеры создания виртуального прибора в среде Lab VIEW.

Основной целью данной магистерской работы является изучение приемов программирования на LabVIEW. Поэтому в ней специально даются самые легкие для понимания примеры и конкретные приемы, которые используются в дальнейшем при сложных программах.

Как правило, примеры не имеют самостоятельного значения, а служат лишь для демонстрации возможностей LabVIEW. Программирование в среде LabVIEW и использование технологии виртуальных приборов в учебном процессе являются актуальными, поскольку создание виртуальных приборов практически полностью заменяет реальные физические стенды.

Компьютер становится комплексным информационно - измерительным устройством, обеспечивающий выполнение всех основных функций автоматизированной системы, в том числе взаимодействие с измерительной и управляющей аппаратурой, и совместимость программного обеспечения на разных уровнях. При создании программ стремятся к тому, чтобы все эти элементы соответствовали аналогичным по назначению элементам, расположенным на лицевой панели традиционных измерительных приборов.

LabVIEW ВП находят применение в самых разнообразных сферах человеческой деятельности: в исследовательских лабораториях, лабораториях фундаментальной науки, на промышленных предприятиях. Все более широко LabVIEW применяется в образовании при создании вузовских лабораторных практикумов (особенно по естественно-научным и общетехническим дисциплинам).

В данной работе созданы несколько ВП, которые изучают закон Ома, логические элементы, влияние температуры на ВАХ p-n перехода, зависимость сопротивления металлов от температуры. Кроме того, создан виртуальная модель цифрового вольтметра, который работает вместе с реальными приборами.

ВВЕДЕНИЕ

В настоящее время к подготовке бакалавров, инженеров и магистров предъявляются высокие требования в области автоматизации измерений, контроля и испытаний. Одним из обязательных элементов освоения данной предметной области является приобретение навыков практического использования систем автоматического проектирования различных компьютерных контрольно-измерительных систем.

Язык LabVIEW (Laboratory Virtual Instrument Engineering Workbench) разработан фирмой «National Instruments». Фирмой выпускаются разнообразные интерфейсные устройства, встраиваемые в компьютер или подключаемые к его портам, устройства генерации и обработки реальных электрических сигналов, датчики, регистрирующие различные физические процессы, и т.п.

LabVIEW позволяет разрабатывать прикладное программное обеспечение для организации взаимодействия с измерительной и управляющей аппаратурой, сбора, обработки и отображения информации и результатов расчетов, а также моделирования как отдельных объектов, так и автоматизированных систем в целом.

В отличие от текстовых языков, таких как C, Pascal и др., где программы составляются в виде строк текста, в LabVIEW программы создаются в виде графических диаграмм, подобных обычным блок-схемам.

LabVIEW – графическая система программирования на уровне функциональных блок-диаграмм, позволяющая графически объединять программные модули в виртуальные инструменты.

С его помощью создается не программа, как мы привыкли ее представлять, а виртуальный инструмент, предназначенный не только для моделирования тех или иных процессов, но и для управления

аппаратными средствами и исследования реальных физических объектов. Простота образных графических конструкций, наглядность и читаемость готовых программ заставляют отдать предпочтение языку LabVIEW перед другими. Он подобен таким системам, как C++ или Basic, однако в отличие от них оперирует не кодовыми строками, а блоками диаграмм, что делает программирование более простым и понятным.

Программа, написанная в среде LabVIEW, называется виртуальным прибором (ВП). ВП симулируют реальные физические приборы. LabVIEW содержит полный набор инструментов для сбора, анализа, представления и хранения данных.

Запуск среды программирования LabVIEW осуществляется либо двойным кликом мыши на ярлыке LabVIEW, который находится на рабочем столе, либо из раздела Пуск-Программы – National Instruments LabVIEW. При входе в главное меню LabVIEW пользователю предлагается создание нового виртуального инструмента (*New VI*) или открытие уже существующе- го (*Open VI*).

ВП состоит из четырех основных компонентов – *лицевой панели*, *блок-диаграммы*, *иконки и соединительной панели*.

Разработка VI (ВП) осуществляется на двух панелях, находящихся в двух окнах, – передней (лицевая панель) и функциональной (блок-диаграмма). Лицевая панель − интерфейс пользователя создается с использованием палитры Элементов (*Controls*). Эти элементы могут быть либо средствами ввода данных – *элементы управления*, либо средствами отображения данных – *элементы отображения*. *Элементы управления* – кнопки, переключатели, ползунки и другие элементы ввода. *Элементы отображения* – графики, цифровые табло, светодиоды и т.д.

Общая характеристика работы

Актуальность темы.

Программирование в среде LabVIEW и использование технологии виртуальных приборов в учебном процессе являются актуальными, поскольку создание виртуальных приборов практически полностью заменяет реальные физические стенды.

Компьютер становится комплексным информационно - измерительным устройством, обеспечивающий выполнение всех основных функций автоматизированной системы, в том числе взаимодействие с измерительной и управляющей аппаратурой, и совместимость программного обеспечения на разных уровнях.

Цель диссертационной работы.

Основной целью данной магистерской работы является изучение приемов программирования на LabVIEW и создание ВП. Поэтому в ней специально даются самые легкие для понимания примеры и конкретные приемы, которые используются в дальнейшем при сложных программах.

Как правило, примеры не имеют самостоятельного значения, а служат лишь для демонстрации возможностей LabVIEW.

Публикация. По теме диссертации опубликовано 3 работы.

Объем и структура работы. Диссертация состоит из введения, трех глав и заключения. Общий объем работы 78 страниц машинописного текста, включающего рисунки, таблицы и список литературы.

ГЛАВА 1. ВВЕДЕНИЕ В LabVIEW. ОСНОВНЫЕ ПОНЯТИЯ И ПРАВИЛА ПОСТРОЕНИЯ ВИРТУАЛЬНЫХ ПРИБОРОВ

1.1 Общие сведения о LabVIEW.

В настоящее время все более увеличивается значение компьютерных технологий при создании информационно- измерительных систем. Это связано с большими возможностями компьютеров по оперативной обработке результатов измерений, накоплению данных, обеспечению широкой перестройки параметров системы в процессе работы и так далее.

Для создания прикладного программного обеспечения компьютерных систем сбора и обработки измерительной информации сегодня применяются специализированные средства, использующие принцип объектно-ориентированного программирования. Среди таких специализированных средств наиболее развитой и универсальной является среда графического программирования LabVIEW фирмы National Instruments. Эта фирма является основоположником данного направления в проектировании программных продуктов (первая версия LabVIEW появилась в 1986 г., а сегодня доступна уже одиннадцатая версия разработки). Представленные на рынке аппаратные средства систем сбора и обработки измерительной информации почти всегда комплектуются драйверами под LabVIEW. Данное обстоятельство является очень важным, так как для разработки эффективных драйверов от программиста требуется высокая квалификация.

Библиотеки LabVIEW для ввода-вывода, обработки и отображения экспериментальных данных весьма обширны. Блоки ввода-вывода обеспечивают работу более чем с десятью стандартными интерфейсами, блоки обработки измерительной информации позволяют выполнять все

типовые виды обработки экспериментальных данных, блоки отображения представляют данные практически во всех известных формах. Очень важна на практике поддерживаемая LabVIEW возможность создания независимых приложений, способных функционировать под управлением типовых операционных систем.

LabVIEW (Laboratory Virtual Instrument Engineering Workbench) представляет собой среду графического программирования, предназначенную для создания прикладного программного обеспечения информационно-измерительных систем, а также различных компьютерных систем сбора и обработки экспериментальных данных. LabVIEW прикладные программы могут использоваться на средах графического программирования LabVIEW платформе ПК различных типов и способны функционировать под управлением практически всех известных к настоящему времени операционных систем ПК.

Созданную в среде LabVIEW прикладную программу принято называть **Виртуальным прибором** (ВП). Для человека, ясно осознающего содержание измерительной задачи, создание таких ВП является довольно несложным делом, а интуитивно понятный пользовательский интерфейс программной среды делает разработку программ и их применение весьма интересным и увлекательным занятием. Имена файлов программ, созданных в LabVIEW, имеют расширение VI (от Virtual Instruments). Если файл содержит несколько программ (библиотеку) виртуальных приборов, то он имеет расширение LLB [1].

В состав LabVIEW прикладной программы входят две основные составляющие:

• лицевая панель виртуального прибора (Front Panel);

• функциональная панель или диаграмма (Diagram).

При работе с готовыми ВП используются только лицевые панели. Диаграмма нужна исключительно для разработки ВП. Лицевая панель

определяет внешний вид ВП и интерфейс взаимодействия пользователя с прибором. Она содержит различные элементы ввода и управления (выключатели, переключатели, поля ввода и т. д.) и элементы вывода (цифровые индикаторы, графические экраны и т. д.). При создании программ стремятся к тому, чтобы все эти элементы соответствовали аналогичным по назначению элементам, расположенным на лицевой панели традиционных измерительных приборов.

Программа LabVIEW может иметь как модульную, так и иерархическую структуру, а сложный ВП может содержать в своем составе более простые ВП.

Концепция LabVIEW существенно отличается от системы традиционных языков программирования. Разработчику ВП предоставляется графическая оболочка, включающая в себя весь набор инструментов, необходимых для сбора данных, их анализа и представления полученных результатов. Создаваемая в процессе программирования диаграмма напоминает привычную для инженера блок-схему. Поэтому разработчик даже при отсутствии навыков программирования затратит на решение своих задач в LabVIEW значительно меньше времени и усилий, по сравнению с написанием программ по традиционной технологии.

В связи с развитием сетевых технологий LabVIEW ВП стали использоваться и в виртуальном пространстве. Все большее число разработчиков создают ВП, предусматривающие удаленное управление и наблюдение за процессами по Internet сетям. Особенно удобно использовать технологию ВП при создании измерительных систем, снабженных такими аппаратными средствами, как встраиваемые в ПК платы аналогового и цифрового ввода-вывода, платы захвата видео установка среды LabVIEW изображения и управления движением, средства измерений, оснащенные стандартными интерфейсами: GPIB

(КОП), PXI, VXI, RS-232/485, USB и т. д.

Интерфейс панели LabVIEW и окно редактирования диаграмм

Запустите LabVIEW. В появившемся окне (рис.1.1) выберите пункт New>Blank VI.

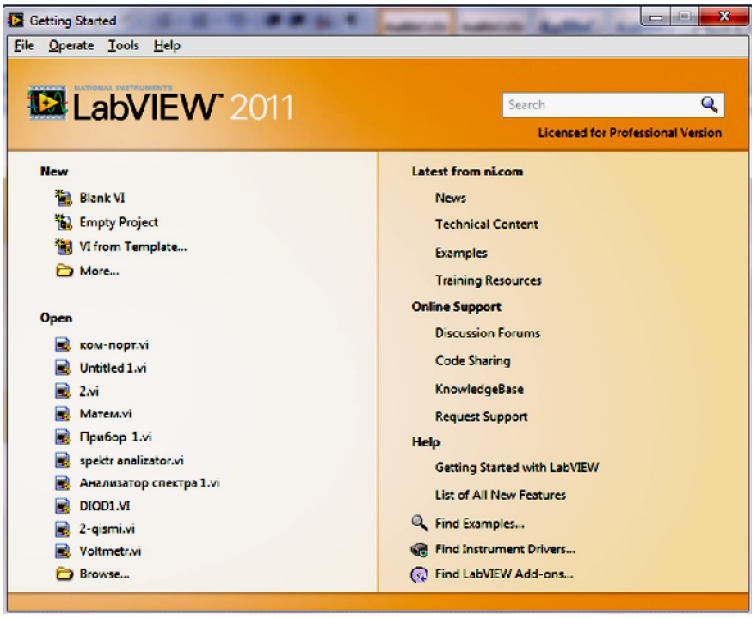


Рис. 1.1. Окно запуска программы.

После выбора создания нового прибора раскрываются два окна: интерфейсная панель (Front Panel) (рис.1.2) и окно редактирования диаграмм (Block Diagram) (рис.1.3), которое по своей сути является программой в графическом виде.

Интерфейсная панель - это интерфейс пользователя. Вы устанавливаете на интерфейсную панель графические элементы управления и всевозможные индикаторные приборы, которые являются соответственно элементами ввода и вывода. Элементы управления - это ручки, регуляторы, ползунковые устройства, кнопки и другие устройства

ввода. Индикаторы - это элементы для вывода/построения графиков, сигнализирующие устройства, такие, как лампочки и т.д.

Установленные на переднюю панель элементы управления и индикаторы, отображаются соответствующими иконками (терминалами) во втором окне – **окне редактирования диаграмм**. Т.е. каждому установленному элементу на интерфейсной панели соответствует иконка в окне редактирования. В этом окне и "пишется" программа - создается графический код VI. При удалении, например, управляющего элемента в интерфейсном окне, исчезнет и соответствующая иконка (терминал) в окне редактирования диаграмм.

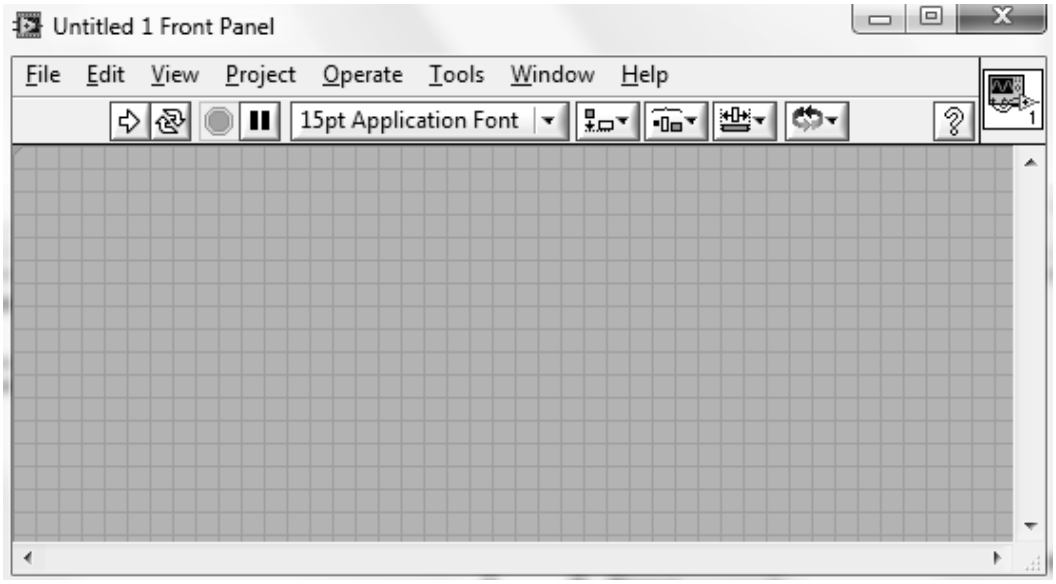


Рис. 1.2. Интерфейсная панель.

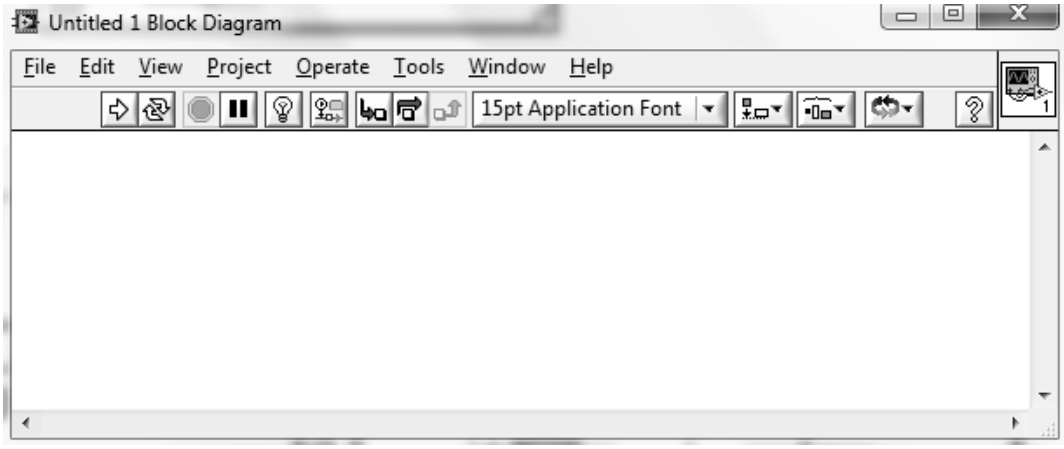


Рис. 1.3. Окно редактирования диаграмм.

Иконки или терминалы показывают тип данных элементов управления или индикаторов. Терминалы осуществляют связь между интерфейсной панелью и диаграммой.

Функции - это объекты окна редактирования диаграмм, которые могут иметь один и/или несколько входов и/или выходов. Функции LabVIEW аналогичны выражениям, операторам, процедурам и функциям текстовых языков программирования.

Связи — это соединительные линии между иконками (терминалами). Они являются аналогом переменных в обычных языках программирования. Причем данные могут передаваться только в одном направлении - от терминала-источника к одному или нескольким терминалам-приемникам. Различный вид и цвет соединений соответствует различным типам передаваемых данных. Неправильная связь терминалов или незаконченное соединение изображается штриховой линией.

Структуры - это графическое представление циклов и операторов выбора в тексториентированных языках программирования. Терминалы, функции, связи и структуры - это весь синтаксис языка программирования LabVIEW.

Линейка инструментов. Оба окна, как интерфейсное, так и окно редактирования диаграмм имеют линейки инструментов, которые содержат служебные кнопки и индикаторы состояния, предназначенные

для контроля Виртуальных Инструментов. Одна из линеек инструментов всегда доступна, и ее вид зависит от того, в каком окне Вы находитесь. Линейка инструментов интерфейсного окна содержит 8 кнопок, рис. 1.4.

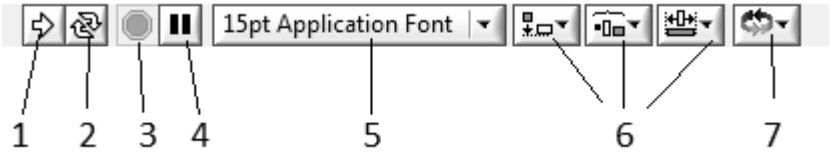


Рис. 1.4. Линейка инструментов интерфейсного окна

- 1 - кнопка запуска программы на выполнение, пока приложение выполняется, значок меняет свой внешний вид;
- 2 - кнопка запуска программы на выполнение в циклическом режиме, пока приложение выполняется, значок меняет свой внешний вид;
- 3 - когда приложение запущено, эта кнопка находится в активном состоянии, используйте ее для прекращения выполнения программы;
- 4 - кнопка "ПАУЗА" приостанавливает исполнение программы до последующего нажатия на эту же кнопку;
- 5 - выпадающее меню редактирования свойств шрифта: тип, размер, стиль и цвет;
- 6 - выпадающие меню, позволяющие осуществлять выравнивание и позиционирование объектов, распределять графические объекты передней панели VI.
- 7 - если вы устанавливаете объект поверх другого объекта, то нижний объект может быть перекрыт и недоступен, используйте это выпадающее меню размещения объекта поверх или под желаемым.

Если приложение не может быть запущено на выполнение по какой-либо причине, то линейка инструментов примет следующий вид:

Рис. 1.5. Линейка инструментов – программа не может быть запущена

Линейка инструментов окна редактирования диаграмм имеет такие же кнопки, как и интерфейсная панель, плюс свои собственные, предназначенные для отладки (debugging) программы, рис. 1.6:

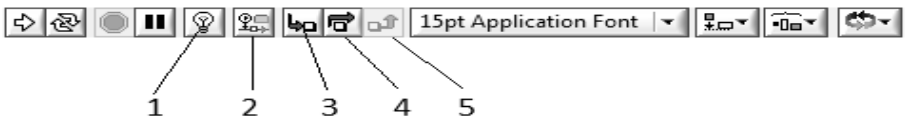


Рис. 1.6. Линейка инструментов окна редактирования диаграмм

- 1 - нажав эту кнопку, и запустив программу на выполнение, Вы сможете отлаживать программу и следить за следованием данных между объектами;
- 2 – включение или отключение сохранения значений связей;
- 3 - нажатие на эту кнопку позволяет Вам в процессе отладки "входить" в структуры, например, в циклы и подпрограммы;
- 4 - нажав эту кнопку, Вы активизируете пошаговый режим отладки, т.е. данные будут передаваться от иконки к иконке, останавливая программу, подсвечивая при этом текущий элемент, и ожидать следующего шага – нажатия на эту же кнопку;
- 5 - нажатие на эту кнопку позволяет Вам в процессе отладки "выходить" из структуры, например, из цикла, и перейти к следующему "узлу".

Выпадающее меню. Установив элемент управления на интерфейсную панель или иконку в окне редактирования диаграмм, мы всегда можем, например, изменить свойства этого объекта. Для этого нужно подвести указатель мыши к желаемому объекту и нажать правую

кнопку мыши. В результате появится выпадающее меню, позволяющее редактировать объект.

Один из основных служебных компонентов, который позволяет создавать, модифицировать и отлаживать виртуальные инструменты – это Панель Инструментов (Tools), рис. 1.7.

Рис.1.7. Панель инструментов

Вызывается из основного меню **View > Tools Palette**.

Operate Value - управление объектами, например, для перемещения ползунка или поворота ручки;

Position/Size/Select - установка, "перетягивание", перемещение, изменение размеров объекта – стрелочка меняет свой внешний вид, когда доступна опция изменения размеров объекта;

Edit text- инструмент для ввода текста, как в окне интерфейсной части программы, так и в окне редактирования диаграмм;

Connect Wire - используется для связи (соединения) объектов;

Object Shortcut Menu - инструмент выполняет аналогичную функцию, что и нажатие на правую кнопку мыши на желаемом объекте, т.е. служит для вызова выпадающего меню;

Scroll Window – перемещение (прокрутка) содержимого окна одновременно;

Set/Clear Breakpoint - установка точек (меток) останова программы при ее отладке;

Probe Data - устанавка пробника в окне редактирования диаграмм, который, во время выполнения или отладки программы показывает значение параметра, где установлен пробник;

Get Color - копирование цвета в палитру - необходимо подвести указатель и нажать левую кнопку мыши;

Set Color - установка цвета объекта и его фона [14].

Панель Управления и Функциональная панель представляют собой структурированный набор иконных меню, предназначенных для доступа к библиотекам элементов интерфейса и соответствующих функций. Вызов необходимой панели осуществляется автоматически, при переключении между окном редактирования и интерфейсной панелью.

Используя **Панель Управления,** Вы можете устанавливать (добавлять) элементы управления и индикаторы. Каждая опция иконизированного меню содержит подменю, в котором находятся соответствующие объекты. Вызывается из основного меню **View>Controls Palette**, рис. 1.8.

