Суранов А. Я.

LabVIEW 8.20

Справочник по функциям

Москва, 2007

УДК 621.38
ББК 32.973.26-108.2
Б 28

Суранов А. Я.
LabVIEW 8.20: Справочник по функциям. – М.: ДМК Пресс, 2007. – 536 с.

ISBN 5-94074-207-6

В книге приведено описание функциональных элементов среды проектирования виртуальных приборов LabVIEW 8.20. Описание выполнения функций сопровождается примерами их использования. Для большинства Экспресс-ВП приведены окна конфигурирования с переводом их содержимого. В справочнике большое внимание уделено функциям программирования, математики, обработки сигналов, коммуникации, управления приборами и обмена данными по стандартным интерфейсам. Рассмотрены также новые элементы LabVIEW 8.20 – проект, разделяемая переменная, элементы объектно-ориентированного программирования и язык MathScript. Справочник может быть полезен широкому кругу специалистов, решающих задачи измерения, обработки или моделирования сигналов.

УДК 621.38
ББК 32.973.26-108.2

ISBN 5-94074-207-6

СОДЕРЖАНИЕ

Введение

Появление в течение последних полутора лет новых версий LabVIEW 8.0 и 8.20 стало еще одним свидетельством быстрого развития этой среды графического программирования. Компания National Instruments выдвигает в качестве ключевых особенностей данных версий концепцию распределенного интеллекта, или распределенной логики, которая включает следующие аспекты:

- формирование на базе LabVIEW единой графической платформы для программирования таких устройств, как настольные компьютеры, системы реального времени, ПЛИС, КПК, встроенные микропроцессоры и сигнальные процессоры, представляющие все узлы распределенной системы – как ведущий, так и целевые. Программирование перечисленных устройств обеспечивается с помощью соответствующих обновленных модулей;
- включение в состав LabVIEW нового **Проекта** (Project) как единого инструмента для обзора системы и обеспечения доступа ко всем ее узлам. Проект позволяет с помощью одного окна просматривать, редактировать, запускать и отлаживать код, работающий на любом целевом блоке;
- использование **Разделяемой переменной** (Shared Variable) – нового упрощенного программного интерфейса для совместного использования данных. С помощью разделяемой переменной можно передавать данные между системами, в том числе и системами реального времени, без потери скорости. Для осуществления передачи необходимо лишь сконфигурировать эту переменную с помощью несложных диалоговых окон;
- способность синхронизации внутри и между распределенными устройствами и системами. Измерение времени и синхронизация остаются ключевыми проблемами при построении эффективных измерительных и управляющих систем.

Помимо этого, в новых версиях LabVIEW произошли определенные изменения как в структуре, так и в содержании палитр элементов лицевой панели и в особенности палитр функций блок-диаграммы. Появились элементы, поддерживающие объектно-ориентированное программирование, а также текстовый язык математических расчетов MathScript.

Перечисленные и ряд других нововведений определили необходимость соответствующей модернизации справочника по функциям среды LabVIEW [1].

Модернизация коснулась прежде всего структуры и содержания глав с описанием функций, которые приведены в соответствие с новой организацией палитр блок-диаграммы, а именно в соответствие с категориями функций. Так, в частности, глава 2 содержит теперь функции из категории **программирования** (Programming), глава 3 – **математические функции** (Mathematics), а глава 4 – функции из категории **обработка сигналов** (Signal Processing). При этом в связи с большим объемом функций программирования глава 2 разделена на две подглавы, содержащие базовые и дополнительные функции. Помимо этого, многочисленные функции работы с осциллограммами, доступные как в палитре программирования, так и в палитре обработки сигналов, рассмотрены в отдельной главе 5.

Последующие три главы (с 6 по 8) посвящены функциям, входящим в состав категорий **обмена данными** (Data Communication), **поддержки взаимодействия приложений** (Connectivity), **контроля ввода/вывода** (Measurement I/O) и **связи с приборами** (Instrument I/O) и отражают содержимое соответствующих палитр.

В справочник добавлен указатель функций, в котором функции сгруппированы по категориям, а в каждой категории оригинальные названия функций упорядочены по алфавиту. Отсутствие номера страницы после названия функции означает, что эта функция в справочнике не рассмотрена.

В приложение вынесена информация о синтаксисе узла Формула и о «горячих» клавишах.

Разделы, посвященные определенной группе функций, начинаются с кратких пояснений, после чего приводится описание функций, оформленное в виде набора таблиц. Порядок следования функций обычно соответствует порядку их просмотра в палитре – слева направо и сверху вниз. В конце раздела могут быть приведены примеры применения описанных функций. В качестве примеров в большинстве случаев использовались модернизированные в той или иной степени ВП из набора примеров NI Example Finder LabVIEW.

В каждой таблице, посвященной определенной функции, приводятся, как правило, два изображения функции с подключенными элементами управления и индикаторами. При этом первое (левое) изображение имеет ярлыки (labels) элементов управления и индикаторов на английском языке, а второе – на русском. В нижней части таблицы даются пояснения по назначению и параметрам входов и выходов функции. При этом обязательные входы функции на изображении выделяются полужирным шрифтом, а текст пояснений к рекомендуемым и необязательным входам имеет уменьшенный шрифт. В большинстве случаев для экономии места входы и выходы ошибок на изображении функций ввиду их однотипности не подключались.

Благодарности

Автор выражает благодарность руководителю инновационных программ NI в Российской Федерации **П. Р. Сепояну** за всемерную поддержку работы по подготовке этого справочника.

Организация среды LabVIEW и технология программирования

1

1.1. Панели, палитры и окна LabVIEW

Запуск LabVIEW 8.20 приводит к выводу окна **первоначального запуска** (Getting Started) (рис. 1.1), которое предлагает две группы вариантов дальнейших действий пользователя: **Файлы** (Files) и **Ресурсы** (Resources). Первая группа содержит два раздела: **Новый** (New) и **Открыть** (Open). С помощью строк меню раздела **Новый** можно открыть **чистый виртуальный прибор (ВП)** (Blank VI), **пустой проект** (Empty project), **ВП из шаблона** (VI from Template) или перейти к более подробному варианту диалогового окна **Новый** (строка меню More...). В свою очередь, строки меню раздела **Открыть** позволяют открыть существующие файлы.

Строки меню раздела **Ресурсы** позволяют вызвать справочную информацию по различным аспектам работы в LabVIEW, ознакомиться с новыми элементами LabVIEW 8.20, обратиться к Web-ресурсам и к примерам разработанных ВП.

При выборе в разделе **Новый** строки **чистый ВП** (Blank VI) открываются два окна, содержащие лицевую панель (рис. 1.2) и панель блок-диаграммы (рис. 1.3) виртуального прибора (ВП).

В верхней части каждого окна размещена традиционная для приложений Windows полоса главного меню с одинаковыми для обоих окон пунктами File, Edit, View, Project, Operate, Tools, Windows и Help. Ниже полосы меню расположена полоса инструментальной панели, служащая для запуска и редактирования ВП. Полоса инструментальной панели окна блок-диаграммы отличается дополнительными кнопками для отладки ВП. В правом верхнем углу каждой панели находится иконка, наложенная на соединительную панель ВП (последняя показана на лицевой панели).

Свободное пространство каждой панелей образует рабочую область, снабженную горизонтальной и вертикальной полосами прокрутки. При построении ВП в рабочей области лицевой панели визуально размещаются элементы управления

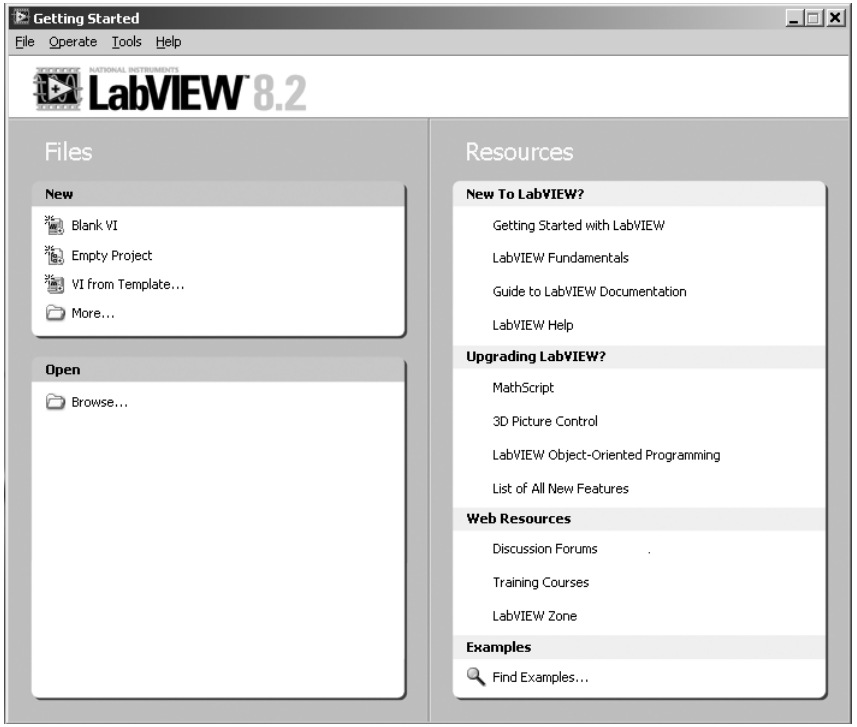


Рис. 1.1. Вид окна первоначального запуска LabVIEW 8.2

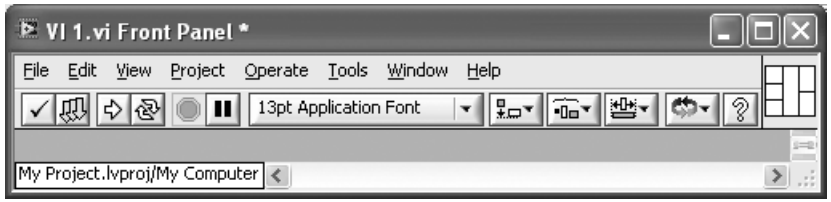


Рис. 1.2. Вид лицевой панели ВП

Рис. 1.3. Вид панели блок-диаграммы ВП

и индикации, формирующие интерфейс пользователя, а на панели блок-диаграммы составляется блок-диаграмма – графический исходный код ВП. Для одновременного отображения данных панелей в левой и правой половинах экрана целесообразно использовать меню **Окно ⇒ Панели слева и справа** (Windows ⇒ Tile Left and Right) или нажать «горячую» клавишу «T». Клавиша становится «горячей» при нажатии одновременно с ней одной или более служебных клавиш. В данном случае должна быть нажата клавиша «Ctrl», далее такое сочетание обозначается <Ctrl+T>. Перечень «горячих» клавиш приведен в приложении 2.

Построение ВП осуществляется с помощью трех вспомогательных палитр: палитры **Элементы управления** (Controls Palette), палитры **Функции** (Functions Palette) и палитры **Инструменты** (Tools Palette). Все перечисленные палитры можно вывести для постоянного или временного отображения и разместить в любом месте экрана. Вывод для постоянного отображения осуществляется с помощью разделов меню **Вид** (View). Так, в частности, при активном окне лицевой панели с помощью строки **Палитра элементов управления** (Controls Palette) меню **Вид** на эту панель можно вывести палитру элементов, а при активном окне панели блок-диаграммы на нее можно вывести палитру функций, пользуясь строкой **Палитра функций** (Functions Palette) этого же меню. Для вывода палитры инструментов необходимо использовать строку **Палитра инструментов** (Tools Palette) меню **Вид**.

Однако может оказаться, что пользователю будет более удобен временный вывод первых двух палитр, который реализуется как вызов контекстного меню каждой панели с помощью щелчка на ее рабочем пространстве правой кнопкой мыши (ПКМ). Выбор конкретного объекта из палитры элементов или палитры функций производится путем перемещения курсора мыши по разделам палитр. Выбранный объект берется из палитры с помощью щелчка левой кнопкой мыши (ЛКМ) и переносится в заданную область соответствующей панели, после чего фиксируется в этой области повторным щелчком ЛКМ (технология **Перенес и бросил** (Drag and Drop)). Эту же операцию можно выполнить с помощью щелчка ЛКМ на выбранном объекте, последующего удержания клавиши во время переноса объекта и отпускания клавиши в момент его фиксации. Такие объекты палитры функций, как **Структуры** (Structures), или строковые константы, перед фиксацией могут быть увеличены до необходимых размеров путем рисования модифицированным курсором мыши прямоугольного контура объекта при постоянно нажатой ЛКМ.

Временную версию палитры инструментов можно вывести с помощью щелчка ПКМ при нажатой клавише <Shift>.

Рассмотрим более подробно назначение пунктов главного меню, кнопок инструментальных панелей, палитр инструментов, элементов и функций.

Выше были перечислены пункты главного меню, среди которых можно выделить пункты, встречающиеся в большинстве приложений Windows, такие как File, Edit, View, Windows, Help, и пункты, являющиеся характерными для LabVIEW, такие как Project, Operate и Tools. Ниже в таблице приведено краткое описание функций пунктов главного меню.

Файл (File)	Используется для открытия новых или существующих ВП и проектов, закрытия, сохранения и вывода на печать ВП и проектов, а также для доступа к свойствам ВП
Правка (Edit)	Применяется для редактирования панелей ВП, поиска объектов и удаления неисправных проводников с блок-диаграммы, создания подприборов и установления значений элементов по умолчанию
Вид (View)	Используется для вывода доступных палитр, списка ошибок, иерархии ВП и взаимосвязей между ВП и подприборами, работы с проводником классов
Проект (Project)	Позволяет работать с проектом: создавать новый, открывать или сохранять существующий, добавлять элементы, получать информацию о файлах или устанавливать свойства проекта
Управление (Operate)	Реализует запуск и прерывание выполнения ВП, соединение с удаленной панелью и отладку приложения или общей библиотеки
Инструменты (Tools)	Используется для запуска программы анализа измерений и автоматизации (MAX) и поиска драйверов приборов, поиска ВП на диске, открытия окна MathScript, управления библиотеками ВП, управления соединением с удаленными ВП и публикацией панелей ВП в Web, конфигурирования ВП и для выполнения ряда прикладных функций
Окно (Window)	Используется для отображения окон LabVIEW и списка открытых ВП и проектов
Справка (Help)	Служит для получения информации об элементах и о функциях LabVIEW

На рис. 1.4 показан вид полосы инструментальной панели на лицевой панели, а на рис. 1.5 – вид аналогичной панели на панели блок-диаграммы.

Рис. 1.4. Вид инструментальной панели на лицевой панели

Рис. 1.5. Вид инструментальной панели на панели блок-диаграммы

Далее в таблице кратко описаны функции кнопок инструментальных панелей.

	Кнопка **Запуск** (Run) работоспособного ВП
	Вид кнопки **Запуск** (Run) при наличии ошибок в блок-диаграмме ВП
	Вид кнопки **Запуск** (Run) ВП в процессе выполнения
	Вид кнопки **Запуск** (Run) в процессе выполнения подприбора
	Кнопка **Непрерывный запуск** (Run Continuously) вызывает непрерывный запуск ВП до момента нажатия кнопки **Стоп** (Stop) или **Прервать** (Abort)


Кнопка **Прервать выполнение** (Abort Execution) вызывает остановку выполняющегося ВП


Кнопка **Пауза** (Pause) временно останавливает выполнение ВП


Кнопка **Синхронизировать с другими экземплярами приложения** (Synchronize with Other Application Instances) вносит изменения ВП во все экземпляры приложения. После нажатия этой кнопки отмена изменений невозможна. Эта кнопка появляется только при редактировании ВП, который открыт в нескольких экземплярах приложения

Следующие пять кнопок инструментальной панели блок-диаграммы используются при отладке программы, в том числе и при пошаговой отладке.

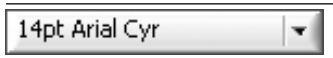

Кнопка **Подсветка выполнения** (Highlight Execution) вызывает режим анимационного показа процесса передачи данных по блок-диаграмме и отображения значений данных на выходе узлов и терминалов


Кнопка **Сохранять (Не сохранять) значения провода** (Retain (Do Not Retain) Wire Values) позволяет сохранить последнее значение, переданное по проводу, и просмотреть его после окончания выполнения при установке **Пробника данных** (Probe Data)


Кнопки **Начало пошагового выполнения** (Start Single Stepping) или **Шаг через** (Step Over) вызывают пошаговое выполнение ВП


Кнопка **Выход из пошагового выполнения** (Step Out) завершает пошаговое выполнение ВП

Кнопки, рассмотренные ниже, позволяют форматировать текстовые объекты панелей, изменять размеры и расположение объектов панелей.



Кнопка **Установки текста** (Text Settings) позволяет выбирать и устанавливать шрифт, размер, стиль и цвет текста LabVIEW


Кнопка **Выровнять объекты** (Align Objects) позволяет выровнять объекты панелей по горизонтали или по вертикали вровень с каким-либо краем или по центру


Кнопка **Распределить объекты** (Distribute Objects) позволяет распределить объекты панелей равномерно относительно их центов или краев, установить равномерные промежутки (Gaps) между объектами или удалить промежутки между ними


Кнопка **Изменить размеры объекта** (Resize Objects) позволяет изменить размеры объектов на лицевой панели


Кнопка **Изменить порядок** (Reorder) позволяет изменить порядок расположения объектов на панели при их перекрытии или зафиксировать положение объектов на панели


Кнопка **Ввести текст** (Enter Text) служит для завершения ввода текста


Кнопка **Показать окно контекстной справки** (Show Context Help Window) позволяет открыть окно контекстной справки

Все операции по созданию, редактированию и отладке ВП выполняются с помощью палитры **Инструменты** (Tools Palette) (рис. 1.6).

Рис. 1.6

При выборе определенного инструмента из палитры инструментов значок курсора мыши приобретает форму этого инструмента. При включенном автоматическом выборе инструмента наведение курсора на объект лицевой панели или блок-диаграммы LabVIEW приводит к автоматическому выбору соответствующего инструмента из палитры инструментов. Автоматический выбор инструментов включается нажатием кнопки **Автоматический выбор инструмента** (Automatic Tool Selection)  палитры инструментов или нажатием клавиш <Shift+Tab>. Выбор любого другого инструмента приводит к отключению автоматического выбора инструмента. При этом можно циклически менять инструменты с помощью клавиши <Tab>. Для переключения между инструментами **Перемещение** и **Соединение** на блок-диаграмме или между инструментами **Перемещение** и **Управление** на лицевой панели достаточно нажать пробел.

Ниже в таблице приведены краткие пояснения по инструментам, входящим в палитру.


Инструмент **Управление** (Operate Value, «палец») используется для изменения значений элементов управления или ввода текста. При работе со строковыми элементами управления вид инструмента изменяется на следующий: 


Инструмент **Перемещение** (Position/Size/Select, «стрелка») служит для выбора, перемещения или изменения размеров объектов. Для изменения размеров в LabVIEW 8.20 используются подвижные прямоугольные элементы, появляющиеся в зависимости от допустимого направления изменения в центре сторон или на углах контура объекта при установке инструмента **Перемещение** внутри этого контура


Инструмент **Редактирование текста** (Edit Text, «буква») используется для ввода и редактирования текста и создания свободных меток. При создании текстовых элементов вид инструмента изменяется:


Инструмент **Соединение** (Connect Wire, «катушка») применяется для соединения объектов на блок-диаграмме. Он также используется для условного (невидимого) подключения элементов управления и индикаторов лицевой панели к терминалам соединительной панели ВП


Инструмент **Контекстное меню объекта** (Object Shortcut Menu) вызывает контекстное меню соответствующего объекта при щелчке на нем ЛКМ


Инструмент **Быстрая прокрутка окна** (Scroll Window) используется для просмотра окна без обращения к полосам прокрутки


Инструмент **Контрольная точка** (Set/Clear Breakpoint) позволяет размещать и удалять контрольные точки на ВП, функциях, узлах, проводниках данных, структурах и приостанавливать в них выполнение программы


Инструмент **Пробник данных** (Probe Data) позволяет наблюдать данные в проводниках блок-диаграммы при выполнении ВП


Инструмент **Получить цвет** (Get Color, «пипетка») служит для копирования цвета с последующей вставкой с помощью инструмента **Установить цвет**


Инструмент **Установить цвет** (Set Color) предназначен для изменения цвета объекта. Он также отображает текущие цвета переднего и заднего плана

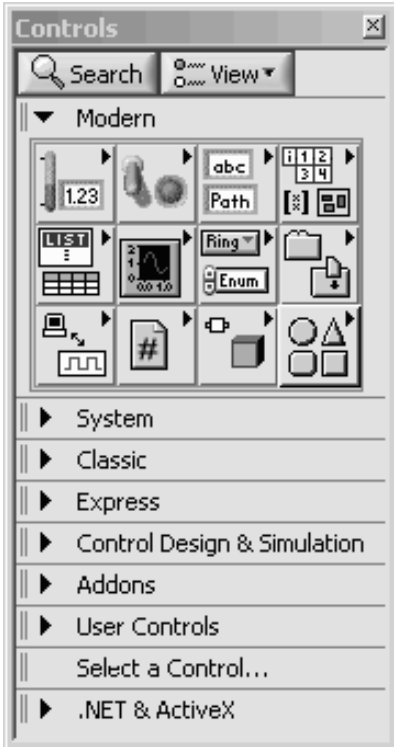


Рис. 1.7. Вид палитры элементов лицевой панели в стандартном режиме

Палитра элементов лицевой панели может отображать входящие в ее состав подпалитры элементов по категориям без надписей под иконками (стандартный вид) или с надписями (Icons & Text), в виде папок, организованных в структуре Дерево, или в виде набора вложенных текстовых папок. На рис. 1.7 показан вид палитры элементов лицевой панели, отображаемой в стандартном режиме по категориям. При этом автоматически разворачивается содержимое категории, находящейся на верхнем уровне. В данном случае это категория элементов, имеющих стиль **Современные** (Modern).

Текущее изменение вида отображения палитры производится с помощью нажатия кнопки **Вид** (View) прикрепленной палитры (рис. 1.7) и последовательного выбора строки **Видеть эту палитру как** (View This Palette As) и желаемого вида палитры. Для долговременного изменения вида палитры необходимо с помощью меню **Инструменты** ⇒ **Опции** (Tools ⇒ Options) вызвать диалоговое окно **Опции**, выбрать в списке меню категорию **Палитры элементов управления/функций** (Controls/Functions Palettes) и в поле со списком **Формат** (Format) выбрать вид палитры.

Набор отображаемых категорий может быть изменен с помощью той же кнопки **Вид** и последовательного выбора строки **Всегда видимые категории** (Always Visible Categories) и необходимых категорий в открывающемся списке. Положение категории в палитре может быть изменено с помощью ее контекстного меню, имеющего пункты **Переместить эту категорию вверх** (Move this Category Up), **Переместить эту категорию вниз** (Move this Category Down), **Переместить в верхнюю часть (Расширяемая по умолчанию)** (Move to Top (Expandable by Default)), или простым захватом в области левого края строки и ее переносом в нужное место с помощью мыши.

Изменение содержания подпалитр производится путем их перевода в режим редактирования с помощью меню **Инструменты** ⇒ **Дополнительные** ⇒ **Редактирование набора палитры**... (Tools ⇒ Advanced ⇒ Edit Palette Set...) и вызова контекстного меню подпалитры с набором операций редактирования.

Раскрытая на рис. 1.7 палитра элементов содержит следующие подпалитры.


Числовые элементы (Numeric). Элементы подпалитры используются в качестве источников или приемников числовых данных


Логические элементы (Boolean). Подпалитра содержит набор различных переключателей, кнопок и индикаторов, имитирующих действие лампочек и светодиодов. Все элементы могут находиться в двух состояниях, отображающих два состояния логической функции: ИСТИНА (True) и ЛОЖЬ (False)


Строка и путь (String & Path). Элементы подпалитры представляют типы данных, которые содержат последовательность литер, символов, массивов


Массив, матрица и кластер (Array, Matrix & Cluster). Подпалитра содержит структуры, которые позволяют создавать массивы или кластеры элементов. Массивы и кластеры представляют упорядоченное множество элементов соответственно одного или различных типов. Элементами массива могут быть числовые или логические элементы, строки или кластеры. Тип элементов массива определяется типом данных, помещаемых из палитры элементов в шаблон массива.

Матрицы группируют строки и столбцы действительных или комплексных чисел для выполнения операций линейной алгебры


Список и таблица (List & Table). Элементы подпалитры представляют собой управляющие или управляемые элементы, позволяющие заносить или отображать буквенную, символьную и цифровую информацию в виде набора строк или ячеек


График (Graph). Подпалитра содержит набор объектов, которые применяются для отображения временных или функциональных зависимостей реальных или расчетных сигналов


Кольцевой список и перечень (Ring & Enum). Элементы подпалитры представляют собой специальные числовые объекты, которые ставят в соответствие 16-битовым целым числам без знака строки, рисунки или то и другое


Контейнеры (Containers). Элементы подпалитры представляют объекты, внутри которых могут размещаться элементы управления и индикации, лицевые панели подприборов и элементы ActiveX


Ввод/вывод (I/O). Подпалитра содержит элементы управления и индикаторы, используемые для передачи установленных пользователем имен каналов DAQ, имен средств VISA и логических имен IVI к ВП ввода/вывода, обеспечивающих связь с приборами или устройствами сбора данных


Ссылка (Refnum). Подпалитра содержит идентификаторы, которые связаны с открытым приложением или файлом

Вариант и класс (Variant & Class)

Оформление (Decorations). Подпалитра служит для размещения на лицевой панели разнообразных графических элементов: линий, стрелок, рамок различной формы, надписей и т. п.

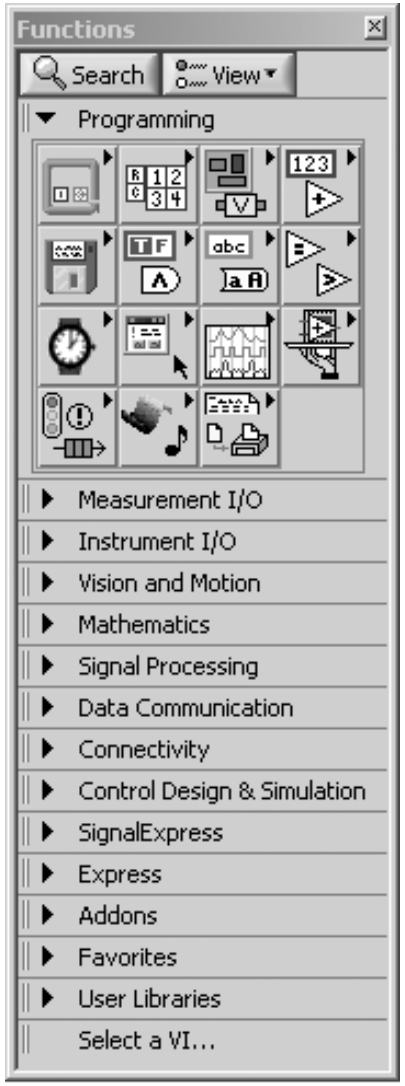


Рис. 1.8. Вид палитры функций панели блок-диаграммы

Палитра функций панели блок-диаграммы имеет такую же организацию и методику настройки, что и палитра элементов (рис. 1.8). В ее состав входят следующие категории функций: **Программирование** (Programming), **Математика** (Mathematics), **Обработка сигнала** (Signal Processing), **Обмен данными** (Data Communication), **Средства взаимодействия** (Connectivity), **Связь с приборами** (Instrument I/O) и **Экспресс** (Express). Порядок перечисления категорий соответствует порядку следования глав справочника, в которых эти функции рассматриваются.

Из перечисленных категория функций **Программирование** (раскрыта на рис. 1.8) является наиболее содержательной и играет ключевую роль при разработке широкого круга ВП. Она включает следующие подпалитры (перечисляются в порядке расположения слева направо и сверху вниз).

Подпалитра **Структуры** (Structures) содержит набор структур, управляющих выполнением ВП. Перечень структур приведен в последующей таблице

Подпалитра **Массив** (Array) включает функции выделения или замены элементов массивов, выделения подмассивов, транспонирования, сдвига, изменения размерности и т. п. Функции работы с массивами рассмотрены в разделе 2.1.5

Подпалитра **Кластер и Вариант** (Cluster & Variant) включает функции формирования кластеров и массивов кластеров, выделения элементов из кластеров, а также кластерную константу. Более подробно функции работы с кластерами описаны в разделе 2.1.6

Подпалитра **Числовые** (Numeric) содержит полный набор математических функций, функций преобразования форматов чисел и набор констант. Функции подпалитры рассмотрены в разделе 2.1.1

Подпалитра **Логические** (Boolean) включает набор функций **И** (AND), **ИЛИ** (OR), **Исключающее ИЛИ** (Exclusive Or), **НЕ** (Not) и логические константы. Более подробно логические функции рассмотрены в разделе 2.1.2

Подпалитра **Строковые** (String) содержит ряд функций обработки строковых переменных, функции взаимного преобразования чисел и строк, а также строковые константы. Перечень строковых функций и пояснения к ним приведены в разделе 2.1.3

Функции подпалитры **Сравнение** (Comparison) формируют логическую переменную в зависимости от результата сравнения входных переменных или позволяют определять соотношение чисел, нахождение числа в заданном диапазоне, тип числа или тип символа. Данным функциям посвящен раздел 2.1.4

Функции подпалитры **Установления времени** (Timing) позволяют задавать или определять временные интервалы или текущее время. Указанные функции рассмотрены в разделе 2.1.7

Функции подпалитры **Диалога и интерфейса пользователя** (Dialog & User Interface) используются для создания диалоговых окон, содержащих сообщения для пользователя. Эти функции рассмотрены в разделе 2.2.1

Функции подпалитры **Файловый ввод/вывод** (File I/O) выполняют файловые операции записи и считывания данных. Перечень функций файлового ввода/вывода и пояснения к ним приведены в разделе 2.1.8

Подпалитра **Осциллограмма** (Waveform) включает разделы **Аналоговая осциллограмма** (Analog Waveform), **Цифровая осциллограмма** (Digital Waveform), **Файловый ввод/вывод осциллограмм** (Waveform File I/O), **Измерения осциллограмм** (Waveform Measurements) и **Генерация осциллограмм** (Waveform Generation). Функции подпалитры Осциллограмма рассмотрены в главе 5

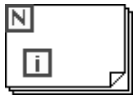
Функции подпалитры **Управление приложением** (Application Control) позволяют программно управлять приложением (LabVIEW) или ВП на локальном или удаленном компьютере. Функции подпалитры рассмотрены в разделе 2.2.2

Функции подпалитры **Синхронизация** (Synchronization) используются для синхронизации параллельно выполняющихся задач и для передачи данных между такими задачами. Функции подпалитры рассмотрены в разделе 2.2.3

Функции подпалитры **Графики и звук** (Graphics & Sound) позволяют строить трехмерные графики, производить запись/чтение графических файлов, формировать различные объекты на рисунке и осуществлять запись/чтение звуковых файлов. Функции преобразования графических файлов и формирования изображений рассмотрены в разделе 2.2.4. Функции работы со звуковой картой и звуковыми файлами рассмотрены в разделе 2.2.5

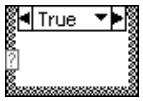
Функции подпалитры **Создание отчета** (Report Generation) позволяют формировать отчеты в виде HTML-страниц

В состав подпалитры **Структуры** (Structures) входят следующие элементы.

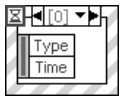


Цикл с фиксированным числом итераций (For Loop) осуществляет заданное число итераций выполнения кода внутри данной структуры

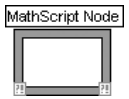
Цикл по условию (While Loop) осуществляет итерационное выполнение кода внутри данной структуры до выполнения заданного условия



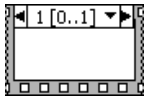
Структура **Вариант** (Case Structure) управляет выполнением одного из двух или более фрагментов кода и при выборе по условию аналогична оператору **if-then-else** текстовых языков, а при выборе по значению числовой или строковой переменной аналогична оператору **case**



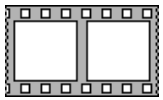
Структура **Событие** (Event Structure) ожидает наступления заданных событий на лицевой панели и производит их обработку



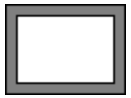
Узел MathScript исполняет функции и записи, написанные на языке LabVIEW MathScript, синтаксис которого аналогичен языку MATLAB



Структура **Стековая последовательность** (Stacked Sequence Structure) позволяет управлять последовательностью выполнения отдельных фрагментов кода путем их размещения в кадрах данной структуры



Структура **Открытая последовательность** (Flat Sequence Structure) отличается от предыдущей возможностью передачи данных между кадрами без вспомогательных переменных и возможностью вывода данных из любого кадра структуры



Узел **Формула** (Formula Node) позволяет включить фрагмент кода в текстовом представлении. Текстовый язык похож на C, но не идентичен ему

Структура отключения диаграммы (Diagram Disable Structure) имеет одну или несколько поддиаграмм или вариантов, из которых выполняется только одна поддиаграмма под заголовком **Разрешенная** (Enabled). Структура используется для отключения фрагмента блок-диаграммы

Структура отключения по условию (Conditional Disable Structure) имеет одну или несколько поддиаграмм или вариантов, из которых в зависимости от конфигурации LabVIEW использует только одну для продолжения исполнения

Узел **Обратная связь** (Feedback Node) используется для передачи значений между итерациями структур циклов

Разделяемая переменная (Shared Variable) позволяет передавать текущие данные между различными ВП проекта или по сети, при этом источниками или приемниками данных могут быть элементы лицевой панели или блок-диаграммы

Глобальная переменная (Global) используется для передачи данных между ВП на одном компьютере

Локальная переменная (Local) используется для передачи данных между элементами управления или индикаторами без применения проводов

Более подробно особенности построения и функционирования структур рассмотрены в разделе 1.3.

Помимо разработки лицевой панели и блок-диаграммы самого ВП, важное значение имеют конфигурирование его входов-выходов и формирование графического представления ВП для последующего использования в других ВП в качестве подприбора (подпрограммы). Перечисленные функции выполняются с помощью иконки и соединительной панели, размещенных в правом верхнем углу панелей. Изображение, помещенное на иконке, придает разработанному ВП индивидуальность и в большинстве случаев несет информацию о его функциональном назначении. Соединительная панель определяет картину расположения входных и выходных терминалов, посредством которых производятся ввод и вывод данных при использовании ВП в качестве подпрограммы. Для того чтобы элементы лицевой панели могли обмениваться данными с ВП верхнего уровня, они должны быть подключены к терминалам соединительной панели.

В качестве примера на рис. 1.9 показаны иконка и соединительная панель ВП **Записать в файл табличного формата** (Write To Spreadsheet File), находящегося в подпалитре **Файловый ввод/вывод**.

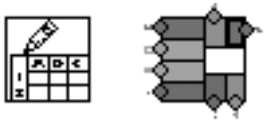


Рис. 1.9

Вызов диалогового окна **Редактор иконки** (Icon Editor) для редактирования изображения иконки осуществляется с помощью строки **Редактировать иконку** (Edit Icon) контекстного меню иконки на лицевой панели. Вызов функций редактирования соединительной панели производится с помощью строки **Показать соединительную панель** (Show Connector) того же меню.

Ниже в таблице приведено краткое описание инструментов для создания иконки в диалоговом окне **Редактор иконки**.

Инструмент	Название	Функция
	Карандаш	Рисует и стирает элементы изображения
	Линия	Рисует прямые линии
	Пипетка	Копирует цвет переднего плана с элемента в иконке
	Наполненное ведро	Заполняет выделенную область цветом переднего плана
	Прямоугольник	Рисует прямоугольник в цвете заднего плана
	Заполненный прямоугольник	Рисует прямоугольник, окаймленный цветом переднего плана и заполненный цветом заднего плана
	Выбор	Выбирает область иконки для перемещения, копирования или других изменений

Инструмент	Название	Функция
A	Текст	Вводит текст в иконку. Изменение атрибутов шрифта производится с помощью двойного щелчка на этом инструменте
	Передний план/ задний план	Показывает текущий цвет переднего и заднего планов. Выбор цвета производится с помощью щелчка мышью на соответствующем прямоугольнике

Новым элементом рабочей среды LabVIEW 8.0 и 8.20 является **оболочка управления проектами** – LabVIEW Project, призванная обеспечить разработку распределенных приложений. Проект также поддерживает коллективную разработку больших приложений за счет включения интегрированных средств управления исходными текстами (Visual SourceSafe, Perforce, Rational ClearCase, PVCS, MKS и CVS) и **библиотек проектов** (Project Libraries), содержащих исходные коды в виде модульных, унифицированных функций, которые можно многократно вызывать из различных подсистем. Проект позволяет создавать загружаемые модули программ в виде **автономного приложения** (Stand-Alone Application) или динамически подключаемой библиотеки (DLL), а также zip-файлы или файлы **распространения исходной программы** (Source Distribution). Таким образом, простая и дружественная пользователю оболочка проекта LabVIEW позволяет наблюдать, редактировать, загружать, выполнять и отлаживать программный код, работающий на любом узле системы.

Для создания и редактирования проектов LabVIEW служит окно **Проводника проекта** (Project Explorer) (рис. 1.10). Окно отображается при выборе меню **Файл ⇒ Новый проект** (File ⇒ New Project) или **Проект ⇒ Новый проект** (Project ⇒ New Project).

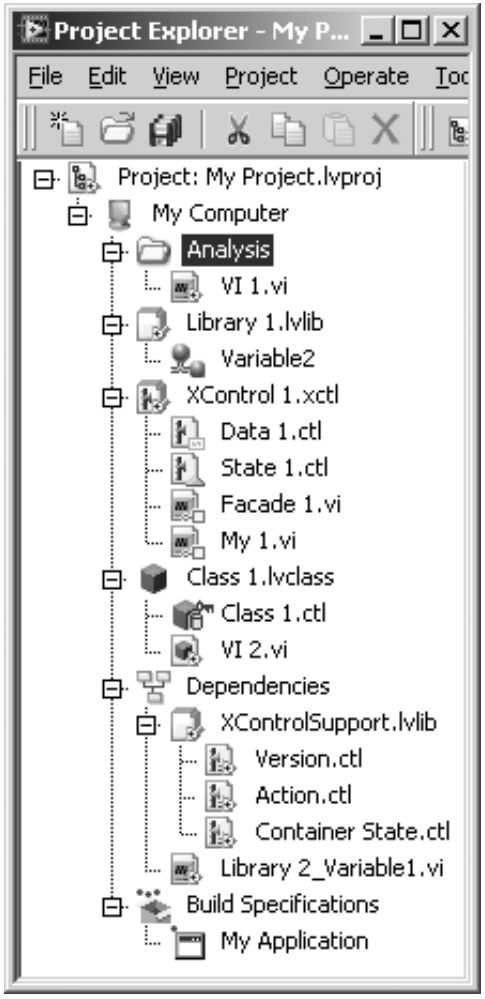


Рис. 1.10. Вид окна проводника проекта

По умолчанию окно Проводника проекта включает следующие узлы:

- **Корень проекта** (Project root) – содержит все прочие узлы окна Проводника проекта. Надпись в этом узле включает имя файла проекта;
- **Мой компьютер** (My Computer) – представляет локальный компьютер в качестве платформы проекта;
- **Зависимости** (Dependencies) – включает элементы, которые необходимы для ВП, находящихся под узлом платформы;
- **Спецификации создания** (Build Specifications) – включает конфигурации формирования файлов распространения исходных программ и других видов приложений на основе инструментальных средств и модулей LabVIEW. При наличии установленных LabVIEW Professional Development System или Application Builder можно использовать узел **Спецификации создания** для конфигурации автономных приложений (EXE), библиотек динамической компоновки (DLL), инсталляторов и zip-файлов.

