ФЕДЕРАЛЬНОЕ АГЕНТСТВО ПО ОБРАЗОВАНИЮ

Нижегородский государственный университет им. Н.И. Лобачевского

Национальный проект «Образование»

Инновационная образовательная программа ННГУ. Образовательно-научный центр

«Информационно-телекоммуникационные системы:

физические основы и математическое обеспечение»

С.Ю. Лупов, С.И. Муякшин, В.В. Шарков

LabVIEW в примерах и задачах

Учебно-методические материалы по программе повышения квалификации «Обучение технологиям National Instruments»

Нижний Новгород

2007

Учебно-методические материалы подготовлены в рамках инновационной образовательной программы ННГУ: Образовательно-научный центр «Информационно-телекоммуникационные системы: физические основы и математическое обеспечение»

Лупов С.Ю., Муякшин С.И., Шарков В.В. LabVIEW в примерах и задачах. Учебно-методические материалы по программе повышения квалификации «Обучение технологиям National Instruments». Нижний Новгород, 2007, 101 с.

«LabVIEW в примерах и задачах» - базовый курс по основам графического программирования в графической среде программирования LabVIEW 8.20. Пакет LabVIEW формализует этап создания алгоритма работы прибора, описывая этот алгоритм в виде блок-схемы. В учебно-методическом материале отражены все этапы создания виртуального прибора: регистрация сигнала, обработка, отображение. Первый раздел учебного пособия посвящен основам программирования, второй – основным принципам цифровой обработки сигналов, третий – созданию систем сбора данных.

ПРЕДИСЛОВИЕ

LabVIEW (*Laboratory Virtual Instrument Engineering Workbench*) - среда разработки прикладных программ, созданная фирмой *National Instruments* (США). В ней используется интуитивно понятный язык графического программирования *G*. Его освоение не требует знания традиционных текстовых языков программирования. LabVIEW предоставляет широкие возможности для проведения вычислений и математического моделирования. В этом отношении среда LabVIEW конкурентоспособна с такими известными системами компьютерной математики, как MATLAB, MathCAD, Mathematica, MAPLE.

Однако наиболее полно возможности LabVIEW раскрываются при создании приборов и систем для измерений физических величин в научных экспериментах, лабораторных и промышленных установках. Важным достоинством LabVIEW является возможность управления процессом измерения в автоматическом или интерактивном режиме. Для обработки и анализа данных используется обширный набор функциональных библиотек (общего назначения и специализированных). Взаимодействие с исследователем или оператором осуществляется с помощью продуманного и простого в программировании графического интерфейса. С помощью программ-драйверов LabVIEW эффективно взаимодействует с разнообразными платами ввода/вывода аналоговых и цифровых сигналов, модулями ввода видеосигналов, а также со специализированными модульными приборами (осциллографы, анализаторы спектра, генераторы сигналов и т.д.).

Последние версии LabVIEW ориентированы на создание распределенных и дистанционных систем измерений. Это позволяет обеспечить доступ на расстоянии к уникальным экспериментальным стендам и организовать дистанционное обучение. Возможности базового пакета могут быть расширены с помощью специализированных модулей и функциональных библиотек.

Базовый пакет LabVIEW прошел 20-ти летний эволюционный путь развития и обновляется практически каждый год. На момент начала работы над данным пособием актуальной являлась версия LabVIEW 8.20.

Цель данного пособия - познакомить с основами разработки приложений в среде LabVIEW, включая создание простых систем сбора и обработки данных с использованием DAQ-карт. Изложение в основном следует курсам фирмы-разработчика "Basic I" и "Data

acquisition". Отдельные темы раскрыты несколько шире, что отражает собственный опыт преподавания предмета. Как явствует из названия, обучение основано на примерах частично оригинальных, частично заимствованных из источников, перечисленных ниже. Содержание оригинальных примеров и задач отражает специфику Радиофизического факультета.

1. ОСНОВЫ ПРОГРАММИРОВАНИЯ В ГРАФИЧЕСКОЙ СРЕДЕ LABVIEW

1.1. Графическая среда программирования *LabVIEW*

1.1.1 Основные элементы среды программирования *LabVIEW*

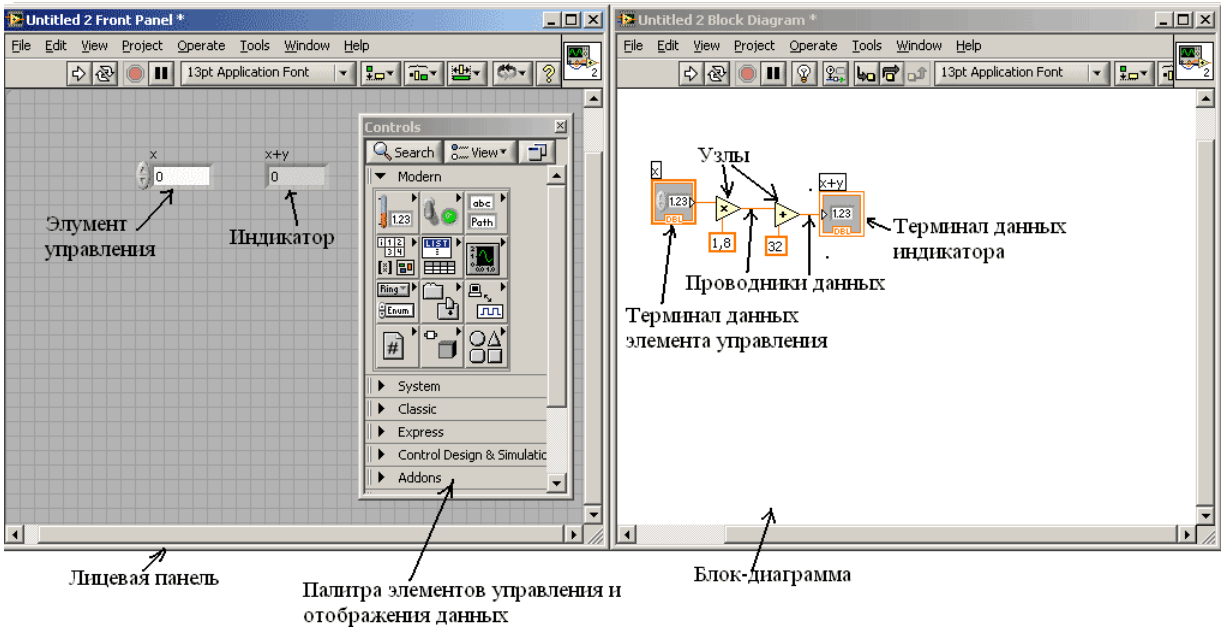


Рис.1 Графическая среда программирования *LabVIEW*

Создание программ в графической среде программирования *LabVIEW* производится в двух окнах, называемых:

Front Panel (лицевая панель);

Block Diagram (блок-диаграмма).

На лицевой панели разрабатывается внешний вид будущего виртуального прибора (все программы разработанные на *LabVIEW* называются виртуальными приборами (Virtual Instruments). На ней создаются элементы управления и отображения, которые являются интерактивными средствами ввода и вывода данных этого виртуального прибора.

Элементы управления (Controls) – кнопки и другие устройства ввода данных.

Элементы отображения (Indicators) – графики, светодиоды и другие индикаторы.

Элементы управления моделируют устройства ввода данных и передают данные на блок-диаграмму ВП. Элементы отображения моделируют устройства вывода и отображения данных, которые получает или генерирует блок-диаграмма (рис.1).

На блок-диаграмме разрабатывается исходный код будущего виртуального прибора. В отличии от классических языков программирования, таких как Си, Паскаль, ФОРТРАН, исходный код *LabVIEW* представляет собой блок-диаграмму, где все команды, операторы циклов и сравнения изображаются графическими обозначениями. Блок-диаграмма состоит из узлов, терминалов и проводников данных (рис.1).

Узлы (Node) (рис.1) – это объекты на блок-диаграмме, которые имеют одно или более полей ввода/вывода данных и выполняют алгоритмические операции виртуального прибора. Они аналогичны операторам, функциям и подпрограммам текстовых языков программирования. Узлы включают в себя функции (*functions*), подпрограммы (*SubVI*) и структуры (*Structures*). Подпрограмма - виртуальный прибор (ВПП), который можно использовать на блок-диаграмме другого ВП в качестве подпрограммы. Структуры *(Strucures)* - это элементы управления процессом, такие как структура Последовательности (*Flat sequence*), структура Варианта (*Case*), цикл по условию (*While*) и т.д.

Объекты лицевой панели на блок-диаграмме отображаются в виде *терминалов данных (Data terminals)* (рис.1). *Терминалы данных* обеспечивают обмен данными между лицевой панелью и блок-диаграммой. Различают терминалы данных следующих типов: терминалы управления и отображения данных, терминалы узлов. Терминалы управления и отображения относятся к средствам управления и отображения данных на лицевой панели. Данные, введенные в элементы управления на лицевой панели, поступают на блок-диаграмму через терминалы управления.

Данные между объектами блок-диаграммы передаются по соединительным линиям - по *проводникам данных (Wires)* (рис.1). *Проводник данных* аналогичен переменным в текстовых языках программирования. Каждый проводник данных имеет единственный источник данных, но может передавать их ко многим функциям. Проводники данных различаются цветом, стилем и толщиной линии, в зависимости от типа передаваемых данных.

В среде LabVIEW объекты соединяются проводниками данных после их помещения на блок-диаграмму.

Запустим графическую среду разработчика программного обеспечения *National Instruments LabVIEW 8.2*. В появившемся окне выберем раздел *Blank VI* (пустой виртуальный

прибор). После чего откроются два окна (рис.1): лицевая панель (*Front Panel*) и блок-диаграмма (*Block Diagram*).

Переключаться между окнами можно несколькими способами:

- щелкнув мышкой по соответствующему окну;
- комбинацией клавиш <*Ctrl*><*E*>

Ряд кнопок, расположенный под главным меню, называется инструментальной панелью.

Кнопка однократного запуска *Run* - запускает виртуальный прибор.

Кнопка непрерывного запуска *Run Continuously* – виртуальный прибор выполняется многократно до момента принудительной остановки

Во время выполнения виртуальный прибор активируется кнопка *Abort Execution*. Эта кнопка используется для немедленной остановки выполнения виртуального прибора.

1.1.2. Палитра элементов управления и индикаторов

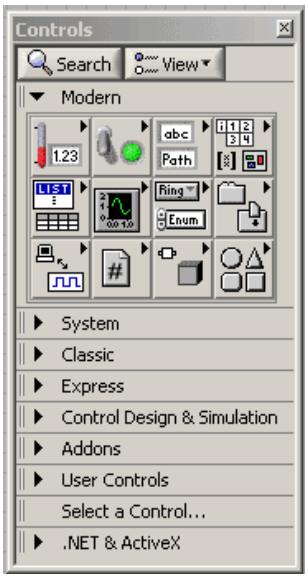


Рис.2 Палитра элементов управления и отображения

Для размещения элементов управления и отображения данных на лицевую панель используется палитра элементов управления и отображения (рис.2). Палитра элементов управления и отображения доступна только с лицевой панели. Для вывода палитры на экран следует щелкнуть правой кнопкой мыши в рабочем пространстве лицевой панели.

Все элементы управления и отображения на палитре сгруппированы по разделам:

- *Modern* – элементы управления и отображения имеют современный стиль (в данном пособии будет использоваться в основном этот раздел);
- *System* –элементы управления и отображения имеют стиль соответствующий данной операционной системе (кнопки, полосы прокрутки и т.д.);
- *Classic* – элементы управления и отображения имеют классический стиль (альтернатива стилю *Modern*)
- *Express* – распространенные элементы управления и отображения
- и т.д.

Каждый раздел может делиться, в свою очередь, на подразделы (числовые, строковые индикаторы, кнопки и т.д.).

1.1.2. Палитра функций

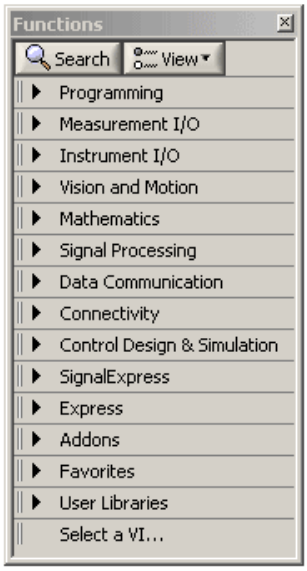


Рис.3 Палитра функций

Как было сказано ранее на блоке - диаграмм разрабатывается исходный текст программы. Для этого используется палитра функций (*Functions pallete*). Для вывода палитры на экран следует щелкнуть правой кнопкой мыши в рабочем пространстве блока-диаграмм.

Все элементы на палитре сгруппированы по разделам:

- *Programming* – здесь собраны все основные функции, структуры цикла, сравнения, операторы сложения, вычитания, необходимые для создания большинства приложений;
- *Instrument I/O* – здесь собраны функции для работы с различными устройствами ввода-вывода (COM-порт и др.);

- *Mathematics* – здесь собраны функции для математических вычислений (решение системы уравнений, решение дифференциальных уравнений и др.);
- *Signal Processing* – здесь собраны функции связанные с цифровой обработкой и анализом дискретных сигналов (цифровые фильтры, быстрое преобразование Фурье и др.);
- *Express* – распространенные функции, связанные как с вычислениями, так и с обработкой данных;
- *Select a VI...* – функции созданные пользователем.

1.1.3. Палитра Инструментов

Рис.4 Палитра инструментов

Создавать, изменять и отлаживать ВП можно, используя палитру инструментов (*Tools Palette*). палитра инструментов доступна как на лицевой панели, так и на блок-диаграмме. Термин инструмент подразумевает специальный операционный режим курсора мыши. При выборе определенного инструмента значок курсора изменяется на значок данного инструмента. Палитра инструментов вызывается одновременным нажатием клавиши <Shift> и правой клавиши мыши. Палитру Инструментов можно размещать в любой области рабочего пространства блок-диаграммы и лицевой панели.

Если включен автоматический выбор инструмента, то при наведении курсора на объект лицевой панели или блок-диаграммы LabVIEW автоматически выбирает соответствующий инструмент из палитры инструментов. Автоматический выбор инструмента включается нажатием на кнопку *Automatic Tool Selection* палитры инструментов (прямоугольник в правом верхнем углу палитры: зеленый – включен, черный – выключен) или нажатием клавиш <*Shift-Tab*>.

Инструмент *УПРАВЛЕНИЕ* (<палец>) используется для изменения значения элементов управления или ввода текста. При наведении курсора на такой элемент как строковый элемент управления, значок инструмента меняется:

Инструмент *ПЕРЕМЕЩЕНИЕ* (<стрелка>) используется для выбора, перемещения или изменения размеров объектов. При наведении инструмента на объект изменяемого размера значок инструмента меняется:

Инструмент *ВВОД ТЕКСТА* (<А>) используется для редактирования текста и создания свободных меток. При создании свободных меток значок инструмента меняется:

Инструмент *СОЕДИНЕНИЕ* (<катушка>) создает проводники данных, соединяя объекты на блок-диаграмме.

Инструмент *ВЫЗОВ КОНТЕКСТНОГО МЕНЮ* (<меню>) вызывает контекстное меню соответствующего объекта по щелчку левой кнопки мыши.

Инструмент *БЫСТРАЯ ПРОКРУТКА ЭКРАНА* (<рука>) используется для просмотра окна без использования полосы прокрутки.

Инструмент *ВВОД КОНТРОЛЬНОЙ ТОЧКИ* (<контрольная точка>) позволяет расставлять контрольные точки на ВП, функциях, узлах, проводниках данных, структурах и приостанавливать в них выполнение программы.

Инструмент *УСТАНОВКА ОТЛАДОЧНЫХ ИНДИКАТОРОВ* (<пробник>) дает возможность исследовать поток данных в проводниках блок-диаграммы

Используется для просмотра промежуточных значений при наличии сомнительных или неожиданных результатов работы ВП.

Инструмент *КОПИРОВАНИЕ ЦВЕТА* (<пипетка>) предназначен для копирования цвета с последующей вставкой с помощью инструмента РАСКРАШИВАНИЕ (<кисть>).

Инструмента *РАСКРАШИВАНИЕ* (<кисть>) позволяет изменить цвет объекта. Он также отображает текущий передний план и параметры настройки цвета фона.

Если автоматический выбор инструмента выключен, можно менять инструменты палитры инструментов с помощью клавиши <*Tab*>. Для переключения между инструментом *ПЕРЕМЕЩЕНИЕ* и *СОЕДИНЕНИЕ* на блок-диаграмме или между инструментом *ПЕРЕМЕЩЕНИЕ* и *УПРАВЛЕНИЕ* на лицевой панели - достаточно нажать пробел.

1.1.4. Справочная система в LabVIEW

Окно контекстной справки (*Context Help*) выводится на экран из пункта главного меню Помощь (*Help→Show Context Help*) или вводом <*CtrI-H*> с клавиатуры.

При наведении курсора на объект лицевой панели или блок-диаграммы в окне контекстной справки (*Context Help*) появляются иконка подпрограммы ВП, функции, константы, элементов управления или отображения данных с указанием всех полей ввода/вывода данных. При наведении курсора на опции диалогового окна в окне контекстной справки (*Context Help*) появляется описание этих опций. При этом поля, обязательные для соединения, выделены жирным шрифтом, рекомендуемые для соединения поля представлены обычным шрифтом, а дополнительные (необязательные) поля - выделены серым или вообще не показаны.

Для отображения встроенной помощи (*LabVIEW Help*) можно нажать кнопку *Detailed Help* в окне контекстной справки (*Context Help*)

Встроенная помощь *LabVIEW* содержит детальные описания большинства палитр, меню, инструментов, виртуальных приборов и функций, включает в себя пошаговую инструкцию использования особенностей *LabVIEW* и связана с руководством пользователя (*LabVIEW Tutorial*), PDF версией учебника *LabVIEW* и технической поддержкой на Web-сайте *National Instruments*.

1.2. Примеры программ на языке графического программирования *LabVIEW*

1.2.1. Первая программа

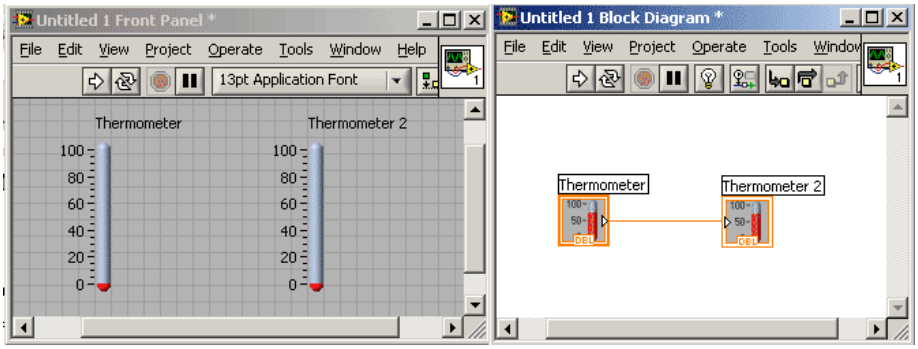


Рис.5 Первая программа

Создадим программу, которая выводит на индикатор значения с элемента управления (рис.5).

Для этого на лицевой панели разместим два индикатора *Thermometer* (на палитре элементов управления и индикаторов выберем: *Modern→Numeric→ Thermometer*). На блок – диаграмме появились две иконки, называемые терминалами данных, которые соответствуют

размещенным на лицевой панели индикаторам. Один из индикаторов необходимо сделать элементом управления. Для этого, наведем на него курсор мышки (это можно сделать, как на лицевой панели, так и на блок – диаграмме) и, нажав правую клавишу во всплывающем меню, выберем пункт *Change to Control* (сменить на элемент управления). При этом действии на блок – диаграмме у соответствующей иконки треугольник с правой стороны переместился на левую (рис.5). У индикаторов треугольник означает вход, на который необходимо подавать соответствующие значения, а у элементов управления – выход, с которого считываются значения. При подведении курсора мышки к треугольнику (выходу) элемента управления, курсор примет вид катушки. После однократного нажатия левой клавиши мышки, потяните мышку ко входу индикатора *Thermometer 2*. За курсором потянется проводник. Подведем проводник к входу индикатора и, как только курсор превратится снова в катушку, произведем однократное нажатие мышки. После этого блок диаграмма и лицевая панель будут выглядеть как на рис.5.

Работоспособность программы лучше проверять нажатием кнопки *Run Continuously* (циклический запуск). При изменении мышкой значения элемента управления (*Thermometer*) изменяются значения на индикаторе (*Thermometer 2*).

Задание:

Остановите программу нажатием клавиши *Stop*, разорвите проводник соединяющий элемент управления и индикатор (для этого выделите его мышкой и нажмите клавишу <*Delete*>). Запустите программу циклически еще раз и сравните результат работы.

1.2.2. Программа преобразующая температуру представленную в градусах Цельсия в температуру по Фаренгейту

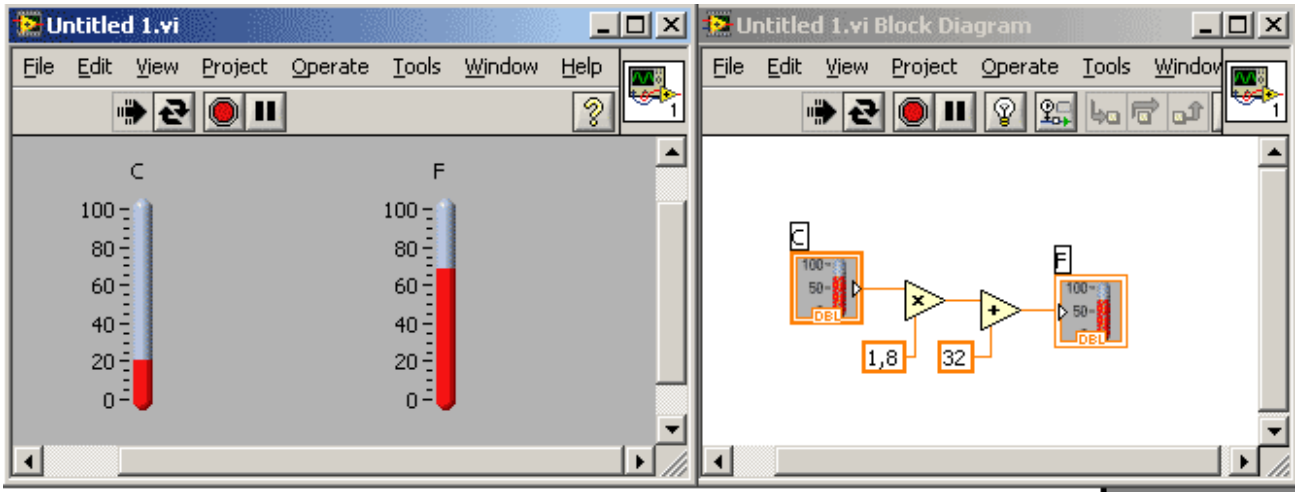


Рис.6 Программа, конвертирующая температуру в градусах Цельсия в температуру по Фаренгейту

Создадим простую программу, конвертирующую температуру представленную в градусах Цельсия в температуру по Фаренгейту, используя формулу:

$$F=1{,}8\cdot C+32,$$

где F – температура по Фаренгейту, C – температура в градусах Цельсия.

Выполним следующие действия:

- Создайте новый виртуальный прибор (*Blank VI*);
- На лицевой панели разместите два индикатора *Thermometer* (на палитре элементов управления и индикаторов выберем: *Modern→Numeric→ Thermometer*);
- Двойным нажатием левой клавиши мышки по меткам индикаторов выделите их и переименуйте в “*C*” и “*F*”;
- Индикатор с меткой “*C*” сделайте элементом управления;
- Перейдите на блок-диаграмму.
- Выберите функции Умножение (*Multiply*) и Сложение (*Add*) из палитры функций в разделе Арифметические функции (*Programming→Numeric*);
- Поместите выбранные функции на блок-диаграмму.
- Создайте числовую константу у соответствующего входа функции умножения. Для этого подведите курсор мышки к этому входу, чтобы появился инструмент Катушка и однократно нажмите правую клавишу мышки. Во всплывающем меню выберите

Create→Constant. После размещения числовой константы на блок-диаграмме поле ввода ее значений подсвечивается и готово для редактирования. Присвойте ей значение 1,8.

- Точно также создайте вторую константу и присвойте ей значение 32, как показано на рис.6
- Запустите программу, нажав клавишу *Run Continuously*.

Задания:

1. Измените программу так, чтобы она преобразовывала значение температуры по Фаренгейту в градусы Цельсия.
2. Создайте новую программу. На лицевой панели разместите два числовых элемента управления, назовите их "*X*" и "*Y*" и два числовых индикатора (рис.7). На блок-диаграмме реализуйте алгоритм, такой, чтобы на одном индикаторе выводилась сумма, а на другом разность значений, введенных в элементы управления.

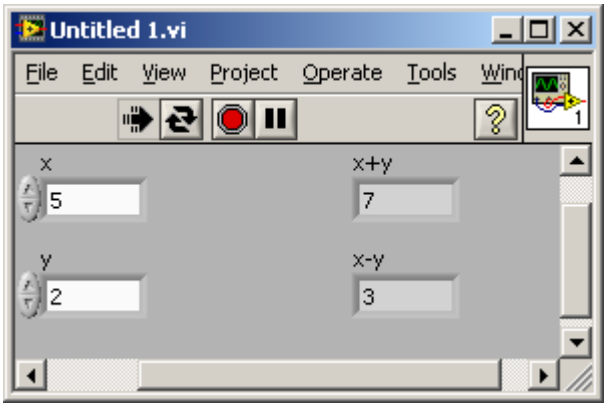


Рис.7 Лицевая панель виртуального прибора для последнего задания

1.3. Циклы

Для выполнения программы или части программы определенное количество раз или пока не выполнится какое-то условие, например, нажатие кнопки *Stop*, в *LabVIEW*, используются циклы

1.3.1. Цикл по Условию (While)

Цикл по условию (*While*) аналогичен циклу *While*, используемому в текстовом языке программирования *Cи*, выполняет многократное повторение операции над потоком данных,

пока не выполнится логическое условие выхода. Цикл *While* расположен на палитре функций в разделе Структуры (*Programming→Structures*)

После того как цикл найден и выбран на палитре функций, следует с помощью курсора изменить промежуточные границы структуры для выделения части 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 объекта.

Блок-диаграмма цикла по условию (*While*) выполняется до тех пор, пока не выполнится условие выхода. По умолчанию, терминал условия выхода указывает, что цикл будет выполняться до поступления на терминал значения *ЛОЖЬ* (*FALSE*). В этом случае терминал условия выхода называется терминалом «*Продолжение Если Истина* (*Continue If True*)».

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

Пример цикла *While*:

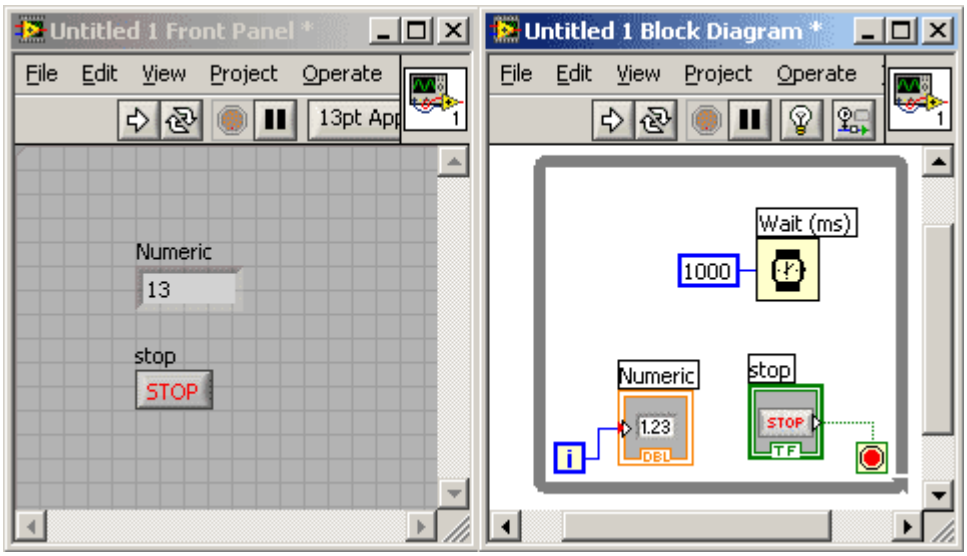


Рис.8 Блок-диаграмма и лицевая панель программы с циклом *While*

Программа (рис.8) увеличивает значение индикатора *Numeric* на 1 с интервалом в одну секунду.

Создайте программу, приведенную на рисунке 8. Числовой индикатор *Numeric* и кнопку *Stop* удобнее создать на блоке-диаграмм. Для этого подведите курсор мышки к соответствующему терминалу, пока курсор не превратиться в катушку, произведите

однократное нажатие на правую клавишу мышки и выберите *Create→Indicator* для терминала «*i*» или *Create→Control* для терминала условия выхода.

Функция *Wait* находится в палитре функций в разделе *Programming→Timing*. Входной параметр определяет время задержки в миллисекундах.

Запускать программу на выполнение следует кнопкой *Run*, расположенной на инструментальной панели, а останавливать – кнопкой *Stop*, созданной на лицевой панели.

Измените предыдущий пример так, чтобы цикл прекращал выполняться после ста итераций, если до этого не была нажата кнопка *Stop*. Для этого следует использовать функцию *Equal* (равенство) в разделе *Programming→Comparison* и функцию *Or* (логическое *ИЛИ*) в разделе *Programming→Boolean*.

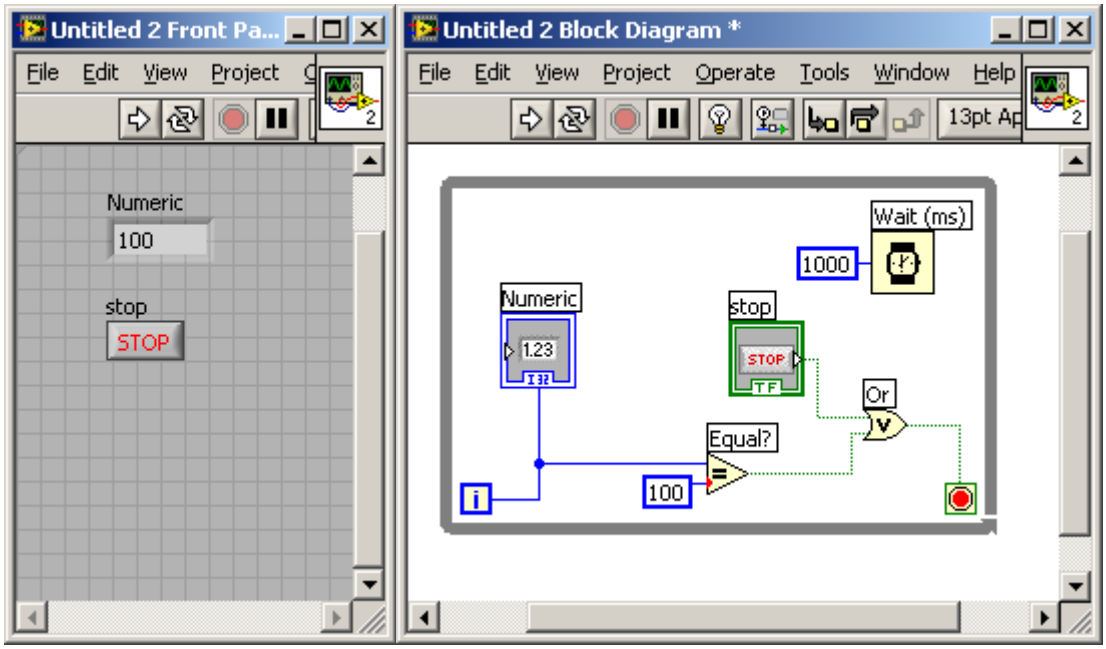


Рис.9 Модификация виртуального прибора, представленного на рис.8

1.3.2. Цикл с фиксированным числом итераций (For)

Цикл с фиксированным числом итераций (*For*) выполняет повторяющиеся операции над потоком данных определенное количество раз. Цикл с фиксированным числом итераций (*For*), расположен на палитре функций в разделе *Programming→Structures*. Значение, присвоенное терминалу максимального числа итераций «*N*» цикла определяет максимальное количество повторений операций над потоком данных. Терминал счетчика итераций «*i*»

содержит количество выполненных итераций. Начальное значение счетчика итераций всегда равно нулю.

Цикл с фиксированным числом итераций (*For*) отличается от цикла по условию (*While*) тем, что завершает работу, выполнив заданное максимальное число итераций «*N*». Цикл по Условию (*While*) завершает работу после выполнения заданного условия выхода из цикла.