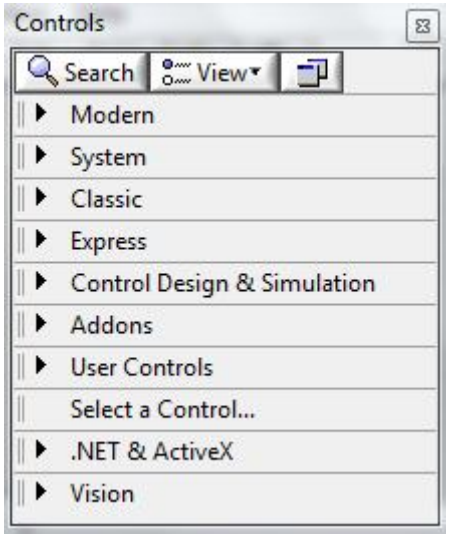


Рис. 1.8. Панель управления

Элементы Функциональной Панели используются для создания диаграмм, т.е. алгоритма работы VI. Функциональная панель содержит необходимые функции для работы с различными типами и структурами данных, и позволяет реализовывать

алгоритмы любой сложности от простых арифметических вычислений до функционально сложных, таких, например, как спектральный анализ. Вызывается из основного меню **View>Show Functions Palette**, рис. 1.9.

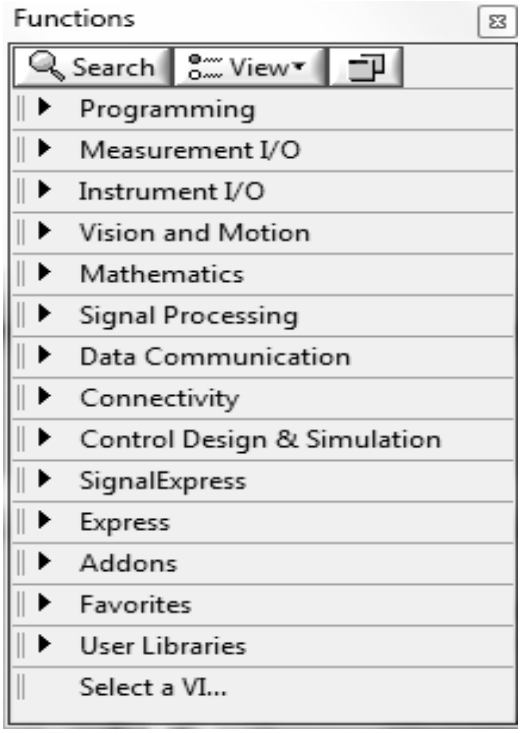


Рис. 1.9. Функциональная панель

Рис. 1.10. Панель Controls

- Array & Cluster (массивы и кластеры). Состоит из элементов управления и элементов отображения для группировки наборов типов данных;
- Graph (виртуальные осциллографы). Состоит из элементов отображения для построения графиков данных в графах или диаграммах в реальном масштабе времени;
- Path & Refnum (пути и ссылки). Состоит из элементов управления

и элементов отображения для путей и ссылок;

• Decorations (оформление). Состоит из элементов управления и элементов отображения графических объектов для настройки дисплеев лицевой панели;

• Select Control (выбор регулятора). Отображает диалоговое окно для загрузки самодельных элементов управления;

• User Controls (средства управления пользователя). Состоит из специальных средств управления, которые формирует сам пользователь;

• ActiveX (объекты ActiveX). Состоит из средств управления, позволяющих внедрить объекты ActiveX на лицевую панель;

• Dialog (диалоговая панель). Состоит из стандартных объектов для формирования диалога с пользователем;

• IMAQ Vision (обработка изображений). Состоит из средств обработки и анализа изображений;

• Internet Toolkit (работа с Internet). Состоит из средств управления, располагаемых на передней панели, позволяющих организовывать работу виртуальных инструментов в сети Internet (http, электронная почта, telnet, CGI и другие).

После помещения *элементов управления* или *отображения* данных на *лицевую панель* они получают свое графическое отображение (в виде терминала данных) на блок-диаграмме. Символы на терминале соответствуют типу данных терминала. Например, *DBL* – терминал представляет данные в виде вещественных чисел с двойной точностью, *TF* – логический терминал, *I16* – терминал 16-битных целых и др.

При активировании функциональной панели становится доступной палитра *Functions* (рис.1.9), которая аналогично панели *Controls* включает систематизированные наборы стандартных элементов в виде отдельных пиктограмм, из которых осуществляется составление блок-схемы ВП. Палитра *Functions* вызывается либо щелчком правой кнопки мыши в рабочем пространстве блок- схемы, либо путем выбора в пункте главного меню *Window → Show Function Palette*.

Рассмотрим основные подпанели панели Functions:

 Structures (структуры). Состоит из управляющих структур программы, таких как циклы For Loop, While Loop и др.;

 Numeric (числовые функции). Состоит из тригонометрических, логарифмических и других функций;

 Boolean (булевы функции). Состоит из логических и булевых функций;

 String (строковые функции). Состоит из функций для работы со строковыми величинами;

 Array (массивы). Состоит из функций для обработки массивов;

 Cluster (кластеры). Состоит из функций для обработки кластеров;

- *Comparison* (сравнение). Состоит из функций для сравнения переменных;
- *Time & Dialog* (время и диалог). Состоит из функций для диалоговых окон, синхронизации и обработки ошибок;
- *File I/O* (ввода/вывода файла). Состоит из функций для осуществления операций по вводу/выводу файлов;
- *Instrument I/O* (инструменты ввода/вывода). Состоит из ВП для связи и управления приборами различной архитектуры;
- *Instrument Drivers* (драйверы приборов). Состоит из ВП, способных управлять внешними приборами, осциллоскопами, генераторами и т.д., через IB;
- *Data Acquisition* (сбор данных). Состоит из ВП для использования плат сбора данных;
- *Signal Processing* (обработка сигналов). Состоит из ВП для генерации и обработки сигналов;
- *Mathematics* (математические). Состоит из оптимизационных, алгебраических, интегральных, дифференциальных и других функций;
- *Graphics & Sound* (графика и звук). Состоит из ВП для работы с трехмерной графикой, изображениями и звуком;
- *Communication* (связи). Состоит из виртуальных приборов для работы с сетями TCP, DDE и др.;
- *Application Control* (управление приложением). Состоит из ВП, управляющих виртуальными приборами;
- *Advanced* (расширенная). Состоит из разных функций типа функции библиотечного запроса, манипуляции данных и др [1].

1.2. Краткая история развития высокоуровневого программирования LabVIEW.

В 1983 году компания National Instruments начала поиски способов сокращения времени, необходимого для программирования измерительных систем. В результате появилась концепция виртуального прибора LabVIEW - сочетания интуитивного пользовательского интерфейса лицевой панели с передовой методикой блок-диаграммного программирования, позволяющего создавать эффективные измерительные системы на основе графического программного обеспечения.

Первая версия LabVIEW увидела свет в 1986 году. Она была предназначена только для компьютеров Macintosh. Несмотря на то что Macintosh довольно редко использовались в задачах измерений и автоматизации, графическая оболочка их операционной системы MacOS наилучшим образом соответствовала технологии LabVIEW. Довольно скоро и другие наиболее распространенные операционные системы перешли на графический пользовательский интерфейс и начали поддерживать эту технологию.

К 1990 году разработчики National Instruments полностью переделали LabVIEW, сочетая новые компьютерные технологии с анализом отзывов пользователей.

И, что более важно, вторая версия LabVIEW включала компилятор, делающий скорость исполнения виртуальных приборов сравнимой со скоростью выполнения программ, созданных на языке программирования С. Патентное ведомство США (United States Patent Office) выдало National Instruments несколько патентов, признающих новизну технологии LabVIEW. С появлением новых графических операционных систем, подобных MacOS, компания National Instruments перенесла ставшую уже признанной технологию LabVIEW на другие платформы - персональные компьютеры и рабочие станции. В 1992 году благодаря новой открытой архитектуре появились версии LabVIEW для Windows и Sun.

Третья версия LabVIEW появилась в 1993 году сразу для трех операционных систем: Macintosh, Windows и Sun. Виртуальные приборы версии 3, разработанные на одной из платформ, могли без изменений запускаться на другой. Эта межплатформенная совместимость дала пользователям возможность выбора платформы разработки и уверенность, что созданные ВП будут функционировать и на других платформах (обратите внимание, что это было реализовано за пару лет до появления Java). В 1994 году список платформ, поддерживающих LabVIEW,

увеличился и стал включать Windows NT, Power Macs, рабочие станции Hewlett Packard и в 1995 году - Windows 95.

LabVIEW 4 была выпущена в 1996 году и обеспечивала большую гибкость оболочки среды разработки, которая позволила пользователям создавать программы, подходящие типу их деятельности, уровню опыта и навыкам разработки. Кроме этого, LabVIEW 4 включала в себя мощные инструменты редактирования и отладки более совершенных измерительных систем, в том числе обмен данными на основе OLE-технологии и распределенных средств исполнения.

Версии LabVIEW 5 и 5.1 (в 1999 году) продолжают наращивать возможности системы: появляется встроенный Internet-сервер, подсистема динамического программирования и управления (сервер виртуального прибора), интеграция с ActiveX и особый протокол для упрощения обмена данными через Internet - DataSocket. Также введена долгожданная возможность отката действий пользователя (undo), которая уже присутствовала в большинстве компьютерных программ.

Вышедшей в 2000 году новой версии - LabVIEW 6 (известной также как 6.1) сделали «подтяжку лица»: в нее встроили новый комплект объемных элементов управления и индикации, поскольку в то время компьютерная индустрия обнаружила, что внешний вид программного продукта имеет весьма серьезное значение (чему способствовало появление систем Apple iMac и G4). В LabVIEW 6 воплотилась очень серьезная работа по обеспечению как простого и интуитивного интерфейса среды программирования (особенно для непрограммистов!), так и поддержки множества передовых технологий программирования, например объектно - ориентированного программирования, многопоточности (multithreading), распределенных вычислений (distributed computing) и т.д. И пусть простота графической оболочки LabVIEW не вводит вас в заблуждение: LabVIEW - это инструмент, который является

достойным соперником систем программирования C++ или Visual Basic, да еще и превосходит их в удобстве работы, как отмечают тысячи пользователей. В версии LabVIEW 6.1, вышедшей в 2001 году, было введено событийно-управляемое (event-oriented) программирование, удаленное управление LabVIEW через Internet и другие улучшения.

Совершенно особой разновидностью LabVIEW, на которую следует обратить внимание, является LabVIEW RT. RT означает Real Time - реальное время. LabVIEW RT представляет собой совокупность аппаратного и программного обеспечения, которая позволяет выделять части кода LabVIEW и загружать их для выполнения на отдельном контроллере, работающем под управлением собственной операционной системы реального времени. Таким образом гарантируется, что выделенные участки LabVIEW-приложения будут выполняться в точно определенные моменты времени, даже если Windows «зависнет» и компьютер перестанет работать.

LabVIEW является мощным инструментом программирования, пригодным для решения практически любых задач, например компьютерного моделирования, тем не менее он чаще всего используется для сбора экспериментальных данных и управления приборами и установками, и поэтому содержит множество виртуальных приборов, разработанных специально для этой цели. Например, LabVIEW может управлять встраиваемыми многофункциональными устройствами сбора данных (plug-in data acquisition - DAQ), которые предназначены для ввода и/или вывода аналоговых и цифровых сигналов. Например, вы можете совместно использовать многофункциональные платы и LabVIEW для мониторинга температуры, формирования управляющих сигналов для экспериментальной установки или определения частоты неизвестного сигнала. LabVIEW также обеспечивает передачу команд и данных по каналу общего пользования (КОП) или через стандартный

последовательный порт компьютера. Канал общего пользования часто применяется для взаимодействия с осциллографами, сканерами и мультиметрами, а также для дистанционного управления подобными приборами. С помощью программного обеспечения LabVIEW допустимо управлять сложными измерительными системами стандарта VXI, приборами с сетевым интерфейсом Ethernet или через порт USB.

Получив со встроенной платы или внешнего прибора массив данных, вы можете использовать множество содержащихся в LabVIEW виртуальных приборов анализа для всесторонней обработки этих данных и их преобразования. Часто полезен обмен данными не только с измерительными приборами, но и с другими программными продуктами и удаленными компьютерами!. В LabVIEW встроены функции, которые упрощают этот процесс, поддерживая несколько сетевых протоколов, вызов внешнего программного кода или динамических библиотек (DLL) и автоматизацию ActiveX.

1.3. Преимущества концепции графического программирования NI LabVIEW.

Преимущества этого языка проявляются при выполнении следующих задач:

- исследование возможности контроля измерительной системой параметров объекта измерений;
- встроенные шаблоны и готовые примеры проектов;
- Он-лайн доступ к курсам по основам программирования в LabVIEW;
- высокая стабильность работы;
- новые средства для высокопроизводительного анализа и обработки изображений;
- высокая производительность за счет обратной связи с сообществом программистов;
- мобильные приложения для отображения и управления данными на iPad.
- анализ и оптимизация структуры ККИС;

- расчет и анализ статистических и динамических характеристик и ее компонентов.

Более 20 лет инженеры и ученые используют NI LabVIEW для разработки измерительных систем, испытательных стендов и систем управления. В основе LabVIEW лежит графический язык программирования G. Помимо самой возможности программирования среда LabVIEW предоставляет в распоряжение пользователя широкий спектр инструментов и библиотек: от интерактивных мастеров настройки и пользовательских интерфейсов до встроенных компилятора, компоновщика и средств отладки [2].

LabVIEW: графическое потоковое программирование. Существует два основных отличия LabVIEW от других языков программирования. Во-первых, LabVIEW реализует концепцию графического программирования G, поэтому исходный код представляет собой блок-диаграмму (соединенные друг с другом пиктограммы элементов языка), которая затем компилируется в машинный код. Несмотря на такой подход в языке G используются те же конструкции и методы программирования, что и в других языках: типы данных, циклы, переменные, рекурсия, обработка событий и объектно-ориентированное программирование.

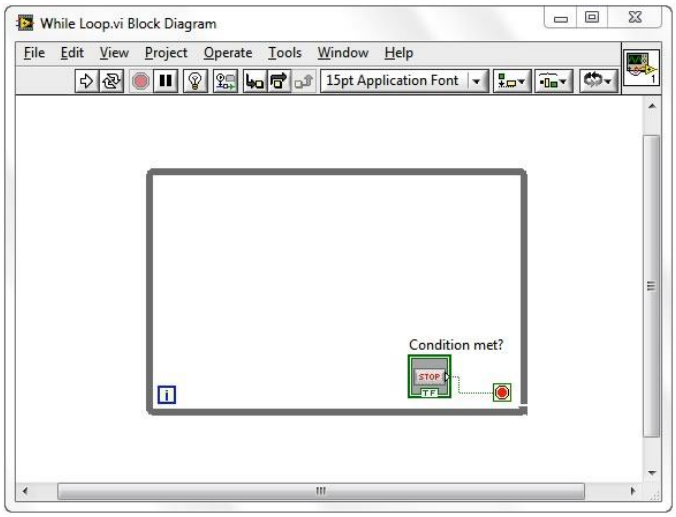


Рис.1.11. *Условный цикл (цикл While) представлен в языке G в виде интуитивно понятной структуры, которая циклически выполняется до тех пор, пока не будет достигнуто условие выхода из цикла*

Вторая отличительная особенность LabVIEW - это поддержка выполнения кода, написанного на языке G, в режиме потока данных (потоковое программирование), в то время как традиционные текстовые языки (например, C и C++) обеспечивают выполнение кода в виде последовательности команд. В основе языков потокового программирования (таких как G, Agilent VEE, Microsoft Visual Programming Language и Apple Quartz Composer) лежит концепция потока данных, который и определяет последовательность выполнения функциональных узлов программы.

Поначалу может показаться, что отличие подобного подхода от традиционного не существенно, однако на практике оказывается иначе. А именно, потоковое программирование в среде LabVIEW позволяет разработчику полностью сфокусироваться на данных и путях их обработки. Узлы программы - функции, циклы и прочие конструкции языка - получают данные через входы, производят их обработку и выводят данные с помощью выходов. Как только значения параметров поступают на каждый из входных терминалов узла, происходит выполнение кода узла (обработка поступивших данных), после чего значения выходных параметров оказываются доступными на выходных терминалах узла для дальнейшей их передачи на другие узлы согласно логике потока данных. Соответственно, из двух последовательно связанных узлов, второй сможет быть выполнен только после получения данных от предыдущего.

Интуитивное использование средств графического языка. Как и большинство людей, многие инженеры и ученые решают поставленные перед собой задачи, оперируя образами или символами. Подобный поход

развивается в процессе обучения и применения соответствующих инструментов обработки информации – различных схем и диаграмм. Однако большинство языков программирования требуют изучения специфического синтаксиса и адаптации моделей прикладной области к возможностям языка. В тоже время, графический язык G позволяет работать с интуитивно понятными структурами.

Код языка G обычно удобнее для инженеров и ученых, потому что они привыкли к визуальной работе с данными, моделированию процессов с помощью блок-схем и диаграмм состояний, которые также отражают потоки данных. Помимо этого, потоковое программирование обусловливает необходимость работать в терминологическом поле прикладной области задачи. Например, типичное приложение на языке G сперва получает данные с нескольких каналов датчиков температуры, затем передает данные функции, выполняющей анализ, и, наконец, сохраняет данные на диск. Графическое представление программы наглядно демонстрирует порядок выполнение операций и потоки данных.

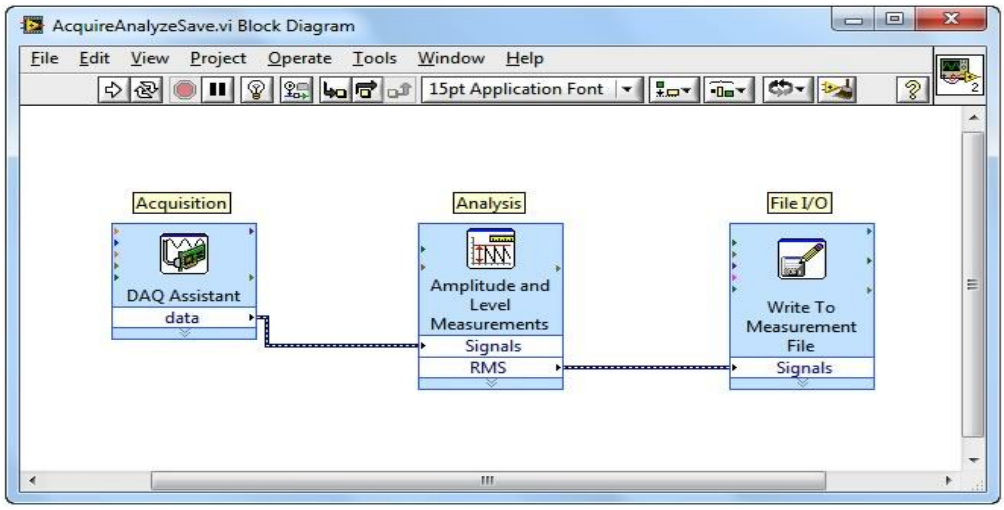


Рис.1.12. *Данные появляются в результате работы функции сбора данных, а затем передаются узлу анализа и, следом за ним, узлу сохранения результатов в файл***.**

Интерактивные средства отладки. Поскольку концепция языка G проста для понимания, LabVIEW предоставляет в распоряжение

пользователя столь же удобные и интуитивно понятные инструменты среды разработки. Например, уникальные средства отладки позволяют наглядно отобразить процесс распространения данных по проводникам, а также отобразить соответствующие значения на входах и выходах узлов кода (речь идет об анимации выполнения).

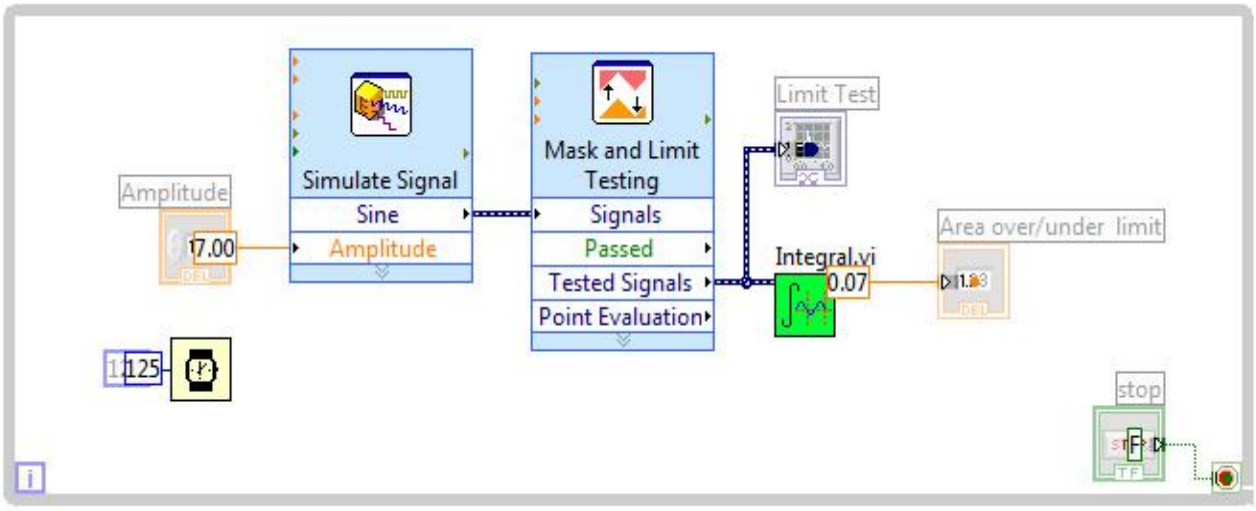


Рис.1.13. *Анимация выполнения наглядно демонстрирует последовательность выполнения кода G.*

Кроме того, LabVIEW предоставляет разработчику набор инструментов отладки, аналогичных имеющимся в других средах разработки. С помощью пиктограмм на инструментальной панели блок-диаграммы вы можете запустить пошаговое выполнение кода, установить точки останова и включить анимацию выполнения.

Рис. 1.14. *На инструментальной панели блок-диаграммы расположены пиктограммы стандартных отладочных средств (например, пошаговое выполнение кода).*

Отладочные средства позволяют установить пробники одновременно на многих участках программы, приостановить выполнение и выполнить вход в подпрограмму. Подобный функционал есть и в других средах разработки, однако LabVIEW, благодаря графической сути языка G, в более удобной форме отображает текущее состояние программы и взаимосвязи параллельных участков кода.

Интеграция языка G с другими языками. Несмотря на то, что язык G идеально подходит для организации распараллеливания кода, а также скрывает тонкости управления памятью, он не является наиболее оптимальным выбором для решения некоторых задач. В частности, математические формулы и уравнения могут быть более наглядно представлены в текстовом виде, поэтому в LabVIEW реализованы механизмы добавления на блок-диаграмму текстового кода.

Например, в LabVIEW есть узел, называющийся Formula Node, который позволяет рассчитывать значения по текстовым формулам и выполнять текстовые программы с С-подобным синтаксисом.