Перед сохранением проекта все его новые несохраненные файлы должны быть сохранены.

К существующему проекту могут быть добавлены новые или существующие папки или файлы, ВП, библиотеки ВП, библиотеки и подбиблиотеки проекта, элементы XControl, разделяемые переменные и объекты классов. Добавление новых элементов может производиться как с помощью строк **Новый** (New), **Добавить файл** (Add File), **Добавить папку** (Add Folder) контекстного меню узла, в который производится добавление, так и с помощью выделения этого узла и вызова меню **Проект ⇒ Добавить к проекту** (Project ⇒ Add to Project).

Библиотеки ВП могут быть добавлены к проекту LabVIEW как в виде папки, так и в виде файла. При добавлении библиотеки в виде папки LabVIEW использует имя библиотеки для наименования папки и добавляет ВП библиотеки как элементы в новую папку. Поиск библиотеки производится в диалоговом окне, открывающемся при выборе контекстного меню **Добавить папку** (Add Folder) заданного узла или расположенной под ним папки. Добавление или удаление элементов папки в окне не влияет на файл .llb на диске. С помощью контекстного меню папки **Преобразовать в библиотеку** (Convert To Library) папка может быть преобразована в **библиотеку проекта** (Project Libraries).

При добавлении библиотеки ВП в виде файла входящие в ее состав ВП появляются в окне Проводника проекта. Поиск библиотеки инициализируется с помощью строки **Добавить файл** (Add File) контекстного меню заданного узла. При выборе папки открывается диалоговое окно, в котором можно выделить отдельные файлы ВП или всю библиотеку, установив строку View All, и нажать кнопку **Добавить файл** (Add File).

Библиотеки проекта LabVIEW представляют наборы ВП, определений типа, разделяемых переменных, объектов классов, файлов палитр меню и других файлов, включающих другие библиотеки проектов. При создании и сохранении новой библиотеки проекта LabVIEW создает файл библиотеки проекта (.lvlib), который включает свойства библиотеки проекта и ссылки к файлам, которые входят в ее состав. Файл библиотеки проекта в отличие от библиотеки ВП не содержит самих этих файлов.

Структуру библиотеки проекта можно просматривать в окне Проводника проекта или в автономном окне библиотеки проекта.

Библиотеки проекта могут быть полезны для организации файлов в единую иерархию элементов, потенциального дублирования имен ВП, отслеживания версий, ограничения открытого доступа к определенным файлам, ограничения возможности редактирования набора файлов и установки палитры меню для группы ВП.

Библиотеки проекта могут применяться для уточнения имен ВП и других файлов. LabVIEW распознает ВП по имени файл, поэтому LabVIEW может непреднамеренно загрузить и установить ссылку на ВП, имеющий такое же имя, что и другой ВП (проблема, известная как перекрестное связывание). Разные библиотеки проекта могут использовать ВП с одинаковыми именами.

Пользователь может устанавливать номера версий в библиотеке проекта для различения изменений набора файлов с течением времени. Установка номеров версий осуществляется в разделе **номер версии** (Version Number) на странице **общие установки** (General Settings) диалогового окна **свойства библиотеки проекта** (Project Library Properties) (рис. 1.11), вызываемого с помощью строки **Свойства** (Properties) контекстного меню библиотеки.

На этой же странице в разделе **Защита** (Protection) могут быть установлены различные степени ограничения на редактирование библиотеки проекта. В частности, может быть установлено **блокирование (без пароля)** (Locked (no password)) или **защита паролем** (Password-protected). При блокировании библиотеки проекта пользователи не могут добавлять или удалять элементы и не видят элементы, установленные как закрытые.

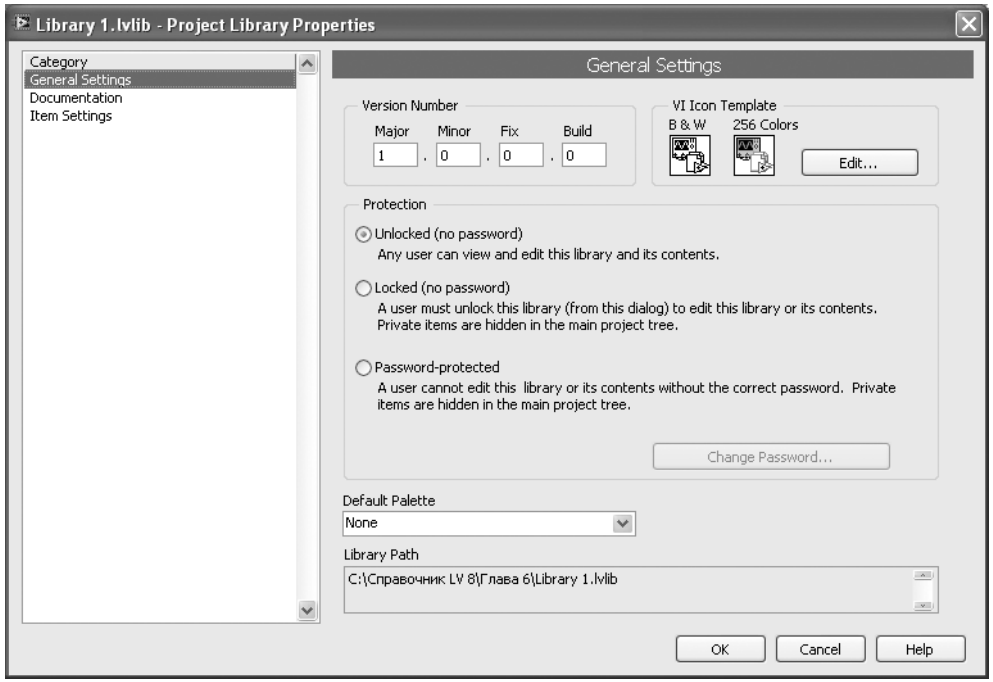


Рис. 1.11. Вид диалогового окна ***Свойства библиотеки проекта***

Библиотека проекта также позволяет ограничить общий доступ к определенным типам файлов. С помощью раздела **область доступа** (Access Scope), расположенного на странице **установки параметров элементов** (Item Settings) диалогового окна **свойства библиотеки проекта**, можно сконфигурировать доступ к элементам и папкам библиотеки проекта как **открытый** (Public) или как **закрытый** (Private). При установке ВП как закрытого для доступа другие ВП, которые не входят в данную библиотеку, не могут вызывать его.

В LabVIEW 8.20 как в проект, так и в другие элементы среды введены средства поддержки **объектно-ориентированного программирования** (ООП). С ООП связаны такие понятия, как **структура класса**, **инкапсуляция** и **наследование**. Эти понятия используются для создания кода, который можно обновлять и модифицировать без изменения других частей приложения.

В LabVIEW ООП может использоваться для создания типов данных, определяемых пользователем. Такие типы данных создаются с помощью формирования **классов** LabVIEW. Классы LabVIEW определяют данные, связанные с **объектом**, а также **методы**, задающие действия, которые можно выполнять над данными.

Классы LabVIEW могут быть созданы в окне Проводника проекта (рис. 1.12) с помощью контекстных меню таких пунктов, как проект, My Computer, папка и библиотека проекта. В этом окне класс LabVIEW отображается синим кубиком, расположенным слева от названия. После сохранения класса LabVIEW создает файл **библиотеки класса** с расширением .lvclass, который определяет новый тип данных. В этот файл записываются элемент управления закрытыми данными и информация о любых созданных ВП – членах класса, например такая как перечень и различные свойства ВП. Библиотека класса подобна библиотеке проекта с расширением .lvlib, но отличается тем, что создает новый тип данных. Вид и цвет проводника, по которому передается информация класса, настраивается на странице **внешний вид проводника** (Wire Appearance) диалогового окна **свойства** класса.

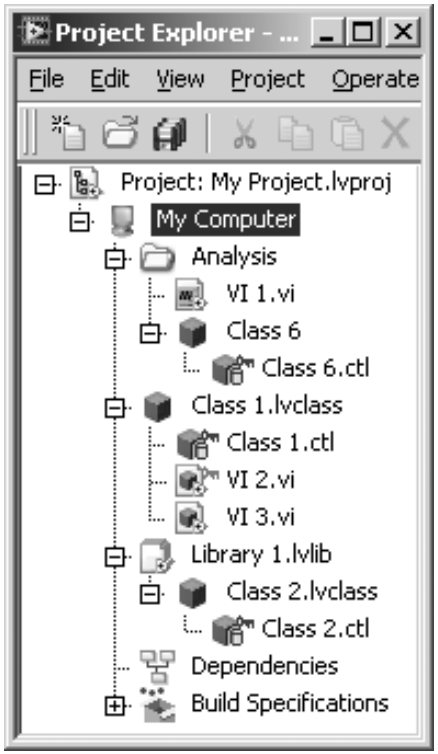


Рис. 1.12. Различные варианты создания классов в LabVIEW

При создании класса LabVIEW автоматически создает элемент управления данными с таким же названием и расширением .ctl. В окне Проводника проекта этот элемент отображается в виде синего кубика с зеленым цилиндром, символизирующим сохранение данных. В LabVIEW данные классов являются **закрытыми** (private). Это означает, что доступ к ним могут получить только ВП, являющиеся **членами** класса. Закрытый характер данных обозначается на значке объекта значком ключа красного цвета. Элемент управления данными является единственным файлом библиотеки класса, который определяет кластер закрытых данных для нового типа данных и вид проводника, по которому эти данные передаются. Сохранение элемента управления закрытыми данными в файле библиотеки класса, а не на диске позволяет LabVIEW гарантировать предоставление корректных данных вместе с определением класса.

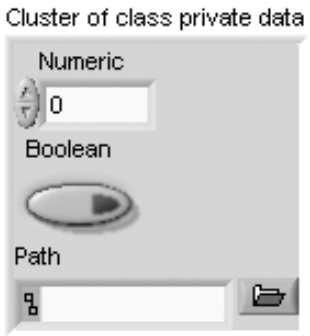


Рис. 1.13. Вид кластера закрытых данных

Настройка элемента управления закрытыми данными осуществляется в окне **Редактор элемента управления** (Control Editor), которое открывается после двойного щелчка на элементе управления в окне Проводника проекта. Определение закрытого типа данных класса LabVIEW производится путем размещения элементов управления и индикаторов в **Кластере класса закрытых данных** (Cluster of class private data) (рис. 1.13). Значения по умолчанию, которые устанавливаются для элементов кластера, являются значениями по умолчанию для этого класса.

Для доступа к закрытым данным создаются методы, которые реализуются с помощью ВП – членов класса. Эти ВП появляются в окне Проводника проекта под элементом управления закрытыми данными класса. Объединение закрытых данных и соответствующих методов и характеризует такой аспект ООП, как **инкапсуляция** (encapsulation).

ВП – члены класса могут быть созданы из **чистого ВП** (Blank VI), из **шаблона ВП** (VI template), который включает обработку ошибок и объекты класса, а также из **ВП – члена родительского класса** (Ancestor Member VI). Эти варианты создания реализуются с помощью соответствующих строк контекстного меню класса: **новый ⇒ ВП** (New ⇒ VI), **новый ⇒ динамический ВП** (NewDynamic VI) и **новый ⇒ корректировка ВП** (New ⇒ Override VI). При создании ВП – члена класса из ВП – члена родительского класса у них должны совпадать следующие настройки: реентерабельность, предпочтительное исполнение, приоритет и объем доступа, а также терминалы и конфигурация соединительной панели. LabVIEW не подсвечивает опцию **новый ⇒ корректировка ВП**, если нет действительного ВП – члена базового класса.

В то время как данные класса всегда закрыты, для ВП – членов класса можно устанавливать различную степень доступа. В разделе **область доступа** (Access Scope) страницы **параметры установки элемента** (Item Settings) диалогового

окна **свойства класса** (Class Properties) для методов могут быть установлены следующие варианты доступа:

- **Открытая область** (Public scope) – любой ВП может вызывать ВП – члена класса как **подприбор** (subVI);
- **Защищенная область** (Protected scope) – только ВП, находящиеся внутри того же класса или класса-потомка могут вызывать ВП – члена класса. Защищенный ВП – член класса отображается в окне Проводника проекта с темно-желтым значком ключа;
- **Закрытая область** (Private scope) – только ВП, находящиеся внутри того же класса, могут вызывать ВП – члена класса. Закрытый ВП – член класса отображается в окне Проводника проекта с красным значком ключа.

Рассмотренный выше вариант создания ВП – члена класса из ВП – членов родительского класса связан с таким понятием ООП, как **наследование** (inheritace). Наследование позволяет использовать существующий класс в качестве исходной точки для нового класса, в частности использовать открытые и защищенные ВП – члены родительского класса. Варианты наследования устанавливаются на странице **наследование** диалогового окна **свойства** класса.

При упаковке или распаковке класса LabVIEW узлы показывают терминалы только для закрытых данных текущего класса, а не для любых данных, которые класс наследует от родительских классов. Родительские данные являются закрытыми и их можно модифицировать с помощью функций, которые обеспечиваются ВП – членами класса. Эти ВП подобно любому ВП в LabVIEW могут вызывать любой из открытых ВП – членов класса. Но они могут также вызывать и защищенные ВП – члены родительского класса. При создании ВП – члена родительского класса как защищенного член – ВП любого дочернего класса может вызвать метод, но никакие ВП вне иерархии наследования не могут сделать это.

Объектом LabVIEW называется специальный класс, который является первоначальным родителем дерева наследования в ООП LabVIEW. По умолчанию все классы LabVIEW наследуют от объекта LabVIEW. Объект LabVIEW можно использовать для создания ВП, которые могут выполнять общие операции на множестве классов LabVIEW. Для представления объекта LabVIEW на лицевой панели служит одноименный элемент управления, размещенный в подпалитре **Вариант и класс** (Variant & Class).

Большинство методов можно определить с помощью единственного ВП – члена класса в одном классе, но некоторые методы можно определить путем создания нескольких ВП – членов класса с одинаковым именем, находящихся в иерархии класса. В первом случае методы называются **статическими**, во втором – **динамическими**. Статический или динамический тип определяется в соединительной панели ВП – члена класса. Если эта панель включает входной терминал динамической отсылки, то ВП является частью динамического метода. В противном случае ВП – член определяет статический метод. Данные, поступающие на входной терминал динамической отсылки, определяют экземпляр ВП – члена класса, который будет использоваться LabVIEW для реализации метода.

1.2. Технология проектирования виртуальных приборов

Для проектирования ВП в среде LabVIEW необходимо сформировать его лицевую панель и разработать блок-диаграмму. При формировании лицевой панели производятся выбор и установка на ней элементов управления и индикаторов из палитры элементов данной панели. Аналогично при разработке блок-диаграммы производятся выбор и установка на ней функциональных элементов и подприборов из палитры функций данной панели.

Установка каждого элемента на лицевой панели сопровождается появлением соответствующего **терминала** данных (terminal) на панели блок-диаграммы. Терминалы элементов управления представляют порты ввода информации в блок-диаграмму, а терминалы индикаторов – порты вывода информации из блок-диаграммы на лицевую панель. Для обработки введенной информации и программного управления параметрами и режимами работы элементов лицевой панели на панели блок-диаграммы размещаются необходимые константы, **функции** (Functions), **подприборы** (SubVI) и **структуры** (Structures), которые также имеют терминалы для ввода и вывода информации. Все перечисленные элементы представляют **узлы** (nodes) блок-диаграммы, которые соединяются с терминалами элементов управления и индикации и между собой линиями, называемыми **проводниками** (wires). В такой схеме через узлы в процессе обработки проходит **поток данных** (data flow), идущий по проводникам от входных терминалов к выходным. Узлы – это объекты на блок-диаграмме, которые имеют одно или более полей ввода/вывода данных и выполняют алгоритмические операции ВП. Они аналогичны операторам, функциям и подпрограммам текстовых языков программирования.

Таким образом, описанная технология формирования ВП является основой для **потоковой модели** обработки данных, когда поток данных входит (втекает) в узлы-источники, проходит через узлы обработки данных и выходит (вытекает) через узлы – приемники данных. При этом порядок обработки данных определяется целиком полнотой подхода данных к терминалам узлов. Такая концепция работы программы в LabVIEW существенно облегчает, по сравнению с текстовыми языками, разработку **многозадачных** и **многопоточных программ**.

Терминалы данных имеют прямоугольную форму и содержат буквенно-графическое обозначение, характеризующее тип и форму представления воспринимаемых ими данных. Таким образом, по виду терминала можно определить, является ли он источником или приемником данных, какие типы данных он воспринимает – числовые, логические или строковые, а для числовых – является ли число целым или вещественным. Для определения таких различий используются различия в толщине внешней рамки терминала и направлении треугольной стрелки внутри него, цвет терминала и буквенное или графическое обозначение. Кроме того, вид терминала можно определить и по содержанию контекстного меню.

Еще один способ идентификации терминала связан с отображением иконки соответствующего элемента. Такое отображение включается с помощью опции

Отображать в виде иконки (View as Icon) контекстного меню терминала. В качестве примера можно сравнить вид терминала числового элемента ввода данных в форме с плавающей запятой одинарной точности в его традиционном представлении  и в представлении в виде иконки  . Однако представление в виде иконки приводит к потере пространства блок-диаграммы, что не всегда удобно.

Перечень и вид терминалов элементов ввода/вывода данных LabVIEW 8.20 с характеристикой типа, цвета и значения по умолчанию приведены в таблице.

Элемент управления	Элемент индикации	Тип данных	Цвет	Значение по умолчанию
		Числовой с плавающей запятой одинарной точности	Оранжевый	0,0
		Числовой с плавающей запятой двойной точности	Оранжевый	0,0
		Числовой с плавающей запятой расширенной точности	Оранжевый	0,0
		Комплексный с плавающей запятой одинарной точности	Оранжевый	0,0+i0,0
		Комплексный с плавающей запятой двойной точности	Оранжевый	0,0+i0,0
		Комплексный с плавающей запятой расширенной точности	Оранжевый	0,0+i0,0
		Числовой байтовый со знаком	Синий	0
		Числовой 16-битовый со знаком	Синий	0
		Числовой 32-битовый со знаком	Синий	0
		Числовой 64-битовый со знаком	Синий	0
		Числовой байтовый без знака	Синий	0
		Числовой 16-битовый без знака	Синий	0
		Числовой 32-битовый без знака	Синий	0
		Числовой 64-битовый без знака	Синий	0
		<64,64>-битовая отметка времени		12:00:00.000 AM 1/1/1904
		Нумерованный	Синий	–

Элемент управления	Элемент индикации	Тип данных	Цвет	Значение по умолчанию
		Логический	Зеленый	FALSE
		Строковый	Розовый	Пустая строка
		Массив – заключает тип данных в квадратные скобки и принимает цвет этого типа данных	Различный	–
		Матрица из вещественных чисел – используется для выполнения матричных операций и отличается от массива типом проводника	Оранжевый	
		Матрица из комплексных чисел	Оранжевый	
		Кластер – заключает разные типы данных. Кластерный тип данных имеет коричневый цвет, если все элементы кластера числовые, и розовый, если элементы разных типов. Кластер ошибкиокрашен в темно-желтый цвет, а кластер объекта класса LabVIEW – в темно-красный	Коричневый, розовый, темно-желтый или темно-красный	–
				
				
				
		Путь – сохраняет положение файла или каталога, используя стандартный синтаксис используемой платформы	Морской волны	<Не путь>
		Динамический (Dynamic) – (Express VIs). Включает данные, связанные с сигналом и атрибутами, которые обеспечивают информацию о сигнале, такую как имя сигнала или дата и время получения данных	Фиолетовый	
		Осциллограмма – кластер элементов, которые содержат дату, начальное время и интервал выборок сигнала	Коричневый	–
		Цифровая осциллограмма – переносит начальное время, Δx, цифровые данные, любые атрибуты цифровой осциллограммы	Зеленый	
		Цифровые данные – содержат цифровые данные, отображаемые в битовом виде по строкам и столбцам	Зеленый	

Элемент управления	Элемент индикации	Тип данных	Цвет	Значение по умолчанию
		Ссылка – действует как уникальный идентификатор для таких объектов, как файл, устройство или сетевое соединение	Морской волны	–
		Вариант – включает имя элемента управления или индикатора, информацию о типе данных и сами данные	Фиолетовый	–
		I/O name – передает имя канала платы сбора данных (DAQ), ресурсное имя VISA или логическое имя IVI с целью их конфигурации	Фиолетовый	–
		Рисунок – отображает рисунок, который содержит линии, окружности, текст и другие графические элементы	Синий	–

Настройка параметров объектов лицевой панели (элементов управления и индикаторов) и терминалов блок-диаграммы производится с помощью контекстного меню, содержащего команды и опции. Контекстное меню открывается с помощью щелчка ПКМ на объекте. Опции, входящие в состав контекстного меню, зависят от типа объекта. Вместе с тем в состав контекстного меню многих объектов входит ряд одинаковых пунктов. Сочетание общего и индивидуального можно показать на примере контекстных меню числового элемента управления (рис. 1.14) и его терминала (рис. 1.15).

Краткие описания функций пунктов контекстных меню приведены в таблице.

Видимые объекты (Visible Items)	Позволяет показывать или скрывать определенные элементы оформления, такие как ярлыки, заголовки, полосы прокрутки или соединительные терминалы
Найти терминал (Find Terminal)	LabVIEW находит и выделяет на блок-диаграмме терминал элемента лицевой панели
Найти элемент управления (Find Control)	LabVIEW находит и выделяет на лицевой панели элемент управления, имеющий заданный терминал на блок-диаграмме
Заменить на индикатор (Change to Indicator)	Позволяет заменить элемент управления лицевой панели на индикатор. Для индикатора аналогичный пункт выполняет обратное действие
Сделать элемент управления невидимым (Hide Control)	Позволяет сделать элемент управления невидимым. Для невидимого элемента аналогичный пункт меню выполняет обратное действие

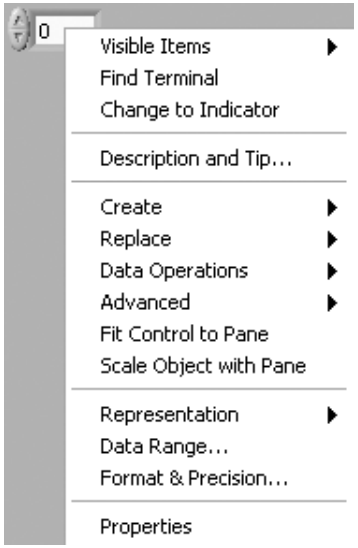


Рис. 1.14. Вид контекстного меню числового элемента управления

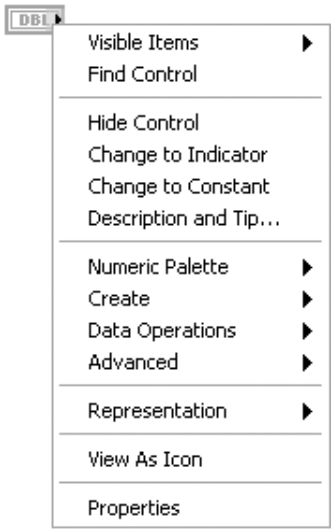


Рис. 1.15. Вид контекстного меню терминала числового элемента управления

Описание и подсказка (Description and Tip)	Позволяет описать объект и сделать подсказку к нему
Создать (Create)	Позволяет создавать локальную переменную, ссылку, узел свойств или методов
Заменить (Replace)	Позволяет входить в палитры панелей и заменять выбранный объект другим
Операции с данными (Data Operations)	Позволяет устанавливать текущие значения как значения по умолчанию, удалять, копировать и вставлять данные
Дополнительно (Advanced)	Позволяет производить тонкую настройку элементов управления и индикаторов
Свойства (Properties)	Позволяет вызывать диалоговое окно свойств объекта

Пункты, перечисленные в таблице, являются специфичными для числовых элементов.

Представление (Representation)	Позволяет выбирать и устанавливать тип представления числовых данных
Диапазон данных (Data Range)	Позволяет устанавливать определенный действующий диапазон числовых значений и приращений данных
Формат и точность (Format&Precision)	Позволяет выбирать формат и точность представления данных

Новой в LabVIEW 8.20 является возможность вызова контекстного меню элемента во время исполнения (run-time). Это меню может редактироваться или удаляться с помощью диалогового окна Shortcut Menu Editor, вызываемого путем вызова последовательности строк **Расширенное ⇒ Контекстное меню периода исполнения ⇒ Редактировать** (Advanced ⇒ Run-Time Shortcut Menu ⇒ Edit) контекстного меню элемента.

Представление числовых данных в LabVIEW включает следующие типы.

Повышенной точности

Двойной точности

32-битовый

Целочисленный 64-битовый со знаком

Целочисленный 32-битовый со знаком

Целочисленный 16-битовый со знаком

Целочисленный байтовый со знаком

Целочисленный 64-битовый без знака

Целочисленный 32-битовый без знака

Целочисленный 16-битовый без знака

Целочисленный байтовый без знака

Комплексный повышенной точности

Комплексный двойной точности

Комплексный 32-битовый

Для логических элементов управления специфичным является пункт контекстного меню **Механическое действие** (Mechanical Action), определяющий характер срабатывания логического элемента при его нажатии. Варианты срабатывания приведены в таблице.

Изображение	Режим	Действие
	**Включить при нажатии** (Switch When Pressed)	Значение изменяется при каждом щелчке по элементу инструментом управления
	**Включить при отпускании** (Switch When Released)	Значение изменяется после отпускания кнопки мыши при каждом щелчке по элементу инструментом управления

Изображение	Режим	Действие
	Включить до отпускания (Switch Until Released)	Значение изменяется при щелчке и автоматически возвращается в исходное состояние после отпускания кнопки мыши
	Сработать при нажатии (Latch When Pressed)	Значение изменяется при щелчке и автоматически возвращается в исходное состояние после однократного считывания виртуальным прибором нового значения
	Сработать при отпускании (Latch When Released)	Значение изменяется после отпускания кнопки мыши и автоматически возвращается в исходное состояние после однократного считывания виртуальным прибором нового значения
	Сработать до отпускания (Latch Until Released)	Значение изменяется при щелчке и возвращается в исходное состояние после однократного считывания виртуальным прибором нового значения или после отпускания кнопки мыши

В контекстном меню строковых элементов управления и индикации специфичным является раздел, определяющий вид отображения содержимого этих элементов. Интерпретация пунктов данного раздела меню приведена в таблице.

Нормальное отображение (Normal Display)	Позволяет отображать содержимое в том виде, в котором оно вводится с клавиатуры
"\" Кодированное отображение (Codes Display)	Позволяет принимать и отображать символы, которые являются неотображаемыми. Скрытые символы появляются в виде обратного слэша, за которым следует соответствующий код. Ниже в таблице приведены слэш-коды LabVIEW

Код	Выполнение в LabVIEW
\00-\FF	Шестнадцатеричное значение 8-битового символа, буквы должны быть заглавными
\b	Знак возврата на один символ назад (ASCII BS, эквивалент \08)
\f	Знак смещения формы (ASCII FF, эквивалент \0C)
\n	Новая строка (ASCII LF, эквивалент \0A)
\r	Знак возврата каретки (ASCII CR, эквивалент 0D)
\t	Табуляция (ASCII HT, эквивалент \09)
\s	Пробел (эквивалент \20)
\\	Обратный слэш (ASCII \, эквивалент \5S)

Скрытое отображение (Password Display)	Переводит элемент управления или индикации в режим показа «звездочки» вместо каждого введенного символа
Шестнадцатеричное отображение (Hex Display)	Преобразует строку алфавитно-цифровых знаков в шестнадцатеричные символы
Ограничить до одной строки (Limit to Single Line)	Выбор функции приводит к блокированию ввода символа «возврат каретки», и строка всегда остается единственной. Ввод символа «возврат каретки» приводит к автоматическому завершению ввода текста
Обновлять значение во время ввода (Update Value While Typing)	Выбор функции приводит к обновлению данных во время набора текста. Если эта строка не выбрана, то вводимый текст передается от строкового элемента управления к его терминалу на блок-диаграмме только после щелчка мышью за пределами элемента или по кнопке 

Наряду с терминалами элементов управления и отображения данных выделяют терминалы узлов, которые представляют поля ввода/вывода данных. Количество и тип терминалов узлов на панели блок-диаграммы можно определить по **соединительной панели** (connector panes) узла путем выбора строки контекстного меню **Видимые элементы** ⇒ **Терминалы** (Visible Items ⇒ Terminals). Так, например, схема терминалов функции **Сложить** (Add)  выглядит следующим образом:  . Выходы на соединительной панели функции выделяются более толстым контуром.

При подводе инструмента соединения к терминалу узла на всех его выводах появляются короткие отрезки проводов (stubs), а выбранная область терминала начинает мерцать  . При этом цвет и толщина отрезков проводов характеризуют тип передаваемых через них данных.

Каждое соединение должно быть согласовано по типу выводов и по типу передаваемых и принимаемых данных. Согласование по типу выводов включает следующие аспекты:

- к линии передачи данных должен быть подключен только один источник;
- к одному источнику может быть подключено неограниченное число приемников данных;
- не допускается соединение одних приемников.

Нарушение правил соединения проявляется в сохранении пунктирной соединительной линии после окончания соединения и в разрыве кнопки запуска прибора. Такое состояние в ряде случаев может быть легко устранено, например путем соединения одних приемников с источником данных. В других случаях может потребоваться анализ типов соединяемых выводов элементов.

Перемещение и удаление проводов производится так же, как и любого другого объекта блок-диаграммы. При этом одиночный щелчок инструментом перемещения выделяет прямолинейный участок провода (сегмент), двойной щелчок выделяет весь провод с изгибами до первого узла (ветвь), тройной щелчок выделяет все провода разветвленного соединения. Удаление всех поврежденных (broken) проводов осуществляется с помощью выбора опции **Удалить поврежденные про-**

водники (Remove Broken Wires) в меню **Правка** или с помощью «горячей» клавиши <Ctrl+B>.

При сборке блок-диаграммы ВП, помимо типа, необходимо знать функциональное назначение всех выводов элементов, входящих в состав прибора. Информацию о назначении выводов можно получить с помощью окна **Контекстная помощь** (Context Help), вызываемого с помощью опции **Справка ⇒ Показать контекстную помощь** (Help ⇒ Show Context Help), или с помощью инструмента соединения, подводимого к выводам. Назначение выводов подприбора можно определить, открыв его лицевую панель с помощью двойного щелчка на иконке подприбора или с помощью строки **Открыть лицевую панель** (Open Front Panel) контекстного меню.

Состав и схему расположения терминалов самого ВП на соединительной панели можно просмотреть, как было отмечено в разделе 1.1, с помощью выбора строки **Показать соединительную панель** (Show Connector) контекстного меню иконки ВП, находящейся в правом верхнем углу лицевой панели. Если после этого инструментом соединения последовательно выделять щелчком терминалы на соединительной панели, то на лицевой панели при этом пунктирной линией будут выделяться связанные с данным терминалом элементы. В режиме просмотра соединительной панели с помощью контекстного меню данной панели **Этот вывод является…** (This Connection Is...) можно определить тип вывода – **Обязательный** (Required), **Рекомендуемый** (Recommended) или **Необязательный** (Optional). Подписи входов указанных типов отличаются форматированием текста в окне контекстной помощи. Далее в справочнике при описании функций подписи к обязательным входам на рисунках функций выделены полужирным шрифтом, а текст пояснений к рекомендуемым и необязательным входам отличается уменьшенным размером шрифта.

Можно произвести и обратное действие – связать элемент лицевой панели с выводом терминала соединительной панели. Потребность в таких действиях возникает при формировании нового подприбора, когда необходимо определить его входы и выходы, то есть связать элементы управления и индикаторы с входными и выходными терминалами соединительной панели.

Сам новый подприбор может быть сформирован из фрагмента блок-диаграммы разработанного ВП путем очерчивания инструментом перемещения прямоугольника, охватывающего элементы, которые должны быть включены в состав подприбора, и выбора строки меню **Правка ⇒ Создать подприбор** (Edit ⇒ Create SubVI). Если после формирования подприбора сохранить его в папке библиотеки пользователя (\User.lib), то при новом запуске LabVIEW данный подприбор будет доступен в подпалитре **Библиотеки пользователя** (User Libraries) палитры функций панели блок-диаграммы.

Сохранение ВП производится с помощью одной из опций меню **Файл**. Опция **Сохранить** (Save) позволяет сохранить измененный ВП с сохранением имени и местоположения на диске или же сохранить новый ВП с указанием этих параметров в диалоговом окне. Опция **Сохранить как** (Save as) позволяет изменить имя ВП или его местоположение при сохранении, при этом исходная версия ВП на диске не изменяется. При выборе опции **Сохранить с опциями** (Save with Options) выводится диалоговое окно, в котором пользователь может выбрать ряд вариантов сохранения ВП.

Если сохранение ВП сопровождается вызовом диалогового окна сохранения, то пользователь может выбрать вариант сохранения в виде индивидуального файла или в составе библиотеки ВП.

Для отладки ВП в LabVIEW предусмотрены такие средства, как информационное окно **Список ошибок** (Error List), выводимое при нажатии неисправной кнопки **Запуск** (Run), кнопки пошагового выполнения, подсветка при выполнении программы (Execution Highlighting), инструменты **Пробник** (probe) и **Точки останова** (breakpoints).

На рис. 1.16 показан вид блок-диаграммы ВП **Использование вставляемых пробников** (Using Supplied Probes) из набора примеров NI Example Finder LabVIEW.

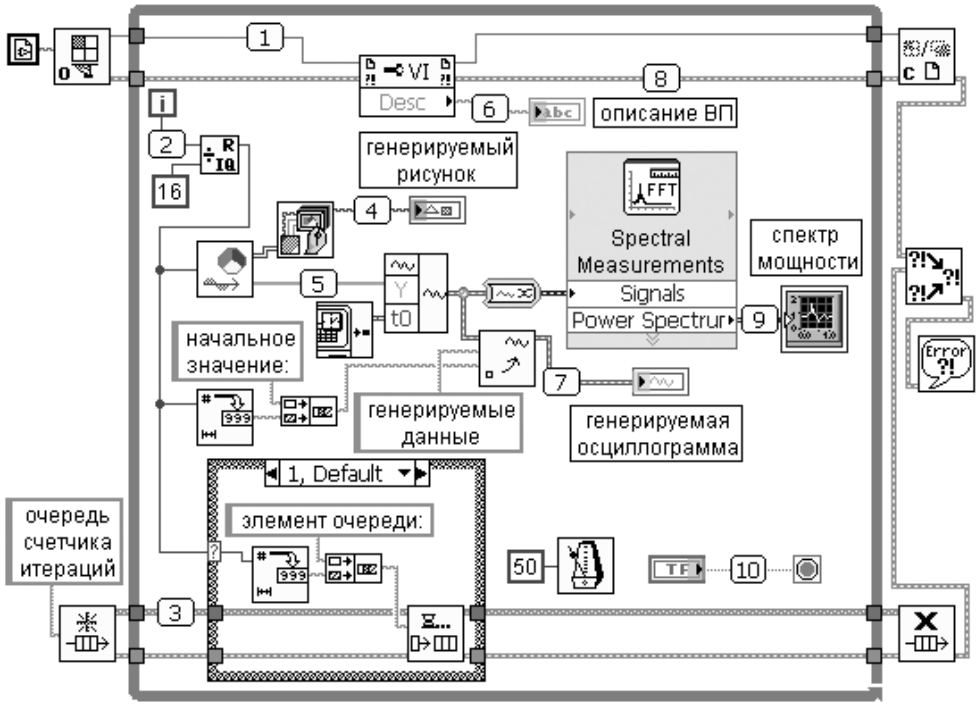


Рис. 1.16. Вид блок-диаграммы ВП ***Использование вставляемых пробников***

Установка пробников производится с помощью выбора опций Probe или Custom Probe контекстного меню провода. Ниже даны пояснения к пробникам различного типа:

1. **Пробник ссылки ВП** (VI Refnum Probe). Показывает путь ВП, путь владельца и числовую ссылку.

2. **Пробник числа в форме I32 с условием** (Conditional Signed32 Probe). Показывает числовые скалярные данные и обеспечивает точку останова по условию.
3. **Пробник ссылки очереди строк** (String Queue Refnum Probe). Показывает различную информацию об очереди, такую как имя и вставленные элементы.
4. **Пробник рисунка** (Picture Probe). Показывает рисунок.
5. **Пробник массива чисел в форме DBL с условием** (Conditional Double Array Probe). Показывает числовой массив данных и обеспечивает точку останова по условию.
6. **Пробник строки с условием** (Conditional String Probe). Показывает строковые данные и обеспечивает точку останова по условию.
7. **Пробник аналоговой осциллограммы** (Analog Waveform Probe). Показывает график осциллограммы, набор данных и атрибуты осциллограммы.
8. **Пробник ошибки с условием** (Conditional Error Probe). Показывает статус ошибки, код и описание. Обеспечивает точку останова по условию.
9. **Пробник данных динамического типа** (Dynamic Data Type Probe). Показывает график динамических данных, набор данных и атрибуты.
10. **Пробник логических данных с условием** (Conditional Boolean Probe). Показывает логические данные и обеспечивает точку останова по условию.

1.3. Структуры, массивы и графические индикаторы среды LabVIEW

В составе подпалитры **Структуры** (Structures) среды LabVIEW можно выделить две группы структур, имеющих ряд общих черт. Первая группа – это структуры **Цикл с фиксированным числом итераций** (For Loop) и **Цикл по условию** (While Loop), вторая группа – структура **Вариант** (Case Structure), структура **Стековая последовательность** (Stacked Sequence Structure) и структура **Открытая последовательность** (Flat Sequence Structure). Именно в таком сочетании происходит взаимная замена данных структур при выборе строки **Заменить на**... (Replace with...) из контекстного меню структуры.

Помимо перечисленных, в состав палитры также входят структуры узел **Формула** (Formula Node), структура **Событие** (Event Structure), **Структура отключения диаграммы** (Diagram Disable Structure), **Структура отключения по условию** (Conditional Disable Structure), **Разделяемая переменная** (Shared Variable), **Локальная переменная** (Local), **Глобальная переменная** (Global) и узел **Обратная связь** (Feedback Node).