Пример цикла *for*:

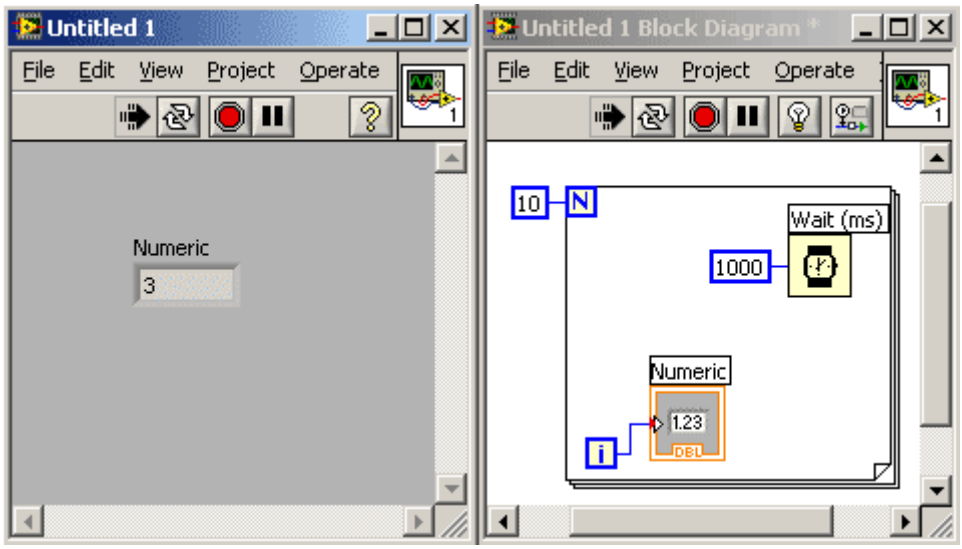


Рис.10 Блок-диаграмма и лицевая панель программы с циклом *For*

Программа (рис.10), также как и предыдущая, увеличивает значение индикатора *Numeric* с 0 до 9 с шагом в 1 и с интервалом в одну секунду.

Изменим предыдущий пример (рис.10) так, чтобы на индикатор *Numeric* выводились числа от 10 до 0 с шагом -1.

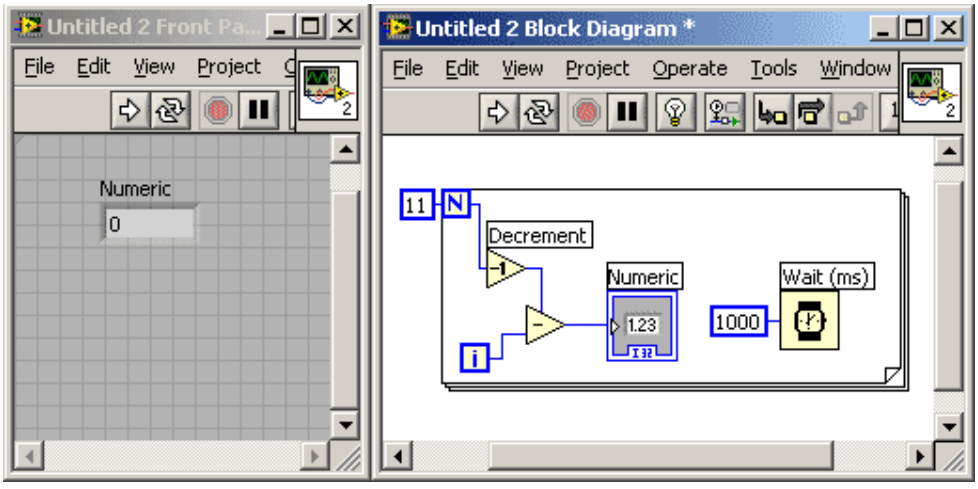


Рис.11 Виртуальный прибор отображающий на индикатор числа от 10 до 0

1.3.3. Сдвиговые регистры

Сдвиговые регистры используются при работе с циклами для передачи значений от текущей итерации цикла к следующей. Сдвиговый регистр создается щелчком правой клавиши мышки на границе цикла и выбором пункта *Add Shift Register* из контекстного меню.

Сдвиговый регистр выглядит как пара терминалов; 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 к следующей итерации.

Сдвиговый регистр передает любой тип данных и автоматически принимает тип первых же данных, переданных на него. Данные, передаваемые на терминалы сдвигового регистра, должны быть одного типа. Предусмотрена возможность создания нескольких сдвиговых регистров в одной структуре цикла. К тому же сдвиговый регистр может иметь несколько левых терминалов сдвигового регистра для возможности работы с несколькими значениями предыдущих итераций.

Сдвиговые регистры можно использовать для запоминания значений предыдущих итераций, что полезно при создании алгоритмов усреднения. Установка дополнительных терминалов сдвигового регистра для переноса значений на следующую итерацию осуществляется щелчком правой кнопкой мышки на левом терминале и выбором Add Element из контекстного меню. Например, если добавить два дополнительных терминала к левому терминалу сдвигового регистра, то значения последних трех итераций поступят на текущую итерацию.

Чтобы инициализировать сдвиговый регистр, необходимо передать на его левый терминал любое значение извне тела цикла. Если не инициализировать регистр, цикл использует значение, записанное в регистр во время последнего выполнения цикла или значение, используемое по умолчанию для данного типа данных, если цикл никогда не выполнялся. Например, если тип данных сдвигового регистра логический (*Boolean*), начальное значение *ЛОЖЬ* (*FALSE*). Точно так же, если тип данных сдвигового регистра числовой, то начальное значение - 0.

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.

Пример цикла со сдвиговым регистром

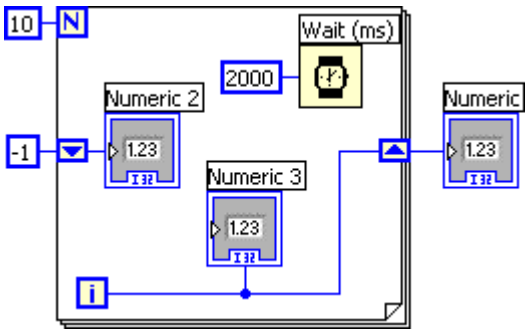


Рис.12 Пример цикла со сдвиговым регистром

На рисунке 12 приведен пример, демонстрирующий работу сдвигового регистра.

На индикаторе *Numeric3* с интервалом 2 секунды будут появляться числа от 0 до 9, на индикаторе *Numeric2* с тем же интервалом будут появляться числа от -1 до 8, соответствующие числам индикатора *Numeric3* на предыдущей итерации (-1 – начальное значение сдвигового регистра), на индикаторе *Numeric* числовое значение 9 появится после выполнения всего цикла.

Пример вычисления суммы от 1 до *N*

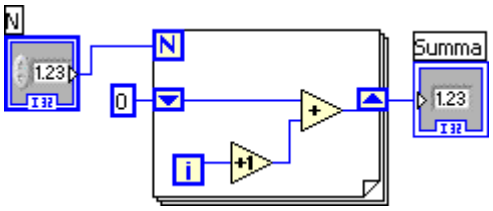


Рис.13 Пример вычисления суммы от 1 до *N*

На рисунке 13 приведен пример вычисления суммы от 1 до *N*.

1.4. Массивы

Массивы объединяют группу элементов одного типа данных. Массивы элементов могут иметь разную размерность. Элементами массива называют группу составляющих его объектов. Размерность массива это совокупность столбцов (длина) и строк (высота). Глубина – общее количество элементов в массиве. Массив может иметь одну и более размерностей, до 2^{31} элементов в каждом направлении, насколько позволяет оперативная память.

Данные, составляющие массив, могут быть любого типа: целочисленного, логического, строкового (integer, Boolean, string).

Использование массивов удобно при работе с группами данных одного типа и при накоплении данных после повторяющихся вычислений. Массивы идеально подходят для хранения данных, накопленных во время работы циклов, при этом одна итерация цикла создает один элемент массива.

Все элементы массива упорядочены. Каждому элементу массива присвоен индекс, что обеспечивает легкий к нему доступ. Индекс первого элемента массива всегда 0. Таким образом, индексы массива находятся в диапазоне от 0 до N-1, где N – число элементов в массиве. Например, для N=10, индекс находится в пределах от 0 до 9.

1.4.1. Создание массива элементов управления и отображения

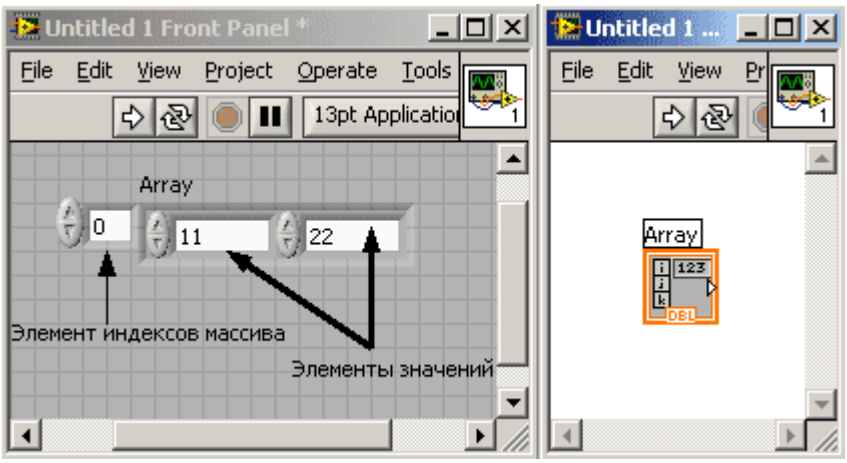


Рис.14 Массив числовых элементов управления

Для создания массива элементов управления или отображения, как показано на примере, необходимо выбрать его шаблон из палитры элементов (Controls) в разделе «Массивы, Матрицы и Кластеры» («*Modern*»→«*Array, Matrix & Cluster*»), поместить его на лицевую панель. Затем поместить в шаблон массива элемент управления либо отображения. Если

производится попытка поместить в шаблон массива такой элемент управления или отображения, как двукоординатный график осциллограмм (XY graph), то разместить такой элемент не удастся.

Поместить объект в шаблон массива следует до того, как он будет использоваться на блок-диаграмме. Иначе на блок-диаграмме шаблон массива не будет инициализирован.

1.4.2. Двумерные массивы

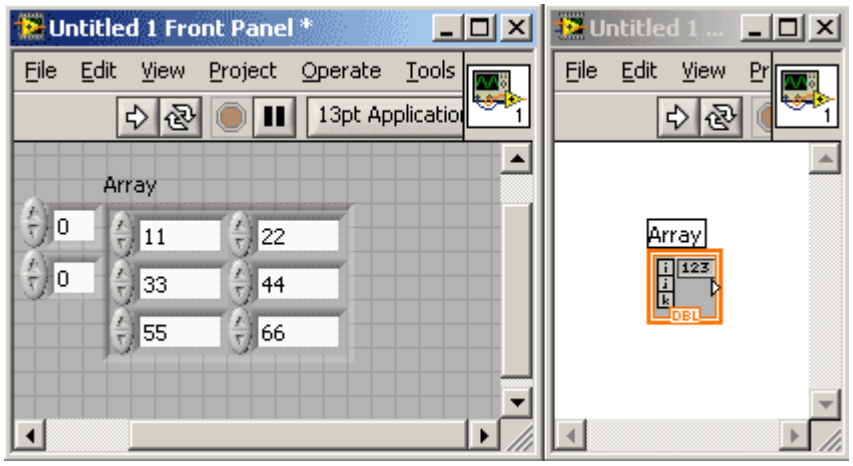


Рис.15 Двумерный массив числовых элементов управления

Двумерный (2*D*) массив хранит элементы в виде матрицы. Таким образом, для размещения элемента требуется указание индекса столбца и строки. На иллюстрации показан двумерный массив, состоящий из 2 столбцов (длина) и 3 строк (высота). Количество элементов в массиве (глубина) – 6 (2·3=6).

Для добавления массиву размерности необходимо щелкнуть правой кнопкой мыши на элементе индекса и выбрать из контекстного меню пункт Добавить Размерность (*Add Dimension*). Также для добавления размерности можно использовать инструмент *ПЕРЕМЕЩЕНИЕ* (<стрелка>), изменив размер элемента индекса.

1.4.3. Создание массива констант

На блок-диаграмме создать массив констант можно, выбрав в палитре функций в разделе Массивы (*Array*) - Шаблон Массива Констант (*Array Constant*) и поместив в него числовую константу или другой объект данных (*Boolean, string*).

1.4.4. Создание массива с помощью цикла

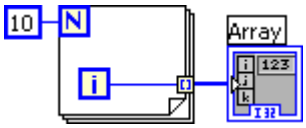


Рис.16 Пример создания массива с помощью цикла

Цикл с фиксированным числом итераций (*For*) и цикл по условию (*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. На рис.16 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.

Автоиндексация отключается щелчком правой кнопки мыши на терминале входа/выхода из цикла и выбором Отключить Автоиндексацию (*Disable Indexing*) из контекстного меню. Автоиндексация отключается в случае, когда на элемент отображения необходимо передать только последнее значение.

Примечание. Ввиду того, что цикл с заданным числом итераций (*For*) часто используется для создания циклов, *LabVIEW* осуществляет автоиндексацию автоматически. Автоиндексация для цикла с заданным числом итераций (*For*) включена по умолчанию. Для цикла по условию (*While*) по умолчанию автоиндексация отключена. Чтобы включить автоиндексацию, необходимо щелкнуть правой кнопкой мыши на терминале входа/выхода из цикла и выбрать в контекстном меню *Включить Автоиндексацию* (*Enable Indexing*).

1.4.5. Создание двумерных (2D) массивов с помощью цикла

Для создания двумерных (2D) массивов необходимо использовать два цикла с заданным числом итераций (For), один внутри другого.

1.4.6. Пример виртуального прибора, создающего двумерный массив 5x5, заполненный числами от 1 до 25

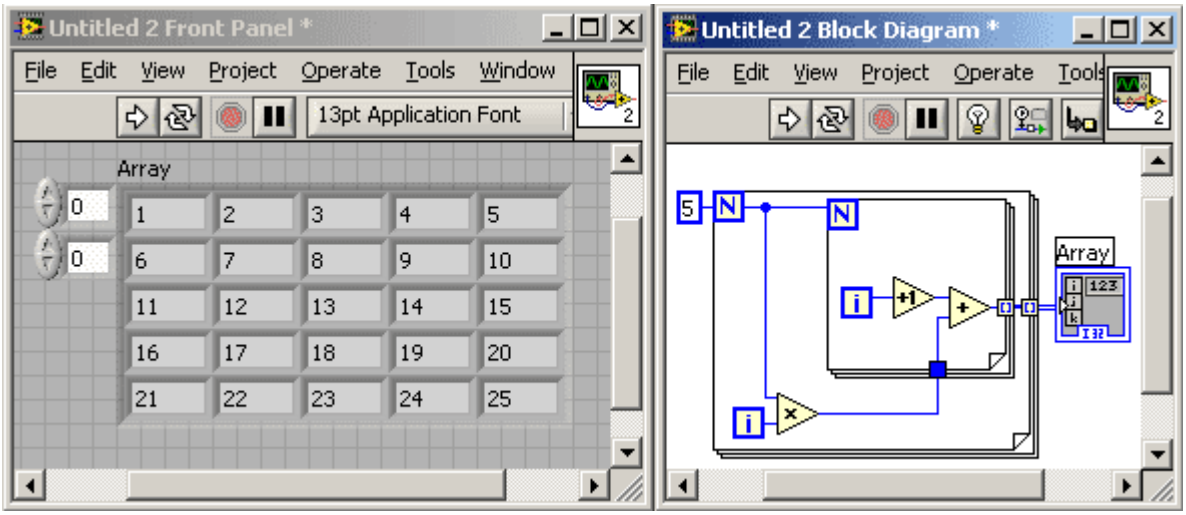


Рис.17 Программа создающая двумерный массив 5x5, заполненный числами от 1 до 25

С помощью циклов создадим двумерный массив 5x5 и заполним его числами от 1 до 25. Один из вариантов такой программы представлен на рис.17.

1.4.7. Пример вычисления чисел Фибоначчи

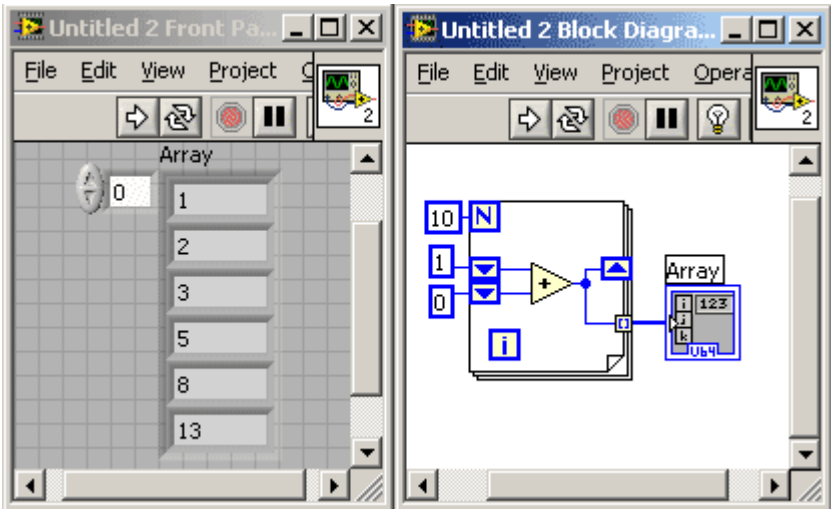


Рис.18 Пример вычисления суммы от 1 до *N*

Числа Фибоначчи – это числа, вычисляемые по следующему алгоритму: каждое новое число вычисляется как сумма двух предыдущих. Первые два числа обычно берут 0 и 1. На рисунке 18 приведен пример вычисления 10 первых чисел Фибоначчи.

1.5. Функции работы с массивами

Для создания и управления массивами используют функции, расположенные на палитре функций в разделе Массивы (*Programming→Array*). Функции обработки массивов включают в себя:

Размерность массива (Array Size) – показывает количество элементов массива в каждой размерности. Если массив *n*-мерный, на выходе функции Array Size будет массив из *n* элементов. Например, для приведенного ниже массива функция Array Size выдаст значение 3.

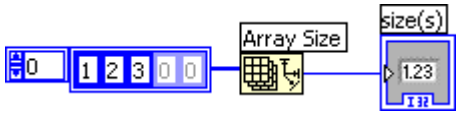


Рис.19 Пример использования функции *Array Size*

Инициализация массива (Initialize Array) – создает *n*-мерный массив, в котором каждый элемент инициализирован значением поля ввода данных «*element*». Для увеличения размерности массива достаточно добавить поля ввода данных, растянув функцию. Например, функция *Инициализация массива* (*Initialize Array*) с заданными полями ввода данных – в поле «*элемент*» («*element*») значение 4, в поле *размерность* (*dimension size*) значение 3 и при наличии одного поля ввода данных *размерность* (*dimension size*) - выдаст массив, показанный на рис.20.

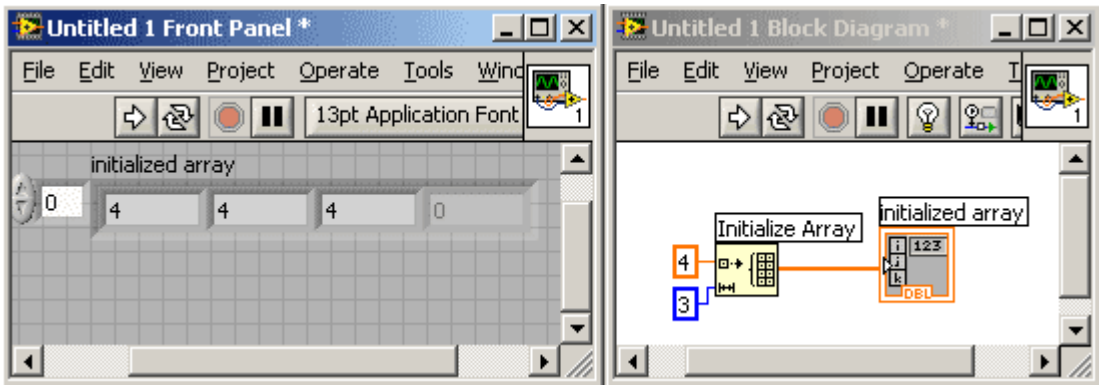


Рис.20 Пример использования функции *Initialize Array*

Компоновка массива (Build Array) – объединяет несколько массивов или добавляет элемент в *n*-мерный массив. Изменение размера функции увеличивает количество полей ввода данных, что позволяет увеличить количество добавляемых элементов. Например, если

объединить два предыдущих массива, то функция *Компоновка массива* (*Build Array*) выдаст на выходе следующий массив.

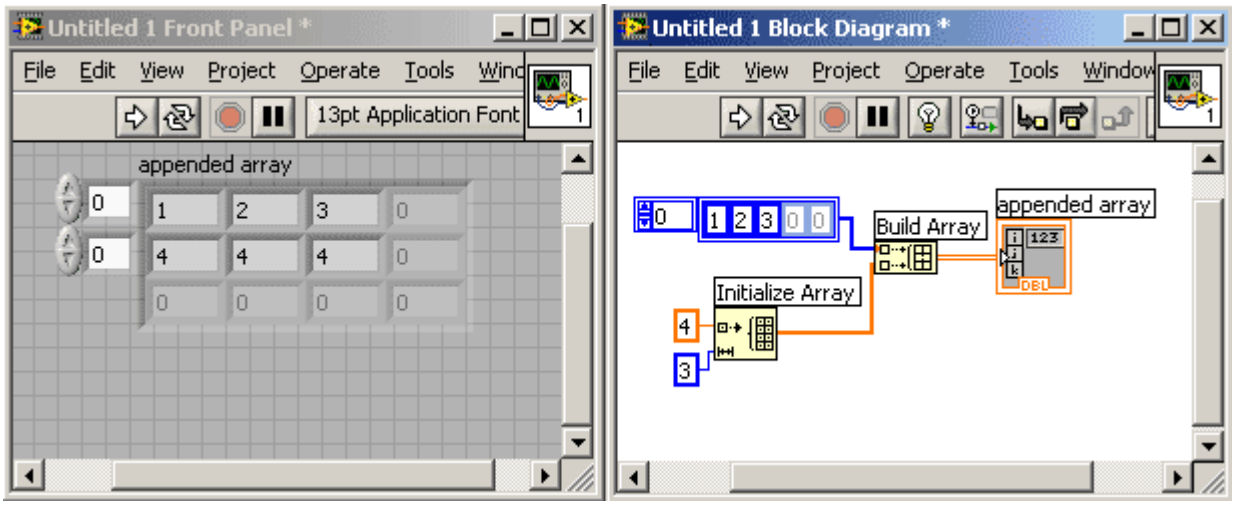


Рис.21 Пример использования функции *Build Array*

Для объединения входных данных в более длинный массив той же размерности, как показано ниже, достаточно щелкнуть правой кнопкой мыши на функции и выбрать из контекстного меню пункт *Объединить вводы* (*Concatenate Inputs*).

Подмножество массива (Array Subset) – выдает часть массива, начиная с индекса, введенного в поле *index*, и длиной, указанной в поле *length*.

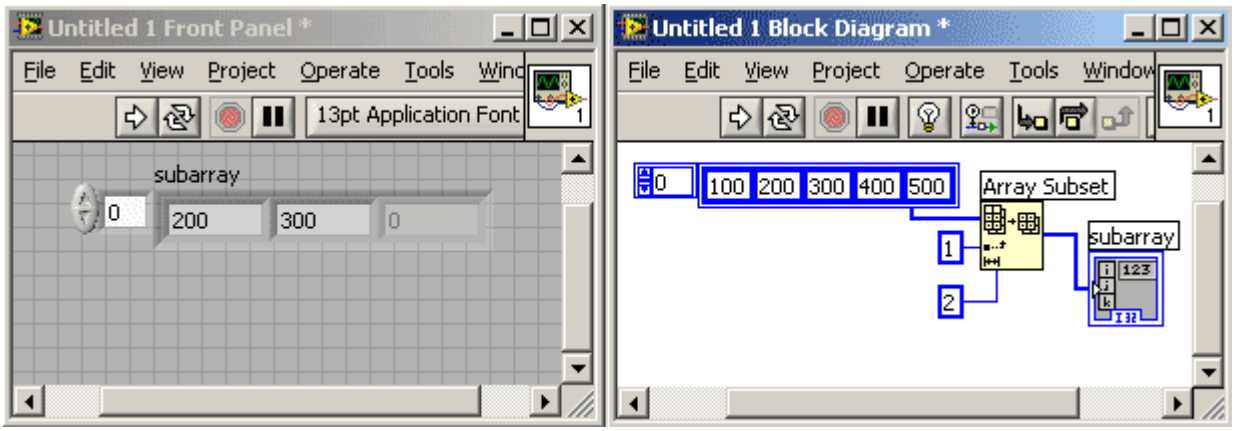


Рис.22 Пример использования функции *Array Subset*

Элемент массива по индексу (Index Array) – выдает элемент, соответствующий индексу, номер которого введен в поле *index*. Например, при использовании предыдущего массива, функция *Index Array* выдаст 200, если в поле ввода данных *index* ввести 1.

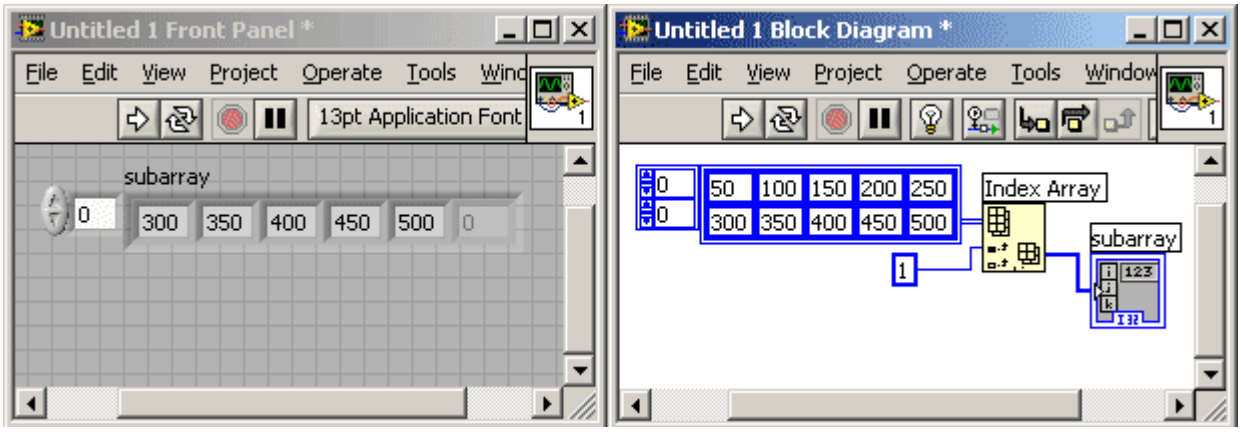


Рис.23 Пример использования функции *Index Array*

Функцию *Index Array* можно использовать для выделения строки или столбца из двумерного массива и дальнейшего отображения в виде подмассива. Для этого двумерный массив надо подать в поле ввода данных функции. Функция *Index Array* должна иметь два поля *index*. Верхнее поле *index* указывает строку, а второе поле *index* указывает столбец. Можно задействовать оба поля *index* для выбора отдельного элемента или только одно, для выбора строки или столбца. Например, в поле ввода данных функции подается массив, показанный на рис.23.

Поиск в одномерном массиве (*Search 1D Array*). Данная функция выполняет поиск заданного элемента в одномерном массиве начиная с заданного индекса. Входными значениями данной функции являются: верхний вход (*1D array*) – массив, в котором производится поиск, средний вход (*element*) – значение которое необходимо найти, нижний вход (*start index*) – с какого индекса массива начинать поиск. Функция возвращает (*index of element*) индекс элемента массива значение которого совпадает со значением на входе "*element*". Если элемент не найден, то возвращается значение -1.

Пример виртуального прибора, использующего данную функцию, приведен на рис.24.

Данный виртуальный прибор создает массив из 1000 целых случайных чисел в диапазоне от 0 до 100, в котором ищется первое вхождение числа 55. Функция *Генератор случайных чисел от 0 до 1* (*Random Number* (*0-1*)) расположена в палитре функций в разделе *Programming→Numeric* и функция преобразования в 8-разрядное целое беззнаковое число (*To Unsigned Byte Integer*) расположена в палитре функций в разделе *Programming→Numeric→Conversion.*

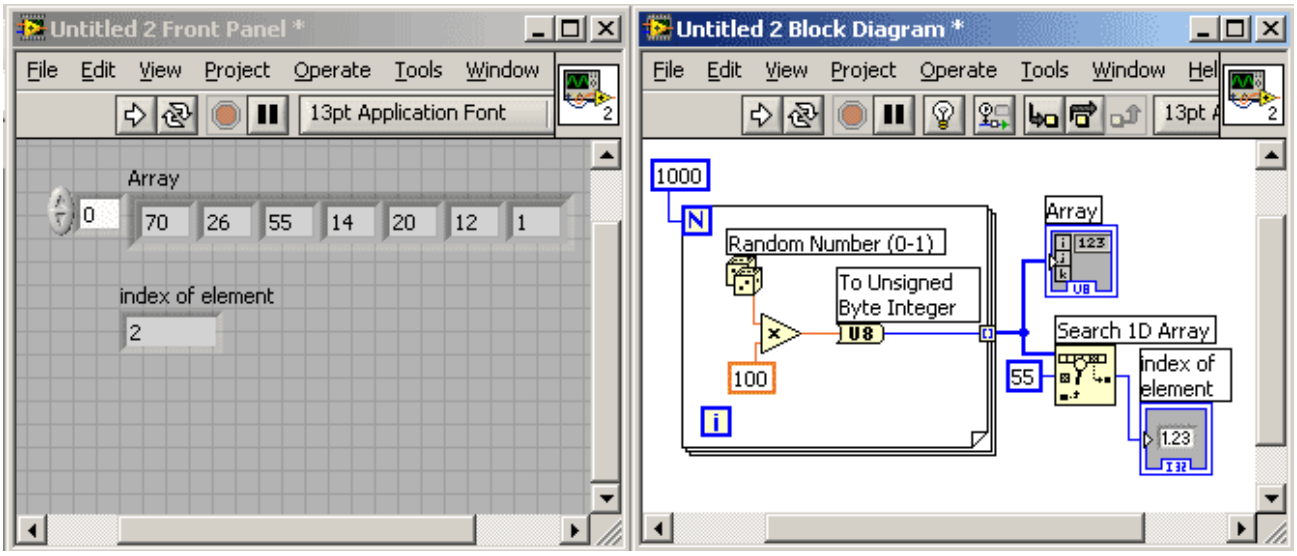


Рис.24 Пример использования функции *Search 1D Array*

Пример виртуального прибора в котором реализован поиск всех индексов массива значение которых совпадает с заданным значеним

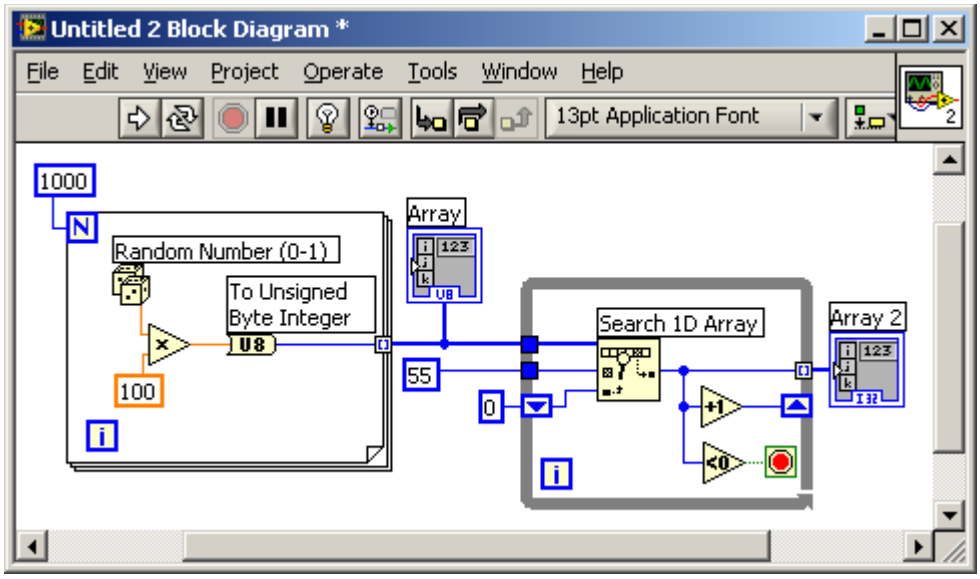


Рис.25 Пример использования функции *Search 1D Array*

В отличии от предыдущего примера (рис.24) в данном виртуальном приборе поиск происходит внутри цикла *While*. После выполнения работы виртуального прибора (рис.25) в массиве индикаторов *Array 2* будут значения индексов массива индикаторов *Array* в которых значения равны числу 55.