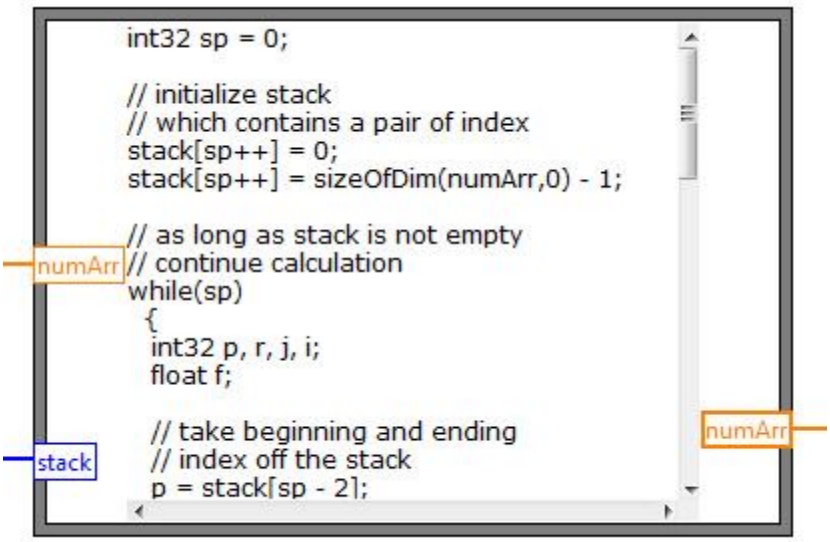


Рис.1.15. *Узел Formula Node позволяет внедрить в программу компактный текстовый код, используя С-подобный синтаксис для математических операций*

Аналогично, узел MathScript Node позволяет встраивать в программу на LabVIEW код **.m** файлов.

– Графический компилятор.

Во многих приложениях скорость выполнения является критичной. LabVIEW – единственная графическая среда программирования с компилятором, который генерирует оптимизированный код. Скорость выполнения LabVIEW близка к скорости выполнения компилированных Си программ. Поэтому, используя данный графический язык, вы можете увеличить свою производительность при создании программ без снижения скорости их выполнения.

Готовые виртуальные инструменты (VI) работают в системе разработчика LabVIEW, а также в LabVIEW **Run-Time System**. Это компактная, недорогая версия LabVIEW может только загружать и запускать VI, но не позволяет редактировать или показывать их диаграмму. Это свойство защищает исходный код Вашего VI. Вы можете использовать Run-Time System как дешевую тестовую станцию или эффективный путь для распространения собственных разработок.

С помощью дополнительной программы **Application Builder** выполняется преобразование VI в обычную исполняемую *.exe программу, которая запускается и выполняется самостоятельно, как любая Windows программа.

– Гибкость.

LabVIEW – открытая среда, которая позволяет легко интегрировать в систему ваши собственные программные и аппаратные разработки. Для включения объектного Си-кода в программу LabVIEW воспользуйтесь 32-х разрядным WATCOM C компилятором. Кроме того, LabVIEW предоставляет доступ к стандартным 16-ти битным библиотекам DLL MS Windows.

– Библиотеки анализа.

Analyses VI Libraries включают статистику, решение уравнений, регрессионный анализ, линейную алгебру, алгоритмы генерации сигналов, анализ в частотной и временной области, процедуры спектрального анализа и цифровые фильтры. Используя эти библиотеки, вы сможете разработать виртуальные инструменты для управления процессами, цифровой обработки сигналов и многих других приложений.

1.4. Инструкции по установке LabVIEW.

Если вы располагаете полной версией LabVIEW и испытываете трудности по ее установке, вы можете изучить документацию, идущую в комплекте с программным продуктом, либо обратиться в службу технической поддержки National Instruments.

Если к настоящему моменту у вас нет LabVIEW, используйте демонстрационную версию LabVIEW, которая содержится на компакт-диске, сопровождающем эту книгу. Срок действия демо-версии - 30 дней.

Для работы с упражнениями вам понадобится каталог Everyone с компакт-диска. Лучше всего скопировать содержимое этого каталога на жесткий диск вашего компьютера с тем, чтобы иметь возможность сохранять результаты самостоятельной работы [5].

Минимальные требования к системе для работы со средой LabVIEW. Основной исполняемый модуль демо-версии LabVIEW такой же, как и у полной версии, так что системные требования практически одинаковы.

Windows XP/2000/NT/Me/9x

- при использовании Windows NT 4.0 обязательно наличие Service Pack 3 или выше;
- минимум 32 Мб ОЗУ (рекомендуется 64 Мб);
- 65 Мб свободного дискового пространства при минимальной установке

LabVIEW, 200 Мб при полной установке;

• процессор класса Pentium-166 и выше.

MacOS

MacOS версии 7.6.1 или выше;

32 Мб ОЗУ (рекомендуется 64 Мб);

100 Мб свободного дискового пространства при минимальной установке LabVIEW, 225 Мб при полной установке^;

процессор PowerPC.

Linux

MacOS версии 7.6.1 или выше;

32 Мб ОЗУ (рекомендуется 64 Мб);

100 Мб свободного дискового пространства при минимальной установке LabVIEW, 225 Мб при полной установке;

процессор PowerPC.

Linux-ядро версии 2.0.x или выше;

дистрибутив Linux с библиотекой GNU C Library версии 2.0.5 или выше (glibc2 или libc.so.6), например:

- RedHat Linux 5.0 или выше;
- SuSE Linux 6.0 или выше;
- SuSE Linux 5.3 с установленной библиотекой shlibs6-98.9.25-0 RPM;
- Caldera Open Linux 1.3 или выше;
- Debian Linux 2.0 или выше;
- 32 Мб ОЗУ (рекомендуется 64 Мб);
- размер своп-файла 32 Мб;
- 65 Мб свободного дискового пространства при минимальной установке LabVIEW, 150 Мб при полной установке;

- сервер XWindows System;
- процессор Pentium-166 или выше.
- Solaris версии 2.5.1 или выше;

- ОЗУ, своп-файл, диск: то же;
- сервер XWindows System;
- процессор SPARC (станция Sun SPARC).

HP-UX

• HP-UX версии 10.20 или выше;

• ОЗУ, своп-файл, диск: то же;

• сервер XWindows System;

• процессор PA-RISC (станция Hewlett Packard 9000 Series 700).

Общие требования

LabVIEW использует специальный каталог для хранения временных файлов (temporary files). Некоторые из этих файлов могут иметь достаточно большой размер, поэтому следует зарезервировать еще несколько мегабайтов на жестком диске под каталог временных файлов. По умолчанию LabVIEW применяет в качестве такого каталога в Windows папку, заданную переменной окружения TEMP, а в MacOS создает временные файлы внутри системной Корзины (Trash Can)[9].

ГЛАВА 2. СОЗДАНИЕ, РЕДАКТИРОВАНИЕ И ОТЛАДКА ВИРТУАЛЬНОГО ПРИБОРА В СРЕДЕ LabVIEW

2.1. Лабораторная работа №1. Создание простейшего виртуального прибора.

В данном виртуальном приборе выполняются алгебраические функции. Уставки назначаются кнопками и индицируются шкальными индикаторами. В соответствующих терминалах получаются результаты выполнения операций.

На лицевой панели, как и положено, располагаются элементы управления программой — кнопки, графики, выключатели и тому

подобное. Блок-схема — это, по сути, и есть сама программа. При написании (а вернее создании, потому что писать приходится не так уж и много) программы используется такое понятие, как «поток данных» (Data Flow). Суть его в том, что все элементы программы (которые представлены графически) связываются между собой связями (проводами, нитками) по которым и происходит передача данных. В общем, описать это довольно сложно, лучше посмотреть на картинку, рис.1.

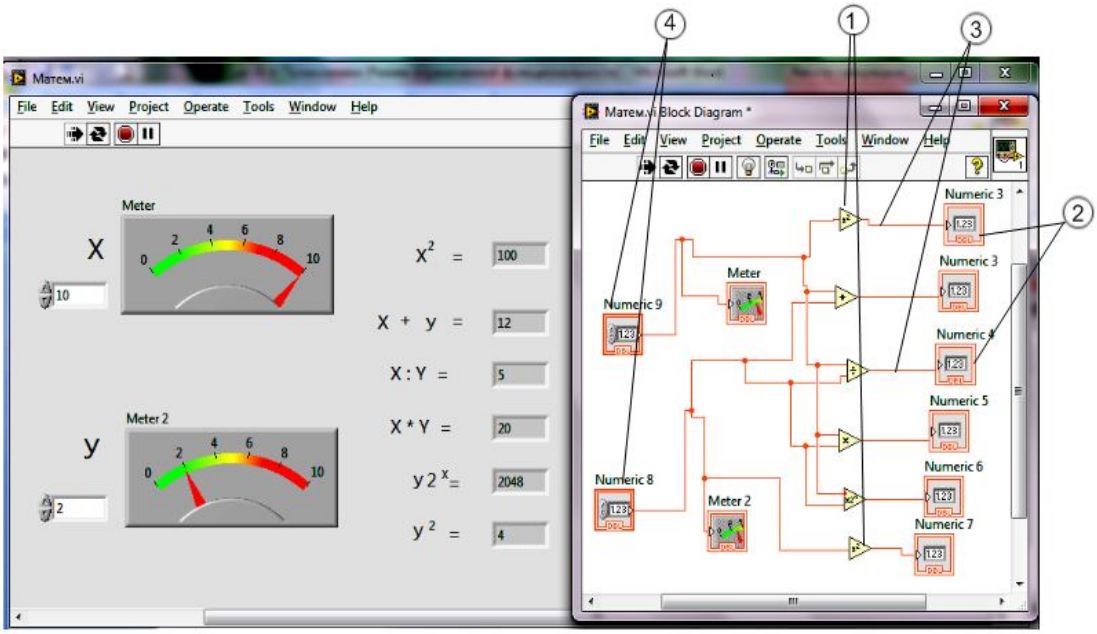


Рис. 2.1. Простейший ВП.

Цифрами обозначены:

- 1- точки, элементы программы (Nodes);
- 2 - терминалы индикаторов (Indicator Terminals);
- 3 - связи (Wires);
- 4 - терминалы управляющих элементов (Control Terminals)

Итак, в LabVIEW вы создаете пользовательский интерфейс (лицевую панель), с управляющими элементами и индикаторами. Управляющие элементы — это тумблеры, кнопки, поля ввода и прочие устройства ввода.

Индикаторы — это графики, шкалы, лампочки, текстовые поля и тому подобное. После создания пользовательского интерфейса, вы добавляете программный код, который управляет объектами на лицевой панели. Этот код содержится в схеме (block diagram). Этот код чем-то напоминает собой блок-схему, хотя отличий много[19].

2.2. Лабораторная работа №2. Виртуальный прибор преобразования °C в °F (градусы по Цельсию и по Фаренгейту).

Цель лабораторной работы:

- изучить компоненты ВП;
- создать ВП (преобразовать °C в °F);
- изучить типы данных и проводники данных;
- отредактировать ВП;
- приобрести практические навыки отладки ВП.

Объекты лицевой панели на блок-диаграмме отображаются в виде терминалов данных (графическое изображение прямоугольной формы с буквенно-численными обозначениями). Терминалы данных обеспечивают обмен данными между лицевой панелью и блок-диаграммой; они подобны переменным и константам текстовых языков программирования. Различают терминалы данных следующих типов – терминалы элементов управления и отображения данных, терминалы узлов.

Узлы – это объекты на блок-диаграмме, которые имеют одно или более полей ввода/вывода данных и выполняют алгоритмические операции ВП. Они аналогичны операторам, функциям и подпрограммам текстовых языков программирования. Узлы включают в себя функции, подпрограммы ВП и структуры. Подпрограмма ВП – виртуальный прибор, который можно использовать на блок-диаграмме другого ВП в

качестве подпрограммы. Структуры – это элементы управления процессом, такие как структура Case (Вариант), цикл While (цикл по условию) и т.д. Узлы Add (Сложение) и Subtract (Вычитание) – узлы функций.

Типы и проводники данных. В среде LabVIEW проводники данных используются для соединения многочисленных терминалов данных. Поля ввода/вывода должны быть совместимыми с типами данных, передаваемыми им по проводникам.

Выполнение лабораторной работы . Преобразование °C в °F.

Ниже приведена последовательность действий для создания ВП, который будет преобразовывать значение температуры из градусов Цельсия (°C) в температуру по Фаренгейту (°F).

Лицевая панель.

1. Выберите пункт главного меню File → New → VI, чтобы открыть новую лицевую панель.

2. Поместите цифровой элемент управления на лицевую панель. В поле собственной метки элемента управления напечатайте «Град. C».

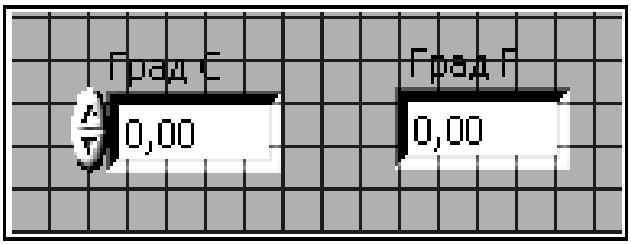


Рис. 2.2. ВП преобразование °C в °F (лицевая панель).

3. Поместите элемент отображения данных на лицевую панель. Он будет использован для отображения значений тем пературы в °F. В поле собственной метки элемента управления напечатайте «Град. F» и щелкните мышью в свободном пространстве лицевой панели или нажмите кнопку Enter. На блок-диаграмме LabVIEW создаст терминалы данных, соответствующие элементам управления и отображения. Терминалы данных представляют тип данных соответствующих элементов. Например,

терминал данных DBL представляет тип числовых данных двойной точности с плавающей запятой.

Терминалы данных, соответствующие элементам управления, имеют более широкий обводной контур по сравнению с терминалами данных, соответствующими элементам отображения[1].

Блок-диаграмма

4. Перейдите на блок-диаграмму, выбрав пункты главного меню Window → Show Diagram.

5. Создайте блок-диаграмму, показанную ниже:

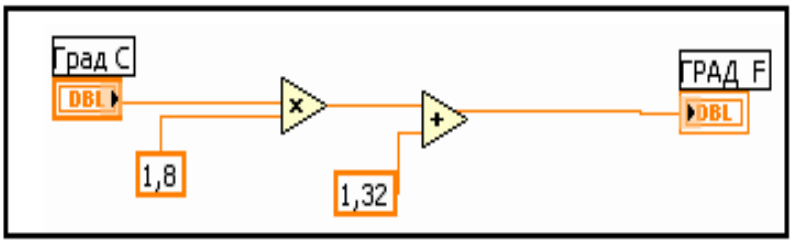


Рис. 2.3. ВП преобразование °C в °F (блок-диаграмма).

6. Выберите функцию Multiply (Умножение) из палитры Функций в разделе Functions → Numeric (Арифметические функции). Поместите ее на блок-диаграмму.

7. Выберите функцию Add (Сложение) из палитры Функций в разделе Functions → Numeric (Арифметические функции). Поместите ее на блок-диаграмму.

8. Выберите числовую константу из палитры Функций в разделе Functions → Numeric (Арифметические функции). Поместите две числовые константы на блок-диаграмму. После размещения числовой константы на блок-диаграмме поле ввода ее значений подсвечивается и готово для редактирования. Одной константе присвойте значение 1,8, другой 32,0.

9. Соедините объекты блок-диаграммы с помощью инструмента

10. Перейдите на лицевую панель, выбрав в главном меню пункт Window → Show Panel.

11. Сохраните ВП, он будет использоваться позднее.

Создание нового ВП на основе предыдущего. ВП преобразует по закону Ома значение напряжения и сопротивления в силу тока (I = U/R).

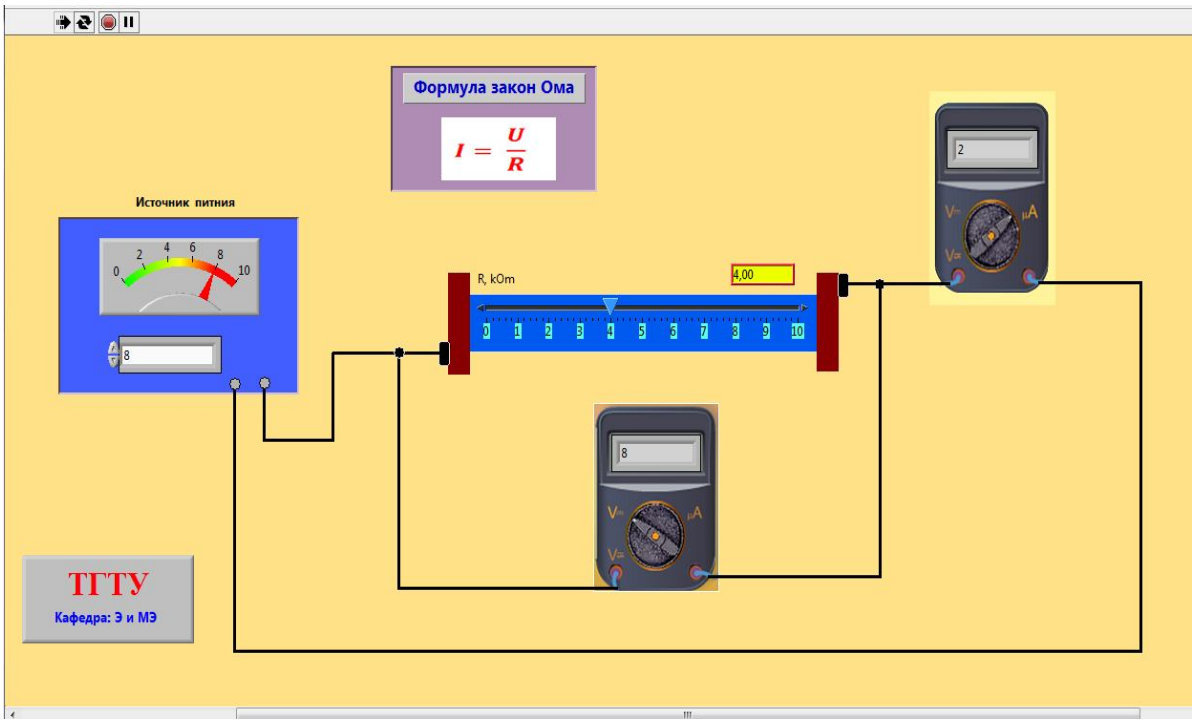


Рис. 2.4. Лицевая панель. Изучение закона Ома.

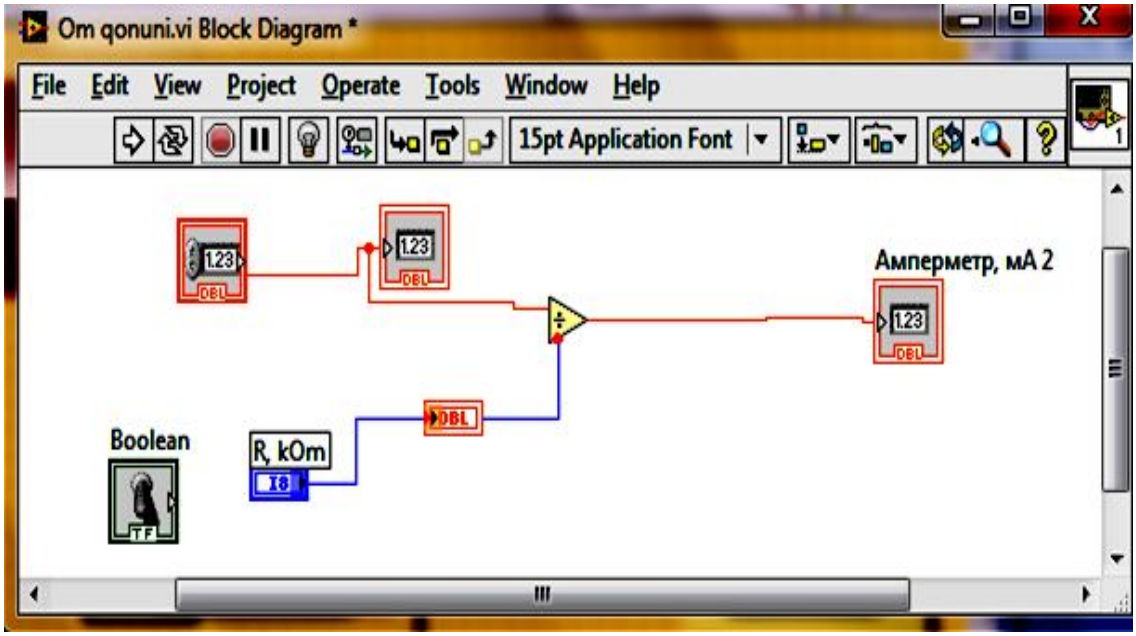


Рис. 2.5. Блок-диаграмма. Изучение закона Ома.

2.3. Лабораторная работа №3. Моделирование работы комбинационных цифровых устройств.

Основные логические функции

В отличие от переменной в обычной алгебре логическая переменная способна принимать только два дискретных значения. Их называют логическим нулем и логической единицей и обозначают символами «1» или «0».

Есть три основных типа связей между логическими переменными: конъюнкция, дизъюнкция и отрицание. Подобно алгебре чисел, используются следующие символы операций:

- конъюнкция: $y = x_1 \wedge x_2 = x_1 \cdot x_2 = x_1 x_2$,
- дизъюнкция: $y = x_1 \vee x_2 = x_1 + x_2$,
- отрицание: $y = \bar{x}$.

Триггера - это большой класс электронных устройств, обладающих способностью находиться в одном из двух устойчивых состояниях и чередовать их под воздействием внешних сигналов. Триггера это

элементы с памятью. Их состояние зависит не только от сигналов приложенных к входу в данный момент времени, но и от сигналов, воздействующих на него раньше.

В зависимости от свойств, числа и назначения входов триггеры делят на следующие виды:

1. RS-триггеры с раздельной установкой в 1 и 0.

2. D-триггеры (другие названия - триггер задержки, триггер данных)

3. Универсальные JK-триггера

4 Т-триггера (счетные триггеры)

Микросхемы К561ИЕ8- десятичные счетчики с дешифратором (рис. 2.6). Микросхемы имеют три входа - вход установки исходного состояния R, вход для подачи счетных импульсов отрицательной полярности CN и вход для подачи счетных импульсов положительной полярности CP. Установка счетчика в 0 происходит при подаче на вход R лог. 1, при этом на выходе 0 появляется лог. 1, на выходах 1-9 - лог. 0.

