Федеральное агентство по образованию

Государственное образовательное учреждение
высшего профессионального образования

Владимирский государственный университет

Кафедра приборостроения
и информационно-измерительных технологий

СОЗДАНИЕ ВИРТУАЛЬНЫХ ПРИБОРОВ В СРЕДЕ LABVIEW

Методические указания к лабораторным работам

Составитель
Н.Ю. МАКАРОВА

Владимир 2010

УДК 519.876.5
ББК 22.18
С54

Рецензент
Кандидат технических наук профессор,
зам. декана факультета радиофизики, электроники и медицинской техники
Владимирского государственного университета
Г.П. Колесник

Печатается по решению редакционного совета
Владимирского государственного университета

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.

С54

Приведены методики выполнения лабораторных работ по построению виртуальных измерительных систем в программном комплексе *LabView* по дисциплине «Электронные методы измерений». Рассматриваются принципы построения простейших виртуальных приборов, изучается язык графического программирования *LabView*.

Предназначены для студентов 4-го курса дневного отделения, обучающихся по специальности 200106 − информационно-измерительная техника и технологии и 200101 − приборостроение.

Ил. 46. Табл. 4. Библиогр.: 5 назв.

УДК 519.876.5
ББК 22.18

Введение

Создание виртуальных приборов в настоящее время сопровождает процесс разработки современных информационно-измерительных систем в учебном процессе и на производстве. Данный курс лабораторных работ посвящен проектированию виртуальных приборов в среде LabView и используется в дисциплинах «Электронные методы измерений», «Программное обеспечение измерительных процессов», «Интеллектуальные средства измерений». Технология виртуальных приборов находится в постоянном развитии за счет развития компьютерной техники и контрольно-измерительного оборудования. Виртуальные приборы позволяют создавать гибкие и легко адаптируемые к новым условиям системы, которые соответствуют требованиям надежности, точности и производительности. На сегодняшний день наиболее мощным и надежным средством разработки виртуальных приборов является среда графического программирования *LabView* компании *National Instruments*, предлагающая наиболее широкий спектр инструментов для работы с различного рода промышленным и контрольно-измерительным оборудованием.

В лабораторных работах изучается инструментарий *LabView* для проектирования приборов (работа № 1), моделируются элементы цифровой электроники (работы № 2 – 5), рассматриваются принципы работы с параллельным портом (работа № 6) и с текстовыми данными (работа № 7). Методические указания предназначены для студентов специальностей 200101 – приборостроение и 200106 – информационно-измерительная техника и технологии очного обучения.

Лабораторная работа № 1
ВВЕДЕНИЕ В *LABVIEW*

Цель работы: ознакомление с программным пакетом *LabView*.

Оборудование: дисплейный класс, среда визуального программирования *LabView* версии 7.0 или выше.

1. Общие сведения

Основные элементы структуры. *LabView*, в отличие от языков программирования *C, PASCAL* или *BASIC*, использует графический язык программирования, предназначенный для создания программ в форме структурных схем. *LabView* содержит обширные библиотеки функций и инструментальных средств, предназначенных для создания систем сбора данных и систем автоматизированного управления. *LabView* также включает стандартные инструментальные средства разработки программ.

Программы в *LabView* называются виртуальными приборами (*VI*, *Virtual instrument* − англ.), так как их вид и функционирование имитируют реальные измерительные приборы. Однако, при этом виртуальные приборы подобны функциям в программах стандартных языков программирования.

Структура виртуального прибора может быть представлена следующими элементами:

- лицевой панелью (лицевая панель может содержать кнопки, переключатели, регуляторы и другие органы управления и индикаторы);
- структурной схемой (структурная схема представляет собой наглядное представление решения задачи и содержит исходные коды для виртуального прибора.

LabView придерживается концепции модульного *программирования*. Можно разделить прикладную программу на несколько более простых подпрограмм, а затем создать несколько виртуальных приборов для выполнения каждой подпрограммы и объединить эти виртуальные приборы на общей структурной схеме, выполняющей основную программу. В результате основной виртуальный прибор верхнего уровня будет содержать совокупность суб-приборов (*subVI*), которые смогут реализовать функции прикладной программы. Многие *subVI* низкого

уровня часто выполняют задачи, общие для нескольких прикладных программ, так что можно разработать специализированный набор *subVI*, хорошо подходящий для прикладных программ, которые будут созданы в дальнейшем.

Лицевая панель. Лицевая панель виртуального прибора − прежде всего комбинация органов управления и индикаторов. Органы управления моделируют инструментальные устройства ввода данных и передают данные на структурную схему виртуального прибора. Индикаторы моделируют инструментальные устройства вывода, которые отображают данные, собранные или сгенерированные структурной схемой виртуального прибора.

Структурная схема. Окно схемы содержит структурную схему виртуального прибора, которая является исходным графическим текстом виртуального прибора в *LabView*. Структурная схема создается посредством соединения объектов, которые посылают или получают данные, выполняют определенные функции и управляют потоком выполнения.

Первичные программные объекты структурной схемы − узлы, терминалы и провода.

При появлении органа управления или индикатора на лицевой панели, *LabView* помещает соответствующий терминал на структурную схему. Пиктограммы функций также имеют терминалы. Данные, которые вводятся в органы управления, поступают с лицевой панели через терминалы органов управления на структурную схему. Затем данные поступают в функции. Когда функции завершают свои внутренние вычисления, они производят новые значения данных на своих выходных терминалах. Данные поступают на терминалы индикаторов и повторно попадают на лицевую панель, где они и отображаются.

Узлы − элементы выполнения программы. Они аналогичны инструкциям, операторам, функциям и подпрограммам в стандартных языках программирования. Функция − один из типов узлов. *LabView* имеет обширную библиотеку функций для математических вычислений, сравнения, преобразования, ввода/вывода и так далее. Другой тип узлов − структура. Структуры являются графическим представлением циклов и операторов выбора традиционных языков программирования, повторяя блоки инструкций или выполняя их по условию. *LabView*

имеет также специальные узлы для взаимосвязи с внешними текстовыми программами и для вычислений по текстовым формулам.

Провода − пути данных между терминалами источника и адресата. Нельзя подключить терминал-источник к другому источнику, нужно подключать терминал-адресат к другому терминалу-адресату. Можно подключать один источник к нескольким адресатам. Провода имеют различный вид или цвет, в зависимости от типа данных, которые по ним передаются.

Принцип, который управляет выполнением программы в *LabView*, называется потоком данных. Запущенный узел выполняется только тогда, когда на всех входах появляются данные; узел выдает данные на все выходные терминалы только тогда, когда он заканчивает выполнение; и данные сразу же поступают от терминала источника на терминал адресата. Поток управления регулируется командами. Поток данных − управляется данными или зависит от данных.

Когда пиктограмма виртуального прибора помещена в схему другого виртуального прибора, первый виртуальный прибор становится *subVI*, то есть подпрограммой в *LabView*. Органы управления и индикаторы *subVI* получают данные от вызывающего виртуального прибора и возвращают их ему же.

Инструменты используются для выполнения определенных функций. Многие из инструментов *LabView* содержатся в палитре «***Tools***» (табл. 1.1). Изначально при создании нового виртуального прибора либо загрузке существующего на экране появляется окно лицевой панели. Перейти к окну лицевой панели из окна структурной схемы можно, выбрав в меню *Windows>>Show Panel*.

Объекты на лицевой панели создаются при выборе их из палитры «***Controls***» (*Windows>>Show Controls Palette*).

При создании объекты лицевой панели появляются с прямоугольником метки, в которую сразу же можно ввести текст − название органа управления или индикатора.

Объектное меню вызывается нажатием правой кнопки мыши, когда курсор в виде руки или стрелки находится на объекте.

Созданная метка объекта редактируется меточным инструментом из палитры «***Tools***».

Таблица 1.1

Основные рабочие инструменты LabView

Рабочий инструмент	Оригинальное название	Русское название	Функции инструмента
	*Operating tool*	Инструмент действия – “*Рука*”	Размещает объекты палитр **«Controls»** и **«Functions»** на лицевой панели и структурной схеме.
	*Positioning tool*	Позиционный инструмент – “*Стрелка*”	Размещает объекты, изменяет их размеры и выбирает их.
	*Labeling tool*	Меточный инструмент	Редактирует тексты меток объектов и создает свободные метки.
	*Wiring tool*	Монтажный инструмент – “*Катушка*”	Подключает объекты друг к другу на структурной схеме.
	*Object pop-up menu tool*	Инструмент объектного меню	Вызывает объектное меню.
	*Scroll tool*	Инструмент прокрутки	Прокручивает окно без использования слайдеров.
	*Breakpoint tool*	Инструмент контрольной точки	Устанавливает контрольные точки на функциях, циклах, структурах.
	*Probe tool*	Инструмент пробы	Создает пробные измерители на проводах.
	*Color Copy tool*	Инструмент копии цвета – “*Пипетка*”	Копирует цвета для вставки с помощью Цветового инструмента.
	*Color tool*	Цветовой инструмент – “*Кисть*”	Устанавливает цвета переднего плана и фона.

Выровнять объекты лицевой панели по какой-либо оси, а также более равномерно распределить их на лицевой панели можно с помощью опций в окнах «*Align Objects*» (Выравнивание объектов) и/или «*Distribute Objects*» (Распределение объектов).

Изменить цвет лицевой панели или ее индикаторов и органов управления можно с помощью инструмента «кисть» из палитры.

Инструмент «катушка» используется для подключения терминалов объектов структурной схемы. Отметкой курсора или рабочим острием «катушки» является конец раскрученного сегмента провода. Область терминала мигает, когда острие монтажного инструмента правильно установлено на терминал, а рядом с терминалом появляется его название.

Когда провода пересекаются, в первом выведенном проводе появляется маленький промежуток, как будто этот провод проходит ниже второго.

При монтаже сложных встроенных узлов или *subVI*, нужно обращать внимание на концы проводов и надписи, которые появляются, когда монтажный инструмент приближается к пиктограмме виртуального прибора. Концы проводов, показанные вокруг пиктограммы виртуального прибора, указывают тип данных своей формой, толщиной и цветом. Точки в концах проводов указывают входы, в то время как выходы не имеют таких точек. Надписи представляют собой названия высвечиваемых входов или выходов.

Вместо того, чтобы создавать константу, орган управления или индикатор, выбирая его из меню, можно щелкнуть мышью на терминале и выбрать «*Create Constant*», «*Create Control*» или «*Create Indicator*». При этом созданные константа, орган управления или индикатор подключаются автоматически.

Аналогичные действия можно производить с выводами функций виртуального прибора, константами и терминалами органов управления или индикаторов лицевой панели.

2. Выполнение работы

Пример создания простого вычислительного устройства

1. Откройте новый виртуальный прибор *File>>New VI.*

2. На лицевой панели (*Front Panel*) разместите два управляющих элементов для ввода двух чисел *A* и *B*. Из подпалитры «***Numeric Controls***» палитры «***Controls***» выберите цифровой элемент ввода чисел «*Numeric Control*». Разместите элемент на поле лицевой панели и в появившейся метке введите название переменных *A* или *B*.

3. На лицевой панели создайте четыре цифровых индикатора для вывода результатов вычисления. Из подпалитры «***Numeric Controls***» палитры «***Controls***» выбрать цифровой элемент ввода чисел «*Numeric Indicator*». Разместить элемент на поле лицевой панели и в появившейся метке введите название «A+B», «A-B», «A*B», «A/B».

4. На данный момент лицевая панель может выглядеть так, как показано на рис. 1.1.

5. Переключитесь на структурную схему *Windows>>Show Diagram*. На структурной схеме размещены терминалы, соответствующие органам управления и индикаторам лицевой панели. Терминалы имеют те же метки, что и соответствующие им объекты лицевой панели.

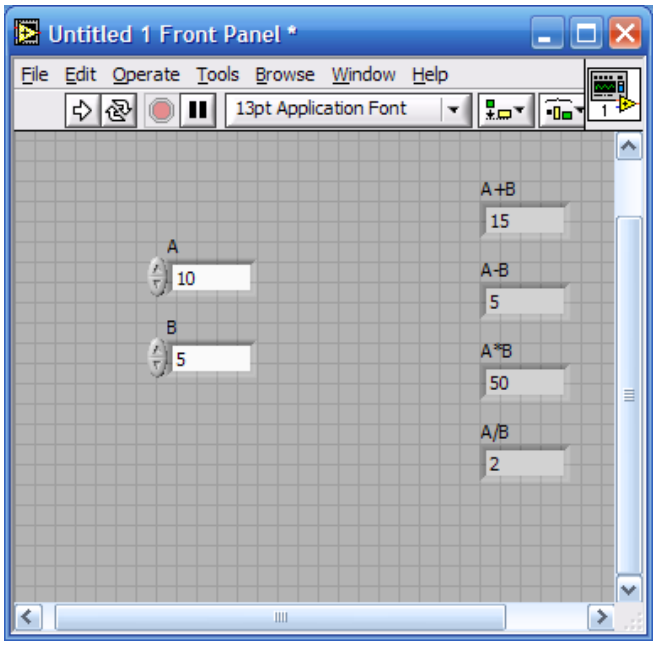


Рис. 1.1. Лицевая панель простого вычислительного устройства

6. Разместите на ней объекты управления (*Controls*) слева, а индикаторы (*Indicators*) справа аналогично тому, как они расположены на лицевой панели.

7. Выберите функции вычислительного устройства из палитры «*Functions*» >> «*Arithmetic and Comparison*» >> «*Express Numeric*»: сложение (*Add*), вычитание (*Subtract*), умножение (*Multiply*) и деление (*Divide*) (рис. 1.2).

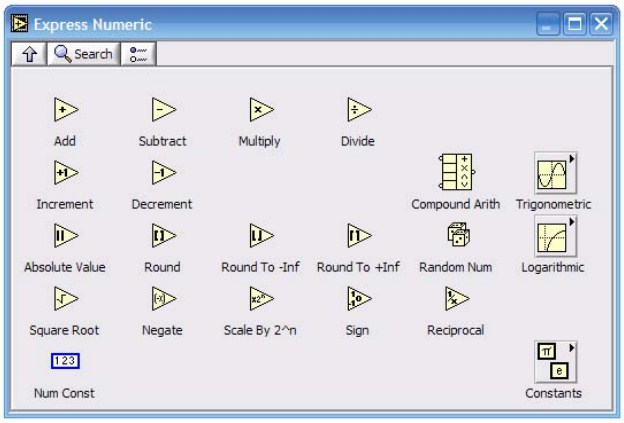


Рис. 1.2. Функциональные элементы палитры «Express Numeric»

8. Из палитры «***Tools***» выберите «катушку». Соедините между собой терминалы органов управления, функций и индикаторов.

9. На этом монтаж структурной схемы закончен. На данный момент структурная схема прибора может выглядеть, так как показано на рис. 1.3.

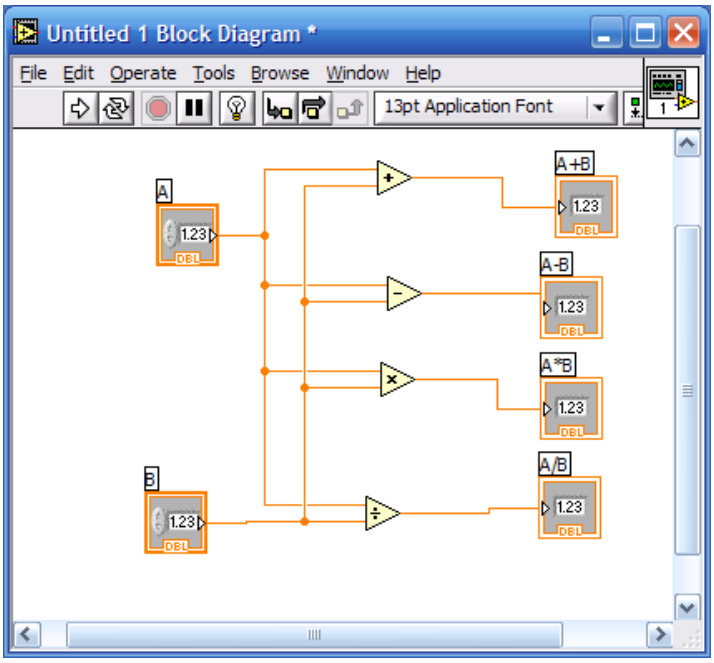


Рис. 1.3. Структурная схема простейшего вычислительного устройства

10. Перейдите в окно лицевой панели и запустите виртуальный прибор, нажав кнопку «*Run*» («запуск») в левом верхнем углу окна.

Для отчета сделайте снимки экрана (screenshots) для структурной схемы и лицевой панели созданного прибора.

Пример создания простого виртуального прибора «Спектральный анализатор прямоугольного импульса»

Основными функциональными узлами этого виртуального прибора является генератор прямоугольного импульса и вычислитель спектра мощности.

Генератор прямоугольного импульса (*Pulse Pattern.vi*) размещен в палитре ***«Functions>>Analysis>>Signal Generation»***. Этот генератор формирует на своем выходе «*Pulse Pattern*» одномерный массив отсчетов прямоугольного импульса с заданной задержкой, шириной и амплитудой. При этом количество отсчетов сигнала задается целым числом на входе «*samples*», длительность и задержка импульса (в количестве отсчетов) задаются целыми числами на входах «*width*» и «*delay*» соответственно, а амплитуда (в условных единицах) задается реальным десятичным числом на входе «*amplitude*».