1.6. Передача массива данных в цикл

Если протянуть проводник от массива, расположенного на блоке диаграмм вне цикла до границы цикла, то на границе цикла образуется тоннель (рис.26). Тоннель может быть ин

дексированным (рис.26) и неиндексированным (рис.27).

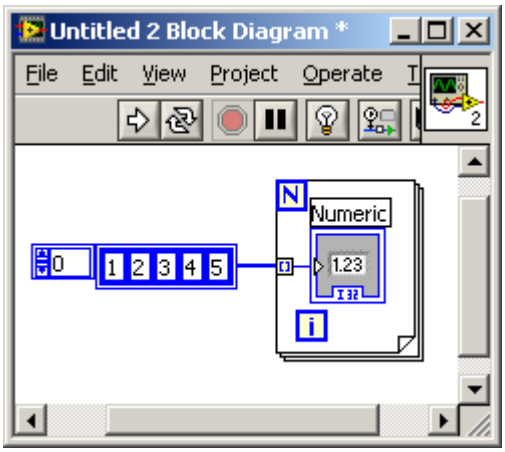


Рис.26 Пример передачи массива данных в цикл *for* через индексированный тоннель

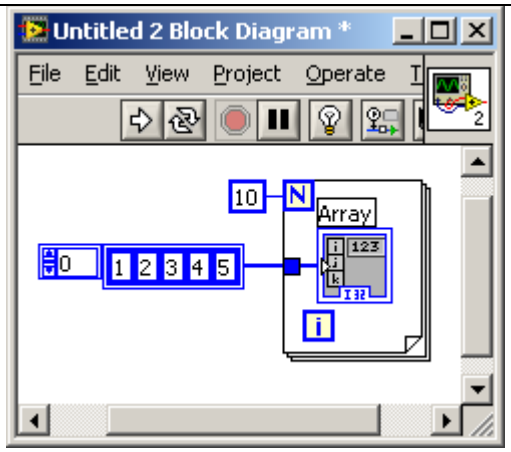


Рис.27 Пример передачи массива данных в цикл *for* через неиндексированный тоннель

В случае неиндексированного тоннеля массив передается в цикл целиком, в случае индексированного – поэлементно (на каждой итерации цикла из массива берется одно значение с индексом, совпадающим со значением переменной цикла *i*). Тоннель можно сделать индексированным и неиндексированным. Для этого необходимо подвести к нему курсор мышки и однократно нажать правую клавишу мышки. Во всплывающем меню выбрать пункт *Enable Indexing* (*Включить индексацию*) или *Disable Indexing* (*Выключить индексацию*). Заметим, что если используется индексированный тоннель, то устанавливать значение *N* необязательно. Количество итераций, которое будет выполнять цикл, будет равно

размерности массива. На рис.26 цикл выполнится 5 раз. Если используется несколько индексированных тоннелей и задается значение *N*, то количество итераций цикла будет соответствовать наименьшему массиву или значению *N*, если оно меньше размерности массивов переданных в цикл через индексированные тоннели.

Примеры использования индексированного тоннеля

Данный виртуальный прибор создает массив из 1000 целых случайных чисел в диапазоне от 0 до 100, в котором ищется первое вхождение числа 55. Функция *Генератор случайных чисел от 0 до 1* (*Random Number* (*0-1*)) расположена в палитре функций в разделе *Programming→Numeric* и функция преобразования в 8-разрядное целое беззнаковое число (*To Unsigned Byte Integer*) расположена в палитре функций в разделе *Programming→Numeric→Conversion*. В отличии от виртуального прибора изображенного на рис.24, в данном приборе не используется функция *Search 1D Array.*

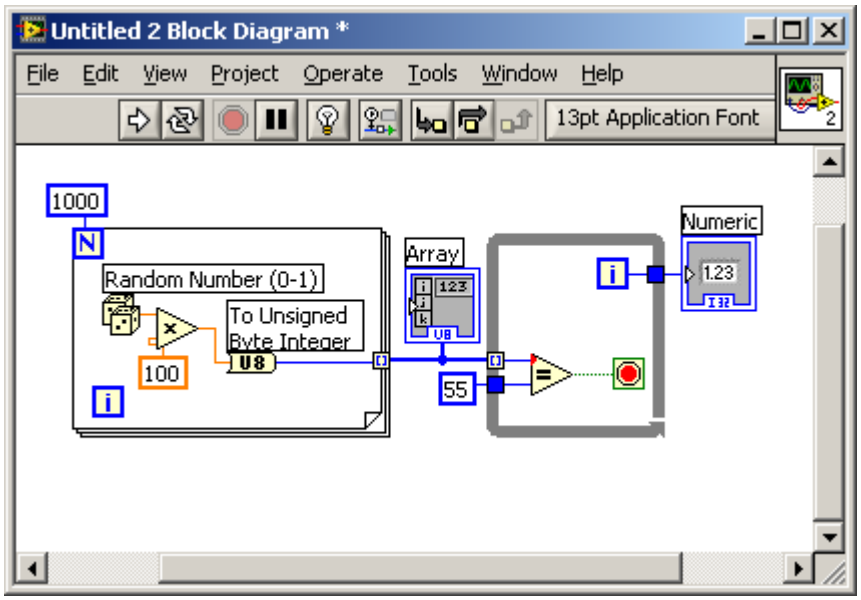


Рис.28 Пример использования входного индексированного тоннеля

Во втором примере (рис.29) реализован алгоритм поиска элемента в массиве с максимальным значением. В первом цикле создается массив из 1000 числовых элементов со значениями расположенными в диапазоне от 0 до 100. Во втором цикле реализован алгоритм поиска.

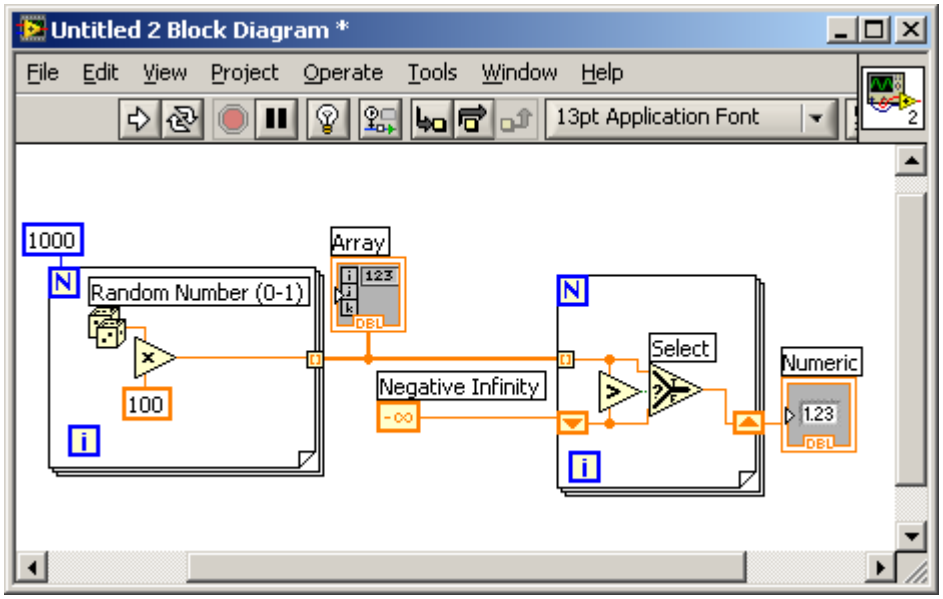


Рис.29 Пример использования входного индексированного тоннеля для нахождения максимального элемента в массиве

Числовая константа *Negative Infinity*, соответствующая минимальному числовому значению, расположена в палитре функций в разделе *Programming→Numeric*. Функция *Select* возвращает значение, соответствующее верхнему входу (*t*), если на средний вход (*s*) подано значение *TRUE* или соответствующее нижнему входу (*f*) , если на средний вход (*s*) подано значение *FALSE*. Функция *Select* расположена в палитре функций в разделе *Programming→Comparison.*

1.7. Полиморфизм

Арифметические функции, расположенные на палитре функций в разделе *Арифметические функции* (*Numeric*) полиморфны. Это означает, что поля ввода данных этих функций могут различаться по структуре данных (скалярные величины, массивы). Например, можно использовать функцию *Сложение* (*Add*) для прибавления скалярной величины к массиву или сложения двух массивов. Если в одно поле ввода данных функции *Сложение* (*Add*) подать скалярную величину 2, а другое соединить с массивом, то функция прибавит 2 к каждому элементу массива.

Если на вход функции *Сложение* (*Add*) подать два предыдущих массива, функция сложит каждый элемент первого массива с соответствующим элементом второго и выдаст результат в виде массива.

Если с помощью функции *Сложение* (*Add*) сложить два массива разной размерности, то функция сложит каждый элемент первого массива с соответствующим элементом второго и выдаст результат в виде массива размерностью меньшей из двух.

1.8. Использование графиков для отображения данных

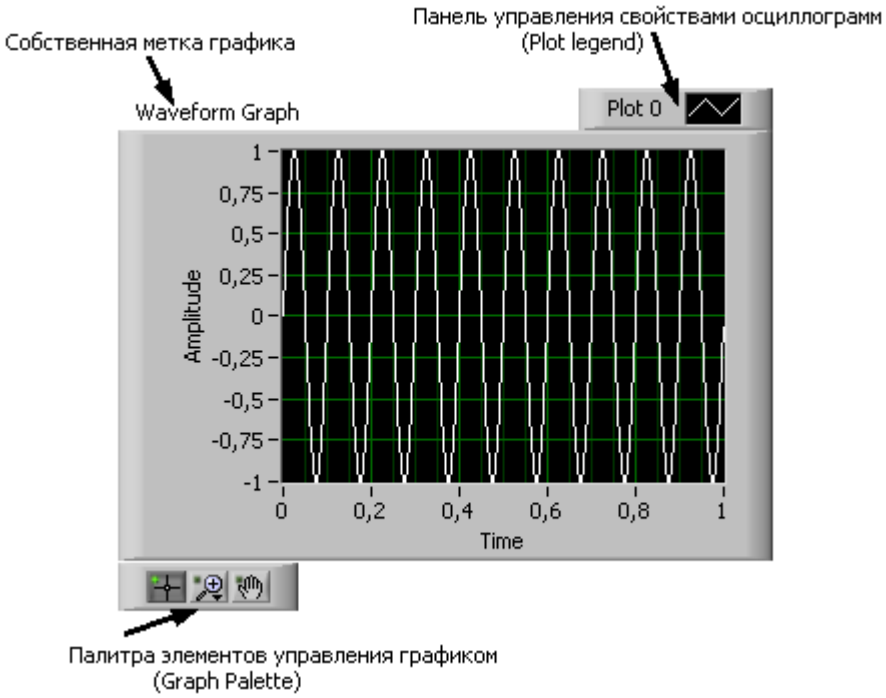


Рис.30 Графический элемент отображения Waveform Graph

С помощью графиков виртуальный прибор обычно отображает накопленные в массив данные в виде осциллограмм. На рис.21 представлен один из элементов графики: график осциллограмм (*Waveform Graph*).

График осциллограмм расположен на палитре элементов управления и отображения в разделе *Графики* (*Modern→Graph*). График осциллограмм (*Waveform Graph*) отображает только однозначные функции, такие как $y=f(x)$, с точками, равномерно распределенными по оси *X*, Двукоординатный график осциллограмм (*XY Graph*) отображает любой набор точек, будь то равномерно распределенная выборка или нет.

Для отображения множества осциллограмм необходимо изменить размер панели управления свойствами осциллограмм (*Plot legend*). График множества осциллограмм используется с целью экономии пространства на лицевой панели и для сравнения осциллограмм данных между собой. График осциллограмм (*Waveform Graph*) и

Двукоординатный график осциллограмм (*X-Y Graph*) автоматически поддерживают режим отображения множества осциллограмм.;

Одиночный график осциллограмм работает с одномерными массивами и представляет данные массива в виде точек на графике, с приращением по оси *X* равным 1 и началом в *X*=0.

График множества осциллограмм работает с двумерными массивами данных, где каждая строка массива есть одиночная осциллограмма данных и представляет данные массива в виде точек на графике, с приращением по оси *X* равным 1 и началом в *X*=0.

Для представления каждого столбца двумерного массива данных в виде осциллограммы на графике необходимо соединить терминал данных массива с терминалом приема данных графика, затем щелкнуть правой кнопкой мыши в поле графика и выбрать из контекстного меню пункт *Транспонирование Массива* (*Transpose Array*).

Двукоординатные графики универсальны, они предназначены для отображения многозначных функций в декартовой системе координат (замкнутые кривые, распределение осциллограммы во времени с переменной временной базой).

Одиночный Двукоординатный график осциллограмм (XY graph) работает с группами данных, содержащими массивы x и y. Двукоординатный график осциллограмм (*XY graph*) также воспринимает массивы точек, где точки – группы данных, содержащие значения по шкалам x и y.

Двукоординатные графики множества осциллограмм работают с массивами осциллограмм, в которых осциллограмма данных – кластер, содержащий массивы значений x и y. Двукоординатные графики множества осциллограмм воспринимают также массивы множества осциллограмм, где каждая осциллограмма представляет собой массив точек. Каждая точка это группа данных, содержащая значения по x и y.

1.8.1. Использование палитры элементов управления графиком

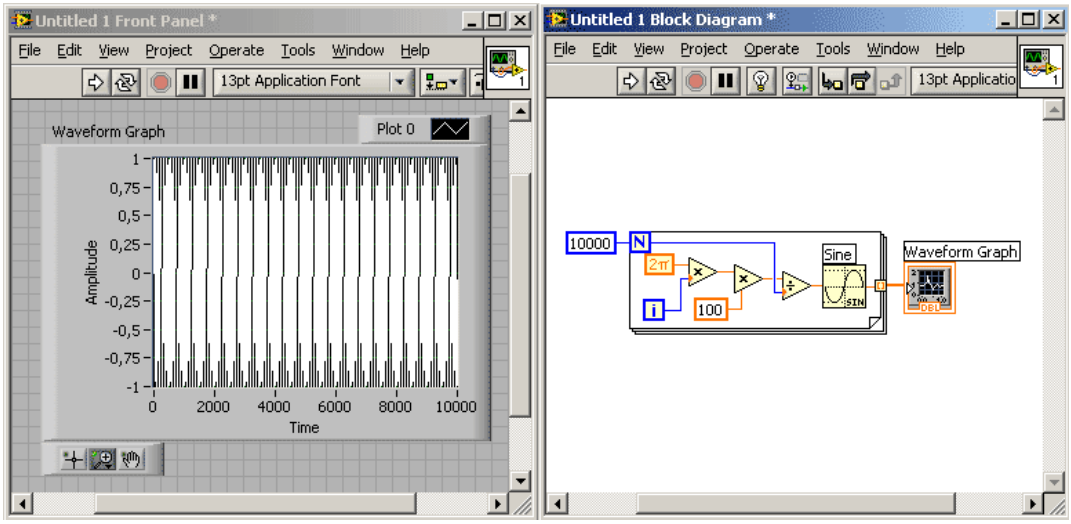


Рис.31 Программа отображающая на графике 100 периодов синусоиды

Создайте виртуальный прибор изображенный на рис.25. Графический элемент *Waveform Graph* можно найти в разделе *Modern→Graph*, функцию *Sine*, вычисляющую синус числового значения – *Mathematics→Elementary*.

Установите палитру элементов управления графиком. Для этого во всплывающем меню на графическом индикаторе *Waveform Graph* установите галку в пункте *Visible Items→Graph Palette*.

Выберите способ выделения участка графика (рис.26).

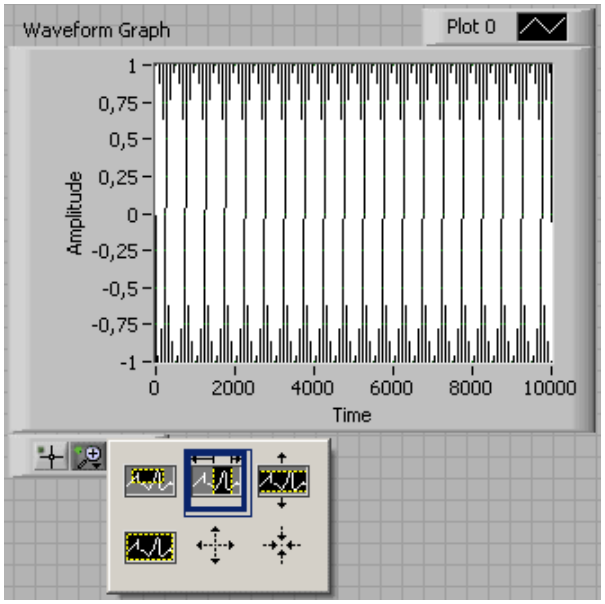


Рис.32 Выбор способа выделения участка графика6

Выделите небольшой участок графика (рис.27).

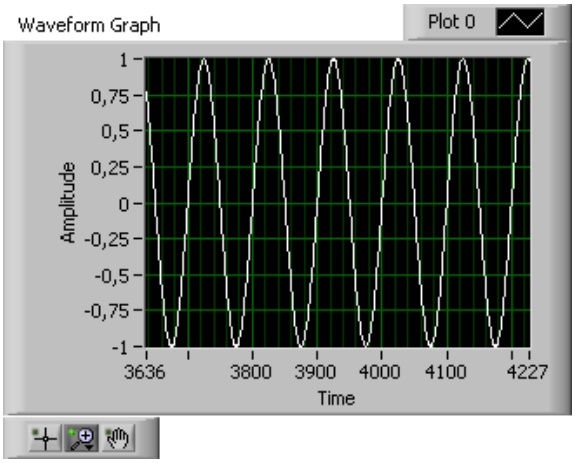


Рис.33 Выделенный участок графика

1.9. Структура Варианта

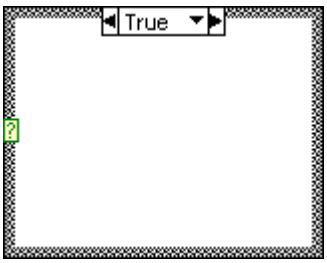


Рис.34 Выделенный участок графика

Структура Варианта (рис.28) имеет две или более поддиаграммы вариантов. Только одна поддиаграмма варианта видима в данный момент времени и только одна поддиаграмма варианта работает при выполнении данной Структуры. Входное значение терминала селектора структуры Варианта определяет, какая поддиаграмма будет выполняться в данный момент времени. Структура Варианта аналогична операторам варианта (switch...case) или логическим операторам (if... else) в текстово-ориентированном языке программирования Си.

Селектор структуры Варианта, расположенный сверху графического изображения Структуры включает в себя указатели: значения варианта в центре и стрелки уменьшения или увеличения по сторонам. Эти стрелки используются для просмотра возможных вариантов.

Входное значение терминала селектора Варианта определяет, какая поддиаграмма структуры или вариант будет выполняться. Допустимо использовать целочисленный (*integer*), логический (*Boolean*), строковый (*string*) типы, а также тип перечисления в качестве

значения терминала *Варианта*, Терминал *Варианта* может располагаться в любом месте левой границы Структуры Варианта. Если терминал *Варианта* логического типа, то структура состоит из двух логических вариантов *ИСТИНА* (*TRUE*) и *ЛОЖЬ* (*FALSE*). Если терминал *Варианта* одного из типов: целочисленный, строковый или перечисления, то количество вариантов может достигать 2^{31}-1 вариантов.

Для использования *Структуры Варианта* необходимо отметить вариант по умолчанию (*default case*). Вариант по умолчанию или поддиаграмма по умолчанию выполняется, если значение терминала *Варианта* выходит за пределы диапазонов или не существуют варианты для возможных значений терминала *Варианта*.

Правый щелчок мыши на границе *Структуры Варианта* позволяет добавлять, дублировать, перемещать и удалять варианты (поддиаграммы), а также отмечать вариант по умолчанию.

Структура Варианта допускает использование терминалов входных и выходных данных. Терминалы входных данных доступны во всех поддиаграммах, но их использование поддиаграммой структуры необязательно. Создание выходного терминала на одной поддиаграмме приводит к его появлению на других поддиаграммах в том же самом месте границы структуры. Если хотя бы в одной поддиаграмме выходной терминал не определен, поле этого терминала окрашивается в белый цвет, что характеризует ошибку определения структуры. Необходимо определять значения выходных терминалов во всех вариантах (поддиаграммах). Кроме того, выходной терминал должен иметь значение одного и того же типа.

Для определения значения выходного терминала следует правым щелчком мыши по терминалу вызвать всплывающее меню и выбрать пункты: *Create→Constant* или *Create→Control*.

1.9.1. Примеры

Следующие примеры показывают, как значения входных терминалов Структуры Варианта складываются или вычитаются в зависимости от значения терминала варианта.

Логическая Структура Варианта

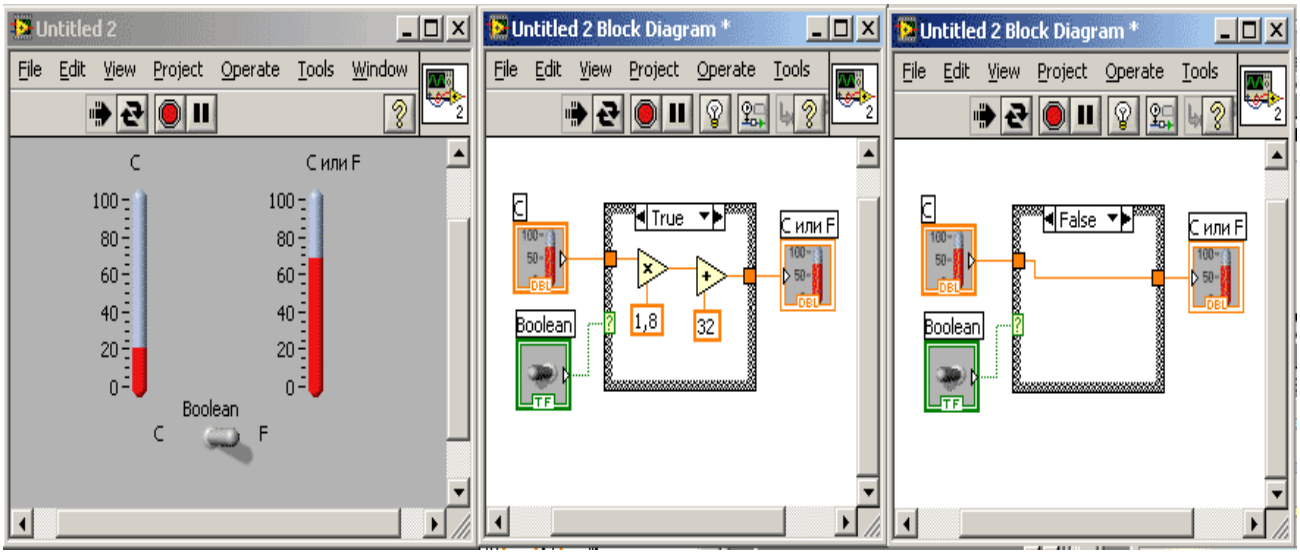


Рис.35 Пример логической Структуры Варианта.

На рисунке 29 пример логической Структуры Варианта.

Если терминал логического элемента управления (Boolean), соединенного проводником с терминалом – *ИСТИНА* (*TRUE*), выполняется преобразование температуры, заданной в градусах Цельсия в температуру по Фаренгейту (блок-диаграмма в центре (рис.28)), если *ЛОЖЬ* (*FALSE*), - выполняется – преобразование не выполняется (рисунок справа).

Логическая Структура Варианта

Изменим предыдущий пример. Логический элемент управления *Boolean* заменим на целочисленный. Для этого, на лицевой панели наведем на него курсор мышки и выполним однократное нажатие правой клавиши мышки. Во всплывающем меню выберем пункт *Replace→Modern→Knob*. Снова наведем на него курсор мышки и снова выполним однократное нажатие правой клавиши мышки. Во всплывающем меню выберем пункт *Representation→Byte* (целые числа от 0 до 255). При этом на блоке диаграмм терминал изменит окраску с оранжевой (соответствующей вещественным числам) на синюю (соответствующую целым числам).

При такой замене на селекторе структуры варианта значение *ИСТИНА* (*TRUE*) изменится на числовое значение 1, значение ЛОЖЬ (*FALSE*) – на значение 0, которое будет значением по умолчанию.

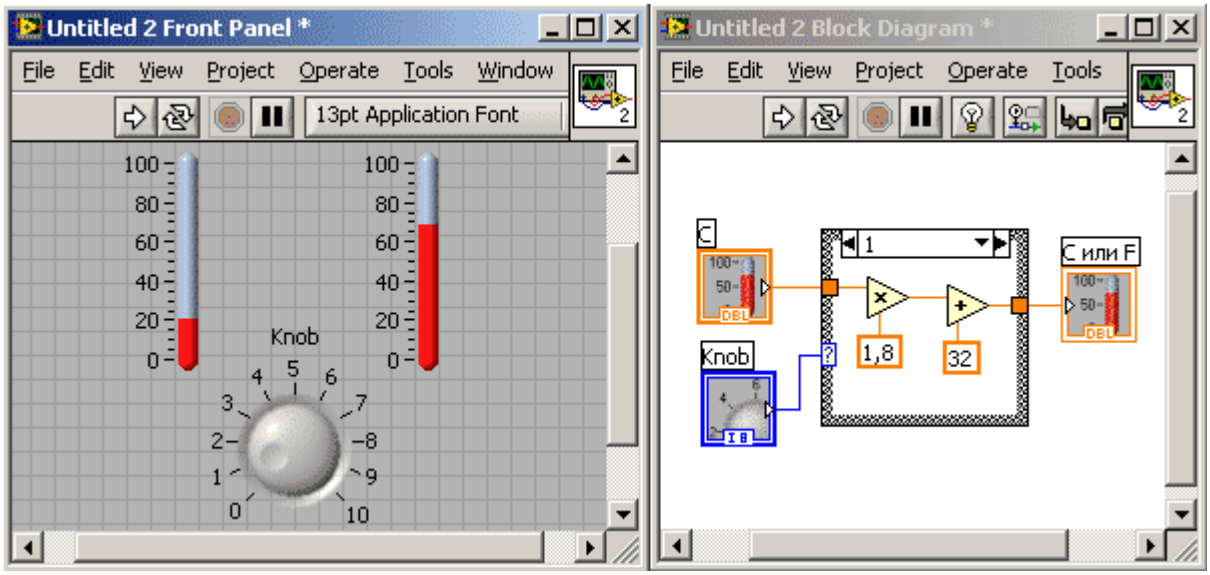


Рис.36 Пример целочисленной Структуры Варианта.

Данный виртуальный прибор будет выполнять преобразование температуры, заданной в градусах Цельсия в температуру по Фаренгейту только, если на элементе управления *Knob* установлено значение 1, во всех остальных случаях преобразование производится не будет.

Чтобы добавить в *Структуру варианта* еще вариант необходимо во всплывающем меню на селекторе *Структуры Варианта* выбрать пункт *Add Case After* или *Add Case Before*. Например, чтобы при установленном на элементе управления *Knob* значении 5 на индикаторе отображалось значение 5. Для этого новый вариант в структуре можно сделать как на рис.31.

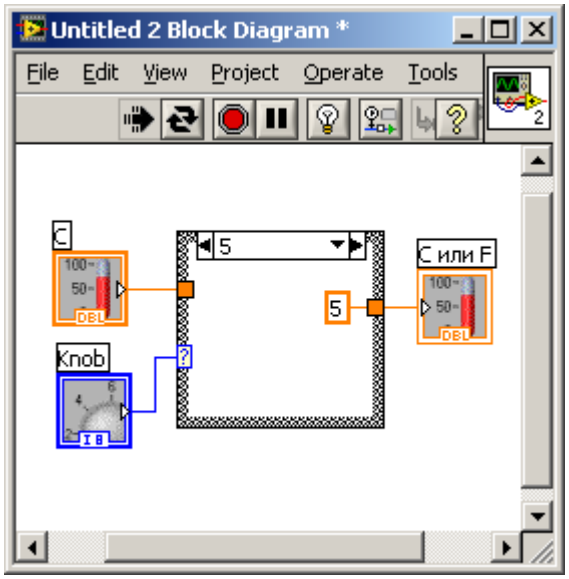


Рис.37 Модификация виртуального прибора изображенного на рис.30

1.9.2. Определение Вариантов

Определение *Варианта* осуществляется либо выбором значения на селекторе *Структуры Варианта* либо вводом значения с помощью инструмента ВВОД ТЕКСТА (<A>). Это значение селектора *Варианта* используется в дальнейшем при выборе *Варианта*. При определении *Варианта* могут использоваться: единственное значение, списки и диапазон значений. Списки значений представляют собой введенные значения через запятую, к примеру, - 1, 4, 6. Диапазон значений определяется как 10..20, т.е. вариант определен значениями в диапазоне от 10 до 20 включительно. При определении варианта также могут использоваться открытые диапазоны. К примеру, диапазон ..100 присваивает Варианту все значения меньше или равные 100. Также можно при определении Варианта использовать комбинацию списков и диапазонов. Например, ..5, 6, 7..10, 12, 13, 14. Если Вы вводите значения селектора, включающие пересекающиеся диапазоны, структура Варианта преобразует запись значения в более компактную форму. Предыдущий пример будет преобразован к ..10,12..14.

Значения селектора Варианта должно быть того же типа, что и тип данных объекта, соединенного с терминалом селектора Варианта. Значение селектора Варианта, окрашенное красным цветом, показывает, что необходимо удалить или отредактировать это значение перед запуском структуры, иначе виртуальный прибор не будет выполняться. Нельзя использовать вещественные числа в качестве значения селектора Варианта, так как возможны ошибки округления и возникновения ситуации неопределенности. Если Вы соединяете вещественный тип с терминалом селектора Варианта, LabVIEW округляет это значение до ближайшего четного целого. Если вещественное значение введено непосредственно в селектор Варианта, то оно окрашивается в красный цвет и должно быть удалено или отредактировано.

2. ГЕНЕРАЦИЯ, АНАЛИЗ И ОБРАБОТКА СИГНАЛОВ

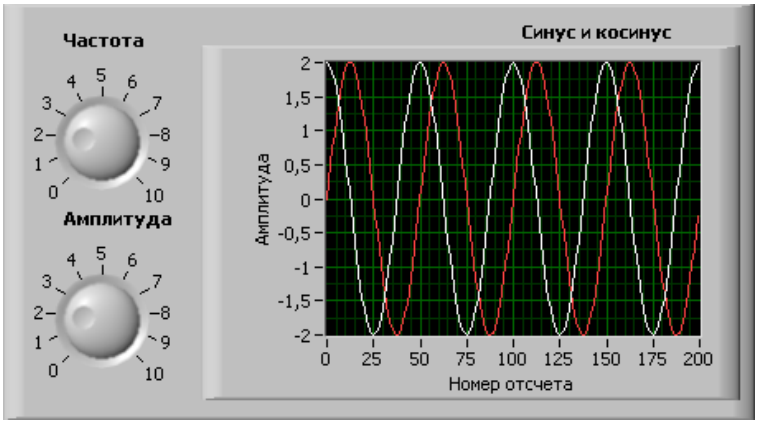
2.1. Генерация сигналов

В среде LabVIEW различают цифровые и «аналоговые» сигналы. Цифровые могут принимать только два значения 0 или 1. Такие сигналы характеризуют состояние входой или выходной цифровой линии, или нескольких линий (регистра). К цифровым сигналам относятся также последовательности импульсов. Цифровым сигналам соответствуют логические (*Boolean*) источники/приемники данных. «Аналоговые» сигналы принимают множество значений, допускаемых их числовым представлением (*Numeric*), и могут являться функциями времени, частоты, пространственных координат и т.д. Ниже мы будем иметь дело только с «аналоговыми» сигналами. Кавычки в данном случае означают, что реальный аналоговый сигнал представлен в доступной компьютеру цифровой форме, т.е. – конечным числом отсчетов, проквантованных по уровню. Без привязки ко времени (или другой независимой переменнной) такой сигнал есть просто числовой массив. В литературе по LabVIEW для таких сигналов используется также термин «*Pattern*», что переводится как *последовательность (отсчетов сигнала)*. Оцифрованный аналоговый сигнал может быть получен с АЦП во время измерений и также является массивом. Несколько сигналов, поступивших через АЦП с однотипных датчиков (каналов), образуют 2-мерный массив. Следовательно, можно говорить о 2-мерном сигнале. В LabVIEW существует специальный формат (кластер) данных – *Waveform* (неудачный «перевод» – осциллограмма), в котором сохраняется информация об абсолютной временной привязке сигнала: время первого отсчета и шаг между последующими. ВП из библиотеки *DAQmx* допускают чтение и запись как массивов, так и осциллограмм. В последнем случае заботиться об указании частоты подачи данных на ЦАП нет необходимости: если это допускается его техническими характеристиками, на выходе получится аналоговый сигнал с правильным временным масштабом. Таким образом, «сгенерировать сигнал» в LabVIEW означает «создать массив». Его временное масштабирование можно выполнить позднее, или вообще обойтись без этого. Следует учесть еще одну тонкость: по умолчанию, на выходе ВП *DAQmx Read* получаются действительные числа с плавающей точкой. Диапазон их изменения и значения определяются максимальным и минимальным значениями, выбранными в ВП *DAQmx Create Channel.vi* и разрядностью АЦП (типичное значение – 16, что дает диапазон целых чисел со знаком от -2^{15}

до $+2^{15}$). ВП *DAQmx Read* может передавать на выход и целые числа без всяких преобразований.