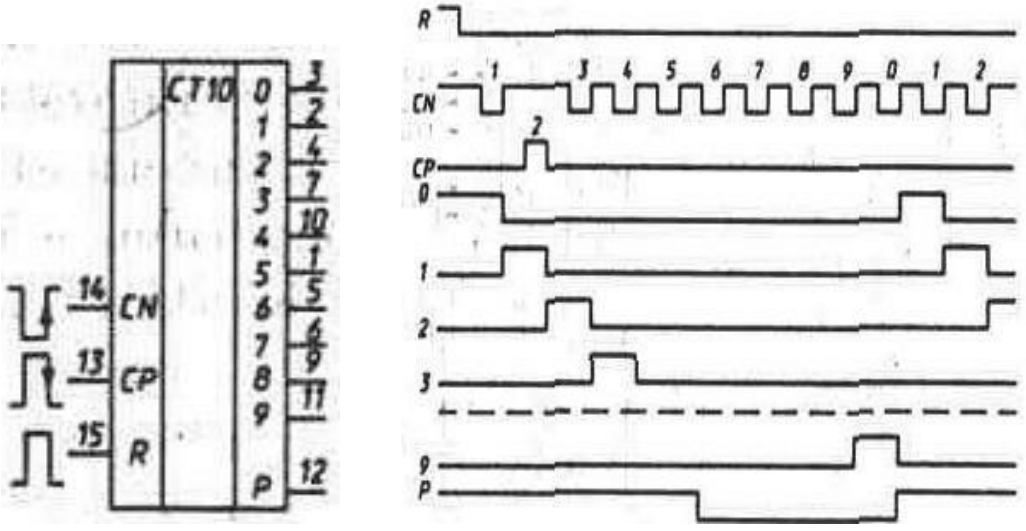
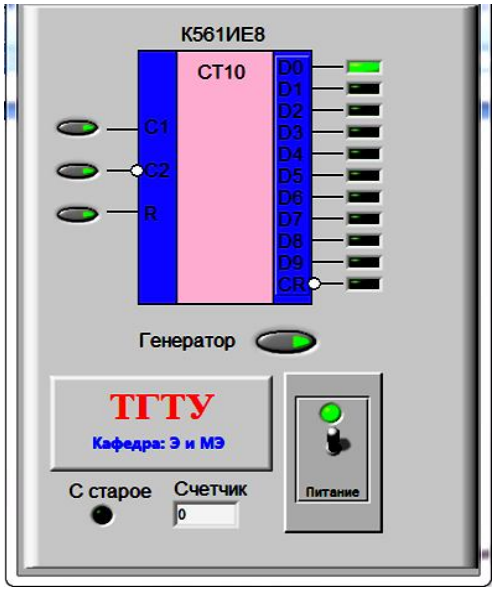
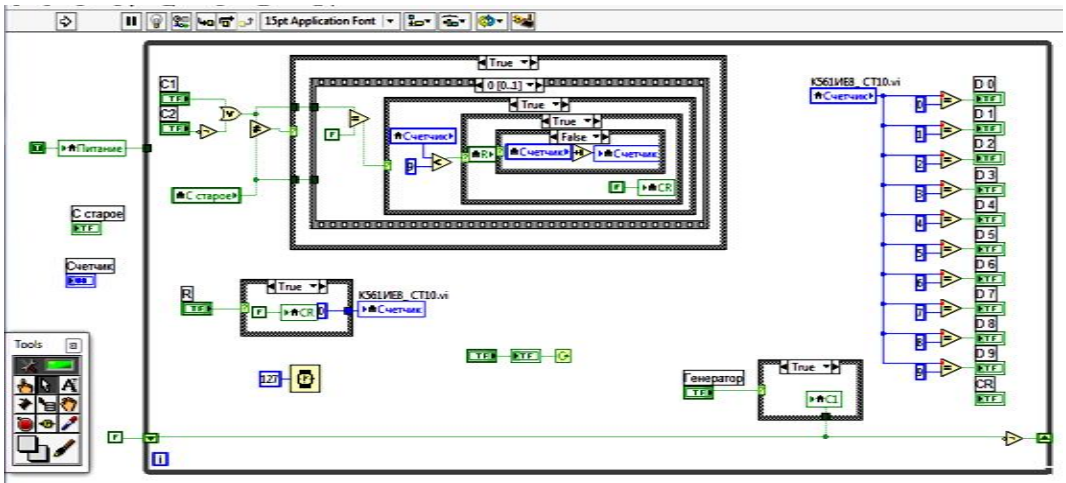


Рис. .2.6. Микросхема К561ИЕ8 и её диаграмма времени.



а)



б)

Рис.2.7. ВП «Микросхема К561ИЕ8». Лицевая панель(а) и блок-диаграмма(б).

Дешифраторы и преобразователи кодов

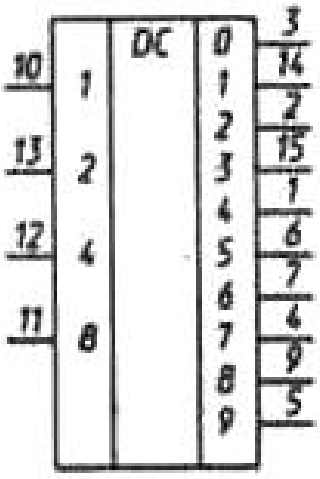


Рис. 2.8. Микросхема К561ИД1.

Микросхемы К176ИД1 и К561ИД1 (рис.2.8) -дешифраторы на 10 выходов. Микросхемы имеют 4 входа для подачи кода 1-2-4-8. Выходной сигнал лог. 1 появляется на том выходе дешифратора, номер которого соответствует десятичному эквиваленту входного кода, на остальных выходах дешифратора при этом лог. 0. При подаче на входы кодов, соответствующих десятичным числам, превышающим 9, активизируются выходы 8 или 9 в зависимости от сигнала, поданного на вход 1 - при лог. 0 на этом входе лог. 1 появляется на выходе 8, при лог. 1 - на выходе 9. Микросхемы не имеют специального входа стробирования, однако для построения дешифраторов с числом выходов более 10 можно использовать для стробирования вход 8 микросхем, так как выходной сигнал может появиться на выходах 0-7 лишь при лог.1.

а) б)

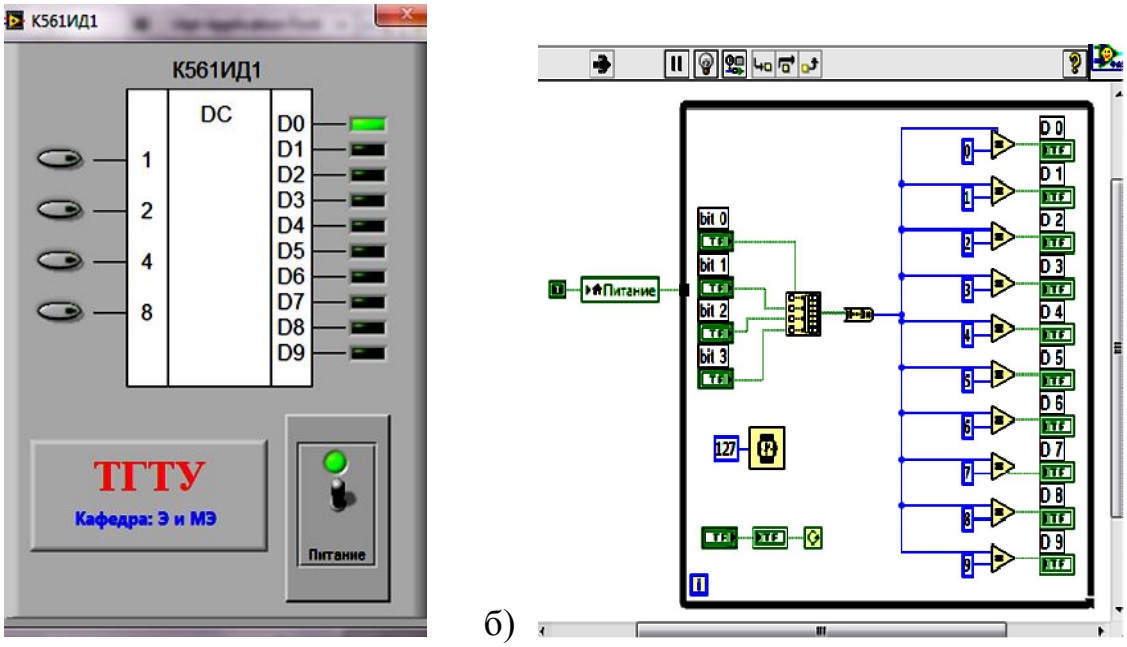
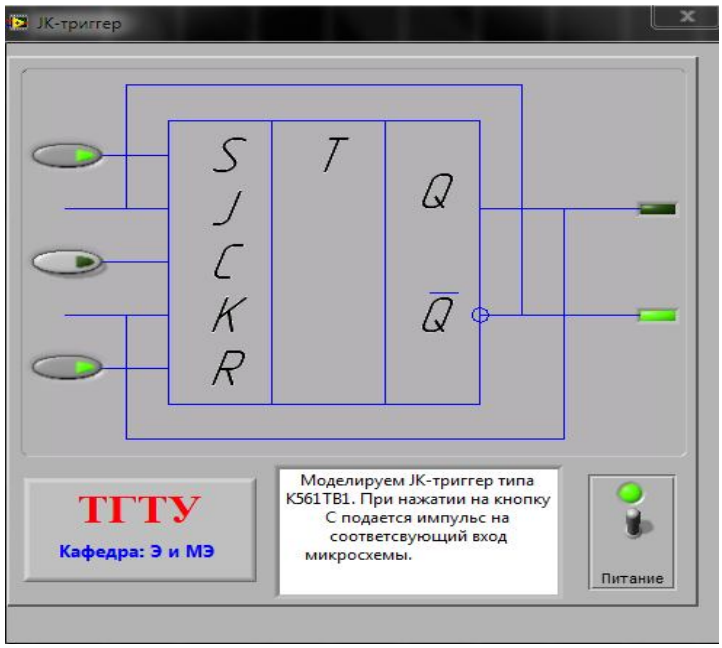


Рис.2.9. ВП «Микросхема К561ИД1». Лицевая панель(а) и блок-диаграмма(б).

УНИВЕРСАЛЬНЫЙ JK-ТРИГГЕР

JK-триггер имеет два информационных входа J и K, тактовый динамический вход, чаще инверсный, и два асинхронных входа установки и сброса.

а)



б)

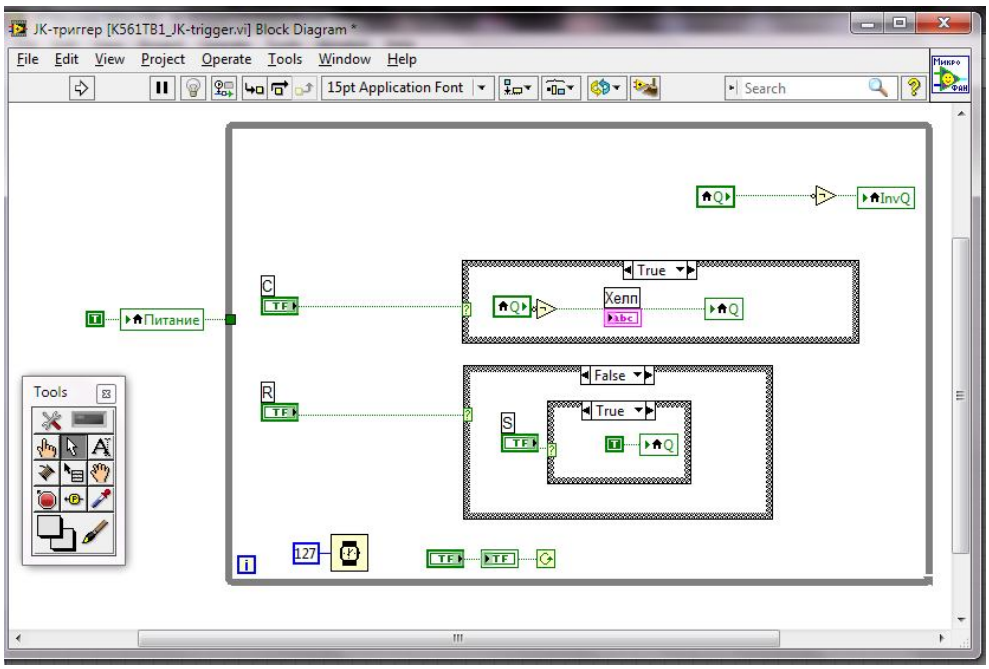


Рис.2.10. ВП «Микросхема К561ИЕ8». Лицевая панель(а) и блок-диаграмма(б).

Микросхема К561ЛЕ5 (К176ЛЕ5)

Микросхема — К561ЛЕ5 (К176ЛЕ5), которая содержит четыре элемента "2ИЛИ-НЕ". Вспомним чем отличаются эти элементы: если на, хотя бы один вход элемента И-НЕ поступает логический ноль, то на его выходе будет единица независимо от того что происходит на его остальных входах. То есть решающую роль играет ноль на входе.

Таким образом элементы И-НЕ и ИЛИ-НЕ работают по сходному принципу, но имеют противоположные функции[16].

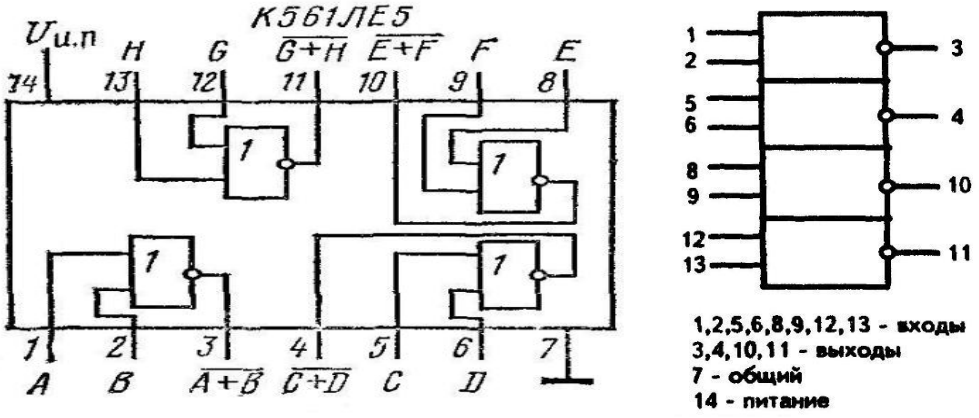
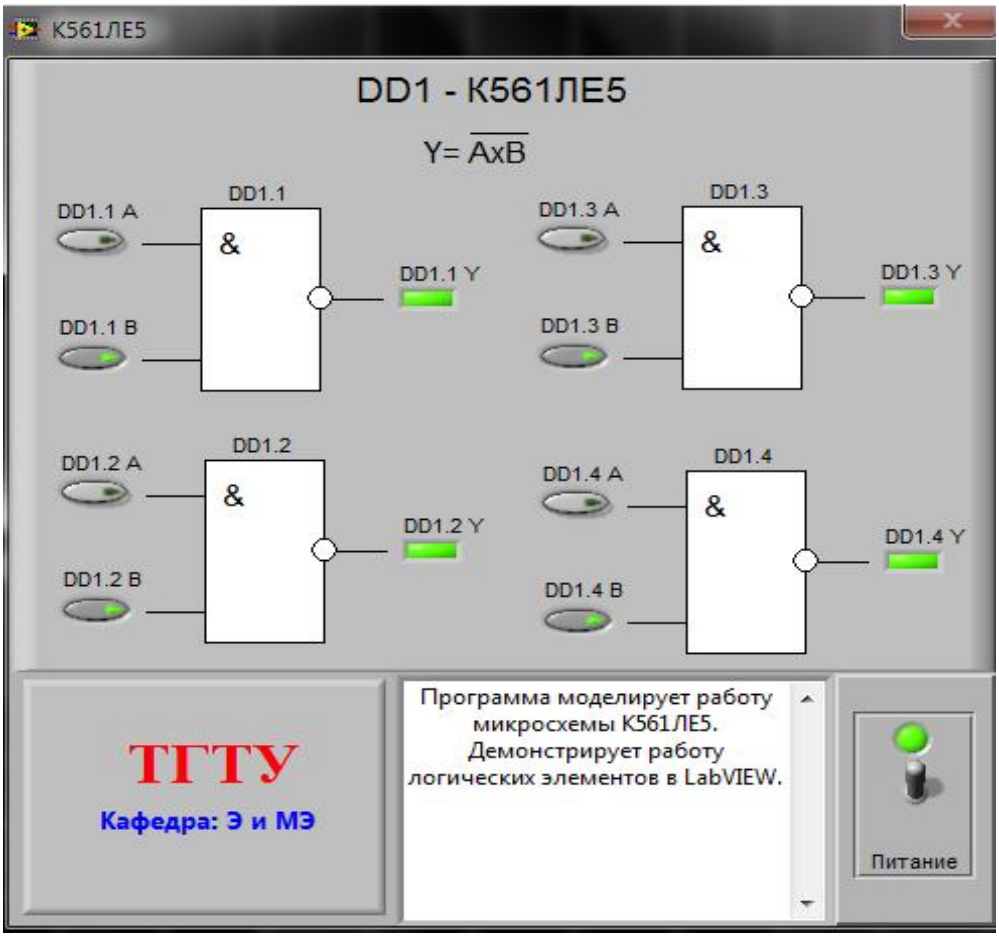


Рис. 2.11. Принципиальная микросхема К561ЛЕ5 (К176ЛЕ5).

а)



б)

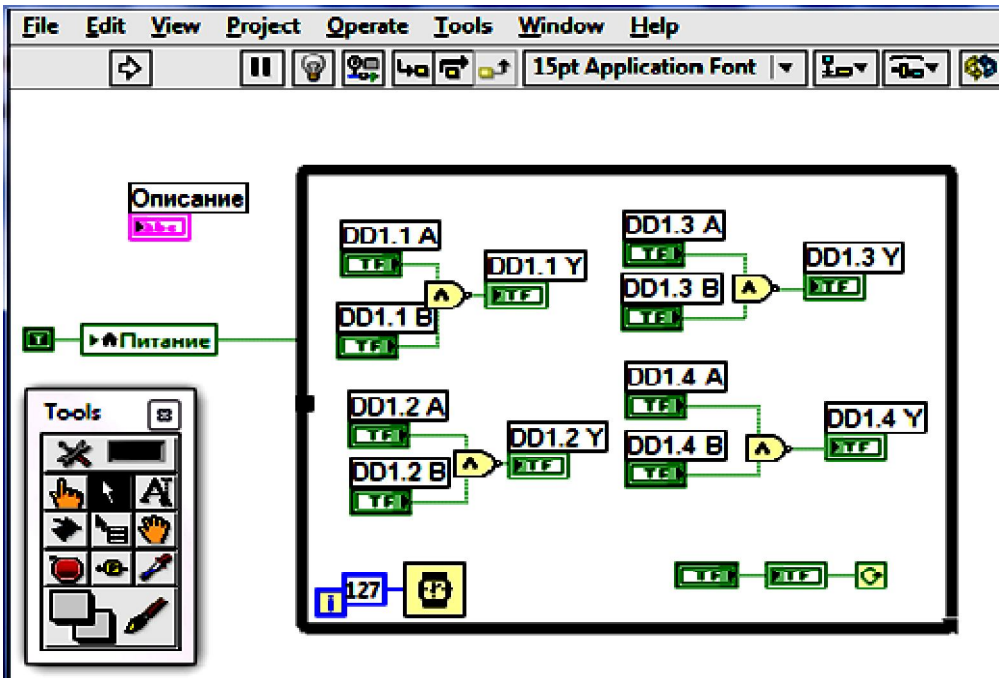
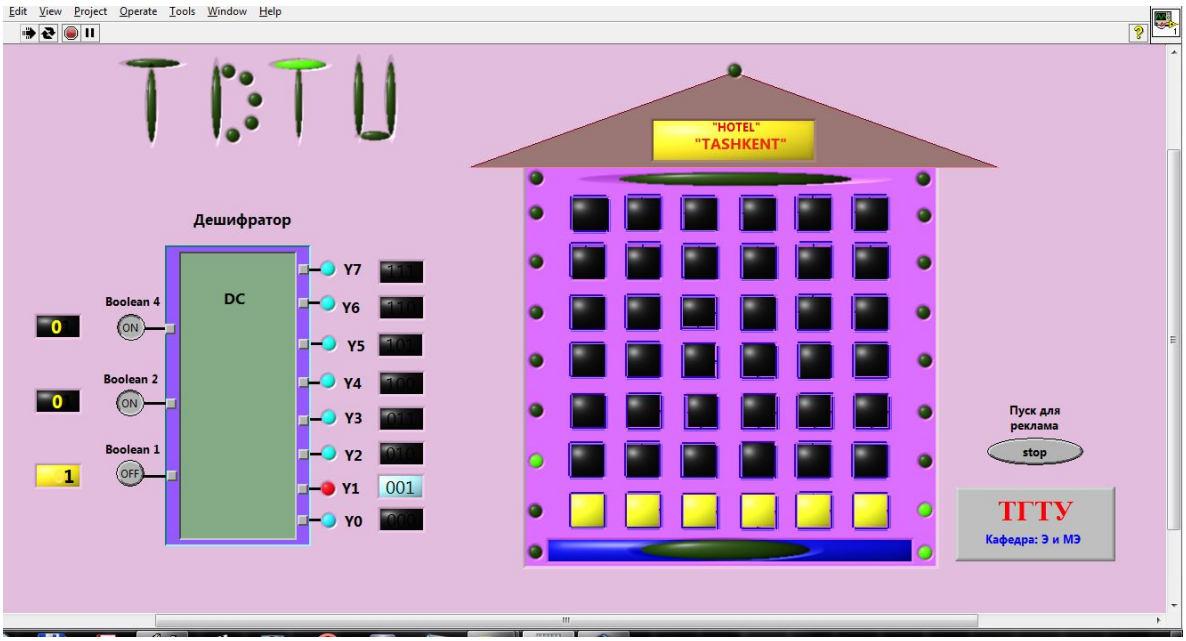
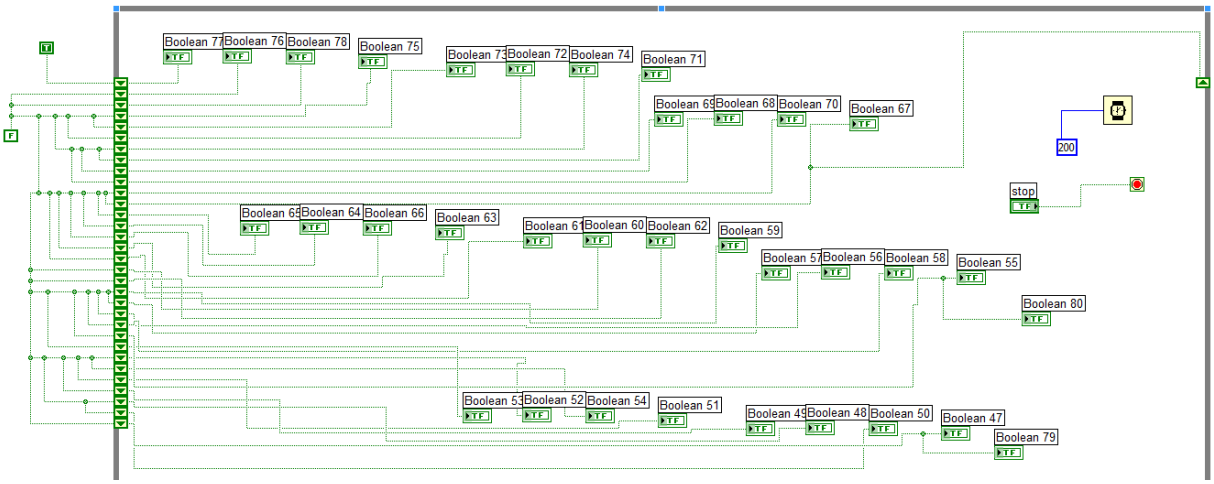


Рис.2.12. ВП «Микросхема К561ЛЕ5». Лицевая панель(а) и блок-диаграмма(б).

Дешифратор. Дешифратор (DC — DeCoder — декодер) — преобразователь n-разрядного двоичного кода в унитарный код «1 из m». Каждой кодовой комбинации на входах дешифратора соответствует активный уровень только на одном из выходов. Условное графическое обозначение и таблица истинности полного дешифратора на два входа (n = 2). Логическая 1 (при активном высоком уровне на выходе) формируется на том выходе дешифратора, адрес которого соответствует набору двоичных сигналов на входах A и B. Выходной код носит название «один из четырех». По таблице истинности легко записать в СДНФ логические функции, связывающие сигналы на каждом выходе дешифратора с его входными сигналами (они показаны на рисунке). Для реализации дешифратора требуются логические элементы И и НЕ[16].