Вычислитель спектра мощности (*Power Spectrum.vi*) размещен в палитре ***«Functions>>Analysis>>Digital Signal Processing»***.

1. Создание нового виртуального прибора лучше всего начать с лицевой панели *Windows>>Show Front Panel*.

2. Выберите из подпалитры «***Graph***» палитры «***Controls***» − осциллограф (*Waveform Graph*) и перенесите его на лицевую панель. В появившейся метке введите название индикатора «Сигнал».

3. Еще один такой же индикатор выберите и разместите на лицевой панели ниже первого индикатора. Введите в метку его название «Спектр». В объектном меню этого индикатора снимите также выделение с опции *X Scale>>AutoScale X*. С помощью меточного инструмента введите конечное значение горизонтальной шкалы 50.

4. Из подпалитры «***Numeric»*** палитры «***Controls»*** выберите три вертикальных ползунковых регулятора (*Vertical Pointer Slide*) для регуляторов «Амплитуда», «Длительность» и «Задержка», разместите их на лицевой панели слева от индикаторов сверху вниз и введите их названия в метки. С помощью меточного инструмента измените конечные значения регулирования на шкале регуляторов «Длительность» и «Задержка» на 50.

5. Из подпалитры «***Boolean»*** палитры «***Controls***» выберите кнопку «СТОП» (*Rectangular Stop Button*) и разместите ее на лицевой панели снизу от регуляторов.

На данный момент лицевая панель может выглядеть так, как показано на рис. 1.4.

6. Из подпалитры «***Numeric***» палитры «***Functions***» выберите числовую константу (*Numeric Constant*) и разместите ее на структурной схеме выше терминалов органов управления. С помощью меточного инструмента введите в константу значение 100.

7. Из палитры «***Functions***» выберите подпалитру «***Analysis***», а из нее − подпалитру «***signal Generation***». Из этой подпалитры выберите генератор прямоугольного импульса (*Pulse_Pattern.vi*) и разместите его на структурной схеме справа от терминалов органов управления.

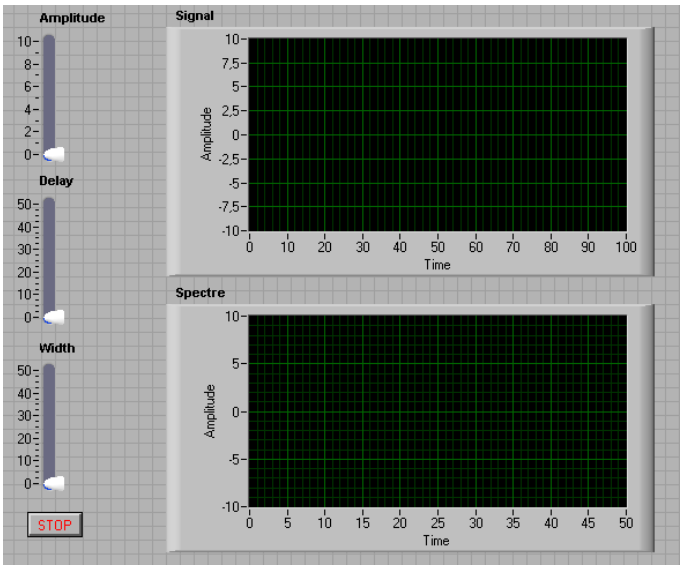


Рис. 1.4. Лицевая панель прибора

8. Из подпалитры «***Analysis***» выберите подпалитру «***Digital Signal Processing***». Из нее выберите вычислитель спектра мощности (*Power_Spectrum.vi*) и разместите его на структурной схеме между генератором прямоугольного импульса и терминалами индикаторов.

9. Из подпалитры «***Advanced***» палитры «***Functions***» выберите функцию «Стоп» (*Stop*) и разместите ее на структурной схеме рядом с терминалом кнопки «*Stop*».

10. Из палитры «***Tools***» выберите «катушку». Соедините между собой терминалы органов управления, функций, констант и индикаторов так, как показано на рис. 1.5.

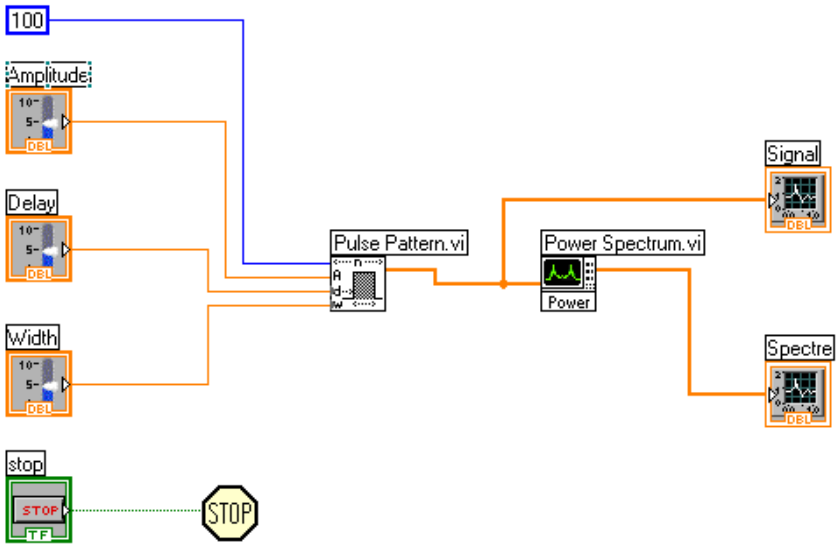


Рис. 1.5. Структурная схема прибора

11. Перейдите в окно лицевой панели и запустите виртуальный прибор, нажав кнопку «*Run Continuously*».

12. Остановить работу виртуального прибора можно, нажав на кнопку «*Stop*» на лицевой панели, либо кнопку «*Abort Execution*» («Прервать выполнение») в левом верхнем углу окна.

При возникновении вопросов по работе с *LabView* можно вызвать интерактивную справку по *LabView*, выбрав из меню *Help>>VI, Functions, & How to Help* или *Help>>Search the LabView bookshelf*. Кроме того, можно вызвать интерактивную справку относительно практически каждого объекта структурной схемы, вызвав *Help>>Show context help*.

Создайте описанный выше виртуальный прибор. Просмотрите работу виртуального прибора, прохождение сигналов по структурной схеме (с помощью кнопки с изображением лампочки наверху окна структурной схемы).

Для отчета сделайте снимки экрана (screenshots) структурной схемы и лицевой панели созданного прибора.

Изучите основные узлы, органы управления, и функции, использованные в виртуальном приборе.

3. Содержание отчета

Отчет оформляется каждым студентом самостоятельно. Защита проходит в начале каждого следующего занятия с демонстрацией работы программы на ЭВМ. Студент, не подготовивший или не защитивший отчет по работе, к следующей лабораторной работе не допускается.

Содержание отчета:

1. Титульный лист.
2. Цель работы.
3. Изображения лицевой панели прибора и структурной схемы.
4. Выводы по работе.

4. Контрольные вопросы и задания

1. В чем отличие программного пакета *LabView* от других языков программирования?

2. Объясните, как Вы понимаете сущность принципа потока данных.

3. Объясните назначение лицевой панели прибора и структурной схемы прибора.

4. Расскажите об основных рабочих инструментах в *LabView.*

5. Объясните по структурной схеме вашего виртуального прибора назначение его узлов, функций, органов управления и индикаторов, порядок работы виртуального прибора.

Лабораторная работа № 2

МОДЕЛИРОВАНИЕ РАБОТЫ БАЗОВЫХ ЭЛЕМЕНТОВ ЦИФРОВОЙ ТЕХНИКИ

Цель работы: изучение и моделирование работы простейших базовых логических элементов в среде *LabView*; ознакомление с типом данных *Boolean*; создание и использование библиотеки подпрограмм.

Оборудование: дисплейный класс, среда визуального программирования *LabView* версии 7.0 или выше.

1. Общие сведения

Логические элементы – это базовые блоки цифровых логических схем. Они могут открываться или закрываться, позволяя или отказывая пропускать логический сигнал. На основе небольшого количества основных логических элементов (И, ИЛИ, исключающее ИЛИ, НЕ) может быть построено огромное количество логических функций.

Логический элемент И. Базовый логический элемент И состоит из двух входов и выхода. Два входа назовем соответственно *A* и *B*. Выход назовем *Q*). Выход находится в состоянии «включено» только тогда, когда оба входа *A* и *B* находятся в состоянии «включено».

Табл. 2.1. Таблица истинности логического элемента И

A	*B*	*Q=A* И *B*
0	0	0
0	1	0
1	0	0
1	1	1

В цифровой электронике состояние «включено» обычно представляется в виде 1, а состояние «выключено» в виде 0. Соотношение между входными и выходными сигналами представляют в виде таблицы истинности, в которой сопоставляются все возможные состояния входов и результирующих выходов. Для логического элемента И существуют четыре возможные комбинации входного состояния: *A*=0, *B*=0; *A*=0, *B*=1; *A*=1, *B*=0 и *A*=1, *B*=1. Эти значения представлены в следующей таблице истинности в левом и среднем столбцах. Выход логического элемента И отображен в правом столбце.

В среде *LabView* можно определять состояние логического входа переключением логического выключателя, а логический светодиодный индикатор может показывать состояние выхода. Поскольку в среде *LabView* элемент И является одной из основных встроенных функций, то вы легко можете создать простейший виртуальный прибор, демонстрирующий работу этого логического элемента, присоединив два выключателя к его входу и светодиодный индикатор к его выходу.

Создайте виртуальный прибор моделирующий работу элемента И. Используйте для передней панели прибора элементы из группы «*Boolean*», для структурной схемы элемент «*AND*» из группы «*Boolean*». Нажимая на две входные кнопки, наблюдайте изменения выходного индикатора. Проверьте таблицу истинности, показанную выше.

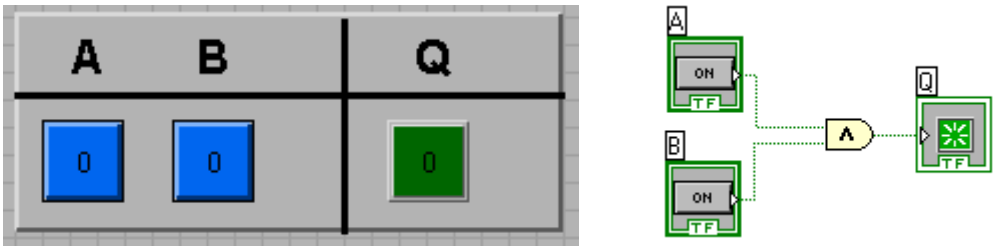


Рис. 2.1. Функция LabView «И», присоединенная к входным и выходному терминалам

Аналогичным образом может быть промоделирована работа элементов И, ИЛИ, НЕ, ИСКЛЮЧАЮЩЕЕ ИЛИ и т. д.

В пакете *LabView* содержаться все основные двухвходовые логические элементы, но вы можете использовать и больше входов. Из двух двухвходовых элементов вы можете построить виртуальный прибор, реализующий элемент И с тремя входами.

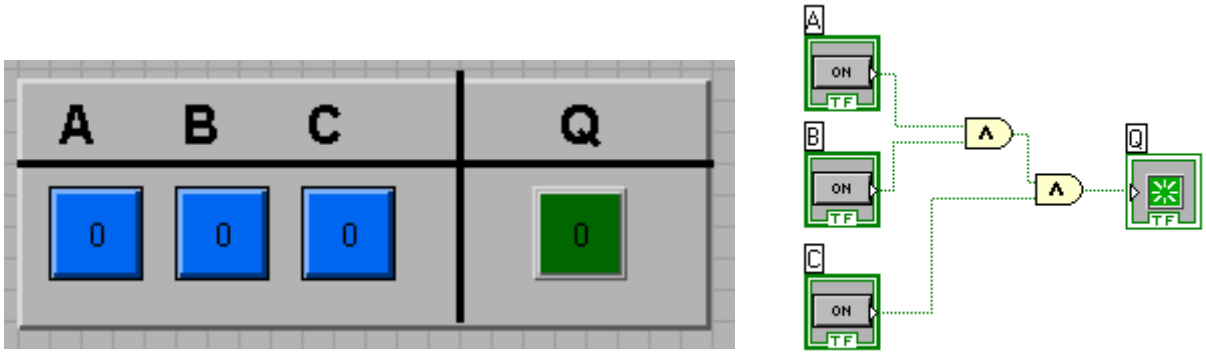


Рис. 2.2. Модель логического элемента И с тремя входами

Подпрограммы. Любая программа (*VI,* виртуальный прибор) может быть использована в блок-схеме другой программы как ее составная часть. Другими словами, она может быть, вложена как подпрограмма, *SubVI.* Эта особенность позволяет основной, главной программе быть модульной, быть легче читаемой и быть проще для понимания.

Рис. 2.3. Работа с подпрограммами через элемент Select a VI

Для вставки ранее разработанного *VI (SubVI)* в программу более высокого уровня необходимо использовать опции «*Select a VI...*» в палитре «*Function*» (рис. 2.3). В ответ на запрос диалогового окна необходимо выбрать файл подпрограммы и установить ее на диаграмме основной программы (рис. 2.4).

Главная программа может иметь множество вызовов подпрограммы.

На диаграмме главной программы подпрограмма появится в виде кубика со значком, заданным по умолчанию.

С помощью двойного щелчка можно открыть эту подпрограмму, а в случае необходимости и провести некоторые настройки.

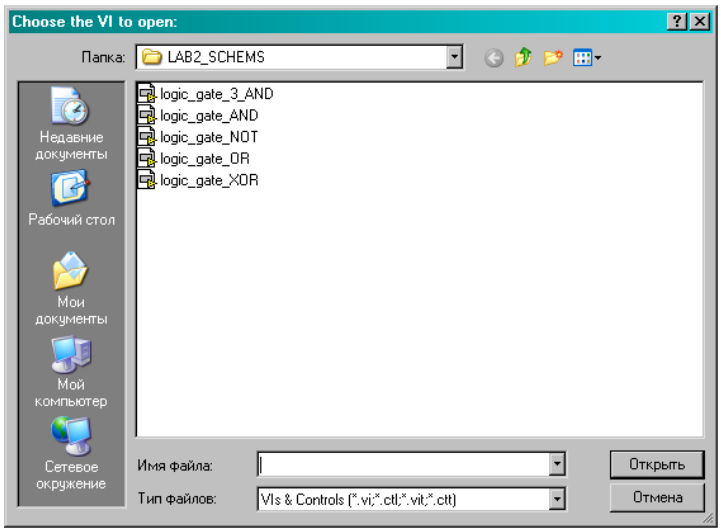


Рис. 2.4. Диалоговое окно выбора подпрограммы

Выше была создана программа, моделирующая работу логического элемента И. Теперь из этой программы создадим полнофункциональную подпрограмму.

Внешний вид лицевой панели и структурной схемы нашей подпрограммы показан на рис. 2.1.

Простое включение кубика подпрограммы ничего не даст главной программе, нет возможности что-то передать и что-то получить обратно. Чтобы подпрограмма получала данные из основной программы и передавала их назад в основную программу необходимо проделать определенные действия.

Сначала следует открыть лицевую панель нашей подпрограммы, и, разместив курсор в правом верхнем углу, на иконке, вызвать контекстное меню (рис. 2.5).

Выбрав в этом меню пункт «*Edit Icon*», можно отредактировать внешний вид иконки создаваемой подпрограммы (рис. 2.6).

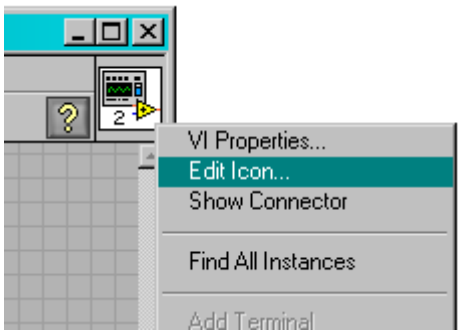


Рис. 2.5. Выбор режима редактирования иконки подпрограммы

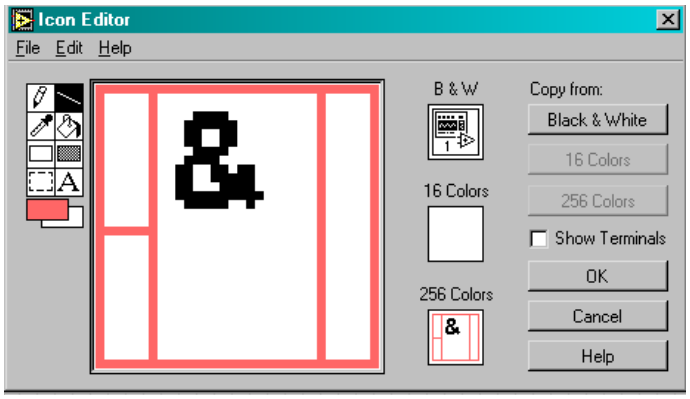


Рис. 2.6. Редактор изображения иконки

Далее снова следует вызвать контекстное меню (рис. 2.5) и выбрать пункт «*Show Connector*». Через этот пункт открывается доступ к параметрам подпрограммы – коннекторам или соединителям, элементам взаимодействия подпрограммы с внешним миром. Коннектор передает и принимает данные от вызывающей программы. Сразу после выбора этого пункта вид иконки нашей программы изменится (рис. 2.7), а курсор приобретет вид соединительной катушки.

