Министерство науки и высшего образования
Российской Федерации

Федеральное государственное бюджетное образовательное
учреждение высшего образования
«Воронежский государственный технический университет»

Кафедра конструирования и производства радиоаппаратуры

СТРУКТУРА И ВОЗМОЖНОСТИ ПАКЕТА LabVIEW. СОЗДАНИЕ ВИРТУАЛЬНОГО ИЗМЕРИТЕЛЬНОГО СРЕДСТВА 2

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

к выполнению лабораторных работ по дисциплине
«Приемоусилительные и видеотелевизионные системы»
для студентов направления 11.03.03 «Конструирования и
технология электронных средств»
(профиль «Проектирование и технология радиоэлектронных
средств») всех форм обучения

Воронеж 2021

УДК 681.3

Составители:

доктор. техн. наук А. В. Башкиров,
канд. техн. наук И.С. Бобылкин.

Изучения работы возможности пакета LabView: методические указания к выполнению лабораторных работ по дисциплине «Приемоусилительные и видеотелевизионные системы» для студентов направления 11.03.03 «Конструирования и технология электронных средств» (профиль «Проектирование и технология радиоэлектронных средств») всех форм обучения/ ФГБОУ ВО «Воронежский государственный технический университет»; сост.: А. В. Башкиров, И.С. Бобылкин. Воронеж: Изд-во ВГТУ, 2021. 31 с.

Методические указания предназначены на изучение и освоение принципов создания виртуальных средств, моделирующих работу процессора вычислительного устройства.

Предназначены для проведения лабораторных работ по дисциплине «Приемоусилительные и видеотелевизионные системы» для студентов 3 курса.

Методические указания подготовлены в электронном виде в текстовом редакторе MS Word 2003 и содержатся в LR2-3TTI.doc
Ил. 18. Библиогр.: 3 назв.

УДК 681.3
ББК 38.54

Рецензент - О. Ю. Макаров, д-р техн. наук, проф. кафедры конструирования и производства радиоаппаратуры ВГТУ

Издается по решению редакционно-издательского совета Воронежского государственного технического университета

же задачи с использованием элемента “Formula Node”. Произвести отладку работы виртуального прибора в циклическом режиме и в режиме однократного запуска.

2. Реализовать вычисления по формуле: z = Ax2+By3. в виде подпрограммы. Создать VI с возможностью задавать значения переменных x и y при помощи элементов «Numeric Control».

3. Реализовать возможность вычисления степени произвольного положительного числа по формуле: : x^y = exp(y * log(x))). Обеспечить возможность задавать значения переменных x и y при помощи элементов «Numeric Control». Вычисление степени реализовать в виде SubVI.

4. Создать виртуальный прибор, отображающий осциллограмму виде фигуры Лиссажу. Обеспечить возможность регулировки частоты исходных сигналов. Сохранить в файле осциллограммы для соотношения частот 1:1, 1:2 и 1:3.

5. Создать виртуальный прибор, имитирующий двухлучевой осциллограф, отображающий гармонический сигнал и прямоугольные импульсы одинаковой частоты. Обеспечить возможность регулировки скважности прямоугольных импульсов.

6. Создать виртуальный прибор, отображающий случайный сигнал с временем выборки 1с. Обеспечить световую сигнализацию превышения некоторого, произвольно задаваемого уровня сигнала.

7. Создать виртуальный прибор, отображающий гармонический сигнал с возможностью наложения шума. Создать элементы управления, позволяющие регулировать амплитуду и частоту гармонического сигнала, а также включение и выключение наложения шумового сигнала.

Лабораторная работа № 2

Многократные повторения и циклы

Цель работы

Получение навыков программирования циклических алгоритмов и сравнение способов их реализации в системе LabView с традиционными языками программирования.

Задачи работы

Получить навыки работы с циклическим алгоритмом For.

Получить навыки работы с циклическим алгоритмом Whil.

Получить навыки работы с оператором выбора Case.

Получить навыки работы с оператором последовательности Sequece.

Получить навыки работы с оператором структуры Event.

Отчет:

- титульный лист;
- цели и задачи лабораторной работы;
- задание на лабораторную работу;
- результаты выполненной работы.

Справочно-методический материал

Структуры являются графическим представлением операторов цикла и операторов Case (Варианта), используемых в текстовых языках программирования. Структуры на блок-диаграмме используются для выполнения повторяющихся операций над потоком данных, операций в определенном порядке и наложения условий на выполнение операций.

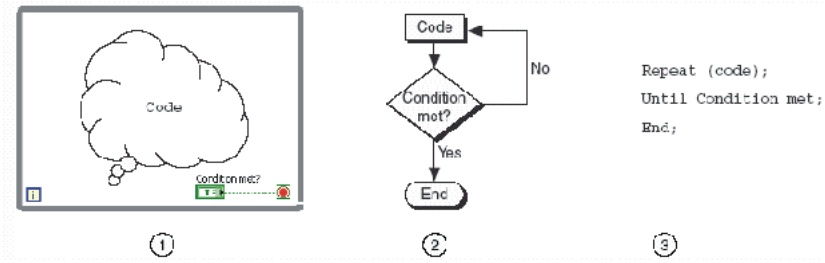
Среда LabVIEW содержит пять структур: Цикл While (по условию), Цикл For (с фиксированным числом итераций), структура Case (Вариант), структура

Sequence (Последовательность), структура Event (Событие), а также Formula Node (узел Формулы).

Рассмотрены структуры – Цикл While (по условию), Цикл For (с фиксированным числом итераций), а также функции, часто используемые с этими структурами, такие как Shift Register (сдвиговый регистр) и Feedback Node (узел обратной связи).