а)





б)

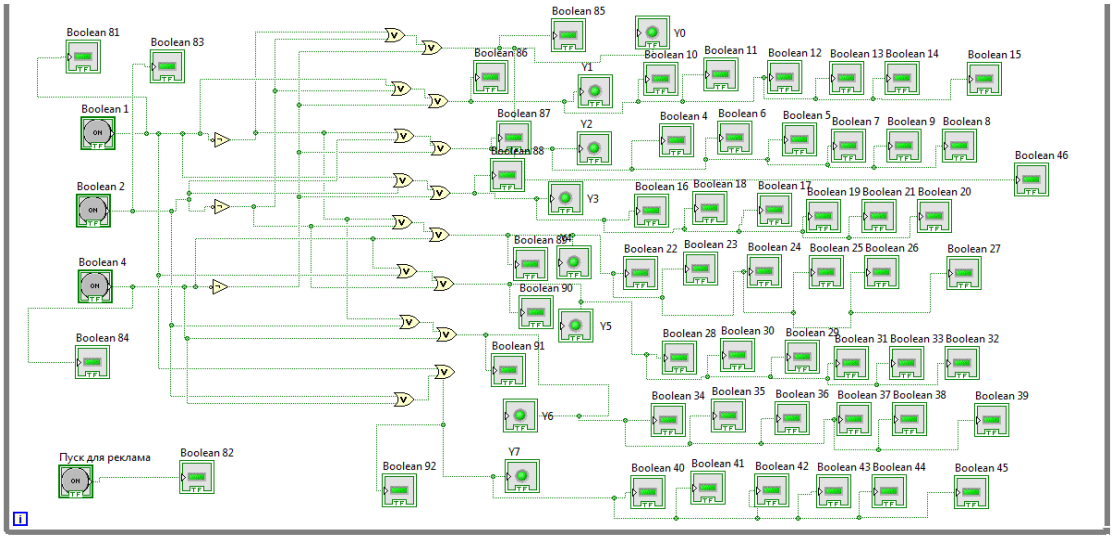


Рис.2.13. ВП «Дешифратор». Лицевая панель(а) и блок-диаграмма(б).

2.4. Лабораторная работа №4. Влияние температуры на вольт - амперную характеристику p-n перехода.

Цель работы – снятие вольт амперной характеристики p-n перехода при различных температурах.

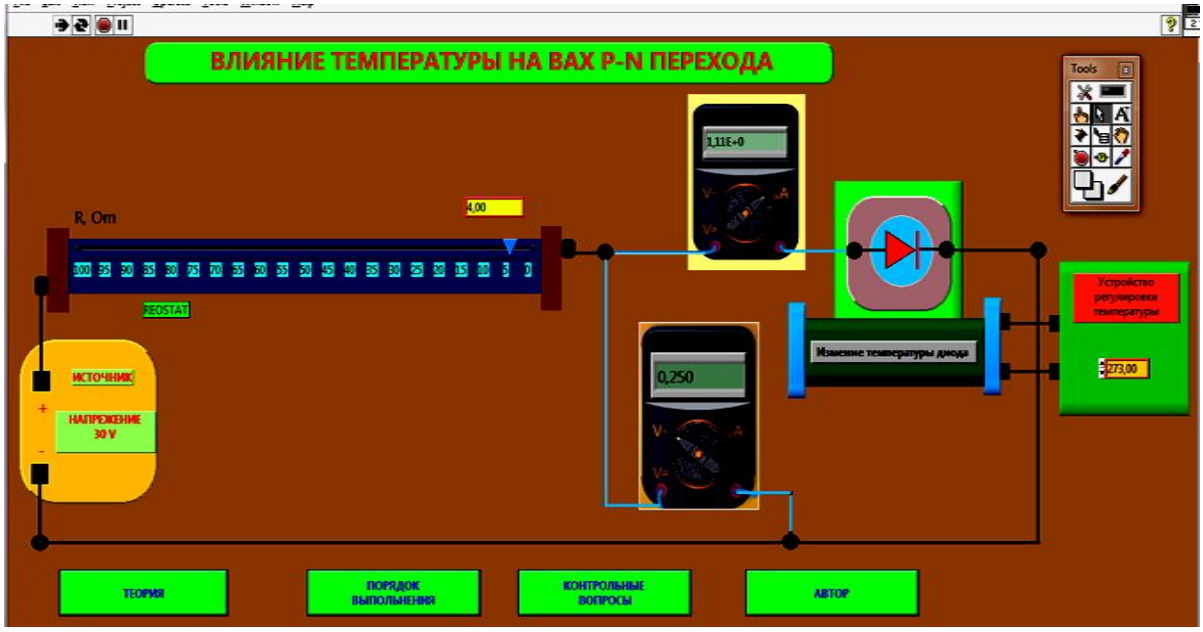


Рис.2.14. Внешний вид виртуальной установки (лицевая панель).

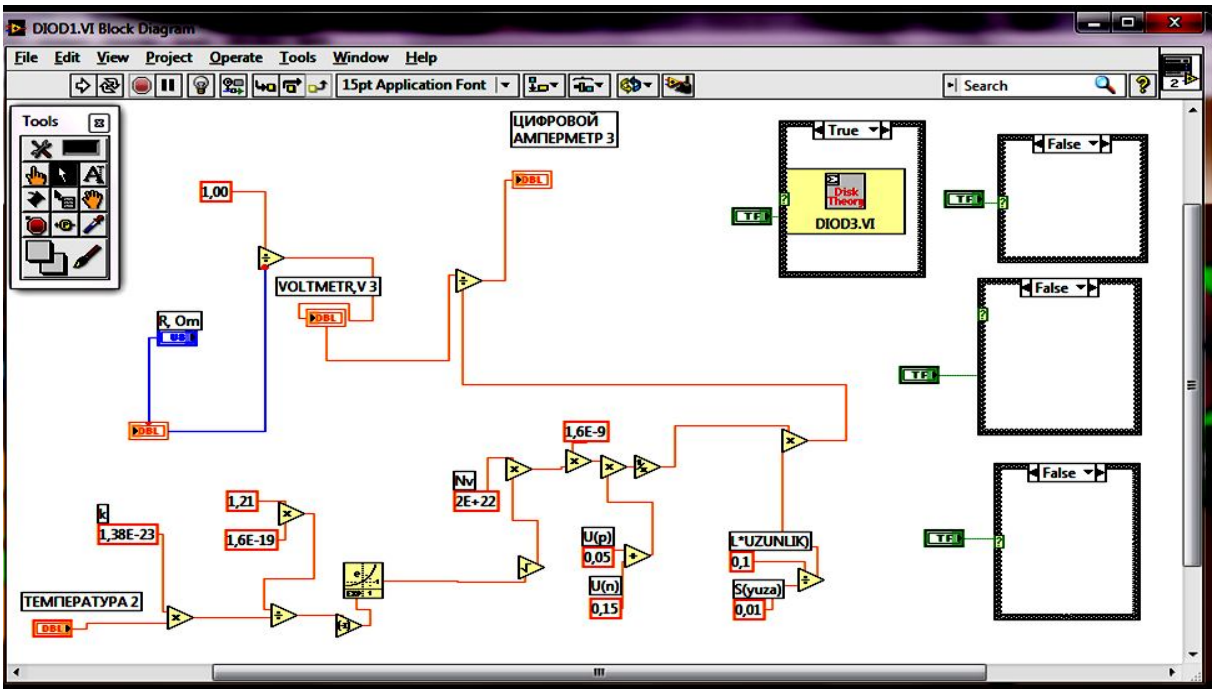


Рис. 2.15. Блок-диаграмма.

Описание виртуальной установки

В работе для различных температур снимается вольт- амперная характеристика (ВАХ) полупроводникового диода. Схема установки приведена на рис.2.14.

Порядок выполнения работы

1. Ознакомление с теоретической частью и описанием виртуальной установки.

2. Снятие ВАХ при комнатной температуре. Необходимо снять 15-20 точек. Особенно тщательно требуется измерить начальный участок ВАХ напряжений 0…. 0,8В.. Напряжение устанавливается с помощью источника питания.

3. На основе полученных результатов, выполнить следующее:

Определить ток неосновных носителей тока I_0 :

$I_0=I_{обр}$ при $U_{обр}=10В$

Определить теоретическое значение прямого тока по формуле:

$$I^{расч}_{пр} = I_0(e^{\frac{eU_{пр}}{KT}} - 1) \quad (1)$$

где e – заряд электрона e = $1{,}602*10^{-19}$кл.

К – постоянная Больцмана К=$1{,}38*10^{-23}$Дж/К

Т – абсолютная температура.

Полученные значения $I_{пр}^{расч}$ занести в табл.1, построить график и сравнить с экспериментальными значениями $I_{пр.}$ Используя данные из таблицы 1 построить ВАХ (график зависимости $I_{пр}=f(U_{пр})$ и

$I_{пр}^{расч}=f(U_{пр})$ (рис. 2.16)

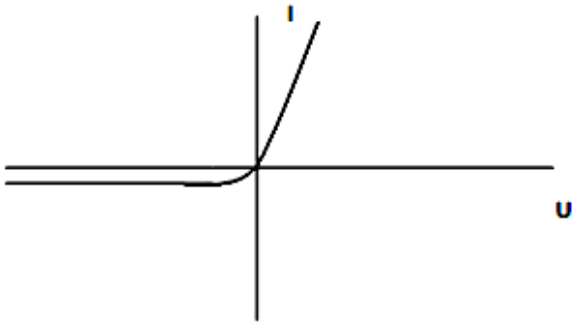


Рис.2.16. Примерная вольтметрная характеристика полупроводникового диода.

Таблица №1.

$U_{пр}$										
$I_{пр}^{расч}$										
$I_{пр}$										

Снятие ВАХ при различных температурах.

4. Поднимайте температуру печи с помощью терморегулятора. Снять прямую ветвь ВАХ, как описано в параграфе рис.2.15. Если температура повышается медленно, то можно снять ВАХ не дожидаясь установления температуры. Правда при этом измерении следует следить, чтобы снятия ВАХ температура изменялась не более чем на 1-2 градуса.
5. Снять ВАХ при других температурах.
6. Построить график зависимости $ln\ I_{пр} = f(U)$

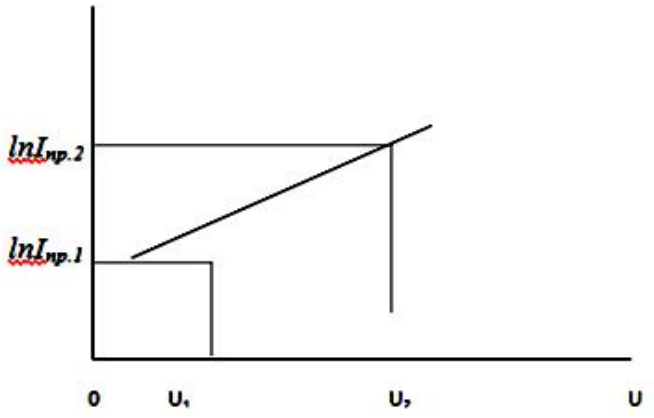


Рис. 2.17. Определить наклон прямой.

$$tg\varphi = \frac{\ln I_{пр2} - \ln I_{пр1}}{U_2 - U_1} \quad (2)$$

Вычислить отношение

$$\frac{l}{K} = Ttg\varphi \quad (3)$$

и сравнить с известным значением, подставив в (3) величину заряда электрона и постоянную Больцмана.

2.5. Лабораторная работа №5. ВП – осциллограф.

Для создания данного ВП используем имеющиеся в LabVIEW элементарные функций. Для формирования процесса используется формула (1),

$$Y = A\sin(2\pi ki/N + \varphi^0\pi/180). \qquad (1)$$

Для выполнения математических операций сложения, вычитания, умножения, деления, а также нелинейных операций используются функциональные узлы. Они находятся в подпалитре Numeric палитры Functions. Блок-схема программы, реализуемой таким образом, показана на рис. 2.18.

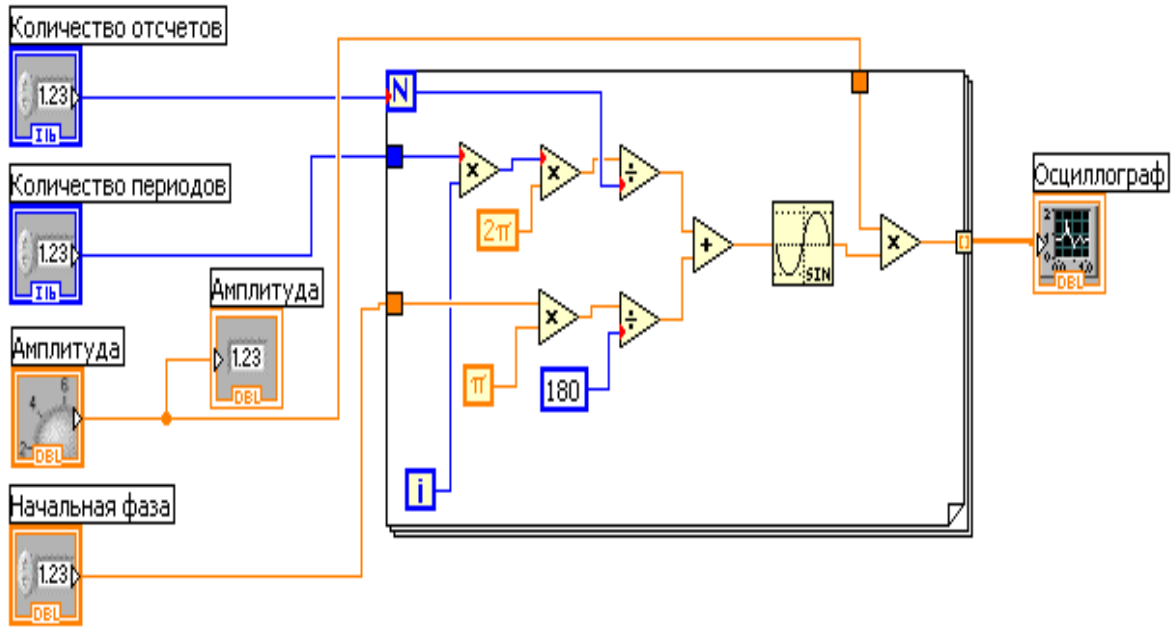


Рис. 2.18. Блок-диаграмма.

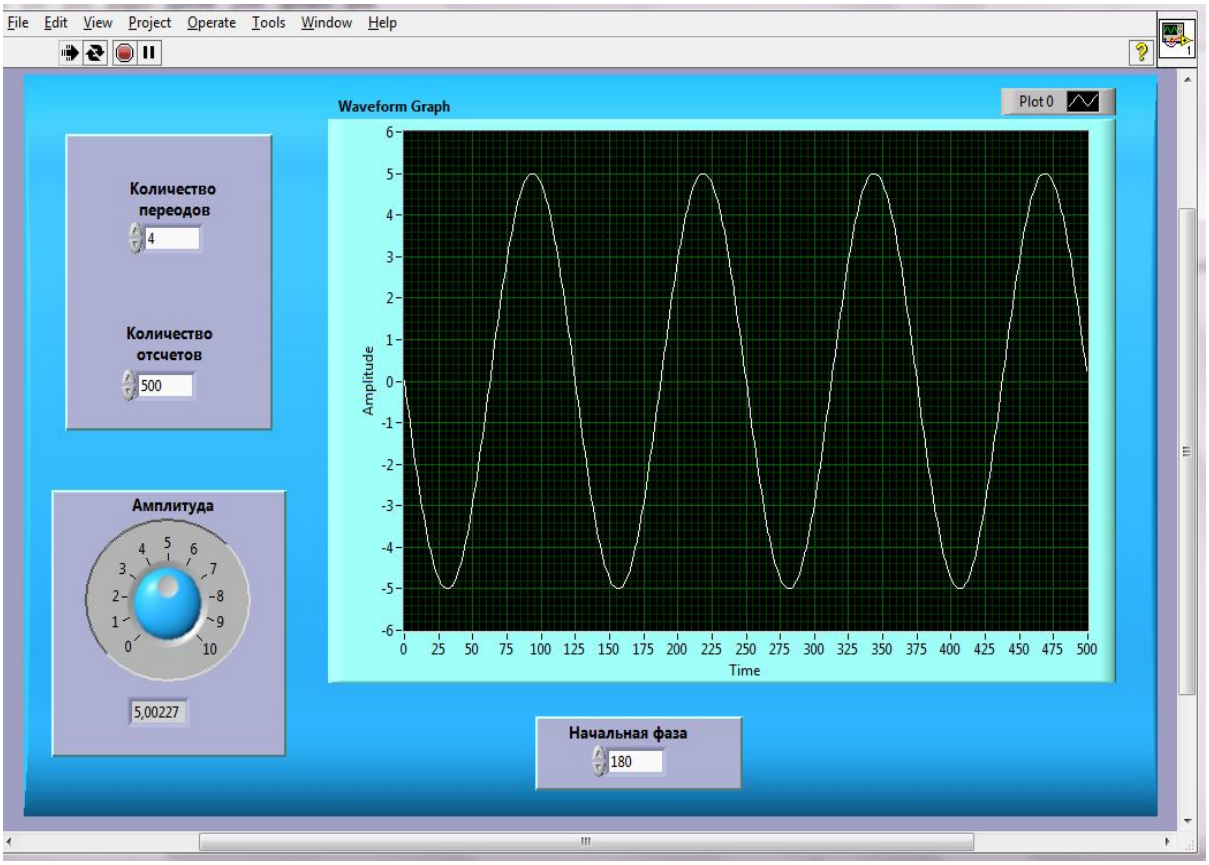


Рис.2.19. Внешний вид виртуальной установки (лицевая панель).

ГЛАВА 3. СВЯЗЬ ПЕРСОНАЛЬНОГО КОМПЬЮТЕРА С ВНЕШНИМИ УСТРОЙСТВАМИ В СРЕДЕ LabVIEW

3.1. Доступ к портам с помощью программыVISA.

LabVIEW поддерживает работу с рядом типов данных, облегчающих работу в составе автоматизированных систем научных исследований.

Разработчик LabVIEW, компания National Instruments для коммуникаций с устройствами предлагает специальную архитектуру VISA. VISA является стандартным программным интерфейсом приложения (API), осуществляющего ввод/вывод и применяется для управления контрольно-измерительным оборудованием. VISA может управлять многими типами приборов (через интерфейсы VXI, GPIB, PXI,

USB или COM-порты), вызывая соответствующие драйверы, которые зависят от типа используемого прибора.

Сессия VISA (VISA session, VISA resource name) используется при программировании внешних устройств в соответствие со стандартом VISA. Драйверы большинства современных приборов и устройств разрабатываются в соответствие со спецификацией VISA и технологией Plug&Play. Наиболее близкий "родственник" из обычных типов данных и в некоторых случаях заменитель – строка [9].

При обращении к контроллеру через VISA необходимо, прежде всего, установить саму подсистему VISA. Если вы уже работали с AVR - USB - ATmega8A через libusb, то нужно в Диспетчере Устройств удалить имеющееся устройство USB. После экспериментов всё легко возвращается в исходное состояние. У нас устройство USB макетной платы AVR - USB - ATmega8A опознавалось как устройство USBasp, находящееся в ветке lib-usb-win32 devices: теперь можно вызвать VISA Driver Wizard: укажите интерфейс USB:выберите наш контроллер по PID и VID. Напоминаем, что USB-устройство на прошивке Сергея Кухтецкого имеет VID 0x16C0 и PID 0x05DC (это бесплатные VID и PID, которые предоставляет библиотека V-USB в свободном использовании):

Так выглядет установочный файл.

При установке файла появится следующее окно (рис. 3.1):

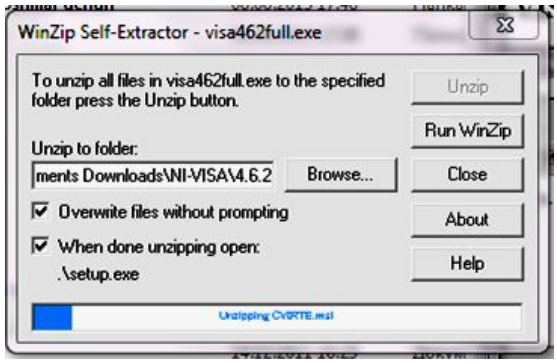


Рис. 3.1. Вид программы при установке.

В следующих картинках изображены ярлык установки VISA и окно VISAInteractiveControl (рис.3.2).

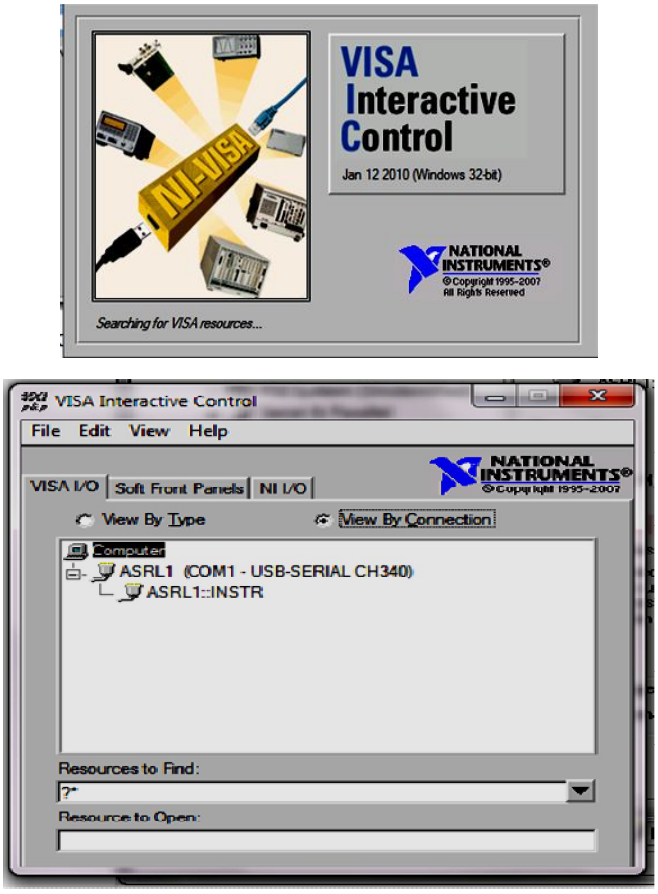


Рис. 3.2. Установочные окна VISA и VISA Interactive Control.

Берем плату с микроконтроллером AVR, соединенной с компьютером по RS232. В контроллер залита прошивка, согласно которой контроллер измеряет значение напряжения на одном из входов АЦП, и передает код АЦП (от 0 до 1023) в компьютер по последовательному каналу. Необходимо написать программу для ПК, которая будет принимать поток данных от АЦП, отображать код АЦП, преобразовывать код АЦП в значение напряжения в вольтах, отображать значение напряжения в вольтах, строить график изменения напряжения во времени.

Также обязательно надо скачать компонент NIVISA. Без этой программы LabView не «увидит» COM - порт на компьютере. VISA содержит в себе функции для работы с комуникационными портами и много чего еще. Скачать ее можно с joule.ni.com. Устанавливаем LabView

и VISA. Установка этого ПО стандартная, каких либо особенностей не имеет.

Первым делом нам нужно убедиться, что VISA нашла в системе COM - порт и корректно с ним работает. Проверить это можно так: запускаем программу Measurement&Automation. Она устанавливается вместе с LabView.