Группа **Тактируемых структур** (Timed Structures) размещена в отдельной подпалитре.

Рассмотрим более подробно структуры LabVIEW в порядке их упоминания в предыдущих абзацах.

Структура **Цикл с фиксированным числом итераций** (For Loop) эквивалентна текстовому оператору **for** i = 0 **to** N-1 **do**....

При помещении структуры на панель блок-диаграммы ее контур в виде прямоугольника должен быть растянут так, чтобы охватить существующий код программы, который должен выполняться циклически заданное число раз, или так, чтобы позволить разместить в нем новый код программы. Если помещаемая в структуру или перемещаемая внутри структуры функция пересекается с ее границей, то граница автоматически расширяется. Эта опция может быть отключена для данной структуры путем снятия отметки строки Auto Grow в контекстном меню структуры или для всего приложения путем снятия флажка **Установить структуры с автоматическим расширением** (Place structures with Auto Grow enabled) в окне категории **Блок-диаграмма** (Block Diagram) диалогового окна **Опции** (Options), вызываемого в меню **Инструменты** (Tools).

Количество циклов может задаваться с помощью константы или элемента управления, подключенных к **терминалу числа итераций** (count terminal) (прямоугольник в левом верхнем углу структуры с буквой N). Текущее число завершенных итераций цикла содержится в **терминале счетчика итераций** (iteration terminal).

В структуре цикла для передачи данных из одной итерации цикла в следующую могут быть установлены **Сдвиговые регистры** (Shift Register) (СР) (рис. 1.17).

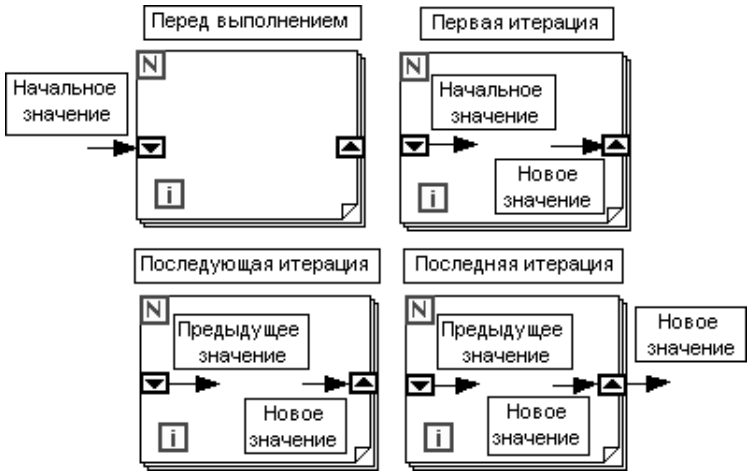


Рис. 1.17. Принцип работы сдвиговых регистров

Установка СР производится на вертикальных сторонах структуры с помощью опции **Добавить сдвиговый регистр** (Add Shift Register), выбираемой из контекстного меню структуры при щелчке правой кнопкой мыши на одной из этих сторон. Тип и размерность СР определяются типом и размерностью данных, подключаемых к правому (входному) терминалу СР. Количество разрядов (тактов задержки) левого терминала СР можно изменить с помощью инструмента перемещения или опций **Добавить элемент** (Add Element), **Удалить элемент** (Remove Element) контекстного меню структуры. По умолчанию при первом запуске ВП

начальные значения левых терминалов СР равны значению по умолчанию для заданного типа данных регистра. Если эти терминалы не были подключены, то при последующих запусках в них будут находиться значения, оставшиеся от предыдущих запусков ВП. Для инициализации левых терминалов СР к ним необходимо подключить константу, элемент управления или функцию инициализации массивов.

Единичные СР могут быть преобразованы в тоннели с помощью опции **Заменить тоннелем** (Replace with Tunnel). После замены тоннель на правой вертикальной границе структуры цикла будет представлять терминал выхода данных из цикла, а тоннель на левой границе – терминал входа данных. Терминал выхода данных, по умолчанию находящийся в состоянии **Включить индексирование** (Enable Indexing), служит точкой, в которой происходят накопление данных и повышение их размерности при их передаче за пределы структуры цикла после окончания ее выполнения. Так, в частности, скалярные данные преобразуются в одномерный массив, одномерный массив – в двумерный массив и т. д. При выборе строки **Отключить индексирование** (Disable Indexing) контекстного меню терминала повышения размерности данных не происходит и за пределы структуры цикла передается последний элемент данных (рис. 1.20). Более подробно индексирование входных и выходных данных структуры цикла будет рассмотрено далее при описании массивов.

Удаление структуры цикла без удаления содержащегося в ней кода производится с помощью строки **Удалить цикл с фиксированным числом итераций** (Remove For Loop) контекстного меню структуры. Для удаления структуры цикла с содержимым необходимо, как обычно, выделить ее инструментом перемещения.

Примеры ВП с использованием структуры цикла с фиксированным числом итераций и сдвиговых регистров приведены на рис. 1.18 и 1.19.

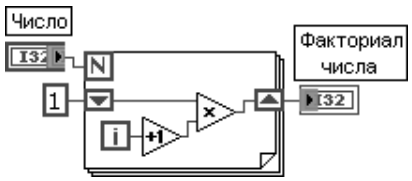


Рис. 1.18. Блок-диаграмма ВП вычисления факториала числа

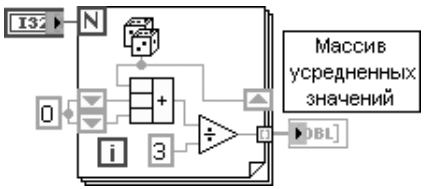


Рис. 1.19. Блок-диаграмма ВП усреднения последовательности случайных чисел

Помимо сдвигового регистра, в структуре цикла может устанавливаться структура узел **Обратная связь** (Feedback Node) (рис. 1.20). Узел Обратная связь автоматически появляется в циклах с фиксированным числом итераций и в циклах по условию при соединении поля вывода данных подприбора, функции или группы подприборов и функций с полем ввода тех же самых подприборов, функций или их групп. Узел Обратная связь может быть заменен сдвиговым регистром с помощью строки **Заменить сдвиговым регистром** (Replace with Shift Register) контекстного меню узла. Пример применения узла Обратная связь для формирования возрастающих знакопеременных чисел приведен на рис. 1.20.

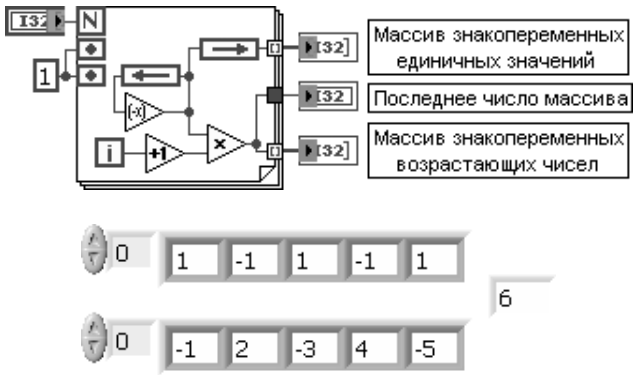


Рис. 1.20 Блок-диаграмма и лицевая панель ВП расчета знакопеременных чисел

Структура **Цикл по условию** (While Loop) эквивалентна следующему псевдокоду: **do** {программа} **while** {условие} ...

Внутри структуры размещаются **терминал счетчика итераций** (iteration terminal) **i** и **терминал условия выхода из цикла** (conditional terminal). Вид структуры с терминалом условия по умолчанию приведен на рис. 1.21а. Код программы, размещенный в структуре, выполняется до подачи на терминал условия логической переменной ИСТИНА (TRUE) в структуре на рис. 1.21а или ЛОЖЬ (FALSE) в структуре на рис. 1.21б.

а *б*

*Рис. 1.21. Вид структуры **Цикл по условию** с различными режимами*

Изменение варианта прекращения выполнения производится с помощью опций **Остановить если истина** (Stop If True) или **Продолжить если истина** (Continue If True) контекстного меню терминала условия. Если терминал условия не подключен к какому-либо выходу, то цикл не будет выполняться. В данной структуре также могут быть установлены сдвиговые регистры и структуры узлов обратной связи. Входные и выходные терминалы данных структуры по умолчанию имеют состояние **Отключить индексирование** (Disable Indexing).

Структура **Вариант** (Case Structure) аналогична операторам **if-then-else** или **case** текстовых языков программирования. По умолчанию структура Вариант является логической и имеет два варианта – ИСТИНА (TRUE) и ЛОЖЬ (FALSE), выбираемых с помощью **терминала селектора структуры варианта**. Структура автоматически преобразуется в числовую или строковую при подключении соответственно числовой или строковой переменной к терминалу селектора. В этом случае структура может иметь практически неограниченное количество вариантов, начиная с нулевого. С помощью строк **Добавить вариант после** (Add Case After) или **Добавить вариант перед** (Add Case Before) можно добавить новый вариант после или до текущего варианта. Одновременно можно наблюдать только один вариант (кадр) структуры. Переход между вариантами производится с помощью **селектора структуры варианта**, расположенного на верхней стороне рамки структуры или контекстного меню структуры. Для использования структуры Вариант необходимо отметить **вариант по умолчанию** (Default).

Ввод и вывод данных в структуру Вариант производятся с помощью входных и выходных терминалов данных (тоннелей). Создание выходного терминала данных на одной поддиаграмме структуры приводит к его появлению на других поддиаграммах в том же самом месте границы структуры. До подключения данных к выходному терминалу во всех поддиаграммах он сохраняет белый цвет и воспринимается как ошибка создания структуры. Выходные терминалы должны иметь значения совместимых типов.

Структура **Последовательность** (Sequence Structure) используется для управления порядком выполнения узлов, которые не зависят друг от друга. Структура Последовательность выглядит как набор кадров и обеспечивает последовательное выполнение размещенных в ее кадрах фрагментов программ. Необходимость в такой структуре вызвана потоковым характером выполнения программ в LabVIEW, когда операции в узлах выполняются при поступлении данных на все входы узлов. При необходимости выполнения программы в ином порядке и используется структура Последовательность.

В LabVIEW структура Последовательность представлена двумя вариантами: **Стековая последовательность** (Stacked Sequence Structure) и **Открытая последовательность** (Flat Sequence Structure). Отличие указанных структур проявляется при увеличении числа кадров с помощью строк **Добавить кадр после** (Add Frame After) или **Добавить кадр перед** (Add Frame Before) (рис. 1.22).

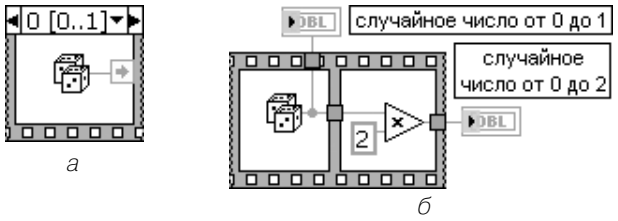


Рис. 1.22. Вид структуры Стековая последовательность (а) и структуры Открытая последовательность (б)

Так же как и в структуре Вариант, ввод и вывод данных производятся с помощью входных и выходных терминалов, при этом данные из входного терминала доступны во всех кадрах. Для передачи данных внутри структуры Стековая последовательность (рис. 1.22а) используется **терминал локальной переменной** (Sequence Local). Терминал локальной переменной создается с помощью строки **Добавить локальную переменную** (Add Sequence Local) контекстного меню структуры. В исходном состоянии терминал локальной переменной, появляющийся в текущем и других кадрах структуры, пуст. После подключения источника данных к локальной переменной в текущем и последующих кадрах появляется стрелка, указывающая направление передачи данных. Данные на выходах структуры Стековая последовательность появляются только после окончания ее выполнения.

В отличие от этого в структуре **Открытая последовательность** (рис. 1.22б) данные между кадрами передаются без дополнительных переменных и выводятся на выход по мере выполнения кода кадров.

Структура узел **Формула** (Formula Node) представляет перестраиваемый по размеру прямоугольник, в котором можно в текстовом виде записывать математические выражения и операторы. Узел Формула целесообразно использовать при наличии множества переменных или выполнении сложных расчетов. Выражения записываются с использованием функций и операторов, перечисленных в двух последующих в таблицах. В первой таблице приведены имена и иконки аналогичных функций языка **G**. Запись каждой формулы должна заканчиваться символом «**;**». Имена входных и выходных переменных вводят, соответственно, во входные и выходные терминалы, формируемые с помощью строк **Добавить ввод** (Add Input) и **Добавить вывод** (Add Output) контекстного меню структуры. Имена вписываются в зачерненный прямоугольник непосредственно после вызова терминала. При записи имен необходимо соблюдать условие одинаковости регистра.

Список функций узла Формула приведен в следующей таблице.

Функция узла	Имя аналогичной функции языка G	Числовая функция	Описание функции
abs(x)	Absolute Value		Абсолютное значение **x**
acos(x)	Inverse Cosine		Арккосинус **x** (рад)
acosh(x)	Inverse Hyperbolic Cosine		Гиперболический арккосинус **x** (рад)
asin(x)	Inverse Sine		Арксинус **x** (рад)
asinh(x)	Inverse Hyperbolic Sine		Гиперболический арксинус **x** (рад)
atan(x)	Inverse Tangent		Арктангенс **x** (рад)

Функция узла	Имя аналогичной функции языка G	Числовая функция	Описание функции
atan2(x,y)	Inverse Tangent (2 Input)		Арктангенс отношения **x/y** (рад)
atanh(x)	Inverse Hyperbolic Tangent		Гиперболический арктангенс **x** (рад)
ceil(x)	Round to +Infinity		Округление до большего целого
ci(x)	Cosine Integral		Рассчитывает интегральный косинус для любого неотрицательного числа **x**
cos(x)	Cosine		Косинус **x**
cosh(x)	Hyperbolic Cosine		Гиперболический косинус **x**
cot(x)	Cotangent		Котангенс **x** (рад)
csc(x)	Cosecant		Косеканс **x**
exp(x)	Exponential		Экспонента **x**
expm1(x)	Exponential (Arg) - 1		Экспонента **x-1**
floor(x)	Round to -Infinity		Округление до меньшего целого **x**
getexp(x)	Mantissa & Exponent (Data Manipulation)		Экспонента числа **x** в случае его представления числом по основанию 2
gamma(*x*)	Gamma		Вычисляет Гамма-функцию или неполную Гамма-функцию
getman(x)	Mantissa & Exponent (Data Manipulation)		Мантисса числа **x** в случае его представления числом по основанию 2
int(x)	Round To Nearest integer		Округление до ближайшего целого **x**
intrz(x)	Round Toward 0	–	Округление до меньшего целого между **x** и 0
ln(x)	Natural Log		Натуральный логарифм **x**
lnp1(x)	Natural Logarithm (Arg +1)		Натуральный логарифм **x+1**
log(x)	Logarithm Base 10		Десятичный логарифм **x**

Функция узла	Имя аналогичной функции языка G	Числовая функция	Описание функции
log2(x)	Logarithm Base 2		Двоичный логарифм **x**
max(x,y)	Max & Min		Максимальное из **x**,**y**
min(x,y)	Max & Min		Минимальное из **x**,**y**
mod(x,y)	Quotient & Remainder		Остаток **x** при округлении частного от деления **x/y** до меньшего целого
rand	Random Number(0-1)		Случайное число от 0 до 1
rem(x)	Remainder		Остаток **x** при округлении частного от деления **x/y** до ближайшего целого
pow(x,y)	Power Of X		Возведение **x** в степень **y**
si(*x*)	Sine Integral		Рассчитывает интегральный синус для любого неотрицательного числа **x**
sec(x)	Secant		Секанс **x**
sign(x)	Sign		Знаковая функция: равна 1 при **x>0**, равна 0 при **x=0**, равна –1 при **x<0**
sin(x)	Sine		Синус **x**
sinc(x)	Sinc		(Синус **x**)**/x**
sinh(x)	Hyperbolic Sine		Гиперболический синус **x**
spike(x)	Spike Function		Генерирует импульсную функцию для любого вещественного числа **x**
sqrt(x)	Square Root		Квадратный корень **x**
step(*x*)	Step Function		Генерирует ступенчатую функцию для любого вещественного числа **x**
tan(x)	Tangent		Тангенс **x**
tanh(x)	Hyperbolic Tangent		Гиперболический тангенс **x**

Синтаксис узла Формула описан в приложении 1.
Список операторов узла Формула (по приоритету) приведен в таблице.

Символ	Значение
**	Возведение в степень
+, -, !, ~, ++, ––	Унарный плюс, унарный минус, логическое НЕ, дополнение бита, пред- и постинкремент, пред- и постдекремент
*, /, %	Умножение, деление, модуль (остаток)
+ и –	Сложение и вычитание
>> и <<	Арифметический сдвиг вправо и сдвиг влево
>, <, >=, <=	Больше, меньше, больше или равно, меньше или равно
!= и ==	Неэквивалентность и эквивалентность
&	Битовое И
^	Битовое исключающее ИЛИ
|	Битовое ИЛИ
&&	Логическое И
||	Логическое ИЛИ
? :	Условное выражение
= оп=	Присваивание Укороченная операция и присваивание оп может быть +, –, *, /, >>, <<, &, ^, |, % **

Примеры записи операторов в узле Формула приведены на рис. 1.23.

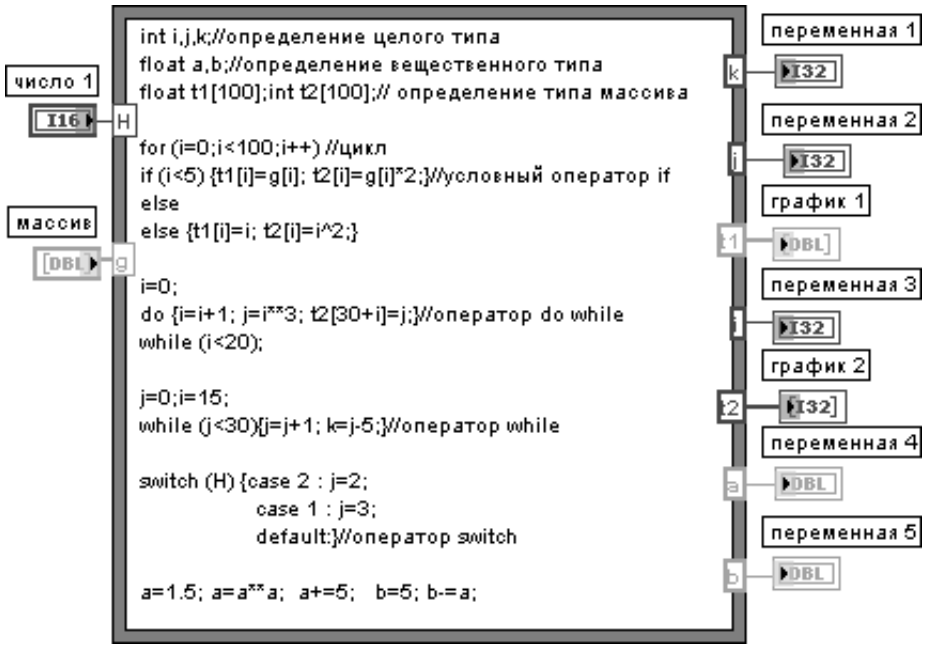


Рис. 1.23. Примеры записи операторов в узле Формула

Структура **Событие** (Event Structure) имеет одну или более поддиаграмм или вариантов событий, из которых только один выполняется при обращении к структуре. Структура Событие ожидает наступления события на лицевой панели, после чего выполняет соответствующий вариант с целью обработки события. С помощью контекстного меню структуры можно добавить новые варианты событий или определить вид обрабатываемого события. Подключение значения к терминалу времени ожидания, находящемуся в левом верхнем углу структуры, позволяет задать величину интервала ожидания события структурой в миллисекундах. По умолчанию значение на входе этого терминала равно –1, что соответствует отсутствию времени ожидания. Более подробно данная структура рассмотрена в разделе 2.2.1.

Структуры и ВП из подпалитры **Тактируемые структуры** (Timed Structures) используются для повторения фрагментов кода или для выполнения кода в определенном порядке с заданными задержками или временными интервалами. Их следует использовать для тактирования с разной частотой или для точного тактирования, для обеспечения обратной связи при выполнении цикла, у которого временные характеристики изменяются динамически, или для поддержки нескольких уровней приоритетов выполнения. Каждая такая структура имеет характерную рамку с изменяемыми размерами, которая охватывает область кода, выполняющуюся в соответствии с правилами структуры. Эта часть кода называется поддиаграммой.

В подпалитру **Тактируемые структуры** входят структуры **Тактируемый цикл** (Timed Loop) и **Тактируемая последовательность** (Timed Sequence), а также ряд соответствующих ВП.

Тактируемые структуры могут состоять из одной или нескольких поддиаграмм (кадров) (рис. 1.24). На левой и правой границе структуры с внешней стороны находятся соответственно **узел ввода** (Input Node) (1) и **узел вывода** (Output Node) (4), а внутри каждого кадра – **левый узел данных** (Left Data node) (2) и **правый узел данных** (Right Data Node) (3). Узлы ввода и вывода обеспечивают запись данных конфигурирования и возвращают из структуры ошибку и информацию о синхронизации. Левый узел данных предоставляет информацию о временных параметрах и параметрах состояния предыдущего и текущего кадра в структуре Тактируемая последовательность, или аналогичную информацию о параметрах предыдущей итерации цикла в структуре Тактируемый цикл. Если подключить выходы левого

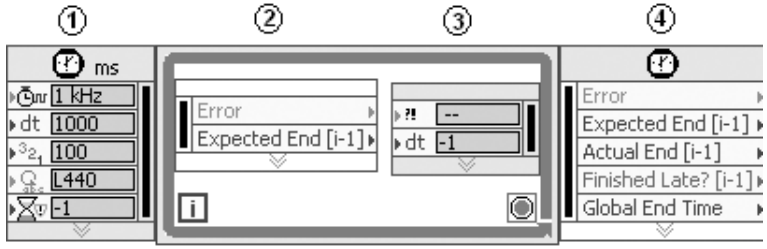


Рис. 1.24. Вид структуры Тактируемый цикл

узла данных к соответствующим входам правого узла, то тем самым будет обеспечено динамическое изменение опций следующего кадра или итерации.

С помощью контекстного меню структуры можно добавлять, удалять, вставлять или объединять кадры. В свою очередь, контекстное меню **узла ввода** (Input Node) позволяет с помощью строки **конфигурировать узел ввода** (Configure Input Node) открыть диалоговое окно **конфигурировать Тактируемый цикл** (Configure Timed Loop) и настроить параметры структуры. Альтернативным способом настройки является подключение элементов управления или констант к терминалам узла ввода. По умолчанию узлы структур отображаются в сокращенном виде. Полное раскрытие узлов можно выполнить с помощью строки **показать все входы (выходы)** (Show All Inputs (Outputs)) контекстного меню.

При наличии на блок-диаграмме нескольких тактируемых структур каждая из них выполняется в отдельном **потоке** (thread), и для регулирования их выполнения во времени используется механизм **приоритетов** (priority). Структура с более высоким приоритетом выполняется в первую очередь. Число, определяющее приоритет, лежит в диапазоне от 1 до 2 147 480 000. Приоритет тактируемых структур по отношению к приоритету ВП находится посередине между **высоким приоритетом** (high priority) и **критичным по времени** (самым высоким) **приоритетом**. Это значит, что тактируемые структуры будут выполняться в первую очередь, если ВП не сконфигурировано на критичный по времени приоритет.

Для регулирования времени выполнения также используется задание **временного сдвига** (Offset) и **периода**.

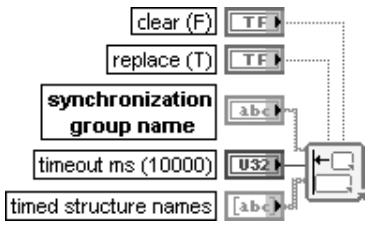
Ниже в таблице приведены ВП из подпалитры **Тактируемые структуры**.

Create Timing Source

Создать источник синхронизации

Этот полиморфный ВП создает источник синхронизации, который используется в Тактируемом цикле (Timed Loop). Каждый источник синхронизации имеет свою единицу синхронизации и/или время запуска и не начинает работать до тех пор, пока не запустится первый использующий его Тактируемый цикл. Выбор экземпляра ВП производится с помощью селектора или контекстного меню

Synchronize Timed Structure Starts



Синхронизовать начало выполнения тактируемых структур

ВП синхронизирует начало выполнения **Тактируемых циклов** (Timed Loops) или **Тактируемых последовательностей** (Timed Sequences), названия которых задаются на входе **имена тактируемых структур** (timed structure names), путем добавления этих имен к группе синхронизации, которая определяется на входе **имя группы синхронизации** (synchronization group name). Все тактируемые структуры в группе синхронизации ожидают готовности к исполнению всех структур.

При установке на входе **очистить (F)** (clear (F)) состояния ИСТИНА производятся стирание всех тактируемых структур, указанных на входе **имя группы синхронизации**, и удаление всей группы перед добавлением к группе тактируемых структур, заданных на входе **имена тактируемых структур**. Такая установка требуется для удаления любых тактируемых структур, которые не соответствуют Тактируемому циклу. По умолчанию установлено состояние ЛОЖЬ.

Вход **заменить (T)** (replace (T)) заменяет тактируемые структуры, указанные в **имени группы синхронизации**, на аналогичные структуры, введенные на входе **имена тактируемых структур**. Если на входе **заменить** установлено состояние ЛОЖЬ и источник тактирования относится к другой иерархии, ВП возвращает ошибку.

Вход **имя группы синхронизации** (synchronization group name) представляет название группы тактируемых структур, которые предполагается синхронизировать.

Вход **время ожидания, мс (10000)** (timeout ms (10000)) задает интервал времени, в течение которого тактируемая структура ожидает прихода других членов группы синхронизации в точку синхронизации. Если любая из тактируемых структур группы не приходит вовремя, то тактируемая структура возвращает ошибку. Задание –1 на этом входе приводит к неограниченному ожиданию.

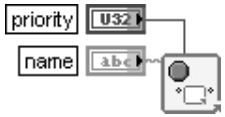
Вход **имена тактируемых структур** (timed structure names) определяет имена тактируемых структур, которые предполагается синхронизировать. По умолчанию ВП удаляет тактируемую структуру из текущей группы и добавляет ее к группе, заданной в **имени группы синхронизации**. Если на входе **заменить (T)** установлено состояние ЛОЖЬ и если тактируемая структура, заданная на входе **имена тактируемых структур,** является членом другой группы, то ВП возвращает ошибку, которая указывает на то, что он не может добавить тактируемую структуру к группе синхронизации

Build Timing Source Hierarchy

Создать иерархию источников синхронизации

ВП создает иерархию источников синхронизации на основе имен, которые введены на входе **имена источников синхронизации** (timing source names). Иерархия определяет порядок, в котором начинают работать источники синхронизации. Родительский источник синхронизации не начинает работать до тех пор, пока не начнут работать источники синхронизации, указанные на входе **имя иерархии** (hierarchy name). Этот ВП целесообразно использовать при наличии источников синхронизации, у которых имеются зависимости сигналов, например когда счетчики DAQ управляют соединениями аналогового ввода. В этом случае источник синхронизации на основе счетчика действует как родитель

Stop Timed Structure



Остановить тактируемую структуру

ВП останавливает Тактируемый цикл или Тактируемую последовательность, название которой указано на входе **имя** (name). При попытке остановки выполняющегося Тактируемого цикла эта структура выполняет следующую итерацию и возвращает сообщение **прервано** (Aborted) на выходе **Причина таймерного включения** (Wakeup Reason) **левого узла данных** (Left Data node)

Clear Timing Source

Очистить источник синхронизации

ВП останавливает и удаляет источник синхронизации, который был создан или определен для использования другим ресурсом. Если источник синхронизации связан с задачей DAQmx, ВП также очищает и задачу. ВП не может повторно использовать имя до тех пор, пока не завершатся все Тактируемые циклы, привязанные к источнику синхронизации

Структура **Локальная переменная** (Local Variable) используется для обеспечения доступа к элементам управления или индикации из более чем одной точки блок-диаграммы. Локальные переменные всегда содержат действительные значения объектов лицевой панели, с которыми они ассоциированы. Локальную переменную можно устанавливать в режим как записи, так и чтения данных. Таким образом, с помощью локальной переменной можно записывать данные в элемент управления или считывать данные с индикатора. Помимо этого, для одного элемента управления можно определить несколько локальных переменных, установленных в режим записи или считывания. Таким способом можно управлять параллельными циклами с помощью одной переменной.

Наиболее просто локальная переменная элемента управления или индикатора может быть создана с помощью строки **Создать ⇒ Локальная переменная** (Create ⇒ Local Variable) контекстного меню этих элементов или их терминалов. Локальная переменная также может быть установлена из палитры **Структуры**, однако при этом ее необходимо связать с определенным элементом управления или индикатором с помощью опции **Выбрать элемент** (Select Item) контекстного меню локальной переменной. Выбираемый элемент должен иметь ярлык.

Структура **Глобальная переменная** (Global Variable) используется для передачи данных между одновременно работающими ВП. Глобальная переменная является встроенным элементом LabVIEW, имеющим лицевую панель, но не имеющим диаграмму. В зависимости от направления передачи и типа данных на лицевой панели глобальной переменной устанавливаются один или несколько элементов управления, которые идентифицируются своими ярлыками. Таким образом, лицевая панель глобальной переменной является контейнером, с помощью которого различные ВП могут обмениваться данными.

Глобальная переменная выбирается из палитры **Структуры**. После установки на блок-диаграмме с помощью контекстного меню или двойного щелчка на терминале глобальной переменной необходимо открыть ее лицевую панель, установить необходимые элементы управления и индикаторы и сформировать их ярлыки. Выбор ярлыка из списка в строке **Выбрать элемент** (Select Item) контекстного меню терминала глобальной переменной позволяет связать эту переменную с соответствующим элементом управления или индикатором.

Массив LabVIEW – это набор индексированных данных одного типа. Он может иметь любую размерность и содержать до 2^{31} элементов на размерность. Элементом массива может быть любой тип данных, за исключением массива, таблицы или графика. Доступ к элементам осуществляется с помощью индексов. Значения индексов лежат в диапазоне от 0 до N-1, где N – количество элементов массива.

В LabVIEW массивы могут быть созданы как вручную на лицевой панели или на панели блок-диаграммы, так и программно. На лицевой панели могут быть созданы массивы элементов управления или индикаторов, на панели блок-диаграммы – массивы констант. Программно массивы создаются с помощью структур и соответствующих функций.

Для формирования массивов на лицевой панели необходимо разместить на ней **шаблон массива** (array shell) из подпалитры **Массив и кластер** (Array & Cluster) палитры элементов управления (рис. 1.25а). При этом терминал массива имеет черный цвет и отображает пустые скобки. В **окно отображения элемента** может быть помещен **объект данных** – элемент управления или индикатор в соответствии с типом формируемого массива, за исключением типов, перечисленных выше. Помещение объекта сопровождается мерцанием оболочки, а его фиксация в окне приводит к присвоению терминалу массива цвета, типа и надписи, присущих помещенному объекту. После задания типа массива он может использоваться для ввода или вывода данных.

Аналогичным образом создается массив констант на блок-диаграмме. Для создания массива констант необходимо разместить на диаграмме **шаблон массива констант** (Array Constant) из палитры **Массив** (Array) и поместить в него константу необходимого типа.

На рис. 1.25 показан вид массивов различного типа на лицевой панели и соответствующих терминалов на блок-диаграмме: массив числовых элементов управления (б), массив логических индикаторов (в), массив строковых элементов управления (д) и двумерный массив числовых индикаторов (г).

Для программного формирования и обработки массивов используются рассмотренные выше структуры **Цикл с фиксированным числом итераций** (For Loop) и **Цикл по условию** (While Loop), имеющие в своем составе индексную переменную **i**.

Если элементы массива формируется в структуре цикла с фиксированным числом итераций, то их преобразование в массив происходит в терминале вывода данных, находящемся по умолчанию в режиме **Включить индексирование** (Enable Indexing) (рис. 1.26).

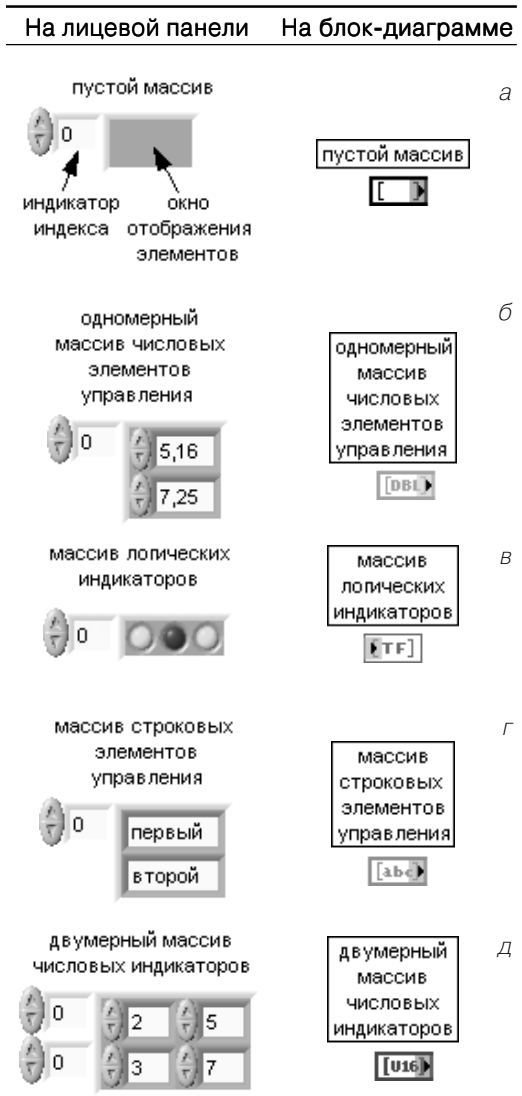


Рис. 1.25. Вид массивов на лицевой панели и их терминалов

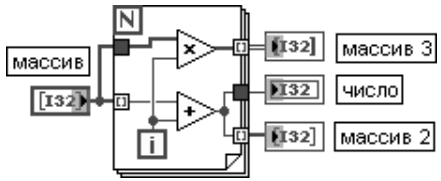


Рис. 1.26. Варианты ввода и вывода массивов в структуре цикла с фиксированным числом итераций

В этом режиме терминал вывода данных представляет пустотелый двойной квадрат. Формирование массива сопровождается изменением толщины провода с данными после его выхода из структуры. Выключение режима индексирования осуществляется с помощью строки **Отключить индексирование** (Disable Indexing) контекстного меню терминала. При этом из структуры будет выводиться только последний элемент.

Описанный режим автоматического индексирования и аккумулирования массивов на границе цикла путем добавления одного нового элемента в каждой итерации цикла называется **автоиндексированием** (auto-indexing).

При вводе массива в структуру цикла терминал ввода данных по умолчанию также находится в состоянии **Включить индексирование** (рис. 1.26). В этом режиме терминал передает в цикл по одному элементу в каждую итерацию. Тогда структура автоматически определяет размер массива и нет необходимости задавать значение терминала числа итераций. При подаче на вход нескольких массивов разной длины структура настраивается на самый короткий массив. То же самое происходит и при одновременно подключенном терминале числа итераций.

Если терминал ввода массива перевести в состояние **Отключить индексирование**, то массив будет вводиться в цикл целиком (рис. 1.26). При этом, естественно, способность автоиндексирования теряется и значение числа циклов должно быть задано явно.

На рис. 1.26 видно, что при выводе одномерного массива из цикла в режиме автоиндексирования он преобразуется в двумерный массив. Массив такой же размерности может быть создан на основе скалярных переменных при использовании структуры, содержащей внутренний и внешний циклы. Увеличение размерности массива, созданного оператором на лицевой панели или в блок-диаграмме, может быть выполнено с помощью строки **Добавить размерность** (Add Dimension) контекстного меню элемента управления/отображения **индекса массива**. Такое же действие может быть выполнено и инструментом перемещения.

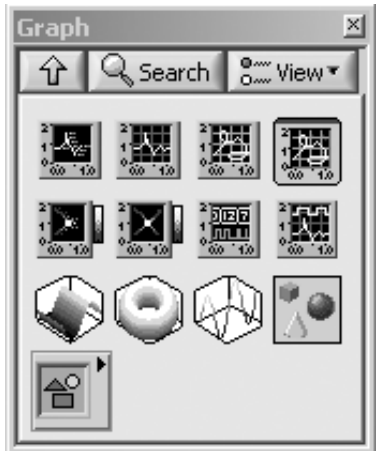


Рис. 1.27. Палитра Графики

В структуре **Цикл по условию** ситуация с индексацией противоположная, то есть по умолчанию терминалы ввода и вывода данных находятся в состоянии **Отключить индексирование**.

Наряду с описанными выше элементами индикации значений массивов, представляющими упорядоченные наборы элементов индикации соответствующих скаляров, образующих массив, в LabVIEW для отображения массивов и наборов числовых данных применяются специальные графические индикаторы – **Графики** (Graph) (рис. 1.27).

В состав палитры Графики входят следующие графические индикаторы.

Развертка осциллограммы (Waveform Chart) – графический индикатор, имитирующий работу самописца. Поступающие данные нумеруются по оси абсцисс целыми числами. Поскольку индикатор запоминает всю подаваемую на него информацию в виде отдельных чисел, он устанавливается внутри структур **Цикл с фиксированным числом итераций** и **Цикл по условию** (рис. 1.28). Индикатор может быть многолучевым и многоэкранным.
Для отображения двух и более наборов данных от разных источников необходимо объединить их в кластер с помощью функции **Объединить** (Bundle). Стирание информации производится с помощью строки **Операции с данными ⇒ Очистить развертку** (Data Operations ⇒ Clear Chart) контекстного меню графика

Рис. 1.28. Блок-диаграмма ВП контроля температуры с индикаторами Развертка температуры и График температуры

График осциллограммы (Waveform Graph) – графический индикатор, имитирующий работу осциллографа. Он принимает данные в виде массива чисел и отображает их с равномерным шагом. Для отображения двух и более массивов данных они должны быть объединены в двумерный массив с помощью функции **Сформировать массив** (Build Array). Для отображения графика с заданной начальной точки и с заданным шагом предусмотрено формирование кластера из трех элементов с помощью функции **Объединить** (Bundle) на верхний вход которой подается начальное смещение, а на средний – шаг отображения (рис. 1.28)

Двухкоординатный график (XY Graph) – графический индикатор, позволяющий отображать функциональные зависимости $y = f(x)$. Для отображения на данном индикаторе массива точек с произвольными координатами по осям необходимо сформировать массив кластеров или объединить два массива координат **X** и **Y** в кластер. С целью отображения двух и более графиков необходимо

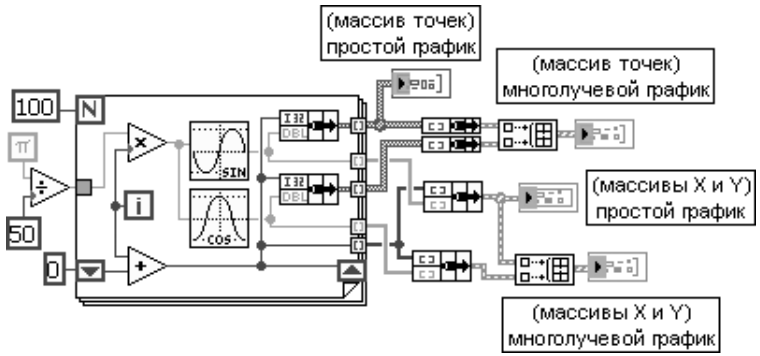


Рис. 1.29. Блок-диаграмма ВП XY Graph из набора примеров NI Example Finder

использовать функцию **Сформировать массив** (Build Array) для формирования массива кластеров (рис. 1.29).