Цикл While (по Условию)

Цикл While (по условию) работает до тех пор, пока не выполнится логическое условие выхода из цикла. Цикл While аналогичен циклам Do и Repeat_Until, используемым в текстовых языках программирования. На рисунке 9 представлены 1 – цикл While в среде LabVIEW, 2 – эквивалентную блок-схему работы цикла While, 3 – пример текстового аналога кода работы цикла While.



1. LabVIEW цикл While
2. Блок-схема
3. Текстовый аналог когда

Рис. 1. Представление оператора цикла While

Цикл While находится в палитре Functions»Structures. После того как цикл выбран в палитре Functions (Функций), следует с помощью курсора выделить часть блок-диаграммы, которую необходимо поместить в цикл. После отпускания кнопки мыши, выделенная область блок-диаграммы помещается в тело цикла.

Добавление объектов блок-диаграммы в тело цикла

осуществляется помещением или перетаскиванием объектов.

Блок-диаграмма цикла While выполняется до тех пор, пока не выполнится условие выхода из цикла. По умолчанию, терминал условия выхода имеет вид, показанный слева. Это значит, что цикл будет выполняться до поступления на терминал условия выхода значения TRUE. В этом случае терминал условия выхода называется терминалом Stop If True (Остановка если Истина).

Терминал счетчика итераций, показанный слева, содержит значение количества выполненных итераций. Начальное значение терминала всегда равно нулю.

На блок-диаграмме, показанной на рисунке 10, условие выхода из цикла ^ While определяется значением выходного параметра подпрограммы ВП большего или равного 10,00 и состоянием терминала элемента управления Enable. Функция And (Логическое «И») на выходе выдает значение TRUE, если оба на поля ввода данных функции поступают значения TRUE. В противном случае функция на выходе выдает значение FALSE и работа цикла завершается.

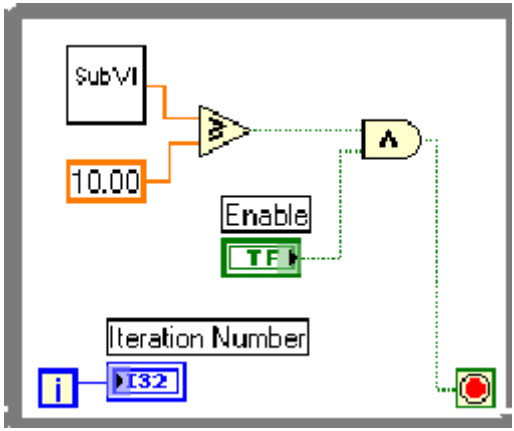


Рис. 2. Блок-диаграмма цикла While «продолжение Если Ложь»

В предыдущем примере велика вероятность получения бесконечно выполняемого цикла. Обычно стремятся получить единственное условие выхода из цикла, нежели одновременное выполнение двух условий.

Предусмотрена возможность изменения условия выхода и соответствующего ему изображения терминала условия выхода. Щелчком правой кнопки мыши по терминалу условия выхода или по границе цикла необходимо вызвать контекстное меню и выбрать пункт Continue If True (Продолжение если Истина). Также можно воспользоваться инструментом УПРАВЛЕНИЕ, щелкнув им по терминалу условия выхода. Изображение терминала условия выхода поменяется на показанное слева Continue If True (Продолжение Если Истина). В результате условием выхода из цикла становится поступающее на терминал условия выхода значение FALSE, как показано на следующей блок-диаграмме (рисунок 3).

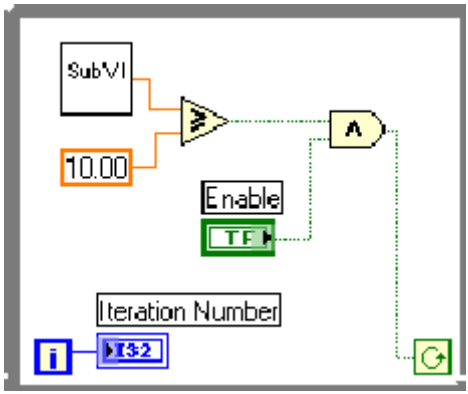


Рис.3. Блок-диаграмма цикла While «продолжение Если Истина» Цикл While выполняется до тех пор, пока выходные данные подпрограммы ВП остаются меньше «10».

Терминалы входных/выходных данных цикла

Данные могут поступать в цикл While (или выходить из него) через терминалы входных/выходных данных цикла. Терминалы входных/выходных данных цикла передают данные из структур и в структуры. Терминалы входных/выходных данных цикла отображаются в виде сплошных прямоугольников на границе области цикла While.

Прямоугольник принимает цвет типа данных, передаваемых по терминалу. Данные выходят из цикла по его завершении. В случае если данные поступают в цикл While через терминал входных/выходных данных цикла, выполнение цикла начинается при поступлении данных в терминал.

На следующей блок-диаграмме (рисунок 4) терминал счетчика итераций присоединен к терминалу выхода цикла. Значения из терминала выхода цикла не поступают к элементу отображения номера итерации до завершения цикла While.