Для генерации сигналов - массивов (или последовательностей) в LabVIEW имеется широкий выбор возможностей. Во-первых, можно применить встроенные функции из палитры *Numeric***,** вызывая их циклически. Наличие генератора случайных чисел уже на этом этапе позволяет добавлять к ним шум. Используя узлы *Expression Node* и *Formel Node*, можно определить каждый отсчет сигнала в виде формулы как функцию его номера. Во-вторых, с помощью набора функций *Signal Generation* из палитры *Signal Processing* можно создавать ряд стандартных сигналов. Эти функции генерируют последовательности отсчетов. В-третьих, с помощью библиотеки ВП *Waveform Generation* из палитры *Signal Processing* сигналы можно сразу создавать масштабированными во времени. Отметим, что данные ВП написаны с использованием функций *Signal Generation* в качестве ВПП.

2.1.1. Генерация с помощью функций из палитры *Numeric* и цикла *For Loop*



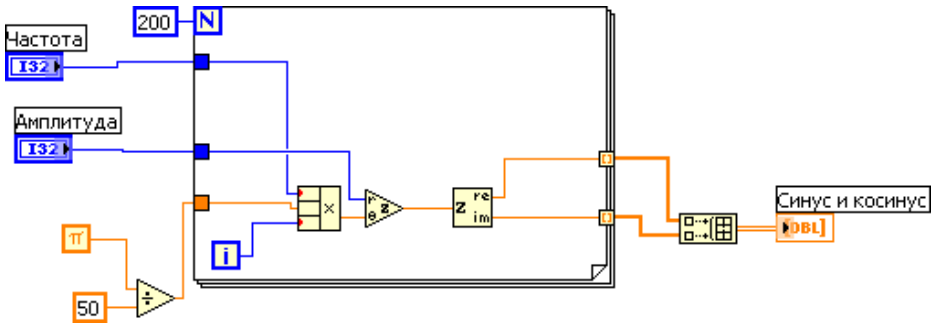


Рис.38 Создание синусоиды и косинусоиды с помощью формулы Эйлера

Обратите внимание на использование компаундной арифметики для задания фазы комлексной экспоненты.

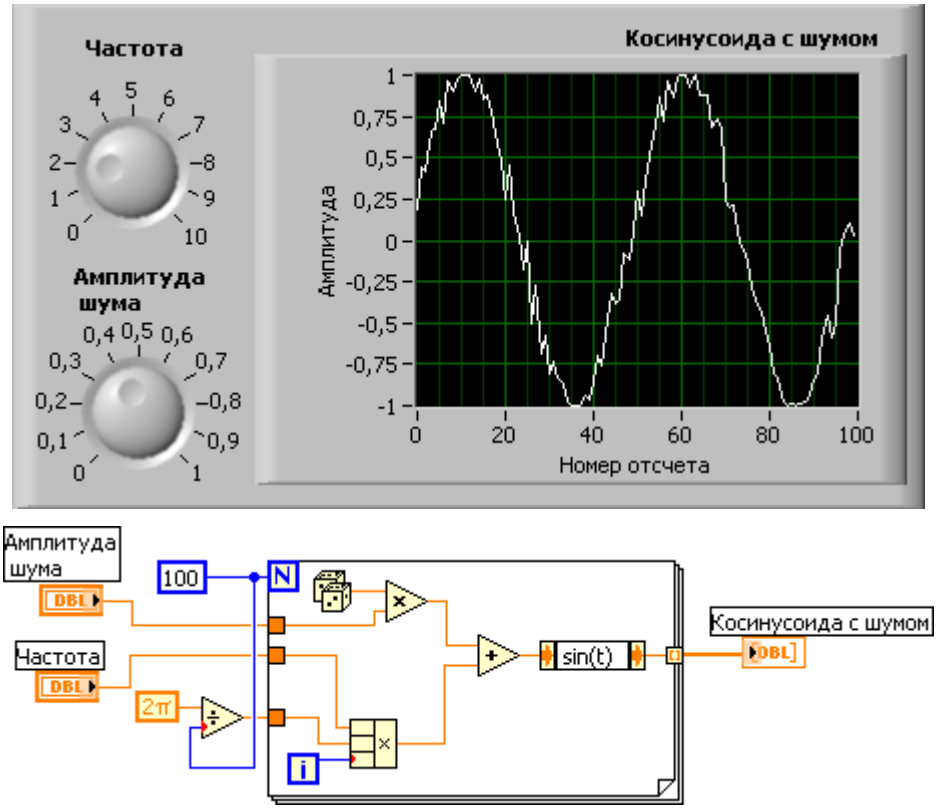


Рис.39 Создание «зашумленной» косинусоиды с помощью узла *Expression Node* и генератора случайных чисел.

2.1.2. Генерация с использованием специальных ВП из *Signal Processing→Waveform Generation*

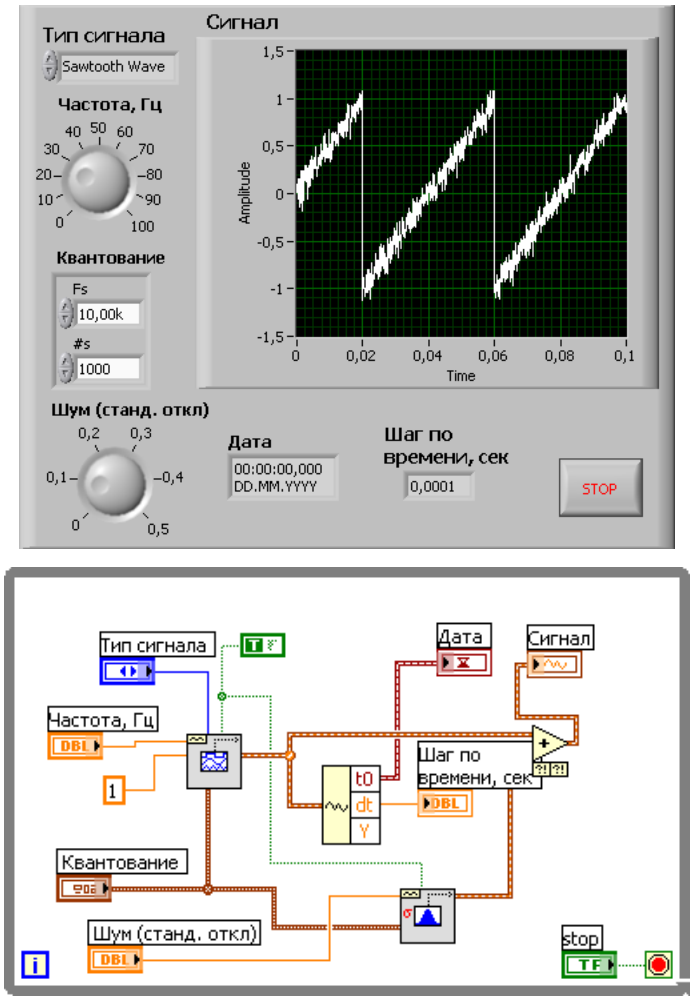


Рис.40. Генерация осциллограмм различных стандартных сигналов с добавлением шума с помощью ВП ***Basic Function Generator.vi*** и ***Gaussian White Noise Waveform.vi*** (*Wave_form_gen.vi*). Параметры квантования на оба ВП поступают с одного управляющего элемента «Квантование». Для суммирования сигнала и шума используется полиморфная функция сложения. Если параметры квантования будут отличаться, при суммировании возникнет ошибка. Доступ к временным параметрам сгенерированной осциллограммы дает функция ***Get Waveform Components*** с палитры **Waveform.** Индикатор «Дата» пуст. Чтобы записать в него системное время компьютера, можно воспользоваться функцией ***Get Date→Time in Second*** с палитры ***Time&Dialog*** и ***Build Waveform*** с палитры ***Waveform***, включив ее в разрыв выходного проводника ВП – генератора стандартных функций.

2.2. Анализ сигналов в частотной области с помощью дискретного Фурье-преобразования (ДПФ) и быстрого Фурье-преобразования (БПФ)

Алгоритм, использующийся для перевода выборок данных из временной области в частотную, известен как *Дискретное Преобразование Фурье* или ДПФ. Он широко используется в областях спектрального анализа, прикладной механики, акустики, воспроизведении изображений в медицине, численного анализа, приборостроения и телекоммуникаций.

Если сигнал оцифровывается с частотой выборки (квантования) f_s Гц , то временной интервал между отсчетами (то есть, интервал выборки) будет Δt , где :

$$\Delta t = \frac{1}{f_s}$$

Выборки сигнала обозначены $x[i]$, $0 \le i \le N-1$ (то есть, имеется N отсчетов). При использовании дискретного преобразования Фурье этих N отсчетов, осуществляемого с помощью формулы

$$X[k] = \frac{1}{N}\sum_{i=0}^{N-1} x[i]e^{-j2\pi ik/N} \quad \text{для } k = 0,1,2,\ldots,N-1$$

результирующий выходной сигнал $X[k]$, $0 \le k \le N-1$ является частотным представлением $x[i]$. Заметим, что и временное представление **x** , и частотное представление X имеют всего N отсчетов. Аналогично *временному интервалу* Δt между выборками x во временной области, мы имеем частотный интервал

$$\Delta f = \frac{f_s}{N} = \frac{1}{N\Delta t}$$

между компонентами X в частотной области. Интервал Δf также известен как *частотное разрешение*. Для увеличения частотной разрешающей способности (уменьшение Δf) нужно

либо увеличить число отсчетов N (с той же f_s), либо уменьшить частоту выборки f_s (при постоянном N).

Поскольку ДПФ – последовательность комплексных чисел, оно несет два информационных сообщения – об амплитуде и фазе. Для действительных сигналов ДПФ симметрично относительно N/2 (показатель Найквиста) со следующими свойствами:

$$|X[k]| = |X[N-k]|$$

и

$$\text{фаза}\,(X[k]) = -\,\text{фаза}\,(X[N-k])$$

В терминах, которые используются для описания этой симметрии, мы говорим, что величина $X[k]$ *четна*, а фаза $(X[k])$ – *нечетна* относительно частоты $\Delta f \cdot N/2$. Конечным эффектом этой симметрии в ДПФ действительных сигналов является то, что существует копия полезной информации, содержащейся в N комплексных выборках ДПФ. Благодаря этому повторению информации необходимо вычислять или визуализировать только половину отсчетов ДПФ реального сигнала, а другая половина может быть найдена, исходя из свойств симметрии.

Прямое применение ДПФ для N выборок данных требует приблизительно N^2 сложных операций. Однако, если размер последовательности есть 2 в некоторой степени, $N = 2^m$, где $m = 1, 2, 3, \ldots$ вычисление ДПФ может быть сокращено примерно до $N \log_2(N)$ операций. Это делает вычисления ДПФ намного быстрее, и в литературе по цифровой обработке сигналов (DSP) эти алгоритмы встречаются под названием Быстрого Преобразования Фурье (БПФ). БПФ есть не что иное, как алгоритм быстрого вычисления ДПФ, если количество отсчетов (N) равно 2 в некоторой целой степени. В некоторых случаях БПФ может быть применено для последовательностей со специфическими длинами (например, равными произведению небольших простых чисел).

Преимущества БПФ включают в себя скорость и эффективность использования памяти. Размер входной последовательности, однако, должен быть степенью числа 2. ДПФ может обрабатывать последовательность любой длины, но ДПФ медленнее, чем БПФ, и использует больше памяти, так как должно резервировать место под промежуточные результаты во время обработки.

В практических применениях, для того, чтобы размер входной последовательности соответствовал степени числа 2, надо добавить в ее конец столько нулей, чтобы общее число отсчетов было равно большему ближайшему значению 2-х в степени. Например, если есть 10

отсчетов сигнала, можно добавить 6 нулей и получить общее число выборок, равное 16-ти (=2^4— что есть 2 в степени). Для реализации этой процедуры можно использовать *Zero Padder.vi* из *Signal Processing→ Signal Operation.*

В добавление к тому, что общее число выборок доводится до 2-х в степени и тогда для увеличения скорости вычислений можно использовать БПФ, добавление нулей также увеличивает частотную разрешающую способность (напоминаем, что $\Delta f = f_s/N$) путем увеличения количества отсчетов N. Этот метод, однако, не особенно увеличивает оценочное разрешение частоты, а в большей степени обеспечивает интерполяцию в частотной области.

Формула для БПФ может быть обращена, т.е. по дискретному спектру может быть восстановлен дискретный же сигнал:

$$x[i] = \sum_{k=0}^{N-1} X[k] e^{j 2\pi i k / N}$$

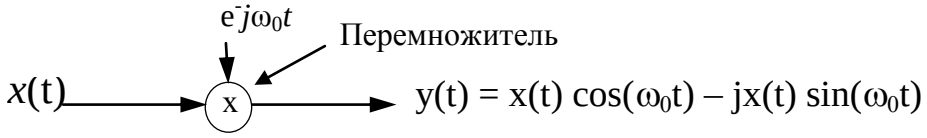
Это преобразование называется Обратным Дискретным Преобразованием Фурье (ОДПФ). При $N=2^m$ его вычисляют с помощью ОБПФ (*IFFT*).

2.3. Виртуальные приборы БПФ (*FFT и FFT*$^{-1}$) в палитре *Signal Processing→ Transforms*

Наряду с набором ВП генерации сигналов *Signal Processing→Signal Generation,* рассмотренным в параграфе 2.1., библиотека *Signal Processing* содержит подразделы *Transforms, Signal Operation, Filters* и *Windows*. ВП, которые вычисляют ДПФ и ОДПФ сигнала, находятся в подразделе *Signal Processing→ Transforms.* При $N=2^m$ эти ВП реализуют алгоритм БПФ, в противном случае – вычисляется ДПФ. На вход ДПФ можно подать действительный и комплексный (аналитический) сигналы. На выходе как ДПФ, так и ОДПФ, всегда получится последовательность комплексных чисел. ДПФ и ОДПФ принимают на вход одно- и двумерные последовательности.

Большинство сигналов, порождаемых физическими объектами, действительно-значны. Но в радио- и гидролокации, в телекоммуникационных системах в процессе модуляции/демодуляции сигналы очень часто представляют в виде «синфазной» и «квадратурной» компонент. В этом случае синфазная компонента должна играть роль действительной составляющей входного сигнала для БПФ, а квадратурная – мнимой. Спектр такого сигнала ассиметричен, в этом случае он уже не содержит избыточной информации.

Приведенная ниже схема иллюстрирует процесс однополосной модуляции, которая, если отвлечься от технических подробностей, представляет собой сдвиг спектра низкочастоного сигнала *x(t)*, содержащего передаваемую информацию, по частотной оси на величину ω_0. Уместно вспомнить здесь соответствующее свойство Фурье-преобразования.



Задание. Разработать ВП, который моделирует алгоритм однополосной модуляции. Для генерации информационного сигнала использовать какой-либо ВП из набора *Signal Processing→Signal Generation*, генерирующий сигнал с широким спектром. Сравнить спектры сигналов *x(t)* и *y(t)* при различных соотношениях между несущей частотой ω_0 и шириной спектра *x(t).*

В подразделе *Signal Processing→ Spectral* представлен также ряд ВП более высокого уровня, предназначенных для спектрального оценивания: *Power Spectrum.vi, Auto Power Spectrum.vi, Cross Power Spectrum* и т.д., Некоторые математические преобразования: Гильберта (*Fast Hilbert Transform.vi* и *Inverse Fast Hilbert Transform.vi*), Хартли (*FHT.vi* и *IFHT.vi*) и т.д. содержатся в подпалитре *Transforms.* Их рассмотрение выходит за рамки данного пособия. Преобразование Гильберта будет рассмотрено ниже.

2.4. Преобразование и анализ сигналов во временной области с помощью свертки и корреляции. Преобразование Гильберта и аналитический сигнал.

Как уже говорилось выше, наряду с инструментами частотного анализа, в палитре *Signal Processing* содержатся ВП анализа сигналов во временной области: *Signal Processing→ Signal Operation.* Мы ограничимся здесь только сверткой (*Convolution.vi*) и взаимной корреляцией (*Cross Correlation.vi*). Отметим, что при $N=2^m$ свертка и корреляция вычисляются через БПФ и ОБПФ (вспомните соответствующие теоремы!). Приведенный ниже пример иллюстрирует согласованную фильтрацию, которая, как известно, во временной области сводится к вычислению взаимной корреляции задержанного излученного и принятого сигналов. Нижний график на рис.41 показывает, что ЛЧМ сигнал с исходной длительностью 128 отсчетов и

амплитудой 1 сжимается до короткого импульса, амплитуда которого возрастает в десятки раз. Для сравнения показано, что свертка не дает нужного результата.

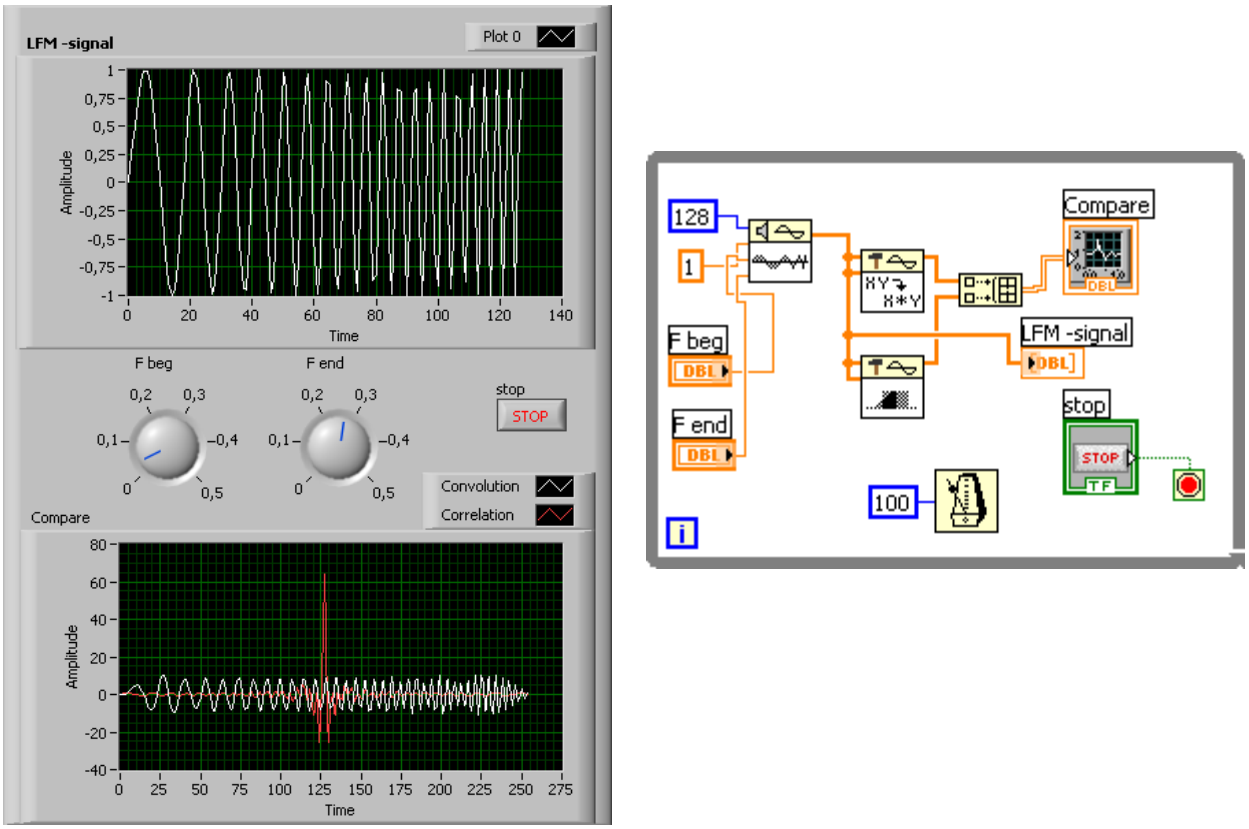


Рис.41 Сравнение корреляции и свертки на примере согласованной фильтрации ЛЧМ-сигнала

Задание. На основе ВП рис.41 разработать модель импульсной локационной системы. Для этого ввести блок генерации нескольких эхосигналов с различными задержками и амплитудами и добавить к ним аддитивный шум. Сравнить результаты выделения эхосигналов из шума с использованием согласованной фильтрации и без нее. Исследовать разные типы сигналов (простой радиоимпульс, ЛЧМ, шумовой сигнал с переменной шириной спектра).

Обратим внимание на несколько ВП, которые осуществляют операции, полезные при обработке эхосигналов. *Peak Detector.vi* ищет пики (или провалы) в сигнале независимо от их уровня. При этом используется квадратичная аппроксимация в окрестности локального максимума (минимума). Это дает возможность определить положение пика (провала) с

лучшим разрешением, чем шаг временной дискретизации сигнала. *Treshold Peak Detector.vi* ищет пики, превышающие заданный уровень.

Преобразование Гильберта - *Fast Hilbert Transform.vi* (ВП расположен в *Signal Processing→Transforms*) осуществляет преобразование сигналов во временной области (приведены выражения для непрерывных функций времени, интеграл берется в смысле главного значения):

$$Q(t)=\frac{1}{\pi}\int_{-\infty}^{\infty}\frac{S(t')dt'}{t-t'}$$ - прямое и $$S(t)=\frac{1}{\pi}\int_{-\infty}^{\infty}\frac{Q(t')dt'}{t-t'}$$ - обратное.

Из произвольного действительного сигнала *S(t)* и его преобразования Гильберта *Q(t)* можно составить т.н. «аналитический сигнал»: *Z(t)=S(t)+jQ(t)*. Он широко используется для теоретических исследований в теории связи, радио- гидролокации. Через действительную *S(t)* и мнимую *Q(t)* составляющие могут быть выражены его «огибающая» *E(t)*, «фаза» *Φ(t)* и «мгновенная частота» *Ω(t)=dΦ(t)/dt*:

$$E(t)=\sqrt{S(t)^2+Q(t)^2}\ ,\ \Phi(t)=arctg(S(t)/Q(t))\ ,\ \Omega(t)=(S'Q-SQ')/(S(t)^2+Q(t)^2)\ .$$

Одно из замечательных свойств аналитического сигнала иллюстрирует следующий пример (см. рис.42). В нем сравниваются спектры действительного ЛЧМ – сигнала и соответствующего ему сигнала аналитического. Видно, что спектральная амплитуда исходного сигнала симметрична относительно частоты $f_s/2$, тогда как спектральная амплитуда аналитического сигнала уже несимметрична и существенно отлична от нуля только в области $f_s/2<f<f_s$. Если входы узла *Re/Im to Complex* поменять местами, то спектр переместится в область $0<f<f_s/2$.

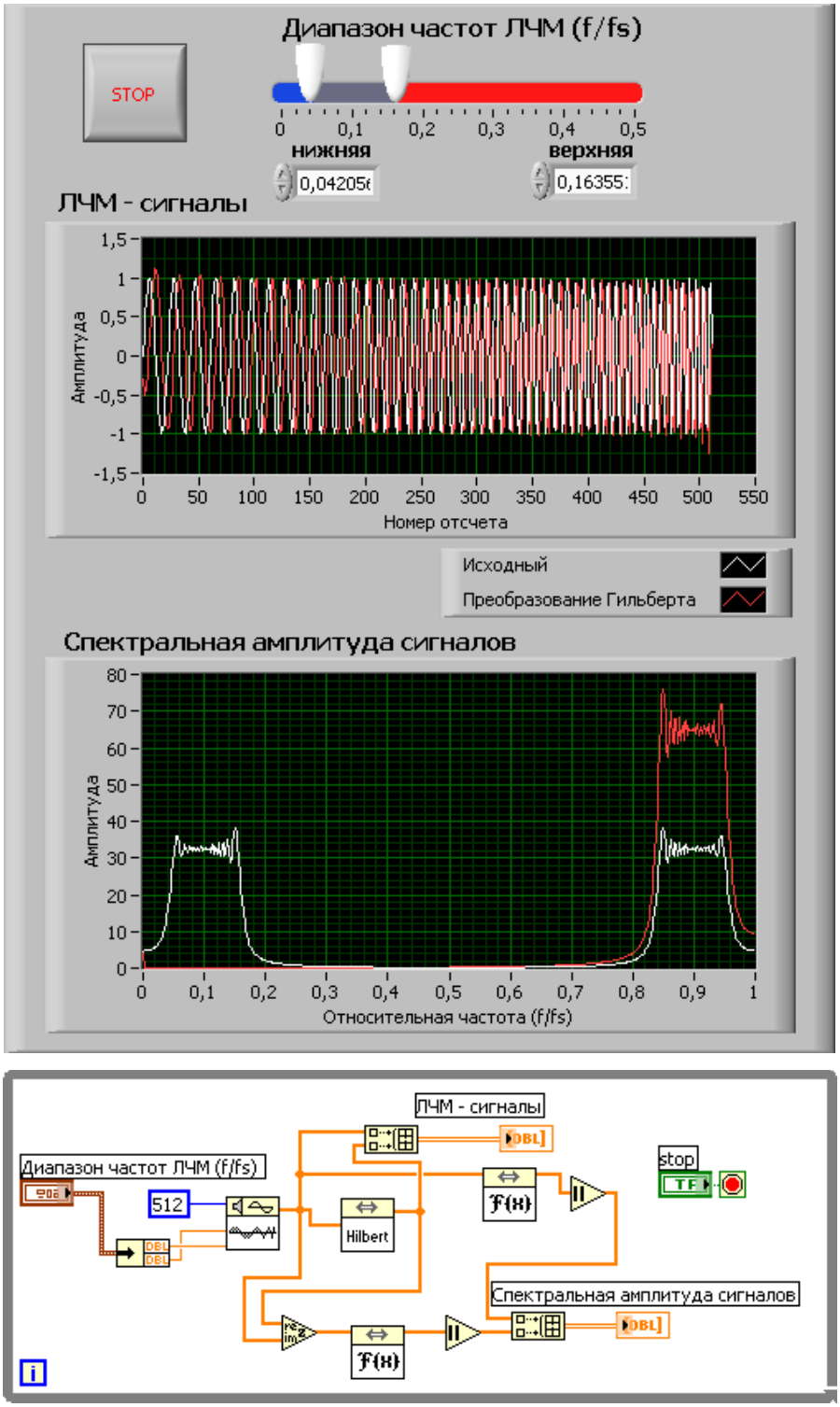


Рис.42 Преобразование действительного ЛЧМ-сигнала в аналитический

2.5. Цифровая фильтрация (*Signal Processing→Filters*)

Фильтрация – это процесс, с помощью которого изменяется частотный спектр сигнала. Это одна из наиболее широко используемых процедур обработки сигналов в научном эксперименте, радио- и гидролокации, технике связи, автоматическом регулировании, медицинской диагностической аппаратуре, аудио- и видеоаппаратуре и т.д. В связи с все более широким проникновением в эти области цифрового представления сигналов аналоговая фильтрация вытесняется цифровой. Теории, программной и аппаратной реализации цифровых фильтров (ЦФ) посвящена обширная литература. Изложить даже основы этого специального направления в рамках данного пособия невозможно. Будем предполагать, что изучающий это пособие освоил курсы высшей математики и теоретических основ радиотехники (теории цепей и сигналов) и напомним основные понятия и определения, которые необходимы для понимания примеров и выполнения упражнений.

1) Фильтры изменяют или удаляют ненужные частоты. В зависимости от частотного диапазона, который фильтры либо пропускают, либо ослабляют, они могут быть классифицированы на следующие типы:

   Низкочастотный фильтр – пропускает низкие частоты, но ослабляет высокие.

   Высокочастотный фильтр – пропускает высокие частоты, но ослабляет низкие.

   Полосовой фильтр – пропускает определенную полосу частот.

   Заграждающий фильтр – ослабляет определенную полосу частот.

   На рис. 43 схематически показаны частотные характеристики перечисленных фильтров. Обозначения: ПП - полоса пропускания, ПР – полоса режекции (непропускания), ПО – переходная область, f_c – частота среза.

2) Импульсная характеристика ЦФ ***h*[k] –** это его отклик на единичный импульс, т.е. сигнал, заданный последовательностью: $x[0]=1$ и $x[i]=0$ при $i \neq 0$.

3) ДПФ от импульсной характеристики есть частотная характеристика или коэффициент передачи фильтра $H[n]$:

$$H[n] = \frac{1}{N}\sum_{k=0}^{N-1} h[k] \cdot \exp[-j2\pi kn/N]$$

4) По виду импульсной характеристики ЦФ делятся на две группы: КИХ-фильтры (*FIR-filters*) - фильтры с импульсной характеристикой, имеющей конечную длительность, и

БИХ-фильтры (*IIR-filters*), у которых импульсная характеристика длится бесконечно долго.

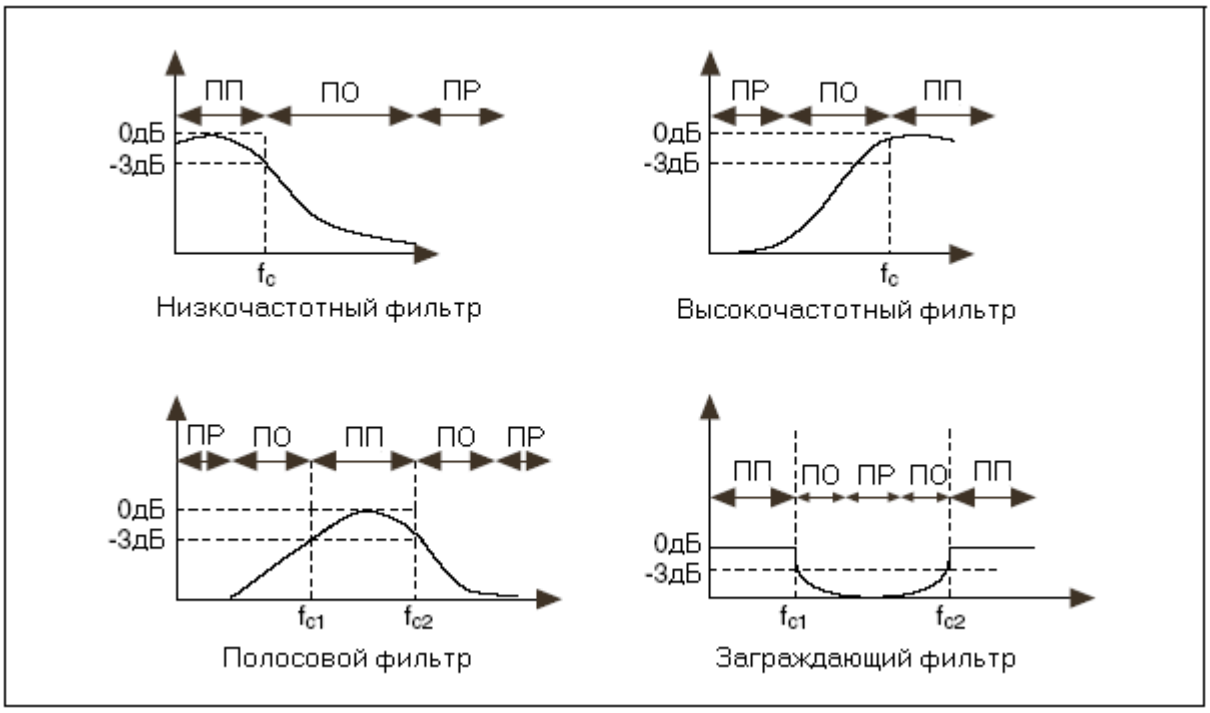


Рис. 43 Схематический вид частотных характеристик различных фильтров

5) По аналогии с линейным аналоговым фильтром, который описывается линейным дифференциальным уравнением с постоянными коэффициентами, цифровой фильтр описывается линейным уравнением в конечных разностях:

$$Y[k]=\left(\sum_{m=0}^{M_B-1}B[m]X[k-m]-\sum_{n=1}^{N_A-1}A[n]Y[k-n]\right)\frac{1}{A[0]}$$

здесь: X[n] и Y[k] - входная и выходная последовательности (сигналы); B[m] – «прямые» коэффициенты (*Forward Coefficients*); A[n] – «обратные» коэффициенты (*Reverse Coefficients*).

Если все $A[n]=0$ получим КИХ-фильтр, если $A[n]\neq 0$ – БИХ-фильтр. Поскольку для вычисления выходного сигнала используются его же предыдущие значения, такой фильтр также называют рекурсивным. Обычно значение M_B равно N_A, а *порядок* фильтра равен $N_A - 1$. В БИХ-фильтрах, реализованных в LabVIEW, A[0] всегда равен 1.