Экспресс-ВП Двухкоординатный график (XY Graph)

Графические индикаторы **Развертка интенсивности** (Intensity Chart) и **График интенсивности** (Intensity Graph) служат для отображения двумерных массивов данных в виде графиков интенсивности такой же размерности. В исходном состоянии на **шкале интенсивностей** (Ramp) отображается три цвета – черный, соответствующий нижней половине диапазона, синий, соответствующий верхней половине диапазона, и белый, соответствующий превышению диапазона. Однако при включении режима **интерполяции** (Interpolate Color) в шкале интенсивностей градации интенсивности при переходе от черного к белому становятся плавными. При желании изменить палитру цветов необходимо сформировать **таблицу цветов** (Color Table), то есть рассчитать массив цифровых значений цветов, которые будут соответствовать градациям величины данных, и подключить таблицу к **узлу свойств** графического индикатора (Property Node) с установленным свойством **таблица цветов** (Color Table)

График цифровой осциллограммы (Digital Waveform Graph) служит для отображения массива целых чисел в виде диаграмм логических сигналов, соответствующих двоичным разрядам чисел. Для работы индикатора необходимо сформировать кластер, содержащий начальное значение, шаг, отображаемые данные и число портов

График смешанного сигнала (Mixed Signal Graph) используется для отображения данных разного типа, расположенных на одной оси **x**. График смешанного сигнала принимает все типы данных, которые воспринимаются графиком осциллограммы, двухкоординатным графиком и графиком цифровой осциллограммы. Помимо этого, он принимает любой кластер, который содержит любую комбинацию данных с такими типами.
При добавлении к графику смешанного сигнала отображаемых данных разного типа для каждого из них устанавливается свой масштаб по оси **y**. График смешанного сигнала автоматически создает области построения графиков для размещения наборов аналоговых и цифровых данных.
Панель редактирования графика смешанного сигнала содержит элемент управления Дерево и отображается слева от областей построения графика. Каждый элемент управления Дерево представляет одну область построения графика. Панель редактирования графиков может использоваться для перемещения графиков между областями их построения

Все перечисленные выше графические индикаторы имеют в меню настройки более обширный раздел **Видимые элементы** (Visible Items) и дополнительные строки, связанные с настройкой осей. Так, в частности, в состав меню **Видимые элементы** этих индикаторов входят следующие разделы (рис. 1.30): **Ярлык** (Label), **Заголовок** (Caption), **Панель редактирования графика** (Plot Legend), **Панель редактирования шкалы** (Scale Legend), **Палитра элементов управления графиком** (Graph Palette), **Панель редактирования курсора** (Cursor Legend), **Линейка прокрутки** (Scrollbar), **Шкала X** (X Scale), **Шкала Y** (Y Scale).

Рис. 1.30. Вид графика осциллограммы с панелями и атрибутами

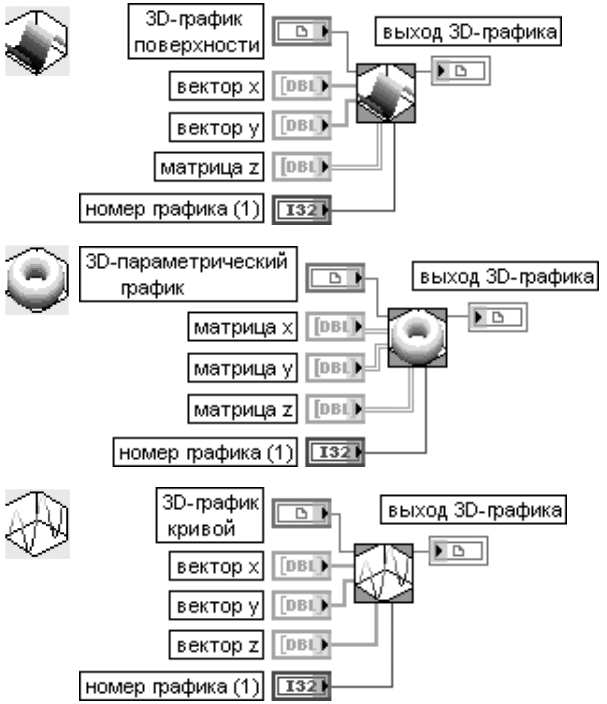
Меню **Видимые элементы** графического индикатора **Развертка осциллограммы** отличается тем, что на месте строки **Панель редактирования курсора** находится строка **Цифровой индикатор**.

Панель редактирования графика может перестраиваться по размеру в вертикальном направлении для настройки параметров набора графиков. Панель редактирования графика имеет свое контекстное меню, которое позволяет настраивать тип графиков, их цвет, ширину и тип линии, тип точек и вид их соединения.

Панель редактирования шкалы позволяет фиксировать или сбрасывать в исходное состояние масштаб по осям, установленный пользователем с помощью одного из инструментов, входящих в состав **палитры элементов управления графиком**. Помимо этого, она позволяет настраивать параметры осей: формат, точность, характер расположения меток, видимость оси и подписи, цвет сетки. Эти и ряд других параметров могут быть также установлены и из контекстного меню осей индикатора.

LabVIEW 8.20 предоставляет для графических индикаторов такие новые возможности, как нанесение аннотаций для точек графика и экспорт упрощенных изображений графиков, разверток, таблиц, цифровых данных и цифровых осциллограмм в форматах .emf, .bmp и .eps. Нанесение аннотаций производится с помощью диалогового окна **Создать аннотацию** (Create Annotation), вызываемого с помощью контекстного меню **Операции с данными ⇒ Создать аннотацию** (Data Operations ⇒ Create Annotation) графика. Экспорт упрощенных изображений осуществляется с помощью диалогового окна **Экспортировать упрощенное изображение** (Export Simplified Image), вызываемого из контекстного меню графика **Операции с данными ⇒ Экспортировать упрощенное изображение** (Data Operations ⇒ Export Simplified Image).

В состав палитры Графики входят также трехмерные графики. Установка таких графиков на лицевой панели сопровождается появлением на блок-диаграмме связки из **ссылки** (Refnum) на элемент управления ActiveX CWGraph3D и соответствующего ВП.



Графический индикатор **3D-график поверхности** (3D Surface Graph) предназначен для отображения двумерных данных в виде поверхности на трехмерном графике

Графический индикатор **3D- параметрический график** (3D Parametric Graph) отображает данные с помощью трех двумерных массивов, характеризующих проекции графика на плоскости **x**, **y** и **z**

Графический индикатор **3D-график кривой** (3D Curve Graph) используется для отображения данных в виде пространственной кривой

Настройка графиков производится с помощью диалогового окна Свойства: CWGraph3D Control, вызываемого с помощью строки CWGraph3D ⇒ Свойства... контекстного меню графика.

Функции программирования LabVIEW

2

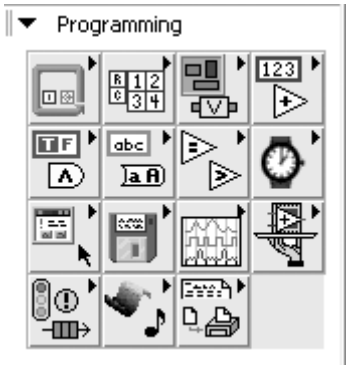


Рис. 2.1. Вид подпалитр функций категории программирования

Функции программирования расположены в категории Programming палитры функций (рис. 2.1) и сгруппированы по 15 подпалитрам. В связи с большим числом и определенным отличием в уровне сложности функций, входящих в состав подпалитр, их целесообразно разделить на две группы: **базовые** и **дополнительные**. К числу базовых функций программирования могут быть отнесены функции, расположенные в следующих подпалитрах: **Числовые** (Numeric), **Логические** (Boolean), **Строковые** (String), **Сравнение** (Comparison), **Массив** (Array), **Кластер и Вариант** (Cluster & Variant), **Установление времени** (Timing) и **Ввод/вывод файлов** (File I/O). Именно в такой последовательности перечисленные функции рассматриваются далее в разделе 2.1. В свою очередь, дополнительные функции программирования включают функции, находящиеся в подпалитрах **Диалог и Интерфейс пользователя** (Dialog & User Interface), **Управление приложением** (Application Control), **Синхронизация** (Synchronization), **Графики и звук** (Graphics & Sound) и **Создание отчета** (Report Generation). Функции из перечисленных подпалитр рассматриваются в разделе 2.2. Содержание и работа структур LabVIEW, находящихся в подпалитре **Структуры** (Structures), были рассмотрены ранее в разделе 1.3. Функции, входящие в подпалитру **Осциллограмма** (Waveform), рассматриваются в главе 5.

2.1. Базовые функции LabVIEW

2.1.1. Числовые функции и функции манипуляции данными

Числовые функции используются для выполнения арифметических и комплексных операций с числовыми данными, а также для преобразования типов числовых данных. На рис. 2.2 показан вид основной палитры числовых функций (рис. 2.2а) и дополнительных подпалитр **Преобразование** (Conversion) (рис. 2.2б), **Функции манипуляции данными** (Data Manipulation) (рис. 2.2в), **Комплексные** (Complex) (рис. 2.2г) и **Математические и научные константы** (Math & Scientific Constants) (рис. 2.2д).

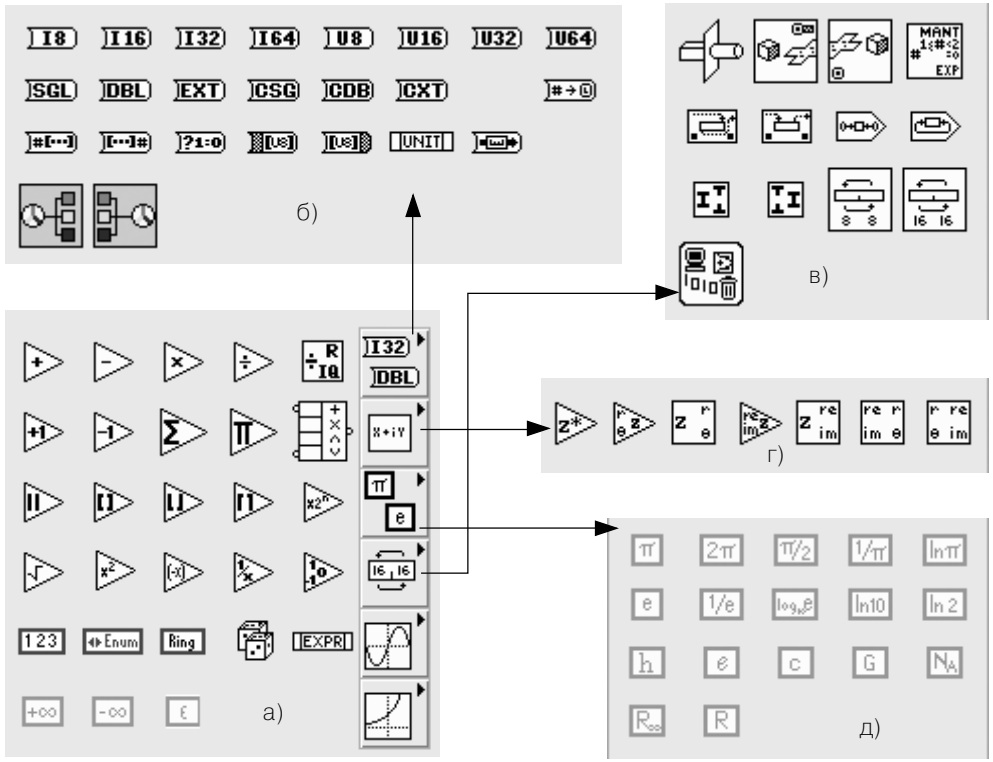


Рис. 2.2. Вид основной палитры (а) и дополнительных подпалитр (б)–(д) числовых функций

Входившие в состав этой палитры в предыдущей версии LabVIEW подпалитры функций **Тригонометрические** (Trigonometric) и **Экспоненциальные** (Exponential) по умолчанию устанавливаются в категории **Математика** (подпалитра

Элементарные и специальные функции и ВП (Elementary & Special Functions and Vis)), однако они могут быть установлены в палитре числовых функций с помощью диалогового окна **Вставить подпалитру** (Insert Subpalette), в котором необходимо установить отметку в строке **Связать с существующим файлом палитры** (Link to existing palette file) и выбрать в следующем диалоговом окне Select a .mnu file в папке menus\Categories необходимый файл с расширением .mnu. Окно **Вставить подпалитру** вызывается с помощью контекстного меню **Вставить ⇒ Подпалитра**... (Insert ⇒ Subpalette...) подпалитры числовых функций, переведенной в режиме редактирования. Переход к такому режиму осуществляется путем выбора меню **Инструменты ⇒ Расширенные ⇒ Редактировать состав палитры** (Tools ⇒ Advanced ⇒ Edit Palette Set). Изменения вида подпалитры сохраняются с помощью кнопки **Сохранить изменения** (Save Changes) в открывающемся при этом окне **Редактировать вид палитры элементов управления и функций** (Edit Controls and Functions Palette Set).

Далее будем полагать, что такая операция выполнена и тригонометрические и экспоненциальные функции установлены в палитре **Числовые**, что позволяет рассмотреть их в данном разделе.

Таблица основной палитры числовых функций:

Add

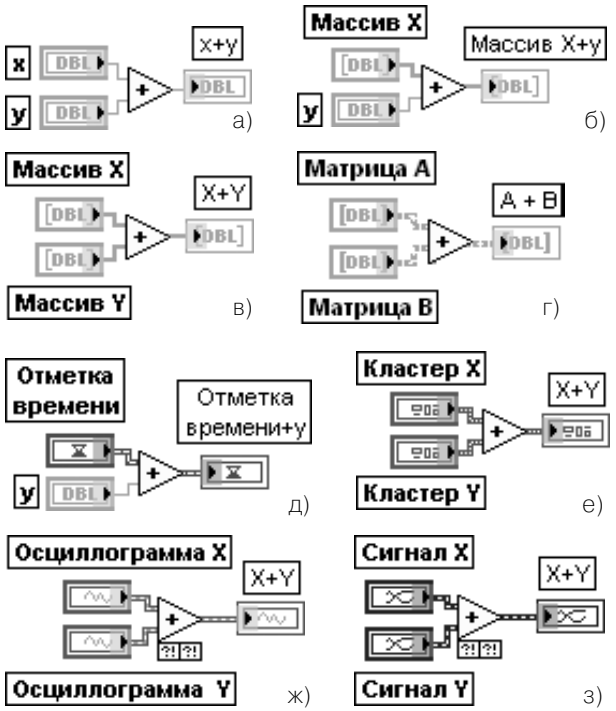
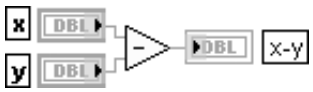


Рис. 2.3. Варианты подключения функции ***Сложить***

Сложить

Функция рассчитывает сумму входов. Если ко входам функции подключаются две осциллограммы или два набора значений с динамическим типом данных, то рядом с функцией появляются терминалы **вход ошибки** (error in) и **выход ошибки** (error out) (рис. 2.3ж, 2.3з). Не допускается суммирование двух значений меток времени. Функция является **полиморфной**, поэтому входы могут быть числовыми скалярами, массивами, матрицами или кластерами чисел, массивами кластеров чисел, отметками времени и т. д. На рисунках показаны варианты суммирования двух скаляров (а), скаляра и массива (б), двух массивов (в), двух матриц (г), отметки времени и скаляра (д), двух кластеров (е), двух осциллограмм (ж) и двух наборов данных динамического типа (з)

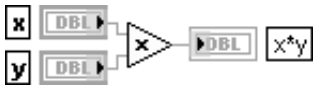
Subtract



Вычесть

Функция рассчитывает разность входов. Вычитание значений двух меток времени дает числовое значение (интервал времени), а вычитание числового значения из значения метки времени дает значение метки времени. Недопустимо вычитание метки времени из числового значения

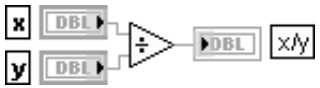
Multiply



Умножить

Функция возвращает произведение входов

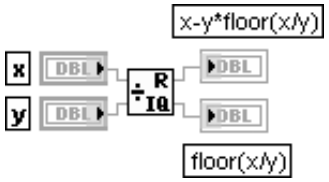
Divide



Разделить

Функция возвращает частное от деления значений на входах

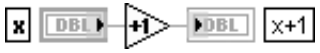
Quotient & Remainder



Частное и остаток

Функция рассчитывает целое **частное** (floor(x/y)) и **остаток** (x-y*floor(x/y)) от деления **x** на **y**

Increment



Инкремент

Функция возвращает значение входа, увеличенное на 1

Decrement

Декремент

Функция возвращает значение входа, уменьшенное на 1

Add Array Elements

Сложить элементы массива

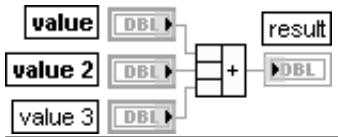
Функция возвращает сумму всех элементов входного **числового массива** (numeric array)

Multiply Array Elements

Перемножить элементы массива

Функция возвращает **произведение** (product) элементов входного **числового массива** (numeric array)

Compound Arithmetic

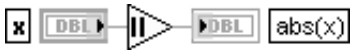


Составная арифметика

Функция позволяет выполнять арифметические операции сложения, вычитания, умножения и деления с произвольным количеством числовых величин. Вид операции выбирается с помощью строки **Изменить режим** (Change Mode) контекстного меню функции.

Знак выхода и каждого входа может быть изменен путем выбора опции **Инвертировать** (Invert) контекстного меню

Absolute Value

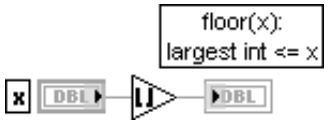


Абсолютное значение
Функция возвращает абсолютное значение входа

Round To Nearest

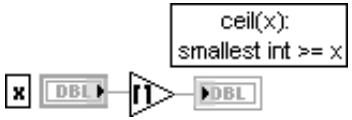
Округление до ближайшего целого
Функция округляет входное значение до ближайшего целого. Если входное значение находится посередине между двумя целыми (например, 1,5 или 2,5), то функция возвращает ближайшее четное значение (2)

Round To -Infinity



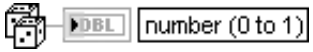
Округление до меньшего целого
Функция усекает входное значение до меньшего целого значения. Например, если входное значение равно 3,8, то результат будет равен 3. Если на входе –3,8, то результат будет равен –4

Round To +Infinity



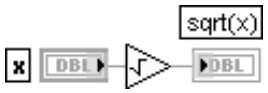
Округление до большего целого
Функция округляет входное значение до большего целого. Например, если входное значение равно 3,1, то результат будет равен 4. Если на входе –3,1, то результатом будет –3

Random Number (0-1)



Случайное число в диапазоне (0–1)
Функция генерирует случайные числа с равномерным амплитудным распределением в диапазоне от 0 до 1

Square Root



Квадратный корень
Функция рассчитывает квадратный корень входного значения. Если входное значение отрицательное, то возвращается значение NaN

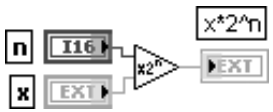
Square

Квадрат
Функция рассчитывает квадрат входного значения

Negate

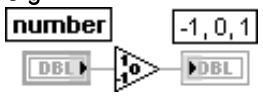
Отрицание
Функция изменяет знак входной величины на противоположный

Scale By Power Of 2



Масштабирование по степени числа 2
Функция умножает **x** на число 2, возведенное в степень **n**. Если **n** является числом с плавающей запятой, то функция округляет **n** перед масштабированием **x** (0,5 округляется до 0; 0,51 округляется до 1). Если **x** целое, то эта функция эквивалентна арифметическому сдвигу

Sign

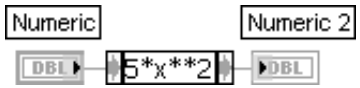


Знак
Функция возвращает значение –1, если входное число отрицательное, возвращает 0, если оно равно 0, и возвращает 1, если число положительное

Reciprocal

Обратная величина
Функция делит 1 на входное значение

Expression Node



Узел Выражение
Узел **Выражение** (Expression Node) используется для расчета выражений или уравнений, которые содержат единственную переменную. В формулах могут использоваться следующие встроенные функции: abs, acos, acosh, asin, asinh, atan, atanh, ceil, cos, cosh, cot, csc, exp, expm1, floor, getexp, getman, int, intrz, ln, lnp1, log, log2, max, min, mod, rand, rem, sec, sign, sin, sinc, sinh, sqrt, tan, tanh. Узел Выражение воспринимает только точку в качестве десятичного разделителя

Числовая константа (Numeric Constant)

Константа перечисления (Enum Constant)

Кольцевая константа (Ring Constant)

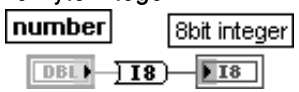
Отрицательная бесконечность (Negative Infinity)

Положительная бесконечность (Positive Infinity)

Машинная эпсилон (Machine Epsilon)
(возвращает значение 2,2204460492503131e–16)

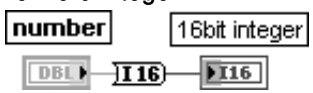
Таблица функций подпалитры **Преобразование** (Conversion):

To Byte Integer



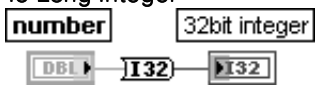
В байтовое целое число
Функция преобразует входное число в 8-битовое целое в диапазоне от –128 до 127

To Word Integer



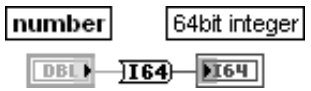
В целое слово
Функция преобразует входное число в 16-битовое целое в диапазоне от –32768 до 32767

To Long Integer



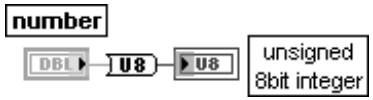
В длинное целое число
Функция преобразует входное число в 32-битовое целое в диапазоне от –(2^ 31) до (2^ 31)–1

To Quad Integer



В 64-битовое целое
Функция преобразует входное число в 64-битовое целое в диапазоне от – (2^ 63) до (2^ 63)–1

To Unsigned Byte Integer



В байтовое целое без знака
Функция преобразует входное число в 8-битовое целое число без знака в диапазоне от 0 до 255

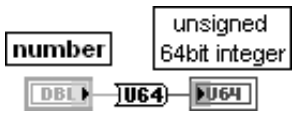
To Unsigned Word Integer

В целое слово без знака
Функция преобразует входное число в 16-битовое целое число без знака в диапазоне от 0 до 65535

To Unsigned Long Integer

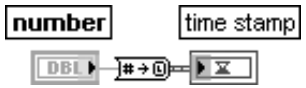
В длинное целое число без знака
Функция преобразует входное число в 32-битовое целое число без знака в диапазоне от 0 до (2^32)–1

To Unsigned Quad Integer



В 64-битовое целое без знака
Функция преобразует входное число в 64-битовое целое число без знака в диапазоне от 0 до (2^64)–1

To Time Stamp



В метку времени
Функция преобразует входное число в метку времени

To Single Precision Float

В число с плавающей запятой с одинарной точностью
Функция преобразует входное число в число с плавающей запятой с одинарной точностью

To Double Precision Float

В число с плавающей запятой с двойной точностью
Функция преобразует входное число в число с плавающей запятой с двойной точностью

To Extended Precision Float

В число с плавающей запятой с расширенной точностью
Функция преобразует входное число в число с плавающей запятой с расширенной точностью

To Single Precision Complex

В комплексное число с одинарной точностью
Функция преобразует входное число в комплексное число с плавающей запятой с одинарной точностью

To Double Precision Complex

В комплексное число с двойной точностью
Функция преобразует входное число в комплексное число с плавающей запятой с двойной точностью

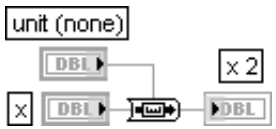
To Extended Precision Complex

В комплексное число с расширенной точностью
Функция преобразует входное число в комплексное число с плавающей запятой с расширенной точностью

Convert Unit

Преобразовать размерность
Функция преобразует физическое число (размерное) в безразмерное число, и наоборот

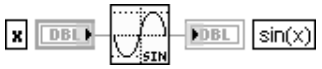
Cast Unit Bases



Изменить базовые единицы
Функция изменяет базовые единицы, связанные с входом **x**, на базовые единицы, связанные с входом **единица измерения** (unit), и возвращает результаты на выходной терминал

Таблица функций подпалитры **Тригонометрические** (Trigonometric):

Sine

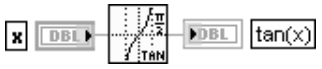


Синус
Функция рассчитывает синус входного значения **x** (рад)

Cosine

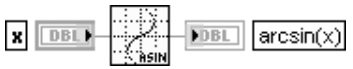
Косинус
Функция рассчитывает косинус входного значения **x** (рад)

Tangent



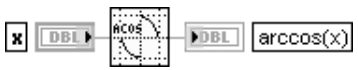
Тангенс
Функция рассчитывает тангенс входного значения **x** (рад)

Inverse Sine



Арксинус
Функция рассчитывает значение арксинуса **x**. Результат выражен в радианах

Inverse Cosine

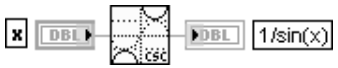


Арккосинус
Функция рассчитывает значение арккосинуса **x**. Результат выражен в радианах

Inverse Tangent

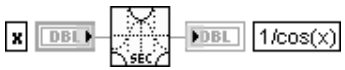
Арктангенс
Функция рассчитывает значение арктангенса **x**. Результат выражен в радианах

Cosecant



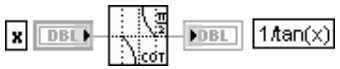
Косеканс
Функция рассчитывает косеканс входного значения **x** (рад)

Secant



Секанс
Функция рассчитывает секанс входного значения **x** (рад)

Cotangent

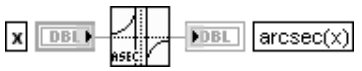


Котангенс
Функция рассчитывает котангенс входного значения **x** (рад)

Inverse Cosecant

Арккосеканс
Функция рассчитывает значение арккосеканса **x**. Результат выражен в радианах

Inverse Secant

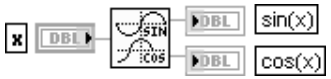


Арксеканс
Функция рассчитывает значение арксеканса **x**. Результат выражен в радианах

Inverse Cotangent

Арккотангенс
Функция рассчитывает значение арккотангенса **x**. Результат выражен в радианах

Sine & Cosine



Синус и косинус
Функция рассчитывает синус и косинус входного значения **x** (рад)

Inverse Tangent (2 Input)

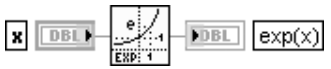
Арктангенс (2 входа)
Функция рассчитывает арктангенс отношения **y**/**x**, выраженного в радианах. Эта функция может рассчитывать арктангенс для углов в любом квадранте плоскости **x**-**y**, в то время как функция **Арктангенс** рассчитывает арктангенс только в двух квадрантах

Sinc

Функция Sin(x)/x
Функция рассчитывает значение **sin(x)/x**, где значение **x** выражено в радианах

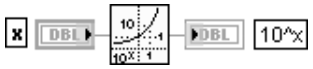
Таблица функций подпалитры **Экспоненциальные** (Exponential):

Exponential



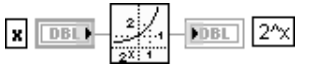
Экспонента
Функция рассчитывает значение числа **e**, возведенного в степень **x**

Power Of 10



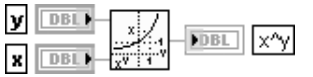
Степень числа 10
Функция рассчитывает значение числа 10, возведенного в степень **x**

Power Of 2



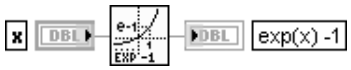
Степень числа 2
Функция рассчитывает значение числа 2, возведенного в степень **x**

Power Of X



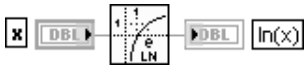
Степень числа X
Функция рассчитывает значение числа **x**, возведенного в степень **y**

Exponential (Arg) -1



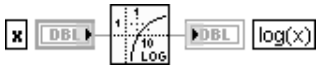
Функция Exp(x) – 1
Функция рассчитывает уменьшенное на 1 значение числа **e**, возведенного в степень **x**. При очень малых **x** данная функция является более точной по сравнению с функцией **Exponential**, у которой единица вычитается на выходе

Natural Logarithm



Натуральный логарифм
Функция рассчитывает натуральный логарифм числа **x**

Logarithm Base 10

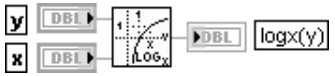


Логарифм по основанию 10
Функция рассчитывает десятичный логарифм числа **x**

Logarithm Base 2

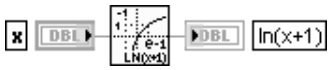
Логарифм по основанию 2
Функция рассчитывает логарифм числа **x** по основанию 2

Logarithm Base X



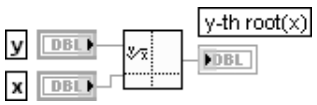
Логарифм по основанию X
Функция рассчитывает логарифм числа **y** по основанию **x**

Natural Logarithm (Arg +1)



Функция Ln(x+1)
Функция рассчитывает натуральный логарифм увеличенного на 1 значение числа **x**. При значениях **x**, близких к 0, данная функция является более точной по сравнению с добавлением 1 к **x** в функции **Натуральный логарифм**

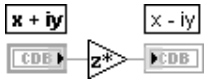
y-th root of x



Корень из x степени y
Возвращает корень степени **y** из входного значения **x**. Если число **x** не является комплексным, оно должно быть больше или равно 0, иначе результат будет равен NaN

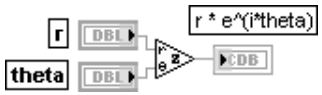
Таблица функций подпалитры **Комплексные функции** (Complex):

Complex Conjugate



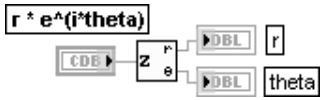
Комплексно-сопряженное значение
Функция формирует комплексно-сопряженное значение **x-iy** для входного значения **x+iy**. **x + iy** может быть комплексным числом, массивом или кластером комплексных чисел, массивом кластеров комплексных чисел и т. д.

Polar To Complex



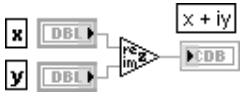
Перевести значения из полярных координат в комплексное значение
Функция формирует комплексное значение из двух значений, заданных в полярных координатах

Complex To Polar



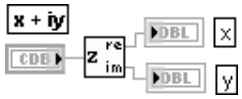
Перевести комплексное значение в значения полярных координат
Функция разделяет комплексное значение на два значения, заданные в полярных координатах

Re/Im To Complex



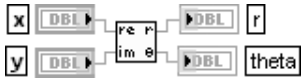
Перевести значения из декартовых координат в комплексное значение
Функция формирует комплексное значение из двух значений, заданных в декартовых координатах

Complex To Re/Im



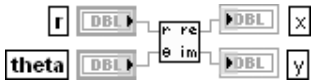
Перевести комплексное значение в значения декартовых координат
Функция разделяет комплексное значение на два значения, заданные в декартовых координатах

Re/Im To Polar



Перевести значения из декартовых координат в полярные
Функция переводит два значения из декартовых координат в полярные координаты

Polar To Re/Im



Перевести значения из полярных координат в декартовые
Функция переводит два значения из полярных координат в декартовые

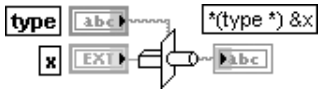
Функции **манипуляции данными** служат для преобразования типов данных, выполнения логических и циклических сдвигов битов чисел, разделения слов на байты и объединения байтов в слова, перестановки байтов и слов. Потребность в таких преобразованиях данных может возникать, например, при работе с сетевым протоколом TCP/IP или при обмене данными с внешними устройствами.

При анализе функций манипуляции данными необходимо иметь в виду, что все данные в LabVIEW имеют два компонента: сами данные и дескриптор (описатель) типа данных. Дескриптор типа данных представляет массив целых (I16)

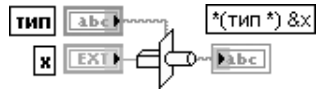
чисел, которые образуют код, описывающий представление данных. Дескриптор содержит информацию о длине (в байтах) данных, а также дополнительную информацию об их типе и структуре. При выполнении такой функции преобразования данных, как **Приведение типа** (Type Cast), изменяется дескриптор типа данных, а сами данные остаются неизменными. В то же время при преобразовании числа в десятичную строку с помощью функции **Преобразовать в строку** (Format Into String) изменяется дескриптор типа данных и сами данные преобразуются по определенному алгоритму.

Ниже в таблицах приведены описания функций манипуляции данными.

Type Cast



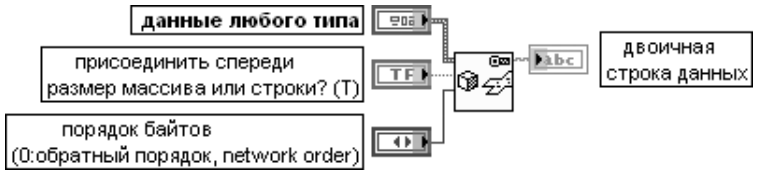
Приведение типа



Функция преобразует данные на входе **x** к типу данных, заданному на входе **тип** (type), преобразуя их сначала к типу **приведенный** (flatten), а затем, используя новый тип данных, выполняет обратное преобразование типа с целью получения требуемого. Соединения отображают тип данных этой полиморфной функции.
Вход **тип** (type) представляет тип данных, к которому производится преобразование. Он является только «шаблоном», служащим для определения типа, любые реальные данные на этом входе игнорируются.
Вход **x** используется для передачи данных, преобразуемых к заданному типу. Вход **x** может быть любого типа.
Выход ***(тип*)&x** (*(type *) &x) отображает значение, поданное на вход **x** и преобразованное к заданному **типу**.
Для правильного использования этой функции необходимо быть уверенным в совместимости типов исходных и преобразованных данных. В противном случае эта функция может генерировать непредсказуемые или бесполезные данные

Flatten To String

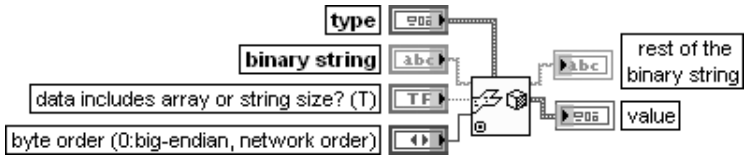
Перевести в строку



Функция преобразует **данные любого типа** (anything) в **приведенные данные** (flattened) и возвращает **двоичную строку данных** (data string), содержащую двоичное представление данных, и **строку типа данных** (type string), содержащую дескриптор типа, который описывает тип **данных любого типа** (anything).

Выход **строка типа данных** (type string) на самом деле является массивом, содержащим закодированное двоичное описание **двоичной строки данных**. **Строка типа данных** отличается от входа **тип данных** в функции **Восстановить из строки** (Unflatten From String).
Выход **двоичная строка данных** (data string) отображает сформированные функцией данные приведенного типа. **Двоичная строка данных** перед каждым нескалярным компонентом размещает заголовочную информацию, описывающую его размер. Такая строка может быть сохранена в файле и передана по сети. При передаче строки по сети получатель должен быть способен интерпретировать ее. Обычно LabVIEW хранит данные в виде независимых фрагментов с косвенной адресацией. Эта функция копирует данные в LabVIEW в непрерывный буфер **двоичной строки данных**. Для преобразования строки данных обратно в данные любого типа необходимо использовать функцию **Восстановить из строки** (Unflatten From String).
Элементы **двоичной строки данных** представлены в машинно независимой форме с более значимым первым байтом (big-endian). Именно так получатель должен интерпретировать строку.
Строка, сформированная функцией **Перевести в строку** (Flatten To String), является строкой LabVIEW. Строки LabVIEW имеют 4-байтовое (I32) число в начале строки, содержащее информацию о длине строки. Это позволяет строке LabVIEW включать символ NULL [ASCII символ нуль (0)]. Если строка LabVIEW передается внешнему коду и используется им как строка языка **C**, символ NULL, включенный в строку, может вызвать проблемы, поскольку в строке **C** первый же символ NULL интерпретируется как символ конца строки

Unflatten From String

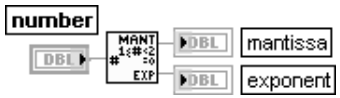


Восстановить из строки

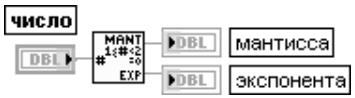
Функция преобразует **двоичную строку** (binary string) с типом, указанным на входе **тип данных** (type), в соответствующее значение на выходе **значение** (value). Вход **двоичная строка** должен содержать данные **приведенного типа** (flatten), которые могут быть преобразованы к типу, указанному на входе **тип**.
Вход **двоичная строка** представляет строку, которая обычно формируется функцией **Перевести в строку** (Flatten To String). Двоичная строка перед каждым нескалярным компонентом содержит заголовок с информацией о размере этого компонента. Элементы двоичной строки имеют машинно независимую форму.