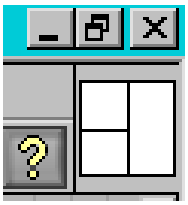


Рис. 2.7. Иконка программы в режиме Show Connector

LabView позаботится о том, чтобы появились коннекторы – квадратики и прямоугольник – по числу элементов управления и индикации на лицевой панели прибора. Теперь необходимо установить соответствия между коннектором и элементом управления или индикации на лицевой панели. Для этого необходимо указать курсором-катушкой элемент на лицевой панели и щелкнуть по коннектору-квадратику, за которым он будет закреплен. Признаком подсоединения будет изменение цвета этих квадратиков. В данном примере квадратики слева – это логические сигналы A и B, прямоугольник справа – результат логического умножения Q.

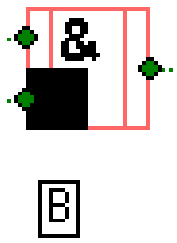


Рис. 2.8. Элемент подпрограммы в главной, вызывающей программе

Теперь подпрограмма, после ее установки в главную программу, даст возможность видеть входные и выходные переменные (рис. 2.8).

При очень большом количестве подпрограмм их целесообразно объединить в библиотеки, файлы с расширением «*LLB*».

Это можно сделать на этапе сохранения программы. В диалоге сохранения необходимо выбрать кнопку «*New VI Library*», указав после нажатия этой кнопки имя новой библиотеки, а далее сохранить текущий файл в созданной библиотеке.

Для обслуживания библиотек может быть использован пункт меню *LabView «LabView VI library manager*».

2. Выполнение работы

Задание 1.

Создайте виртуальный прибор, демонстрирующий работу основных логических элементов, лицевая панель прибора должна быть исполнена примерно так, как показано на рис. 2.9.

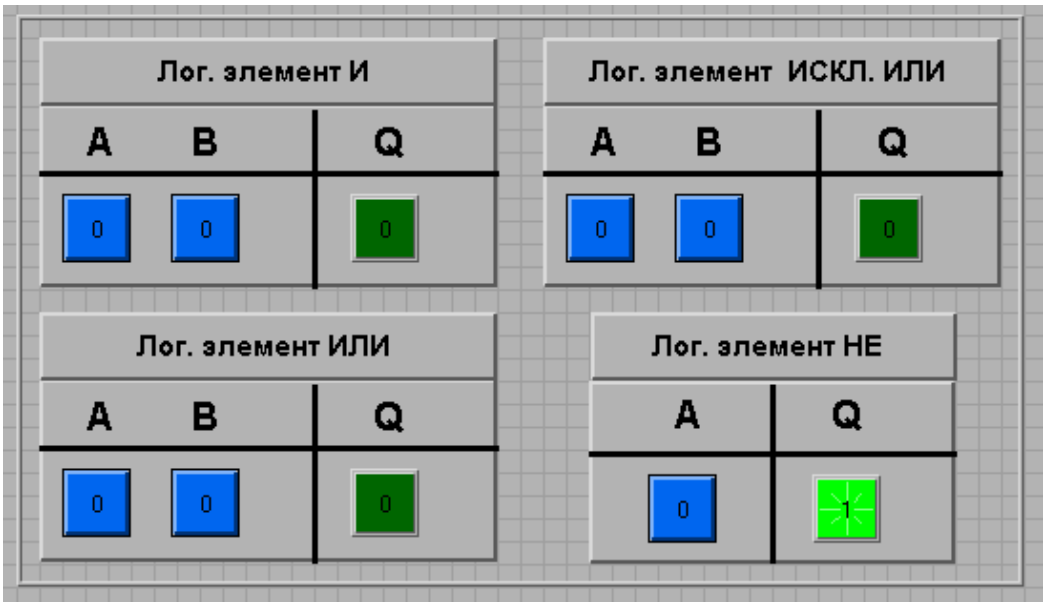


Рис. 2.9. Возможный вид лицевой панели разрабатываемого прибора

Для отчета сохраните снимок экрана (*screenshot*) лицевой панели прибора и структурной схемы прибора. Разработанный виртуальный

прибор сохраните на своем носителе информации для его демонстрации при защите лабораторной работы.

Задание 2.

Оформите созданные вами виртуальные приборы – И, ИЛИ, НЕ, ИСКЛ. ИЛИ в виде подпрограмм и сохраните их в библиотеке с именем «*lab_2_library.llb*». Данная библиотека понадобится на следующих лабораторных работах, сохраните ее на своем носителе информации.

Задание 3.

Реализуйте логическую функцию:

$$Y = \overline{(A \cdot \overline{B}) \cdot (\overline{A} \cdot B)}.$$

При ее реализации использовать только *SubVI*, созданные в ходе выполнения задания 2. Лицевая панель прибора должна содержать два переключателя и один элемент индикации типа «*Boolean*».

Для отчета сохраните снимок экрана (*screenshot*) лицевой панели прибора и структурной схемы прибора. Разработанный виртуальный прибор сохраните на своем носителе информации для его демонстрации при защите лабораторной работы. Определите, какую из изученных простейших логических функций реализует данный прибор.

3. Содержание отчета

Отчет оформляется каждым студентом самостоятельно. Защита проходит в начале каждого следующего занятия с демонстрацией работы программы на ЭВМ. Студент, не подготовивший или не защитивший отчет по работе, к следующей лабораторной работе не допускается.

Содержание отчета:

1. Титульный лист.
2. Цель работы.
3. Виртуальный прибор, демонстрирующий работу логических элементов И, ИЛИ, НЕ, ИСКЛ. ИЛИ; лицевая панель данного виртуального прибора и структурная схема.
4. Виртуальный прибор, реализующий логическую функцию задания 3, его лицевая панель и структурная схема.
5. Выводы по работе.

4. Контрольные вопросы и задания

1. В данной работе все виртуальные приборы оперируют с типом данных «*Boolean*», что это за тип данных?

2. Зачем нужно создавать подпрограммы (*SubVI*)? Какие преимущества они дают?

3. Какие из базовых логических функций уже реализованы в *LabView*?

4. Зачем нужны библиотеки подпрограмм? Можно ли без них обойтись?

5. Сколько раз основная программа может иметь в своем теле вызовов подпрограмм?

6. Может ли основная программа при вызове подпрограммы передавать туда какие-либо данные и получать их назад?

7. На структурных схемах приборов в данной лабораторной работе вы видели соединительные провода зеленого цвета, что он обозначает в *LabView*?

Лабораторная работа № 3
МОДЕЛИРОВАНИЕ РАБОТЫ КОМБИНАЦИОННЫХ ЦИФРОВЫХ УСТРОЙСТВ

Цель работы: изучение и моделирование работы комбинационных цифровых устройств в среде *LabView*: шифратор, дешифратор, мультиплексор, демультиплексор; создание библиотеки подпрограмм данных элементов; изучение числовых типов данных; конвертация одного типа данных в другой; использование массивов и кластеров; изучение структуры «Вариант».

Оборудование: дисплейный класс, среда визуального программирования *LabView* версии 7.0 или выше.

1. Общие сведения

Логические устройства разделяют на два класса: комбинационные и последовательные.

Устройство называют комбинационным, если его выходные сигналы в некоторый момент времени однозначно определяются входными сигналами, имеющими место в этот момент времени.

Иначе устройство называют последовательным или конечным автоматом (цифровым автоматом, автоматом с памятью). В последовательных устройствах обязательно имеются элементы памяти. Состояние этих элементов зависит от предыстории поступления входных сигналов. Выходные сигналы последовательных устройств определяются не только сигналами, имеющимися на входах в данный момент времени, но и состоянием элементов памяти. Таким образом, реакция последовательного устройства на определенные входные сигналы зависит от предыстории его работы.

Дешифратор. Дешифратором называется комбинационное устройство, преобразующее *n*-разрядный двоичный код в логический сигнал, появляющийся на том выходе, десятичный номер которого соответствует двоичному коду. Число входов и выходов в так называемом полном дешифраторе связано соотношением $m=2^n$, где n − число входов, а m − число выходов.

Работу дешифратора с тремя входами и восемью выходами можно представить следующей таблицей истинности:

Табл.3.1. Таблица истинности дешифратора

X0	X1	X2	Y0	Y1	Y2	Y3	Y4	Y5	Y6	Y7
0	0	0	1	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	0	0
0	1	1	0	0	0	1	0	0	0	0
1	0	0	0	0	0	0	1	0	0	0
1	0	1	0	0	0	0	0	1	0	0
1	1	0	0	0	0	0	0	0	1	0
1	1	1	0	0	0	0	0	0	0	1

Для создания подобного виртуального дешифратора в *LabView* расположите на лицевой панели три переключателя и восемь светодиодных индикаторов (рис. 3.1). Переключатели будут моделировать входной цифровой код поступающий на дешифратор, а светодиодные индикаторы – выходной сигнал с дешифратора.

Реализация структурной схемы дешифратора возможна несколькими способами. Первый способ – создать структурную схему данного прибора, основываясь на базовых логических элементах изученных в лабораторной работе № 1 и таблице истинности данного устройства (рис. 3.2).

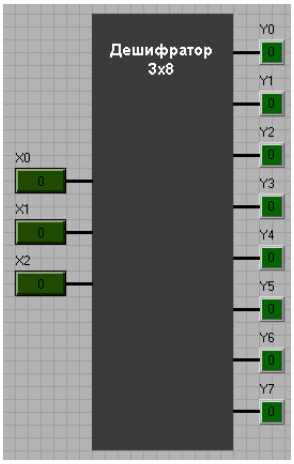


Рис. 3.1. Дешифратор 3x8, лицевая панель прибора

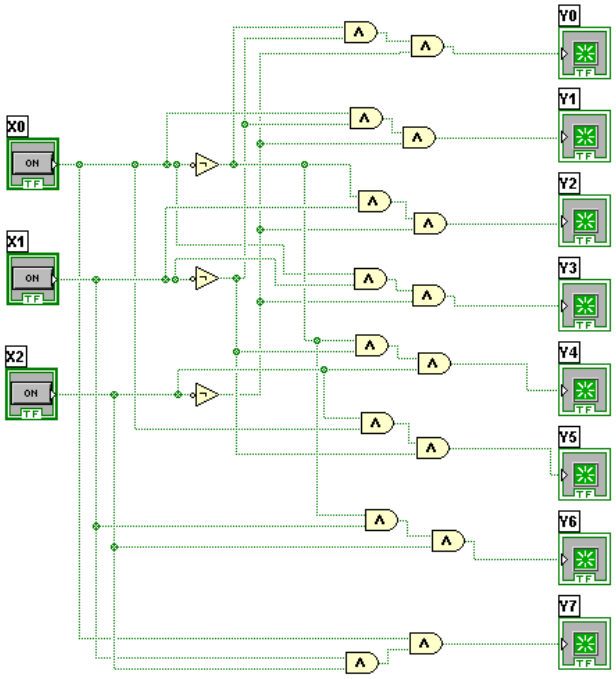


Рис. 3.2. Дешифратор 3x8 на логических элементах

Второй способ – при создании структурной схемы прибора использовать управляющие структуры *LabView* («*Case Structure*») (рис. 3.3). Структура *LabView* «Вариант» («*Case Structure*») – управляет выполнением одного из двух или более фрагментов кода и при выборе по условию аналогична оператору «*If-Then-Else*» текстовых языков программирования, а при выборе по значению числовой или строковой переменной аналогична оператору «*Case*». Структура может иметь внутри себя одну или более поддиаграмм-условий, которые будут работать при выполнении заданных пользователем условий.

На структурной схеме, представленной на рис. 3.3, сигнал с входных терминалов с помощью функционального узла «*Boolean to 0,1*» (рис. 3.4, *а*) конвертируется в числовой тип, далее с помощью узла «*Multiply*» (рис. 3.4, *б*) умножается на весовую константу бита входного сигнала, и в блоке «*Compound Arithmetics*» (рис. 3.4, *в*) складывается. На вход структуры «Вариант» подается целое двоичное число, которое закодировано битами входных терминалов прибора.

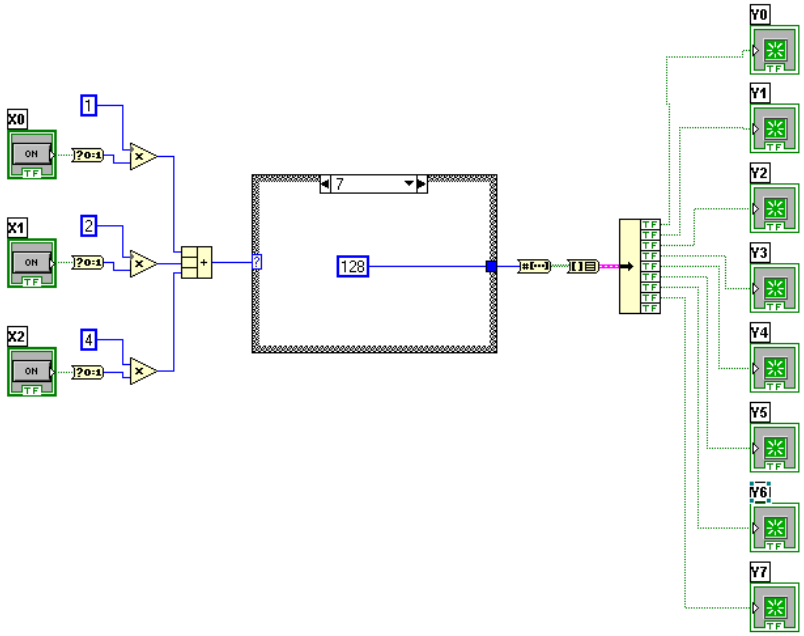
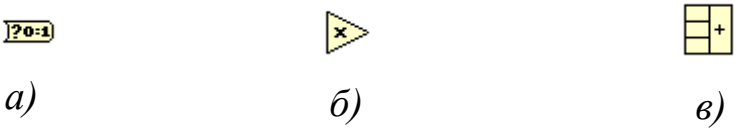


Рис. 3.3. Дешифратор 3х8 с использованием структуры «Вариант»



а) *б)* *в)*

Рис. 3.4. Входные компоненты и дешифраторы: а − функциональный узел «Boolean to (0,1)»; б − блок арифметического умножения «Multiply»;в − блок составная арифметика «Compound Arithmetic»

Далее сигнал поступает в структуру «Вариант», в которой задано восемь условий, для каждого из восьми возможных чисел на входе. На выход структуры поступает двоичное число, соответствующее отображению в восьми выходных битах для одного из случаев входной комбинации бит.

Для того чтобы отобразить это число на светодиодных индикаторах, оно из двоичного представления числа преобразуется в одномерный массив бит (рис. 3.5, *а*), который преобразуется в кластер, состоящий из 8 бит (рис. 3.5, *б*), далее в блоке «*Unbundle*» (рис. 3.5, *в*) кластер разбивается на составляющие его потоки бит, которые поступают на выходные светодиодные индикаторы.

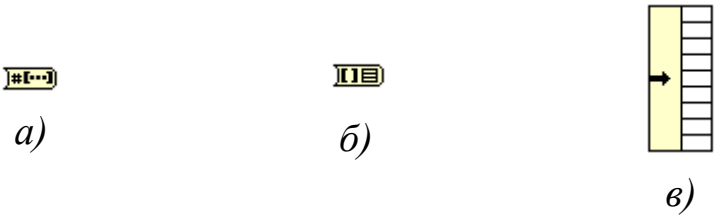


Рис. 3.5. Выходные компоненты и дешифраторы: а − «Number to boolean array»; б − «Array to cluster»; в − «Unbundle»

Хотя в данном случае второй способ создания дешифратора оказывается более сложным, он дает возможность понять основные приемы работы с логическими и целыми типами данных, способы их конвертирования из одного типа в другой; также здесь вводится понятие массива и кластера и приемы работы с управляющей структурой «Вариант».

Аналогичным образом можно создать виртуальные приборы − шифратор, мультиплексор и демультиплексор.

2. Выполнение работы

Задание 1.

Получите у преподавателя виртуальный прибор − дешифратор, описанный выше. Изучите работу данного прибора для обоих случаев его реализации. Изучите назначение всех новых для вас функциональных узлов использованных на структурной схеме и поймите принципы их функционирования.

Для отчета сделайте снимки экрана (*screenshot*) лицевой панели и структурной схемы данного прибора. В отчете также дайте описание всех функциональных узлов, представленных на структурной схеме данного прибора.

Задание 2.

Создайте виртуальные приборы – шифратор, мультиплексор и демультиплексор. При создании шифратора используйте первый способ реализации описанный выше. При создании мультиплексора используйте второй способ реализации описанный выше. При создании демультиплексора используйте любой из способов реализации или придумайте свой способ реализации данного прибора.

Задание 3.

Оформите все созданные виртуальные приборы – дешифратор, шифратор, мультиплексор, демультиплексор в виде подпрограмм и сохраните их в библиотеке с именем «*lab_3_library.llb*». Данная библиотека понадобится на следующих лабораторных работах, сохраните ее на своем носителе информации.