Синтезу или проектированию фильтров, т.е. методам расчета коэффициентов, при которых его АЧХ и ФЧХ удовлетворяют определенным требованиям, посвящена обширная литература. БИХ-фильтры хороши тем, что позволяют получить заданный спад АЧХ при меньшем числе коэффицентов по сравнению с КИХ-фильтрами. Однако БИХ-фильтры склонны к неустойчивости. Поэтому даже если формально фильтр устойчив, ошибки вычислений могут превратить его в «генератор» сигнала. Кроме этого, ФЧХ БИХ-фильтров нелинейна. Для КИХ-фильтров характерна линейная ФЧХ, что обеспечивает постоянство групповой задержки.

2.5.1. Обзор КИХ-фильтров из палитры *Signal Processing→Filters*.

КИХ – фильтры представлены набором из 4-х ВП, каждый из которых реализует один из 4-х перечисленых выше типов фильтров с одинаковым уровнем неравномерности в полосах пропускания и режекции: *Equi-Ripple LowPass.vi* ... *Equi-Ripple BandStop.vi*. Отдельный ВП *FIR Widowed Filter.vi* является универсальным и в зависимости от значения входа *filter type* позволяет реализовать любой тип фильтра. Значение входа «Отводы» (*Taps*) должно быть четным для высокочастотных и заграждающих фильтров, так как если оно четное, то амплитудная характеристика стремится к нулю при приближении значения частоты к половинному значению частоты дискретизации. Чем больше эта величина, тем длиннее импульсная характеристика фильтра и, соответственно, дольше длится переходный процесс (тем больше задержка появления сигнала на выходе фильтра).

Приведенный ниже пример позволяет познакомиться с характеристиками универсального КИХ-фильтра. Комментарии: 1) значение **f вч** влияет на работу ВП только для полосовых фильтров (пропускающего и заграждающего) и определяет верхнюю частоту полосы; 2) ошибка отличается от нуля, если задано сочетание параметров, при котором невозможен синтеза фильтра; 3) обратите внимание, что ФЧХ фильтров всегда линейна; 4) для получения непрерывной зависимости фазы от частоты использован ВП *Unwrap Phase.vi*; 5) для получения на выходе фильтра импульсной характеристики на его вход подается одиночный импульс, генерируемый ВП *Impuls Pattern.vi*; 6) частотная характеристика вычисляется как БПФ от импульсной.

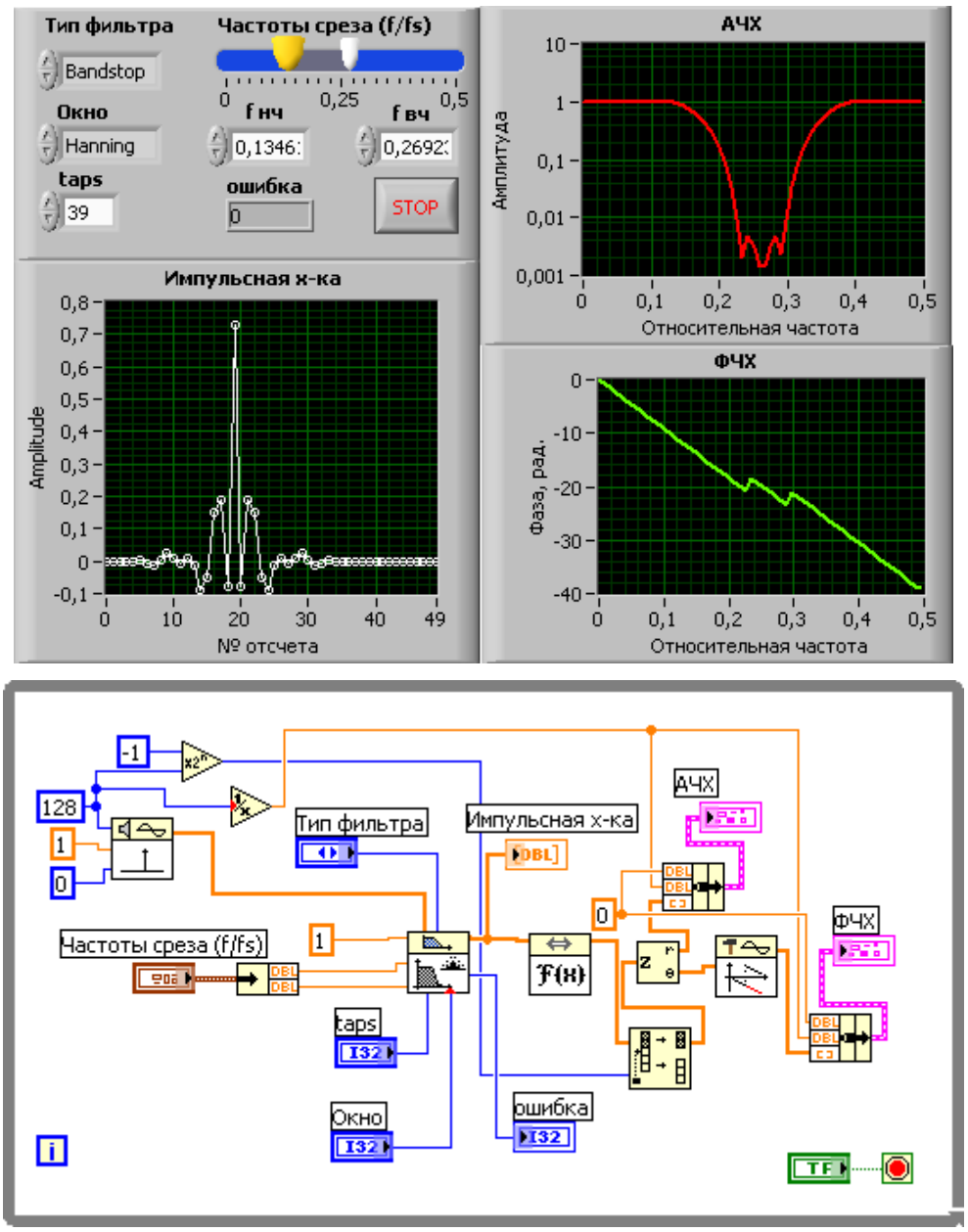


Рис.44 Пример для ознакомления с работой ВП *FIR Windowed Filter.vi*

2.5.2. Обзор БИХ-фильтров из палитры *Signal Processing→Filters***.**

Первый ряд палитры *Signal Processing→Filters* содержит набор ВП БИХ-фильтрации различных видов **(**см. ниже таблицу). Тип фильтра выбирается соответствующим элементом управления. Не рекомендуется выбирать порядок БИХ-фильтров больше 20, т.к. это может привести к неустойчивости.

Справа расположены входы в подпалитры ВП синтеза коэффициентов БИХ и КИХ-фильтров, а также ВП, реализующих различные алгоритмы вычислений при цифровой фильтрации. Работа с ними требует более глубокого понимания теории ЦФ.

Модель БИХ-фильтра	Частотные характеристики	Ширина переходной области для указанного порядка фильтра	Требуемый порядок для удовлетворения заданным техническим требованиям
Баттерворта Чебышева Инверсный Чебышева Эллиптический	Нет волнистости Волнистость в ПП Волнистость в ПР Волнистость в ПП и ПР	Самая широкая Самая узкая	Самый высокий Самый низкий

2.5.3. Нелинейный медианный фильтр (*Signal Processing→Filters*)

Наряду с линейными фильтрами, в LabVIEW включен т.н. «медианный фильтр» (*Median Filter.vi*). Его основное назначение – устранять из сигналов импульсные помехи. При этом такой фильтр практически без искажений пропускает «ступеньку». Это свойство делает его 2-мерный вариант незаменимым при очистке изображений от коротких импульсных помех. Контуры элементов изображения при этом не размываются. Применение традиционного ФНЧ привело бы к расплыванию изображения.

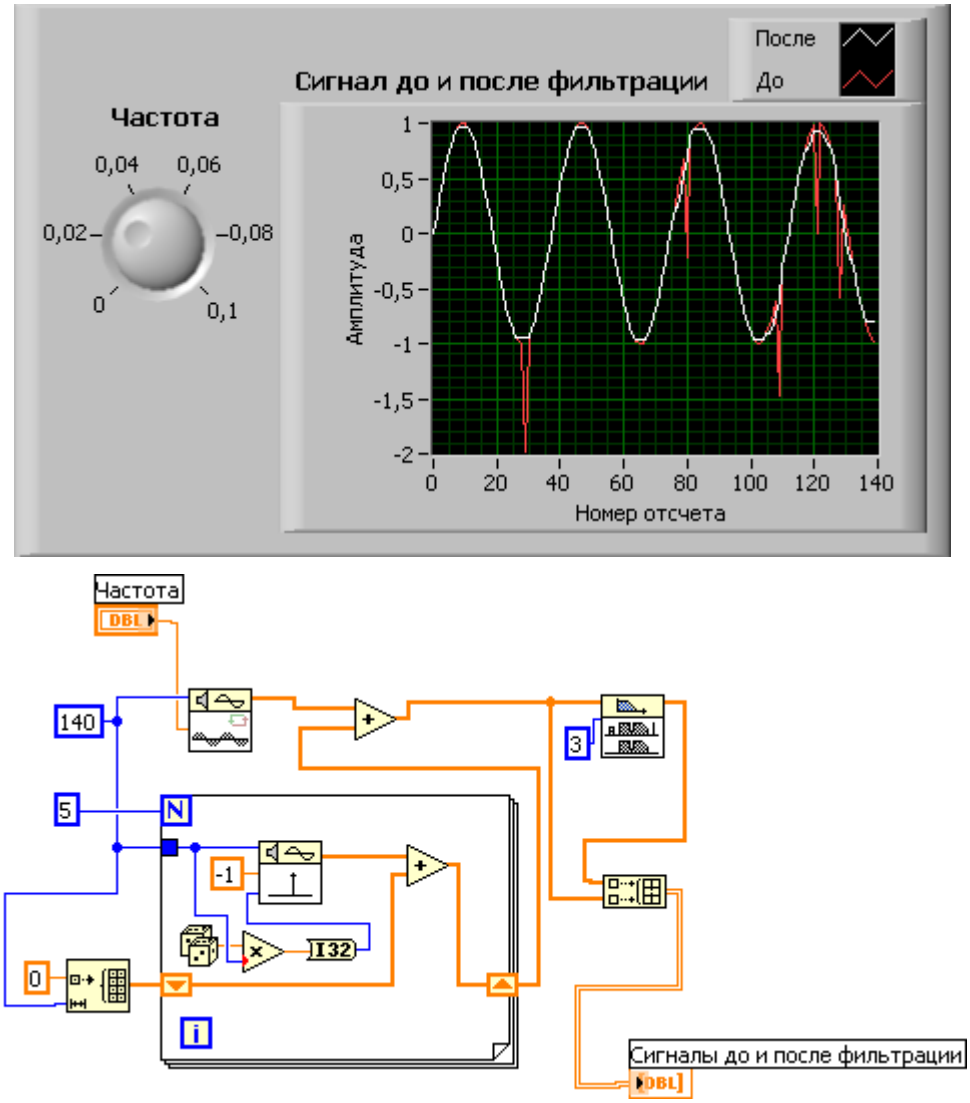


Рис.45 Удаление импульсной помехи из сигнала с помощью медианного фильтра (*Fir_med.vi*). Обратите внимание на использование генератора случайных чисел для задания положения импульса и сдвигового регистра и суммирования в цикле **For Loop** для создания сигнала, состоящего из импульсов. В ВП ***Sine Wave.vi*** частота задается как f= число периодов / число отсчетов.

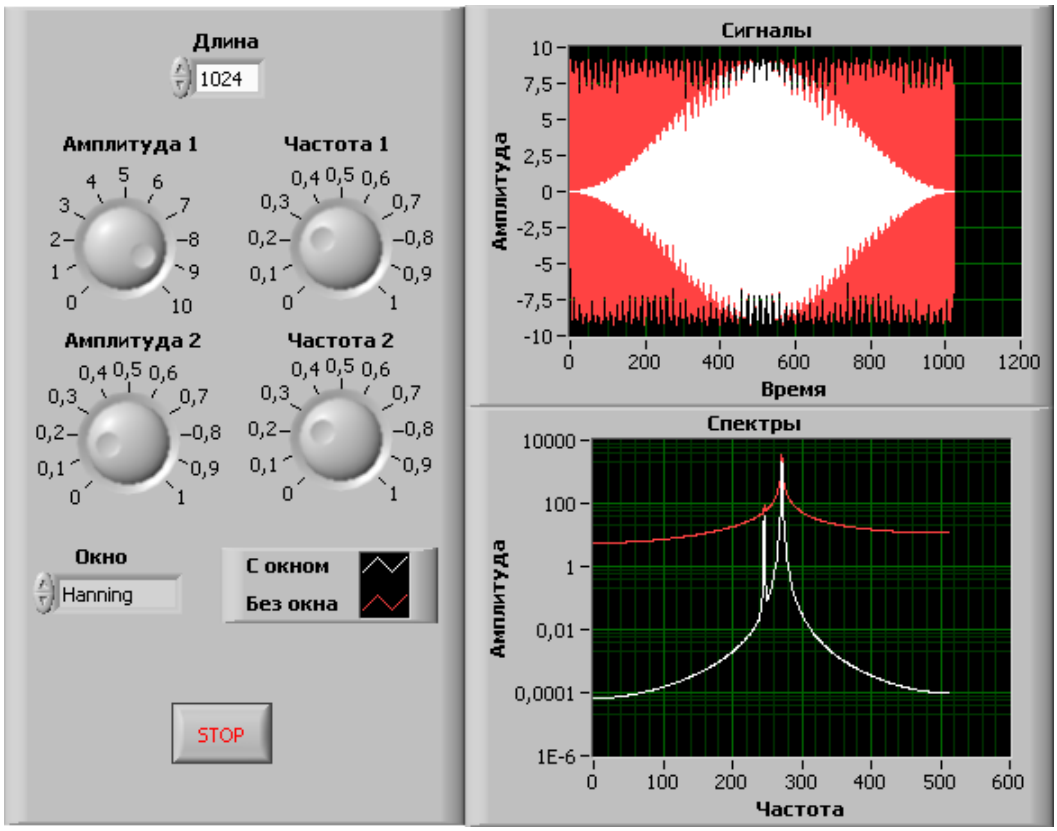
2.6. Использование окон при спектральном анализе сигналов (*Signal Processing→Windows).*

Спектральное окно – это весовая функция, на которую во временнной области умножается исследуемый сигнал перед спектральным анализом (СА). Необходимость применения окон обусловлена следующими причинами.

Во-первых, это явление «просачивания» спектральной энергии (*Spectral Leakage*). Его можно проиллюстрировать на примере СА отрезка синусоиды. Если в окно укладывается целое число периодов, то дискретный спектр отличен от нуля только в одной точке. Если же число периодов не целое, спектр отличен от нуля во всех точках. Нетрудно убедиться, что в этом случае на концах окна анализа в сигнале возникают разрывы. Для уменьшения их влияния сигнал умножают на функцию, спадающую на концах интервала анализа.

Во-вторых, при использовании СА для выделения слабого гармонического сигнала на фоне сильного с близкой частотой, умножение на окно позволяет добиться быстрого спада спектрального отклика при отстройке от частоты сильного сигнала.

В палитре *Signal Processing→Windows* располагаются ВП, умножающие входную последовательность на различные окна: *Hamming Window.vi, Hanning Window.vi, Flat Top Window.vi* и т.д. Приведенный ниже пример (см. рис.45) знакомит со свойствами некоторых из этих окон применительно к упоминавшейся выше задаче выделения слабого гармонического сигнала на фоне сильного.



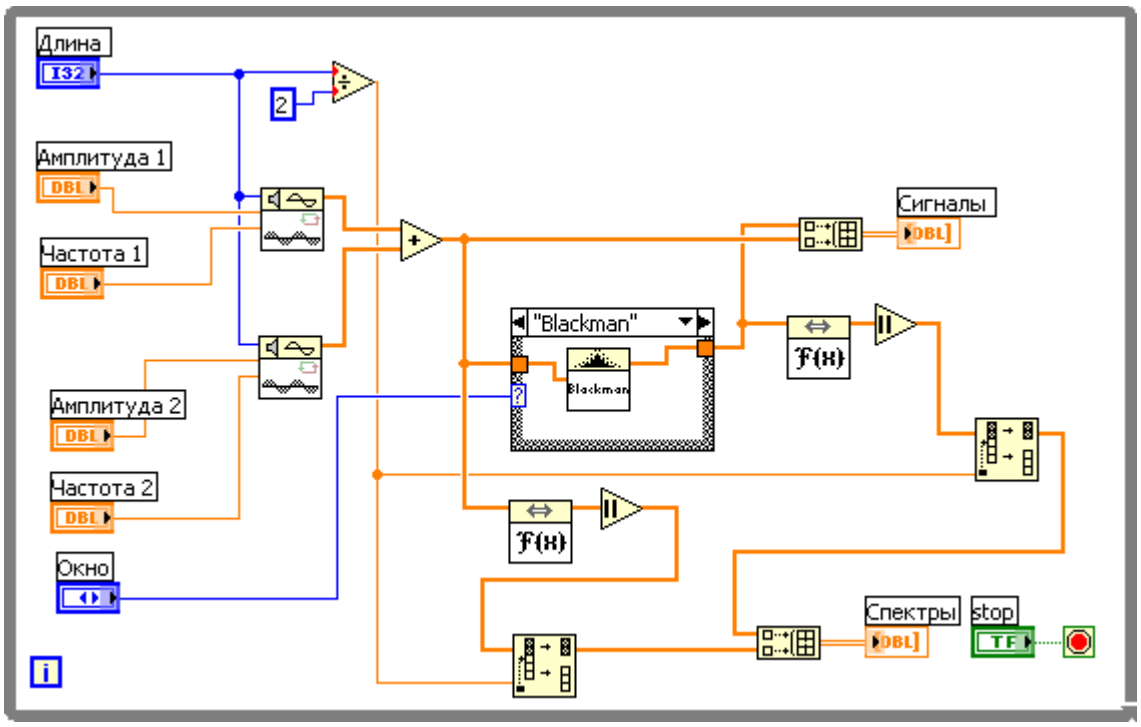


Рис.46 Сравнение различных окон при спектральном анализе

3. ВВОД И ГЕНЕРАЦИЯ АНАЛОГОВЫХ СИГНАЛОВ

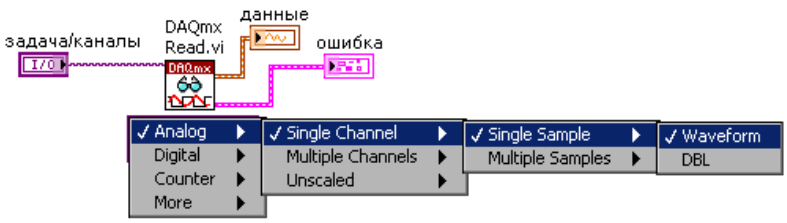
В этом разделе мы опишем решения, необходимые для получения аналоговых сигналов и записи их на диск, а также ряд возможностей LabVIEW, которые можно использовать совместно с виртуальными приборами драйвера NI-DAQmx.

3.1. Аналоговый ввод

Цифровая обработка сигналов обладает многими преимуществами, поэтому перед обработкой в компьютере аналоговые сигналы преобразуются в цифровую форму. Цифровым сигналом называется такой сигнал, который может принимать ограниченный ряд значений.

3.2. Использование ВП DAQmx Read

Виртуальный прибор (ВП) DAQmx Read (DAQmx Чтение), расположенный в палитре DAQmx - Data Acquisition, считывает выборки из заданной задачи или канала. Вводы этого полиморфного ВП задают формат возвращаемых выборок, считывание одной или нескольких выборок, и считывание из одного или нескольких каналов. Используйте выпадающее меню для конфигурации ввода этого ВП, как показано на рисунке ниже.



В первом меню вы можете выбрать следующие типы ввода:

• Аналоговый (Analog)

• Цифровой (Digital)

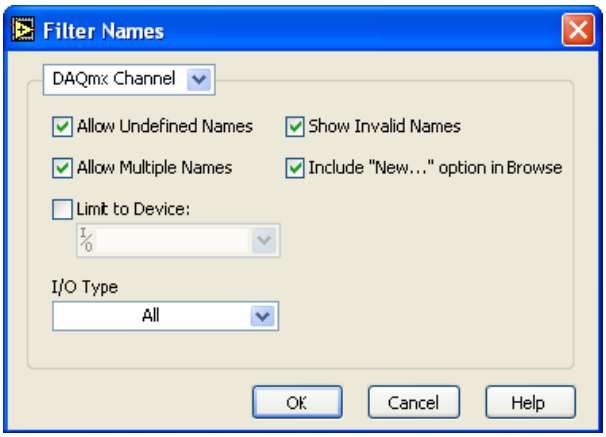
• Ввод счетных сигналов (Counter)

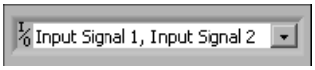
• Дополнительный (необработанные данные) (More (Raw Data))

Второе меню имеет следующие опции: один канал для чтения (single channel), несколько каналов для чтения (multiple channels) и считывание данных непосредственно с устройства без предварительного масштабирования (unscaled). В третьем меню выбирается режим сбора данных: одноточечный (single sample) или многоточечного (multiple samples). При выборе одноточечного сбора данных используйте четвертое меню для задания формата

возвращаемых данных: осциллограмма (waveform) или число с плавающей запятой удвоенной точности (DBL). При выборе многоточечного сбора данных используйте это меню для возврата данных в виде осциллограммы или массива числовых значений с плавающей запятой удвоенной точности.

При адресации аналогового ввода или вывода, возможно, вам понадобится обращаться одновременно к нескольким каналам. Если эти каналы имеют одинаковые типы тактирования и триггеров, объедините их в задачу. В противном случае используйте инструмент фильтрации имени (I/O Name Filtering) в контекстном меню элемента управления/константы имени NI-DAQmx задачи или канала и выберите опцию **Allow Multiple Names** (**Разрешить несколько имен**). Разделяйте имена каналов запятыми. Вы не сможете обращаться к нескольким задачам одновременно.





3.3. Тип данных осциллограмма

Тип данных осциллограмма – кластер, состоящий из следующих элементов:

• Y – Одномерный массив числовых данных, которыми могут быть единственная точка или осциллограмма, в зависимости от операции сбора данных. Элементы массива имеют представление чисел с плавающей запятой удвоенной точности (DBL).

• t0 – Скалярное значение, представляющее время получения первой точки массива Y в соответствии с системными часами. Этот элемент называю также начальным временем или отметкой времени (timestamp).

• t – Скалярное значение, представляющее временной интервал между точками данных массива Y.

• Attributes (Атрибуты) – Строка, позволяющая добавлять к осциллограмме дополнительную информацию, такую как номер устройства или канала.

Тип данных осциллограммы имеет много преимуществ перед обычным массивом масштабированных данных.

• Наличие t0 – Когда этого типа данных не существовало, вы не могли определить время получения данных. Теперь же тип данных осциллограмма автоматически возвращает реальное время в виде элемента t0.

• Более легкое отображение на графике – Тип данных осциллограмма упрощает построение данных на графике. Предыдущие версии LabVIEW требовали объединения значения начальной точки (x0) и интервала между точками (x) с данными (массив Y). Тип данных осциллограмма содержит все эти элементы, так что все, что вам остается сделать – это присоединить тип данных осциллограмма к графику осциллограмм.

• Более легкое построение нескольких графиков – Тип данных осциллограмма упрощает построение нескольких графиков. Предыдущие версии LabVIEW требовали объединения x0, x и массива Y для каждого из графиков, затем построения из этих кластеров массива для отображения нескольких графиков. Используя тип данных осциллограмма, вы просто присоединяете 1D массив осциллограмм к графическому индикатору для построения на нем нескольких графиков. Если вы собираете данные из нескольких каналов при помощи ВП аналогового ввода, то последний возвращает 1D массив, который вы можете подсоединить непосредственно к графическому индикатору.

Задание:

Получить аналоговый сигнал, используя устройство сбора данных.

В результате должен быть создан ВП, измеряющий напряжение в нулевом канале платы сбора данных.

Решение:

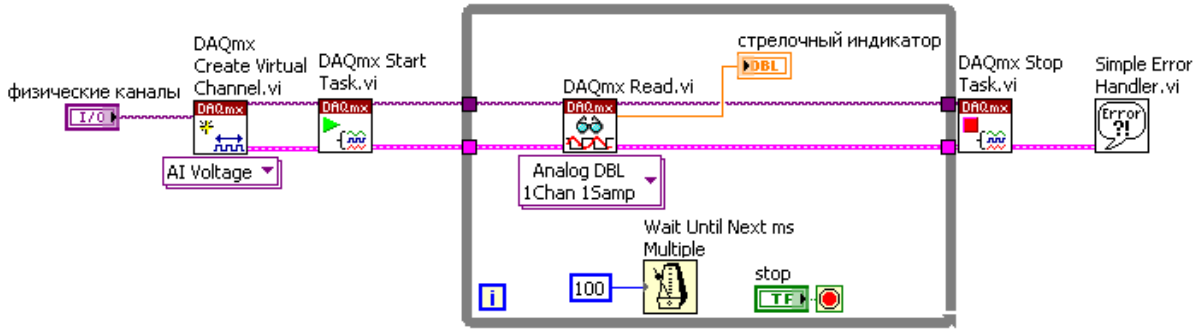
Лицевая панель

1. Откройте новый ВП и создайте следующую лицевую панель.

Настройте шкалу стрелочного индикатора для отображения диапазона 0.0 – 0.4. Дважды щелкните на отметке 10.0 и напечатайте 0.4. Возможно, вам придется увеличить индикатор для более подробного отображения шкалы.

Блок-диаграмма

2. Создайте следующую блок-диаграмму.



Описание:

ВП DAQmx Create Virtual Channel, расположенный в палитре Functions»All Functions»NI Measurements»DAQmx – Data Acquisition, создает виртуальный канал такого типа, который задается в выпадающем меню конфигуратора этого ВП. Выберите тип AI Voltage из этого выпадающего меню.

ВП DAQmx Start Task, расположенный в палитре Functions»All Functions»NI Measurements»DAQmx – Data Acquisition, запускает измерительную задачу.

ВП DAQmx Read, расположенный в палитре Functions»All Functions»NI Measurements»DAQmx – Data Acquisition, выполняет операцию чтения, заданную вами в меню конфигуратора. Выберите следующие опции настройки ввода: Analog»Single Channel»Single Sample»DBL. При такой настройке прибор возвращает одну выборку данных в виде числа удвоенной точности с плавающей запятой из одного канала аналогового ввода.

ВП DAQmx Stop Task, расположенный в палитре Functions»All Functions»NI Measurements»DAQmx – Data Acquisition, останавливает выполнение измерительной задачи.

На лицевой панели установите для физического канала значение Dev X/ai0, где X – это номер вашего DAQ устройства в MAX.

3.4. Терминология, применяемая при дискретизации сигналов

Тактовый генератор выборки (Sample Clock) – Последовательность импульсов, используемых для запуска сбора данных. Каждый раз, когда тактовый генератор выборки (ТГВ) выдает импульс, принимается одна выборка из одного канала.

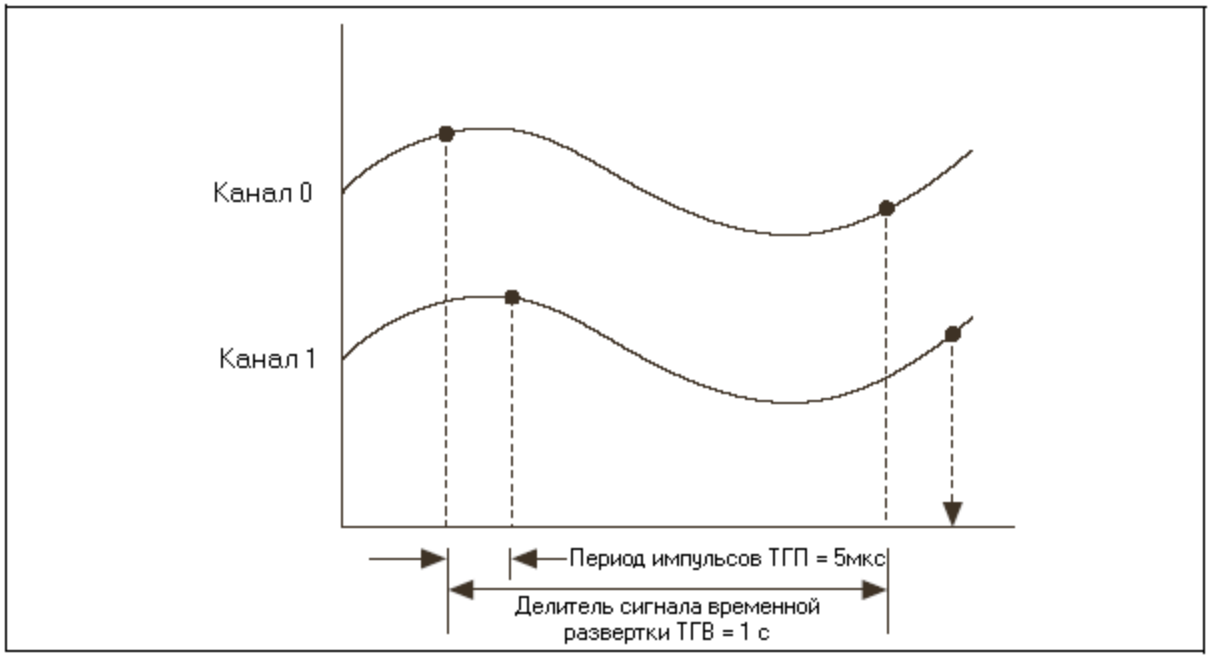
Тактовый генератор преобразователя аналогового ввода (AI Convert Clock) (ТГП) – Последовательность импульсов, используемых для запуска АЦП.

Длительность выборки (Sample Duration) – Время, которое занимает получение одного набора выборок из каналов. Используйте следующую формулу для вычисления длительности выборки:

Длительность выборки = (# каналов – 1) × период импульсов ТГП

3.5. Интервальная выборка

При оцифровке сигнала применяются различные способы дискретизации: интервальный, циклический и одновременный. Следующая иллюстрация показывает пример интервальной выборки.



В наиболее распространенном методе интервальной выборки (interval sampling) все каналы устройства использует один АЦП. Эту архитектуру можно найти в большинстве устройств E-серии. При этом для управления мультиплексором (multiplexer – MUX) используются тактовые генераторы выборки и преобразователя аналогового ввода. Для того чтобы понять, каким образом взаимодействуют эти генераторы, рассмотрим пример получения данных из двух каналов. Когда генератор выборки дает сигнал о начале сбора данных, мультиплексор подключает первый канал к АЦП, и однократно срабатывает тактовый генератор преобразователя. Как только ТГП посылает импульс, АЦП получает одну точку данных из первого канала. Перед тем как ТГП снова пошлет импульс, мультиплексор подключит второй канал к АЦП. Как только это произойдет, со следующим импульсом ТГП АЦП возьмет одну точку данных уже из второго канала. По завершении времени (длительности) выборки, ТГВ снова генерирует импульс, и цикл повторяется. ТГВ задает частоту, с которой устройство производит выборку данных из всех каналов. А тактовый генератор преобразователя фактически управляет получением выборок. Поскольку при интервальной выборке используются ТГВ и ТГП, устройство может производить выборку из каналов за короткий промежуток времени.

3.6. Многоточечный (буферизированный) аналоговый ввод

Для того чтобы получить несколько точек данных за раз, выберите настройку конфигуратора входов ВП DAQmx Read, отвечающую за считывание нескольких выборок. Для создания ВП для буферизированного аналогового ввода с аппаратным тактированием используйте ВП DAQmx Read совместно с виртуальными приборами DAQmx Timing, DAQmx Start Task и DAQmx Stop Task.

• Сбор данных с аппаратным тактированием – Сигналы, сгенерированные аппаратно, такие как импульсы тактового генератора выборки или преобразователя, управляют скоростью сбора данных. Подобные тактовые генераторы гораздо более быстры, чем созданные программно с использованием циклов. Поэтому вы можете оцифровывать сигнал в большем диапазоне частот без опасения, что возникнет эффект наложения спектров. Кроме этого, аппаратные тактовые генераторы более точны, чем программные циклы. На последние могут оказывать влияние различные действия с компьютером, такие как открытие другой программы, в то время как тактовый генератор не подвержен подобного рода воздействиям.

• Буферизированный сбор данных – Получение нескольких точек данных за один вызов устройства. Перед считыванием в LabVIEW, выборки передаются из устройства в промежуточный буфер в памяти.