Вход **тип** является типом данных LabVIEW, а не дескриптором типа с выхода функции **Перевести в строку** (Flatten To String).
Выход **ошибка** (err) принимает значение ИСТИНА, если преобразование не было успешным.
Выход **значение** (value) имеет тот же тип данных и структуру, что и **тип**

Mantissa & Exponent

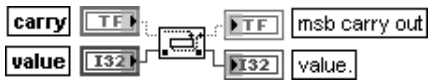


Мантисса и экспонента

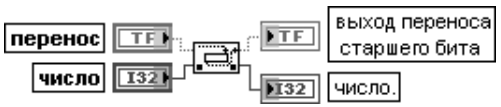


Функция возвращает мантиссу и экспоненту входного **числа** (number) так, что **число** = **мантисса** * $2^{\text{экспонента}}$. Если число равно 0, то мантисса и экспонента равны 0. В ином случае абсолютное значение мантиссы больше или равно 1 и меньше 2, а значение экспоненты является целым.
Вход **число** (number) может иметь любое десятичное представление.
Выходы **мантисса** и **экспонента** имеют то же десятичное представление, что и **число**

Rotate Left With Carry

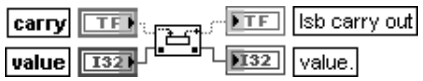


Циклический сдвиг влево с переносом

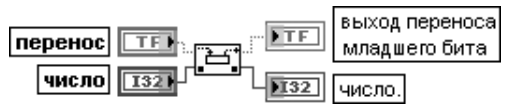


Функция сдвигает каждый бит входного **числа** (value) влево (от менее значимого к более значимому биту), вставляет **перенос** (carry) в младший бит и возвращает старший бит.
Вход **перенос** передает новый младший бит числа.
Вход **число** должен быть целым. Он не может быть массивом или кластером.
Выход переноса старшего бита (msb carry out) отображает прежнее значение старшего бита числа на входе **число**.
Выход **число** представляет новое значение. Тип данных выхода **число** определяется типом данных на одноименном входе

Rotate Right With Carry

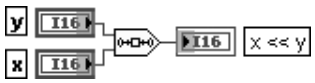


Циклический сдвиг вправо с переносом

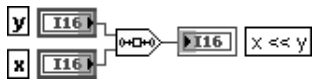


Функция сдвигает каждый бит входного **числа** (value) вправо (от более значимого к менее значимому биту), вставляет **перенос** (carry) в старший бит и возвращает младший бит.
Вход **перенос** передает новый старший бит числа.
Вход **число** должен быть целым. Он не может быть массивом или кластером.
Выход переноса младшего бита (lsb carry out) отображает прежнее значение младшего бита **числа**.
Выход **число** представляет новое значение. Тип данных выхода **число** определяется типом данных на одноименном входе

Logical Shift



Логический сдвиг



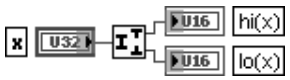
Функция сдвигает число на входе **x** на число битов, определяемых значением на входе **y**. Соединение отражает тип данных по умолчанию для этой полиморфной функции.
Вход **y** может иметь любое числовое представление. Если **y** больше 0, то функция сдвигает **x** влево на **y** бит (от менее значимого к более значимому биту) и вставляет нули в младшие биты. Если **y** меньше 0, функция сдвигает **x** вправо на **y** бит (от более значимого к менее значимому биту) и вставляет нули в старшие биты.
Вход **x** может быть любого целого типа. Если **x** является 8-, 16-, или 32-битовым целым и **y** больше 8, 16, или 32 или меньше чем –8, –16, или –32, соответственно, то выходное значение будет нулевым.
Выход **x << y** отображает результат сдвига и имеет такое же числовое представление, что и **x**

Rotate

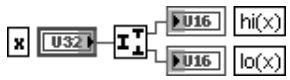
Циклический сдвиг

Функция сдвигает число на входе **x** на число битов, определяемых значением на входе **y**. Если **y** больше 0, функция сдвигает **x** влево на **y** бит (от менее значимого к более значимому биту) и вставляет старшие биты на место младших битов. Если **y** меньше 0, функция сдвигает **x** вправо на **y** битов (от более значимого к менее значимому биту) и вставляет младшие биты на место старших битов.
Если **x** является 8-, 16-, или 32-битовым целым, тогда при любом значении **y** сдвиг на **y** ± 8, **y** ± 16 или **y** ± 32 соответственно даст такое же выходное значение, что и сдвиг на **y**. Например, если **x** является 8-битовым целым, **y** = 1 и **y** = 9 даст тот же результат.
Выход **x циклически сдвинутое влево на y** (x rotated left by y) представляет результат циклического сдвига. Тип данных на выходе **x циклически сдвинутое влево на y** определяется типом данных на входе **x**

Split Number

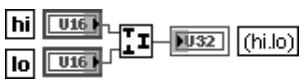


Разделить число

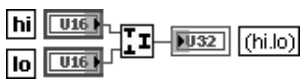


Функция разделяет число на составляющие его байты или слова.
Вход **x** может быть 16- или 32-битовым целым числом, массивом или кластером чисел с таким представлением.
Выходы **hi**(**x**) и **lo**(**x**) отображают целые числа, занимающие половину ширины **x**. **hi**(**x**) и **lo**(**x**) могут быть 8-битовыми и 16-битовыми целыми числами без знака, массивом или кластером чисел с таким представлением

Join Numbers

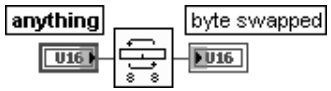


Объединить числа



Функция формирует число из двух байтов или слов.
Входы **hi** и **lo** могут быть 8-битовыми или 16-битовыми числами, массивами или кластерами чисел с таким представлением.
Выход (**hi.lo**) отображает целое число с удвоенной по сравнению с **hi** и **lo** шириной. (**hi.lo**) является 16-битовым или 32-битовым целыми числами без знака, массивом или кластером чисел с таким представлением. Если **hi** и **lo** имеют различную ширину, то (**hi.lo**) будет иметь удвоенную по сравнению с более широким числом ширину

Swap Bytes



Переставить байты

Функция переставляет старший и младший байты в каждом слове на входе **данные любого типа** (anything). Эта функция не влияет на строки, числа с плавающей точкой и байтовые целые числа.
Вход **данные любого типа** может быть целым числом, массивом целых или кластером целых, содержащим числа, в которых необходимо произвести перестановку байтов. В случае кластера, который содержит целые, эта функция производит перестановку байтов только целых чисел. Если требуется выполнить перестановку байтов числа с плавающей точкой, необходимо использовать функцию **Приведение типа** (Type Cast) для преобразования значения в одномерный массив байтов. Затем использовать функцию **Обратить одномерный массив** (Reverse 1D Array) и преобразовать обратно в число с плавающей запятой.
Выход **данные с переставленными байтами** (byte swapped) имеет тот же тип данных и структуру, что и входные **данные любого типа** (anything)

Swap Words

Переставить слова

Функция переставляет старшее и младшее 16-битовое слово в каждом 32-битовом целом числе на входе **данные любого типа** (anything). Эта функция не влияет на строки, числа с плавающей запятой, байтовые и 16-битовые целые числа.
Вход **данные любого типа** может быть целым числом, массивом целых или кластером целых чисел, содержащим числа, в которых необходимо произвести перестановку слов. В случае кластера, который содержит целые числа, эта функция производит перестановку слов только целых чисел.
Выход **данные с переставленными словами** (word swapped) имеет тот же тип данных и структуру, что и **данные любого типа**

Перечень констант подпалитры **Математические и научные константы** (Math & Scientific Constants):

Число π (3,1415926535897932)	**Число 2π** (6,28318530717958650)
Число π/2 (1,5707963267948966)	**Число 1/π** (0,318309886183790670)
Натуральный логарифм числа π (1,1447298858494002)	**Основание натурального логарифма** (число **e**) (2,7182818284590452)
Значение 1/e (0,36787944117144232)	**Десятичный логарифм числа e** (0,43429448190325183)
Натуральный логарифм числа 10 (2,3025850929940597)	**Натуральный логарифм числа 2** (0,69314718055994531)
Постоянная Планка (6,62606876e–34) Дж с	**Гравитационная постоянная** (6,673e–11) Н м²/кг²
Скорость света (2,99792458e8) м/с	**Число Авогадро** (6,02214199e23) 1/моль
Элементарный заряд электрона (1,602176462e–19) Кл	**Постоянная Ридберга** (1,0973731568549e7) 1/м
Константа цвета (Color Box Constant)	**Молярная газовая постоянная** (8,314472) Дж/(моль K)

К числовым функциям относятся Экспресс-ВП **Формула** (Formula), **Масштабирование и отображение** (Scaling and Mapping) и **Математическая обработка во временной области** (Time Domain Math).

Формула (Formula)

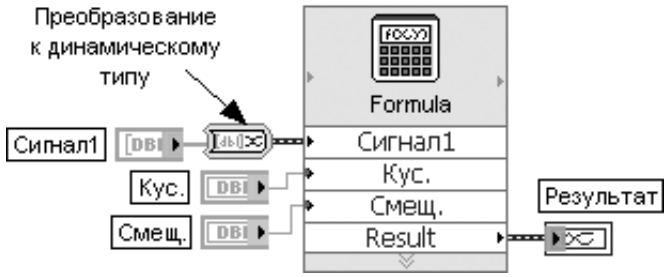


Рис. 2.4. Блок-диаграмма варианта подключения Экспресс-ВП

Экспресс-ВП **Формула** (Formula) (рис. 2.5) позволяет производить математическую обработку входных данных, обеспечиваемую базовыми научными калькуляторами. Входы и выходы Экспресс-ВП рассчитаны на подключение данных динамического типа. При подключении данных другого типа производится преобразование типов, в том числе и с помощью Экспресс-ВП **Преобразование в данные динамического типа** (Convert to Dynamic Data) (рис. 2.4).

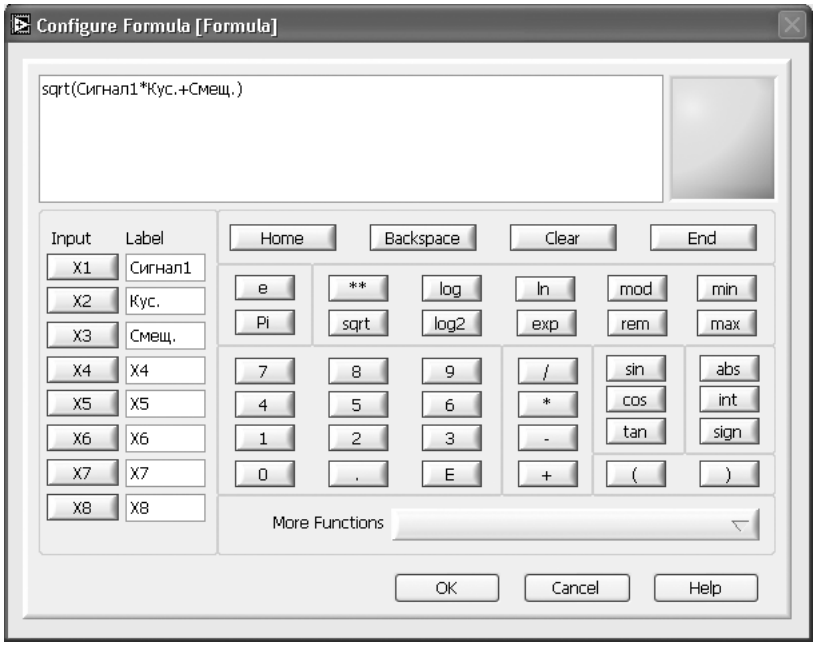


*Рис. 2.5. Вид диалогового окна конфигурирования Экспресс-ВП **Формула** (Formula)*

В Экспресс-ВП **Формула** используется функциональность следующих ВП: **Узел Выражение** (Expression Node), **Узел Формула** (Formula Node), **Оценка узла формулы** (Eval Formula Node), **Оценка строки формулы** (Eval Formula String), **Оценка массива значений параметрической функции** (Eval Multi-Variable Array)

Масштабирование и отображение (Scaling and Mapping)

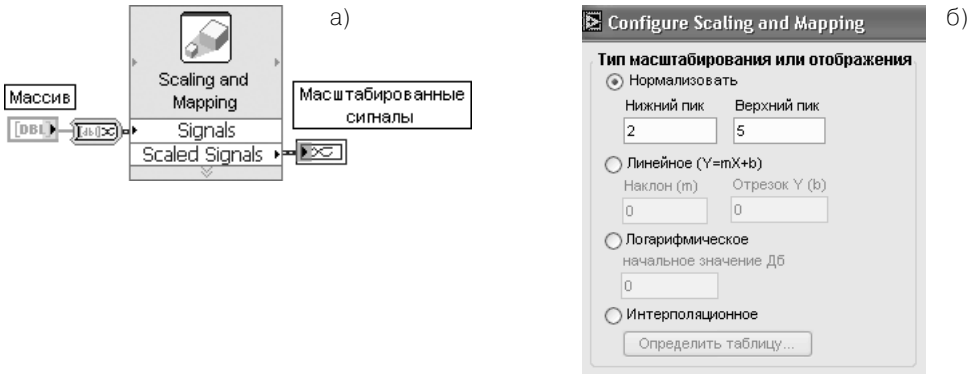


Рис. 2.6. Блок-диаграмма (а) и вид диалогового окна конфигурирования (б) Экспресс-ВП

Экспресс-ВП **Масштабирование и отображение** (Scaling and Mapping) (рис. 2.6) изменяет диапазон и характер отображения входных данных с помощью их масштабирования и выбора вида преобразования.
Содержит следующие опции:
Нормализовать (Normalize) определяет масштаб и смещение, необходимые для преобразования данных, так чтобы их максимум был равен значению **Верхний пик** (Highest peak), а минимум – **Нижний пик** (Lowest peak).
Линейное **(Linear (Y=mX+b)) устанавливает линейное преобразование значений.**
Логарифмическое (Logarithmic) устанавливает логарифмическое преобразование входных значений. При этом параметр **начальное значение, дБ** (db reference) определяет начальную точку шкалы в децибелах.
Интерполяционное (Interpolated) определяет шкалу, основанную на таблице значений, которые интерполируются линейно с целью получения масштабного параметра.
Таблица значений задается в диалоговом окне **Определить сигнал** (Define Signal), которое вызывается с помощью кнопки **Определить таблицу** (Define Table).
В Экспресс-ВП используется функциональность ВП **Масштаб и смещение осциллограммы** (Waveform Scale and Offset)

Математическая обработка во временной области (Time Domain Math)

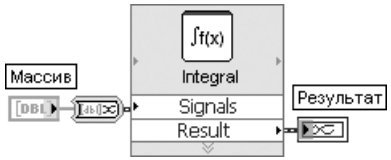


Рис. 2.7. Блок-диаграмма подключения Экспресс-ВП

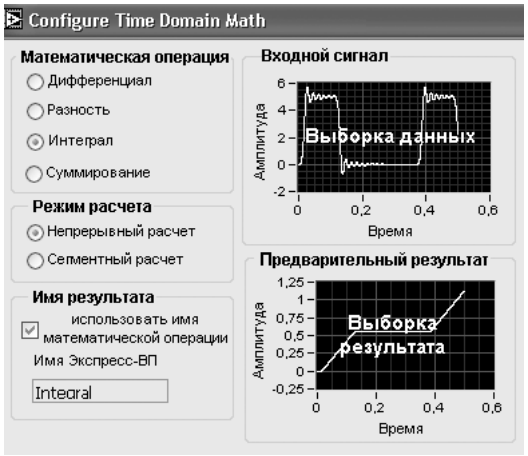


Рис. 2.8. Вид диалогового окна конфигурирования Экспресс-ВП ***Математическая обработка во временной области*** *(Time Domain Math)*

Экспресс-ВП **Математическая обработка во временной области** (Time Domain Math) (рис. 2.7) выполняет одну из операций математической обработки сигналов во временной области.
Набор математических операций включает следующие (рис. 2.8): **Дифференциал** (Differential), **Разность** (Difference), **Интеграл** (Integral) и **Суммирование** (Summation).
В Экспресс-ВП используется функциональность ВП **Производная x(t)** (Derivative x(t)) и **Интеграл x(t)** (Integral x(t))

2.1.2. Логические функции

Логические функции (рис. 2.9) используются для выполнения логических операций над значениями как скаляров, так и массивов логических величин. Функции **И** (And), **ИЛИ** (Or), **Исключающее ИЛИ** (Exclusive Or), **НЕ** (Not), **И-НЕ** (Not And), **ИЛИ-НЕ** (Not Or), **Исключающее ИЛИ-НЕ** (Not Exclusive Or), **Исключение** (Implies) являются полиморфными. Оба входа этих функций должны иметь логические или числовые значения, причем эти значения могут быть скалярами, массивами или кластерами. При обработке числовых значений перечисленные функции выполняют побитовую обработку чисел.

Ниже в таблице приведены пояснения к набору логических функций.

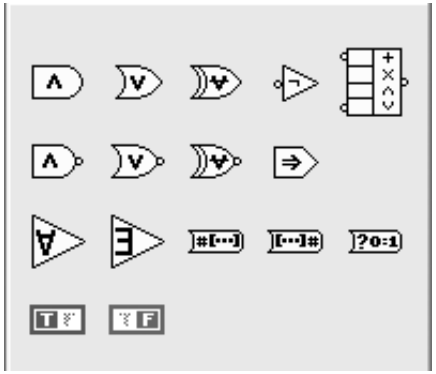


Рис. 2.9. Вид палитры логических функций

And

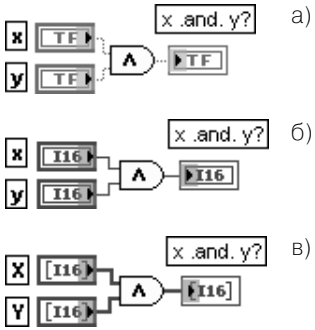
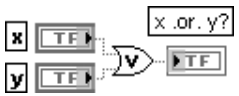


Рис. 2.10. Варианты подключения функции ***И***

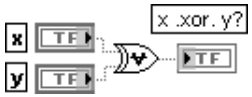
Логическая функция И
Возвращает значение ИСТИНА (1) только при подаче на оба входа значений ИСТИНА (1), иначе возвращает значение ЛОЖЬ (0). На рис. 2.10а приведены примеры использования данной функции для обработки скаляров логического типа, на рис. 2.10б – скаляров числового типа, на рис. 2.10в – массивов числовых значений

Or



Логическая функция ИЛИ
Возвращает значение ЛОЖЬ (0) только при подаче на оба входа значений ЛОЖЬ (0), иначе возвращает значение ИСТИНА (1)

Exclusive Or

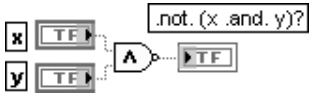


Логическая функция Исключающее ИЛИ
Возвращает значение ЛОЖЬ (0) только при подаче на оба входа значений ЛОЖЬ (0) или значений ИСТИНА (1), иначе возвращает значение ИСТИНА (1)

Not

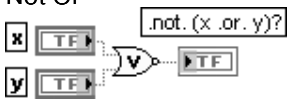
Логическая функция НЕ
Возвращает значение ЛОЖЬ (0) при подаче на вход значения ИСТИНА (1), и наоборот

Not And



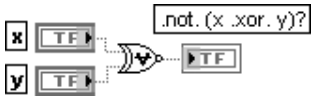
Логическая функция И-НЕ
Возвращает значение ЛОЖЬ (0) только при подаче на оба входа значений ИСТИНА (1), иначе возвращает значение ИСТИНА (1)

Not Or



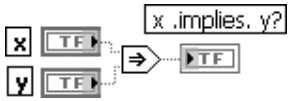
Логическая функция ИЛИ-НЕ
Возвращает значение ИСТИНА (1) только при подаче на оба входа значений ЛОЖЬ (0), иначе возвращает значение ЛОЖЬ (0)

Not Exclusive Or



Логическая функция Исключающее ИЛИ-НЕ
Возвращает значение ИСТИНА (1) только при подаче на оба входа значений ЛОЖЬ (0) или значений ИСТИНА (1), иначе возвращает значение ЛОЖЬ (0)

Implies



Исключение
Инвертирует **x** и затем выполняет операцию **Логическое ИЛИ** с входом **y**. Если на входе **x** значение ИСТИНА и на входе **y** значение ЛОЖЬ, то функция возвращает ЛОЖЬ. Иначе возвращает ИСТИНА

And Array Elements

Логическая функция И для элементов массива
Возвращает значение ИСТИНА, если все элементы **Логического массива** (Boolean array) имеют значение ИСТИНА. Иначе возвращает ЛОЖЬ. Функция воспринимает массивы любого размера

Or Array Elements

Логическая функция ИЛИ для элементов массива
Возвращает ЛОЖЬ, если все элементы **Логического массива** (Boolean array) имеют значение ЛОЖЬ. Иначе возвращает ИСТИНА. Функция воспринимает массивы любого размера

Number To Boolean Array

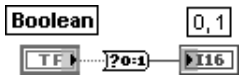
Число в логический массив
Преобразует целое число в логический массив из 8, 16 или 32 элементов в зависимости от числа битов целого числа. Нулевой элемент логического массива соответствует младшему разряду двоичного представления числа

Boolean Array To Number

Логический массив в число
Преобразует **Логический массив** (Boolean array) в 32-битовое целое число без знака, интерпретируя массив как двоичное представление целого числа, причем нулевой элемент массива соответствует младшему биту числа.

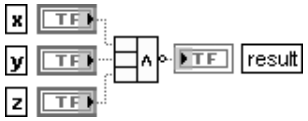
Логический массив является одномерным массивом логических значений. Функция исключает часть логического массива, если он превышает заданную длину, и дополняет значениями ЛОЖЬ, если массив короткий

Boolean To (0,1)



Логическое значение в число
Преобразует логические значения ЛОЖЬ или ИСТИНА в 16-битовое целое число, имеющее значение соответственно 0 или 1

Compound Arithmetic



Составная арифметика
Функция позволяет выполнять логические операции **И**, **ИЛИ** и **Исключающее ИЛИ** с произвольным числом числовых или логических величин. Вид операции выбирается с помощью строки **Изменить режим** (Change Mode) контекстного меню функции. При этом на выходе и на любом входе функции с помощью строки **Инвертировать** (Invert) того же меню может быть установлена операция инверсии.
Операция **Исключающее ИЛИ** при числе входов более двух выполняется последовательно сначала с парой входов, затем с результатом и следующим входом и т. д.

2.1.3. Строковые функции

Строки представляют собой последовательности отображаемых и неотображаемых символов в стандарте ASCII. Часто строки используются в качестве простых текстовых сообщений. В процессе управления приборами цифровые данные передаются в виде символьных строк, которые преобразуются затем в цифры. Процедура запоминания цифровых данных в файле также может потребовать их строковой организации.

На рис. 2.11а приведен вид основной палитры строковых функций с одним Экспресс-ВП и вид ряда дополнительных подпалитр: **Дополнительные строковые функции** (Additional String Functions) (рис. 2.11б), **Функции взаимного преобразования строк и чисел** (String/Number Conversion) (рис. 2.11в), **Функции взаимного преобразования строк, массивов байтов и путей** (String/Array/Path Conversion) (рис. 2.11г), а также **Функции работы с файлами XML** (XML) (рис. 2.11д).

Ниже в таблице рассмотрены строковые функции из основной палитры.

String Length

Длина строки

Функция возвращает число символов (байтов) в строке.
На вход **строка** (string) может быть подана строка, кластер строк или массив кластеров строк

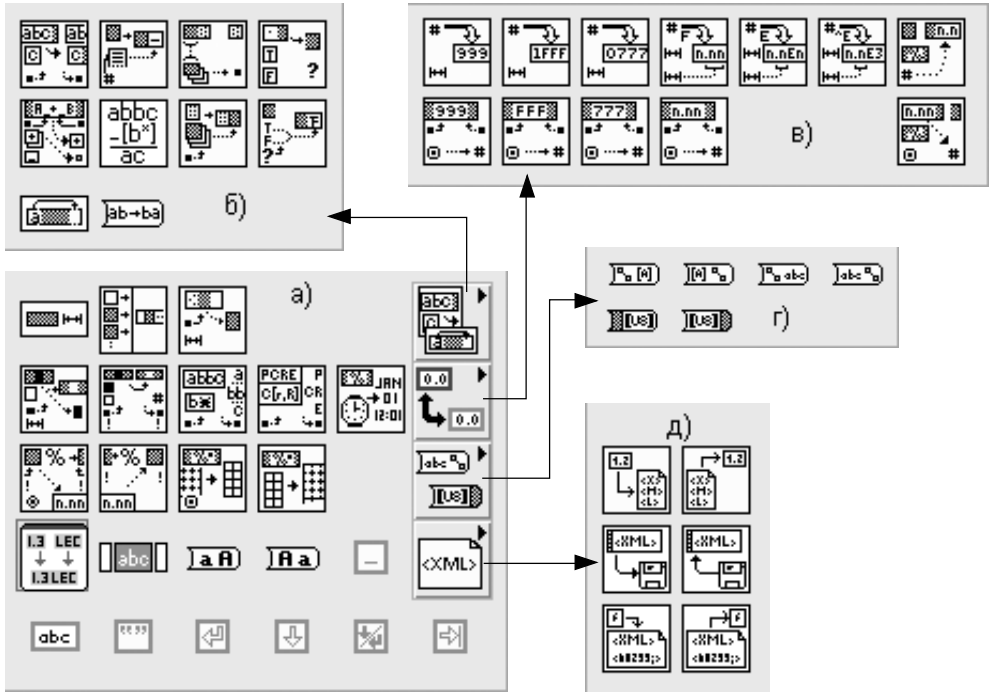
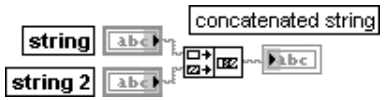
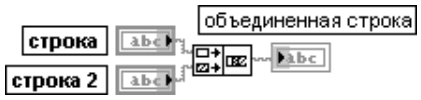


Рис. 2.11. Вид основной палитры (а)
и дополнительных подпалитр (б)–(д) строковых функций

Concatenate Strings

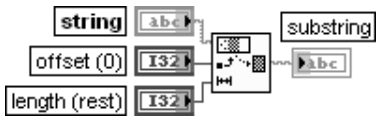


Объединить строки

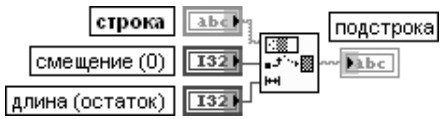


Функция объединяет входные строки и одномерные массивы строк в единственную выходную строку. Для массива строк в объединенную строку входит каждый элемент массива. Добавление/удаление входов функции производится с помощью строки **Добавить вход/Удалить вход** (Add Input/Remove Input) контекстного меню функции или путем изменения размера функции по вертикали с помощью инструмента **Перемещение**

String Subset



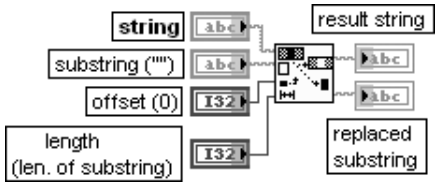
Выделение подстроки



Функция возвращает часть входной **строки** (string), начинающуюся со **смещения** (offset) и содержащую число символов, заданное на входе **длина** (length). Начальный адрес первого символа в строке равен 0.

Выход **подстрока** (substring) является пустым, если смещение больше длины строки или если **длина** меньше или равна 0. Если **длина** больше или равна длине **строки** минус **смещение**, то на выходе **подстрока** выводится остаток строки, начинающийся со смещения

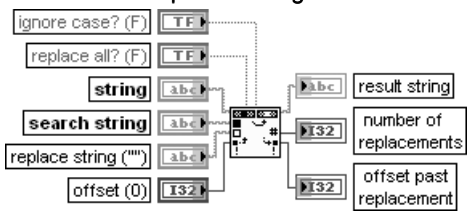
Replace Substring



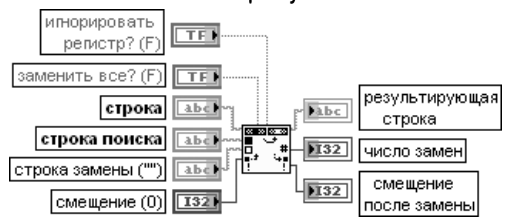
Заменить подстрокой

Функция удаляет часть **строки** (string), количество символов которой задано на входе **длина** (length), начиная со **смещения** (offset), и заменяет удаленную часть строки содержимым **подстроки** (substring).
Если **длина** равна 0, то функция вставляет **подстроку**, начиная со **смещения**. Если **подстрока** является пустой, то функция удаляет число символов, заданное на входе **длина**, начиная со **смещения**.
На выходе **результирующая строка** (result string) выводится строка с замененной или удаленной подстрокой.
На выходе **замененная подстрока** (replaced substring) выводится замененная подстрока

Search and Replace String



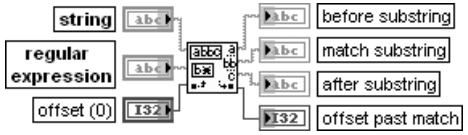
Найти и заменить строку



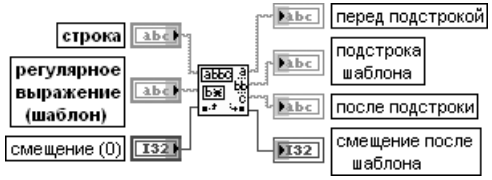
Функция заменяет одну или все образцы подстроки другой подстрокой. Данная функция проверяет **строку** (string) на наличие образцов **строки поиска** (search string), начиная с позиции, заданной величиной **смещения** (offset). Функция заменяет первый встретившийся образец искомой строки на **строку замены** (replace string). Если на входе **заменить все?** (replace all?) установлено состояние ИСТИНА, то функция производит замену всех найденных образцов строки поиска.
Выход **результирующая строка** (result string) содержит **строку** (string) с одной или всеми образцами строки поиска, замененными на строку замены. Если строка замены является пустой, то результирующая строка содержит входную строку с удаленной строкой поиска.
Выход **смещение после замены** (offset past replacement) отображает смещение в результирующей строке символа, находящегося в позиции непосредственно за последним замененным символом. Если **заменить все?** имеет значение ЛОЖЬ, то следующий поиск, если он будет выполняться, начнется с этой точки. Если функция не находит строку поиска, то значение **смещение после замены** равно –1.

Для выполнения более глубокого поиска целесообразно использовать ВП **Найти и заменить шаблон** (Search and Replace Pattern) или функцию **Сопоставить с шаблоном** (Match Pattern)

Match Pattern



Сопоставить с шаблоном



Функция осуществляет поиск **регулярного выражения** (шаблона) (regular expression) в **строке** (string), начиная со **смещения** (offset). При обнаружении **регулярного выражения** строка разделяется на три части: часть строки **перед подстрокой** (before substring), **подстрока шаблона** (match substring) и часть строки **после подстроки** (after substring). Если функция не находит **регулярное выражение**, то выход **подстрока шаблона** будет пустым, на выход **перед подстрокой** будет передана **строка**, а на выходе **смещение после шаблона** (offset past match) установится константа –1. **Смещение** (offset) первого символа в строке равно 0.
При записи **регулярного выражения** для детализации поиска могут использоваться специальные символы, приведенные в таблице.

Специальный символ	Интерпретация символа функцией Сопоставить с шаблоном
.	Определяет совпадение с любым символом в данной позиции. Например, шаблон **l.g** позволит найти слова **lag**, **leg**, **log**, и **lug**
?	Определяет совпадение с одним или меньшим (0) числом символов, предшествовавших **?**. Так, например, шаблон **be?t** позволит найти слова **bt** и **bet**, но нет **best**
\	Отменяет интерпретацию любых специальных символов, приведенных в данной таблице. Например, **\?** определяет совпадение с символом вопроса, а **\.** определяет совпадение с символом точки. Ниже в таблице приведены примеры записи для пробела и неотображаемых символов: \b **удаление символа** (backspace) \s **пробел** (space) \f **конец стоки** (form feed) \r **возврат каретки** (carriage return) \n **новая строка** (newline) \t **табуляция** (tab) \xx любые символы, где **xx** является шестнадцатеричным кодом, в котором используются цифры от **0** до **9** и заглавные буквы от **A** до **F**
^	Если символ **^** является первым символом регулярного выражения, то он привязывает шаблон к смещению в строке . Поиск будет успешным только в случае совпадения регулярного выражения с частью строки, которая начинается от смещения. Если символ **^** не является первым символом, то он воспринимается как символ регулярного выражения
$	Если символ **$** является последним символом регулярного выражения, то он привязывает шаблон к последнему элементу строки. Поиск будет успешным только в случае совпадения регулярного выражения с последними символами строки. Если символ **$** не является последним, он воспринимается как символ регулярного выражения

Специальный символ	Интерпретация символа функцией Сопоставить с шаблоном
[ ]	Заключает альтернативные символы. Например, [**abc**] определяет совпадение с символами **a**, **b**, или **c**. Указанные ниже символы имеют специальное значение, когда используются в скобках следующим образом: –(тире) Указывает диапазон, заданный крайними цифрами или буквами нижнего или верхнего регистров, например [**0-5**], [**a-g**], или [**L-Q**] ~ Определяет совпадение любых символов, включая неотображаемые, за исключением символов или диапазона символов, находящихся в скобках. Например, [**~0-9**] определяет совпадение любых символов за исключением символов цифр из диапазона от **0** до **9** ^ Определяет совпадение любых отображаемых символов, включая пробел и табуляцию, за исключением символов или диапазона символов, находящихся в скобках. Например, [**^0-9**] определяет совпадение любых символов за исключением символов цифр из диапазона от **0** до **9**
+	Определяет совпадение с совокупностью из одного или более символов, предшествующих +. Например, **be+t** определяет совпадение с **bet** и **beet**, но не с **bt**
*	Определяет совпадение с любым числом (включая 0) символов, предшествующих *. Например, **be*t** определяет совпадение с **bt**, **bet** и **beet**

Следующая таблица показывает примеры записи регулярных выражений и обнаруженных фрагментов строк для функции **Сопоставить с шаблоном**.

Регулярное выражение	Обнаруженные символы
VOLTS	VOLTS
[**Vv**][**Oo**][**Ll**][**Tt**][**Ss**]	Все варианты слова **volts**, содержащие буквы верхнего и нижнего регистра, такие как **VOLTS**, **Volts**, **volts**
[_+ –]	Пробел, знак плюса или знак минуса
[**0-9**]+	Последовательность одной или более цифр
\s* или _* (последнее означает пробел перед *)	Нулевое или большее число пробелов
[\t \r \n \s]+	Один или более пробелов, символов табуляции, новой строки или возврата каретки
[~**0-9**]+	Один или более символов, отличающихся от цифр
^Level	Слово **Level**, если только оно начинается в строке с позиции смещения
Volts$	Слово **Volts**, если только оно находится в конце строки
(**.***)	Наиболее длинная строка в круглых скобках
([~**( )**]*)	Наиболее длинная строка в круглых скобках, но не содержащая круглые скобки

Ниже на рис. 2.12 приведена блок-диаграмма ВП **Извлечь числа** (Extract Numbers) из набора примеров NI Example Finder, в котором функция **Сопоставить с шаблоном** используется для поиска чисел в строке и вывода их в виде числового массива и массива строк (рис. 2.13). При этом числа могут иметь любой из следующих форматов: 123 1.23 .123 0.123 -1.23.

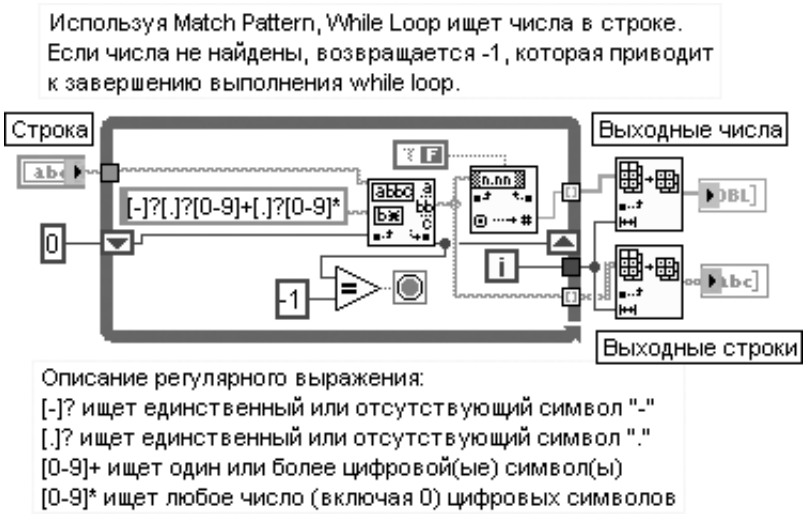


Рис. 2.12. Блок-диаграмма ВП ***Извлечь числа***

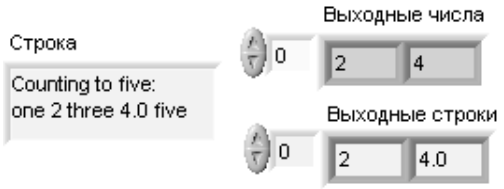
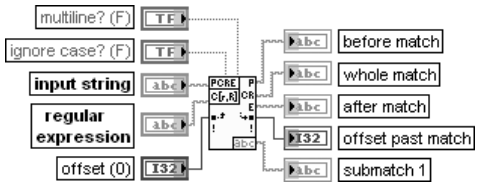
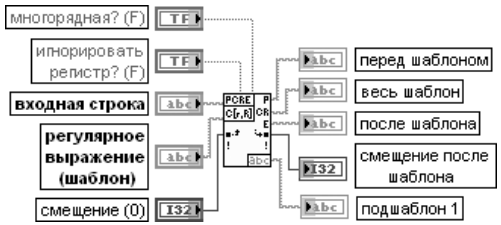


Рис. 2.13. Вид лицевой панели ВП

Match Regular Expression



Сопоставить с регулярным выражением



Функция осуществляет поиск **регулярного выражения** (regular expression) во **входной строке** (input string), начиная с установленного **смещения** (offset). При обнаружении **регулярного выражения** строка разделяется на три подстроки и любое число **подшаблонов** (submatches). Для просмотра найденных подшаблонов необходимо увеличить размер иконки функции в вертикальном направлении с помощью инструмента перемещения.

Эта функция дает больше возможностей для поиска заданных фрагментов строки, чем рассмотренная выше функция **Сопоставить с шаблоном**, однако она выполняется медленнее.

Вход **многорядная?** (multiline?) устанавливает возможность восприятия входной строки как многорядной. Он влияет на обработку совпадений символов ^ и $. При установке на входе состояния ЛОЖЬ (по умолчанию) ввод «^» обрабатывает совпадение только с началом **входной строки**, а ввод «$» – только с ее концом. При установке на входе состояния ИСТИНА «^» обрабатывает совпадение с началом любого ряда **входной строки**, а «$» – с его концом.

Вход **игнорировать регистр** (ignore case?) определяет чувствительность к регистру процедуры поиска в строке. Если на входе установлено состояние ЛОЖЬ (по умолчанию), то процедура чувствительна к регистру.

На вход **входная строка** (input string) подается строка, в которой производится поиск.

На входе **регулярное выражение** (regular expression) задается шаблон, который должен быть найден во **входной строке**. Если функция не находит совпадения, то выходы **весь шаблон** и **после шаблона** отображают пустую строку, на выходе **перед шаблоном** выводится вся входная строка, выход **смещение после шаблона** (offset past match) возвращает –1, а все выходы **подшаблонов** возвращают пустые строки. Искомые подстроки необходимо заключать в круглые скобки.

Вход **смещение** (offset) определяет номер символа во входной строке, с которого функция начинает поиск.

Выход **перед шаблоном** (before match) возвращает строку, содержащую все символы перед совпавшей частью.