3. Содержание отчета

Отчет оформляется каждым студентом самостоятельно. Защита проходит в начале каждого следующего занятия с демонстрацией работы программы на ЭВМ. Студент, не подготовивший или не защитивший отчет по работе, к следующей лабораторной работе не допускается.

Содержание отчета:

1. Титульный лист.
2. Цель работы.
3. Виртуальный прибор − дешифратор с описанием всех элементов его структурной схемы, лицевая панель и структурная схема прибора.
4. Виртуальный прибор, реализующий шифратор, лицевая панель и структурная схема прибора, пояснения к его работе, если требуется.
5. Виртуальный прибор, реализующий мультиплексор, лицевая панель и структурная схема прибора, пояснения к его работе, если требуется.
6. Виртуальный прибор, реализующий демультиплексор, лицевая панель и структурная схема прибора, пояснения к его работе, если требуется.
7. Библиотека подпрограмм с реализованными устройствами.
8. Выводы по работе.

4. Контрольные вопросы и задания

1. С какими типами данных вы оперировали в данной работе?

2. Какой тип данных обозначается на рис. 3.3 синими проводниками? Какой тип данных обозначается жирным розовым проводником? Какой тип данных обозначается жирным зеленым проводником?

3. Каково назначение функционального узла «*Multiply*»? Каково назначение функционального узла «*Compound Arithmetic*»?

4. Каково назначение функционального узла «*Unbundle*»? Каково назначение функционального узла «*Array to cluster*»?

5. Каково назначение функциональных узлов «*Boolean to* (0,1)», «*Number to Boolean array*»?

6. Что вы знаете о структуре «Вариант» («*Case structure*»)?

7. Для какого типа данных производила выбор условий для дешифратора в данной работе структура «Вариант»? Сколько условий было реализованной в структуре «Вариант»?

Лабораторная работа № 4
МОДЕЛИРОВАНИЕ РАБОТЫ СЕМИСЕГМЕНТНЫХ ИНДИКАТОРОВ

Цель работы: изучение и моделирование работы семисегментных индикаторов в среде *LabView*; создание «вспомогательных устройств» для индикации; создание библиотеки подпрограмм данных элементов; изучение структуры «*While Loop*»; изучение сдвиговых регистров «*Shift Register*».

Оборудование: дисплейный класс, среда визуального программирования *LabView* версии 7.0 или выше.

1. Общие сведения

Цифровые дисплеи являются связующим звеном между цифровым миром нулей и единиц и числами, с которыми работают люди. В наиболее простых приборах используются цифровые дисплеи, состоящие из семи индикаторов (сегментов), для представления чисел от 0 до 9. Каждый сегмент управляется одним битом. Комбинация включенных и выключенных сегментов может изобразить все числа от 0 до 9 и еще ряд букв, таких как *A, B, C, D, E* и *F*.

Семисегментный индикатор. В дисплее с семью сегментами используются семь независимых светоизлучающих диода, сконфигурированных в виде числа 8 (рис. 4.1).

Независимые сегменты расположены по часовой стрелке и обозначены буквами *a, b, c, d, e, f*. Последний сегмент (*g*) представляет собой центральную перекладину. Когда на светодиод приложено прямое смещение, он излучает свет. Сам сегмент образуется расположением светодиода вертикально или горизонтально. Многие выходные устройства, такие как параллельные порты компьютера, работают с восьмиразрядными сигналами. В некоторых семисегментных дисплеях употребляется восьмой светодиод в форме точки для индикации разделения десятичных разрядов.

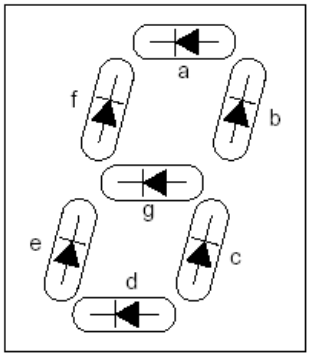


Рис. 4.1. Семисегментный дисплей, собранный из семи светодиодных индикаторов

Для моделирования семисегментных индикаторов в среде *LabView* можно использовать любые понравившиеся светодиодные индикаторы из группы элементов «*Boolean*» (рис. 4.2).

Рис. 4.2. Модель семисегментного дисплея

Семисегментный дешифратор. Большинство дисплеев с семью сегментами управляются семисегментными дешифраторами, которые преобразовывают двоично-кодированный полубайт в подходящий семисегментный код. Для чисел от 10 до 15 используются буквы от *A* до *F* шестнадцатеричного представления.

В среде *LabView* для создания семисегментного дешифратора существует множество вариантов. Можно составить таблицу истинности его функционирования и по ней, используя базовые логические элементы собрать его схему. Но гораздо проще использовать структуру «Вариант». Один из вариантов создания семисегментного дешифратора показан на рис. 4.3.

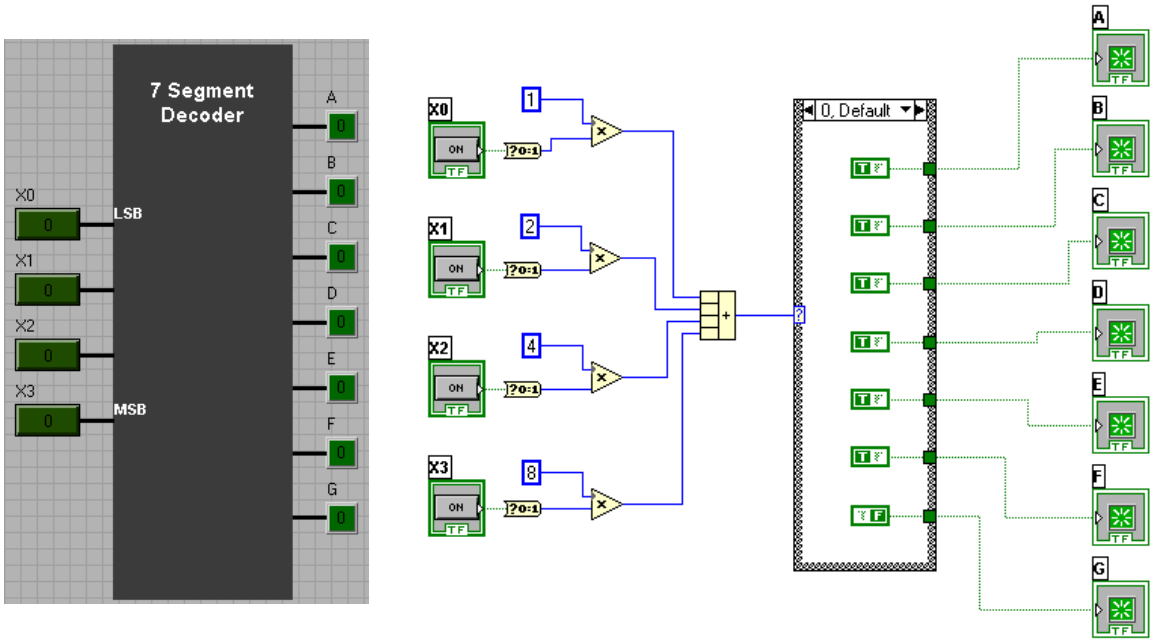


Рис. 4.3. Семисегментный дешифратор

Триггеры и регистры. Обычно то, что отображается на семисегментном индикаторе где-то хранится в данном устройстве. В общем случае это может быть оперативная память (ОЗУ) или регистр. При измерении различных сигналов результаты измерения, также обрабатываются и сохраняются в устройстве для дальнейшего их отображения на индикаторах или передачи в другое устройство. Во всех таких случаях мы говорим об устройствах с памятью − триггеры, регистры, ОЗУ.

В предыдущих работах рассмотрены комбинационные схемы, в которых входное состояние полностью определяло выходное. Поведение таких схем не зависит от предыстории, или по-другому, от того, каким образом вы создали текущее состояние. Это значит, что в подобные схемы нельзя встроить функцию запоминания.

Одной из простейших запоминающих схем является «защелка» данных или регистр *D*-типа. Когда на синхронизирующий вход такого устройства приходит сигнал, оно запоминает состояние на своем входе и передает это состояние на выход. Даже в том случае, если входной сигнал изменяется, выходное состояние остается неизменным до тех пор, пока не поступит запрос на обновление. Обычно вход регистра *D*-типа обозначают буквой *D*, а выход буквой *Q*. Команда обновления поступает через синхронизирующий вход в форме изменения уровня сигнала от высокого к низкому или от низкого к высокому. Такие устрой-

ства называются устройствами, тактируемые фронтом сигнала (синхронный динамический *D*-триггер). Синхронные статические триггеры воспринимают информационные сигналы при подаче на вход *C* логической единицы или логического нуля.

В пакете *LabView* вы можете смоделировать регистры *D*-типа, используя сдвиговые регистры в цикле по условию («*While loop*», «*Shift register*»). Пример виртуального прибора, моделирующего работу *D*-триггера, показан на рис. 4.4.

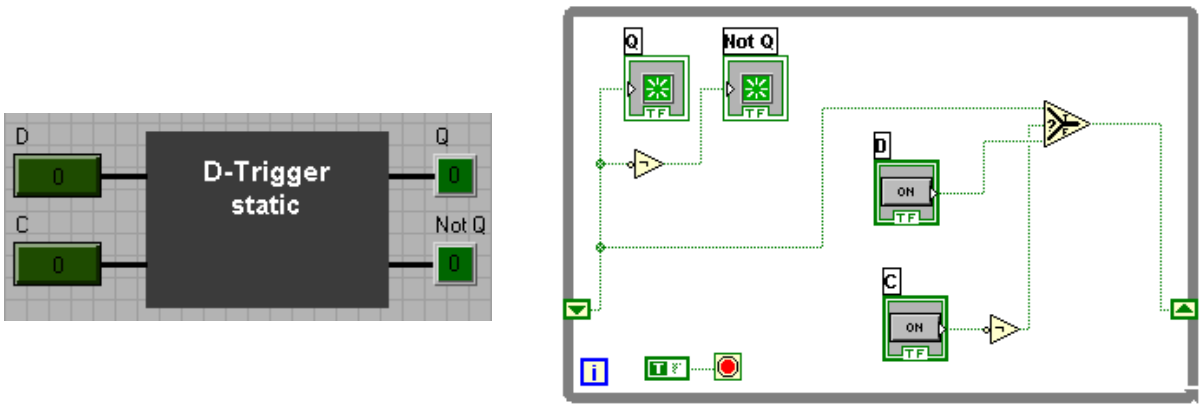


Рис. 4.4. Модель регистра D-типа

Если вы захотите, следуя законам цифровой техники, собрать *D*-триггер из базовых логических элементов введя линии обратной связи, то у вас ничего не получится. Среда *LabView* не разрешит создать соединения такого типа.

Для обхода данного «ограничения» в виртуальном приборе, изображенном на рис. 4.4, использовалась структура «цикл по условию» («*While loop*»), которая осуществляет итерационное выполнение кода внутри данной структуры до выполнения заданного условия (рис. 4.5).

Рис. 4.5. Структура «While loop» со сдвиговыми регистрами «Shift register»

Особое место в использовании структуры «*While loop*» занимают сдвиговые регистры «*Shift register*», которые используются для передачи результатов вычисления переменных от текущей итерации к последующей или от последующей к предыдущей. Направление передачи опре-

деляется значком треугольника в его обозначении, своего рода указующей стрелки передачи результатов.

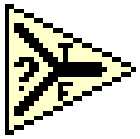


Рис. 4.6. Функциональный узел выбора «Select»

В виртуальном приборе на рис. 4.4 на «выходной сдвиговый регистр» (стрелка на правой части структуры «*While loop*») подается сигнал, в который установится триггер, который зависит от состояния входов *D* и *C*, а также от предыдущего состояния триггера, которое получается из «входного сдвигового регистра (стрелка на левой части структуры «*While loop*»). В узле выбора «*Select*» (рис. 4.6), непосредственно происходит выбор, того сигнала, который поступит на выход триггера в зависимости от предыдущего состояния триггера и входов *D* и *C*.

Для создания в среде *LabView* регистра, используется тот же принцип, что и при создании триггеров. Функциональные схемы приборов отличаются лишь, количеством сдвиговых регистров («*Shift register*») и количеством входных и выходных терминалов. Пример построения 4-битного регистра показан на рис. 4.7.

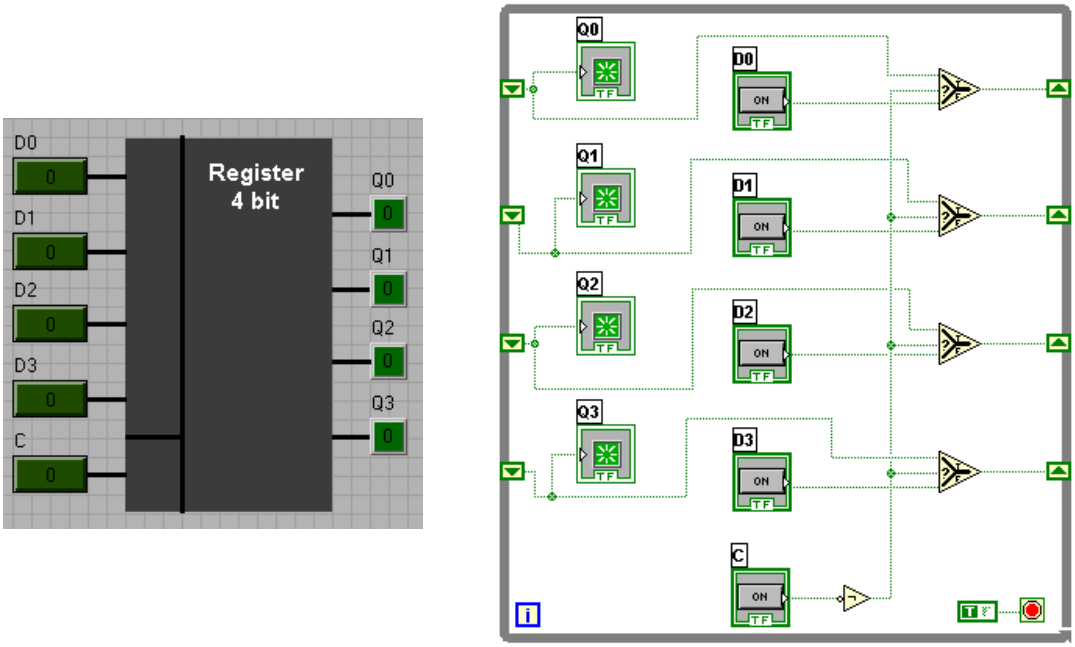


Рис. 4.7. 4-битный регистр

2. Выполнение работы

Задание 1.

Создайте одну или несколько моделей семисегментного цифрового дисплея показанного на рис. 4.2. Для отчета сделайте снимки экрана (*screenshot*) лицевой панели и структурной схемы данного прибора. В отчете также дайте описание всех узлов структурной схемы прибора. Создайте библиотеку подпрограмм с именем «*lab_4_library.llb*» и сохраните в ней созданный виртуальный прибор.

Задание 2.

Создайте виртуальный прибор – семисегментный дешифратор, показанный на рис. 4.3. Изучите принципы работы новых для вас функциональных узлов данного устройства и дайте их описание в отчете. Для отчета сделайте снимки экрана (*screenshot)* лицевой панели и структурной схемы данного прибора. Оформите созданный виртуальный прибор в виде подпрограммы и сохраните его в библиотеке подпрограмм.

Задание 3.

Создайте виртуальный прибор – триггер *D*-типа; 4 битный и 8 битный регистры на подобие тех, которые показанные на рис. 4.4 и 4.7. Изучите принципы работы новых для вас функциональных узлов данных устройств и дайте им описание в отчете. Для отчета сделайте снимки экрана (*screenshot*) лицевой панели и структурной схемы данных приборов. Оформите созданные виртуальные приборы в виде подпрограммы и сохраните их в библиотеке подпрограмм.

Задание 4.

Для демонстрации работоспособности созданных виртуальных приборов для заданий 1 и 2, создайте прибор, показанный на рис. 4.8 и продемонстрируйте его работу преподавателю.

Сигнал с входных терминалов поступает на семисегментный дешифратор, где преобразуется в семисегментный код для отображения на выходных терминалах.

Для отчета сделайте два или более снимков экрана (*screenshot*) лицевой панели с индикацией цифры 7 и буквы b. Также сделайте снимки экрана (*screenshot*) структурной схемы прибора.