3.7. ВП DAQmx Timing

ВП DAQmx Timing (ВП DAQmx Тактирование) настраивает частоту выборки, число выборок для сбора или генерации и создает буфер в случае необходимости. Конфигуратор этого полиморфного ВП задает тип тактирования, используемого в задаче. Существуют следующие типы тактирования: Sample Clock (Тактовые импульсы выборки), Handshaking (Квитирование), Implicit (Неявное) и Use Waveform (Использовать осциллограмму).

Для ввода аналогового сигнала, выберите опцию Sample Clock в выпадающем меню этого ВП. При такой настройке конфигуратора ВП содержит следующие параметры:

• sample mode (режим выборки) – Определяет, будет ли задача выполняться непрерывно или в течение ограниченного промежутка времени.

• samples per channel (количество выборок на канал) – Задает количество выборок для сбора или генерации, если вход sample mode имеет значение Finite Samples (Ограниченное

кол-во выборок). Эта величина определяет размер промежуточного буфера памяти, который хранит данные во время их передачи из DAQ устройства в LabVIEW.

• rate (частота) – Задает частоту выборки в единицах количества выборок на канал в секунду. Если вы используете внешний источник тактовых импульсов выборки, установите значение этого входа в максимальное ожидаемое значение внешнего генератора.

• source (источник) – Задает терминал источника тактовых импульсов выборки. Оставьте этот ввод неподключенным для использования стандартного генератора, встроенного в DAQ устройство.

• active edge (активный фронт) – Определяет, какой фронт тактового импульса использовать для получения или генерации выборок. Можно выбрать нарастающий (rising) или спадающий (falling) фронты импульсов ТГВ.

• task/channels in (входная задача/каналы) – Задает имя задачи или список виртуальных каналов, в которых будет проводиться сбор данных. При использовании списка каналов NI-DAQmx автоматически создает задачу.

Опция Handshaking (Квитирование) конфигуратора ВП DAQmx Timing определяет число выборок дискретного сигнала для сбора или генерации с использованием подтверждения установления связи (квитирования) между DAQ устройством и периферийным устройством. За большей информацией о данной настройке конфигуратора ВП DAQmx Timing обращайтесь к восьмому занятию, Цифровой ввод/вывод.

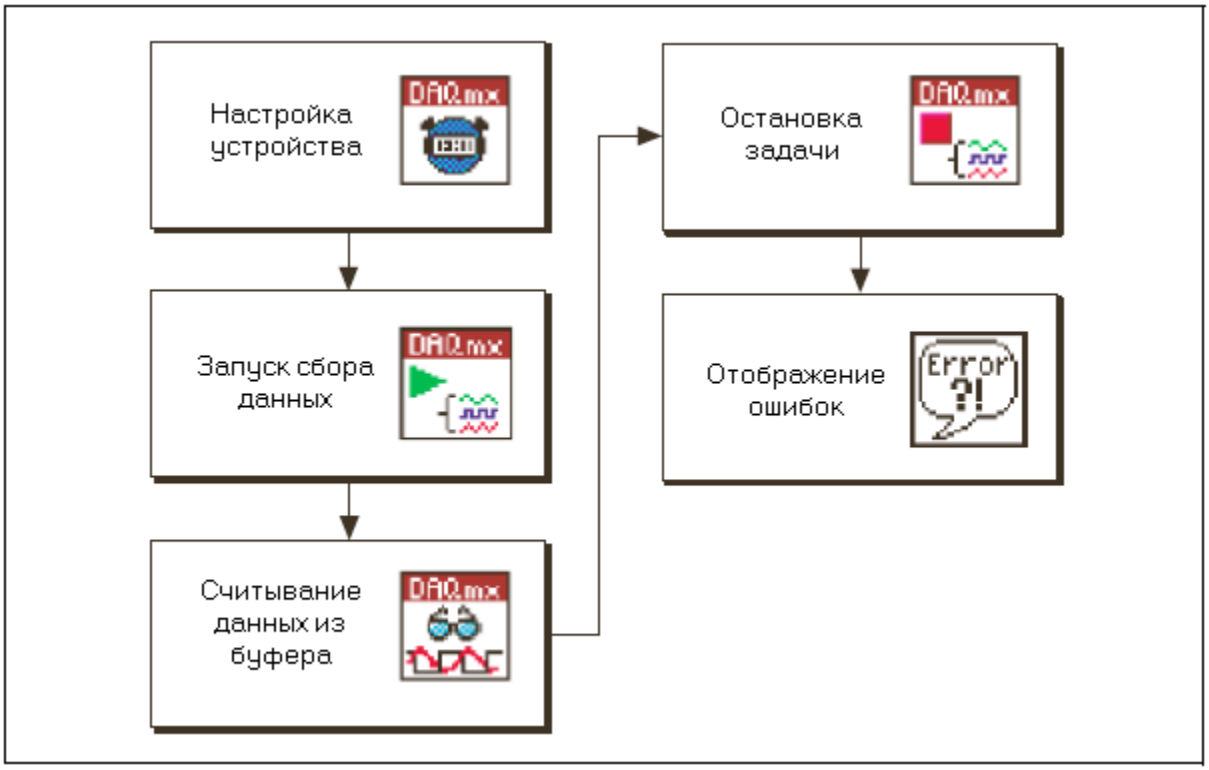
При выборе неявного (Implicit) типа тактирования можно задать только число выборок для сбора или генерации без какой-либо информации о временных параметрах. Обычно эта опция используется, когда задача не требует тактирования выборки, например, в случае использования счетчиков для буферизированного измерения частоты/периода или генерации последовательности импульсов.

При выборе опции Use Waveform (Использовать осциллограмму) конфигуратора ВП DAQmx Timing для определения частоты ТГВ используется компонента dt осциллограммы, подключенной к входу waveform. dt – это время между выборками, выраженное в секундах. Если на вход sample mode подано значение Finite Samples, то NI-DAQmx генерирует столько выборок, сколько их есть в осциллограмме. Этот ВП на самом деле не генерирует ни одной выборки. Для того чтобы сгенерировать сигнал, необходимо соединить этот ВП с ВП DAQmx

Write. За большей информацией о данной настройке конфигуратора ВП DAQmx Timing обращайтесь к седьмому занятию, Аналоговый вывод.

3.8. Блок-схема буферизированного сбора данных

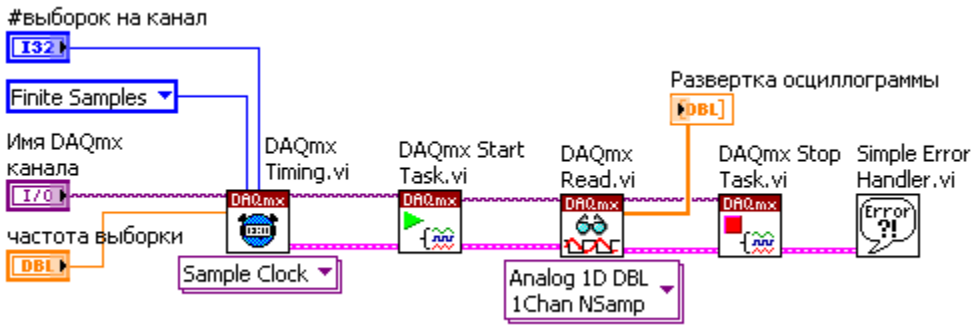
На следующей блок-схеме отображены основные моменты буферизированного сбора данных. Он требует задания определенного количества точек для получения с определенной частотой. Для настройки временных параметров и буфера устройства используйте ВП DAQmx Timing. Для запуска сбора данных применяйте ВП DAQmx Start Task. Далее, ВП DAQmx Read ждет, пока все выборки каждого из каналов не будут получены, затем возвращает данные и продвигает задачу дальше. ВП DAQmx Stop Task останавливает задачу и высвобождает ресурсы, выделенные устройству. Обработчик ошибок Error Handler показывает все ошибки, возникшие при выполнении.



3.9. Пример буферизированного сбора данных

Следующий пример демонстрирует создание ВП для буферизированного сбора данных. ВП DAQmx Timing задает задачу/канал, тактирование и количество выборок на канал (размер буфера). Затем, ВП DAQmx Start Task запускает сбор данных. После этого программа ждет на этапе ВП DAQmx Read, пока не заполнится весь буфер. Когда это произойдет, ВП DAQmx

Read возвращает данные из буфера, ВП DAQmx Stop Task останавливает сбор данных, и ВП Simple Error Handler показывает возникшие ошибки.



Поскольку ввод числа выборок на канал (number of samples per channel) ВП DAQmx Read остался неподключенным, NI-DAQmx автоматически определяет количество выборок для чтения, основываясь на конфигурации ВП DAQmx Timing. NI-DAQmx автоматически определяет это значение и устанавливает ввод number of samples per channel в значение –1. ВП DAQmx Read возвращает 2D массив, который можно непосредственно подать на график. В отличие от типа данных «осциллограмма», массив не содержит никакой временной информации.

Всегда соединяйте входные и выходные терминалы кластеров ошибок ВП между собой. Если на вход error in какого либо из ВП DAQmx Start Task, DAQmx Read или DAQmx Stop Task поступит ошибка, то ВП возвратит информацию об ошибке на свой выходной терминал error out и не будет продолжать выполняться. Например, предположим, что возникла в ВП DAQmx Start Timing. Этот ВП прекратит выполнение и передаст информацию об ошибке в ВП DAQmx Start Task. Последний выполняться не будет – он просто передаст ошибку в следующий ВП. Таким образом, информация об ошибке проходит через каждый ВП и поступает в Error Handler для отображения.

Чтобы понять, что происходит при буферизированном сборе данных, рассмотрим его более подробно.

При получении аналоговый сигнал проходит через инструментальный усилитель в АЦП. Затем он поступает в FIFO (от First In First Out – первым поступил – первым выводится) буфер, расположенные в устройстве, который хранит данные до тех пор, пока они не будут переданы из устройства в компьютер. После этого данные поступают из устройства в буфер

ПК по каналу прямого доступа к памяти (Direct Memory Access – DMA) или с использованием запроса прерывания (Interrupt Request – IRQ).

Буфер персонального компьютера представляет собой область памяти, в которой хранятся данные после получения их из устройства. Ввод number of samples per channel ВП DAQmx Timing (или buffer size в ВП DAQmx Configure Input Buffer) задает буфер ПК, который хранит данные, пока ВП DAQmx Read не будет готов считать их. После этого ВП DAQmx Read передает данные в буфер LabVIEW, который затем может быть отображен на лицевой панели. Буфер LabVIEW может передать данные на график осциллограмм, в массив или в виде типа данных «осциллограмма» в зависимости настройки конфигуратора ВП DAQmx Read и способа подключения выходных терминалов ВП DAQmx Read.

3.10. Межбуферная передача данных

Передача данных между буфером ПК и буфером LabVIEW является одной из важнейших операций аналогового ввода. Ввод number of samples per channel ВП DAQmx Timing назначает буфер ПК. При выполнении буферизированного сбора данных он начинается, когда вы вызываете ВП DAQmx Start Task. После начала сбора данных буфер ПК заполняется данными до тех пор, пока не станет полным. Скорость наполнения определяется частотой, которую вы установили в ВП DAQmx Timing. После заполнения буфера ПК ВП DAQmx Read переносит данные из него в буфер LabVIEW. При буферизированном сборе ВП DAQmx Read перемещает все данные за раз.

Задание:

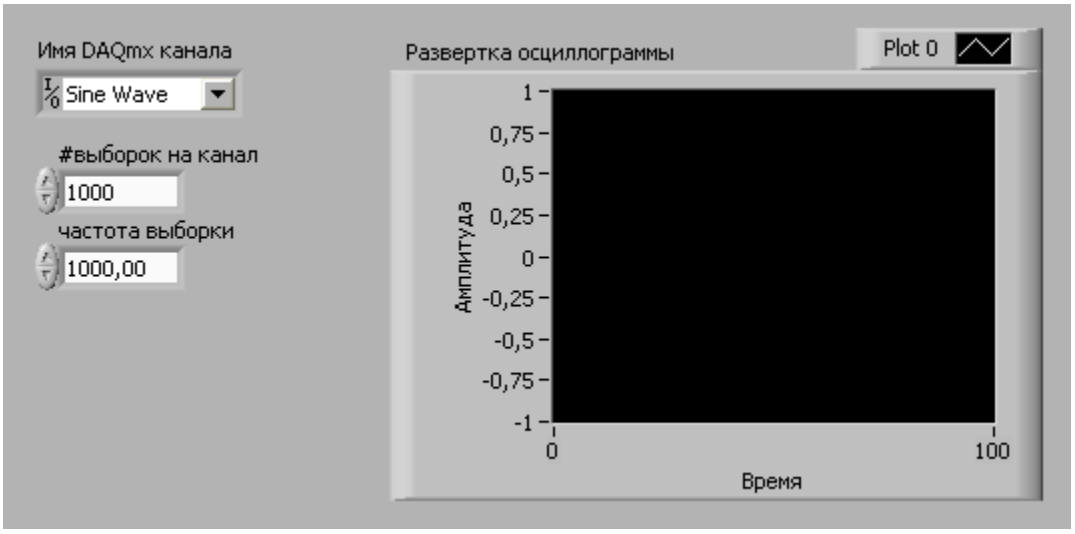
Получить массив данных, используя конфигурацию буферизированного ввода ограниченного количества данных.

При использовании буферизированного ввода ограниченного количества данных LabVIEW задает, сколько точек необходимо получить и с какой частотой. После этого вся забота о тактировании ложится на DAQ устройство. При буферизированном вводе DAQ устройство управляет всеми аспектами сбора данных. В противоположность этому, при сборе данных с программным тактированием за управление сбором отвечает только компьютер, что может быть проблематично, если компьютер вдруг не сможет дать достаточного приоритета процессу сбора данных.

Решение:

Лицевая панель

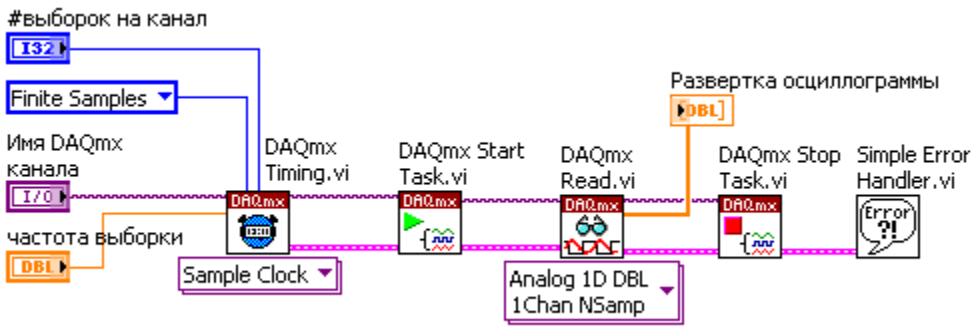
1. Откройте новый ВП и создайте следующую лицевую панель.



Большинство показанных выше элементов управления вы можете создать с блок-диаграммы, вызывая контекстное меню соответствующих терминалов виртуальных приборов и выбирая в них опцию Create»Control (Создать»Элемент управления).

Блок-диаграмма

2. Постройте следующую блок-диаграмму.



3. Переключитесь на лицевую панель и запустите ВП. На графике должен отобразиться сигнал.

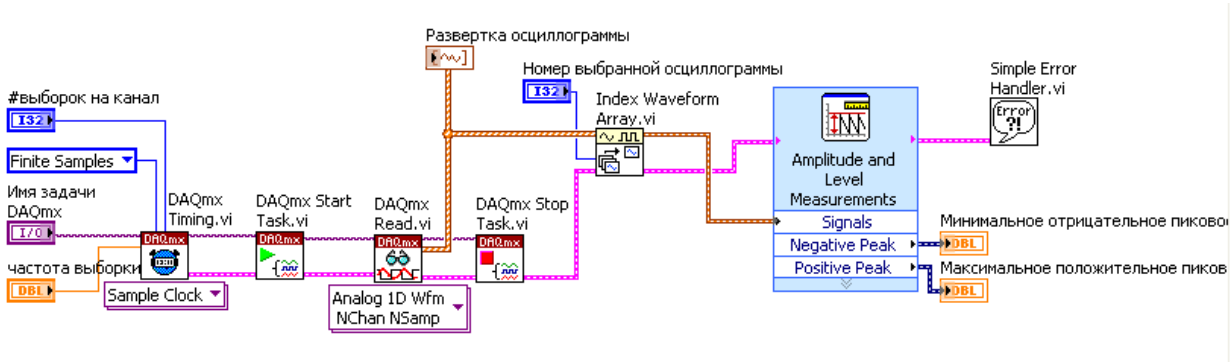
Задание:

Получить осциллограммы, используя буферизированный сбор данных, и проанализировать эти данные для нахождения максимального и минимального значений.

Данный ВП позволит вам найти максимальное и минимальное значения синусоидального сигнала. Эти значения помогут определить, работает ли генератор в пределах заявленных технических характеристик.

Решение:

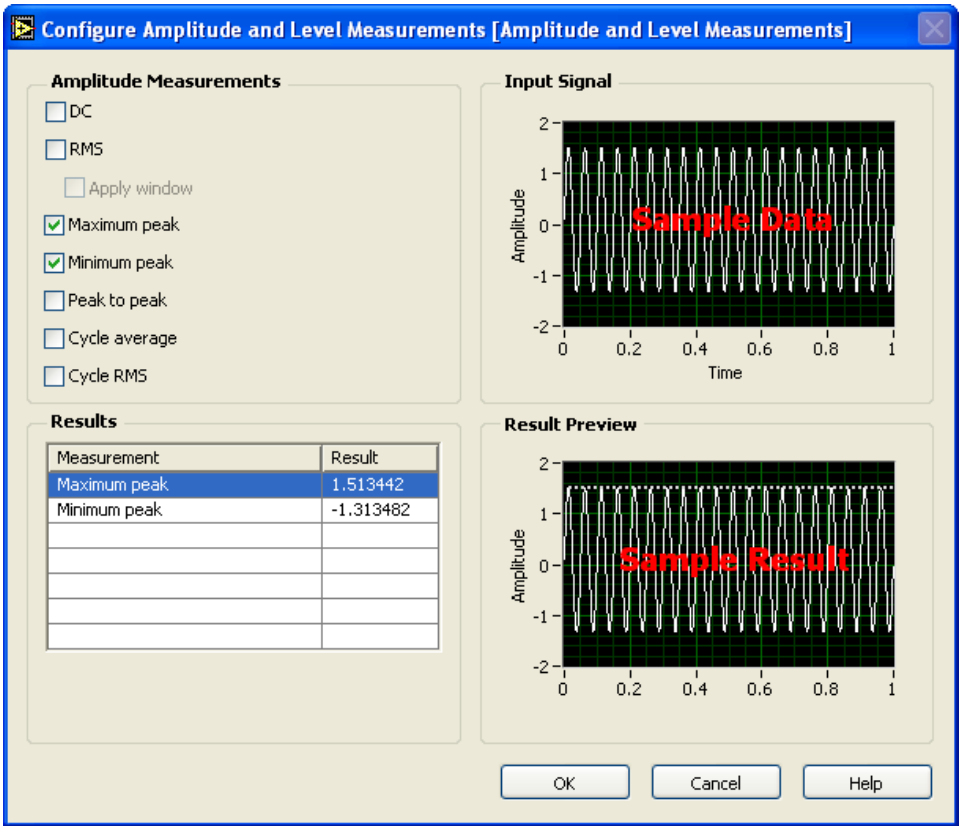
2. Создайте следующую блок-диаграмму.



Экспресс-ВП Amplitude and Level Measurements, расположенный в палитре Functions»Signal Analysis, определяет максимальное и минимальное значения сигнала.

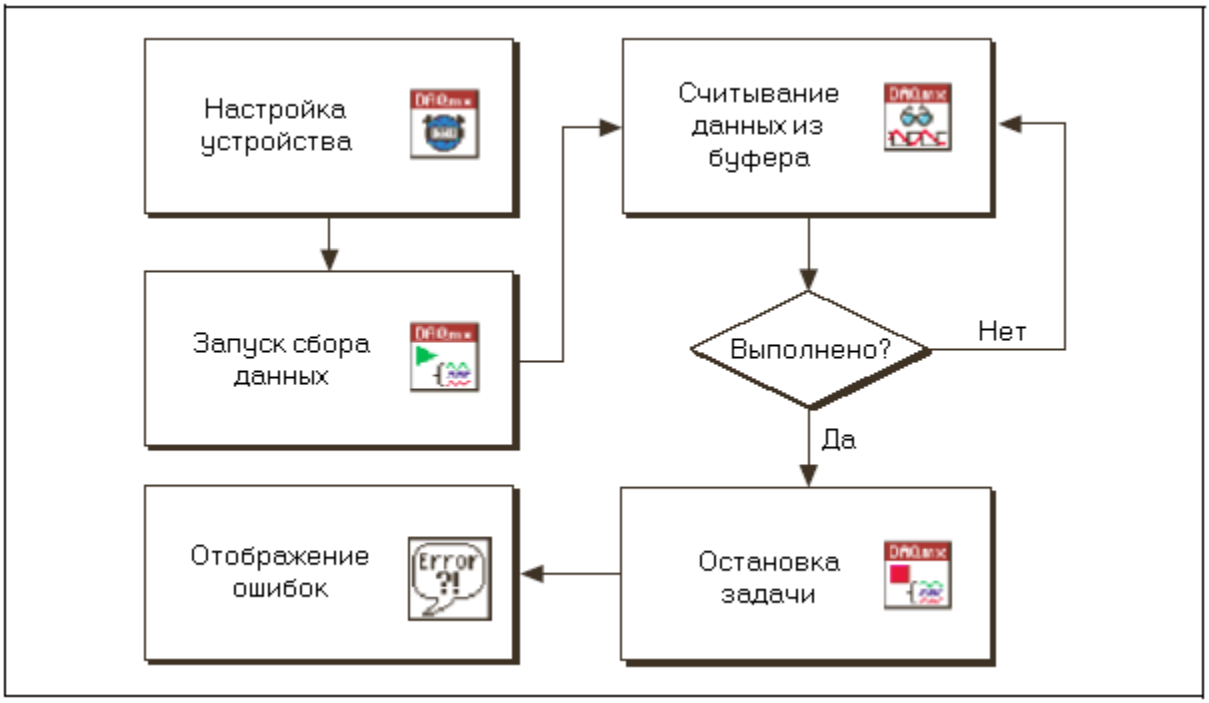
– В диалоговом окне Configure Amplitude and Level Measurements (Настройка измерений амплитуды и уровня) поставьте отметки напротив опций Maximum peak и Minimum peak.

– Нажмите кнопку OK для применения изменений и закрытия диалогового окна.



3.11. Блок-схема непрерывного сбора данных

Основное отличие буферизированного сбора ограниченного количества данных от непрерывного буферизированного сбора данных заключается в количестве точек, которое необходимо получить. В первом случае вы получаете некоторое число точек. Во втором – вы можете собрать неограниченное количество данных. Следующая блок-схема демонстрирует непрерывный буферизированный сбор данных.



Первые три этапа на этой блок-схеме совпадают с оными на блок-схеме буферизированного сбора данных. Они отвечают за настройку DAQ устройства виртуальным прибором DAQmx Timing, за запуск сбора данных ВП DAQmx Start Task и подготовку к считыванию данных ВП DAQmx Read. Поскольку вы собираете данные непрерывно, и считывать данные также необходимо непрерывно. Поэтому поместите ВП DAQmx Read в цикл. Цикл завершит выполняться, если возникнет ошибка, или если вы остановите его с лицевой панели. При каждом выполнении цикла ВП DAQmx Read будет возвращать данные. Когда же выполнение цикла закончится, ВП DAQmx Stop Task остановит задачу и высвободит ресурсы. ВП Simple Error Handler покажет ошибки, возникшие в процессе выполнения.

3.12. Непрерывный буферизированный сбор данных

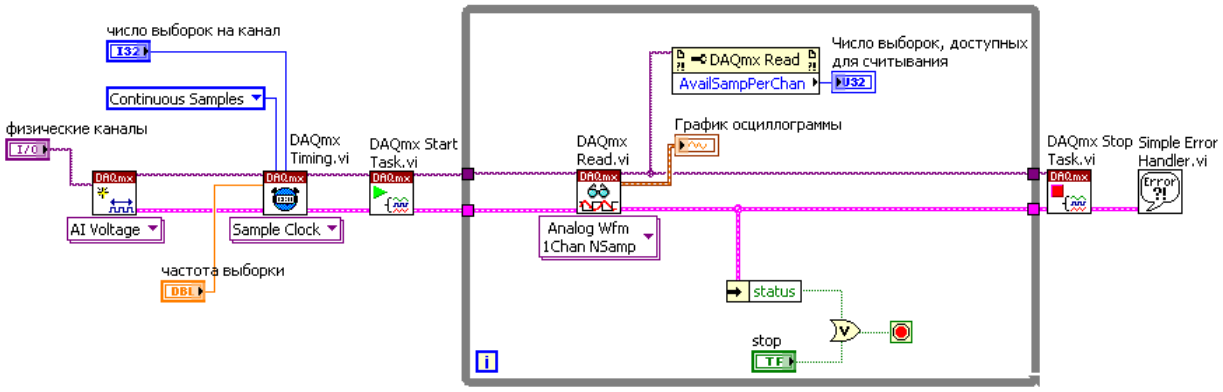
Следующая блок-диаграмма ВП непрерывного буферизированного сбора данных похожа на буферизированный сбор данных со следующими изменениями:

• ВП DAQmx Read находится внутри цикла по условию.

• Ввод число выборок на канал определяется пользователем. При сборе ограниченного числа данных NI-DAQmx автоматически определяет количество выборок для чтения. Если вы

оставите этот ввод неподключенным или выставите значение –1, NI-DAQmx считает полное количество выборок, имеющихся в буфере.

• Отслеживается количество выборок, доступных для считывания (backlog).



Для применения непрерывного буферизированного сбора данных вначале используйте ВП Timing, настраивая тактирование, число выборок на канал для чтения (буфер) и частоту сбора данных. Далее используйте ВП DAQmx Start для запуска сбора данных. Затем ВП DAQmx Read, помещенный в цикл по условию, будет считывать данные из буфера.

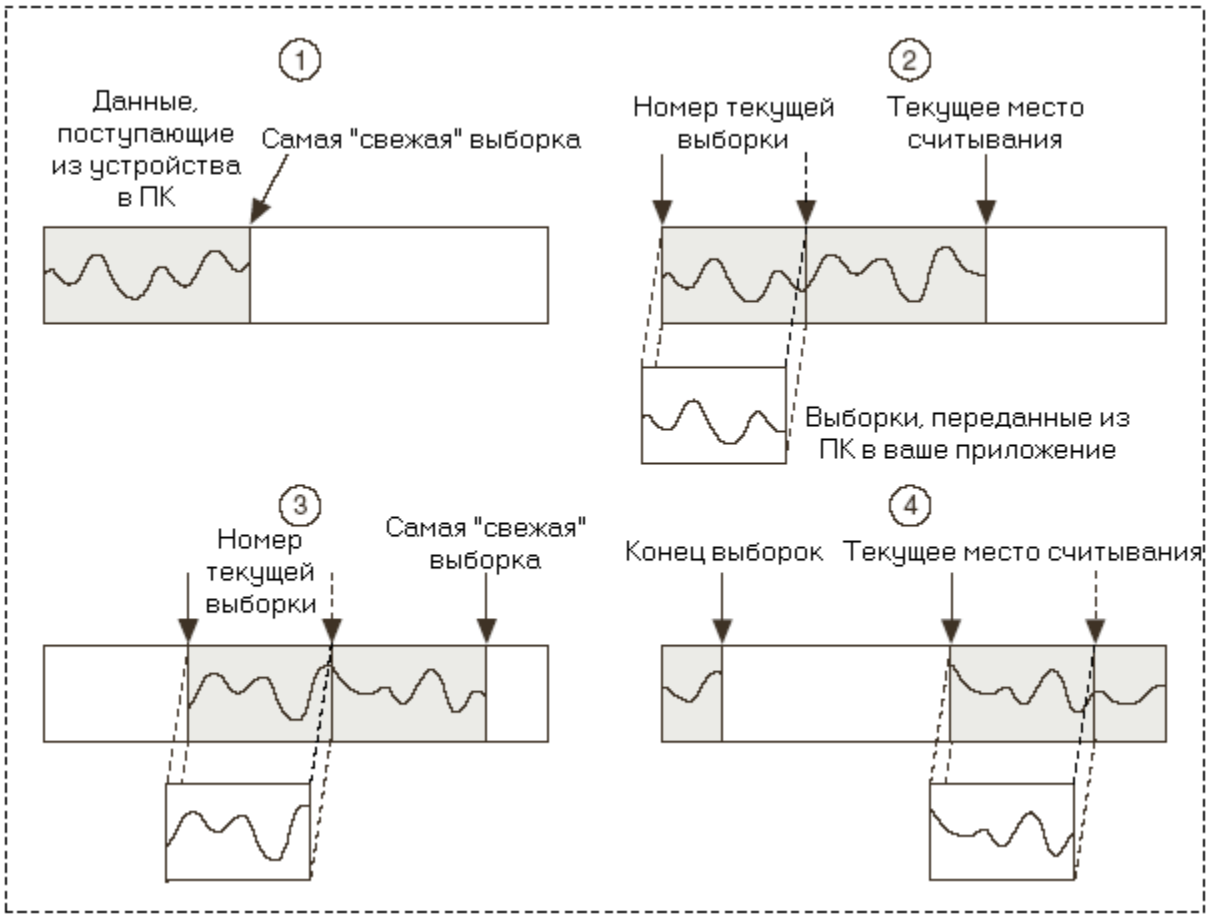
Для предотвращения переполнения буфера число выборок на канал для чтения (number of samples per channel to read) не может быть большим или равным размеру буфера. Обычно при непрерывном сборе данных устанавливают значение ввода number of samples per channel to read равным четверти либо половине размера буфера. Поскольку LabVIEW непрерывно отправляет данные в буфер, важно постоянно следить за числом доступных выборок в буфере, чтобы быть уверенным, что буфер опустошается достаточно быстро.

Если же количество доступных выборок на канал (backlog) постоянно возрастает, то буфер может переполниться, и возникнет ошибка. Цикл по условию, содержащий ВП DAQmx Read, может быть остановлен либо пользователем, нажавшим кнопку на лицевой панели, либо при возникновении ошибки в ВП DAQmx Read, такой как переполнение буфера. После остановки цикла ВП DAQmx Stop Task остановит задачу и высвободит ресурсы, а ВП Simple Error Handler отобразит все возникшие ошибки.

3.13. Циклический буфер

Операция непрерывного буферизированного сбора данных достаточно сложна, поскольку компьютер использует один буфер, а вы получаете большее количество данных, чем может

вместить буфер. Чтобы выполнить это, используйте циклический буфер. Следующий рисунок иллюстрирует работу циклического буфера.



Циклический буфер похож на обычный, только вместо завершения работы при достижении конца циклического буфера, вы начинаете запуск сначала. Буфер ПК назначается при задании значения на входе samples per channel (#выборок на канал) ВП DAQmx Timing. Когда ВП DAQmx Start Task запускает сбор данных, буфер ПК начинает заполняться данными. Процесс получения данных происходит внутри цикла по условию.

Предположим, что вы задали число выборок для чтения (number of samples per channel to read) равным значению между четвертью и половиной размера буфера. Когда число выборок на канал в буфере ПК станет равным количеству выборок для чтения, ВП DAQmx Read переместит это число выборок из буфера ПК в буфер LabVIEW. ВП DAQmx Read

устанавливает флаг (метку), называемый текущее положение выборки, чтобы в дальнейшем продолжить считывание с места, где оно было остановлено.

Между тем, буфер ПК продолжает заполняться данными. Пока это происходит, ВП DAQmx Read продолжает передавать данные из буфера ПК в буфер LabVIEW. Когда метка конца данных достигает предела буфера ПК, то новые данные начинают записываться в начале этого буфера. Разница между меткой конца выборок и текущим положением выборки равна числу доступных выборок на канал (backlog). LabVIEW должна считывать данные достаточно быстро, чтобы избежать случая, когда метка конца данных догонит текущее положение выборки. В противном случае, новые данные запишутся поверх старых, и LabVIEW выдаст ошибку.

3.14. Ошибка наложения записей

Наиболее распространенная ошибка, с которой вы можете столкнуться при выполнении непрерывного буферизированного сбора данных, это ошибка наложения записей (overwrite error). Эта ошибка возникнет, когда метка конца выборок догонит текущее положение выборки, и данные будут накладываться друг на друга. Это происходит, если LabVIEW не достаточно быстро считывает данные из буфера ПК. Существует несколько способов, помогающих избежать этой ошибки, но не все из них можно применять в конкретной ситуации, так как некоторые срабатывают лучше, чем другие.

• Увеличьте число выборок на канал (размер буфера) в ВП DAQmx Timing. Эта мера не снимет проблему, если вы не опустошаете буфер с достаточной скоростью. Запомните норму числа выборок на канал для считывания: от четверти до половины размера буфера. Если увеличение размера буфера приведет к выполнению указанного условия, то это исключит ошибку.