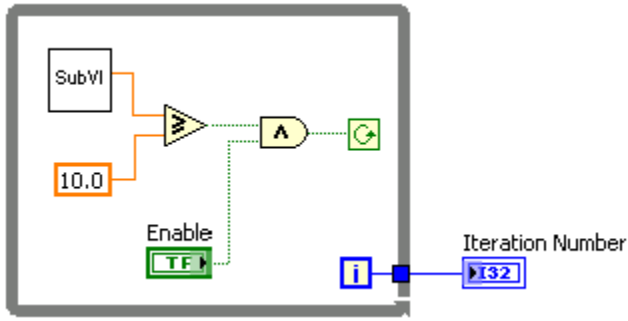


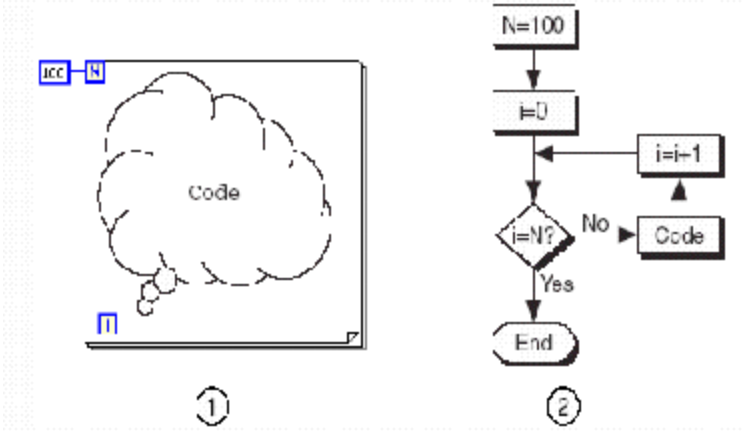
Рис. 4. Блок-диаграмма цикла While со счетчиком итераций на выходе терминала

Лишь последнее значение итерации отображается элементом отображения номера итерации.

Цикл For (с фиксированным числом итераций)

Цикл For (с фиксированным числом итераций)

выполняет повторяющиеся операции над потоком данных определенное количество раз (по заданию). Следующая иллюстрация (рисунок 5) демонстрирует 1 – цикл For в среде LabVIEW, 2 – эквивалентную блок-схему работы цикла For, 3 – пример текстового аналога кода работы цикла For.




```
N=100;
i=0;
Until i=N:
    Repeat (code; i=i+1);
End;
```

3

1. LabVIEW
2. Блок-схема
3. Текстовый аналог кода

Рис. 5. Представление цикла For

Цикл For, расположен в палитре Функций в разделе «Functions» Structures. Значение, присвоенное терминалу максимального числа итераций N цикла, показанного слева, определяет максимальное количество повторений операций над потоком данных.

i Терминал счетчика итераций, показанный слева, содержит значение количества выполненных итераций. Начальное значение счетчика итераций всегда равно 0.

Цикл For отличается от цикла While тем, что завершает работу, выполнив заданное максимальное число итераций N. Цикл While завершает работу при выполнении заданного условия выхода из цикла.

Цикл For, показанный на рисунке 6, генерирует случайное число каждую секунду 60 раз и отображает их в элементе отображения данных.

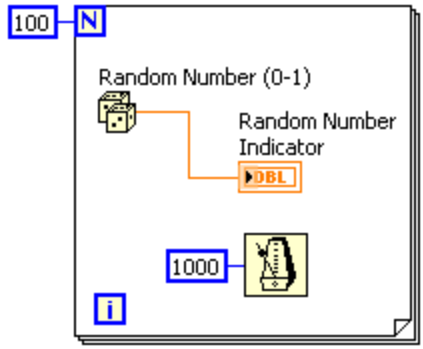


Рис. 6. Блок-диаграмма цикла For с генерацией случайного числа

Функции ожидания

 Функция Wait Until Next ms Multiple, показанная слева, обеспечивает интервал между итерациями, равный интервалу времени, необходимому для того, чтобы

миллисекундный счетчик достиг значения, кратного введенному пользователем. Эта функция используется для синхронизации действий. ФункциюWait Until Next ms Multiple вызывают внутри цикла для контроля скорости выполнения цикла.

Функция Wait Until Next ms Multiple обеспечивает интервал между итерациями, равный интервалу времени, необходимому внутреннему таймеру компьютера для достижения указанного кратного значения.

 Функция Wait(ms), показанная слева, добавляет время ожидания ко времени выполнения программы. Это может вызвать затруднения, если время выполнения программы является переменным.

Организация доступа к значениям предыдущих итераций цикла. При работе с циклами зачастую необходим доступ к значениям предыдущих итераций цикла. Например, в случае ВП, измеряющего температуру и отображающего ее на графике, для отображения текущего среднего значения температуры, необходимо использовать значения, полученные в предыдущих итерациях. Есть два пути доступа к этим данным: Shift Register (сдвиговый регистр) и Feedback Node (узел обратной связи).

Сдвиговые регистры. Сдвиговые регистры используются при работе с циклами для передачи значений от текущей итерации цикла к следующей. Сдвиговые регистры аналогичны статическим переменным в текстовых языках программирования

 Сдвиговый регистр выглядит как пара терминалов, показанных слева. Они расположены непосредственно друг против друга на противоположных вертикальных сторонах границы цикла.

Правый терминал содержит стрелку «вверх» и сохраняет данные по завершению текущей итерации. LabVIEW передает данные с этого регистра в следующую

итерацию цикла. Сдвиговый регистр создается щелчком правой кнопки мыши по границе цикла и выбором из контекстного меню пункта Add Shift Register. Сдвиговый регистр передает данные любого типа, он автоматически принимает тип первых поступивших на него данных. Данные, передаваемые на терминалы сдвигового регистра, должны быть одного типа.

Для того чтоб инициализировать сдвиговый регистр, необходимо передать на его левый терминал любое значение извне цикла. Если не инициализировать сдвиговый регистр, он использует значение, записанное в регистр во время последнего выполнения цикла или значение, используемое по умолчанию для данного типа данных, если цикл никогда не выполнялся.