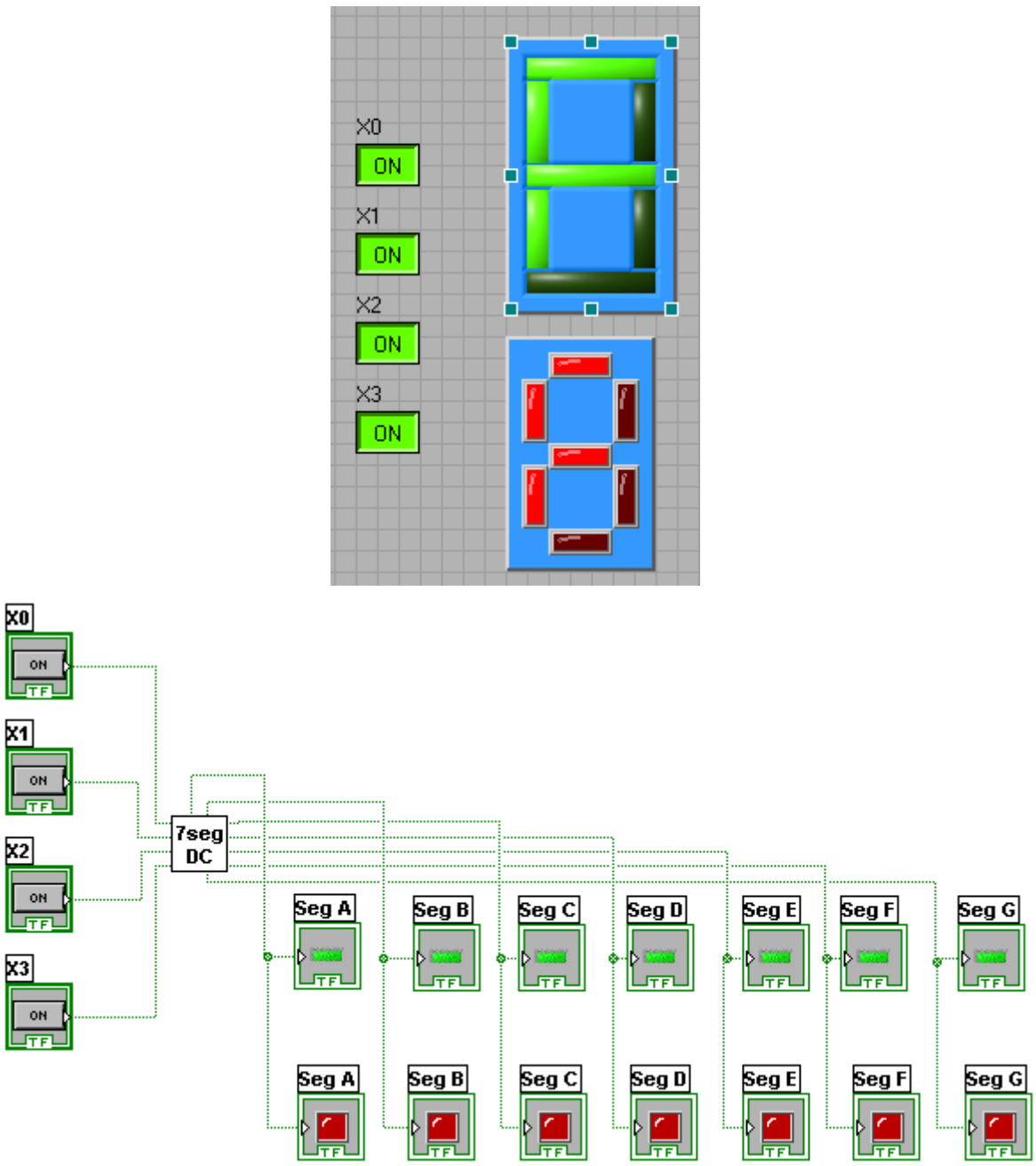


Рис. 4.8. Цифровая индикация состояния входных битов

Задание 5.

Создайте виртуальный прибор, который показан на рис. 4.9 или подобный.

С помощью слайдера («*slide*») выбирается значение в диапазоне 0 − 255, которое при переключении кнопки «*Enter value*» записывается в 8-битный регистр и далее отображается на семисегментных индикаторах в шестнадцатеричном коде.

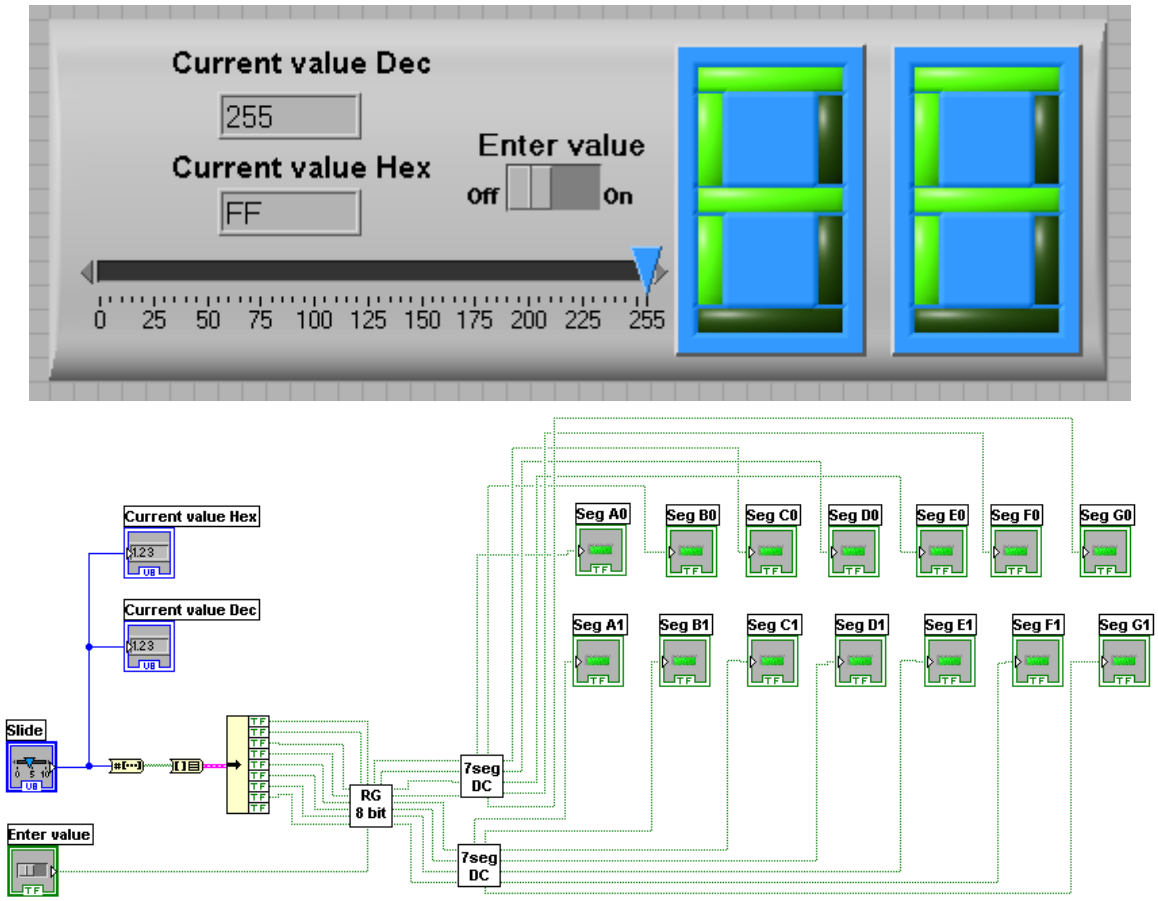


Рис. 4.9. Индикация целого числа записанного в 8-битном регистре без знака

При создании прибора используйте ранее созданные вами подпрограммы.

Для отчета сделайте три или более снимков экрана (*screenshot*) лицевой панели с индикацией значения 127, 63 и 31. Также сделайте снимки экрана (*screenshot*) структурной схемы прибора.

3. Содержание отчета

Отчет оформляется каждым студентом самостоятельно. Защита проходит в начале каждого следующего занятия с демонстрацией работы программы на ЭВМ. Студент, не подготовивший или не защитивший отчет по работе, к следующей лабораторной работе не допускается.

Содержание отчета:

1. Титульный лист.
2. Цель работы.

3. Лицевая панель приборов и структурная схема приборов − семисегментных индикаторов созданных в задании 1.

4. Лицевая панель прибора и структурная схема прибора − семисегментного дешифратора созданного в задании 2; описание новых функциональных узлов прибора.

5. Лицевая панель приборов и структурная схема приборов − *D*-триггера, 4-битного и 8-битного регистров созданных в задании 3; описание новых функциональных узлов приборов.

6. Лицевая панель прибора и структурная схема прибора, изображенного на рис. 4.8.

7. Лицевая панель прибора и структурная схема прибора изображенного на рис. 4.9 – три случая отображения данных; объяснения принципов работы функциональных узлов прибора.

8. Библиотека подпрограмм с реализованными устройствами.

9. Выводы по работе.

4. Контрольные вопросы и задания

1. С какими типами данных вы оперировали в данной работе?

2. Какой тип данных обозначается на рис. 4.3 синими проводниками? Какая структура изображена на данном рисунке?

3. Дайте описание всем функциональным узлам, изображенным на рис. 4.3.

4. Зачем нужна структура «*While loop*»? Принципы ее функционирования.

5. Что вы знаете про сдвиговые регистры «*Shift register*»? Как и зачем вы их применяли?

6. С какими типами данных вы работали, создавая виртуальный прибор, изображенный на рис. 4.9? Что нужно сделать, чтобы отобразить заданное число на семисегментных индикаторах в десятичном коде?

7. Вы уже изучили структуры «*Case structure*» и «*While loop*», какие еще структуры *LabView* вы можете назвать? Принципы их функционирования.

Лабораторная работа № 5
МОДЕЛИРОВАНИЕ РАБОТЫ АЦП И ЦАП

Цель работы: изучение и моделирование работы ЦАП и АЦП в среде *LabView*; создание библиотеки подпрограмм данных элементов; изучение вещественного типа данных; ознакомление с осциллографами в *LabView*.

Оборудование: дисплейный класс, среда визуального программирования *LabView* версии 7.0 или выше.

1. Общие сведения

Цифро-аналоговый преобразователь (сокращенно ЦАП) – это одна из наиболее важных схем сопряжения, применяемых для организации связи между аналоговыми и цифровыми устройствами. ЦАП является основой многих электронных схем и устройств, включая цифровые вольтметры, графопостроители, осциллографы и многие другие устройства, управляемые компьютером.

ЦАП – это электронное устройство, преобразующее цифровой логический сигнал в аналоговый сигнал. Выходное напряжение ЦАП − это набор битов входного сигнала, взвешенных определенным образом:

$$DAC = \sum_{i=0} w_i b_i ,$$

где w_i – весовой коэффициент, b_i – значение бита (1 или 0), i – индекс номера бита. В случае двоичной схемы взвешивания, когда $w_i = 2^i$, полное выражение для 8-битового ЦАП запишется в виде:

$$DAC = 128 \cdot b_7 + 64 \cdot b_6 + 32 \cdot b_5 + 16 \cdot b_4 + 8 \cdot b_3 + 4 \cdot b_2 + 2 \cdot b_1 + 1 \cdot b_0.$$

В модели (рис. 5.1), виртуальный прибор демонстрирует процесс преобразования. Восемь булевых переключателей на лицевой панели устанавливают входные биты от b_0 до b_7. Когда программа запущена, восемь светодиодных индикаторов отображают величину входного байта. Выходной сигнал отображается в виде числового индикатора. На диаграммной панели виден алгоритм работы 8-битного конвертера, реализованного при помощи пакета *LabView*.

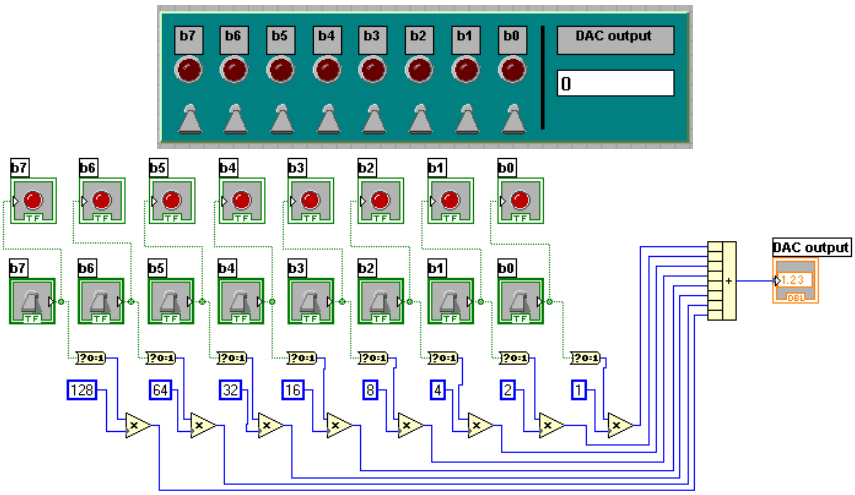


Рис. 5.1. Модель 8-битного ЦАП

Для генерации аналогового сигнала можно использовать любую последовательность битов, подаваемых с постоянной скоростью на вход ЦАП. Простейшая последовательность получается на выходе 8-битного двоичного счетчика. С ее помощью генерируется цифровой сигнал в пределах от 0 до 255 единиц. Для демонстрации такого процесса необходимо присоединить простой счетчик к ЦАП. После этого выходной сигнал подается на графический индикатор. Скорость нарастания графика определяется частотой счета: больше частота – больше угол наклона. Колебательный модуль генерирует тактовый сигнал. Когда происходит переполнение счетчика от (11111111) до (00000000), аналоговый сигнал быстро спадает от 255 до 0. Такой сигнал называют ступенчатым, поскольку он похож на ступеньки лестницы. Модель такого прибора представлена на рис. 5.2.

Счетчик от 0 до 255 организован с помощью сдвиговых регистров («*Shift register*»), и узла сравнения текущего значения счетчика со значением 255. Сигнал с этого счетчика преобразуется в массив бит, а затем в кластер бит, который разбивается на потоки бит и подается на ЦАП.

С выхода ЦАП снимается «аналоговый» сигнал отображаемый на графическом индикаторе «*Waveform Chart*», для режимов 4- и 6-битного ЦАП этот сигнал умножается на константу, чтобы привести его к масштабу 0÷255. Значок с часами – это функциональный узел «*Wait*» – обеспечивает задержку выполнения кода программы на указанное количество миллисекунд.

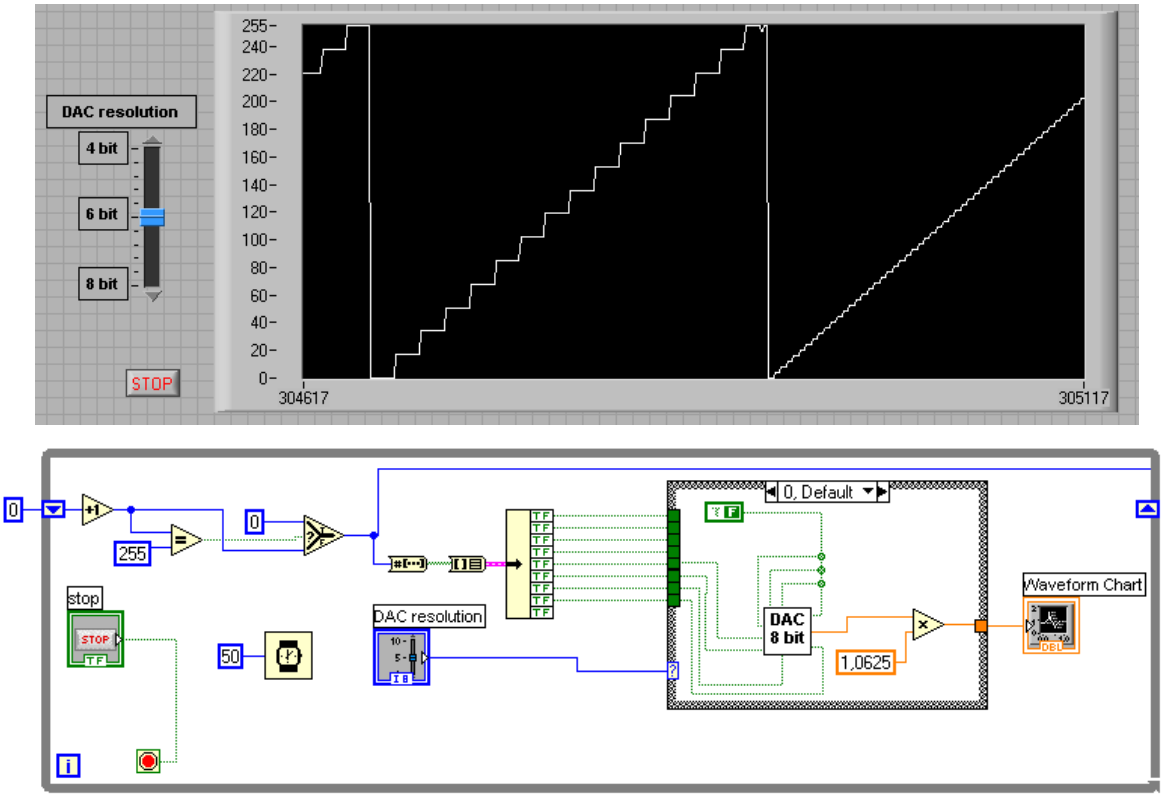


Рис. 5.2. Модель сигнала 4,6 и 8-битных ЦАП

Аналогово-цифровой преобразователь (сокращенно АЦП) – это второй ключевой элемент, обеспечивающий взаимодействие аналоговых и цифровых устройств. АЦП является основой цифровых вольтметров, цифровых авометров, многоканальных анализаторов, осциллографов и многих других приборов. Существует несколько различных типов АЦП. Наиболее распространенными являются интегрирующие, следящие и преобразователи последовательного приближения. В работе исследуются интегрирующие и следящие аналого-цифровые преобразователи.

Назначение АЦП заключается в генерации двоичного цифрового кода, пропорционального входному аналоговому сигналу (рис. 5.3).