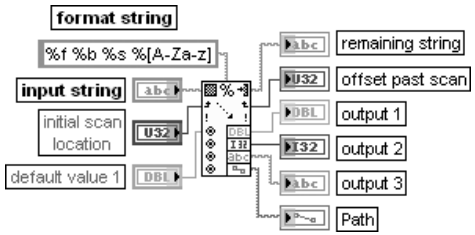
Выход **весь шаблон** (whole match) содержит все символы, совпавшие с выражением, которое было подано на вход **регулярное выражение**. Любые совпадения подстроки, которые находятся функцией, появляются на выходах **подшаблоны**.

Выход **после шаблона** (after match) содержит все символы, следующие за совпавшей частью шаблона, введенного на входе **регулярное выражение**.

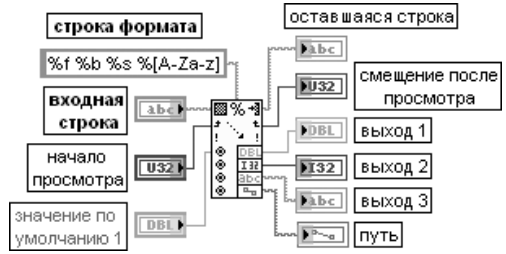
Выход **смещение после шаблона** (offset past match) возвращает указатель первого символа **входной строки**, находящегося после последнего совпавшего фрагмента. Если функция не находит совпадения, то на этом выходе возвращается –1.

Некоторые регулярные выражения, использующие такие последовательности, как (.|\s)*, требуют значительных ресурсов для обработки длинных входных строк. В некоторых случаях могут происходить переполнение стека и вывод сообщения об ошибке. Для устранения ошибки необходимо упростить регулярное выражение или уменьшить размер анализируемой строки.

Scan From String



Просмотр строки



Функция просматривает **входную строку** (input string) с точки **начало просмотра** (initial scan location) и преобразует ее в соответствии с форматом, заданным на входе **строка формата** (format string). Данную функцию целесообразно использовать в случае, когда точно известен формат входной строки.

Вход **строка формата** определяет, как необходимо преобразовывать входную строку в выходные аргументы. По умолчанию такое преобразование осуществляется в соответствии с типом выходных выводов. Тип выходов и соответствующие разделы **строки**

формата могут быть установлены или изменены с помощью диалогового окна, вызываемого с помощью строки **Редактировать строку просмотра** (Edit Scan String) контекстного меню функции.
В данной функции при определении формата используется следующая запись с применением упрощенных синтаксических элементов:
%[Width]Conversion Code,
где % – символ, с которого начинается определение формата;
[Width] – число, определяющее **ширину** используемого поля (необязательный параметр). LabVIEW сканирует только заданное число символов при обработке параметра. Если ширина не задана или равна 0, то для выходного параметра выделяется такая ширина, какая необходима для его представления.
Код преобразования (Conversion Code) – единичные символы, которые определяют способ сканирования или форматирования параметра. Коды преобразования могут быть прописными или строчными, за исключением кодов формата времени, которые чувствительны к регистру.
Перечень кодов преобразования приведен в таблице.

Коды преобразования для целых чисел:
- **x** – шестнадцатеричное целое (например, B8);
- **o** – восьмеричное целое (например 701);
- **b** – двоичное целое (например 1011);
- **d** – десятичное целое со знаком;
- **u** – десятичное целое без знака

Коды преобразования для чисел с плавающей запятой:
- **f** – число с плавающей запятой с дробным форматом (например, 12,345);
- **e** – число с плавающей запятой в научной нотации (например, 1,234E1);
- **g** – число с плавающей запятой в инженерной нотации. При этом LabVIEW использует **f** или **e** в зависимости от экспоненты числа. LabVIEW использует **f** если экспонента больше чем –4 или меньше заданной точности. LabVIEW использует **e** если экспонента меньше чем –4 или больше заданной точности;
- **p** – число с плавающей запятой в записи СИ. При такой записи вместо экспоненты числа выводится буквенное обозначение, соответствующее заданной степени (таблица).

y	z	a	f	p	n	u	m
yocto (10^{-24})	zepto (10^{-21})	atto (10^{-18})	femto (10^{-15})	pico (10^{-12})	nano (10^{-9})	micro (10^{-6})	milli (10^{-3})

k	M	G	T	P	E	Z	Y
kilo (10^{3})	mega (10^{6})	giga (10^{9})	tera (10^{12})	peta (10^{15})	exa (10^{18})	zetta (10^{21})	yotta (10^{24})

Код преобразования для строк включает символ **s**;
s – строка (например, **abc**), которая определяется только до следующего пробела. Пробел определяется одним или большим числом символов пробела

Коды преобразования для значения времени:
- **T** – абсолютное время;
- **t** – относительное время.

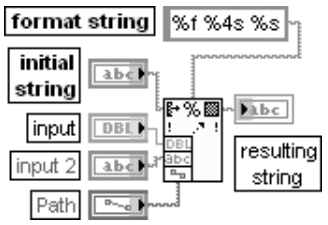
T и **t** могут использоваться только в элементах управления, константах и индикаторах

Вход **начало просмотра** (initial scan location) задает смещение в строке, с которого начинается сканирование. По умолчанию его значение равно 0.

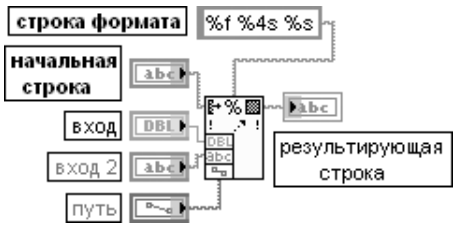
Входы **по умолчанию 1..n** (default 1..n) определяют тип и значение по умолчанию входных параметров. Если входные значения не могут быть найдены в строке, то функция **Просмотр строки** использует значения по умолчанию. Если входы **по умолчанию 1..n** не подключены, то тип выходного значения определяется **строкой формата**, если она является константой. В противном случае значение по умолчанию имеет тип числа с плавающей запятой двойной точности. Значение по умолчанию является нулем или пустой строкой в зависимости от типа выходных данных.
Выход **оставшаяся строка** (remaining string) возвращает часть строки, которая осталась после просмотра всех аргументов.
Выход **смещение после просмотра** (offset past scan) отображает смещение во **входной строке** (input string) после выполнения просмотра.
Выходы **выход 1..n** (output 1..n) определяют выходные параметры. Каждый выход может быть строкой, путем, типом перечисления или любым числовым типом. С этой функцией не могут использоваться массивы и кластеры.
В таблице приведены примеры использования функции **Просмотр строки**.

Входная строка	Строка формата	По умолчанию	Выходы
abc, xyz 12,3+56i 7200	%3s, %s%f%2d	– – 0,00 +0.00 i –	abc xyz 12,30 +56,00 i 72
Q+1,27E-3 tail	Q%f t	–	1,27E-3
0123456789	%3d%3d	–	12 345
X:9,860 Z:3,450	X:%fY:%f	100 (I32) 100,00 (DBL)	10 100,00
set49,4,2	set%d	–	49
color: red	color: %s	blue (enum {red, green, blue})	red
abcd012xyz3	%[a-z]%d%[a-z]%d	–	abcd 12 xyz 3
welcome to LabVIEW, John Smith	%[^,],%s	–	welcome to LabVIEW John

Format Into String



Преобразовать в строку

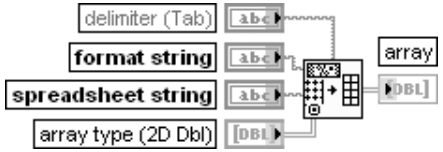


Функция форматирует строки, пути, числовые или логические данные как текст и преобразует входные аргументы в результирующую строку.
В данной функции при задании формата используется следующая запись с применением упрощенных синтаксических элементов:
[-][+][#][^][0][Width][.Precision || _SignificantDigits][{Unit}][<Embedded information>]
Conversion Code

Часть параметров этого выражения была рассмотрена выше при анализе функции **Просмотр строки**.
Далее в таблице приведены только элементы, не рассмотренные ранее.

Синтаксический элемент	Описание
– (тире)	Указывает на выравнивание по ширине влево
+ (плюс)	Используется с числовыми параметрами, вызывает вывод знака числа
. (точка)	Символ, разделяющий значения **ширина** (Width) и **точность** (Precision)
точность (Precision)	Число, определяющее количество цифр справа от десятичной запятой в цифровом поле, когда на вход **число** (number) подается число с плавающей запятой. Если за параметром **ширина** не следует «точка», то дробная часть будет состоять из шести цифр. Если после параметра **ширина** следует «точка», а параметр **точность** отсутствует или равен 0, то дробная часть отсутствует
единица{unit}	Истинная единица измерения

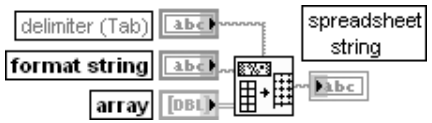
Spreadsheet String To Array



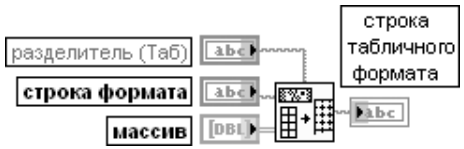
Строку табличного формата в массив

Функция преобразует **строку табличного формата** (spreadsheet string) в цифровой **массив** (array), размерность которого определяется входом **тип массива** (array type). Функция одинаково работает как с массивами строк, так и с массивами чисел. Символ **табуляция** (Tab) разделяет столбцы **строки табличного формата**, а символ **конец стоки** (EOL) разделяет строки. Функция преобразует каждый элемент **строки табличного формата** в соответствии с форматом, указанным на входе **строка формата** (format string), а затем запоминает их в **массиве** (array). Если вход **тип массива** не подключен, то тип массива будет двумерным с числами, представленными в формате с плавающей запятой с двойной точностью

Array To Spreadsheet String



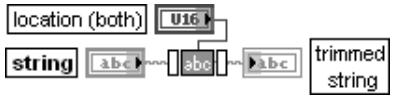
Массив в строку табличного формата



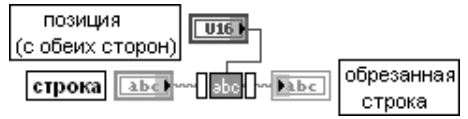
Функция преобразует числовой **массив** (array) любой размерности в **строку табличного формата** (spreadsheet string), в которой символ **табуляция** (tab) отделяет столбцы элементов, а символ **конец строки** (EOL) разделяет строки. Для трехмерных (и более) массивов выделяются **страницы** (pages), как это описано ниже. Функция преобразует все элементы массива в соответствии со **строкой формата** (format string), а затем присоединяет их к **строке табличного формата**.

Для преобразования массива строк в **строку табличного формата** можно указать формат строки **%s**, а для преобразования массива данных – формат **%d** или **%f**.
Для трехмерных (и более) массивов каждой странице предшествует серия индексов следующего формата: [**n,m,...0,0**],
где **n** - самый большой индекс размерности,
m - второй по величине индекс размерности,
0,0 - указывает на элементы первой строки и первого столбца страницы (**n,m**)

Trim Whitespace



Обрезать пробелы



ВП удаляет неотображаемые символы в начале и/или в конце строки.
В состав неотображаемых символов входят символы **табуляция** (tab), **новая строка** (newline), **возврат каретки** (carriage return) и **пробел** (space). Как видно из блок-диаграммы ВП (рис. 2.14), для поиска неотображаемых символов используется функция **Match Pattern**.

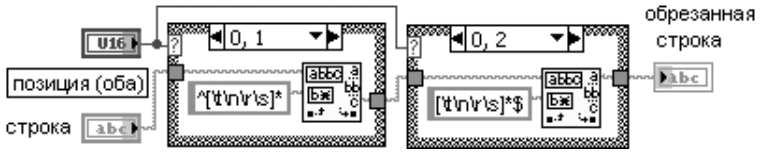
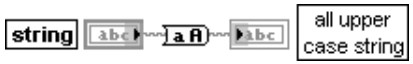


*Рис. 2.14. Блок-диаграмма ВП **Обрезать пробелы** (Trim Whitespace)*

Вход **позиция** (location) кольцевого типа определяет часть строки, в которой производятся поиск и удаление неотображаемых символов. При выборе состояния **с обеих сторон** (both) (по умолчанию) поиск и удаление производятся в начале и в конце строки. Соответственно при выборе состояния **начало строки** (start of string) удаляются неотображаемые символы в начале, а при выборе **конец строки** (end of string) – в конце строки.

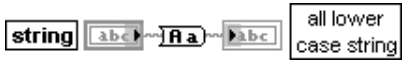
To Upper Case



К верхнему регистру

Функция преобразует буквенные символы **строки** (string) в символы верхнего регистра. Воспринимает все числа в **строке** как ASCII коды символов. Эта функция не действует на символы, не являющиеся буквенными

To Lower Case



К нижнему регистру

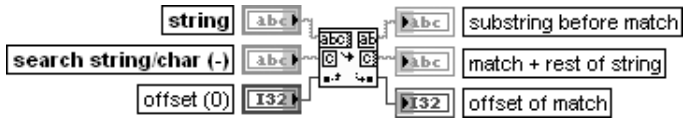
Функция преобразует буквенные символы **строки** (string) в символы нижнего регистра. Воспринимает все числа в **строке** как ASCII коды символов. Эта функция не действует на символы, не являющиеся буквенными

Таблица строковых констант:

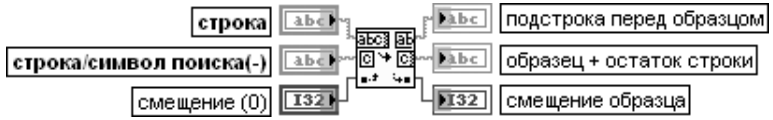
Символ	ASCII код (десятичный)	Название
	32	**Пробел** (Space Constant)
abc		**Строковая константа** (String Constant)
		Пустая строка (Empty String Constant)
	13	**Возврат каретки** (Carriage Return Constant – CR)
	10	**Перевод строки** (Line Feed Constant – LF)
		Конец строки (End of Line Constant) в Windows соответствует последовательности констант CR/LF
	9	**Горизонтальная табуляция** (Tab Constant – Tab)

Таблицы **дополнительных строковых функций** (Additional String Functions):

Search/Split Siring

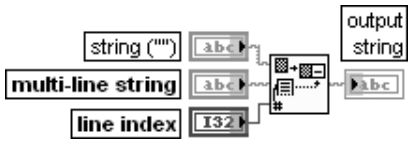


Найти/Разделить строку

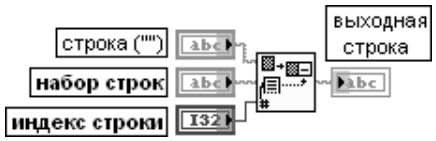


Функция разделяет строку на две части по **строке или символу поиска** (search string/char), начиная от **смещения** (offset). Если функция не находит символ или строку поиска, то на выходе **смещение образца** (offset of match) будет установлена константа –1, на выход **подстрока перед образцом** (substring before match) передается входная строка, а на выходе **образец+остаток строки** (match+rest of string) возвращается пустая строка. Если вход **строка или символ поиска** (search string/char) не подключен или на него подана пустая строка, то функция делит строку по **смещению**

Pick Line



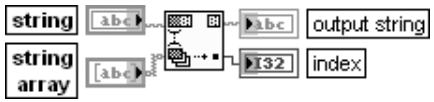
Выбрать строку



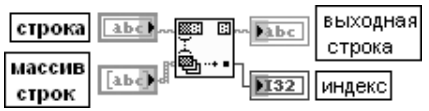
Функция выбирает строку из **набора строк** (multi-line string) по **индексу строки** (line index), присоединяет ее к **строке** (string) и вновь образованную строку передает на выход **выходная строка** (output string).

По умолчанию на вход **строка** подается пустая строка.
Набор строк состоит из одной или более подстрок, разделенных символами «перевод строки».
Вход **индекс строки** служит для выбора строки и должен быть числовым. Нулевой индекс соответствует первой строке. Если индекс строки отрицательный, больше или равен количеству строк в наборе строк, то функция передает на выход **выходная строка** содержимое входа **строка**. Если **индекс строки** является дробным числом, то функция округляет его до целого

Match First String

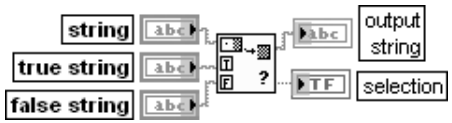


Сопоставить первую строку

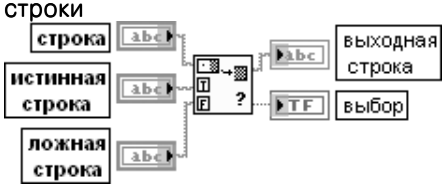


Функция сопоставляет каждую строку из **массива строк** (string array) с началом образцовой **строки** (string) и определяет **индекс** (index) строки при обнаружении совпадения.
Вход **строка** является строкой, используемой для поиска приставок в массиве строк. По умолчанию это пустая строка.
Вход **массив строк** представляет массив строк, которые сопоставляются с образцовой строкой. Если функция встречает пустую строку в массиве строк, то она рассматривает ее как совпадение.
Выход **выходная строка** (output string) содержит исходную строку с удаленными приставками, которые были обнаружены. Если начало **строки** не совпало ни с одной строкой в **массиве строк**, то **выходная строка** соответствует исходной **строке**.
Выход **индекс** является индексом найденной приставки в **массиве строк**. Если начало образцовой **строки** не совпало с какой-либо строкой в **массиве строк**, то индекс принимает значение –1

Match True/False String

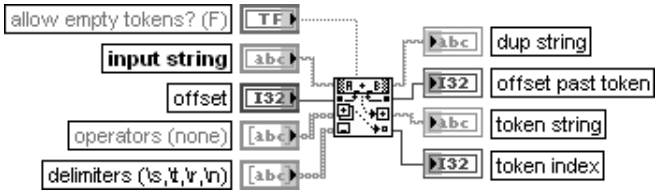


Сопоставить истинную или ложную строки

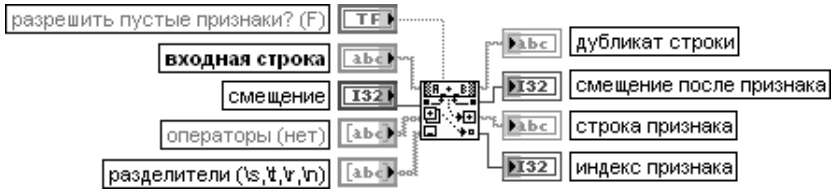


Функция сопоставляет начальную часть **строки** (string) со строками **истинная строка** (true string) или **ложная строка** (false string). Функция возвращает логическое значение ИСТИНА или ЛОЖЬ на выходе **выбор** (selection) в зависимости от того, с какой из строк – истинной или ложной – произошло совпадение.
На выход **выходная строка** (output string) передается содержимое входной **строки** с удаленной совпавшей начальной частью строки. Если сравнения не произошло, то на выход **выходная строка** передается входная **строка**, а на выход **выбор** – значение ЛОЖЬ

Scan String for Tokens



Просмотр строки на наличие строки признаков



Функция проверяет **входную строку** (input string), начиная от **смещения** (offset), на наличие строки **признаков** (token), перечисленных в массиве **операторы** (operators) или окруженных разделителями, перечисленными в массиве **разделители** (delimiters). Как правило, искомые строки признаков представляют ключевые слова, числовые значения или операторы языка, распознаваемые при анализе текстовых документов.

Если вход **разрешить пустые признаки?** (allow empty tokens?) установлен в состояние ЛОЖЬ, то два соседних разделителя могут разделять две искомые строки признаков, в противном случае на выход **строка признака** (token string) возвращается пустая строка. Если входная строка содержит фрагменты, совпадающие с несколькими элементами массива **операторы**, то выбирается наиболее длинный фрагмент. Элементы массива **операторы** могут содержать специальные коды форматирования, которые позволяют находить все числа как простые фрагменты:

% - задает поиск десятичных целых чисел;
%o – поиск восьмеричных целых чисел;
%x – поиск шестнадцатеричных целых чисел;
%b – поиск двоичных целых чисел;
%e, %f, %g – поиск вещественных чисел с плавающей запятой или чисел в научном формате;
%% – поиск символа %.

Если вход **разделители** (delimiters) не подключен, то в качестве разделителей используются неотображаемые символы – **пробел** (space), **табуляция** (tab), **перевод строки** (new line), **возврат каретки** (carriage return).

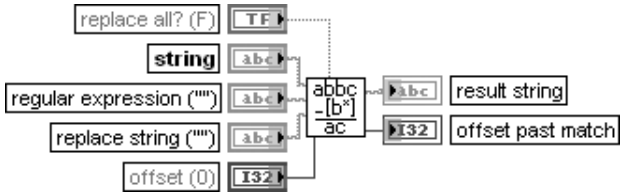
На выход **дубликат строки** (dup string) передается неизменная входная строка.

Выход **смещение после признака** (offset past token) содержит индекс символа последнего фрагмента, найденного во входной строке.

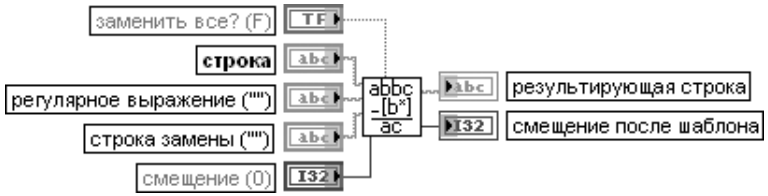
На выход **строка признаков** (token string) передается найденная строка признаков входной строки. Если этот фрагмент соответствует фрагменту, находящемуся в массиве **операторы**, то на выход **индекс признака** выводится индекс фрагмента в данном массиве. Если этот фрагмент не содержится в массиве **операторы**, то выводится –1. При отсутствии найденных фрагментов на выходе **индекс признака** выводится –2.

Таким образом, рассмотренную строковую функцию целесообразно применять для поиска фрагментов в строке в структуре **Цикл по условию**, возвращая значение **смещение после признака** с помощью сдвигового регистра на вход **смещение** для продолжения поиска в оставшемся фрагменте строки. При этом появление значения –2 на выходе **индекс признака** можно использовать для прекращения поиска

Search and Replace Pattern



Найти и заменить шаблон



ВП ищет в **строке** (string) подстроку, которая соответствует **регулярному выражению** (regular expression), и заменяет ее подстрокой, находящейся на входе **строка замены** (replace string). Этот ВП разработан на основе функции **Сопоставить с шаблоном** (Match Pattern) (рис. 2.13) и, соответственно, имеет близкую к ней функциональность.

Если вход **заменить все?** (replace all?) находится в состоянии ИСТИНА, то ВП заменяет все подстроки в **строке** (string), соответствующие регулярному выражению. Если этот вход находится в состоянии ЛОЖЬ (по умолчанию), то ВП заменяет только первую найденную подстроку.

Вход **строка** представляет входную строку, в которой производится поиск.

Вход **регулярное выражение** (regular expression) задает регулярное выражение (шаблон), по которому производится поиск в строке. Особенности формирования регулярного выражения были рассмотрены при анализе функции **Сопоставить с шаблоном** (Match Pattern).

Если ВП не находит регулярное выражение, то **результирующая строка** (result string) будет содержать входную строку, а на выходе **смещение после шаблона** (offset past match) будет установлено значение –1.

Если регулярное выражение соответствует пустой строке, то ВП не производит замену, результирующая строка будет содержать входную строку, а на выходе **смещение после шаблона** возвращается 0 или длина строки, зависящая от того, установлен ли вход **заменить все?** в состояние ЛОЖЬ или ИСТИНА соответственно.

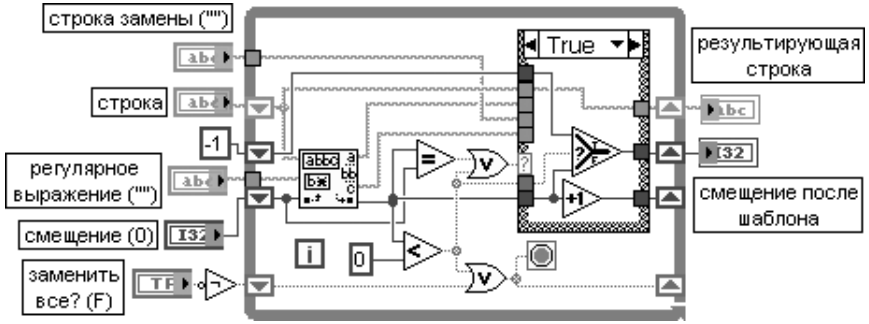
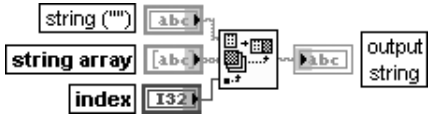


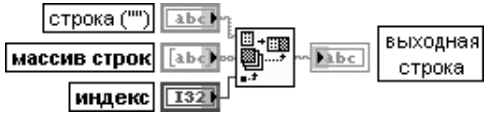
*Рис. 2.15. Блок-диаграмма ВП **Найти и заменить шаблон** (Search and Replace Pattern)*

Вход **строка замены** (replace string) определяет подстроку, которая используется для замены части строки, соответствующей регулярному выражению. По умолчанию это пустая строка.
Вход **смещение** является начальной позицией и должен быть числом. Смещение первого символа в строке равно 0. Если вход не подключен или меньше 0, то по умолчанию его значение равно 0.
Выход **результирующая строка** содержит отредактированную строку с замененными символами.
Выход **смещение после шаблона** содержит индекс в строке первого символа, расположенного после последнего найденного фрагмента. Если ВП не находит шаблон, на этом выходе устанавливается –1.

Index String Array

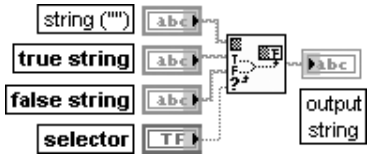


Выбрать строку по индексу

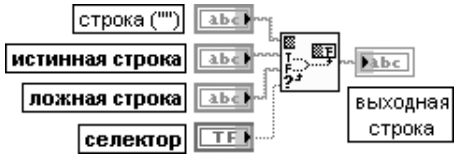


Функция выбирает строку из **массива строк** (string array) по **индексу** и присоединяет ее к **строке** (string)

Append True/False String

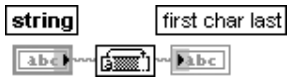


Добавить строку по выбору



Функция выбирает одну из строк – **истинная строка** (true string) или **ложная строка** (false siring) – в зависимости от состояния логического входа **селектор** (selector), присоединяет выбранную строку к **строке** (string) и передает образованную строку на выход **выходная строка** (output string)

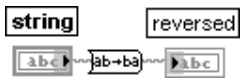
Rotate String



Циклически сдвинуть символы строки

Функция перемещает первый символ **строки** (string) в последнюю позицию выходной строки **первый символ последним** (first char last), сдвигая все другие символы на одну позицию влево. Например, строка **abcd** станет строкой **bcda**.
Вход **строка** может быть строкой, кластером строк или массивом кластеров строк.
Выход **первый символ последним** является циклически сдвинутой строкой и имеет ту же структуру, что и входная строка

Reverse String

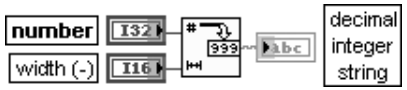


Обратить строку

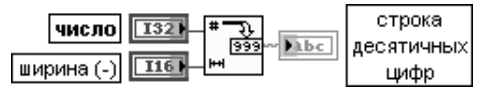
Функция формирует строку, символы которой расположены в обратном порядке по отношению к входной **строке** (string).
Вход **строка** может быть строкой, кластером строк или массивом кластеров строк

Таблицы **функций взаимного преобразования строк и чисел** (String/Number Conversion):

Number To Decimal String



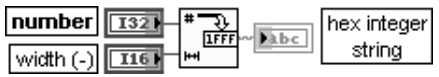
Число в строку десятичных цифр



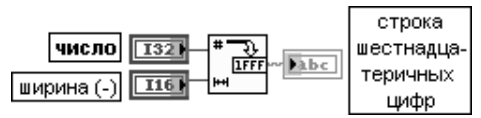
Функция преобразует **число** (number) в **строку десятичных цифр** (decimal integer string) с шириной, равной или большей значения, установленного на входе **ширина** (width). Если **число** дробное, то оно округляется до ближайшего целого.
Число **может быть скаляром, массивом или кластером чисел, массивом кластеров чисел и т. д.**
В таблице показано влияние числовых параметров на входах **число** и **ширина** на выходную строку **строка десятичных цифр**. Здесь и далее символ _ обозначает пробел.

Число	Ширина	Строка десятичных цифр	Комментарии
4,6	2	_5	Числа с плавающей запятой округляются до целых
3,0	4	_ _ _3	Если **ширина** больше необходимой, то слева добавляются пробелы
–311	3	–311	Если **ширина** неадекватна, то **строка десятичных цифр** имеет необходимую ширину

Number To Hexadecimal String



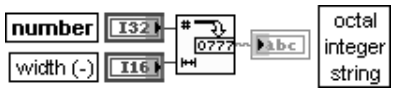
Число в строку шестнадцатеричных цифр



Функция преобразует **число** (number) в **строку шестнадцатеричных цифр** (hex integer string) с шириной, равной или большей значения, установленного на входе **ширина** (width). Цифры A–F всегда отображаются в выходной строке в верхнем регистре. Если **число** дробное, то оно округляется до 32-битового целого перед преобразованием.
Таблица показывает, как числовые параметры на входах **число** и **ширина** влияют на выходную строку **строка шестнадцатеричных цифр**.

Число	Ширина	Строка шестнадцатеричных цифр	Комментарии
3	4	0003	Если **ширина** больше необходимой, то слева добавляются нули
42	3	02A	–
–4,2	3	FFFFFFFC	–4,2 округляется до –4 в формате 32-битового целого. Ширина недостаточна для отображения шестнадцатеричной версии отрицательного числа, поэтому ширина поля увеличена

Number To Octal String

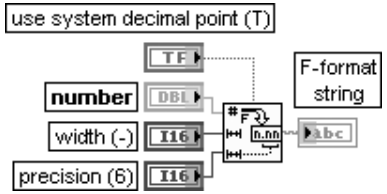


Число в строку восьмеричных цифр

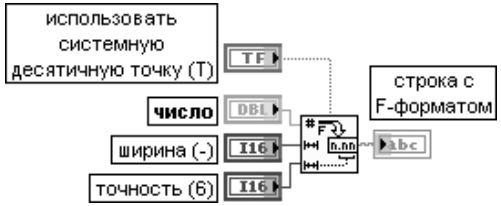
Функция преобразует **число** (number) в **строку восьмеричных цифр** (octal integer string) с шириной, равной или большей значения, установленного на входе **ширина** (width). Если **число** дробное, то оно округляется до 32-битового целого перед преобразованием.
Таблица показывает, как числовые параметры на входах **число** и **ширина** влияют на выходную строку **строка восьмеричных цифр**.

Число	Ширина	Строка восьмеричных цифр	Комментарии
3	4	0003	–
42	3	052	–
–4,2	3	37777777774	–4,2 округляется до –4 в формате 32-битового целого. Ширина недостаточна для отображения восьмеричной версии отрицательного числа, поэтому ширина поля увеличена

Number To Fractional String



Число в строку с дробным форматом

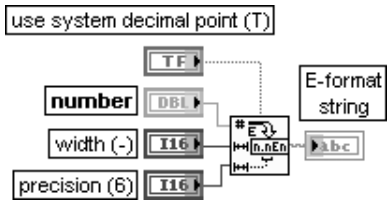


Функция преобразует **число** (number) в **строку с F-форматом** (дробная запись) представления числа с плавающей запятой, имеющую количество символов, равное или большее значения, заданного на входе **ширина** (width).
Вход **использовать системную десятичную точку** (use system decimal point) определяет десятичный разделитель. Если он имеет значение ИСТИНА (по умолчанию), то в качестве десятичного разделителя используется локализованный десятичный разделитель. Если он имеет значение ЛОЖЬ, то десятичным разделителем является точка.
Вход **число** может быть скаляром, массивом или кластером чисел, массивом кластеров чисел и т. д.
Вход **ширина** должен быть числовым. Если он не подключен, то функция использует столько цифр, сколько необходимо для представления числа без излишнего дополнения.
Вход **точность** (precision) должен быть числовым. Функция округляет число цифр после десятичной точки в выходной строке до числа, задаваемого на входе **точность**. Если значение **точность** равно 0, то выходная строка не содержит десятичную точку и содержит по крайней мере три цифры мантиссы. По умолчанию значение **точность** равно 6.
Выход **строка с F-форматом** (F-format string) представляет результирующую дробную строку. **Строка с F-форматом** может быть **Inf**, **-Inf** или **NaN**, если значение, которое подключено ко входу **число**, является бесконечным или не является числом.

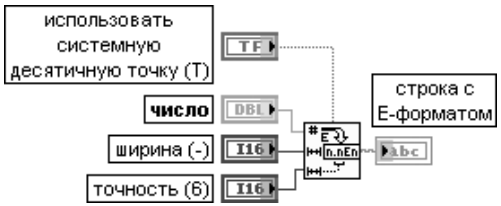
Таблица показывает, как числовые параметры на входах **число**, **ширина** и **точность** влияют на выходную строку **строка с F-форматом**.

Число	Ширина	Точность	Строка с F-форматом	Комментарии
4,911	6	2	_ _4,91	Число округляется и дополняется пробелами слева
,0039268		4	_ _0,0039	Число округляется и дополняется пробелами слева
–287,3	5	0	_–287	Число округляется и дополняется пробелами слева

Number To Exponential String



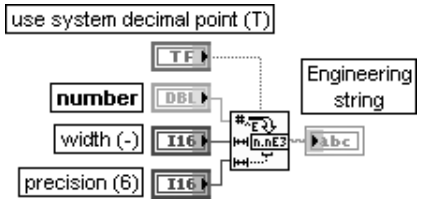
Число в строку с экспоненциальным форматом



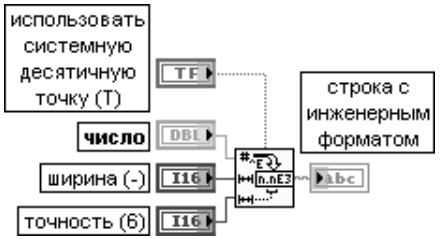
Функция преобразует **число** (number) в **строку с E-форматом** (экспоненциальная запись) представления числа с плавающей запятой, имеющую количество символов, равное или большее значения, заданного на входе **ширина** (width).
Назначение входов идентично назначению одноименных входов рассмотренной выше функции **Число в строку с дробным форматом** (Number To Fractional String).
Таблица показывает, как числовые параметры на входах **число**, **ширина** и **точность** влияют на выходную строку **строка с E-форматом**.

Число	Ширина	Точность	Строка с E-форматом	Комментарии
4,911	5	2	4,91e0	Число округляется, ширина увеличивается
,003926	10	2	_ _ _3,93e–3	Число округляется и дополняется пробелами слева
216,01	5	0	_ _2e2	Число округляется и дополняется пробелами слева

Number To Engineering String



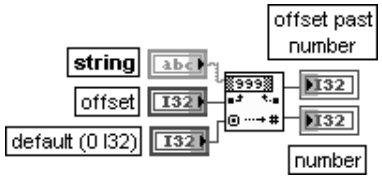
Число в строку с инженерным форматом



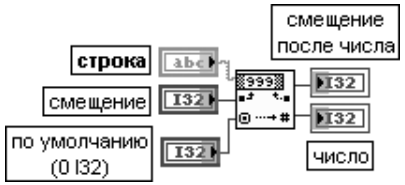
Функция преобразует **число** (number) в **строку с инженерным форматом** представления числа с плавающей запятой, имеющую количество символов, равное или большее значения, заданного на входе **ширина** (width). Инженерный формат аналогичен экспоненциальному, за исключением того, что показатель экспоненты кратен трем, то есть имеет значения (.., –3, 0, 3, 6,..).
Назначение входов функции идентично назначению одноименных входов рассмотренной выше функции **Число в строку с дробным форматом**.
Таблица показывает, как числовые параметры на входах **число**, **ширина** и **точность** влияют на выходную строку **строка с инженерным форматом**.

Число	Ширина	Точность	Строка с инженерным форматом	Комментарии
4,93	10	2	_ _ _ _4,93e0	Число округляется и дополняется пробелами слева
,49	10	2	_ _ _ _490e–3	Число округляется и дополняется пробелами слева
61,96	8	1	_ _62,0e0	Число округляется и дополняется пробелами слева
1789,32	8	2	_ _1,79e3	число округляется и дополняется пробелами слева

Decimal String To Number



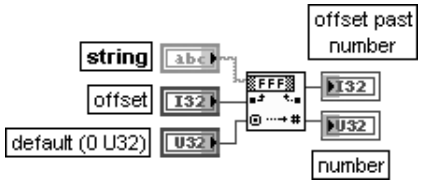
Строку десятичных цифр в число



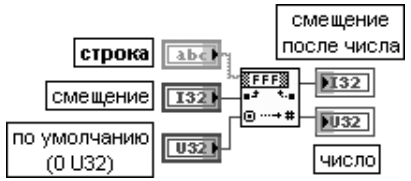
Функция преобразует цифровые символы **строки** (string), начиная от **смещения** (offset), в целое десятичное число и передает его на выход **число** (number).
Если указанный начальный элемент строки не цифра, то функция передает на выход **число** (number) **значение по умолчанию** (default). Если вход **значение по умолчанию** не подключен, то его состояние определяется как 0. Выход **смещение после числа** (offset past number) определяет индекс первого элемента строки, следующего за числом.
Таблица показывает, как числовые параметры на входах **строка**, **смещение** и **по умолчанию** влияют на выходное число.

Строка	Смещение	По умолчанию	Смещение после числа	Число	Комментарии
13ax	0	0	2	13	–
–4,8bcd convers	0	0	2	–4	Поскольку целое число было преобразовано, то преобразование остановилось на десятичной запятой
a49b	0	–9	0	–9	Используется значение по умолчанию, поскольку никаких цифр не было считано

Hexadecimal String To Number



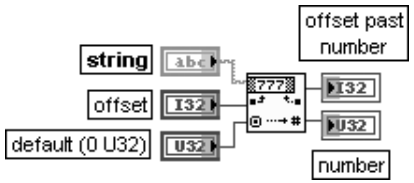
Строку шестнадцатеричных цифр в число



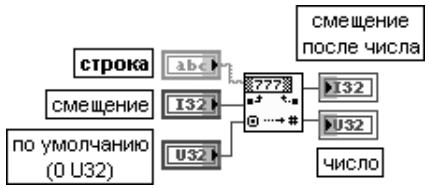
Функция преобразует символы от 0 до 9 и от **A** до **F** (или от **a** до **f**) **строки** (string), начиная от **смещения** (offset), в целое шестнадцатеричное число и передает его на выход **число** (number).
Назначение входов и выходов идентично рассмотренной выше функции **Строку десятичных цифр в число**.
Таблица показывает, как числовые параметры на входах **строка**, **смещение** и **по умолчанию** влияют на выходное число.