Цикл с неинициализированным сдвиговым регистром используется при неоднократном запуске ВП для присвоения выходному значению сдвигового регистра значения, взятого с последнего выполнения ВП.

Чтобы сохранить информацию о состоянии между последующими запусками ВП, следует оставить вход левого терминала сдвигового регистра не определенным. После завершения выполнения цикла последнее значение, записанное в регистр, останется на правом терминале. При последующей передаче данных из цикла через правый терминал будет передано последнее значение, записанное в регистр.

Предусмотрена возможность создания нескольких сдвиговых регистров в одной структуре цикла. Если в одном цикле выполняется несколько операций, следует использовать сдвиговый регистр с несколькими терминалами для хранения данных, полученных в результате выполнения различных операций цикла. На рисунке 7 показано использование двух инициализированных сдвиговых регистров.

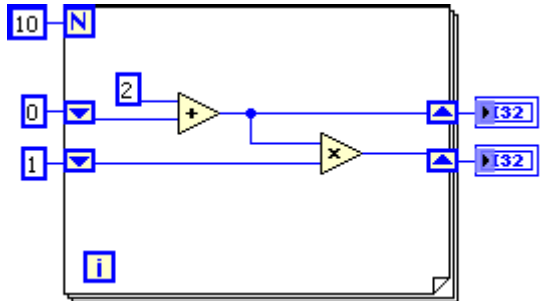


Рис. 7. Блок-диаграмма цикла с двумя сдвиговыми регистрами

Стек сдвиговых регистров.

Для создания стека сдвиговых регистров достаточно щелкнуть правой кнопкой мыши по левому терминалу и выбрать пункт контекстного меню Add Element. Стек сдвиговых регистров осуществляет доступ к значениям предыдущих итераций цикла. Стек сдвиговых регистров сохраняет данные предыдущей итерации и передает эти значения к следующей итерации.

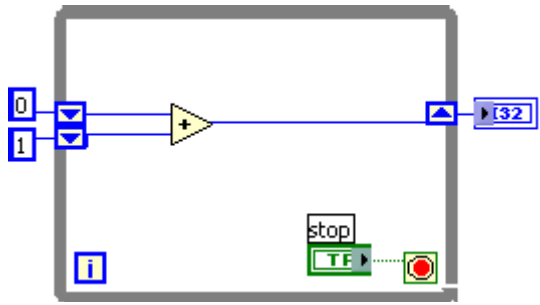


Рис. 8. Блок-диаграмма цикла со стеком сдвиговых регистров

Стек сдвиговых регистров может находиться только в левой части цикла, так как правый терминал лишь передает данные из текущей итерации в следующую. При добавлении еще двух сдвиговых регистров к левому терминалу данные

последних трех итераций переносятся на следующую итерацию, при этом значение последней итерации сохраняется в самом верхнем сдвиговом регистре. Второй терминал сохранят данные, переданные ему с предыдущей итерации, нижний терминал хранит данные, полученные две итерации назад.

Узлы обратной связи


Узел обратной связи, показанный слева, автоматически появляется в циклах While или For при соединении поля вывода данных подпрограммы ВП, функции или группы подпрограмм ВП и функций с полем ввода данных тех же самых подпрограмм ВП, функций или их групп. Как и сдвиговый регистр, узел обратной связи сохраняет данные любого типа по завершению текущей итерации и передает эти значения в следующую итерацию. Использование узлов обратной связи позволяет избежать большого количества проводников данных и соединений.

Можно поместить узел обратной связи внутри цикла ^ While или For, выбрав Feedback Node в палитре Structures. При помещении узла обратной связи на проводник данных до ответвления, передающего данные на выходной терминал цикла, узел обратной связи передает все значения на выходной терминал цикла. При помещении узла обратной связи на проводник после ответвления, передающего данные на выходной терминал цикла, узел обратной связи передаст все значения обратно на поле ввода данных ВП или функции, а затем передаст последнее значение на выходной терминал цикла. Следующее упражнение содержит пример работы узла обратной связи.

Пример1:

1. Доступ к данным предыдущих итераций BasicsI. Лицевая панель этого ВП, показанная на рисунке 9, уже создана

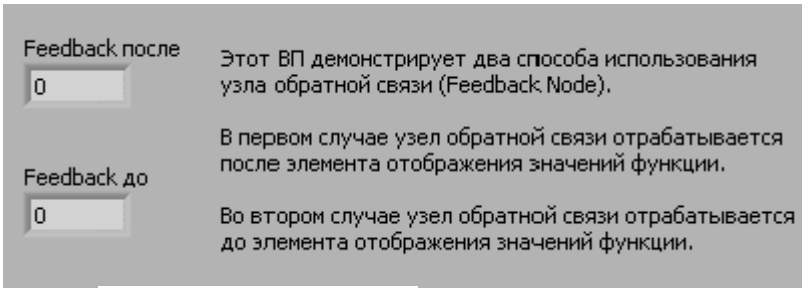


Рис. 9. Блок-диаграмма

2. Отобразите блок-диаграмму, показанную на рис. 18. Разместите лицевую панель и блок-диаграмму на экране так, чтобы они не перекрывали друг друга. Переместите палитры Tools и Functions, если это необходимо.