• Опустошайте буфер быстрее, увеличивая число выборок на канал для считывания (number of samples per channel to read). Не устанавливайте этого числа слишком большим, поскольку при этом возникнет пауза в ВП DAQmx Read до тех пор, пока количество выборок в буфере не достигнет числа выборок для считывания. Время, затраченное на ожидание заполнения буфера выборками, могло быть потрачено на опустошение буфера.

• Уменьшите частоту выборок на канал в ВП DAQmx Timing. Эта мера замедлит скорость, с которой данные будут отправляться в буфер. Однако это не всегда может быть доступно, если вам необходима определенная частота дискретизации.

• Избегайте замедления выполнения цикла из-за ненужного анализа данных внутри него.

3.15. Ошибка переполнения

Другая ошибка, которая может возникнуть при непрерывном буферизированном сборе данных, вызывается переполнением FIFO буфера DAQ устройства. Эта ошибка не настолько распространена, как ошибка наложения записи, но ее и не так легко исправить. Она появляется, когда FIFO буфер не опустошается достаточно быстро. При передаче данных в буфер компьютера состояние FIFO буфера зависит или от DMA или от IRQ, поэтому, когда FIFO буфер опустошается недостаточно быстро, есть всего лишь несколько способов предотвратить ошибку.

• Убедитесь, что, если DMA доступен, вы используете канал DMA для передачи данных. Прямой доступ к памяти (DMA) работает быстрее, чем запрос прерывания (IRQ), и это может значительно увеличить быстродействие. За большей информацией об использовании DMA обращайтесь как справке NI-DAQmx Help и узлу свойств DAQmx канала (свойство Data Transfer Mechanism (Механизм передачи данных)).

• Уменьшите частоту выборки на канал в ВП DAQmx Timing.

• Приобретите устройство с большим FIFO буфером. Однако этот способ может только оттянуть появление проблемы вместо ее решения.

• Приобретите компьютер с более быстрой шиной, чтобы ускорить передачу данных из FIFO буфера в буфер компьютера. Переполнение обусловлено тем, что система не забирает данные из устройства с надлежащей скоростью. Поэтому компьютер с более быстрой шиной может переносить данные из FIFO буфера быстрее.

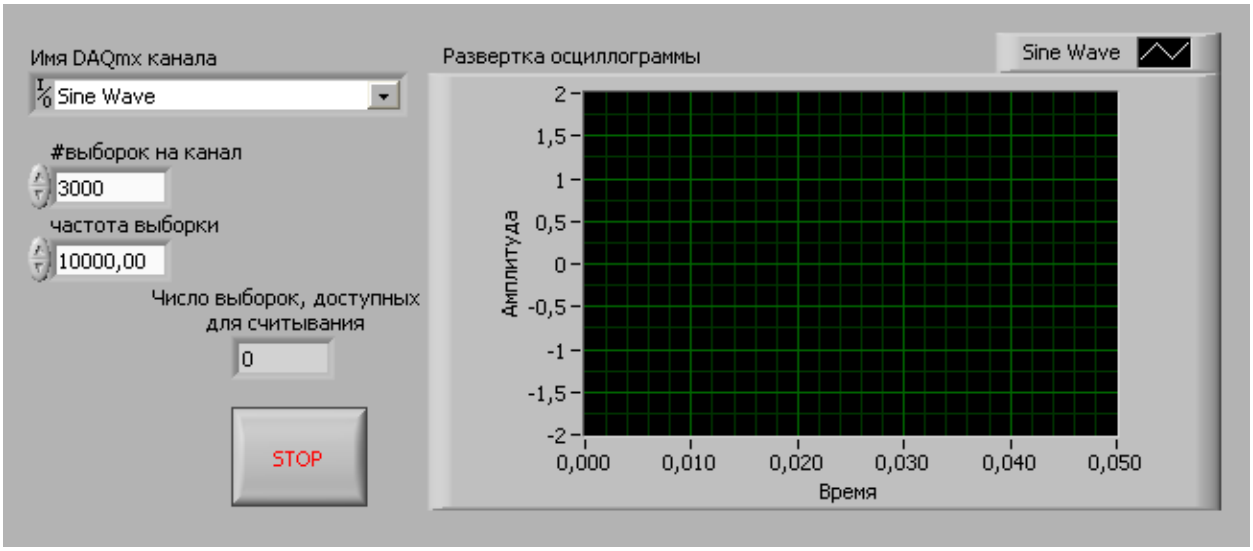
Задание:

Непрерывно получайте данные с помощью DAQ устройства и записывайте их в файл.

Решение

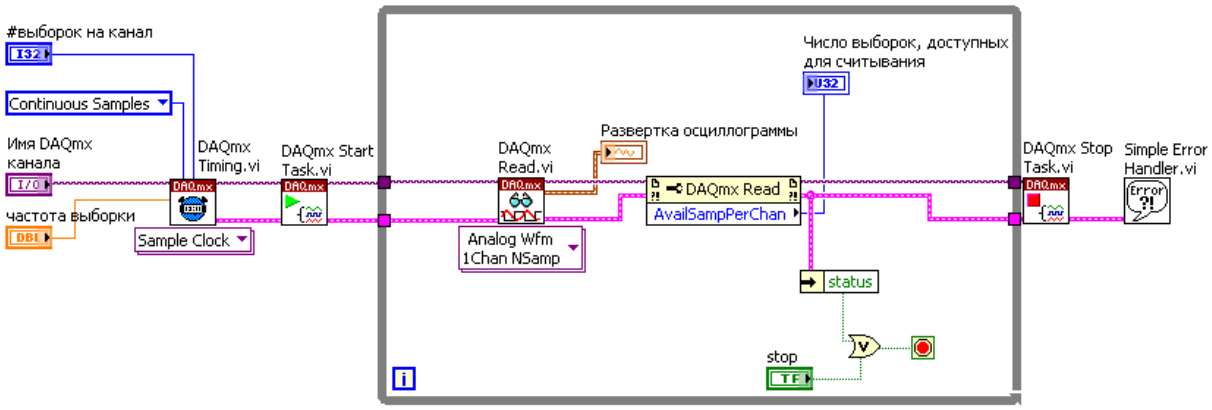
Лицевая панель

1. Откройте новый ВП и создайте следующую лицевую панель.



Блок-диаграмма

2. Постройте следующую блок-диаграмму.



Узел свойств DAQmx Read Property Node, расположенный в палитре Functions»All Functions»NI Measurements»DAQmx - Data Acquisition, используется для получения дополнительной информации о состоянии буфера. Вызовите контекстное меню узла и выберите опцию Properties»Status»Available Samples per Channel. Это свойство отслеживает количество непрочитанных выборок на канал в буфере.

3. Переключитесь на лицевую панель. Запустите ВП. Наблюдайте за данными, которые отображает развертка. Данные заполняют буфер фиксированного размера в памяти, затем переписывают значения, заполняя буфер с начала.

4. Понаблюдайте за величиной Числа выборок, доступных для считывания при уменьшении частоты или количества выборок на канал. Число доступных выборок на канал определяется как количество выборок на канал, принятых в буфер, но еще не считанных. Эта величина является мерой, насколько успешно вы справляетесь с непрерывным сбором данных. Если непрочитанные данные (backlog) непрерывно возрастают, то это признак недостаточной скорости считывания данных из буфера, что, в конце концов, приведет к их потере. Если это произойдет, ВП DAQmx Read возвратит ошибку.

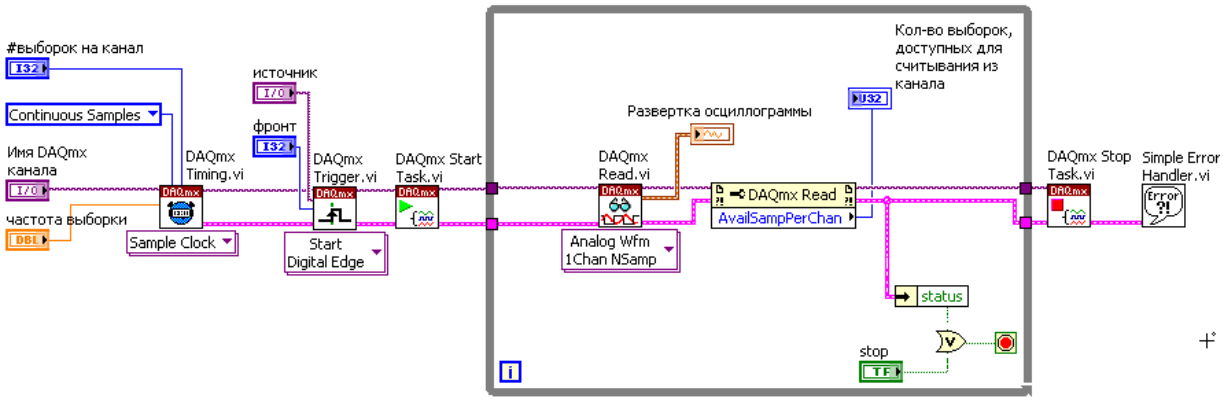
Задание:

Использовать цифровой триггер для запуска операции непрерывного сбора данных.

Решение

Блок-диаграмма

Создайте следующую блок-диаграмму:



ВП DAQmx Trigger, расположенный в палитре Functions»All Functions»NI Measurements»DAQmx – Data Acquisition, настраивает триггер. Вызовите контекстные меню вводов edge и source, выберите Create»Control, и введите следующие значения:

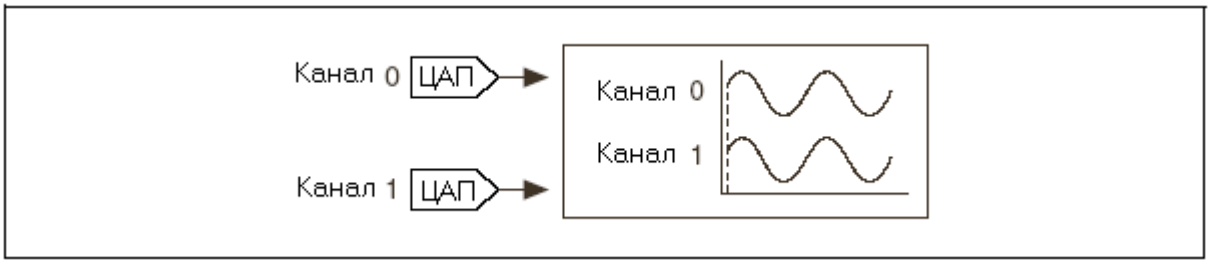
• Источник: /DevX/PFI0, где X соответствует номеру вашего DAQ устройства в MAX.

• Фронт: Rising

3.16. Архитектура вывода аналоговых сигналов

Большинство устройств M-серии имеют цифро-аналоговый преобразователь (ЦАП) для каждого из каналов аналогового вывода. Цифро-аналоговые преобразователи обновляются одновременно, так что для операций вывода необходим лишь один тактовый генератор.

Тактовый генератор аналогового вывода создает сигнал регенерации (update clock). Выходные сигналы каналов аналогового вывода синхронизированы таким же образом, как и каналы аналогового ввода при одновременной выборке сигналов.



ЦАП имеет рабочий диапазон, определяемый опорным напряжением. В качестве опорного напряжения может быть внутренний или внешний сигнал. Внутреннее опорное напряжение равно + 10 вольтам. Рабочий диапазон ЦАП можно установить однополярным либо биполярным.

3.17. Биполярный рабочий диапазон

Биполярный сигнал имеет диапазон, содержащий положительные и отрицательные значения. Если вы установите устройство в режим биполярного сигнала, то рабочий диапазон ЦАП определяется следующим образом:

• Максимальное напряжение = +Vref

• Минимальное напряжение = –Vref

Например, если вы используете внутреннее опорное напряжение величиной +10 В, то рабочий диапазон ЦАП будет установлен от –10 до +10 В. Однако если сигнал будет изменяться от –5 до +5 В, то вы не сможете использовать максимальное разрешение ЦАП. Чтобы разрешение довести до максимума, можно установить внутреннее опорное напряжение величиной +5 В. Теперь рабочий диапазон ЦАП будет равен от –5 до +5 В, совпадая с диапазоном изменения сигнала, и вы будете целиком использовать разрешающую способность ЦАП для генерации сигнала.

3.18. Однополярный рабочий диапазон

Однополярный сигнал имеет диапазон изменений, содержащий только положительные значения. Если вы установите устройство в режим однополярного сигнала, то рабочий диапазон ЦАП определяется следующим образом:

• Максимальное напряжение = +Vref

• Минимальное напряжение = 0 В

Например, если вы используете внутреннее опорное напряжение величиной +10 В, то рабочий диапазон ЦАП будет установлен от 0 до +10 В. Однако если сигнал будет изменяться от 0 до +5 В, то вы не сможете использовать максимальное разрешение ЦАП. Чтобы разрешение довести до максимума, можно установить внутреннее опорное напряжение величиной +5 В. Теперь рабочий диапазон ЦАП будет равен от 0 до +5 В, совпадая с диапазоном изменения сигнала, и вы будете целиком использовать разрешающую способность ЦАП для генерации сигнала.

3.19. Использование ВП Запись DAQmx

ВП DAQmx Write, расположенный в палитре DAQmx - Data Acquisition, записывает выборки в определенные вами задачу или каналы. Конфигуратор этого полиморфного ВП задает формат выборок для записи, записывать ли одну или несколько выборок, и записывать ли в один или несколько каналов. На данном занятии будут описаны настройки аналогового вывода ВП DAQmx Write. Для выбора настроек этого ВП используйте выпадающее меню.

Существует четыре меню выбора параметров ВП DAQmx Write. Первое меню позволяет выбрать тип выходного сигнала:

• Аналоговый

• Цифровой

• Необработанные данные

Во втором меню можно задать число каналов для записи, а также запись данных непосредственно в устройство без предварительного масштабирования (unscaled). Третье меню позволяет выбрать генерацию одной (single sample) или нескольких выборок (multiple samples). При выборе одноточечной генерации данных используйте четвертое меню для задания формата записываемых данных: осциллограмма (waveform) или число с плавающей запятой удвоенной точности (DBL). При выборе генерации нескольких точек используйте это

меню для записи данных в виде осциллограммы или массива числовых значений удвоенной точности.

При генерации одной выборки терминал auto start по умолчанию имеет значение истина. Это происходит потому, что модель состояния задачи может полностью неявным образом контролировать генерацию отдельной выборки. Однако при выводе нескольких точек данных терминал auto start по умолчанию имеет значение ложь. Это обусловлено тем, что теперь необходима настройка дополнительных параметров тактирования при помощи ВП DAQmx Timing, и в этом случае понадобится использование ВП DAQmx Start Task и DAQmx Stop Task. За большей информацией о модели состояния DAQmx задачи обращайтесь ко второму занятию, Оборудование и программное обеспечение систем сбора данных.

3.20. Генерация одной выборки

В случаях, когда уровень сигнала гораздо более важен, чем скорость генерации, выходной сигнал представляет собой только одну выборку. Таким образом, если вам нужно сгенерировать постоянный сигнал генерируйте одну выборку за раз. Управлять моментом генерации можно при помощи программного тактирования, а также тактирования с использованием аппаратных средств.

• Программно управляемое тактирование – Скорость, с которой генерируются выборки, определяется программой и операционной системой, но не устройством сбора данных. Поскольку в этом случае генерация всецело зависит от ресурсов вашей операционной системы, любые прерывания системы могут влиять на генерацию.

• Аппаратно управляемое тактирование – ТТЛ сигнал, такой как тактовые импульсы генератора DAQ устройства, управляет скоростью генерации. В этом случае генерация может происходить с гораздо большей скоростью и точностью, чем при программно управляемом тактировании. Не все устройства поддерживают такой режим тактирования. Чтобы узнать, поддерживает ли ваше устройство аппаратно управляемое тактирование, посмотрите документацию к устройству.

3.21. Настройки тактирования при генерации аналогового сигнала

Для информирования DAQmx об использовании программно или аппаратно управляемого тактирования используйте ВП DAQmx Timing и/или узел свойств DAQmx Sample Timing. Выбирая настройку конфигуратора Sample Clock ВП Timing или устанавливая для узла

свойств Sample Timing Type значение Sample Clock, вы говорите DAQmx, что хотите использовать генератор выборки (Sample Clock) вашего DAQ устройства для управления генерацией. При программно управляемой генерации установите для узла свойств Sample Timing Type значение On Demand. Если вы не задали тип тактирования при помощи ВП DAQmx Timing или узла свойств DAQmx Sample Timing Type, то будет использоваться программно управляемая генерация.

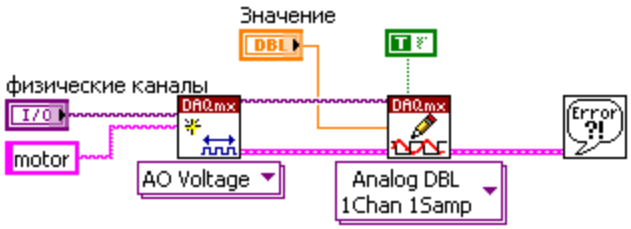
Кроме этого, ВП DAQmx Timing содержит опцию Use Waveform в меню конфигуратора. В этом случае для определения частоты выборки используется компонента осциллограммы dt, которая равна времени (в секундах) между выборками. С такой настройкой будет осуществляться аппаратно управляемая генерация аналогового сигнала. На самом деле, опция Use Waveform ВП DAQmx Timing не заставляет выводить значения осциллограммы, она лишь использует осциллограмму для настройки тактирования. Для того чтобы сгенерировать сигнал, присоедините осциллограмму к ВП DAQmx Write.

Задание:

Создать ВП, генерирующий сигнал величиной + 5 вольт.

Решение:

Откройте новый ВП и постройте следующую блок-диаграмму. Для создания элементов управления и констант щелкните правой кнопкой на входах и выберите Create»Control или Create»Constant в контекстном меню.



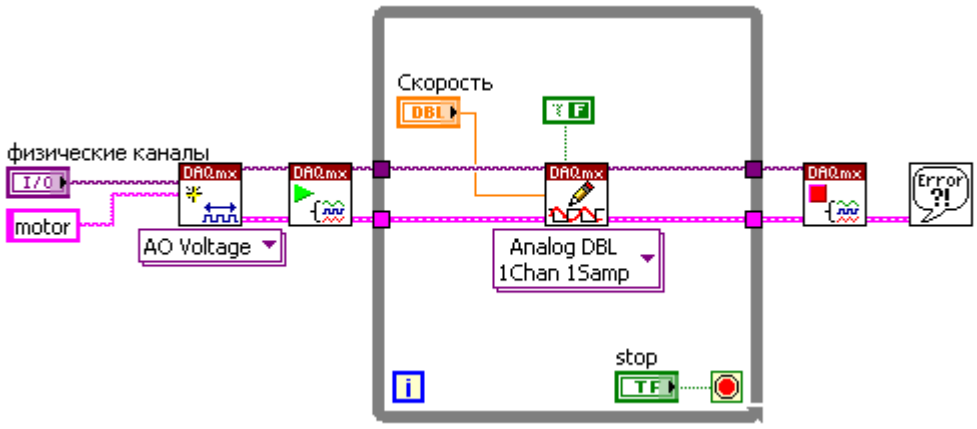
Задание:

Создать ВП для генерации переменного напряжения.

Решение:

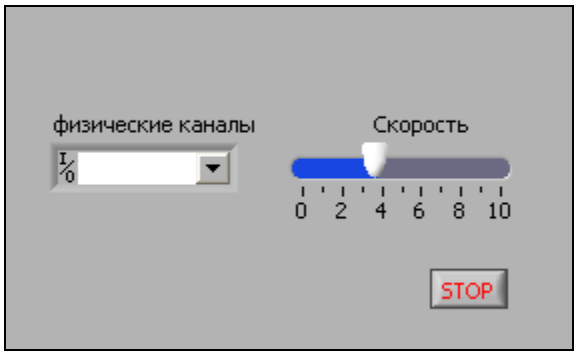
Блок-диаграмма

Измените блок-диаграмму из предыдущего задания, как показано на следующем рисунке.



Поскольку вы используете ВП Start Task, вы должны изменить значение входа auto start на Ложь.

Измените лицевую панель как показано на следующем рисунке.



3.22. ВП буферизированного аналогового вывода

Для генерации нескольких выборок аналогового сигнала выберите в выпадающем меню конфигуратора ВП DAQmx Write опцию multiple samples. Такой тип генерации используется для вывода сигнала, изменяющегося во времени, такого как синусоидальная волна. Многоточечная генерация известна также как буферизированный аналоговый вывод.

Буферизированный аналоговый вывод может осуществлять непрерывную генерацию данных и генерацию данных конечной длительности. В обоих случаях процесс буферизации содержит следующие два этапа:

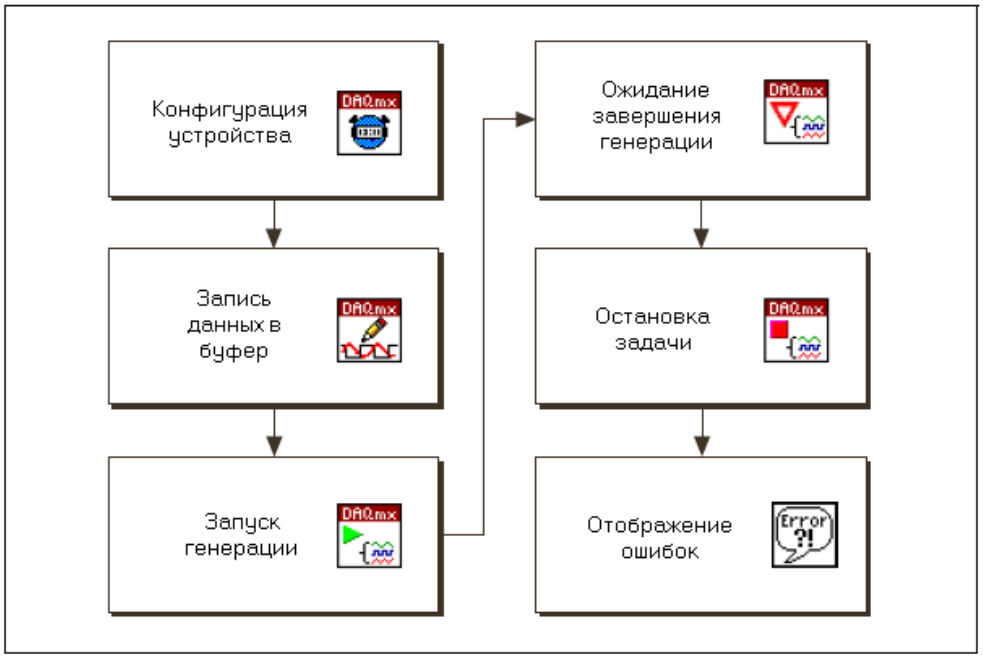
1. Запись выборок в буфер. Точки данных забираются из LabVIEW и помещаются в промежуточный буфер в памяти перед отправкой их в DAQ устройство. Буферизированная генерация напоминает отправление всего электронного письма за раз вместо отправки сообщения по одному слову.

2. Перемещение выборок из буфера в устройство сбора данных. Скорость перемещения зависит от заданных параметров тактирования. Как и при одноточечной генерации, здесь используется программное или аппаратное тактирование.

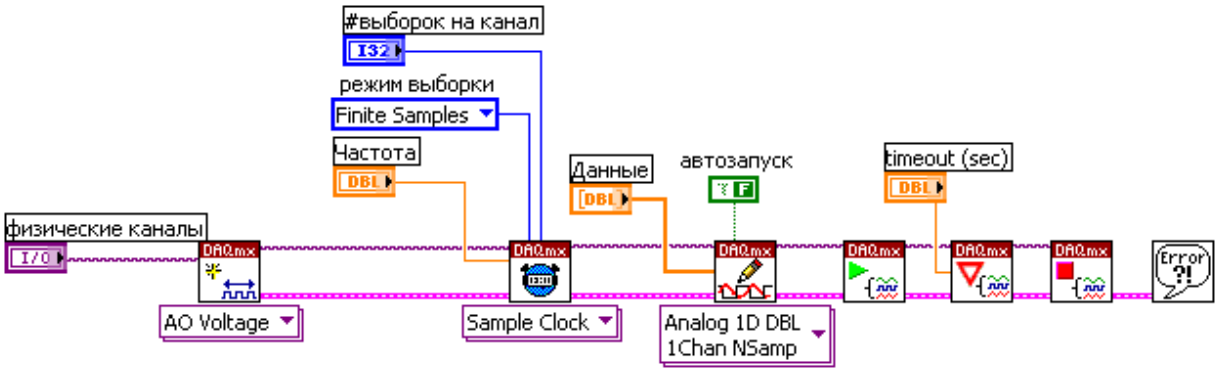
При аппаратном тактировании тактовый сигнал, называемый сигналом регенерации, управляет скоростью генерации. Аппаратный тактовый генератор может работать гораздо быстрее, чем программные циклы, так что вы можете создавать сигналы различной формы в большем диапазоне частот. Кроме этого, тактовый генератор более точен по сравнению с программными циклами. Последние подвергаются влиянию различных действий с операционной системой, таких как открытие других приложений.

3.23. Буферизированная генерация сигналов конечной длительности

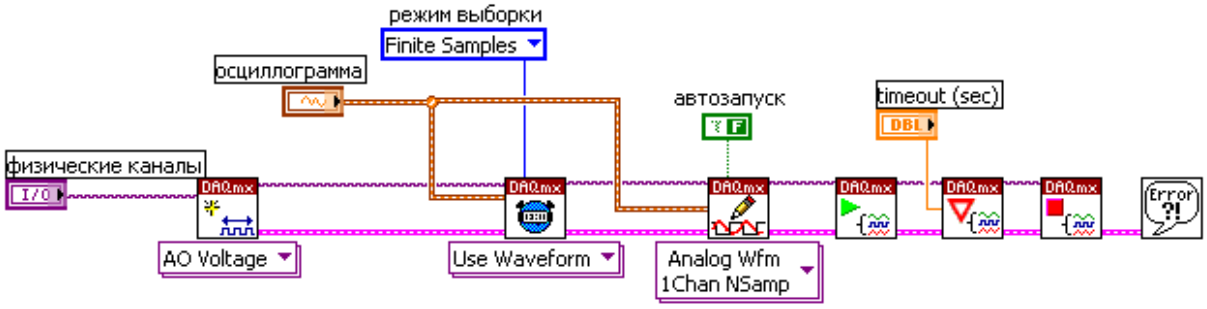
Следующая иллюстрация показывает блок-схему буферизированной генерации сигналов.



Следующий рисунок показывает пример типичной генерации сигналов конечной длительности с использованием тактовых импульсов выборки (Sample Clock) и массива чисел удвоенной точности для выходных данных.



Вы можете использовать также тип данных «осциллограмма» для задания временных параметров генерации и выборок данных. Эта ситуация показана на следующем рисунке.



Существует небольшая разница между двумя указанными типами генерации сигналов конечной длительности – использование массива чисел удвоенной точности совместно с тактовым генератором выборки и использование типа данных «осциллограмма» для задания режима работы тактового генератора и выборок. Опции конфигуратора ВП DAQmx Timing изменяются вместе с данными, подключаемыми к терминалу data ВП DAQmx Write. Оба типа генерации имеют, в общем, одинаковую структуру, к описанию которой мы сейчас и переходим.

Для программного управления созданием виртуального канала аналогового вывода можно использовать ВП Create Virtual Channel. Если вы уже создали виртуальный канал или задачу с

использованием Помощника по сбору данных в MAX, вы можете пропустить этот ВП и присоединить имя канала/задачи к следующему ВП – DAQmx Timing.

ВП DAQmx Timing имеет две опции конфигуратора, которые можно использовать в операциях аналогового вывода, – тактовые импульсы выборки (Sample Clock) и использовать осциллограмму (Use Waveform). Поскольку мы генерируем конечное число выборок, установите для опции sample mode значение Finite Samples в обоих случаях. При использовании тактовых импульсов выборки (Sample Clock) мы должны также задать частоту регенерации rate и число выборок number of samples. Количество выборок определяет размер буфера. При использовании же опции Use Waveform просто присоедините осциллограмму к терминалу waveform. В этом случае частота тактового генератора выборки и число выборок (размер буфера) будут определяться на основе данных, содержащихся в осциллограмме.

ВП DAQmx Write фактически отправляет данные в буфер персонального компьютера. Вы можете выбрать генерацию в виде осциллограммы либо массива чисел удвоенной точности. При выборе опции Use Waveform в конфигураторе ВП DAQmx Timing выберите генерацию в виде осциллограммы в выпадающем меню ВП DAQmx Write. Присоедините ту же осциллограмму, которую использовали для установки тактирования, к терминалу data ВП DAQmx Write. При использовании встроенного генератора выборки (Sample Clock) для тактирования, выберите в выпадающем меню ВП DAQmx Write выход в виде массива чисел удвоенной точности. В этом случае, присоедините массив чисел удвоенной точности, приготовленный для генерации, к терминалу data ВП DAQmx Write.

При генерации за раз нескольких выборок параметр auto start по умолчанию имеет значение Ложь. Поскольку мы будем явно запускать задачу, ожидать ее завершения и затем останавливать задачу, то оставим значение auto start в состоянии Ложь.

ВП DAQmx Start запускает генерацию. ВП DAQmx Wait Until Done ожидает завершения задачи, иначе возникнет пауза. В любом случае управление переходит затем к ВП DAQmx Stop Task, который останавливает задачу. И, как обычно при программировании на LabVIEW в течение данного курса, все ВП соединяет кластер ошибок, поэтому в случае возникновения ошибки возникнет сообщение.

3.24. ВП DAQmx Reset

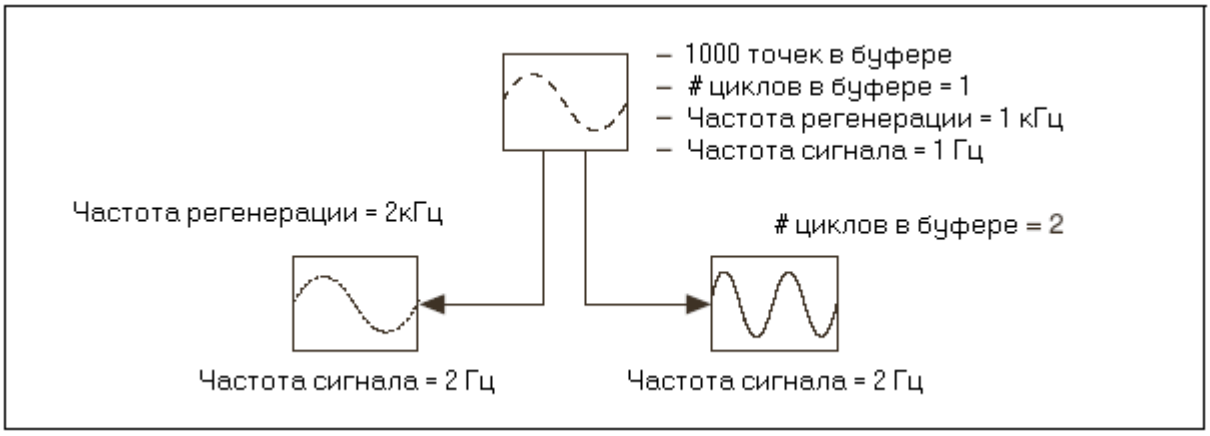
В операциях аналогового вывода при записи значения в выходной канал он продолжает выдавать это напряжение до тех пор, пока новое значение не будет записано в этот канал,

либо устройство не будет приведено в исходное состояние с помощью ВП DAQmx Reset (расположен в палитре DAQmx - Data Acquisition»DAQmx Device Configuration), либо устройство не будет выключено совсем.

Предположим, что вы записываете синусоидальный сигнал в канал аналогового вывода, и последнее значение, сохраненное в буфере равно семи. Вы генерируете осциллограмму целиком, и после того, как генерация завершилась, канал аналогового вывода продолжает выдавать напряжение в семь вольт. Вместо постоянного приведения устройства в исходное состояние, проще записать нулевое значение в канал после окончания генерации. С этой целью можно использовать ВП AO Write One Update, расположенный в палитре Utility.

3.25. Частота генерируемой осциллограммы

Частота генерируемой осциллограммы зависит от частоты регенерации (update rate) и числа циклов осциллограммы, находящихся в буфере. Эта ситуация показана на следующем рисунке.