Строка	Смещение	По умолчанию	Смещение после числа	Число	Комментарии
f3g	0	0	2	243	**g** не является допустимым шестнадцатеричным символом, поэтому преобразование на нем заканчивается
–30	0	0	0	0	Отрицательные числа не разрешены для шестнадцатеричного представления

Octal String To Number



Строку восьмеричных цифр в число

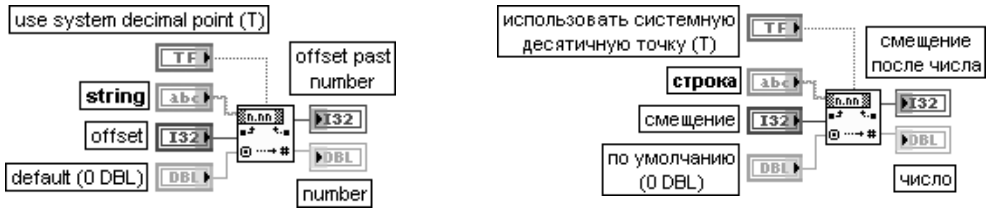


Функция преобразует символы от 0 до 7 **строки** (string), начиная от **смещения** (offset), в целое восьмеричное число и передает его на выход **число** (number).
Назначение входов и выходов идентично рассмотренной выше функции **Строку десятичных цифр в число**.
Таблица показывает, как числовые параметры на входах **строка**, **смещение** и **по умолчанию** влияют на выходное число.

Строка	Смещение	По умолчанию	Смещение после числа	Число	Комментарии
92	0	0	0	0	9 не является восьмеричной цифрой
071a	0	0	3	57	Символ **a** не является восьмеричной цифрой, поэтому преобразование заканчивается на нем

Fract/Exp String To Number

Строку цифр в дробном/экспоненциальном формате в число



Функция интерпретирует символы от 0 до 9, плюс, минус, e, E и десятичную точку **строки** (string), начиная от **смещения** (offset), как число с плавающей запятой в инженерной записи, экспоненциальном или дробном формате и передает его на выход **число** (number).

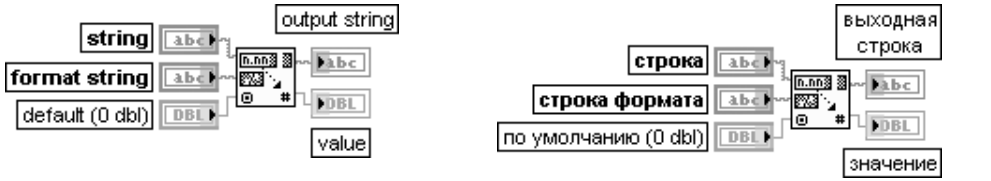
Назначение входов и выходов идентично рассмотренной выше функции **Строку десятичных цифр в число**.

Таблица показывает, как числовые параметры на входах **строка**, **смещение** и **по умолчанию** влияют на выходное число.

Строка	Смещение	По умолчанию	Смещение после числа	Число	Комментарии
–4,7e–3x	0	0	7	–0,0047	Символ **x** не допустим, поэтому преобразование заканчивается на нем
+5,3,2	0	0	4	5,3	Второй десятичный разделитель не допустим, поэтому преобразование заканчивается на нем

Scan Value

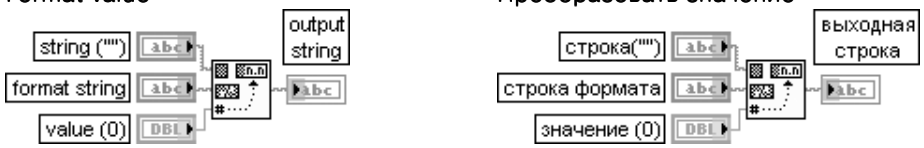
Просмотреть значение



Функция преобразует символы с начала **строки** (string) в тип данных, соответствующий кодам преобразования в **строке формата** (format string), и возвращает преобразованное число на выходе **значение** (value), а остаток строки после найденного числа передает на выход **выходная строка** (output string)

Format Value

Преобразовать значение



Функция преобразует число, подключенное ко входу **значение** (value), в упорядоченную строку в соответствии с форматом, определенным в **строке формата** (format string), и добавляет ее к входной **строке** (string), образуя **выходную строку** (output string). Данная функция аналогична функции **Преобразовать в строку** (Format Into String), но в отличие от нее имеет только один вход числового значения

Таблицы **функций взаимного преобразования строк, массивов байтов и путей** (String/Array/Path Conversion):

Path to Array of Strings

Путь в массив строк

Функция преобразует **путь** (path) в **массив строк** (array of string) и индицирует тип пути. Выход **относительный** (relative) отображает значение ИСТИНА, если путь является относительным, и ЛОЖЬ, если он является абсолютным.

Выход **массив строк** содержит компоненты пути. Первый элемент представляет первый шаг в иерархии пути (имя тома для файловой системы, поддерживающей несколько томов), а последний элемент – файл или каталог, определенные с помощью пути

Array of Strings to Path

Массив строк в путь

Функция преобразует **массив строк** (array of strings) в относительный или абсолютный **путь** (path). Если в массиве строк имеется пустая строка, то положение каталога перед пустой строкой удаляется при формировании выхода пути. Такое поведение аналогично перемещению на уровень вверх в иерархии каталога

Path to Strings

Путь в строку

Функция преобразует **путь** (path) в **строку** (string), описывающую путь в стандартном формате платформы.

Вход **путь** может быть путем, массивом путей, кластером путей или массивом кластеров путей, которые необходимо преобразовать в строку. Если на вход **путь** подано <Not A Path>, функция устанавливает в строке <Not A Path>

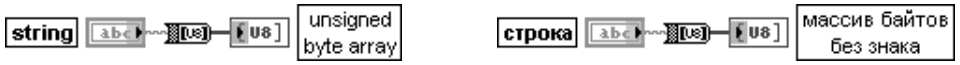
String To Path

Строку в путь

Функция преобразует **строку** (string), описывающую путь в стандартном формате платформы, в **путь** (path). Вход **строка** может быть строкой, кластером строк или массивом кластеров строк

String To Byte Array

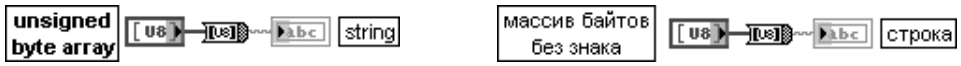
Строку в массив байтов



Функция преобразует строку в массив байтов без знака. Каждый байт в массиве является ASCII кодом соответствующего символа в строке

Byte Array To String

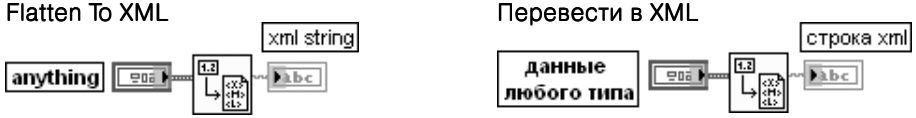
Массив байтов в строку



Функция преобразует массив байтов без знака, представляющих ASCII коды символов, в строку

Flatten To XML

Перевести в XML



Функция преобразует **данные любого типа** (anything) в строку XML в соответствии со схемой преобразования LabVIEW XML. Если **данные любого типа** содержат символы **<**, **>**, или &, то функция преобразует их в символы <, > или & соответственно. Для преобразования других символов, таких как «, необходимо использовать ВП **Специальные символы в XML** (Escape XML).
Вход **данные любого типа** принимает преобразуемые данные LabVIEW. Этот вход является полиморфным.
Выход **строка xml** (xml string) отображает результирующую строку XML, которая представляет данные LabVIEW. В случае преобразования десятичных значений эта функция использует только точку в качестве десятичного разделителя

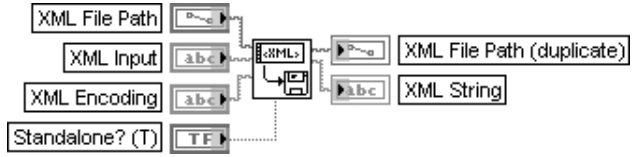
Unflatten From XML

Восстановить из XML

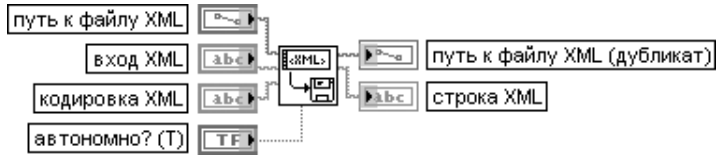
Функция преобразует строку XML в тип данных LabVIEW. Если вход **строка xml** (xml string) содержит символы <, > или &, то функция преобразует их в символы <, > или & соответственно. Для преобразования других символов, таких как ", необходимо использовать ВП **XML в специальные символы** (Unescape XML).
Вход **строка xml** (xml string) представляет строку XML с преобразуемыми данными. **Строка xml** должна соответствовать схеме **LabVIEW XML**, включающей варианты и порядок тегов в схеме. Если **строка xml** не соответствует схеме, функция возвращает ошибку и **значение** (value) содержит **значение по умолчанию** (default value) для типа данных, установленных на входе **тип данных**. Если ВП, содержащий функцию, перед этим уже обращался к ней, то выход **значение** будет содержать результат обращения. На входе **строка xml** эта функция принимает только точку (.) как десятичный разделитель. Функция не распознает локализованные десятичные разделители. Это ограничение предотвращает ошибки при использовании строк XML в операционных системах с различными установками десятичных разделителей.

Вход **тип** (type) определяет тип данных, в который должна преобразовываться **строка xml**.
Выход **значение** (value) представляет данные **строки xml**, имеющие тип, определенный на входе **тип**

Write to XML File

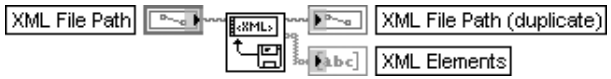


Записать в файл XML

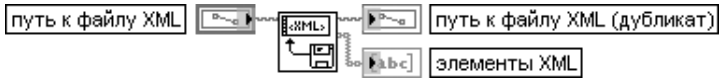


ВП записывает текстовую строку данных XML с сопутствующими тегами заголовка в текстовый файл. Используя данный полиморфный ВП, можно записать данные XML из строки или массива строк. Выбор зависит от типа данных, подаваемых на **вход XML** (XML Input).
Вход **путь к файлу XML** (XML File Path) содержит путь и имя файла, в который должны записываться данные XML. Имя файла должно иметь расширение **.xml**.
Вход XML (XML Input) содержит данные XML, записываемые в файл.
Вход **кодировка XML** (XML Encoding) определяет атрибуты кодировки файла XML.
Вход **автономно?** (Standalone?) определяет значение атрибута автономности в описании XML.
Параметр **автономно?** определяет, существует документ автономно (ИСТИНА) или зависит от других файлов (ЛОЖЬ).
Выход **путь к файлу XML (дубликат)** (XML File Path (duplicate)) отображает путь к файлу, в который ВП записывает данные. Этот выход может использоваться для определения пути, который выбран с помощью диалогового окна. Выход **путь к файлу XML (дубликат)** имеет значение **<Не путь>** (<Not A Path>) при отмене диалогового окна.
Выход **строка XML** (XML String) содержит данные XML, которые ВП записывает в определенный файл

Read From XML File



Читать из файла XML



ВП читает и выполняет грамматический разбор тегов файла XML LabVIEW. Этот полиморфный ВП используется для индексирования и разбора файла XML LabVIEW, путь к которому указан на входе **путь к файлу XML** (XML File Path), и возвращает массив строк или строку. Все данные XML должны следовать стандартной схеме XML LabVIEW.
Вход **путь к файлу XML** (XML File Path) содержит путь и имя файла, из которого должны читаться данные XML. Имя файла должно иметь расширение **.xml**.
Описание выхода **путь к файлу XML (дубликат)** (XML File Path(duplicate)) приведено выше при описании ВП **Запись в файл XML**.
Выход **элементы XML** (XML Elements) возвращает теги XML верхнего уровня, найденные между концами тегов </Version> и </LVData> в массиве строк (при вводе массива строк) или в простой строке (при вводе строки). Этот выход может быть подключен ко входу функции **Восстановить из XML** (Unflatten From XML)

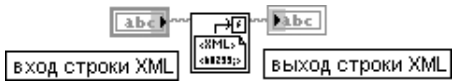
Escape XML

Специальные символы в XML

ВП преобразует специальные символы в синтаксис XML. Такие символы, как <, > или & преобразуются в <, > или & соответственно, функцией **Перевести в XML (**Flatten To XML). Данная функция преобразует в синтаксис XML другие символы, такие как ".
Вход строки XML (XML String In) передает строку XML, в которой должно быть произведено преобразование специальных символов.
Выход строки XML (XML String Out) отображает строку XML, содержащую преобразованные специальные символы

Unescape XML

XML в специальные символы



ВП преобразует синтаксис XML для специальных символов обратно в специальные символы. Функция **Восстановить из XML** (Unflatten From XML) преобразует символы <, > или & в <, > или & соответственно. Данный ВП должен использоваться для преобразования других символов, таких как &quot

В состав палитры строковых функций входит Экспресс-ВП **Сформировать текст (**Build Text):

Сформировать текст (Build Text)
Экспресс-ВП **Сформировать текст** (Build Text) (рис. 2.16) объединяет входные параметры в строку. Если вход не является строкой, то Экспресс-ВП преобразует вход в строку, опираясь на конфигурацию ВП (рис. 2.17)

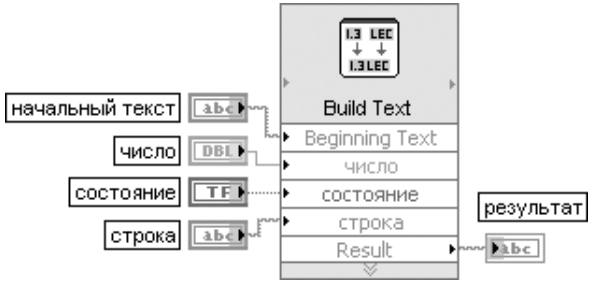


Рис. 2.16. Блок-диаграмма возможного подключения Экспресс-ВП

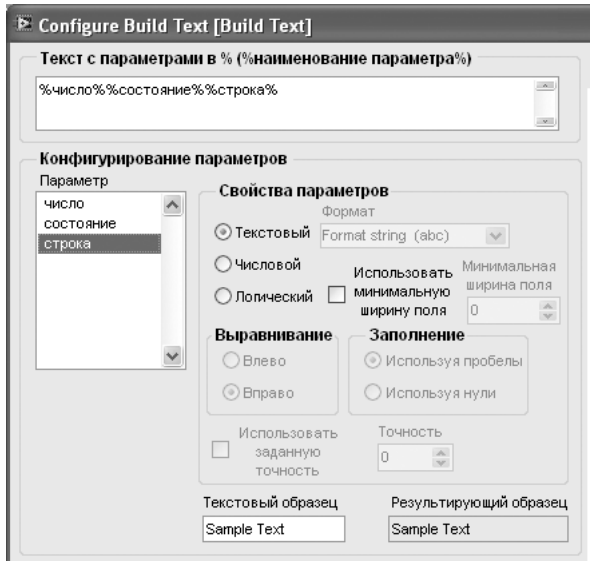


*Рис. 2.17. Вид диалогового окна конфигурирования Экспресс-ВП **Сформировать текст** (Build Text)*

2.1.4. Функции сравнения

Функции сравнения (рис. 2.18) позволяют проверять различные соотношения между скалярными и векторными переменными и константами (равно, не равно, больше, меньше, больше 0, меньше 0 и т. п.), проверять наличие информации, определенного числа или символа, а также выбирать одно из двух значений, выбирать максимальное и минимальное значения, а также проверять нахождение числа в заданном диапазоне.

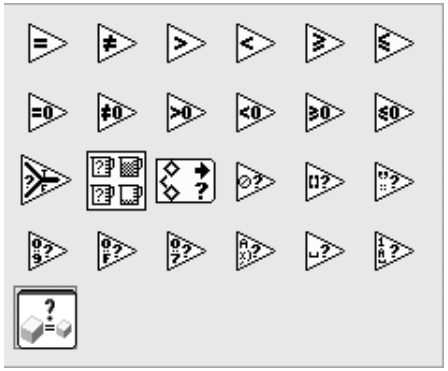
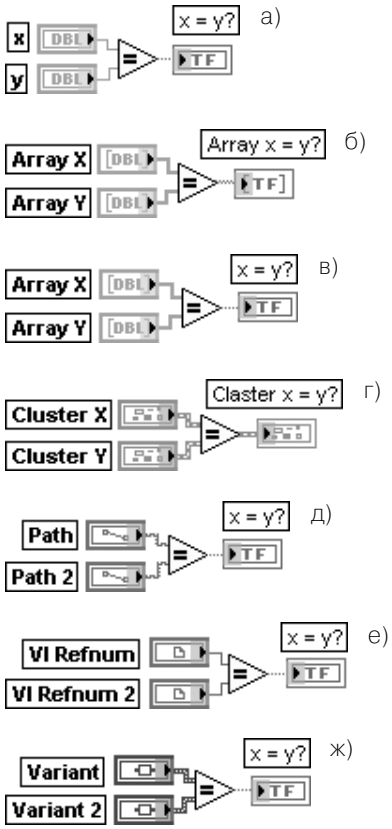


Рис. 2.18. Палитра функций сравнения

Функции сравнения двух аргументов, расположенные в верхней строке палитры, рассмотрены в следующей таблице:

Equal?



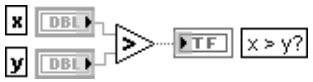
Равно?
Функция возвращает значение ИСТИНА, если **x** равно **y**, иначе возвращается значение ЛОЖЬ. Функция является полиморфной и позволяет сравнивать скаляры, массивы и кластеры констант и переменных числового, логического и строкового типа, пути, ссылки, данные типа Variant, отметки времени (time stamp), а также структуры на их основе, в частности таблицы, деревья и т. п. При сравнении массивов и кластеров предусмотрена возможность режимов **Сравнение элементов** (Compare Elements) (по умолчанию) и **Сравнение совокупности** (Compare Aggregates). Выбор режимов производится с помощью строки **Режим сравнения** (Comparision Mode) контекстного меню функции. При выборе режима **Сравнение элементов** выход **x=y?** представляет массив или кластер логических скаляров, при выборе режима **Сравнение совокупности** – логический скаляр.
На рис. 2.19 последовательно показаны варианты использования функции **Равно?** для сравнения числовых скаляров (рис. 2.19а), числовых массивов в режиме **Сравнение элементов** (рис. 2.19б) и **Сравнение совокупности** (рис. 2.19в), кластеров в режиме **Сравнение элементов** (рис. 2.19г), путей (рис. 2.19д), ссылок (рис. 2.19е), данных типа Вариант (рис. 2.19ж) и таблиц в режиме **Сравнение элементов** (рис. 2.19з)

*Рис. 2.19. Варианты подключения функции **Равно?***

Not Equal?

Не равно?
Функция возвращает значение ИСТИНА, если **x** не равно **y**, иначе возвращается значение ЛОЖЬ

Greater?



Больше?
Возвращает значение ИСТИНА, если **x** больше **y**, иначе возвращается значение ЛОЖЬ

Less?

Меньше?
Возвращает значение ИСТИНА, если **x** меньше **y**, иначе возвращается значение ЛОЖЬ

Greater Or Equal?

Больше или равно?
Возвращает значение ИСТИНА, если **x** больше или равно **y**, иначе возвращается значение ЛОЖЬ

Less Or Equal?

Меньше или равно?
Возвращает значение ИСТИНА, если **x** меньше или равно **y**, иначе возвращается значение ЛОЖЬ

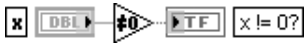
Все вышеприведенные функции имеют те же режимы работы, что и рассмотренная более подробно функция **Равно?**.

Следующие шесть функций производят сравнение входной числовой скалярной или векторной переменной **x** с нулем. Результат сравнения логического типа имеет ту же структуру, что и входная величина.

Equal To 0?

Равно 0?
Возвращает значение ИСТИНА, если **x** равно 0, иначе – ЛОЖЬ

Not Equal To 0?



Не равно 0?
Возвращает значение ИСТИНА, если **x** не равно 0, иначе – ЛОЖЬ

Greater Than 0?

Больше 0?
Возвращает значение ИСТИНА, если **x** больше 0, иначе – ЛОЖЬ

Less Than 0?

Меньше 0?
Возвращает значение ИСТИНА, если **x** меньше 0, иначе – ЛОЖЬ

Greater Or Equal To 0?

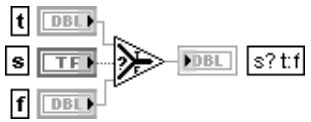
Больше или равно 0?
Возвращает значение ИСТИНА, если **x** больше или равно 0, иначе – ЛОЖЬ

Less Or Equal To 0?

Меньше или равно 0?
Возвращает значение ИСТИНА, если **x** меньше или равно 0, иначе – ЛОЖЬ

Следующие три функции позволяют выбрать одну из двух величин, получить значения большей и меньшей величины и определить нахождение величины в заданном диапазоне.

Select



Выбрать
Функция возвращает значение, подключенное ко входам **t** или **f** в зависимости от состояния входа **s**. Если на входе **s** установлено состояние ИСТИНА, то функция возвращает значение, подключенное ко входу **t**. Если же на входе **s** установлено состояние ЛОЖЬ, то функция возвращает значение, подключенное ко входу **f**. Функция является полиморфной

Max & Min

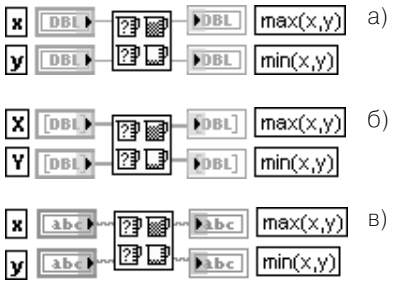
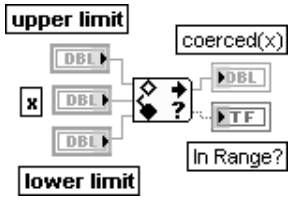


Рис. 2.20. Варианты подключения функции ***Максимум и минимум***

Максимум и минимум
Функция сравнивает **x** и **y** и возвращает большее значение на верхнем выходе, а меньшее значение – на нижнем. Данная функция воспринимает значения **отметок времени** (time stamp), если эти значения поданы на оба входа. В этом случае функция возвращает на верхнем выходе более позднее время, на нижнем – более раннее. Функция является полиморфной.
В качестве примеров на рис. 2.20 приведены варианты использования функции для сравнения скалярных числовых значений (рис. 2.20а), числовых массивов (рис. 2.20б) и строк (рис. 2.20в).
При сравнении массивов на выходах функции также формируются два массива, содержащих соответственно максимальные и минимальные значения, полученные в результате поэлементного сравнения исходных массивов. Сравнение строк производится на основе сравнения величин ASCII кодов символов строк в порядке слева направо

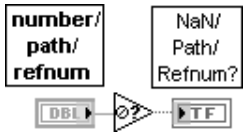
In Range and Coerce



Нахождение в диапазоне и ограничение
Функция определяет нахождение **x** в диапазоне, заданном входами **верхний предел** (upper limit) и **нижний предел** (lower limit), и дополнительно ограничивает выходное значение указанным диапазоном. Нахождение в диапазоне приводит к появлению на выходе **в диапазоне?** (In Range?) значения ИСТИНА. Функция выполняет ограничение только при установке в режим **Сравнение элементов** (Compare Elements) (по умолчанию). Данная функция воспринимает значения отметок времени, если эти значения поданы на все входы

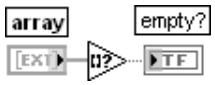
Следующие девять функций позволяют определить тип входной величины или ее класс.

Not A Number/Path/Refnum?



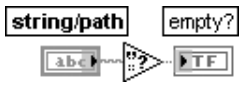
Не число/Путь/Ссылка?
Функция возвращает значение ИСТИНА, если подключенное ко входу **число/путь/ссылка** (number/path/refnum) значение не является **числом** (not a number) (NaN), **путем** (path) или **ссылкой** (refnum). В противном случае данная функция возвращает ЛОЖЬ. Функцию целесообразно использовать для того, чтобы убедиться, что ссылка на такие объекты, как ВП, приложение или элемент управления, еще находится в системной памяти и не закрыта. На вход функции могут подаваться числовые значения, пути, ссылки, массивы или кластеры таких значений

Empty Array?



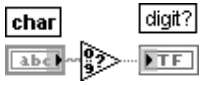
Пустой массив?
Функция возвращает значение ИСТИНА, если на вход **массив** (array) подается **пустой массив** (empty array). В противном случае функция формирует значение ЛОЖЬ. Функция является полиморфной

Empty String/Path?



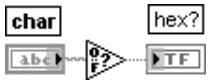
Пустая строка/Путь?
Функция возвращает значение ИСТИНА, если на вход **строка/путь** (string/path) подается **пустая строка** (empty string) или **пустой путь** (empty path). В противном случае функция формирует значение ЛОЖЬ. Функция является полиморфной

Decimal Digit?



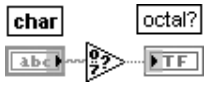
Десятичная цифра?
Функция возвращает значение ИСТИНА, если входной **символ** (char) представляет десятичную цифру, находящуюся в диапазоне от 0 до 9. Если **символ** является строкой, то функция использует первый символ строки. Если **символ** является числом, то данная функция интерпретирует его как значение ASCII кода символа. Во всех других случаях функция возвращает значение ЛОЖЬ. Функция является полиморфной

Hex Digit?



Шестнадцатеричная цифра?
Функция возвращает значение ИСТИНА, если входной **символ** (char) представляет шестнадцатеричную цифру, находящуюся в диапазоне от 0 до 9 и от A до F. Обработка данных на входе **символ** описана выше

Octal Digit?



Восьмеричная цифра?
Функция возвращает значение ИСТИНА, если входной **символ** (char) представляет восьмеричную цифру, находящуюся в диапазоне от 0 до 7. Обработка данных на входе **символ** описана выше

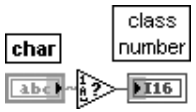
Printable?

Печатный символ?
Функция возвращает ИСТИНА, если входной **символ** (char) представляет печатный ASCII символ. Обработка данных на входе **символ** описана выше

White Space?

Непечатный символ?
Функция возвращает ИСТИНА, если входной **символ** (char) представляет такие непечатные ASCII символы, как **пробел** (Space), **горизонтальная** (Tab) или **вертикальная** (Vertical Tab) **табуляция**, **новая строка** (**перевод строки**) (Newline), **возврат каретки** (Carriage Return), новая страница (Form Feed). Обработка данных на входе **символ** описана выше

Lexical Class



Лексический класс
Функция возвращает **номер класса** (class number) входного **символа** (char). Обработка данных на входе **символ** описана выше.
Типы классов и соответствующие им номера классов приведены в таблице.

Номер класса	Лексический класс
0	Символы расширения (коды от 128 до 255)
1	Неотображаемые ASCII символы (коды от 0 до 31 за исключением 9 и 13)
2	Непечатные символы: пробел, табуляция, возврат каретки, новая страница, перевод строки, вертикальная табуляция (десятичные коды 32, 9, 13, 12, 10 и 11 соответственно)
3	Цифры от 0 до 9
4	Символы букв верхнего регистра от A до Z
5	Символы букв нижнего регистра от a до z
6	Все печатные не арифметические ASCII символы

В состав палитры функций сравнения входит Экспресс-ВП **Сравнение** (Comparison).

Сравнение (Comparison)

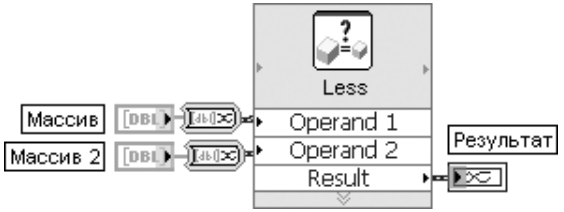


Рис. 2.21. Блок-диаграмма возможного подключения Экспресс-ВП

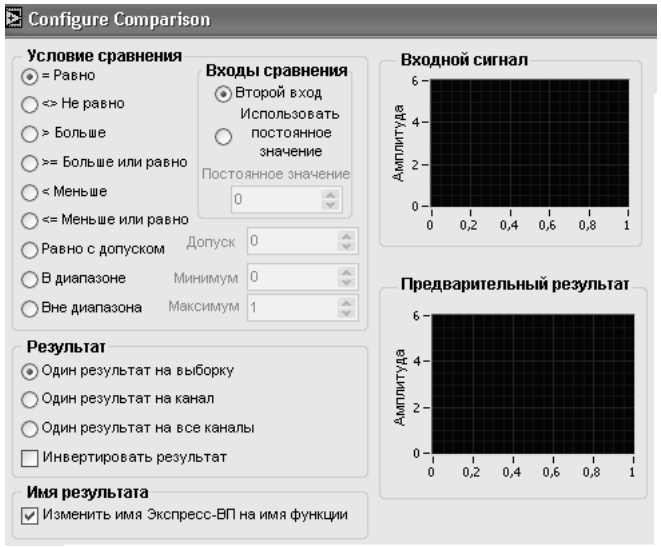


*Рис. 2.22. Вид диалогового окна конфигурирования Экспресс-ВП **Сравнение***

Экспресс-ВП **Сравнение** (Comparison) (рис. 2.21) сравнивает заданные входы с целью определения их равенства или неравенства. Назначение опций диалогового окна конфигурирования очевидно из их перевода (рис. 2.22)

2.1.5. Функции работы с массивами

Функции работы с массивами (рис. 2.23) позволяют инициализировать массивы, определить их размеры, выделить элементы или подмассивы из массивов, удалить элементы или подмассивы, изменить размер или размерность массивов. Большинство функций являются полиморфными и могут работать с массивами

данных числового, логического или строкового типа. При этом они автоматически перестраиваются при подключении массива с определенной размерностью. Часть функций для увеличения числа входов может быть растянута в вертикальном направлении с помощью инструмента перемещения.

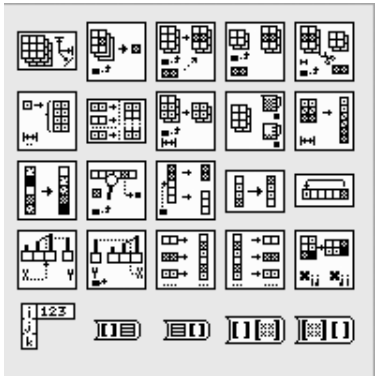


Рис. 2.23. Палитра функций работы с массивами

Ниже в таблице описаны функции работы с массивами.

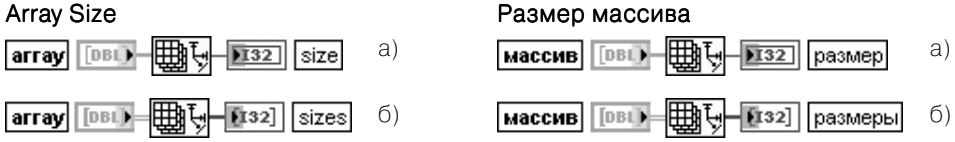
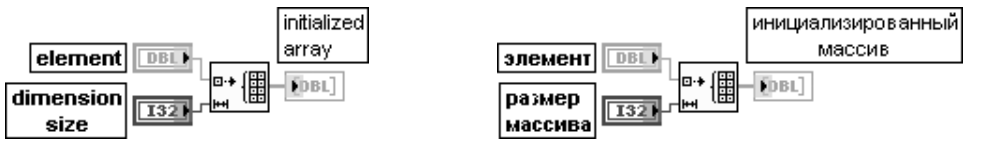


*Рис. 2.24. Варианты подключения функции **Размер массива***

Функция возвращает число элементов массива каждой размерности. Функция является полиморфной, то есть она может определять размеры массива произвольной размерности. На рис. 2.24 показаны варианты определения размеров одномерного (а) и двумерного (б) массивов.
Вход **массив** (array) может быть n-мерным массивом любого типа.
Выход **размер(ы)** (size(s)) является 32-битовым целым, если массив является одномерным. Если массив является многомерным, то возвращаемое значение является одномерным массивом, в котором каждый элемент является 32-битовым целым, представляющим число элементов по соответствующей размерности массива

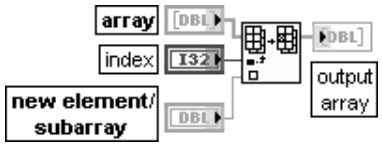


Функция создает n-мерный массив, в котором каждому элементу присваивается значение, заданное на входе **элемент** (element). Размерность выходного массива может быть увеличена растяжением иконки функции по вертикали с помощью инструмента перемещения.
На вход **элемент** может быть подана величина любого скалярного типа.
Вход **размер массива** (dimension size 0..n-1) должен быть числовым. Функция создает пустой массив, если на один из входов **размер массива** подан 0. Для инициализации **n**-мерного массива необходимо иметь **n** терминалов **размер массива**.
Выход **инициализированный массив** (initialized array) является массивом того же типа, что и величина на входе **элемент**

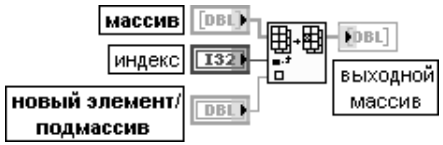
*Рис. 2.25. Варианты подключения функции **Индексировать массив***

Функция возвращает **элемент** (element) или **подмассив** (sub-array) входного **массива** (array), начиная с **индекса** (index). При подключении входного массива функция автоматически перестраивается в соответствии с его размерностью, отображая входы **индекс** для каждой размерности (рис. 2.25б). Число терминалов элементов или подмассивов можно увеличить с помощью инструмента перемещения.
Входной массив может быть **n**-мерным массивом любого типа. Если входной массив является пустым, то на выходах **элемент** или **подмассив** возвращается значение по умолчанию для данного типа элементов массива.
Вход **индекс** должен быть числовым. Число входов **индекс** должно соответствовать размерности входного массива. Если значение **индекс** находится вне диапазона индексов (<0 или >N, где N – размер входного массива), то на выходе **элемент** или **подмассив** возвращается значение по умолчанию.
Выход **элемент** или **подмассив** имеет тот же тип, что и элементы входного массива. Наряду с извлечением элемента массива функция позволяет извлекать подмассив из массива, если один из входов индексирования был оставлен неподключенным. Так, например, оставляя неподключенным вход индексирования по столбцам (рис. 2.25б), можно выделить заданную строку двумерного массива, а оставляя неподключенным вход индексирования по строкам, можно выделить заданный столбец. Неподключенный вход отображается в функции как полый прямоугольник. Задание индекса строки и столбца позволяет выделить элемент двумерного массива. По умолчанию если ни один из входов индексирования функции не подключен, то на ее выход передается содержимое первой строки. Если размер иконки функции по вертикали будет увеличен, то на втором выходе будет отображаться вторая строка и т. д.

Replace Array Subset

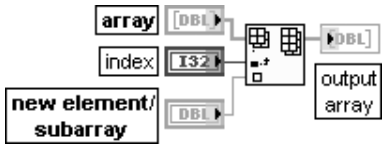


Заменить подмассив

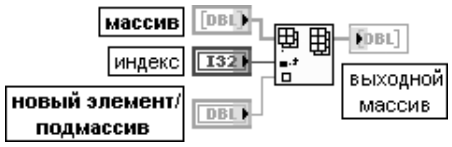


Функция заменяет элемент или подмассив входного **массива**, начиная с точки, определенной на входе **индекс** (index). Функция также перестраивается в соответствии с размерностью подключаемого массива. Если входы индексирования по определенной размерности не подключены, то функция заменяет все элементы по этой размерности.
Вход **массив** (array) представляет массив, в котором заменяются элементы, строки, столбцы или страницы. Этот вход может быть **n**-мерным массивом любого типа.
Вход **индекс** определяет элемент, строку, столбец или страницу массива, которые должны быть заменены.
Вход **новый элемент/подмассив** (new element/subarray) представляет элемент или массив, которые заменяют элемент, строку, столбец или страницу массива, подаваемого на вход **массив**.
Выход **выходной массив** (output array) представляет возвращаемый функцией массив с замененными элементом, строкой, столбцом или страницей.
Базовый тип нового элемента или подмассива должен быть такого же типа, что и входной массив.
Для одновременной замены нескольких элементов или подмассивов иконка функции может быть растянута по вертикали с помощью инструмента перемещения

Insert Into Array

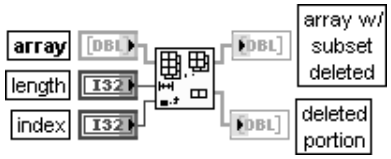


Вставить в массив

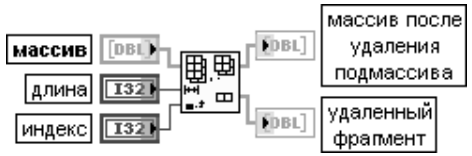


Функция вставляет элемент или подмассив во входной массив, начиная с точки, определенной на входе **индекс** (index). Функция также перестраивается в соответствии с размерностью подключаемого массива. Если входы индексирования по определенной размерности не подключены, то функция добавляет новый элемент или подмассив к концу входного массива.
Вход **массив** (array) представляет массив, в который вставляется элемент, строка, столбец или страница. Этот вход может быть **n**-мерным массивом любого типа.
Вход **индекс** определяет точку массива, начиная с которой вставляется элемент, строка, столбец или страница. Может подключаться только один индексный вход.
Вход **новый элемент/подмассив** (new element/subarray) передает массив или элемент, которые вставляются во входной массив.
Выход **выходной массив** (output array) представляет возвращаемый функцией массив с вставленными элементом, строкой, столбцом или страницей.
Базовый тип нового элемента или подмассива должен быть такого же типа, что и входной массив

Delete From Array

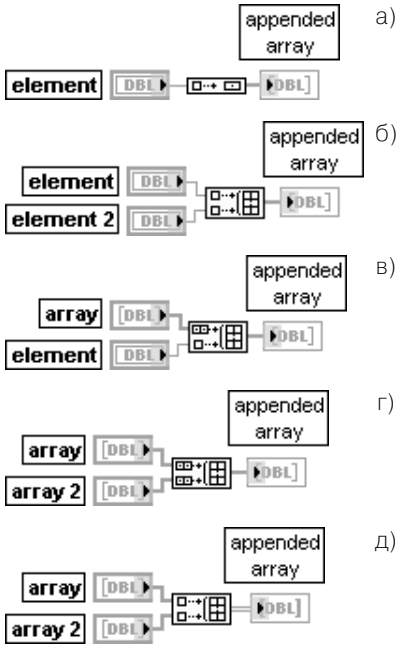


Удалить из массива



Функция удаляет элемент или подмассив из **массива** (array) и возвращает уменьшенный массив на выходе **массив после удаления подмассива** (array w/ subset deleted) и удаленный элемент или подмассив на выходе **удаленный фрагмент** (deleted portion). Функция изменяет число индексов в соответствии с размерностью подключаемого к ее входу массива.
Вход **длина** (length) определяет количество удаляемых элементов, строк, столбцов или страниц. Если вход **длина** подключен, то на выходе **удаленный фрагмент** массив имеет ту же размерность, что и входной **массив**. Если вход не подключен, то размерность на выходе **удаленный фрагмент** на единицу меньше размерности входного массива.
Вход **индекс** (index) определяет удаление элемента, строки, столбца или страницы. Может быть подключен только один вход индексирования

Build Array



Сформировать массив

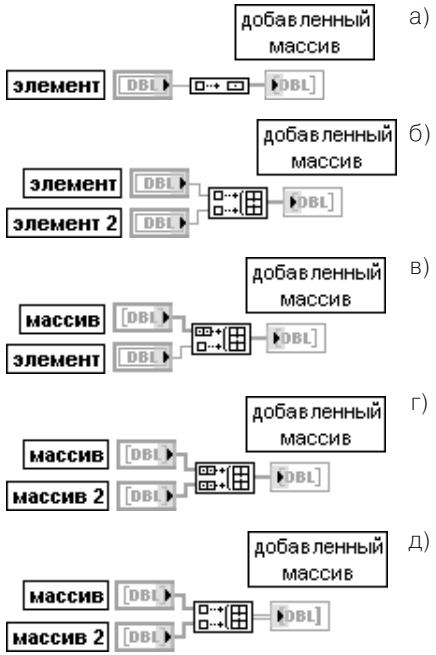


Рис. 2.26. Варианты подключения функции ***Сформировать массив***

Функция объединяет набор массивов или добавляет элементы к **n**-мерному массиву. Для модификации существующего массива можно использовать и функцию **Заменить подмассив**.