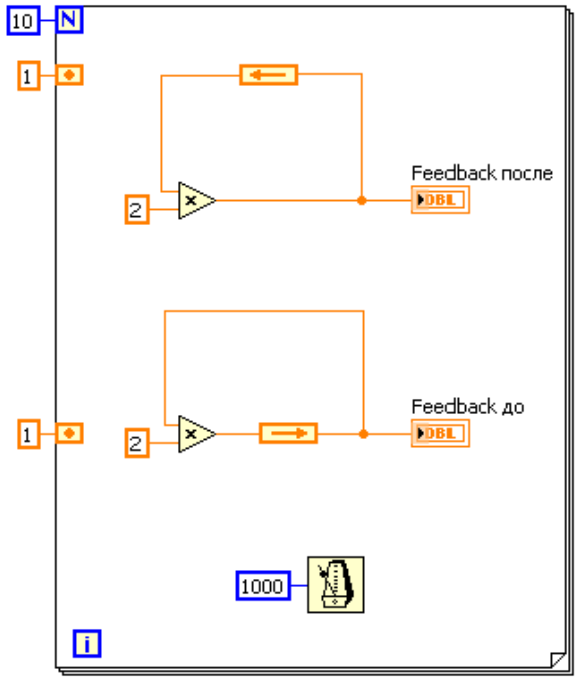


Рис. 10 - блок-диаграмма

Единица, соединенная с левым терминалом цикла For, инициализирует узел обратной связи. Таймер Wait Until Next ms Timer замедляет процесс выполнения программы. Для замедления выполнения процесса можно также использовать режим Highlight Execution (анимации выполнения блок-диаграммы).

На данной блок-диаграмме один и тот же процесс выполняется дважды, при этом узел обратной связи помещен в различных местах соединения.

Запустите ВП. Программа в верхней части сначала считывает значение узла обратной связи, инициализированного значением 1. Затем это значение передается функции Multiply (умножение).

Программа в нижней части сначала считывает значения узла обратной связи, инициализированного значением 1. Затем это значение передается на цифровой элемент отображения. Функция Multiply (умножение) не будет выполняться до следующей итерации цикла.

3. Активируйте режим анимации выполнения блок-диаграммы, нажав на показанную слева кнопку ^ Highlight Execution. Запустите ВП еще раз для наблюдения порядка выполнения программы. Отключите режим анимации для работы ВП в нормальном режиме.

4. Замените узел обратной связи сдвиговым регистром, как показано на следующей блок-диаграмме (рис. 9):

a. Выделите нижний узел обратной связи и нажмите клавишу , чтобы удалить его.

b. Щелкните правой кнопкой мыши по границе цикла и выберите пункт контекстного меню Add Shift Register.

c. Инициализируйте сдвиговый регистр значением 1.

d. Переименуйте верхний и нижний элементы отображения соответственно Узел обратной связи и Сдвиговый регистр.\

5. Запустите ВП. Обратите внимание, что узел обратной связи и сдвиговый регистр выполняют одинаковые

функции.

6. Не закрывайте ВП, перейдите к выполнению дополнительных упражнений или закройте ВП, не сохраняя его.\

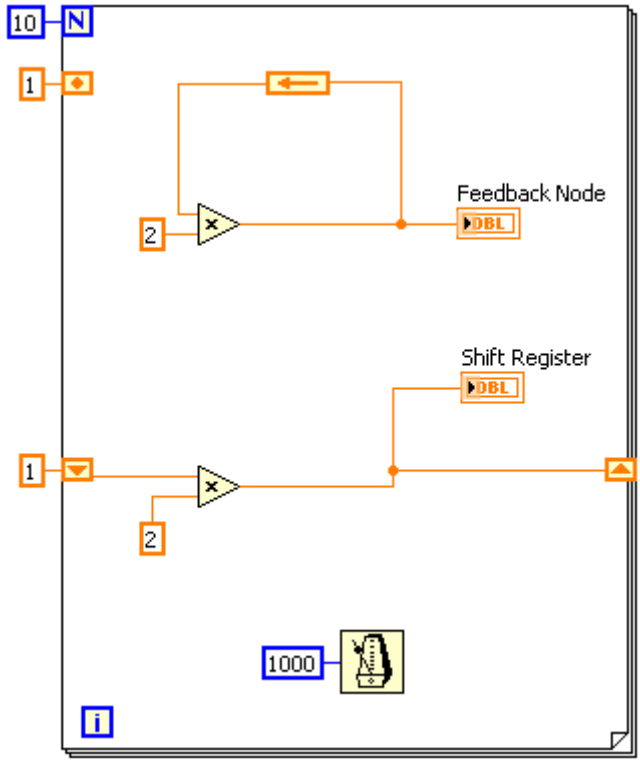


Рис. 11

Варианты заданий

1. Создать виртуальный прибор, генерирующий с интервалом в 1 сек. последовательность чисел Фибоначчи в виде таблицы. Обеспечить возможность очистки значений таблицы в произвольный момент времени.

Используемые компоненты:

Shift register

Add
Build table
Time delay

2. Создать виртуальный прибор, отображающий сумму или произведение двух сигналов (прямоугольного и гармонического) в зависимости от положения тумблера на лицевой панели.

Используемыекомпоненты:
Simulate signal
Formula
Waveform graph
Timedelay
Select
Toggleswitch

3. Создать виртуальный прибор, иллюстрирующий работу логических операций «И» «ИЛИ»«НЕ» на одной лицевой панели.

Используемыекомпоненты:
And
Or
Not
Led
Rocker

4. Создать виртуальный прибор, вычисляющий значение факториала. Реализовать вывод на цифровой индикатор промежуточных значений с интервалом 1сек. Алгоритм вычисления факториала реализовать в виде SubVI

Используемые компоненты:
Forloop
Shift register
Multiply
Increment
Numeric indicator
Timedelay

5. Создать виртуальный прибор, вычисляющий

значение x^y (возведение в степень для целых чисел – циклический алгоритм). Реализовать ввод исходных данных и вывод на цифровой индикатор результата и промежуточных значений с интервалом 1сек. Алгоритм возведения в степень реализовать в виде SubVI

Используемые компоненты:
For loop
Shift register
Multiply
(Decrement)
Numeric indicator
Numeric control
Time delay

6. Реализовать подпрограмму для вычисления суммы: $\sum_{n=0}^{\infty}(-1)^n \frac{x^{2n+1}}{(2n+1)!}$ создать VI с возможностью задания точности вычислений по данной формуле, а также значения переменной *x*.