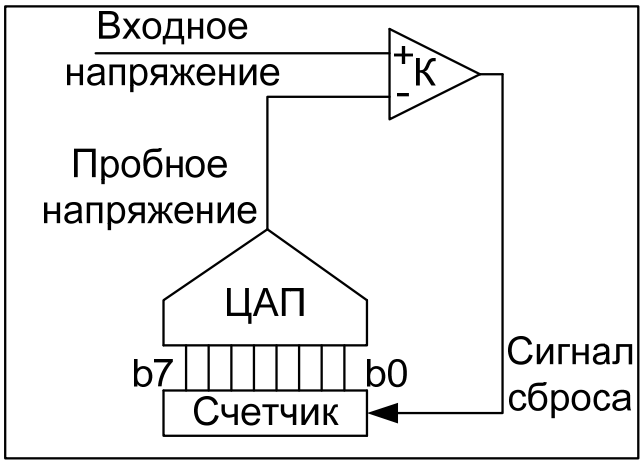


Рис. 5.3. Схема 8-битного аналого-цифрового преобразователя

Счетчик создает пробную двоичную последовательность, которая конвертируется в аналоговое напряжение при помощи цифро-аналогового преобразователя. ЦАП является базовым элементом многих схем АЦП, а его принцип работы уже обсуждался выше. После этого пробное напряжение сравнивается с входным сигналом. Если входное напряжение больше пробного сигнала, счетчик увеличивает значение, чтобы приблизить пробный сигнал к уровню входного напряжения. Если же входной сигнал меньше пробного, счетчик уменьшает свое выходное значение с тем, чтобы уровень пробного сигнала приблизился к уровню входного. Этот процесс продолжается до тех пор, пока компаратор не изменит знак. В этот момент уровень пробного сигнала будет в пределах одного отсчета от уровня входного напряжения. При увеличении числа разрядов счетчика будет увеличиваться и разрешение ЦАП такого типа.

Интегрирующий АЦП. В интегрирующем АЦП для генерации пилообразного пробного напряжения используются двоичный счетчик и цифро-аналоговый преобразователь. Модель прибора представлена на рис. 5.4.

Двоичный счетчик работает в свободном режиме. Всякий раз, когда пробный сигнал становится больше входного, компаратор меняет знак. Такое пересечение пилообразного сигнала с входным напряжением можно наблюдать на графическом индикаторе. Двоичная величина счетчика в точке пересечения – это оцифрованный сигнал. Изменение состояния компаратора свидетельствует о пересечении.

Чтобы смоделировать действительно пилообразный сигнал АЦП, необходимо, чтобы при изменении состояния компаратора происходил возврат счетчика в исходное состояние. Для этого простой двоичный счетчик заменяется на двоичный счетчик с обнулением. Как только уровень пробного сигнала достигает входного, двоичный счетчик возвращается в исходное состояние, а пилообразный цикл начинается снова. На индикаторе, показанном выше, уровень входного сигнала менялся трижды.

Следящие АЦП. Основная задача следящего АЦП – быстро приблизиться к уровню входного сигнала. В точке, определяемой пересечением пилообразного и входного сигналов, следящий алгоритм заканчивает свою работу.

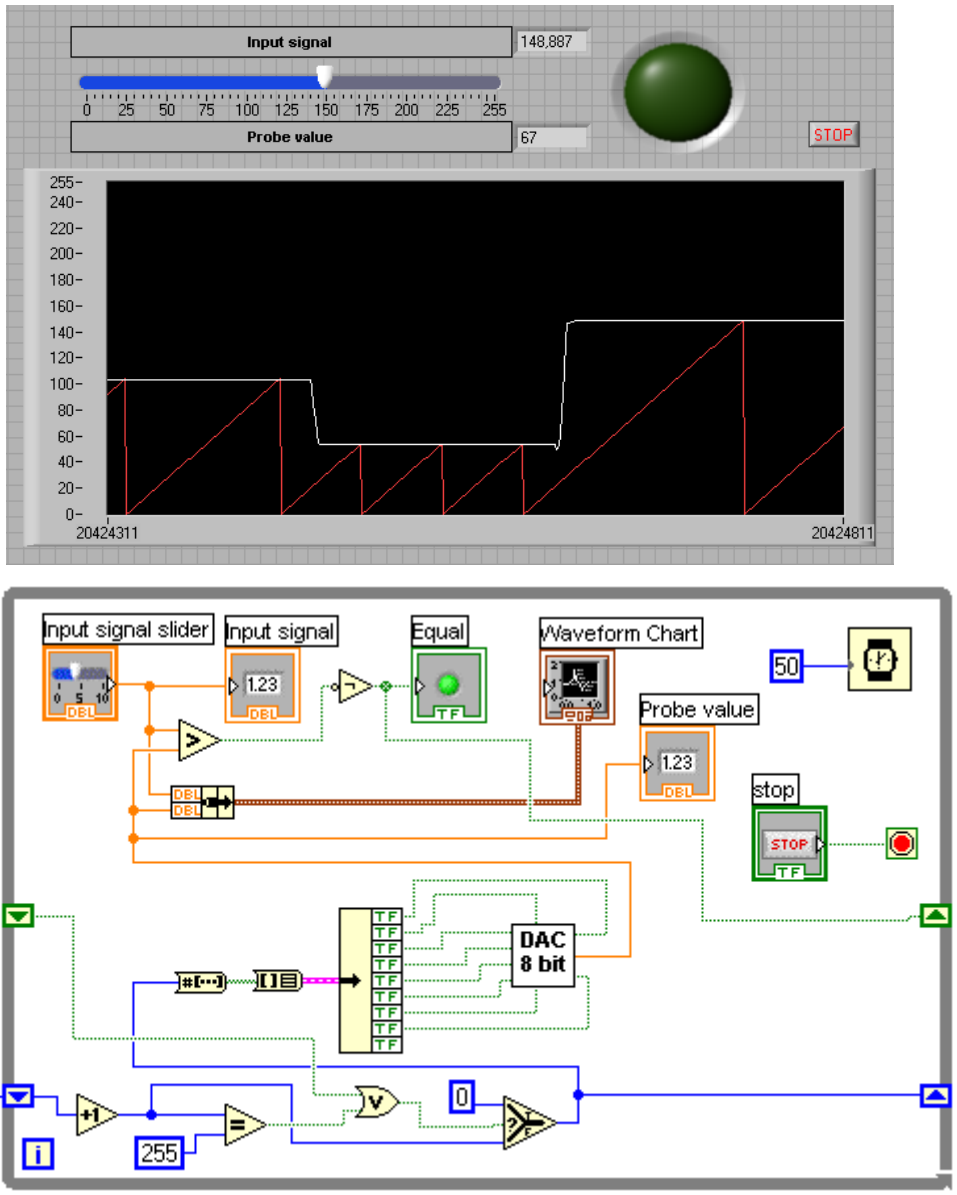


Рис. 5.4. Прибор моделирующий работу интегрирующего АЦП

Следящий алгоритм достаточно прост:

если

уровень пробного сигнала больше уровня входного сигнала,
уменьшить число счетчика на единицу,

если

уровень пробного сигнала меньше уровня входного сигнала,
увеличить число счетчика на единицу,

повторять бесконечно.

Однако если уровень входного сигнала меняется, АЦП должен вернуться к генерации пилообразного напряжения, чтобы нагнать изменение

входного сигнала. В том случае, когда тактовый генератор достаточно быстр, отслеживание происходит оперативно. Но если входной сигнал меняется слишком быстро, оцифрованный сигнал пропадает до тех пор, пока пробный сигнал снова не нагонит входной. В действительности существует предельная отслеживающая скорость АЦП. Она ограничивает максимальную частоту изменения входного сигнала.

На рис. 5.5 представлена лицевая панель прибора, моделирующего следящий АЦП.

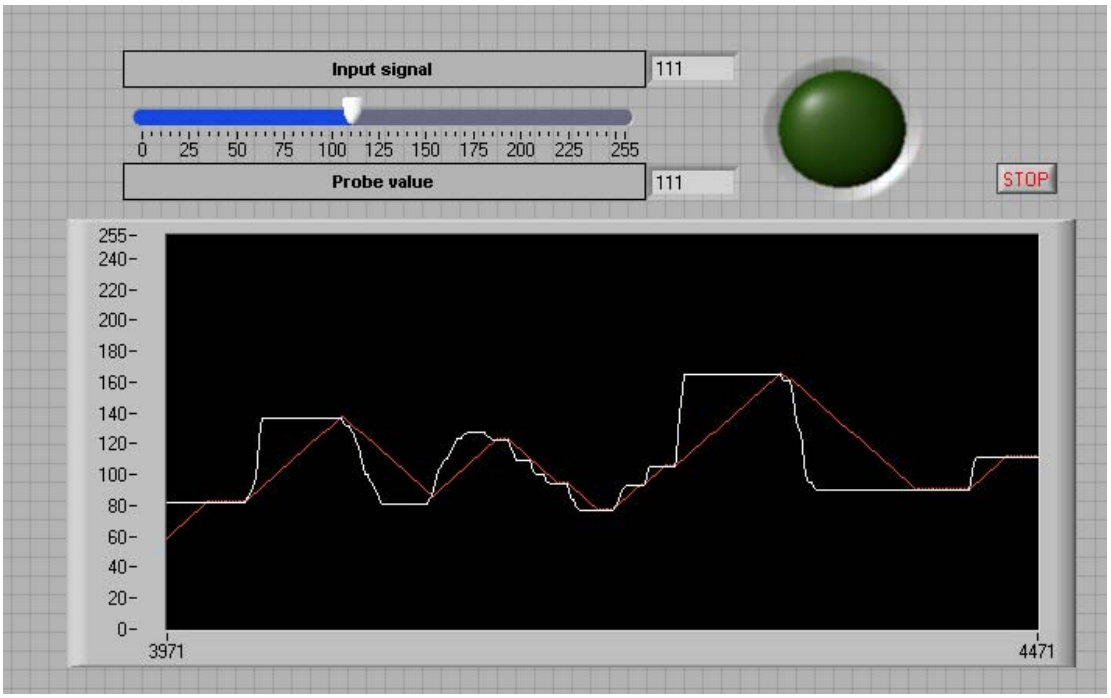


Рис. 5.5. Лицевая панель виртуального прибора, моделирующего следящий АЦП

Поскольку в следящем АЦП используется счетчик, работающий в режиме возрастания (убывания), то когда входной сигнал внезапно спадает ниже (возрастает выше) уровня пробного напряжения, следящий АЦП возвращается к генерации спадающего (возрастающего) пилообразного напряжения до тех пор, пока пробное напряжение достигнет уровня входного сигнала.

2. Выполнение работы

Задание 1.

Создайте модель ЦАП, представленного на рис. 5.1. Для отчета сделайте снимки экрана (*screenshot*) лицевой панели и структурной

схемы данного прибора. Создайте библиотеку подпрограмм с именем «*lab_5_library.llb*» и сохраните в ней созданный виртуальный прибор в виде подпрограммы.

Задание 2.

Создайте виртуальный прибор, представленный на рис. 5.2. Для отчета сделайте снимки экрана (*screenshot*) лицевой панели и структурной схемы данного прибора. Изучите принцип работы новых для вас узлов структурной схемы прибора и опишите их. Разберитесь с новыми типами данных, изучите новые элементы лицевой панели прибора. Сохраните прибор для его демонстрации во время защиты работы.

Задание 3.

Разработайте и создайте виртуальный прибор, лицевая панель которого представлена на рис. 5.5. Для отчета сделайте снимки экрана (*screenshot*) лицевой панели и структурной схемы данного прибора. Дайте пояснения по работе структурной схемы данного прибора, объясните, зачем вы использовали те или иные узлы на структурной схеме. Сохраните прибор для демонстрации во время защиты работы.

3. Содержание отчета

Отчет оформляется каждым студентом самостоятельно. Защита проходит в начале каждого следующего занятия с демонстрацией работы программы на ЭВМ. Студент, не подготовивший или не защитивший отчет по работе, к следующей лабораторной работе не допускается.

Содержание отчета:

1. Титульный лист.
2. Цель работы.
3. Лицевая панель и структурная схема прибора изображенного на рис. 5.1; библиотека подпрограмм с данным прибором оформленным в виде подпрограммы.
4. Лицевая панель прибора и структурная схема прибора изображенного на рис. 5.2; описание узлов: «*Wait*», «*Select*», «*Equal*», «*Increment*», «*Number to Boolean array*», «*Array to cluster*», «*Unbundle*», «*Bundle*», «*Greater*», «*Waveform chart*»; объяснения по типам данных, использованных в структурой схеме: «*integer*», «*boolean*», «*boolean array*», «*double*», данные собранные в кластер.

5. Лицевая панель прибора и структурная схема прибора разработанного в задании 3; объяснения принципов функционирования его структурной схемы.

6. Библиотека подпрограмм с реализованными устройствами.

7. Выводы по работе.

4. Контрольные вопросы и задания

1. С какими типами данных вы оперировали в данной работе?

2. На рисунках 5.2, 5.4 и 5.5 появился элемент «*Waveform chart*», дайте его описание.

3. Зачем используются сдвиговые регистры «*Shift register*» на рис. 5.2?

4. Какие управляющие структуры *LabView* использованы на рис. 5.2? Какая из структур вложена внутрь другой структуры?

5. Опишите узел «*Wait*».

6. Какие функции выполняют узлы «*Bundle*» и «*Unbundle*» на рис. 5.4?

7. Какой тип данных отображается оранжевым цветом на структурной схеме виртуального прибора?

8. Какой тип данных отображается жирными коричневыми проводниками на структурной схеме виртуального прибора?

Лабораторная работа № 6
РАБОТА С ПАРАЛЛЕЛЬНЫМ ПОРТОМ

Цель работы: изучение параллельного порта компьютера, создание программы в среде *LabView* для обмена данными с внешним устройством через параллельный порт.

Оборудование: дисплейный класс, среда визуального программирования *LabView* версии 7.0 или выше.

1. Общие сведения

Параллельный, последовательный и игровой порты – это наиболее распространенные порты ввода/вывода. В некоторых портативных

компьютерах может не быть игрового порта, но параллельный и последовательный входят в стандартную комплектацию для всех типов ПК.

Порт *Centronics* (*LPT* порт), или параллельный, − это промышленный стандарт для подсоединения принтеров к компьютеру. Обычно компьютер имеет, хотя бы один такой порт.

Параллельный порт для связи с принтером (или другим устройством) имеет базовый адрес (по умолчанию) *&H378* (888 в десятичной системе). Адресное пространство порта занимает диапазон от *&H378* до *&H37F*.

Адрес *&H378* служит для передачи или чтения данных. Имеется возможность записать в этот порт какой-либо байт (значение от 0 до 255 или от *&H00* до *&HFF*), включив или выключив соответствующие биты порта, которые выведены на разъем *LPT*-порта через контакты 2-9 (рис. 6.1).

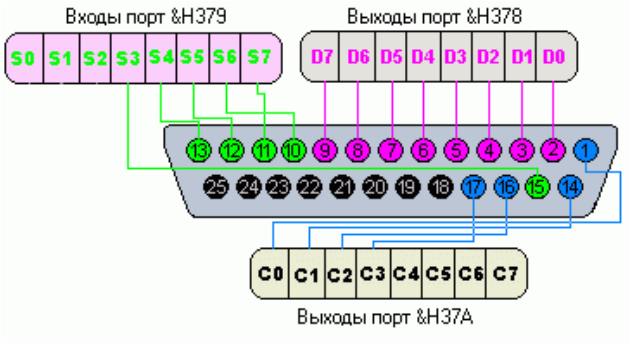


Рис. 6.1. Контакты LPT-порта

Записанное в порт значение сохраняется до тех пор, пока в порт не будет записано любое другое значение. Считать с этого порта можно только последний выведенный через него байт, то есть байт, выведенный из PC, а не состояние линий, подключенных к нему в данный момент. Адрес *&H379* предназначен для приема сигналов с устройства, подключенного к этому порту именно в данный момент, то есть, опрашивая порт *&H379*, можно узнать его состояние в режиме реального времени.

Адрес *&H37A* служит для передачи сигналов к устройству, подключенному к этому порту. Все сказанное для порта с адресом *&H378* справедливо и для этого порта.

Если в порте с адресом *&H378* пользователь может использовать все 8 бит, то в порте с адресом *&H379* ему предоставлены только 5 старших бит, а в порте с адресом *&H37A* только 4 младших бита.

Пользователь имеет в своём распоряжении двенадцать выходов и пять входов (таблица). При использовании различных внешних шифраторов и дешифраторов эти числа можно увеличить многократно. Но,

так как за все приходится платить, уменьшатся функциональные возможности управления. Однако все зависит от конкретной задачи и в некоторых случаях это не так важно.

Последняя колонка таблицы требует некоторого пояснения. «Да» означает, что сигнал инверсный (обратный). Для того, чтобы установить в единицу контакт № 1 разъема, (*C0* порта *&H37A*) мы должны записать в порт логический ноль. Контроллер проинвертирует этот сигнал, т. е. изменит на противоположный (на логическую «1») и установит единицу на контакте № 1 разъёма. Подается 0 получается 1, подается 1 получается 0. Аналогично для бита *S7* входного порта *&H379*. Если на этом входе присутствует логический ноль, то при считывании данных из порта мы получим 1 в бите *S7* и наоборот.