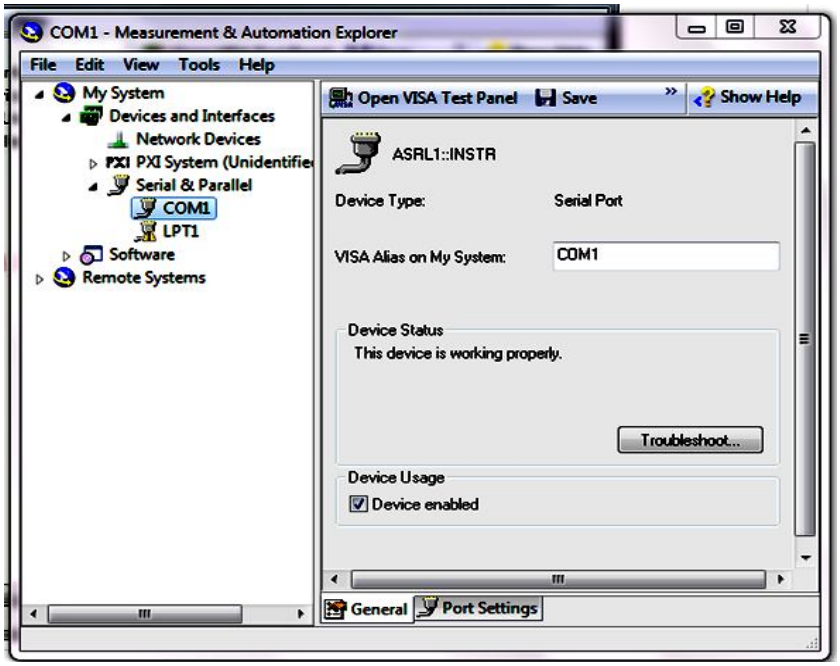


Рис. 3.3. Подпрограмма Measurement&Automation.

Если она не установилась на вашем компьютере, установить можно вручную. На диске (образе с LabView она есть).

В левой части окна мы видим оборудование, обнаруженное в системе. Среди всего прочего находим наш COM - порт. Справа есть кнопка OpenVisatestpanel. С помощью нее можно протестировать выбранное устройство. В случае с COM - портом там можно отправить или принять заданную по умолчанию или произвольную последовательность символов. Если с портом все в порядке, можно приступить непосредственно к созданию нашей программы.

Запускаем LabView. В окне GettingStarted выбираем пункт BlankVi, то есть новый виртуальный прибор.

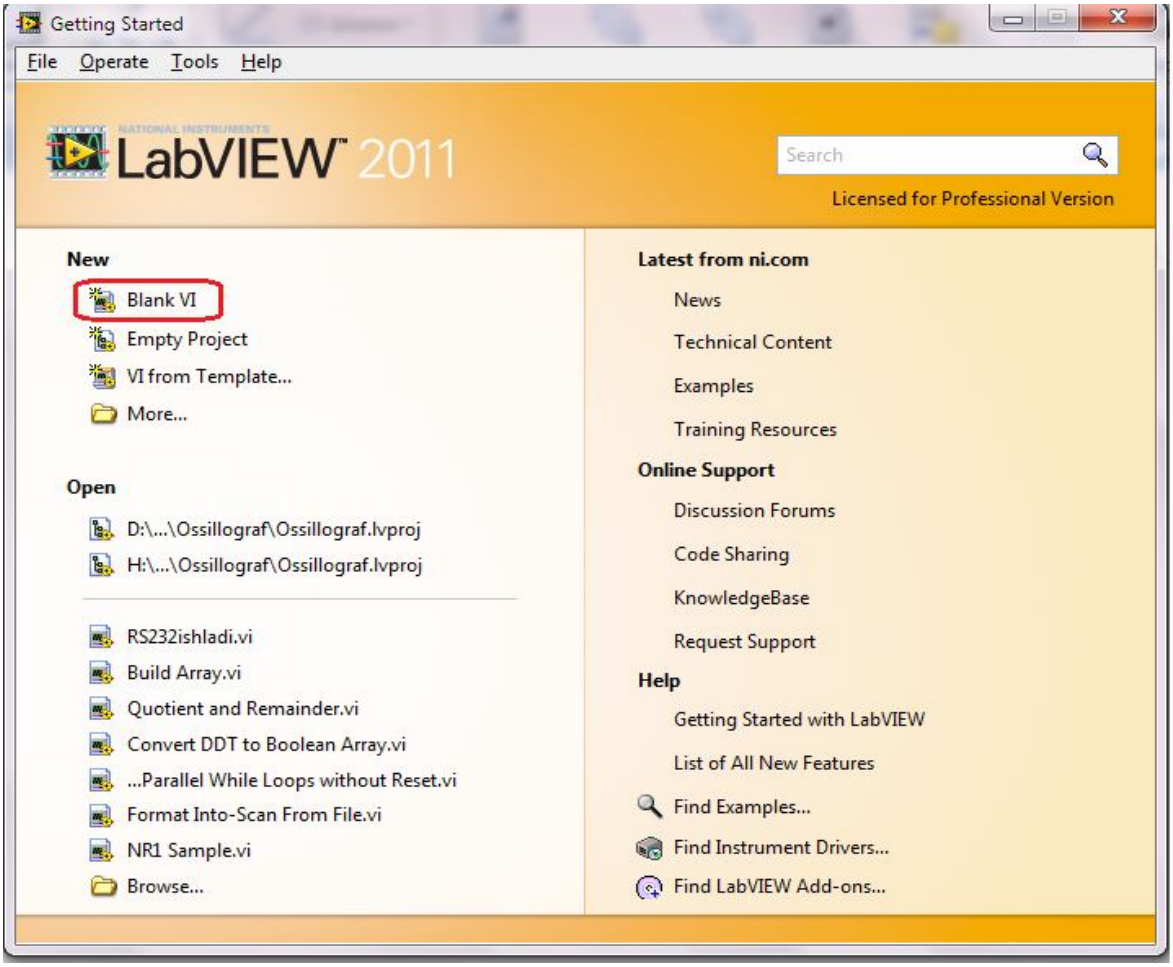


Рис.3.4. Окно GettingStarted.

На экране монитора это выглядет вот так:

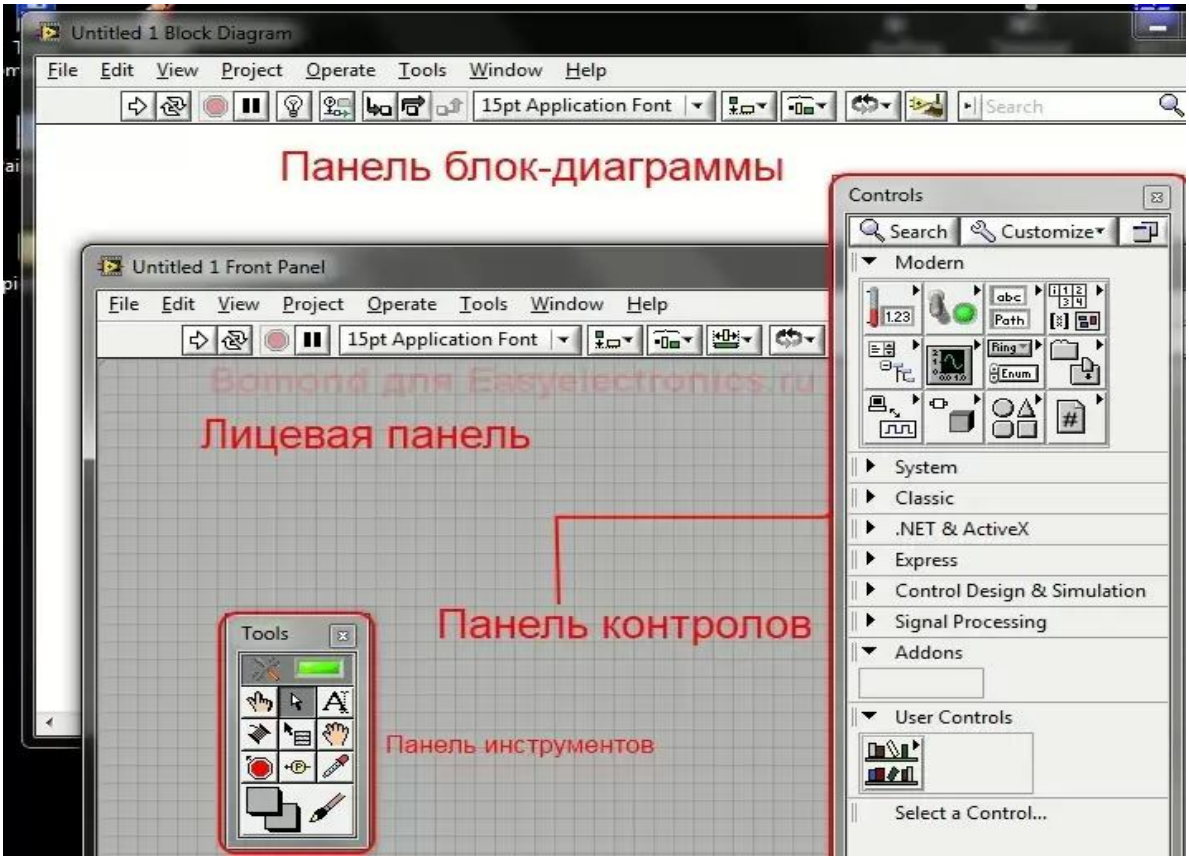


Рис.3.5. Лицевая панель и блок-диаграмма.

Ну а теперь займемся более интересными вещами, а именно сделаем наш простейший вольтметр, о котором я говорил в самом начале.

Итак, нам необходимо сделать следующее. Сначала нужно настроить и проинициализировать последовательный порт, и запустить бесконечный цикл . В цикле мы используем функцию чтения из порта и принимаем информацию. Преобразуем информацию для отображения на графике, пересчитываем код АЦП в значение напряжения в вольтах. При выходе из цикла закрываем порт.

Так в интерфейсе нашей программы не будет никаких управляющих элементов, кроме кнопки Стоп, а будет лишь отображение результата, мы поступим так: сначала создадим блок-диаграмму, а потом добавим недостающие элементы на лицевую панель. Хотя делать нужно наоборот! Но в данном случае так удобнее.

На панели блок-диаграммы помещаем из палитры Structures элемент WhileLoop, это наш бесконечный цикл. Обводим рамкой цикла область, достаточную для размещения внутри алгоритма. В правом нижнем углу есть красная точка, щелкнем по ней ПКМ и выберем CreateControl. На лицевой панели у нас тут же появится кнопка Stop. При щелчке на ней наша программа завершится.

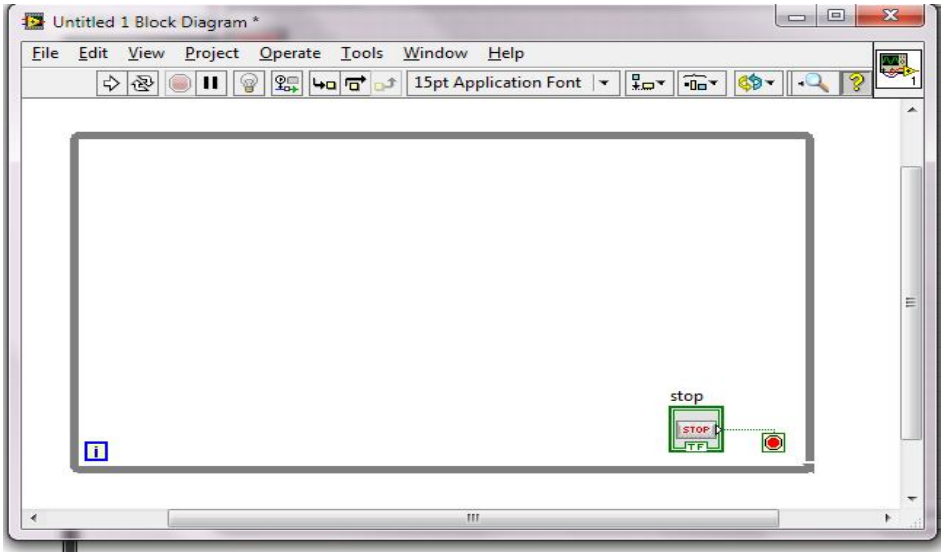


Рис. 3.6. Элемент WhileLoop в блок-диаграмме.

Теперь вне цикла мы должны разместить функции инициализации и закрытия порта. Слева инициализация, справа закрытие. Опять же щелкаем ПКМ и выбираем функции ConfigurePort, Read и Close. Эти функции находятся в палитре InstrumentI/O —>Serial. Функцию чтения помещаем внутрь цикла. Соединяем с помощью катушки с проводами выходы и входы функций. Для функции Read мы должны задать количество байтов, которая она будет принимать. Щелкаем ПКМ на среднем входе функции Read и выбираем Create->Constant, вводим значение, например 200. На данном этапе должно получится следующая картина как на скрине.

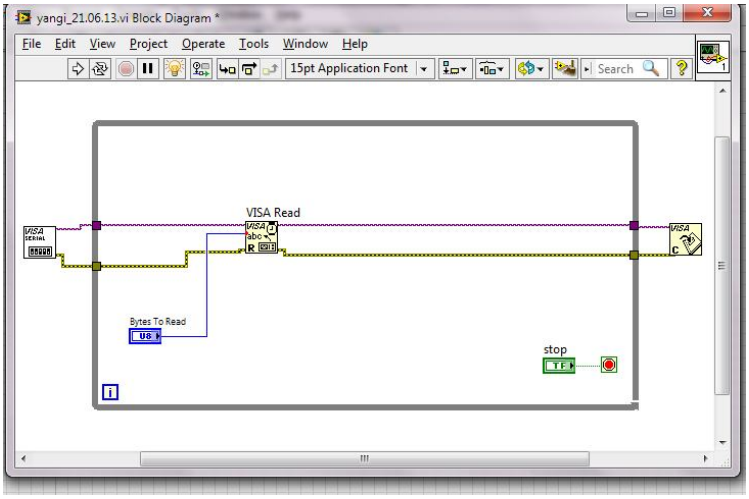


Рис. 3.7. Функции ConfigurePort, Read и Close.

Нужно создать контроль для функции инициализации порта. Нам вполне хватит двух — скорость порта и имя порта. Точно так же как мы создавали константу для функции чтения, создаем контроли. ПКМ на нужных входах функции инициализации и пункт. Нас интересуют два входа: Visaresoursename и BaudRate (по умолчанию 9600). Таперь перейдем на лицевую панель и добавим необходимые компоненты, а именно экран отрисовки графика и метки для отображения кода АЦП и напряжения в вольтах. Соответственно это элементы WaweformChartc палитры Graph и два элемента NumericIndicator с палитры Numeric.

Вернемся к блок-диаграмме и переместим появившиеся элементы внутрь цикла. Мы близимся к завершению. Единственное, нам нужно еще преобразовать строку символов, поступающих с выхода функции Read к формату, который показывают наши индикаторы. И еще реализовать простейшую математику по переводу кода АЦП в вольты. Для преобразования строки мы воспользуемся функцией Scanfromstring из палитры String. Помещаем ее внутрь цикла. Теперь математика, для того чтобы преобразовать код АЦП в значение напряжения в вольтах нужно умножить код на величину опорного напряжения (в нашем случае это пять вольт) и получившееся значение разделить на 1023 (так как АЦП имеет разрядность 10 бит). Необходимые функции умножения и деления, а также

константы (5 и 1023) разместим в цикле. Скрины каждого соединения делать не будем, ибо и так картинок достаточно. Приведем финальный скрин всех соединений. Там все предельно просто. Перейдем к нашему интерфейсу и настроим график. Выделим нижнее значение по оси Y и поставим 0. Выделим верхнее и установим 5. Таким образом наша шкала по оси Y в диапазоне 0-10 вольт. Выбираем COM - порт, вводим скорость обмена, запускаем по кнопке со стрелкой нашу программу и крутим резистор на плате, неотрывно наблюдая при этом на экране результат нашего труда. Щелкаем на кнопке Stop, чтобы остановить программу.

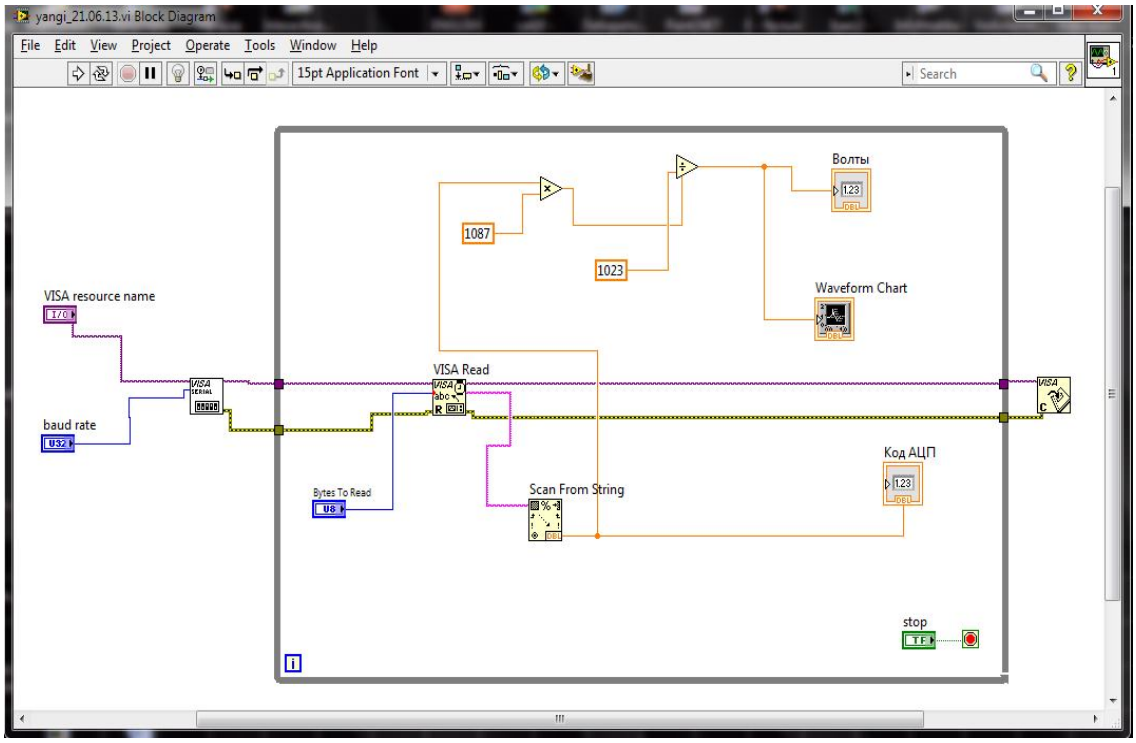


Рис. 3.8. Блок-диаграмма. ВП «Цифровой вольтметр».

Данный пример это лишь незначительная часть всех возможностей LabView.

Вот окончательный вид нашей работы:

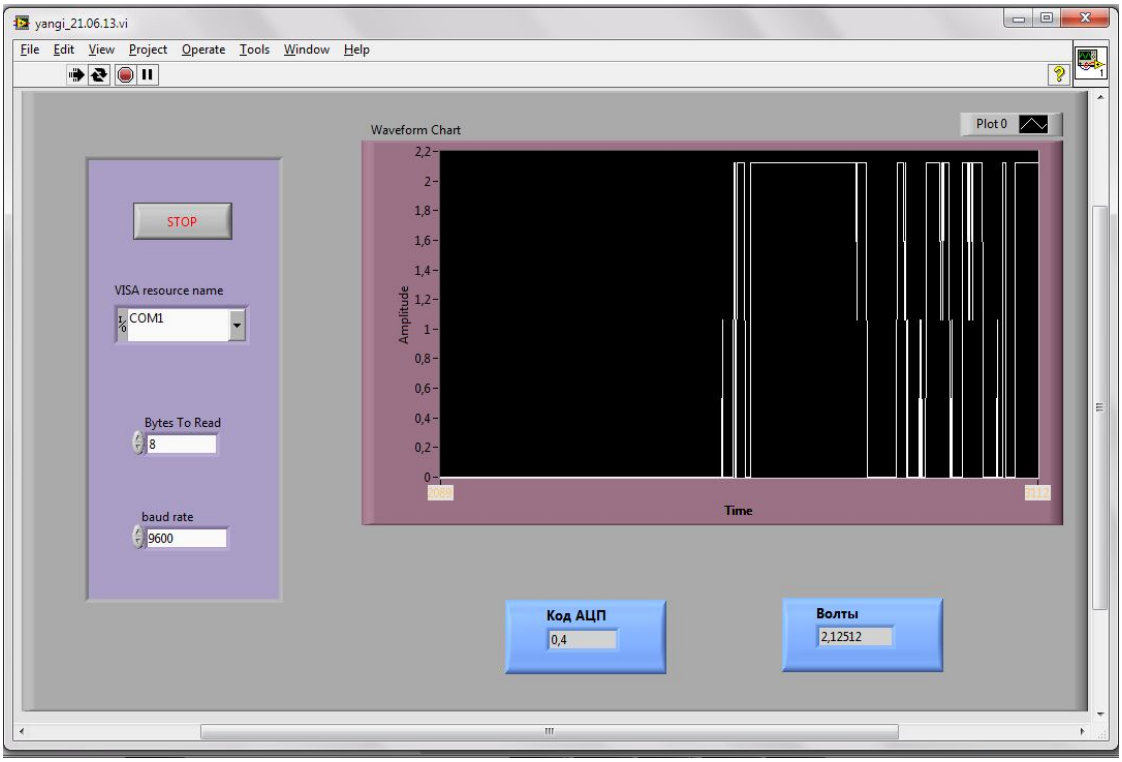


Рис. 3.9. Лицевая панель. ВП «Цифровой вольтметр».

3.2. Вольтметр на основе AVR микроконтроллера.

АЦП – аналогово-цифровой преобразователь (ADC- Analog-to-Digital Converter) преобразует некий аналоговый сигнал в цифровой. Разрядность АЦП определяет точность преобразования сигнала. Время преобразования – соответственно скорость работы АЦП. АЦП встроен во многих микроконтроллерах семейства AVR и упрощает использование микроконтроллера во всяких схемах регулирования, где требуется оцифровывать некий аналоговый сигнал.

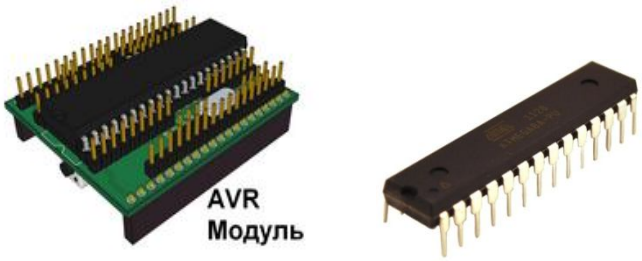


Рис. 3.10. Вид AVRмодуля и микроконтроллера ATmega8A.

Рассмотрим принцип работы АЦП. Для преобразования нужен источник опорного напряжения и собственно напряжение, которое мы хотим оцифровать (напряжение, которое преобразуется, должно быть меньше опорного). Также нужен регистр, где будет храниться преобразованное значение, назовем его Z. Входное напряжение = Опорное напряжение*Z/2^N, где N – разрядность АЦП. Условимся, что этот регистр, как у ATmega8A, 10-ти битный. Преобразование в нашем случае проходит в 10 стадий. Старший бит Z9 выставляется в единицу [23].