7. Реализовать подпрограмму для вычисления суммы: $\sum_{k=0}^{\infty}\frac{(-1)^{k+1}}{(2k)!}\left(\frac{x}{3}\right)^{4k}$ с точностью 0.001. Значение *x* передать вкачестве параметра.

Лабораторная работа № 3
Массивы

Цель работы

Получение навыков работы с динамическими массивами с использованием циклических алгоритмов.

Задачи

Освоение технологии применения туннелей и автоиндексации.

Отчет:

- титульный лист;
- цели и задачи лабораторной работы;
- задание на лабораторную работу;
- результаты выполненной работы.

Справочно-методический материал

Массивы объединяют элементы одного типа данных.

Массив – это набор элементов определенной размерности. Элементами массива называют группу составляющих его объектов. Размерность массива – это совокупность столбцов (длина) и строк (высота), а также глубина массива. Массив может иметь одну и более размерностей, и до 2^{31}–1 элементов в каждом направлении, насколько позволяет оперативная память.

Данные, составляющие массив, могут быть любого типа: целочисленного, логического или строкового. Массив также может содержать элементы графического представления данных и кластеры.

Использовать массивы удобно при работе с группами данных одного типа и при накоплении данных после повторяющихся вычислений. Массивы идеально подходят для хранения данных, полученных с графиков, или накопленных во время работы циклов, причем одна итерация цикла создает один элемент массива.

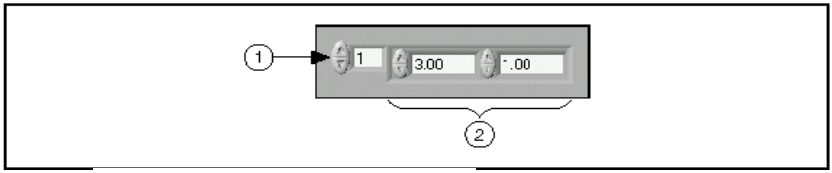
Нельзя создать массив, состоящий из массивов. Все элементы массива упорядочены. Чтобы к ним было легко обращаться, каждому элементу присвоен индекс. Нумерация элементов массива всегда начинается с 0.

Таким образом, индексы массива находятся в диапазоне от 0 до (n-1), где n - число элементов в массиве. Например, в массиве из девяти планет солнечной системы n=9, следовательно, значение индекса находится в пределах от 0 до 8. Земля является третьей планетой от Солнца, поэтому ее индекс равен 2.

Создание массива элементов управления и

отображения

Для создания массива элементов управления или отображения данных, как показано в примере, необходимо выбрать шаблон массива из палитрыControls>>Array & Cluster и поместить его на лицевую панель. Затем поместить в шаблон массива элемент управления либо отображения данных. Поместить в шаблон массива запрещенный элемент управления или отображения, например, двухкоординатный график осциллограмм (XY graph) не удастся.



1. Элемент индекса массива
2. Элементы значений массива

Рис.12. Элемент массива

Поместить объект в шаблон массива следует до того, как он будет использоваться на блок-диаграмме. Если этого не сделать, то шаблон массива не будет инициализирован, и использовать массив будет нельзя.

Двумерные массивы. В двумерном (2D) массиве элементы хранятся в виде матрицы. Таким образом, для размещения элемента требуется указание индекса столбца и строки. На иллюстрации показан двумерный массив, состоящий из 6 столбцов (длина) и 4 строк (высота). Количество элементов в массиве =24 (6.4=24).

Рисунок. 21. Двухмерный массив

Для увеличения размерности массива необходимо щелкнуть правой кнопкой мыши по элементу индекса и выбрать из контекстного меню пункт Add Dimension. С этой целью также можно использовать инструмент ПЕРЕМЕЩЕНИЕ. Для этого надо просто изменить размер элемента индекса. Ниже приведен пример неинициализированного двумерного массива элементов управления.

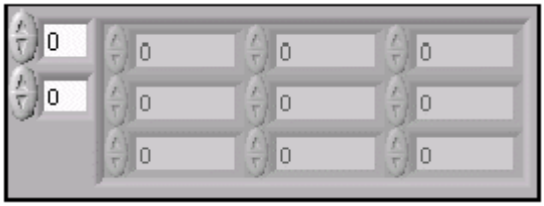


Рис. 13. Неинициализованный двумерный массив

Создание массива констант

Создать массив констант на блок-диаграмме можно, выбрав в палитре Functions>>Array шаблон Array Constant и поместив в него числовую константу. Массив констант удобно использовать для передачи данных в подпрограммы ВП.

Автоматическая индексация. Цикл For и цикл While могут автоматически накапливать массивы и проводить их индексацию на своих границах. Это свойство называется автоиндексацией. После соединения терминала данных массива с терминалом выхода из цикла каждая итерация цикла создает новый элемент массива. Из рисунка 13 видно, что проводник данных, соединяющий терминал данных массива с терминалом выхода из цикла стал толще, а сам терминал выхода из цикла окрашен в цвет терминала данных массива.