Входы и выходы LPT-порта

ножка порта	сигнал	бит	порт	инверсия
1	выход	*C0*	*37AH*	да
2	выход	*D0*	*378H*	нет
3	выход	*D1*	*378H*	нет
4	выход	*D2*	*378H*	нет
5	выход	*D3*	*378H*	нет
6	выход	*D4*	*378H*	нет
7	выход	*D5*	*378H*	нет
8	выход	*D6*	*378H*	нет
9	выход	*D7*	*378H*	нет
10	вход	*S6*	*379H*	нет
11	вход	*S7*	*379H*	да
12	вход	*S5*	*379H*	нет
13	вход	*S4*	*379H*	нет
14	выход	*C1*	*37AH*	да
15	вход	*S3*	*379H*	нет
16	выход	*C2*	*37AH*	нет
17	выход	*C3*	*37AH*	да
18-25	земля	-	-	-

Лицевая панель и структурная схема прибора для вывода данных через адрес *&H378 LPT*-порта представлена на рис. 6.2.

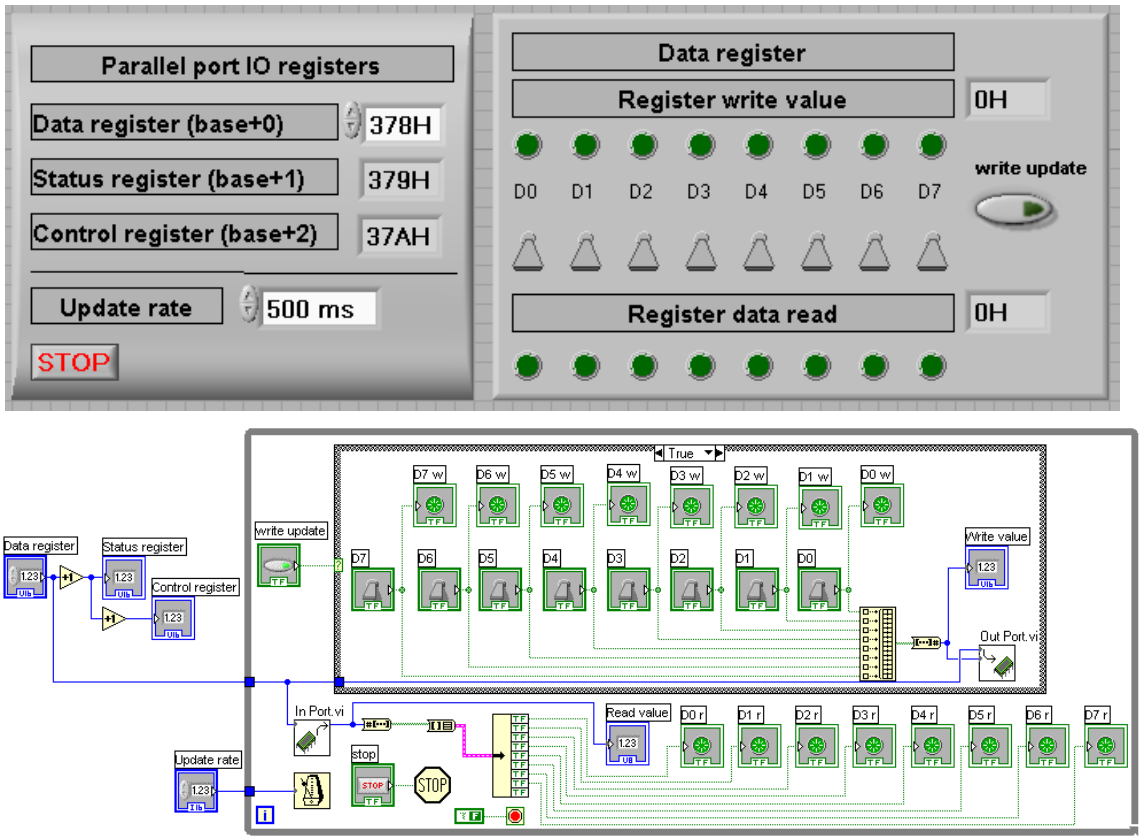


Рис. 6.2. Виртуальный прибор для вывода данных через LPT-порт по адресу &H378

На рис. 6.3 представлена схема электрическая принципиальная, с помощью которой можно определить работоспособность данной программы.

Параллельный порт может давать очень ограниченный ток в нагрузку, поэтому для устройства на рис. 6.3 предполагается использовать сверхъяркие светодиоды с малым токопотреблением. В противном случае обычно устройство, подключаемое к LPT-порту, имеет свой собственный источник питания, а управляющие сигналы с порта подаются через усилители.

На структурной схеме устройства (рис. 6.2) непосредственное взаимодействие с *LPT*-портом происходит в функциональных блоках «*Out port*» для записи данных в порт и «*In port*» для чтения данных из порта.

Чтение данных из порта происходит в цикле структуры «*While loop*» с частотой в данном случае 500 мс. Запись данных в порт происходит по нажатию кнопки «*write update*» на лицевой панели прибора.

При записи в порт логической единицы на каком-либо из выводов можно наблюдать включение светодиода, а при установки ножки порта в логический ноль светодиод гаснет.

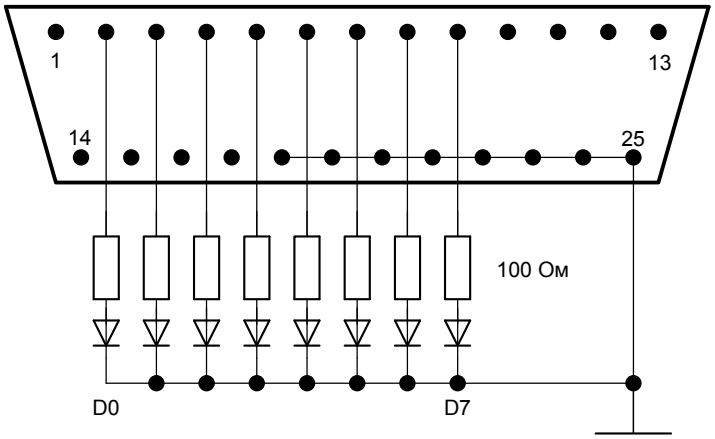


Рис. 6.3. Схема устройства, подключенного к LPT-порту

2. Выполнение работы

Задание 1.

Создайте виртуальный прибор, представленный на рис. 6.2. Для отчета сделайте снимки экрана (*screenshot*) лицевой панели и структурной схемы данного прибора. Создайте библиотеку подпрограмм с именем «*lab_6_library.llb*» и сохраните в ней созданный виртуальный прибор. Для проверки функционирования разработанного виртуального прибора подключите устройство (рис. 6.3) к порту и наблюдайте за поведением светодиодов. Как будет вести себя устройство и виртуальный прибор, если увеличивать/уменьшать число мс в окне ввода «*Update rate*», отразите ваши наблюдения в отчете.

Задание 2.

Разработайте и создайте виртуальный прибор, который позволяет управлять состоянием и контролировать данные по всем трем адресам параллельного порта. Лицевую панель прибора можно оформить в виде, предлагаемом на рис. 6.4.

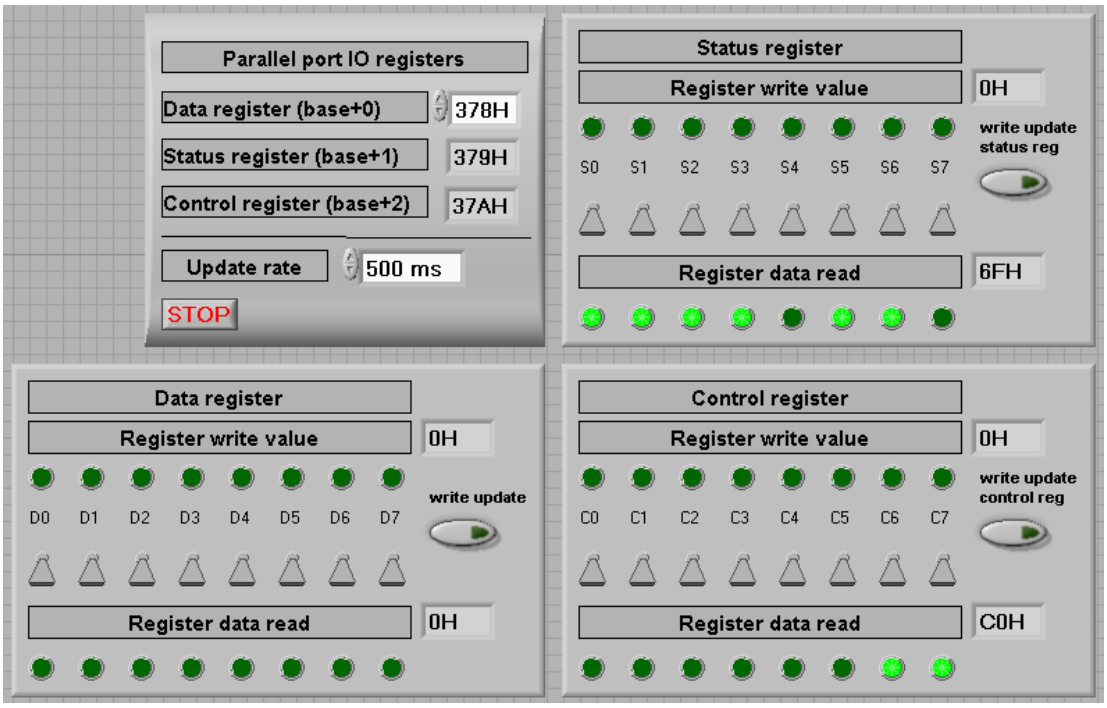


Рис. 6.4. Лицевая панель виртуального прибора для полного контроля и управления LPT-портом

Устройство, которое предлагается подключить к параллельному порту, показано на рис. 6.5.

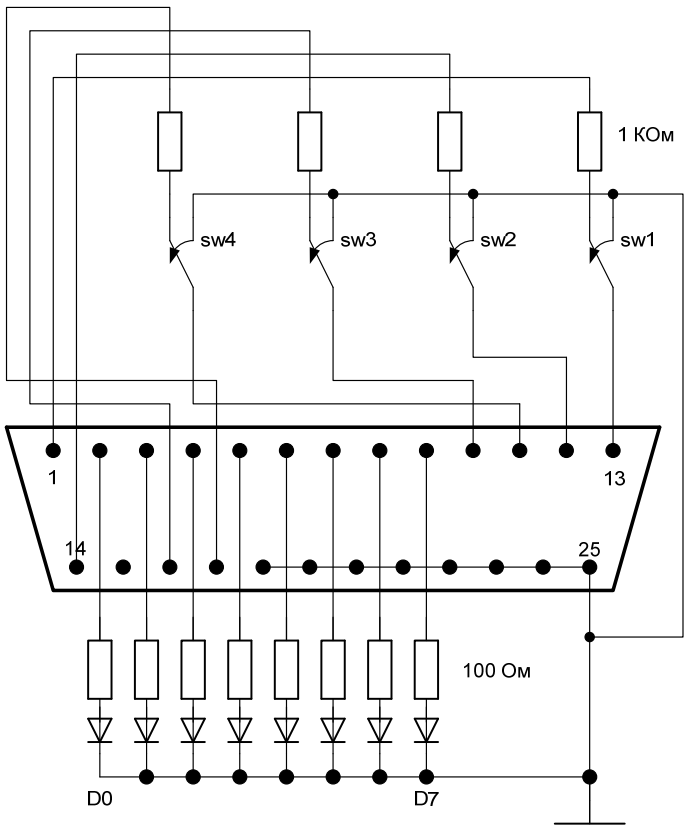


Рис. 6.5. Устройство, подключенное к параллельному порту

С помощью регистра данных можно включать и выключать светодиоды, подключенные к выводам 2÷9 порта. С помощью переключателей *sw*1÷*sw*4 можно выбирать сигнал, подаваемый на ножки регистра статуса: для подачи логического нуля переключатель следует замкнуть на землю, для подачи логической единицы следует замкнуть переключатель на один из выводов регистра контроля и установить сигнал на соответствующей ножке в логическую единицу.

С помощью разработанной программы, посмотрите можно ли вводить данные в компьютер используя параллельный порт. Посмотрите, как считывается и записывается сигнал с ножек регистров контроля и статуса, которые физически не существуют. Отразите ваши наблюдения в отчете.

Задание 3.

В комплект поставки инструментария *LabView* входит несколько виртуальных приборов, которые демонстрируют работу с параллельным портом. Загрузите виртуальный прибор, который называется «*Parallel Port Read and Write Loop.vi*». Посмотрите, как в нем реализовано взаимодействие с параллельным портом. Попробуйте с помощью этого виртуального прибора управлять светодиодами и считывать состояние переключателей подключенных к *LPT*-порту. Отразите полученные данные в отчете. Сравните их с данными, полученными в ходе выполнения задания 2.

Виртуальный прибор «*Parallel Port Read and Write Loop.vi*» можно открыть двумя способами. Первый способ: открыть примеры *LabView* и зайти во вкладку *Hardware Input and Otput -> General -> Parallel Port Read and Write Loop*. Второй способ: найти на диске и запустить файл -> директория с установленным *LabView\ examples\portaccess\parallel port examples.llb* – это библиотека *LabView*, в которой находится данный виртуальный прибор.

3. Содержание отчета

Отчет оформляется каждым студентом самостоятельно. Защита происходит в начале каждого следующего занятия с демонстрацией работы программы на ЭВМ. Студент, не подготовивший или не защитивший отчет по работе, к следующей лабораторной работе не допускается.

Содержание отчета:

1. Титульный лист.
2. Цель работы.
3. Лицевая панель и структурная схема прибора, изображенного на рис. 6.2; описание функционирования данного виртуального прибора и пояснения, как включить или выключить тот или иной светодиод, подключенный к параллельному порту.
4. Лицевая панель прибора и структурная схема прибора, разработанного в задании 2; объяснения принципов его функционирования; наблюдения за поведением порта, проведенными по заданию 2.
5. Лицевая панель прибора и структурная схема прибора «*Parallel Port Read and Write Loop.vi*»; данные полученные в ходе выполнения задания 3.
6. Выводы по работе.

4. Контрольные вопросы и задания

1. Что вы можете сказать про параллельный порт?
2. С помощью каких узлов *LabView* можно взаимодействовать с портами компьютера?
3. Опишите функциональный узел *LabView «Out Port*».
4. Опишите функциональный узел *LabView «In Port*».
5. Какие еще стандартные интерфейсы ЭВМ вы знаете? Как с ними можно взаимодействовать в *LabView*?
6. Можно ли вводить данные в ЭВМ, используя ножки параллельного порта 2÷9 (бит 0÷бит 7)?

Лабораторная работа № 7
РАБОТА СО СТРОКАМИ И ФАЙЛАМИ

Цель работы: изучение функций работы со строками, изучение функций файлового ввода/вывода, создание программы производящей вывод данных в текстовые файлы.

Оборудование: дисплейный класс, среда визуального программирования *LabView* версии 7.0 или выше.

1. Общие сведения

Строки. Строки – это последовательность отображаемых и неотображаемых символов таблицы *ASCII.* Строки обеспечивают независимый от платформы формат обмена данными. Некоторые из наиболее распространенных строковых приложений включают в себя:

- создание простых текстовых сообщений;

- передачу числовых данных в приборы в виде строк символов и преобразование строк в числовые данные;

- сохранение числовых данных на диск, с процедурой преобразования числовых данных из двоичного формата в формат символов *ASCII* строк перед записью в файл;

- диалоговые окна инструкций и подсказок.

Строковые переменные в *LabView* имеют то же значение, что и в языках программирования высокого уровня.

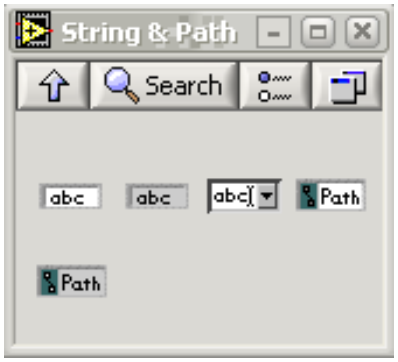


Рис. 7.1. Группа элементов «String&Path»

На лицевой панели приборов строки могут быть представлены в виде таблиц, полей ввода текста и меток. Для размещения строкового элемента на лицевой панели необходимо вызвать палитру элементов и выбрать в ней группу «*String&Path*» (рис. 7.1).

Строка, размещенная на лицевой панели, может иметь несколько типов отображения строковых данных (рис. 7.2), режим стандартного отображения, режим отображения с обратным слэшем для вывода управляющих кодов, режим скрытого отображения текста и режим отображения в виде шестнадцетиричных *ASCII*-кодов символов.

Для смены типа отображения символов нужно правым щелчком мыши вызвать контекстное меню строкового элемента и выбрать необходимый тип отображения – «*Normal Display*», «’\’ *Code Display*», «*Password Display*», «*Hex Display*».

На панели диаграмм строковые элементы будут показаны в виде обычных терминалов, так же, как и другие элементы.

Строки, так же как и все остальные элементы-переменные в *LabView*, могут быть элементами управления и элементами отображения данных, элементами-источниками и элементами-приемниками.

Правила соединения строковых элементов между собой ничем не отличаются от правил соединения числовых и логических элементов. Линии соединения между строковыми элементами рисуются розовым цветом. Строковые элементы так же как численные и логические могут составлять массивы строк и входить в составы кластеров.