Частоту выходного сигнала можно вычислить по следующей формуле:

Частота сигнала = [(циклы)□ □ (частота регенерации)]/(коляво точек в буфере)

Следующий пример иллюстрирует, как частота регенерации (обновления) и число циклов осциллограммы в буфере влияют на частоту сигнала. Предположим, что у вас имеется 1000 точек в буфере, которые составляют один период осциллограммы. Если вы производите генерацию сигнала с частотой обновления 1 кГц, то частота сигнала определяется как

[(1 цикл) □ (1000 точек в секунду)]/(1000 точек) = 1 Гц

Если вы удвоите частоту регенерации, а все остальные параметры оставите прежними, то частота сигнала изменится:

[(1 цикл) □ (2000 точек в секунду)]/(1000 точек) = 2 Гц

Если вы удвоите число циклов в буфере, а все остальные параметры оставите прежними, то частота сигнала будет равна:

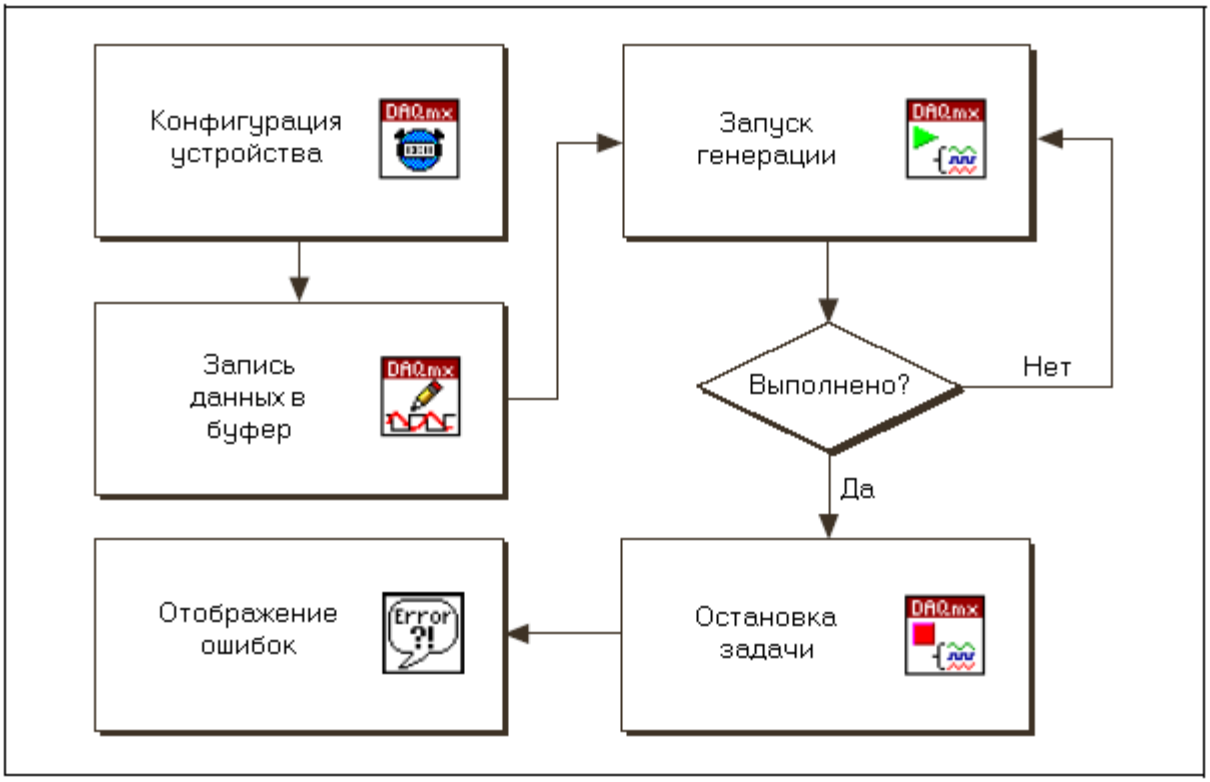
[(2 цикла) □ (1000 точек в секунду)]/(1000 точек) = 2 Гц

Таким образом, при удвоении частоты регенерации или числа циклов вы удваиваете частоту выходной осциллограммы.

3.26. Непрерывная буферизированная генерация сигналов

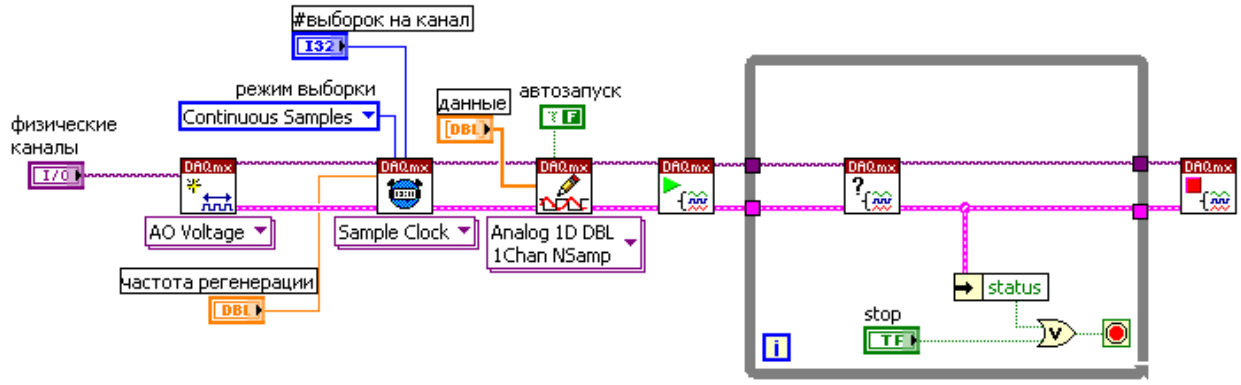
Главная разница между генерацией сигналов конечной длительности и непрерывной буферизированной генерацией заключается в количестве генерируемых точек данных. При буферизированной генерации сигналов конечной длительности вы генерируете данные, находящиеся в буфере, конечное число раз. При непрерывной буферизированной генерации вы можете генерировать данные неограниченное количество раз.

Следующий рисунок похож на буферизированную генерацию сигналов, за исключением некоторых отличий:



• ВП DAQmx Timing установлен в режим непрерывной выборки Continuous Samples.

• В цикле по условию вместо ВП DAQmx Wait Until Done используется ВП DAQmx Is Task Done.



Настройка виртуального канала и параметров тактирования начинается с использования ВП Create Virtual Channel и ВП Timing. Затем происходит запись в буфер при помощи ВП DAQmx Write, и ВП DAQmx Start Task начинает выполнение задачи.

Цикл по условию совместно с ВП DAQmx Is Task Done используется для опроса задачи на предмет окончания ее выполнения. Генерация будет прекращена, либо когда пользователь нажмет кнопку стоп, либо когда возникнет ошибка. После того, как данные в буфере будут целиком обработаны, они будут непрерывно регенерироваться. После остановки цикла по условию ВП DAQmx Stop Task прекратит выполнение задачи и, если ошибки имели место, появится сообщение об ошибках.

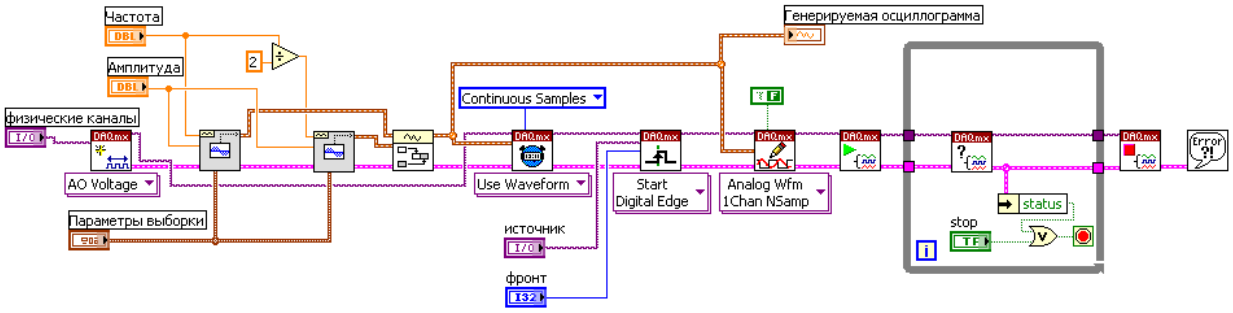
На каждой итерации цикла вы можете записывать новые данные в буфер. Для этого необходимо поместить ВП Write внутрь цикла и присоединить новые данные к терминалу данных этого ВП. Используя узел свойств Write property node, выберите свойство Regeneration Mode и установите его в состояние don’t Allow Regeneration. Вы должны быть уверены, что достаточно быстро генерируете новые данные во избежание выдачи из буфера старых данных. Эта ситуация напоминает операцию непрерывного буферизированного сбора данных. Тогда вам нужно было быть уверенным, что вы считываете данные из буфера достаточно быстро, прежде чем они будут переписаны новой порцией данных.

Задание:

Создать ВП для генерации аналогового сигнала с использованием цифрового триггера.

Решение:

Создайте блок-диаграмму, как показано на следующей иллюстрации.



ВП DAQmx Trigger, расположенный в палитре Functions»All Functions»NI Measurements»DAQmx - Data Acquisition, настраивает триггер для задачи аналогового вывода.

3.27. Синхронизация

На данном занятии описываются явное управление последовательностью выполнения задачи, синхронизация внутри устройства сбора данных и синхронизация нескольких DAQ устройств.

3.28. Явное управление последовательностью выполнения задачи

NI-DAQmx использует модель состояния для управления распределением ресурсов и потоком выполнения задач. Эта модель состояния называется моделью состояния задачи. Модель состояния задачи очень гибка в использовании, и вы можете выбрать взаимодействие с моделью состояния задачи в такой степени, в какой это требуется вашему приложению. Вызов виртуальных приборов DAQmx Start, DAQmx Stop и DAQmx Control Task переводит задачу из одного состояния в другое. Вы можете осуществлять явный переход в каждой из задач, используя ВП DAQmx Control Task, или же позволить NI-DAQmx совершать переход между состояниями неявно. Модель состояния задачи состоит из пяти состояний – Непроверенного, Проверенного, Зарезервированного, Назначенного и Выполнения.

• Unverified (Непроверенное) – Когда задача создана или загружена явно или неявно, то по умолчанию состоянием будет непроверенное. В этом состоянии вы настраиваете тактирование, триггеры и свойства канала в задаче.

• Verified (Проверенное) – При переходе задачи из непроверенного состояния в проверенное проверяются правильность настройки тактирования, триггеров и свойств канала. Если все настройки правильны, задача успешно подтверждается и переходит в проверенное состояние. В противном случае она остается в непроверенном состоянии. Для явного задания перехода вызовите ВП DAQmx Control Task с вводом action (действие) в режиме проверки (verify).

• Reserved (Зарезервированное) – Ресурсы, которые использует задача для совершения определенной операции, выделяются только тогда, когда задача переходит из проверенного состояния в зарезервированное. Этими ресурсами могут быть тактовые генераторы, физические каналы устройства или буферная память в компьютере. Резервирование этих ресурсов предотвращает их использование другими задачами, что могло вызвать сбой либо конфликты при выполнении первой задачи. Если задача может получить доступ ко всем необходимым ресурсам, она переходит в зарезервированное состояние. В противном случае она остается в проверенном состоянии. Для явного задания перехода вызовите ВП DAQmx Control Task с вводом action (действие) в режиме резервировать (reserve).

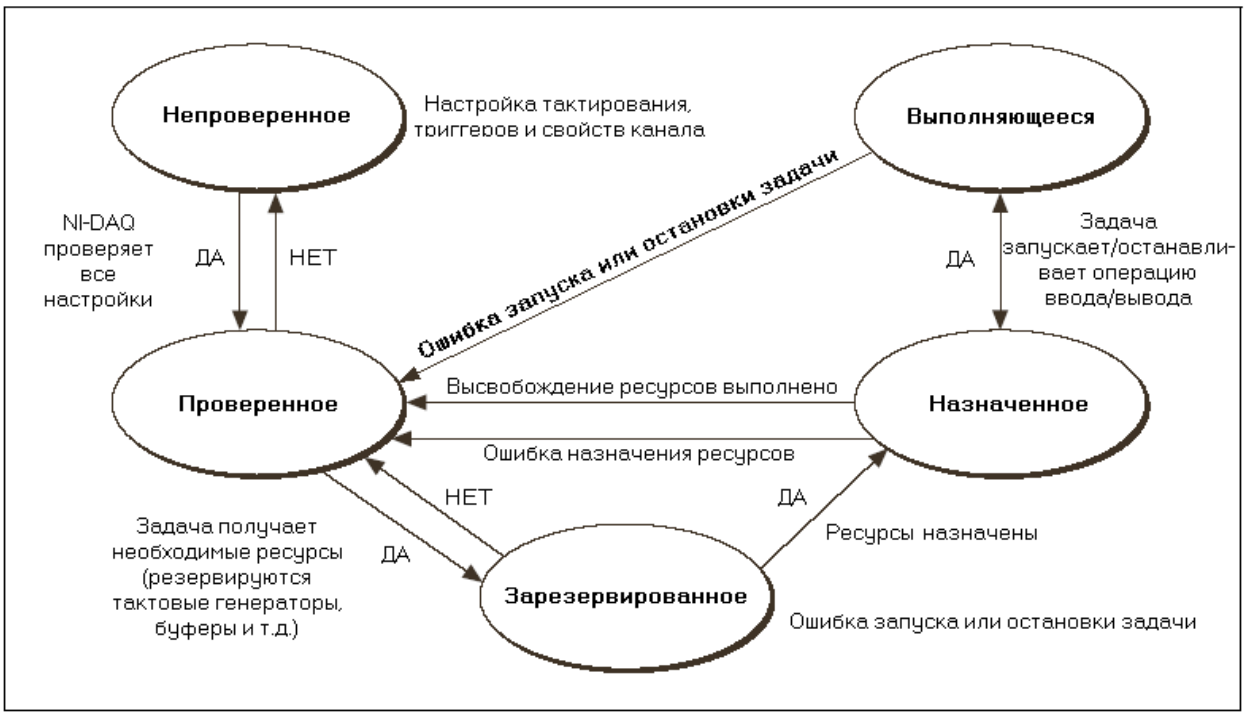
• Committed (Назначенное) – После того, как все необходимые ресурсы были получены, необходимо запрограммировать настройки этих ресурсов. Успешное программирование этих ресурсов приводит к переходу задачи в назначенное состояние. Примером настройки может являться размер буферной памяти в компьютере. Если переход не произошел, то выполнение задачи прерывается, и она будет возвращена в проверенное состояние. Для явного задания перехода вызовите ВП DAQmx Control Task с вводом action (действие) в режиме назначить (commit).

Когда ресурсы задачи для выполнения определенной операции высвобождены, задача осуществляет переход из назначенного состояния в проверенное. Для явного задания этого перехода вызовите ВП DAQmx Control Task с вводом action (действие) в режиме Unreserve. Когда задача успешно освободит все занятые ресурсы, она перейдет назад в проверенное состояние.

• Running (Выполнение) – Когда задача начинает выполнять определенную операцию, она переходит из назначенного состояния в состояние выполнения. Вы можете явно задать этот переход, вызывая ВП DAQmx Start. Обратите внимание, что запуск задачи не обязательно означает запуск сбора данных или генерацию осциллограммы. Например, свойства тактирования и триггера могли быть заданы таким образом, что выборка не будет получена до тех пор, пока не будет вызван ВП DAQmx Read, или осциллограмма не будет генерироваться до тех пор, пока не наступит триггерное событие. Если переход в состояние выполнения не произошел, выполнение задачи прерывается, и она возвращается в проверенное состояние.

Чтобы остановить задачу перед выполнением определенной операции, вызовите ВП DAQmx Stop. Это заставит перейти задачу из состояния выполнения в назначенное состояние. Если этот переход по каким-то причинам не произойдет, то выполнение задачи прекратится, и она возвратится в проверенное состояние.

Следующая иллюстрация обобщает все вышесказанное о модели состояния задачи.



3.29. Явное управление последовательностью выполнения задачи

В большинстве случаев сбора данных вам редко понадобится явное взаимодействие с моделью состояния задачи – можно доверить самой задаче выполнение неявных переходов между состояниями. Однако существуют случаи, когда необходимо использовать явное управление последовательностью выполнения задачи.

• Verify – Если пользователи вашего приложения интерактивно настраивают задачу, устанавливая различные свойства каналов, тактирования и триггеров, то явно определяйте задачу, чтобы, когда это необходимо, оповещать пользователей, что они установили для свойства неработоспособное значение.

• Reserve – В следующем случае необходимо явно резервировать задачу: ваше приложение содержит много различных задач, которые используют один и тот же набор ресурсов, одна из задач непрерывно повторяет выполнение своей операции, и вы хотите быть уверенным, что в момент ее выполнения никакая другая задача не получает этих ресурсов.

Резервирование определенной задачи приводит к тому, что последняя эксклюзивно получает необходимые ресурсы, в то время как остальные задачи не могут получить этих ресурсов. Например, если приложение содержит две задачи, каждая из которых совершает последовательность измерений, и вы хотите быть уверенным, что каждая задача завершит выполнение своей последовательности прежде, чем начнет другая, то можете использовать явное резервирование каждой задачи перед началом ее последовательности измерений.

• Commit – Явное назначение задачи будет полезным, если приложение совершает несколько измерений или генераций, повторяя запуск и остановку задачи. Явное назначение задачи приведет к тому, что она эксклюзивно получит необходимые ресурсы, и будут запрограммированы настройки этих ресурсов. При явном назначении эти операции будут совершены лишь однажды, а не каждый раз при запуске задачи, что значительно сократит время, необходимое для запуска задачи. Например, если приложение циклически совершает аппаратно синхронизированный сбор ограниченного числа выборок, то время, необходимое для запуска задачи, может очень сильно уменьшиться, если вы явно назначите задачу перед циклами этих измерений. Явное назначение задачи также требуется, если необходимы дополнительные операции считывания выборок, полученных задачей, после остановки задачи.

• Start – Если приложение циклически совершает операции чтения или записи, то будет полезен явный запуск задачи. Запуск задачи резервирует используемые ресурсы, программирует некоторые настройки этих ресурсов и начинает выполнение заданной операции. При явном запуске задачи эти действия совершаются единожды, а не при каждом выполнении операций чтения или записи. Этот процесс может значительно уменьшить время, необходимое для выполнения каждой операции чтения или записи. Например, приложение циклически производит выборку одной точки данных или, что то же, программно синхронизированные операции считывания. Тогда, если перед этими циклами вы явно запустите задачу, то время, необходимое для каждой операции считывания, может заметно уменьшиться.

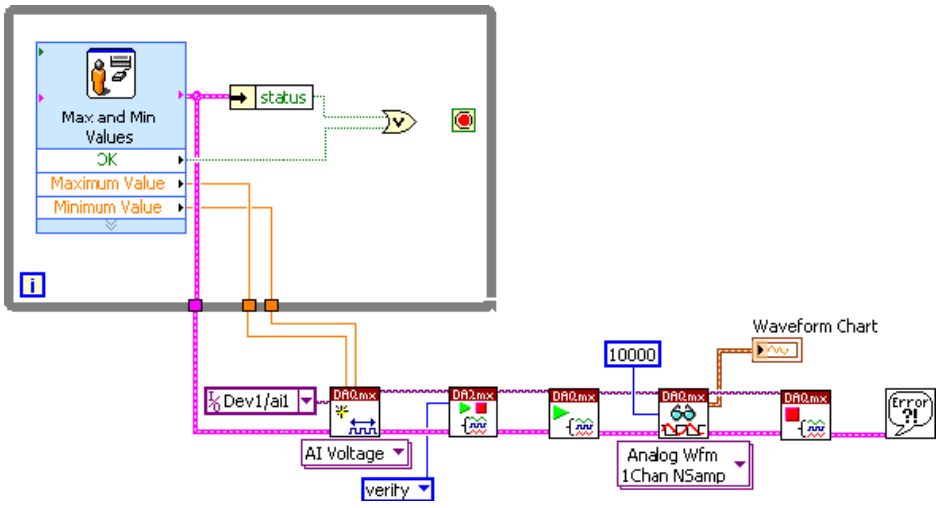
Задание:

Создать ВП для явного перехода между различными состояниями задачи NI-DAQmx

Существуют случаи, когда может понадобится использовать явный переход между задачами в модели состояния задачи.

Решение:

1. Откройте новый ВП и постройте следующую блок-диаграмму:



3.30. Синхронизация внутри устройства сбора данных

Во многих приложениях требуется одновременное выполнение нескольких типов измерений. Синхронные измерения вовлекают различные одновременно происходящие операции, такие как получение данных во входных каналах с одновременной генерацией сигналов в выходных. Однако эти операции не должны быть строго коррелированны между собой. Например, вы можете запустить операцию аналогового ввода одновременно с запуском генерации сигнала, но с этого момента каждая операция может выполняться независимо со своей частотой.

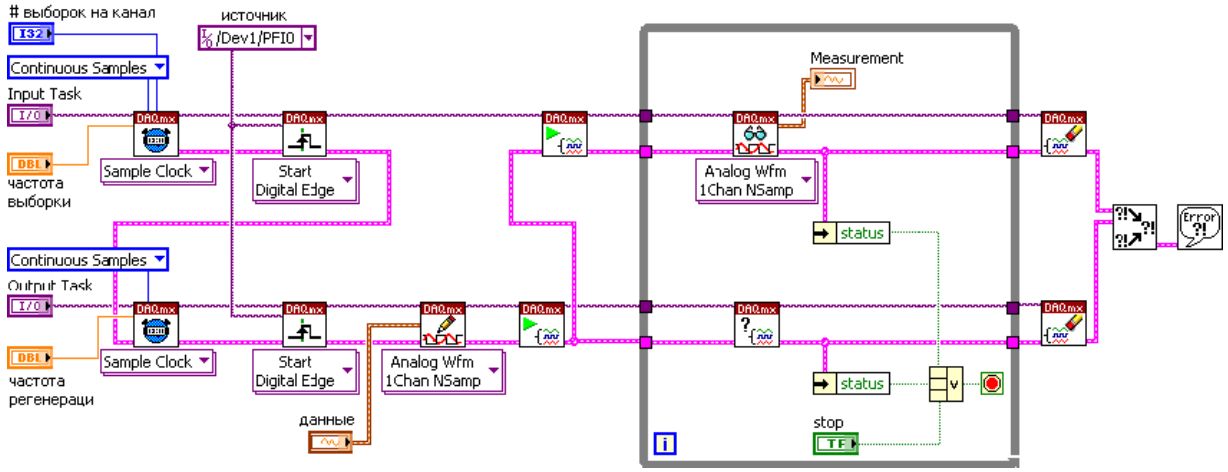
Синхронными называются измерения, происходящие в один момент времени. При синхронных измерениях, например, получении ста значений температуры и скорости, вам необходимо запустить все измерения одновременно. Кроме этого, измерения должны совместно использовать общий тактовый генератор для фиксации данных.

Например, в циклически выполняемых приложениях, необходимо в начале цикла совершить несколько измерений, произвести вычисление на основе полученных данных и затем вывести данные, основываясь на этом вычислении. Такие приложения требуют одновременного запуска всех измерений и их дальнейшей синхронизации посредством общего тактового сигнала. Похожим образом, если вы хотите соотнести измерения, например, графиков скорости и температуры тормозной колодки от времени, то, прежде всего, необходимо синхронизировать между собой измерения скорости и температуры.

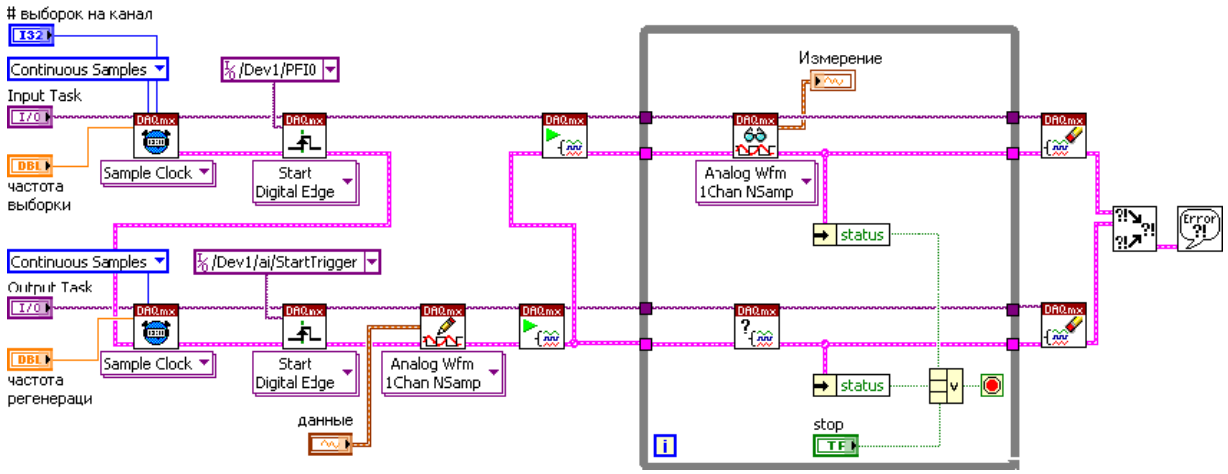
3.31. Одновременный запуск измерений

Для одновременного запуска операций аналогового ввода и вывода инициируйте их с помощью аппаратного или программного триггеров. При использовании аппаратного триггера операции аналогового ввода и вывода запускаются сигналом с одного и того же контакта PFI или RTSI.

Следующая блок-диаграмма демонстрирует выполнение операции с данным типом триггера.



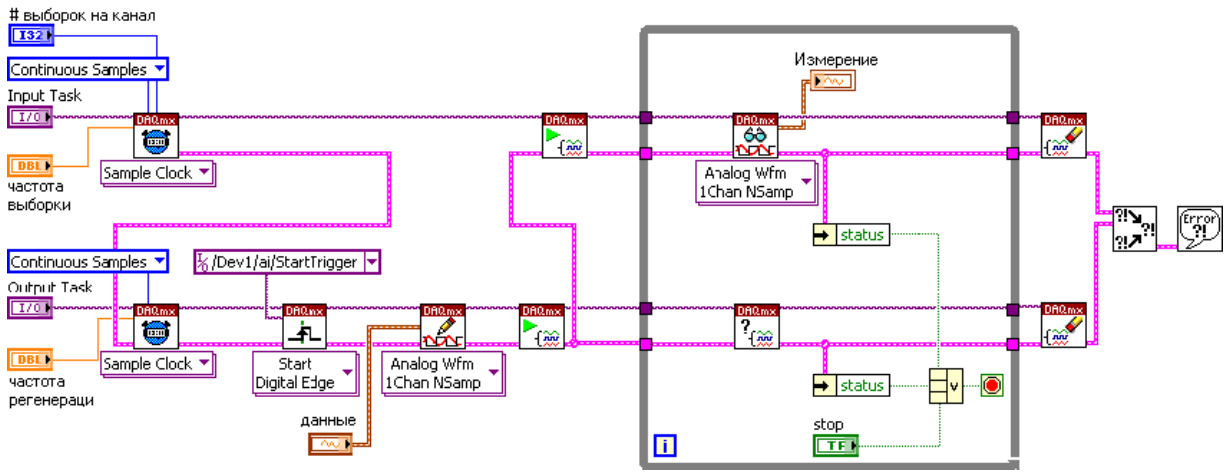
При использовании программного триггера операция аналогового ввода запускается внешним сигналом на контакте PFI или RTSI, а операция аналогового вывода запускается внутренним сигналом AI Start Trigger. AI Start Trigger – внутренний сигнал, который подается по внутренней линии, соединяющей подсистемы аналогового ввода и вывода. Метод программного триггера чуть более точен, чем метод аппаратного триггера, поскольку внешний сигнал должен пройти только по одной главной линии, чтобы достичь обеих подсистем. Однако эта задержка, как правило, несущественна на частотах, с которыми работают многофункциональные устройства семейства MIO. Следующая блок-диаграмма иллюстрирует данную методику.



В этом примере операции настроены таким образом, что аналоговый ввод инициируется триггером запуска на контакте PFI0, настроенным с помощью ВП DAQmx Trigger.

Аналоговый вывод использует ВП DAQmx Trigger для запуска по внутреннему триггеру AI Start Trigger. Обратите внимание, что операция аналогового вывода должна быть начата ранее операции аналогового ввода, чтобы не возникло такой ситуации, что ввод начался и был отправлен внутренний сигнал AI Start Trigger, а операция аналогового вывода еще не была готова получить сигнал запуска.

Другой пример одновременного запуска с использованием программного триггера возникает, когда аналоговый ввод запускается вызовом программы, а не аппаратным триггером. Аналоговый вывод, как и в предыдущем случае, запускается внутренним сигналом AI Start Trigger. Следующая диаграмма иллюстрирует пример приложения с полностью программно запускаемыми операциями ввода/вывода.



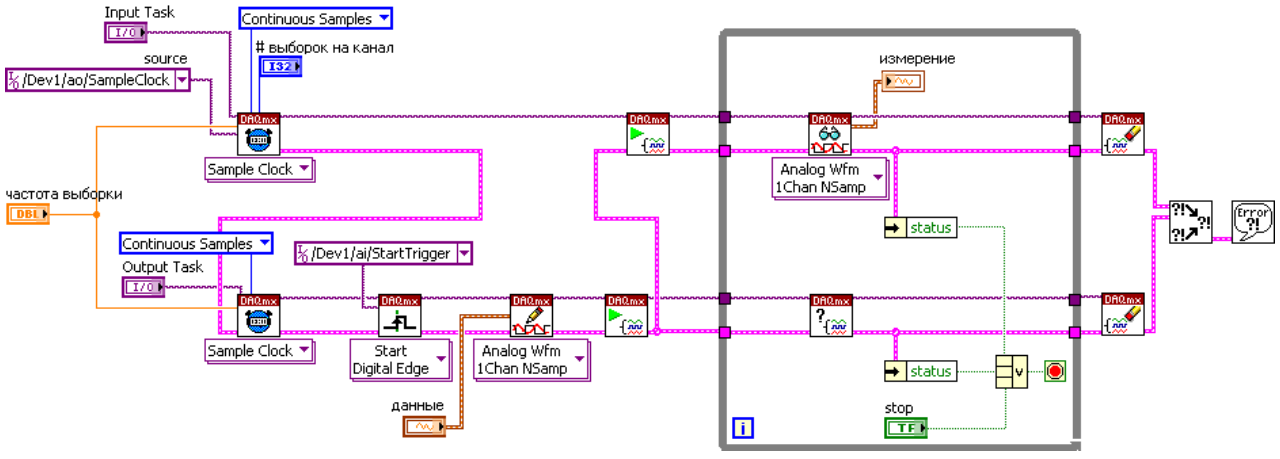
Если операции были запущены одновременно, в дальнейшем они не будут обязательно синхронизированы. Операции ввода и вывода могут быть настроены на сбор/генерацию данных с независимыми друг от друга частотами.

3.32. Синхронизированные измерения

Когда вы хотите полностью синхронизовать операции аналогового ввода и вывода так, чтобы каждая выборка из входного канала происходила одновременно с регенерацией выборки из выходного канала, операции должны использовать общий источник тактовых импульсов. Как и в случае одновременного запуска операций, существуют два метода синхронизации.

В первом методе используется внутренний сигнал AI Start Trigger для запуска аналогового вывода одновременно с аналоговым вводом. В отличие от случая с одновременным запуском теперь мы устанавливаем генераторы выборок для ввода и вывода на одинаковую частоту, что приведет к синхронизации операций в пределах одного устройства. Генераторы выборок для обеих операций тактируются генератором временной развертки в устройстве. Поскольку оба генератора являются производными от одной и той же временной развертки и запускаются в один момент времени, они будут синхронизированы. Но так как подсистемы аналогового ввода и вывода имеют собственные схемы делителей частоты для получения собственных генераторов выборки, то между генераторами будет существовать небольшая разность фаз. Однако эта разность фаз несущественна на тех частотах, с которыми работают устройства M-серии.

Другой метод синхронизации аналогового ввода и вывода заключается в использовании обеими операциями какого-либо одного генератора выборки – ввода или вывода. Следующая диаграмма демонстрирует способ синхронизации аналогового ввода и вывода при совместном использовании генератора выборок аналогового вывода.



Для выполнения синхронизированных операций аналогового ввода и вывода с использованием аппаратного триггера запуска, примените одну их показанных выше методик, и просто добавьте аппаратный триггер запуска к ведущей операции, которой во всех предыдущих примерах была операция аналогового ввода.

СПИСОК ЛИТЕРАТУРЫ

1. Тревис Дж. LabVIEW для всех. М:ДМК Пресс, ПриборКомплект, 2005 – 544 с.
2. Суранов А.Я. Справочник по функциям. М.:ДМК Пресс, 2005. – 512 с.
3. Климентьев Е.К. Основы графического программирования в среде LabVIEW. Учебное пособие. Самара: Самар. гос. аэрокосм. ун-т, 2002 г. - 65 с.
4. Сергиенко А.Б. Цифровая обработка сигналов: учебник для ВУЗов. - 2-е изд. СПб.: Питер, 2006 - 751 с.
5. Учебный курс LabVIEW. Основы I. National Instruments corp., 2002.
6. LabVIEW user manual. National Instruments corp., 2007.

ОГЛАВЛЕНИЕ