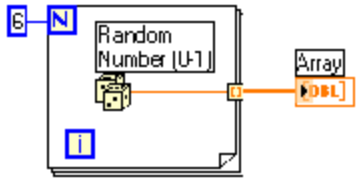


Рис. 14. Автоиндексация при работе циклов

Автоиндексация отключается щелчком правой кнопки мыши по терминалу входа/выхода из цикла и выбором пункта контекстного меню ^ Disable Indexing. Автоиндексацию следует отключать, например, в случае, когда нужно знать только последнее значение.

Ввиду того, что цикл For часто используется при работе с циклами, для него в LabVIEW автоиндексация включена по умолчанию. Для цикла While автоиндексация по умолчанию отключена. Для того, чтобы включить автоиндексацию, необходимо щелкнуть правой кнопкой мыши по терминалу входа/выхода из цикла и выбрать в контекстном меню пункт Enable Indexing.

Создание двумерных (2D) массивов

Для создания двумерных массивов необходимо использовать два цикла For, один внутри другого. Как показано на иллюстрации, внешний цикл создает элементы массива в строке, а внутренний цикл создает элементы массива в столбце.

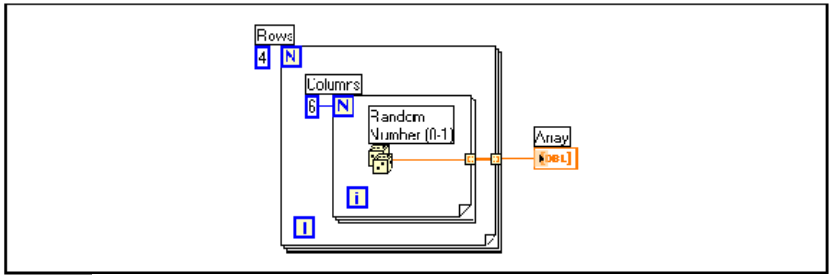


Рис. 15. Создание двумерного массива с помощью двух циклов

Использование автоиндексации для установки значения терминала количества итераций цикла

При включенной автоиндексации массива, подключенного к терминалу входа в цикл For, LabVIEW автоматически устанавливает значение терминала количества итераций цикла N равным размерности массива.

Таким образом, отпадает необходимость задания значения терминалу ^ N. В следующем примере цикл For будет выполнен ровно столько раз, сколько элементов в массиве. Как правило, стрелка на кнопке Run сломана, если терминал количества итераций цикла не подключен, однако в этом примере стрелка цела, что говорит о возможности запуска ВП.

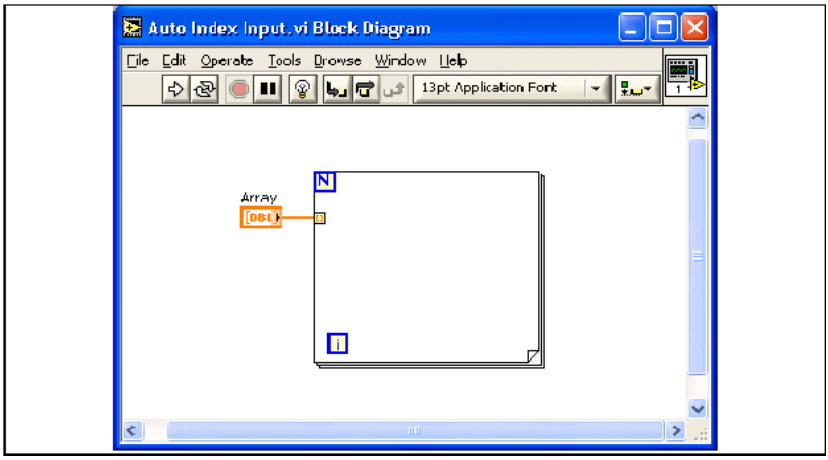


Рис. 16. Задание количества итераций в цикле

Если автоиндексация установлена более чем для одного терминала входа в цикл или явно задано значение терминала количества итераций цикла N, то значением терминала N станет меньшая из величин. Например, если соединить массив из 10 элементов с терминалом входа в цикл, а значение терминала количества итераций установить

равным 15, то цикл выполнит 10 итераций.

Пример: Работа с массивами

1. Откройте новый ВП и создайте лицевую панель, как показано ниже.

a. В палитре ^ Controls>>Array & Cluster выберите шаблон массива.

b. Созданному массиву присвойте имя Массив случайных чисел.

c. Поместите внутрь шаблона массива цифровой элемент отображения, расположенный в палитре Controls>>Numeric.

d. С помощью инструмента ПЕРЕМЕЩЕНИЕ измените размер массива таким образом, чтобы он содержал 10 элементов.

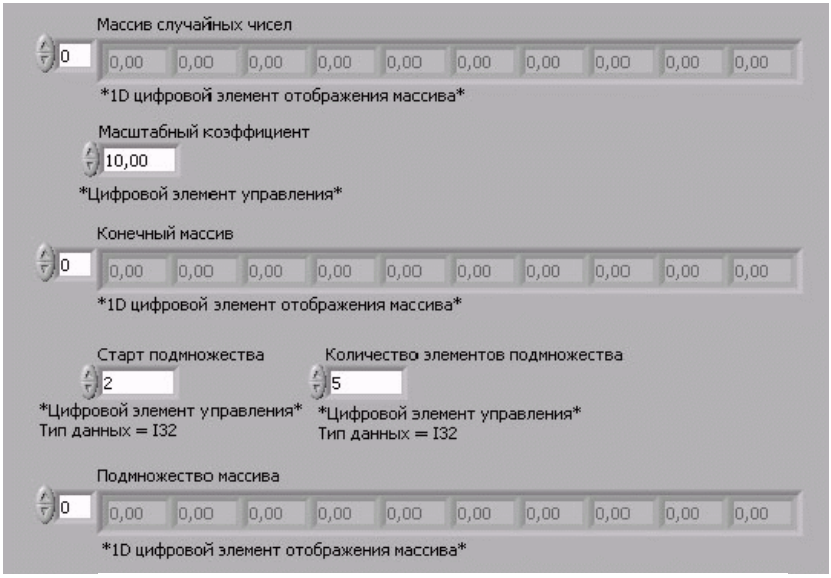


Рис. 17. Лицевая панель ВП «работа с массивами»

e. Нажмите и удерживайте клавишу <^ Ctrl> и, перемещая элемент Массив случайных чисел, создать две его