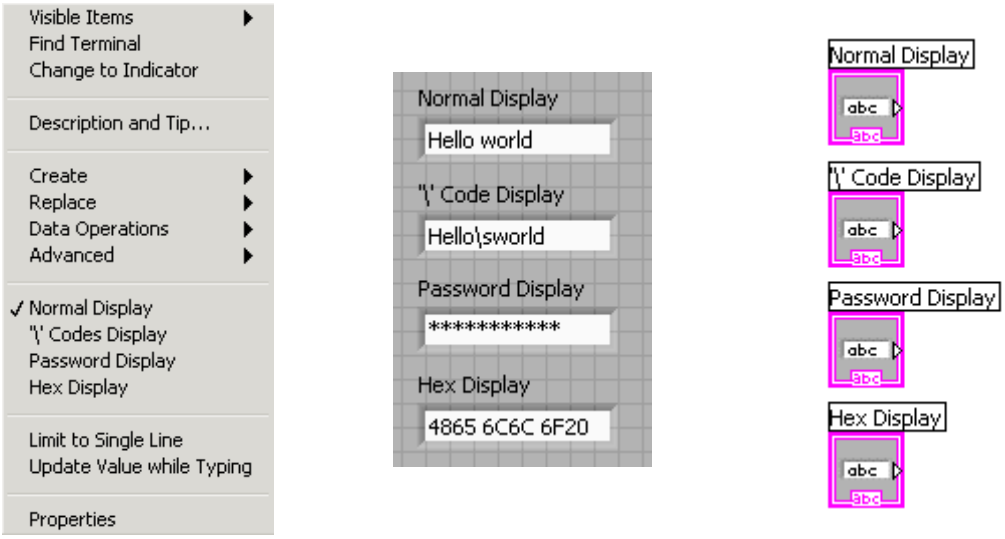


Рис. 7.2. Примеры представления строковых переменных

В палитре функций для выполнения операций над строковыми переменными отведена группа функций «*String*» (рис. 7.3).

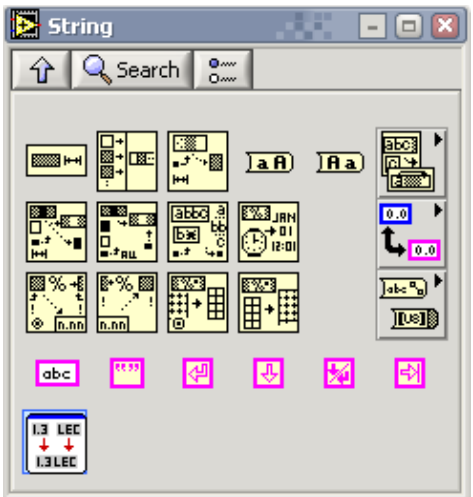


Рис. 7.3. Функции обработки строковых переменных

На рис. 7.4 приведен пример виртуального прибора, который преобразует числовое представление числа в строковое, записанное словами (для простоты построения взят диапазон чисел от 20 до 90).

Файловый ввод/вывод. Функции файлового ввода/вывода производят файловые операции записи и считывания данных. Функции файлового ввода/вывода расположены в палитре функций «*File I/O*» (рис. 7.5) и предназначены для:

- открытия и закрытия файла данных;
- считывания и записи данных из/в файл(а);

- считывания и записи данных из/в файл(а) в виде таблицы символов;
- перемещения и переименования файлов и каталогов;
- изменения характеристик файла;
- создания, изменения и считывания файлов конфигурации.

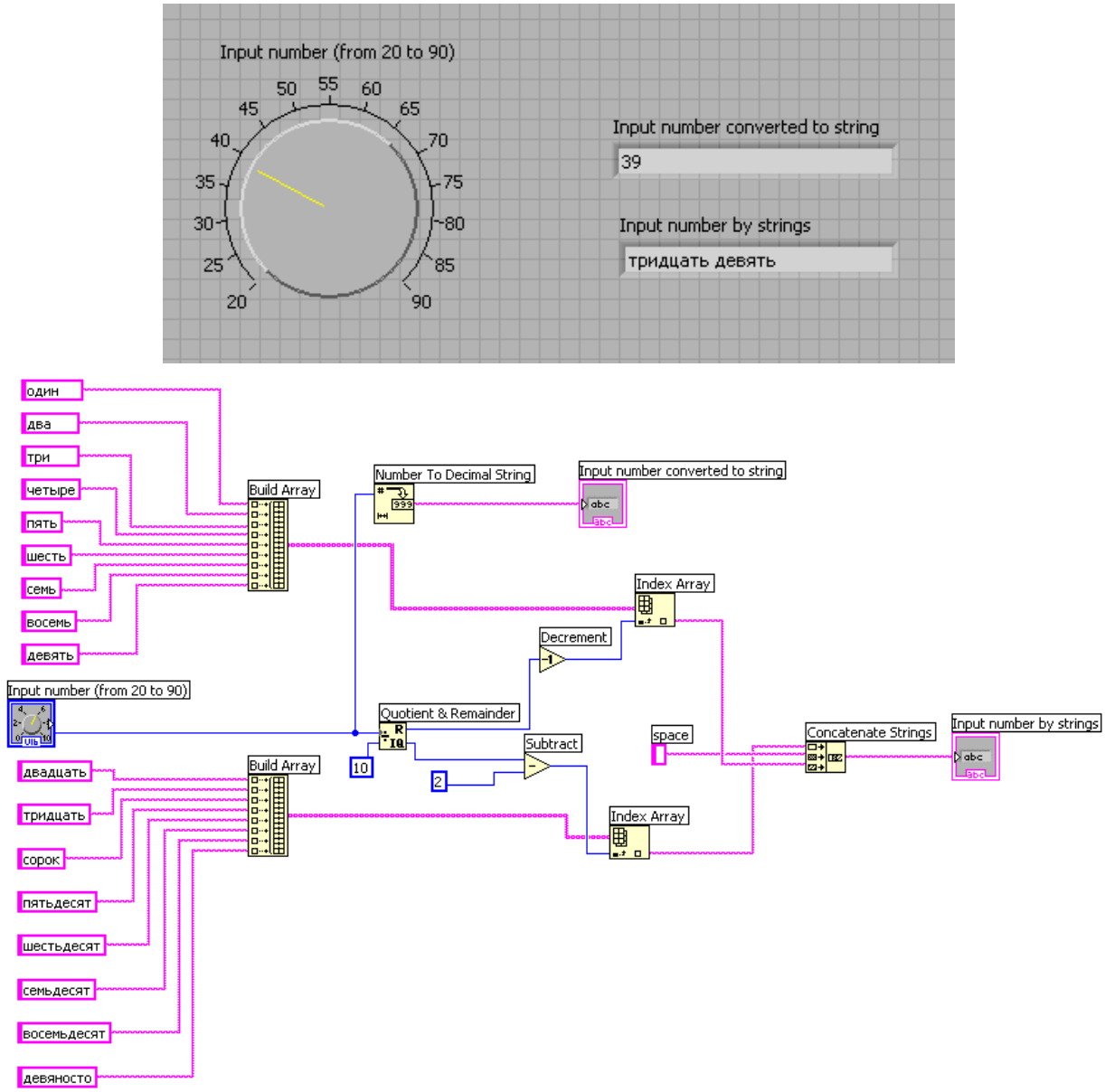


Рис. 7.4. Пример виртуального прибора перевода чисел в их представление в виде слов

Палитру функций работы с файлами можно разделить на три части: функции высокого уровня, функции низкого уровня и подпалитру расширенных возможностей.

Функции файлового ввода/вывода высокого уровня расположены в верхней строке палитры «*File I/O*» и предназначены для выполнения основных операций по вводу/выводу данных.

Функции файлового ввода/вывода низкого уровня расположены в средней строке палитры «*File I/O*». Они используются для создания нового или обращения к ранее созданному файлу, записи и считывания данных и закрытия файла. Функции низкого уровня работы с файлами поддерживают все операции, необходимые при работе с файлами.

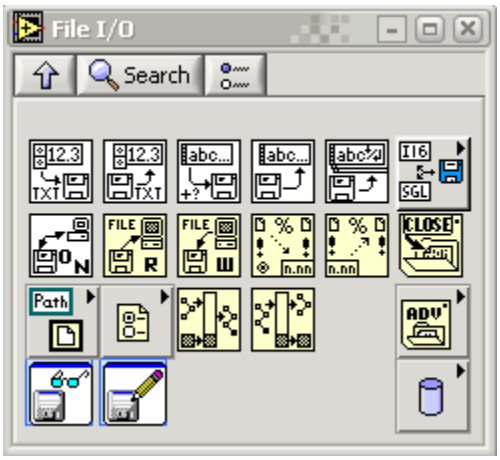


Рис. 7.5. Функции работы с файлами

Для осуществления стандартной процедуры ввода/вывода данных в/из файл(а) необходимо выполнять следующую последовательность действий:

1. Создание или открытие файла. Указание месторасположения существующего файла или пути для создания нового файла с помощью диалогового окна *LabView*. После открытия файл *LabView* создает ссылку на него.
2. Произведение операции считывания или записи данных в/из файл(а).
3. Закрытие файла.
4. Обработка ошибок.

На рис. 7.6 приведен пример простейшего виртуального прибора производящего запись текстовой строки в файл.

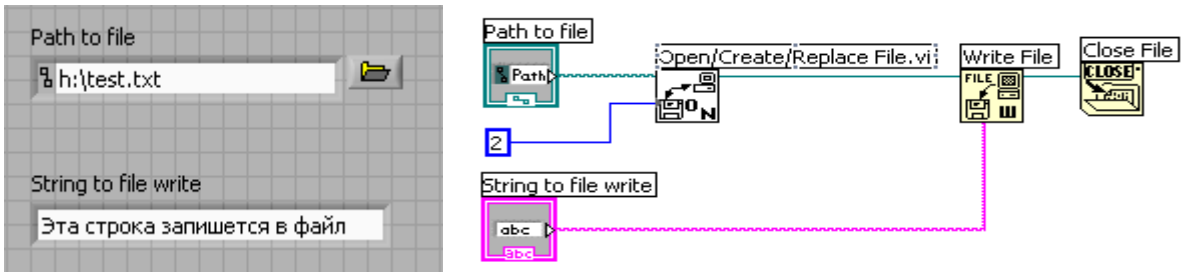


Рис. 7.6. Пример записи данных в файл

Здесь, на передней панели приборов, расположено два элемента: строковый элемент, содержащий текст, который будет записан в файл, и

элемент «*File Path Control*» из палитры «*String&Path*», который содержит путь и имя файла, с которым будет взаимодействовать программа.

На функциональной схеме при помощи узла «*Open/Create/Replace File*» происходит открытие файла и получение файлового указателя «*refnum*». На вход узла подается два сигнала: путь и имя файла, и число обозначающее функцию, которая будет выполняться с указанным файлом (в нашем случае мы подаем число 2 – создать новый файл или перезаписать существующий).

Далее при помощи узла «*Write file*» производится запись данных в файл и закрытие файла при помощи узла «*Close file*».

После выполнения данной программы на диске «*h:*» получим файл «*test.txt*», содержащий текст «Эта строка запишется в файл».

На рис. 7.7 приведен пример программы, которая позволяет считывать данные из файла.

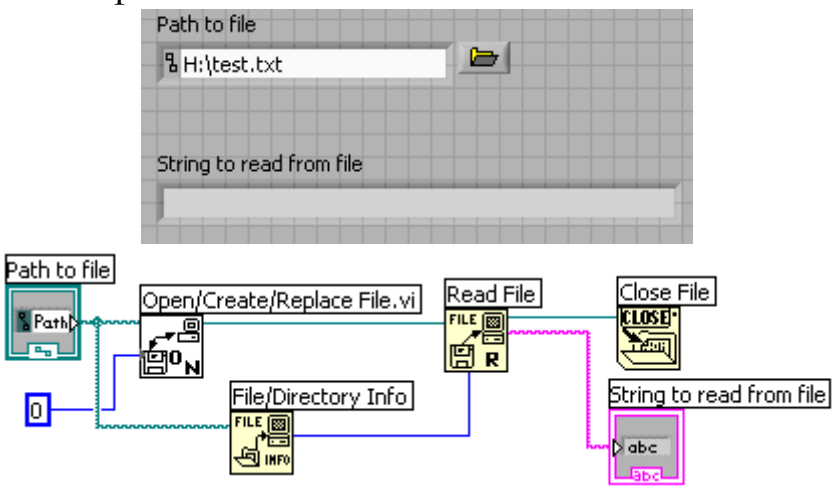


Рис. 7.7. Пример чтения данных из файла

Здесь для чтения информации из файла используется узел «*Read File*», на который подается файловый указатель «*refnum*» и количество символов (байт), которое нужно считать из файла. Размер файла определяется при помощи узла «*File/Directory Info*».

2. Выполнение работы

Задание 1.

Создайте и оформите в виде подпрограммы виртуальный прибор, подобный представленному на рис. 7.4. Данная подпрограмма должна переводить числа в диапазоне от 0 до 100 из числового формата в формат записи числа в виде строк текста.

Для отчета сделайте снимки экрана (*screenshot*) лицевой панели и структурной схемы данного прибора. Создайте библиотеку подпрограмм с именем «*lab_7_library.llb*» и сохраните в ней созданный виртуальный прибор.

Задание 2.

Разработайте и создайте виртуальный прибор, который будет производить следующие действия:

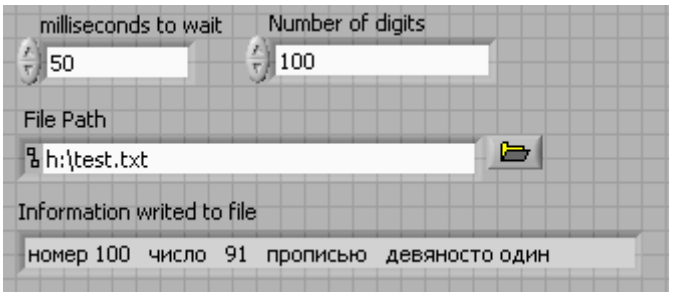
- генерировать случайно число в диапазоне от 0 до 100;

- переводить сгенерированное число из числового формата в формат в виде записи числа из строк;

- последовательно записывать полученные числа в текстовый файл.

Количество генерируемых чисел и имя файла должно задаваться с лицевой панели прибора. Файл должен быть стандартного текстового формата txt. Для перевода сгенерированного числа из числового формата в формат записи числа строкой используйте подпрограмму, разработанную в пункте задания 1.

Пример лицевой панели прибора и данных, выводимых в файл показаны на рис. 7.8.



номер 1 число 80 прописью восемьдесят
номер 2 число 63 прописью шестьдесят три
номер 3 число 77 прописью семьдесят семь
номер 4 число 37 прописью тридцать семь
номер 5 число 40 прописью сорок
номер 97 число 94 прописью девяносто четыре
номер 98 число 8 прописью восемь
номер 99 число 75 прописью семьдесят пять
номер 100 число 91 прописью девяносто один

Рис. 7.8. Пример лицевой панели прибора и результатов работы программы

3. Содержание отчета

Отчет оформляется каждым студентом самостоятельно. Защита проходит в начале каждого следующего занятия с демонстрацией работы программы на ЭВМ. Студент, не подготовивший или не защитивший отчет по работе, к следующей лабораторной работе не допускается.

Содержание отчета:

2. Титульный лист.
3. Цель работы.
4. Лицевая панель и структурная схема прибора, разработанного по пункту задания 1.
5. Описание всех новых изученных элементов лицевой панели и структурной схемы прибора.
6. Лицевая панель прибора и структурная схема прибора, разработанного по пункту задания 2; объяснения принципов его функционирования; распечатка файла с результатами работы программы.
7. Выводы по работе.

4. Контрольные вопросы и задания

1. Какие типы данных вы изучили в данной работе?
2. Какие функции работы со строками вы знаете?
3. Опишите последовательность действий при записи данных в файл.
4. Опишите последовательность действий при чтении данных из файла.
5. Какие операции с файлами можно выполнять в *LabView*?
6. Можно ли при помощи средств *LabView* создать текстовый редактор наподобие редактора «Блокнот» из Windows? Какова будет последовательность ваших действий при его создании?
7. Какой тип данных в *LabView* отображается розовым цветом?

РЕКОМЕНДУЕМАЯ ЛИТЕРАТУРА

1. *Батоврин, В.К.* LabVIEW: практикум по основам измерительных технологий / В.К. Батоврин, А.С. Бессонов, В.В. Мошкин, В.Ф. Папуловский. – М.: ДМК Пресс, 2005. − 208 с. – ISBN 5-94074-267-X.

2. *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.

3. *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.

5. *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.

Оглавление

СОЗДАНИЕ ВИРТУАЛЬНЫХ ПРИБОРОВ
В СРЕДЕ LABVIEW

Методические указания к лабораторным работам

Составитель
МАКАРОВА Наталья Юрьевна

Ответственный за выпуск – зав. кафедрой профессор В.П. Легаев

Подписано в печать 28.04.10.
Формат 60x84/16. Усл. печ. л. 3,49. Тираж 100 экз.
Заказ
Издательство
Владимирского государственного университета
600000, Владимир, ул. Горького, 87.