Далее генерируется напряжение (Опорное напряжение*Z/1024), это напряжение, оно с помощью аналогового компаратора сравнивается с входным, если оно больше входного, бит Z9 становиться равным нулю, а если меньше – остается единицей. Далее переходим к биту Z8 и вышеописанным способом получаем его значения. После того, как вычисление регистра Z окончено, выставляется некий флаг, который сигнализирует, что преобразование закончено и можно считывать полученное значение. На точность преобразования могут очень сильно влиять наводки и помехи, а также скорость преобразования. Чем медленнее происходит преобразования – тем оно точней. С наводками и помехами следует бороться с помощью индуктивности и емкости, как советует производитель в инструкции:

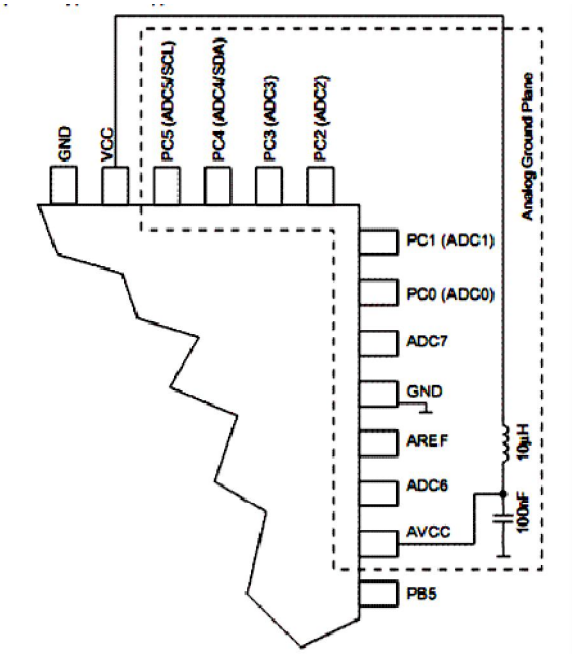


Рис. 3.11. Фрагмент AVR микроконтроллера - ATmega8.

В некоторых корпусах и моделях микроконтроллеров есть отдельные выводы для питания АЦП: AVCC и AGND. Выводы ADCn – каналы АЦП.

С какого канала будет оцифровываться сигнал можно выбрать с помощью мультиплексора.

Теперь продемонстрируем на примере сказанное выше. Соорудим макет, который будет работать как вольтметр с цифровой шкалой. Условимся, что максимальное измеряемое напряжение будет 10В. Также пусть наш макет выводит на ЖКИ содержимое регистра ADC.

Обвязка микроконтроллера и ЖКИ WH1602A стандартна. X1 – кварцевый резонатор на 4 Мгц, конденсаторы C1,C2 – 18-20 пФ. R1-C7 цепочка на выводе reset по 10 кОм и 0,1 мкФ соответственно. Сигнальный светодиод D1 и ограничивающий резистор R2 200 Ом и R3 – 20 Ом.

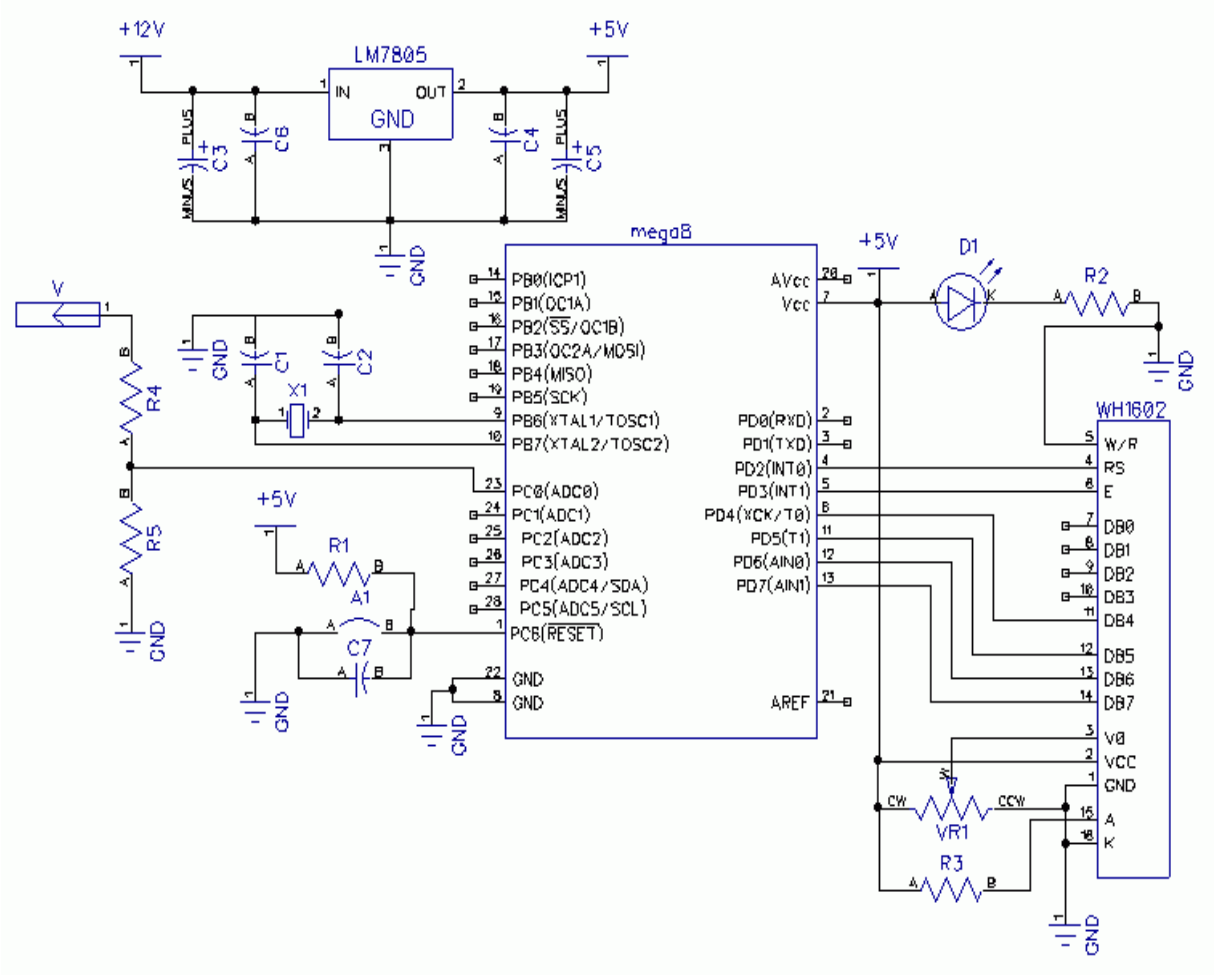


Рис. 3.12. Принципиальная схема цифрового вольтметра.

Регулировка контраста ЖКИ – VR1 на 10 кОм. Источник опорного напряжения мы будем использовать встроенный на 2,56В. С помощью делителя R4-R5 мы добьемся максимального напряжения 2,5В на входе PC0, при напряжении на щупе 10В. R4 – 3 кОм, R5 – 1 кОм, в их номиналу нужно отнестись тщательно, но если нет возможности подобрать точно, можно сделать любой резистивный делитель 1:4 и программно подкорректировать показания, если это потребуется. Дроссель на 10мкГн и конденсатор на 0,1 мкФ для устранения шумов и наводок на АЦП на схеме не показан. Их наличие подразумевается само собой, если используется АЦП.

Составляется программа для установки Atmega8A.

В начале мы инициализируем порты ввода/вывода. Для того, чтобы служить входом АЦП, пинPC0 должен работать на вход. Далее проводим инициализацию ЖКИ и АЦП. Инициализация АЦП заключается в его включении битом ADEN в регистре ADCSRA. И выбора частоты преобразования битами ADPS2, ADPS1, ADPS0 в том же регистре. Также выбираем источник опорного напряжения, биты REFS1 REFS0 в регистре ADMUX и вход АЦП: биты MUX0,MUX1,MUX2, MUX3 (в нашем случаем входом АЦП является PC0, поэтому MUX0.3=0). Далее, в вечном цикле, начинаем преобразования установкой бита ADSC в регистре ADCSRA. Дожидаемся окончания преобразования (бит ADIF в ADCSRA становиться равным 1). Далее вынимаем данные из регистра ADC и выводим их на ЖКИ. Вынимать данные из ADC нужно в такой последовательности: v=(ADCL+ADCH*256); если использовать v=(ADCH*256+ADCL),это в упор не работает.

Так-же есть хитрость, чтобы не работать с дробными числами. Когда производятся вычисления входного напряжения в вольтах. Мы просто будем хранить наши напряжения в милливольтах. Например, значение переменной voltage 4234 означает, что мы имеем 4,234 вольта. Вообще операции с дробными числами занимают очень много памяти микроконтроллера (наша прошивка вольтметра весит чуть больше 4 килобайт, это половина памяти программ ATmega8), их рекомендуется использовать только при особой необходимости. Вычисления входного напряжения в милливольтах просто: voltage=R_division*2.56*u*1.024;

Здесь R_division – коэффициент резистивного делителя R4-R5. Так, как реальный коэффициент делителя может отличаться от расчетного, то наш вольтметр будет врать. Но подкорректировать это просто. С помощью тестера меряем некое напряжение, получаем X вольт, а наш вольтметр пускай показывает Y вольт. Тогда R_division = 4*X/Y, если Y больше X и

4*Y/X если X больше Y. На этом настройка вольтметра завершена, и им можно пользоваться.

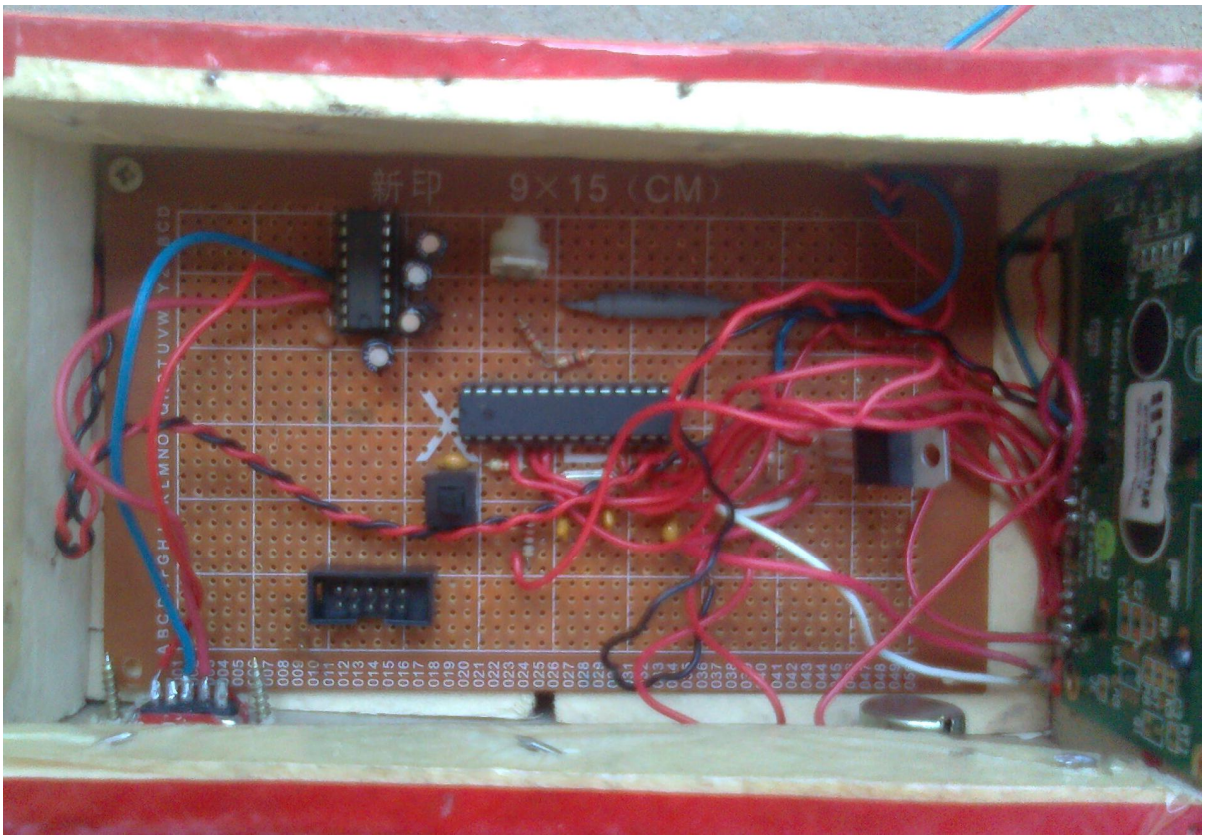


Рис. 3.13. Вид готового цифрового вольтметра.

3.3. Отправка цифрового сигнала через COM - порт.

Почти все микроконтроллеры имеют на борту последовательный порт — UART. Работает он по стандартному последовательному протоколу, а значит его можно без проблем подключить к компу на COM - порт. Но есть тут одна проблема — дело в том, что комповыйRS232 он за логические уровни принимает +/- 12 вольт, а UART работает на пятивольтовых уровнях. Как их совместить? Для этого существует несоколько вариантов схем преобразователей уровня, но самая популярная это все же на специальном преобразователе RS232-TTL. Это микросхема MAX232 и ее аналоги.

Практически каждая фирма делает свой преобразователь, так что тут сгодится и ST232, и ADM232, и HIN232. Схемка простая как три копейки — вход, выход, питание и обвязка из пяти конденсаторов. Конденсаторы обычно ставятся 1μF электролиты, но в некоторых модификациях ставится 0.1μF керамика. Я везде впаивал 0.1μF керамику и обычно этого хватало. :) Работает как часы. Если же на высоких скоростях будет ошибка, то надо будет повышать емкость.

AVR тут не исключение и поддерживает этот протокол в полном объеме полностью аппаратно. По структуре это обычный асинхронный последовательный протокол, то есть передающая сторона по очереди выдает в линию 0 и 1, а принимающая отслеживает их и запоминает. Синхронизация идет по времени — приемник и передатчик заранее договариваются о том на какой частоте будет идти обмен. Если скорость передатчика и приемника не будут совпадать, то передачи может не быть вообще, либо будут считаны не те данные [9].

Протокол. Вначале передатчик бросает линию в низкий уровень — это старт бит. Почуяв что линия просела, приемник выжидает интервал T1 и считывает первый бит, потом через интервалы T2 определяются остальные биты. Последний бит это стоп-бит, говорящий о том, что передача этого байта завершена. Это в самом простом случае.

В конце байта, перед стоп битом, может быть и бит четности. Который получается если суммировать между собой все биты, для контроля качества передачи. Также может быть два стопа, опять же для надежности. Битов может быть не 8, а 9. О всех этих параметрах договариваются на берегу, до начала передачи. Самым же популярным является 8 бит, один старт один стоп, без четности.

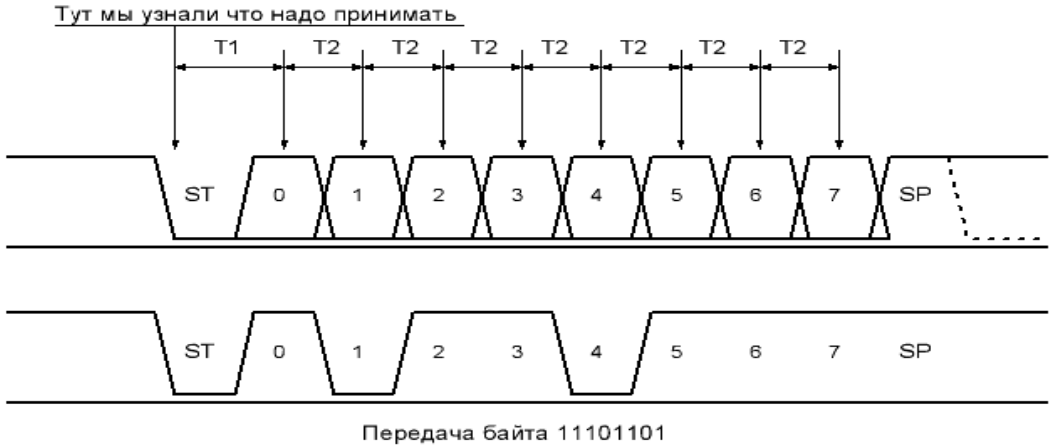


Рис. 3.14. Вид байта передачи информации**.**

Причем протокол передастся правильно — все реализовано аппаратно. Разве что захочется завести второй UART, тогда придется делать его программно. По такому же протоколу работает COM - порт компьютера, разница лишь в разнице напряжений, поэтому именно этот протокол я буду использовать для связи микроконтроллера с компьютером. Для преобразования напряжений можно использовать RS232-TTL конвертер. Мы же применим встроенный в Pinboard мост USB-UART, который образовывает в системе виртуальный COM PORT.

Аппаратная часть. Ну тут все просто, соединяем крест-накрест приемник с передатчиком и готово.

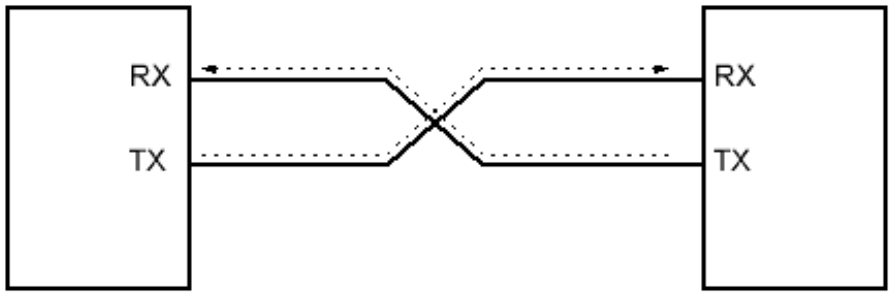


Рис. 3.15. Интерфейсная структура Pinboard мост USB-UART.

Внутри же вначале находится накопительный сдвиговый регистр, в котором происходит сборка байта из битов и регистры данных UDR, куда этот бит передается. Такая структура исключает возможность считать не до конца полученный байт.

Использование UART. У AVR есть регистр UDR это UART DataRegister. На самом деле это два разных регистра, но имеют один адрес. Просто на запись попадает в один (регистр передатчика), а на чтение берет из другого (регистр приемника). Таким образом достаточно просто отправить данные в этот регистр и они передаются приемнику, и, наоборот, считывать их оттуда по приходу.

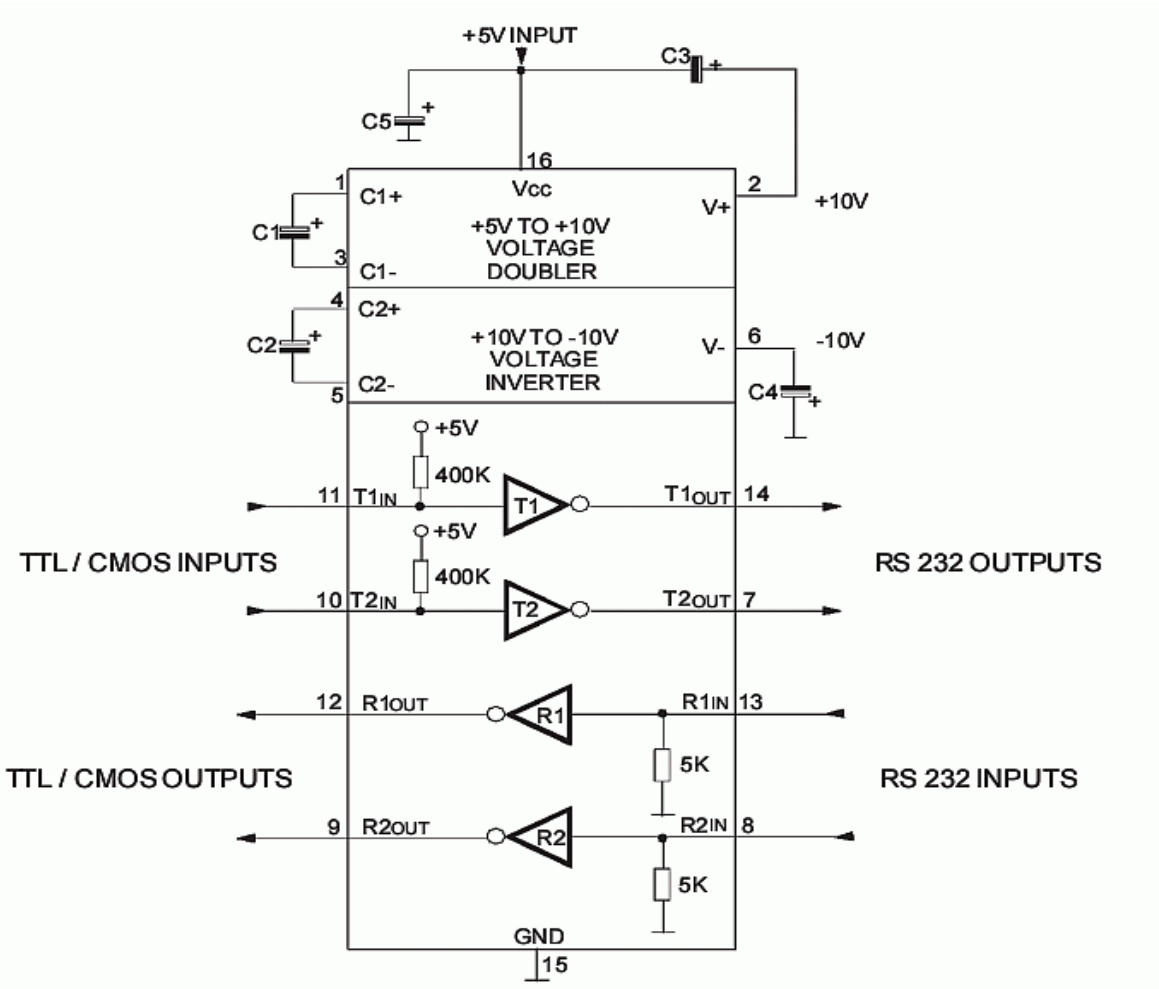


Рис. 3.16. Вид интерфейса RS232.

Кстати, существует еще и MAX3232 это то же самое, но на выходе у него не 5вольт TTL, а 3.3 вольта TTL. Её используют для низковольтных контроллеров.

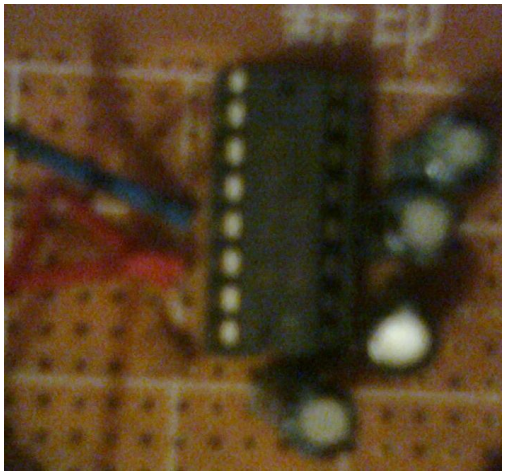


Рис. 3.17. Вид интерфейса RS232 для COM - порта.