копии.

f. Копиям присвойте имена Конечный Массив и Подмножество Массива.

g. Создайте три цифровых элемента управления и присвойте им имена ^ Масштабный коэффициент, Старт подмножества и Количество элементов подмножества.

h. Щелкните правой кнопкой мыши по элементам Старт подмножества и Количество элементов подмножества, в контекстном меню выберите пунктRepresentation, затем пункт I32.

i. Значения элементов управления данных пока не изменяйте.

2. Постройте блок-диаграмму, как показано ниже.

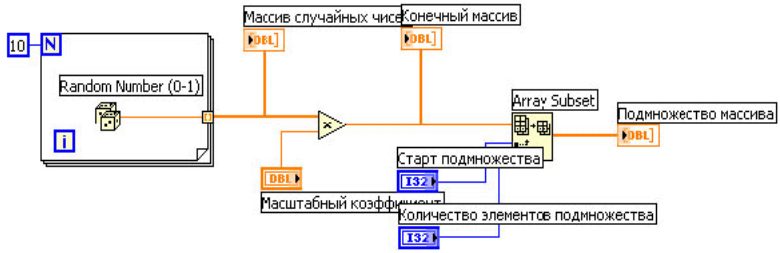


Рис. 18. Блок-диаграмма «Работа с массивами»

 Выберите функцию Random Number (0-1), расположенную в палитре Functions>>Numeric. Эта функция будет генерировать случайное число в пределах от 0 до 1.

 Выберите цикл For, расположенный в палитре Functions>>Structures. Этот цикл на терминале выхода накапливает массив из 10 случайных чисел. Терминалу количества итераций присвойте значение 10.

Выберите функцию Array Subset, расположенную в палитре Functions>>Array. Эта функция выдает подмножество массива, начиная со значения, введенного в элементе Старт подмножества и будет содержать количество элементов, указанное в элементе Количество элементов подмножества.

3. Сохраните ВП под именем ^ *Работа с массивами.vi*

4. Перейдите на лицевую панель, измените значения элементов управления и запустите ВП.

Цикл For совершит 10 итераций. Каждая итерация создаст случайное число и сохранит его в терминале выхода из цикла. В элементе Массив случайных чисел отобразится массив из 10 случайных чисел. ВП умножит каждое значение этого массива на число, введенное в элемент управления Масштабный коэффициент, для создания массива, отображаемого в индикаторе Конечный массив. ВП выделит подмножество из получившегося массива, начиная со значения в элементе Старт подмножества, длиной, указанной в элементе Количество элементов подмножества, и отобразит это подмножество в индикаторе Подмножество массива.

5. Закройте ВП.

Варианты заданий

1. Создать виртуальный прибор, заполняющий двумерный массив случайными числами в бесконечном цикле. Добавить элемент управления (кнопку) с помощью которого происходит остановка цикла и подсчет среднего арифметического значения элементов массива. Размер массива – произвольный.

2. Реализовать VI для заполнения двумерного массива размером 10 х 10 элементов возрастающими числами от 1 до 100.

3. Присвоить элементам двумерного массива расположенным выше главной диагонали случайные значения.

4. Создать SubVI для подсчета суммы элементов произвольной строки двумерного массива. Продемонстрировать его работу на трех различных массивах.

5. Заполнить элементы одномерного массива из 100 элементов числами Фибоначчи. Вычисления последовательности чисел реализовать в виде подпрограммы.

6. Заполнить четные строки двухмерного массива нулями, нечетные заполнить единицами. Размер массива – произвольный.

7. Подсчитать сумму элементов двумерного массива находящихся выше главной диагонали.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Суранов, А. Я. LabVIEW 7: справочник по функциям [Текст]/А.Я. Суранов. — М. : ДМК Пресс, 2005. — 510 с.
2. Тревис, Д. LabVIEW для всех : пер. с англ. [Текст] / Д. Тревис. — М. : ПриборКомплект : ДМК Пресс, 2005. — 537 с.
3. Евдокимов, Ю. К. Labview для радиоинженера: от виртуальной модели до реального прибора [Текст]: учеб. пособие для вузов/ Ю.К. Евдокимов. — М. : ДМК Пресс, 2007. — 400 с.

СОДЕРЖАНИЕ

СТРУКТУРА И ВОЗМОЖНОСТИ ПАКЕТА LabVIEW. СОЗДАНИЕ ВИРТУАЛЬНОГО ИЗМЕРИТЕЛЬНОГО СРЕДСТВА 2

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

к выполнению лабораторных работ по дисциплине «Приемоусилительные и видеотелевизионные системы» для студентов направления 11.03.03 «Конструирования и технология электронных средств» (профиль «Проектирование и технология радиоэлектронных средств») всех форм обучения

Составители:
доктор. техн. наук А. В. Башкиров,
канд. техн. наук И.С. Бобылкин.

Компьютерный набор И.С. Бобылкин.

Подписано к изданию _____.
Уч.-изд. л. _____.

ФГБОУ ВО «Воронежский государственный технический университет»
394026 Воронеж, Московский просп., 14