Плата подключается в разъем COM - порта. Подается 5 вольт питания на схему, а затем замыкаемRx на Tx. Дальше открываем любую терминалку, хоть HyperTerminal, подкючаемся к порту и начинаем посылать байты, они должны тотчас возвращаться обратно. Если этого не произошло — проверяем схему. Если работает, то дальше все просто. Тот провод, который идет от ножки 9 микросхемы MAX232- это передающий вывод, его заводим на ногу RxD контроллера. А тот, который с ножки 10 — принимающий, его смело сажаем на вывод TxD контроллера.

CDC-232 проект представляет из себя простую схему на микроконтроллере AVR (ATtiny45/85, ATtiny461, ATtiny2313, ATmega8/48/88), которая при подключении к порту USB компьютера (далее просто PC) создает виртуальный COM-порт.

В обоих проектах используется бесплатная библиотека V-USB, которая позволяет средствами firmware, прошитого в микроконтроллер AVR, поддержать работу интерфейса USB.

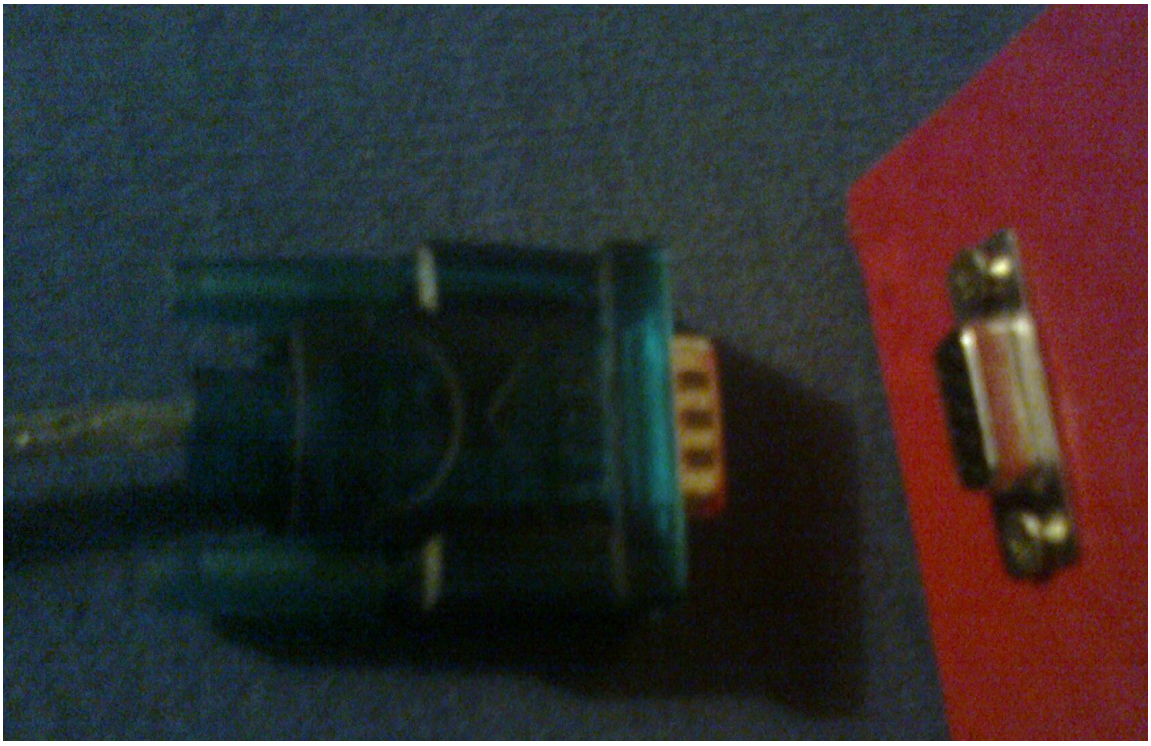


Рис. 3.18. Вид 340 USB кабеля и RS232 COM - порта.

Проект (CDC-232) создает на компьютере виртуальный COM-порт, через который можно обмениваться данными с каким-нибудь другим устройством, имеющим низковольтный RS-232C (например, с микроконтроллером AT89C51 Atmel).

CDC-232 создает виртуальный COM-порт на PC, даже если он не имеет реального порта RS-232C. Это позволяет производить обмен данными RS-232C (без сигналов управления) после подсоединения устройства и установки драйвера.

Запрограммируйте AVR, спаяйте схему и подключите устройство в порт USB компьютера. Установите драйвер (если у Вас операционная система Windows). Получите доступ к устройству через сгенерированный виртуальный COM-порт из программы терминала или написанную Вами программу (которая работает с COM-портом).

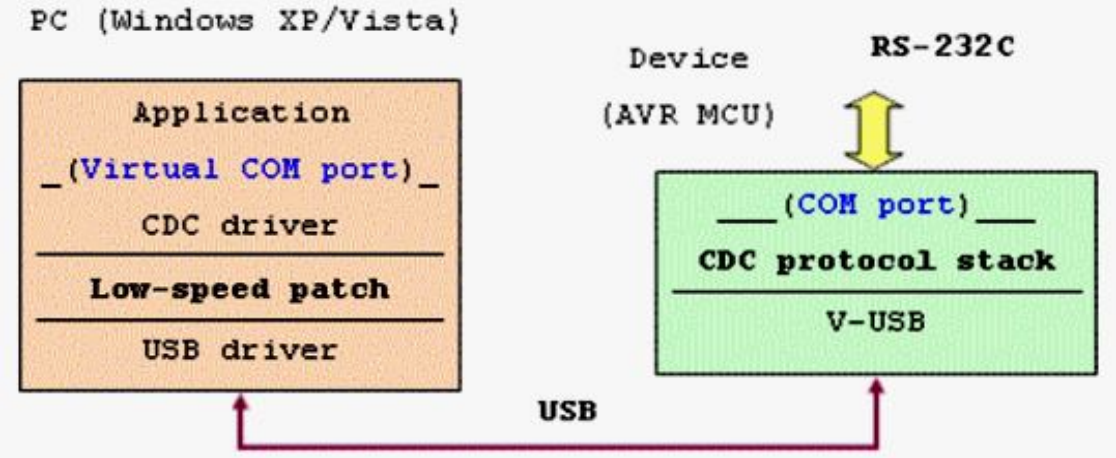


Рис. 3.19. Вид интерфейса RS232 для виртуального COM-порта.

Сигналы управления (DTR, DTS, RTS, CTS) не используются, поэтому настройте программу терминала в режим "noflow-control" (без контроля потока).

Windows запросит установку драйвера заново всякий раз, когда Вы подключите устройство в другой порт USB. Ранее установленный драйвер детектируется автоматически. Будет назначен другой номер COM-порта. Если Вы установите серийный номер в AVR (пересоберите исходный код с новым usbconfig.h), Вы можете получить тот же самый номер COM-порта на любом порте USB. Но в этом случае Вы не можете подключить сразу несколько устройств CDC с одним и тем же серийным номером.

Перед тем, как отсоединить устройство, закройте COM-порт в терминальной программе или в Вашем программном приложении. В противном случае Вы не сможете подключить устройство снова из-за неверного дескриптора файла (brokenfilehandle). В этом случае перезапустите программу терминала или Ваше приложение. Переключиться в режим быстрой передачи можно при использовании "lowcdc.vbs", при этом можно получить скорость выше 9600 bps.

У микроконтроллеров ATmega8/48/88's внутренний UART конфигурируется от PC автоматически. Формат настройки 1200-57600bps, данные 7/8, четность N/E/O, количество стоп-бит ½.

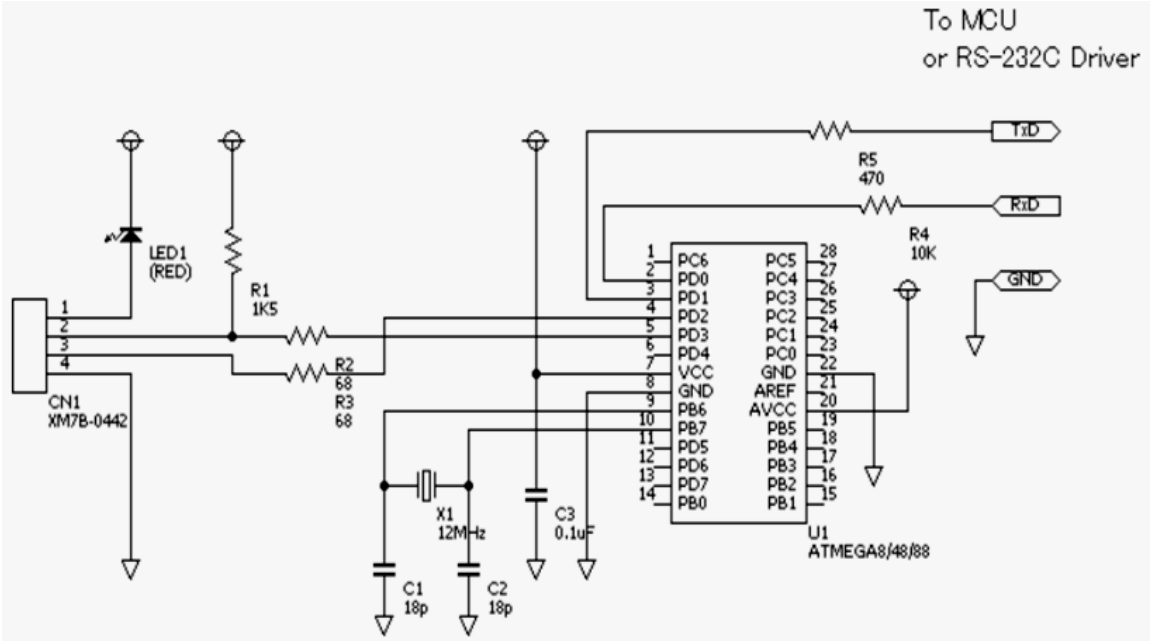


Рис. 3.20. Вид схемы микроконтроллера ATmega8/48/88's и RS232.

Подключение к физическим линиям RS-232C требует инверсии полярности TxD и RxD, а также соответствие уровней сигнала. Используйте специальную IC наподобие MAX232. Вы можете её заменить простыми схемами.

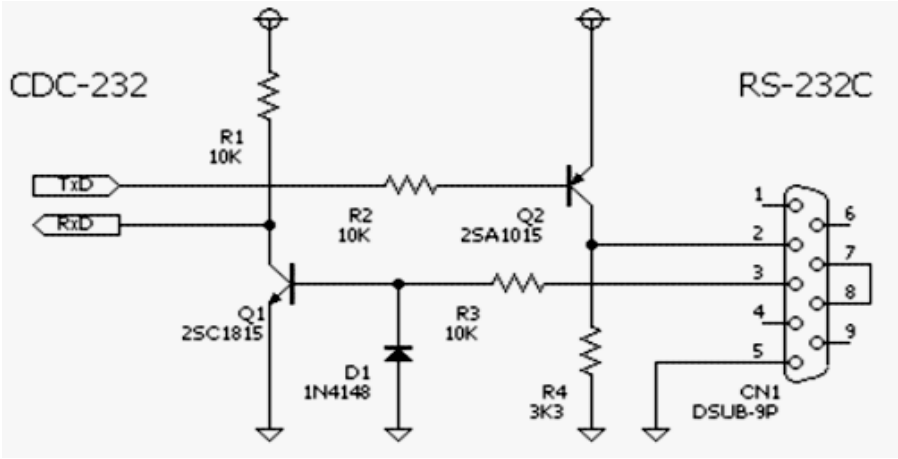


Рис. 3.21. Схемы инверсии RS232 для COM - порта.

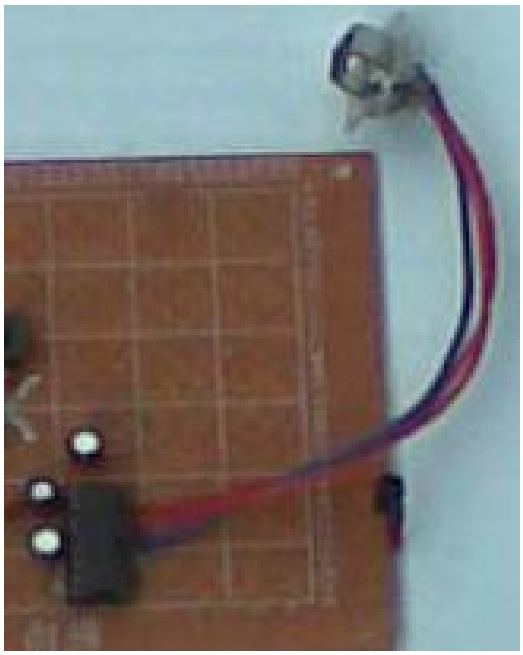


Рис. 3.22. Вид интерфейса нашего готового RS232 для COM - порта.

Когда целевой MCU имеет другое питание Vcc (отличное от 3.3V), образуются обратные токи утечки через сигнальные линии. Это искажает сигнал, либо подпитывает MCU без источника питания. Эта схема не очень хороша, но достаточна для большинства случаев.

Рис. 3.23. Вид COM - порт для внешнего устройства.

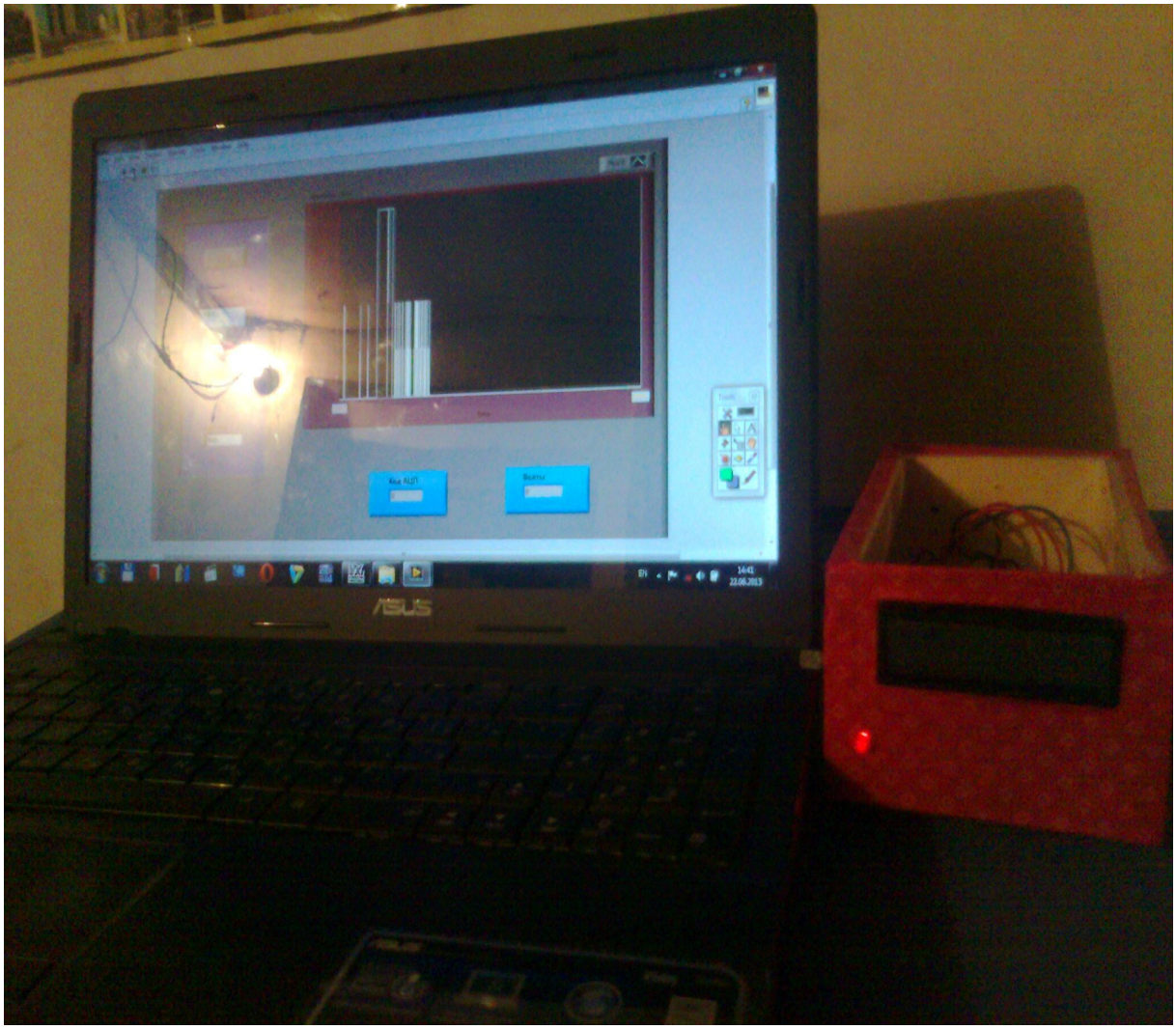


Рис. 3.24. Вид нашего готового виртуального и реального прибора «Цифровой вольтметр».

Заключение

В данной диссертации изучены вопросы создания виртуальных приборов в среде Lab VIEW. Графический язык и концепция потокового программирования LabVIEW позволяет решать задачи более удобными и эффективными методами, чем традиционные текстовые языки. Ключевые особенности программирования на языке G, а именно интуитивно понятный и наглядный графический код, а также управляемое потоком данных выполнение программы, позволяют сделать процесс программирования более близким к процессам мышления, чем другие языки. Несмотря на высокий уровень абстракции кода, производительность программ, написанных в среде LabVIEW, остается сопоставимой с языками типа C, благодаря встроенному компилятору кода.

Среда LabVIEW содержит тысячи функций для анализа данных, обеспечивает удобный интерфейс разработки пользовательских интерфейсов на базе имеющегося и собственных наборов элементов управления, предоставляет удобные инструменты сохранения данных и подготовки отчетов.

Таким образом, высочайшая степень интеграции с оборудованием и множество уникальных особенностей делают LabVIEW лучшей программной платформой для решения комплексных инженерных и промышленных задач.

LabVIEW обладает огромным арсеналом достоинств, таких как полноценный язык программирования, интуитивно понятный процесс графического программирования, широкие возможности сбора, обработки и анализа данных, управления приборами, генерации отчетов и обмена данных через сетевые интерфейсы.

В данной диссертационной работе созданы несколько ВП, а именно:

1. Создание простейшего виртуального прибора, выполняющего простые алгебраические функции.
2. Виртуальный прибор преобразования °C в °F в виде подпрограммы.
3. Моделирование работы комбинационных цифровых устройств.
4. Влияние температуры на вольт - амперную характеристику p-n перехода.
5. ВП – осциллограф.
6. Цифровой вольтметр.

Разработки данной диссертационной работы в дальнейшем будут применяться в учебном процессе.

СПИСОК ЛИТЕРАТУРЫ

1. Автоматизация измерений, контроля и испытаний : учебное пособие / С.В. Мищенко, А.Г. Дивин, В.М. Жилкин, С.В. Пономарев, А.Д. Свириденко. – Тамбов : Изд-во Тамб. гос. техн. ун-та, 2007. – 116 с.

2. Пейч, Л.И. LabVIEW для новичков и специалистов / Л.И. Пейч, Д.А. Точилин, Б.П. Поллак. – М. : Горячая линия – Телеком, 2004. – 384 с.

3. Н.А. Виноградова, Я.И. Листратов, Е.В. Свиридов. «Разработка прикладного программного обеспечения в среде LabVIEW». Учебное пособие – М.: Издательство МЭИ, 2005.

4. Батоврин В. К., Бессонов А. С, Мошкин В. В., Папуловский В. Ф. LabVIEW: практикум по основам измерительных технологий: Учебное пособие для вузов. - М.: ДМК Пресс, 2005. - 208 с.

5. Евдокимов Ю.К, Линдваль В. “Lab VIEW для радиоинженера: от виртуальной модели до реалного прибора”. Москва: “ДМК Пресс” 2007. С. 14-49.

6. Р.Ш. Загидуллин. LabVIEW в исследованиях и разработках. – М.: Горячая линия – Телеком. 2005, - 312с.

7. Плющаев В.И. Компьютерное схемотехническое моделирование радиоэлектронных устройств. – Издательство ВГАВТ, Нижний Новгород, 2001.

8. Суранов А. Я. LabVIEW 7: справочник по функциям. - М.: ДМК Пресс, 2005. - 512 с.

9. Тревис Дж. LabVIEW для всех / Джеффри Тревис: Пер. с англ. Клушин Н. А. – М.: ДМК Пресс; ПриборКомплект, 2005. – 544с,: ил.

10. Карлащук В.И. Электронная лаборатория на IBM PC. Программа Electronics Workbench и ее применение. – М.: Изд. «Солон–Р», 2001. – 726

11. Демирчяна, К.С. Использование виртуальных инструментов LabVIEW / под ред. К.С. Демирчяна, В.Г. Миронова. –М. : Солон-Р; Радио и связь; Горячая линия – Телеком, 1999. – 268 с.

12. Автоматизация физических исследований и эксперимента: компьютерные измерения и виртуальные приборы на основе Lab VIEW 7/ Под. ред. Бутырина П. А. -М.: ДМК Пресс, 2005. 264 с.: ил.

13. Н. А. Виноградова, Я.И.Листратов, Е.В.Свиридов. Разработка прикладного программного обеспечения в среде LabVIEW: Учебное пособие – М.: Издательство МЭИ, 2005. – 50с.

14. Виртуальная электронная лаборатория в инструментальной среде LabVIEW: Методические указания для лабораторно-практических занятий студентов заочного отделения/ Сост. Евдокимов Ю.К., Насырова Р.Г., Байтуллин А.Ф. Казань: Изд-во Казан. гос. техн. ун-та, 2001. 26с.

15. Ю.В. Бобков, В.Д. Циделко Лабораторный практикум в системе дистанционного обучения по курсу «Цифровые измерительные приборы» на базе NI LabVIEW // Образовательные, научные и инженерные приложения в среде LabView и технологии National Instruments: Труды V международной научно-практической конференции / Москва, 2006.

16. Создание виртуальных приборов в среде LabView : метод. указания к лаб. работам / Владим. гос. ун-т ; сост. Н.Ю. Макарова. – Владимир : Изд-во Владим. гос. ун-та, 2010. – 59 с.

17. Шарапов А.В. Микроэлектроника: Учебное пособие. — Томск: Томский межвузовский центр дистанционного образования, 2007. — 158 с.

18. Бессонов, А.С. LabVIEW: практикум по электронике и микропроцессорной технике / А.С. Бессонов, В.В. Мошкин, В.К. Батоврин. М.: ДМК Пресс, 2005. − 182 с. − ISBN 5-94074-204-1.

19. Суранов, А. Я. LabVIEW 8.20. Справочник по функциям / А.Я. Суранов. – М.: ДМК Пресс, 2007. − 536 с. − ISBN: 5-94074-347-1, 978-5-9407-4347-7. 4. Фрике, К. Вводный курс цифровой электроники / К. Фрике. − М.: Техносфера, 2003. – 432 с. − ISBN 5-94836-015-6.

20. Якубовский, С.В. Цифровые и аналоговые интегральные микросхемы: справочник / С.В. Якубовский, Л.И. Ниссельсон, В.И. Кулешова и др.; под ред. С.В. Якубовского. − М.: Радио и связь, 1990. − 496 с.

21. http://www.automationlabs.ru/

22. http://digital.ni.com/

23. http://www.labview.ru/

24. http://ru.wikipedia.org/

25. http://www.Icard.ru/

26. http://www.google.ru/