Входы **массив** (array) или **элемент** (element) могут быть **n**-мерным массивом или скалярным элементом. Все входы должны быть элементами и одномерными массивами или **n**-мерными и (**n**–1)-мерными массивами. Все входы должны иметь один и тот же базовый тип.
Выход **добавленный массив** (appended array) отображает результирующий массив. При помещении функции на блок-диаграмму она имеет только один доступный вход (рис. 2.26а). Использование ее в таком виде позволяет преобразовать скалярную величину в одномерный массив, содержащий один элемент.
Количество входов можно увеличить с помощью вызова строки **добавить вход** (Add Input) контекстного меню или увеличивая размер функции в вертикальном направлении с помощью инструмента перемещения (рис. 2.26б–д).
Функция **Сформировать массив** выполняется в двух режимах в зависимости от выбора опции **объединить входы** (Concatenate Inputs). Если эта опция установлена в контекстном меню функции, то функция добавляет все входы друг за другом, формируя выходной массив с размерностью, которая равна наибольшей размерности входного массива (рис. 2.26г). Если же эта опция не выбрана, то функция формирует выходной массив с размерностью, на единицу большей размерности входного массива (рис. 2.26д). При этом входы должны иметь одинаковую размерность. Функция добавляет каждый вход в порядке подключения, формируя подмассив, элемент, строку или страницу выходного массива. Входы дополняются, если это необходимо, до размера наиболее длинного массива.
Если входы являются массивами одинаковой размерности, то контекстное меню функции позволяет устанавливать или снимать опцию **объединить входы** (Concatenate Inputs). Если входы имеют различную размерность (рис. 2.26в), то опция **объединить входы** выбирается автоматически и не может быть снята. Если все входы являются скалярными элементами (рис. 2.26б), то опция **объединить входы** автоматически снимается и не может быть выбрана, а выход является одномерной функцией, содержащей элементы, которые следуют в порядке подключения.
При выборе опции **объединить входы** маленькие квадраты в иконке **Сформировать массив** изменяются с целью отражения различий двух типов входов. Входы, имеющие такую же размерность, что и выходы, обозначаются двумя квадратами, в то время как входы, имеющие меньшую по сравнению с выходом размерность, обозначаются одним квадратом

Array Subset

Подмассив

Функция возвращает часть **массива** (array), начинающуюся с **индекса** (index) и содержащую число элементов, заданное на входе **длина** (length). При подключении массива функция автоматически перестраивается в соответствии с его размерностью.
Вход **массив** может быть **n**-мерным массивом произвольного типа.
Вход **индекс** должен быть числовым. Если **индекс** меньше 0, то функция воспринимает его как 0. Если **индекс** больше или равен размеру массива, то функция возвращает пустой массив.
Вход **длина** должен быть числовым. Если **индекс** плюс **длина** превышает размер массива, то функция возвращает столько данных, сколько возможно. По умолчанию это значение равно длине от индекса до конца массива.
Выход **подмассив** (subarray) имеет тот же тип, что и **массив**

Rotate 1D Array

Циклически сместить одномерный массив

Функция циклически смещает элементы **массива** (array) на число позиций и в направлении, определяемом значением на входе **n**.
Вход **n** должен иметь числовой тип данных. Функция преобразует **n** к 32-битовому целому числу при подключении числа с другим представлением.
Вход **массив** может быть одномерным массивом произвольного типа.
Выход **массив** (**последние n элементов первыми**) (array (last n elements first)) представляет выходной массив.
Например, если **n** равно 1, то нулевой элемент **массив**[0] станет первым элементом выходного массива **массив** (**последние n элементов первыми**)[1], первый элемент **массив**[1] станет вторым элементом **массив** (**последние n элементов первыми**)[2] и т. д., и элемент **массив**[m–1] станет нулевым элементом **массив** (**последние n элементов первыми**)[0], где **m** – число элементов массива. Если **n** равно –2, нулевой элемент массива **массив**[0] станет элементом **массив** (**последние n элементов первыми**)[m–2], первый элемент **массив**[1] станет элементом **массив** (**последние n элементов первыми**)[m–1] и т. д., и входной элемент массива **массив**[m–1] станет элементом **массив** (**последние n элементов первыми**)[m–3]

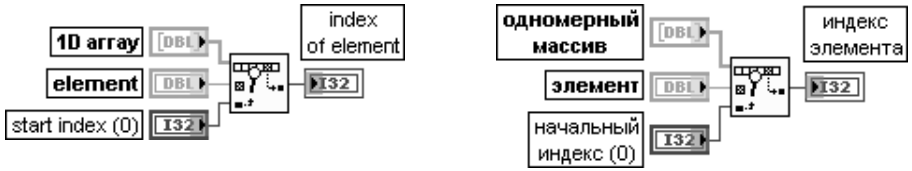
Reverse 1D Array

Обратить элементы одномерного массива

Функция обращает порядок элементов одномерного массива произвольного типа

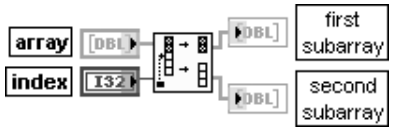
Search 1D Array

Искать в одномерном массиве

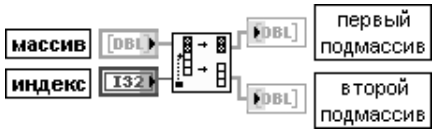


Функция осуществляет поиск **элемента** (element) в **одномерном массиве** (1D array) с **начального индекса** (start index). Так как поиск является линейным, то нет необходимости в предварительной сортировке массива.
Вход **одномерный массив** может быть одномерным массивом любого типа.
Вход **элемент** представляет значение, которое ищется во входном массиве. Представление элемента должно соответствовать представлению одномерного массива.
Вход **начальный индекс** должен быть числовым. По умолчанию его значение равно 0.
Выход **индекс элемента** (index of element) представляет индекс найденного элемента. Если функция не находит элемент, то **индекс элемента** имеет значение –1

Split 1D Array



Разделить одномерный массив



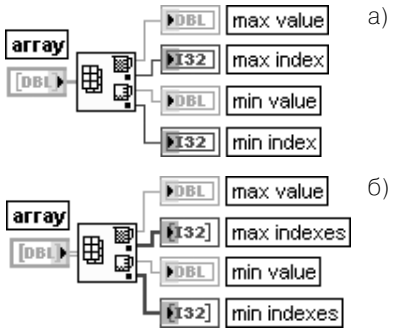
Функция делит **массив** (array) по **индексу** (index) и возвращает два подмассива.
Вход **массив** может быть одномерным массивом любого типа.
Вход **индекс** должен быть числовым. Если **индекс** меньше или равен 0, то **первый подмассив** (first subarray) является пустым. Если **индекс** больше или равен размеру **массива**, то **второй подмассив** (second subarray) является пустым.
Выход **первый подмассив** (first subarray) содержит элементы массива от **массив**[0] до **массив**[**индекс**-1].
Выход **второй подмассив** содержит оставшиеся элементы **массива**, не содержащиеся в первом подмассиве

Sort 1D Array

Сортировать одномерный массив

Функция возвращает отсортированную версию входного **массива** (array) с элементами, расположенными в порядке возрастания. Если **массив** является массивом кластеров, то функция сортирует элементы, сравнивая первые элементы кластеров. Если первые элементы совпадают, то функция сравнивает вторые и последующие элементы

Array Max & Min



Максимум и минимум массива

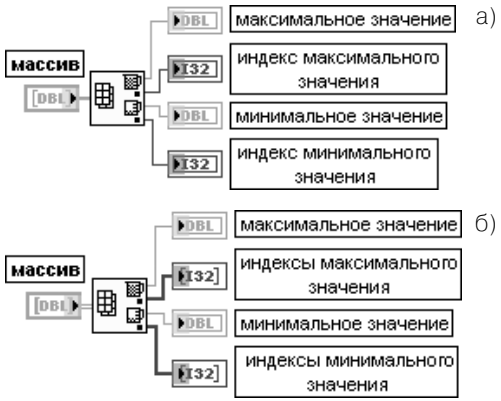


Рис. 2.27. Варианты подключения функции ***Максимум и минимум массива***

Функция возвращает максимальное и минимальное значения, найденные в **массиве** (array), вместе с индексами каждого значения.
Вход **массив** может быть **n**-мерным массивом произвольного типа.
Выходы **максимальное значение** (max value) и **минимальное значение** (min value) имеют тот же тип и структуру данных, что и элементы входного массива.

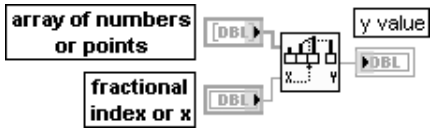
Выход **максимальный индекс** (max index) отображает индекс первого максимального значения. Если **массив** является многомерным, то выход **максимальные индексы** (max indexes) представляет массив, элементами которого являются индексы первого максимального значения массива.
Выход **минимальный индекс** отображает индекс первого минимального значения. Если **массив** является многомерным, то выход **минимальные индексы** представляет массив, элементами которого являются индексы первого минимального значения массива.
Если числовой **массив** является одномерным, то выходы **максимальный индекс** и **минимальный индекс** являются скалярными целыми (рис. 2.27а). Если числовой **массив** является многомерным, то эти выходы представляют одномерный массив, который содержит индексы максимального и минимального значений (рис. 2.27б)

Transpose 2D Array

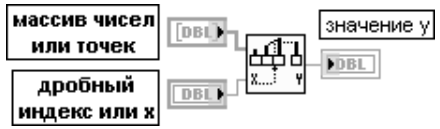
Транспонировать двумерный массив

Функция переставляет элементы в **двумерном массиве** (2D array) так, что **двумерный массив**[i,j] становится **транспонированным массивом**[j,i]

Interpolate 1D Array



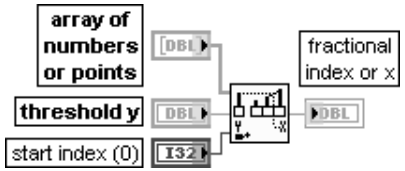
Интерполировать одномерный массив



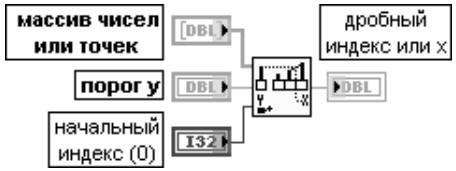
Функция получает с помощью линейной интерполяции десятичное **значение y** (y value) из **массива чисел или точек** (array of numbers or points), используя значение **дробного индекса или x** (fractional index or x). На вход данной функции может быть подключен массив числовых значений или массив точек данных. При подключении массива числовых значений функция интерпретирует **дробный индекс или x** как ссылку к элементам массива. При подключении массива наборов точек данных функция интерпретирует **дробный индекс или x** как ссылку к элементам значений **x** в каждом наборе точек данных. В последнем случае точки данных должны быть отсортированы по возрастанию значений **x**.
Вход **массив чисел или точек** может быть массивом чисел или массивом наборов точек данных, в котором каждый набор является кластером координат **x** и **y**. Если на этот вход подан массив наборов точек, то функция использует первый элемент в кластере (**x**) для получения дробного индекса с помощью линейной интерполяции. После этого функция использует этот дробный индекс для расчета **выходного значения** (y value) из второго элемента кластера (**y**).
Вход **дробный индекс или x** является индексом или значением **x**, для которого функция должна вернуть значение **y**. Например, если вход **массив чисел или точек** содержит два числа двойной точности с плавающей запятой 5 и 7, а на входе **дробный индекс или x** установлено значение 0,5, то функция возвращает значение 6,0, которое расположено посередине между значениями элементов 0 и 1.

Если **массив чисел или точек** содержит массив наборов точек данных, функция возвращает **значение y** (y value), полученное с помощью линейной интерполяции при значении **x**, соответствующем величине **дробный индекс или x**. Например, если массив содержит две точки (3, 7) и (5, 9) и на входе **дробный индекс или x** установлено значение 3,5, то функция вернет значение 7,5.
Вход **дробный индекс или x** не интерполирует за границами массива или набора точек данных. Для корректной работы функции значение **дробный индекс или x** должно быть расположено непосредственно в точке или между двумя точками.
Выход **значение y** отображает интерполированное значение элемента при значении дробного индекса или интерполированное значение **y** при дробном индексе набора точек данных массива **массив чисел или точек**

Threshold 1D Array



Порог одномерного массива

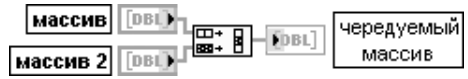


Функция сравнивает **порог y** (threshold y) и значения **массива чисел или точек** (array of numbers or points) с **начального индекса** (start index) до нахождения пары таких соседних элементов, что **порог y** будет больше значения первого элемента и меньше или равен значению второго элемента.
Вход **массив чисел или точек** идентичен одноименному входу рассмотренной выше функции **Интерполировать одномерный массив** (Interpolate 1D Array).
Вход **порог y** является пороговым значением функции. Если **порог y** меньше или равен значению массива с индексом **начальный индекс** (start index), то функция возвращает **начальный индекс** на выходе **дробный индекс или x** (fractional index or x). Если **порог y** больше каждого значения массива, то функция возвращает индекс последнего значения. Если массив является пустым, функция возвращает значение NaN.
Вход **начальный индекс** должен быть числом. По умолчанию его значение равно 0, что означает, что функция возвращает результат, рассчитанный для всего массива, а не для его определенной части.
Выход **дробный индекс или x** представляет результат линейной интерполяции, рассчитанный LabVIEW для одномерного массива **массив чисел или точек**.
Так, например, если вход **массив чисел или точек** является массивом из четырех чисел [4, 5, 5, 6], **начальный индекс** равен 0 и **порог y** равен 5, тогда **дробный индекс или x** равен 1, соответствующей индексу первого найденного функцией значения 5. Если массив содержит элементы 2,3, 5,2, 7,8, 7,9 и 10,0, **начальный индекс** равен 0 и **порог y** равен 6,5, то тогда выходное значение будет равно 1,5, поскольку 6,5 находится посередине между 5,2 (индекс 1) and 7,8 (индекс 2). Если **порог y** равен 7 для того же набора данных, то выходное значение будет равно 1,69.
Если входной массив содержит массив точек, в котором каждая точка является кластером координат **x** и **y**, выход представляет интерполированное значение **x**, соответствующее интерполированному положению порога **порог y**, а не дробному индексу массива. Если интерполированное положение **порог y** находится посередине между индексами 4 и 5 массива, имеющего значения по **x** –2,5 и 0 соответственно, то на выходе будет не значение индекса 4,5, как это было для числового массива, а значение **x**, равное –1,25.

Эта функция не определяет индекса пересечения порога при отрицательном наклоне и возвращает некорректное значение, если **порог y** меньше значения элемента, имеющего **начальный индекс**. Ее необходимо использовать для массивов с неубывающими по порядку элементами. Для лучшего анализа массивов целесообразно использовать ВП **Пороговый детектор пиков** (Threshold Peak Detector)

Interleave 1D Arrays

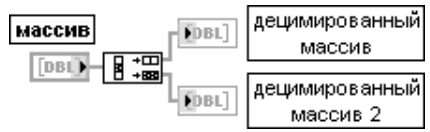
Чередовать одномерные массивы



Функция поочередно размещает элементы с равными индексами из входных массивов в один выходной массив.
Входы **массив 0..n-1** (array 0..n-1) должны быть одномерными массивами. Если входные массивы имеют разный размер, то число элементов в **чередуемом массиве** (interleaved array) равно произведению числа элементов в самом коротком массиве на число массивов.
Выход **чередуемый массив**[0] содержит **массив 0**[0], **чередуемый массив**[1] содержит **массив 1**[0], **чередуемый массив**[n-1] содержит **массив n-1**[0], **чередуемый массив**[n] содержит **массив 0**[1] и т. д., где **n** – число входных терминалов

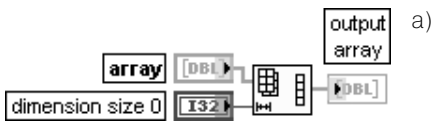
Decimate 1D Array

Дециммировать одномерный массив

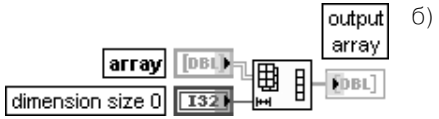


Функция разделяет элементы входного **массива** (array) на ряд выходных массивов, размещая элементы на выходах последовательно. Число выходных терминалов может быть увеличено путем растяжения иконки функции в вертикальном направлении с помощью инструмента перемещения.
Вход **массив** может быть одномерным массивом любого типа, за исключением логического.
Выход **дециммированный массив** (decimated array) представляет первый выходной массив и содержит элементы 0, n, 2n,.. Функция размещает элемент **массив**[0] по индексу 0 первого выходного массива, элемент **массив**[1] – по индексу 0 второго выходного массива, элемент **массив**[n-1] – по индексу 0 последнего выходного массива, элемент **массив**[n] – по индексу 1 первого выходного массива и т. д., где **n** – число выходных терминалов этой функции.
Например, если **массив** имеет 16 элементов и к функции подключено четыре выходных массива, то первый выходной массив получит элементы с индексами 0, 4, 8 и 12. Второй выходной массив получит элементы с индексами 1, 5, 9 и 13, третий массив – элементы 2, 6, 10 и 14, и последний массив – элементы 3, 7, 11 и 15.
Выход **дециммированный массив 2** представляет второй выходной массив и содержит элементы 1, n+1, 2n+1 ... и т. д.

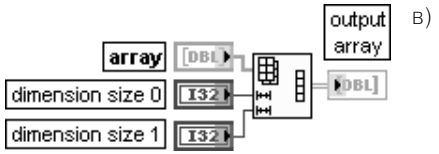
Reshape Array



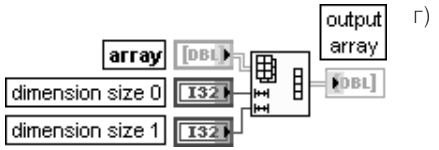
а)



б)

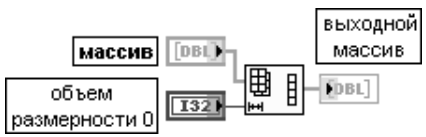


в)

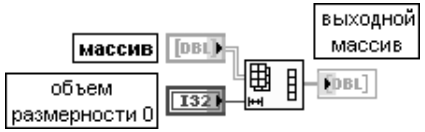


г)

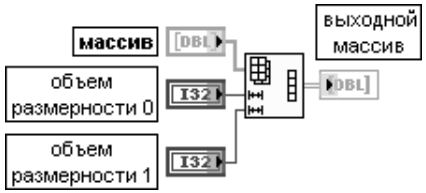
Изменить форму массива



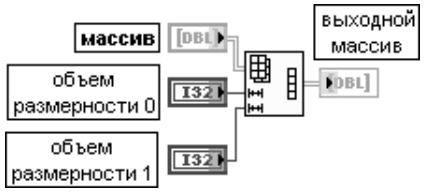
а)



б)



в)



г)

Рис. 2.28. Варианты подключения функции ***Изменить форму массива***

Функция изменяет размер или размерность массива в соответствии со значениями входов **объем размерности 0..m-1** (dimension size 0..m-1).
Вход **массив** может быть **n**-мерным массивом любого типа.
Входы **объем размерности 0..m-1** должны быть числовыми. Функция формирует пустой массив, если объем размерности равен 0. Для формирования **m**-мерного выходного массива функция должна иметь **m** входов **объем размерности**.
Если произведение всех значений объема размерности превышает число элементов входного массива, то функция дополняет новый массив значениями по умолчанию, имеющими тип данных элементов входного массива. Если это произведение меньше числа элементов входного массива, функция усекает массив.
Данная функция не перемещает данные в памяти, а только изменяет их восприятие в соответствии с параметрами, задающими объем каждой размерности. Так, например, если на вход функции передается одномерный массив с 9 элементами {0,1,2,3,4,5,6,7,8}, а на входах объема размерности заданы значения 2 и 3, то функция возвращает двумерный массив, содержащий {{0,1,2},{3,4,5}}. Функция усекает последние три входных элемента, поскольку выходной массив может разместить только шесть элементов.
Число входов **объем размерности** может быть увеличено путем растяжения иконки функции в вертикальном направлении с помощью инструмента перемещения.
На рис. 2.28 показаны варианты применения функции для изменения размера одномерного массива (рис. 2.28а), преобразования одномерного массива в двумерный (рис. 2.28б), изменения размера двумерного массива (рис. 2.28в) и преобразования двумерного массива в одномерный (рис. 2.28г)

Array To Cluster

Массив в кластер

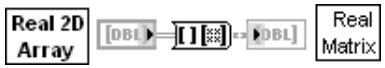
Функция преобразует одномерный массив в кластер элементов того же типа, что и элементы массива. Число элементов кластера устанавливается с помощью опции **Размер кластера** (Cluster Size) контекстного меню функции. По умолчанию число элементов равно девяти. Максимальный размер кластера для этой функции равен 256

Cluster To Array

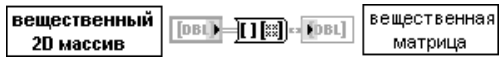
Кластер в массив

Функция преобразует кластер элементов одного типа в одномерный массив данных того же типа. Компоненты кластера не могут быть массивами. Порядок элементов в массиве такой же, как и в кластере

Array To Matrix



Массив в матрицу



Полиморфная функция преобразует массив (одномерный или двумерный, реальный или комплексный), подключенный ко входу **Вещественный 2D массив** (Real 2D Array), в матрицу элементов того же типа. Выбор варианта реализации осуществляется автоматически или с помощью строки **Выбрать тип** (Select Type) контекстного меню функции

Matrix To Array

Матрицу в массив

Полиморфная функция преобразует матрицу элементов (реальную или комплексную), подключенных ко входу **Вещественная матрица** (Real Matrix), в массив того же типа. Выбор варианта реализации осуществляется автоматически или с помощью строки **Выбрать тип** (Select Type) контекстного меню функции
В качестве примера эффективного использования функций работы с массивами на рис. 2.29 приведена блок-диаграмма ВП **Уравнение нагрева** (Heat Equation) из набора примеров NI Example Finder LabVIEW. ВП рассчитывает распределение температуры прямоугольной пластины при заданных значениях начальных температур ее сторон.

2.1.6. Функции работы с кластерами и данными с типом Вариант

Функции работы с кластерами (рис. 2.30) позволяют собирать кластеры из отдельных элементов и разделять кластеры на элементы, а также формировать массивы кластеров.

В таблице рассмотрены функции работы с кластерами.

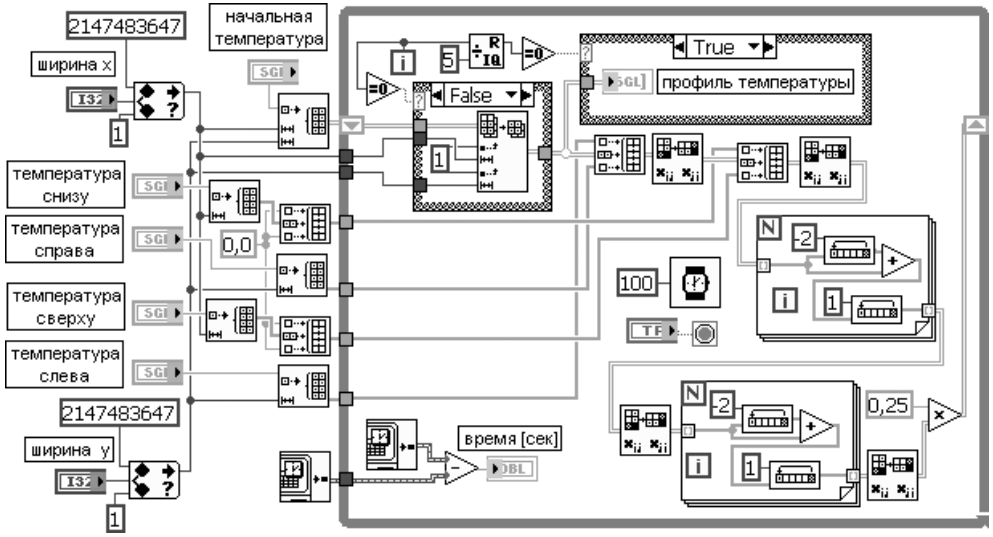


Рис. 2.29. Блок-диаграмма ВП ***Уравнение нагрева*** *(Heat Equation)*

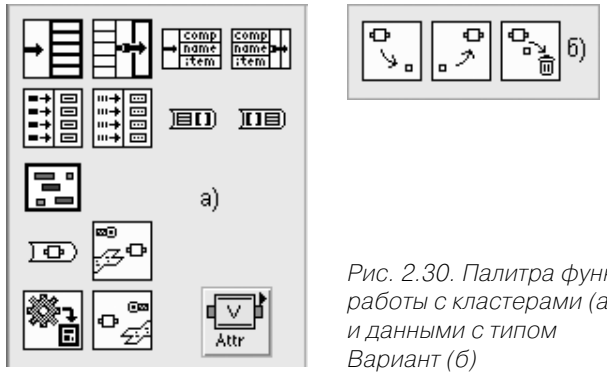
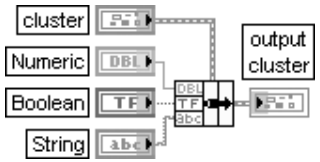


Рис. 2.30. Палитра функций работы с кластерами (а) и данными с типом Вариант (б)

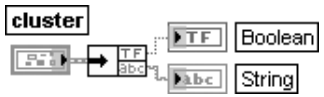
Bundle



Сборка кластера
Функция собирает кластер из отдельных элементов. Эту же функцию можно использовать для изменения значений отдельных элементов существующего кластера без необходимости определения новых значений для всех элементов. Для выполнения этого необходимо подключить изменяемый кластер к средней части терминала данной функции (рисунок диаграммы). При подключении кластера к функции она автоматически изменяет размер для отображения каждого элемента кластера.

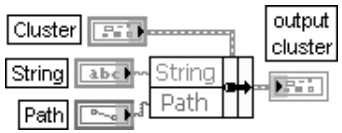
При создании нового кластера количество входов данной функции может быть установлено путем растяжения терминала функции по вертикали с помощью инструмента перемещения. Если ко входу **кластер** (cluster) ничего не подключено, то все другие входы терминала функции должны быть подключены. При подключении кластера ко входу кластер количество входов уже не может быть изменено. При этом существующие входы являются опциональными. LabVIEW заменяет только те элементы, которые подключены

Unbundle



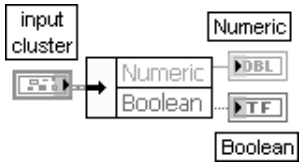
Разделение кластера
Функция разделяет **кластер** (cluster) на его отдельные элементы. При подключении кластера к данной функции она автоматически изменяет размер с целью отображения выходов всех элементов подключенного кластера. Данная функция создает выходы этих элементов в том же порядке, в каком они существуют в кластере. Число выходов этой функции должно соответствовать числу элементов кластера

Bundle By Name



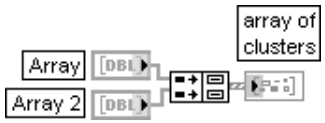
Сборка кластера по имени
Функция заменяет один или более элементов кластера. Эта функция при выборе элементов кластера учитывает имя элемента, а не его позицию в кластере. После подключения узла к входному кластеру с помощью строки **выбрать пункт** (Select Item) контекстного меню терминала имени можно выбрать имя элемента, соответствующее данному входу. Эту же операцию можно выполнить и с помощью инструмента **Управление**, щелкнув им на терминале имени и выбрав имя из списка

Unbundle By Name



Разделение кластера по имени
Функция возвращает элементы кластера с их именами. Эта функция не требует соответствия между количеством элементов в кластере и числом ее выходов

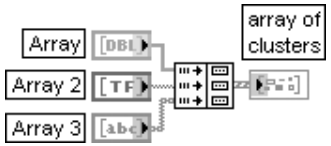
Build Cluster Array



Создание массива кластеров
Функция преобразует каждый компонент в кластер и собирает все кластеры компонентов в массив кластеров. Каждый кластер содержит единственный компонент.

LabVIEW не позволяет создавать массив из массивов, однако с помощью данной функции можно создать массив кластеров, где каждый кластер содержит массив. Пример формирования такого массива кластеров приведен на рисунке

Index & Bundle Cluster Array



Индексирование и формирование массива кластеров

Функция индексирует набор массивов и создает массив кластеров, в котором **i**-й элемент (кластер) содержит **i**-е элементы каждого входного массива

В следующих таблицах приведены описания функций работы с данными, имеющими тип **Вариант** (Variant).

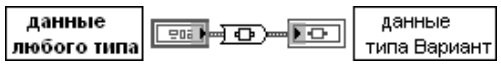
Вариант (Variant) представляет тип данных, который хранит значение и метаданные, определяющие способ интерпретации данных. Вариант как тип данных широко применяется в Visual Basic и доступен в C++.

Данные типа Вариант применяются для передачи значений между программами, написанными с помощью различных компиляторов и использующими различные типы данных. Для выполнения этой функции тип Вариант должен соответствовать общему формату. Этот формат определяется как часть спецификации Microsoft COM (Component Object Model). Много компонентов ActiveX, которые могут быть включены в LabVIEW, используют Вариант при вызове методов и свойств.

Вариант позволяет манипулировать данными без указания их типа во время компиляции.

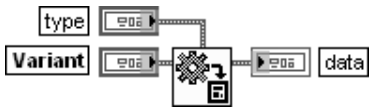
To Variant

Преобразовать к типу Вариант

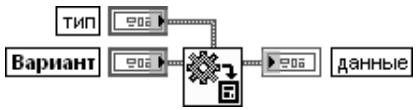


Функция преобразует данные LabVIEW любого типа в данные типа **Вариант**. Эту функцию можно использовать и для преобразования данных ActiveX в данные типа **Вариант**.
Вход **данные любого типа** (anything) служит для передачи данных LabVIEW любого типа, которые необходимо преобразовать. Этот вход поддерживает полиморфность функции

Variant To Data



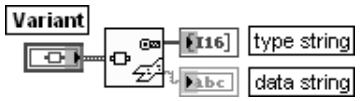
Данные типа Вариант в данные LabVIEW



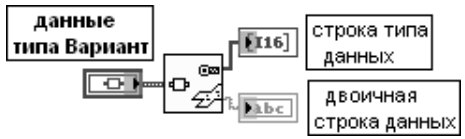
Функция преобразует данные типа **Вариант** в данные LabVIEW так, что LabVIEW может отображать или обрабатывать данные. Эту функцию можно использовать для преобразования данных типа **Вариант** в данные ActiveX.

Вход **тип** (type) задает тип данных LabVIEW, к которому необходимо преобразовать данные типа **Вариант**. **Тип** может быть произвольным типом данных. Если LabVIEW не может преобразовать данные, подающиеся на вход **Вариант**, к типу, заданному на входе **тип**, то функция возвращает ошибку. Если данные являются целыми, то необходимо преобразовать их к представлению с плавающей запятой.
Вход **Вариант** передает данные типа **Вариант**, которые необходимо преобразовать к типу данных LabVIEW, заданному на входе **тип**.
Выход **данные** (data) отображает данные типа **Вариант**, измененные в соответствии с типом, установленным на входе **тип**. Если **Вариант** не может быть преобразован к заданному типу данных, то выход **данные** возвращает значение с типом данных по умолчанию

Variant To Flattened String

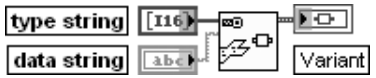


Данные типа Вариант в приведенную строку

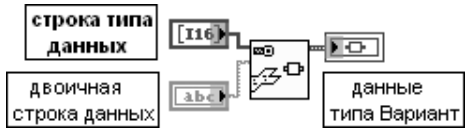


Функция преобразует данные типа **Вариант** в **приведенные данные** (flattened) и возвращает **двоичную строку данных** (data string), содержащую двоичное представление данных, и **строку типа данных** (type string), содержащую дескриптор типа, который описывает тип данных **Вариант**.
Выходы **двоичная строка данных** (data string) и **строка типа данных** (type string) были рассмотрены выше при описании функции **Перевести в строку** (Flattened To String)

Flattened String To Variant

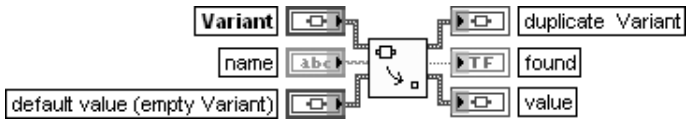


Приведенную строку в данные типа Вариант

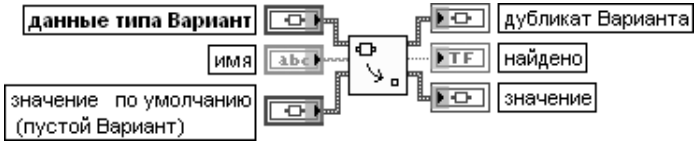


Функция преобразует приведенную строку в данные типа **Вариант**. Для формирования **двоичной строки данных** (data string) и **строки типа данных** (type string) необходимо использовать функцию **Перевести в строку** (Flatten to String).
Входы **двоичная строка данных** (data string) и **строка типа данных** (type string) были рассмотрены выше при описании функции **Перевести в строку** (Flattened To String)

Get Variant Attribute



Получить атрибуты данных типа Вариант



Функция извлекает **имена** (names) и **значения** (values) всех атрибутов или **значение** (value) простого атрибута в зависимости от вида параметра на входе **имя** (name). Соединительная панель отображает типы данных по умолчанию для этой полиморфной функции.

Вход **Вариант** представляет данные типа **Вариант**, для которых необходимо извлечь атрибут(ы) и значение(я).

Вход **имя** (name) задает имя атрибута, значение которого необходимо извлечь. Если необходимо извлечь все атрибуты, связанные с определенным **Вариантом**, этот вход не должен быть подключен. При подключении **имени** выход **имена** (names) преобразуется в логический выход **найдено** (found), массив **выходные значения** (output values) преобразуется в выход, называемый **значение** (value) того же типа, что и **значение по умолчанию** (default value), и функция ищет только определенный атрибут.

Вход **значение по умолчанию** (default value) представляет задаваемые значение и тип данных. Если функция не находит атрибут, определенный в **имени** (name), то она возвращает **значение по умолчанию**. Если **значение по умолчанию** подключено, то необходимо подключать и вход **имя**.

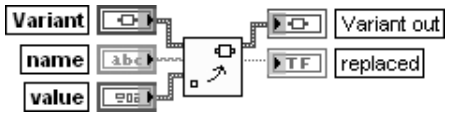
Выход **дубликат Варианта** (duplicate Variant) повторяет данные типа **Вариант**, поданные на одноименный вход.

Выход **имена** (names) возвращает одномерный массив, содержащий имена всех атрибутов, связанных с Вариантом. При подключении **имени** выход **имена** преобразуется в логический выход **найдено** (found). Выход **найдено** возвращает значение ИСТИНА, если функция находит атрибут, определенный в **имени**.

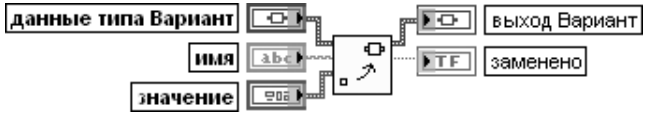
Выход **значения** (values) возвращает одномерный массив, содержащий значения, связанные с каждым атрибутом определенного Варианта, в формате варианта. Необходимо **восстановить** (unflatten) значение каждого атрибута в соответствующий тип данных. Если вход **имя** подключен, то этот выход изменяется на простое **значение** (value), имеющее тип **Вариант**. Если функция не находит атрибут, который определен в **имени** (name), то она возвращает значение, подаваемое на вход **значение по умолчанию** (default value).

Таким образом, рассматриваемая функция имеет два режима поведения, зависящих от подключения параметра **имя**. По умолчанию функция возвращает **имена** (names) всех атрибутов и их соответствующие **значения** (values) в виде одномерного массива. При подключении **имени** выход **имена** преобразуется в логический выход **найдено**, выход **значения** преобразуется в простое **значение** (value), имеющее тип **Вариант**, и функция ищет только заданный атрибут. Если функция не находит заданный атрибут(ы) или она не может преобразовать атрибут(ы) к **значению по умолчанию**, то выход **найдено** выводит ЛОЖЬ, а выход **значение** отображает содержание **значения по умолчанию**

Set Variant Attribute



Установить атрибуты данных типа Вариант



Функция изменяет или создает атрибут и значение для данных типа **Вариант**.

Вход **Вариант** передает данные типа Вариант, для которых необходимо создать атрибут и значение или заменить значение.

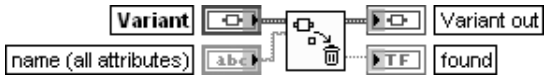
Вход **имя** (name) представляет имя атрибута, который редактируется или создается. Если имя совпадает с атрибутом, то эта функция заменяет атрибут значением, заданным на входе **значение** (value). Если имя не совпадает с атрибутом, то эта функция создает новый атрибут.

Вход **значение** (value) представляет значение атрибута. Этот вход является полиморфным, и на него можно подавать данные любого типа.

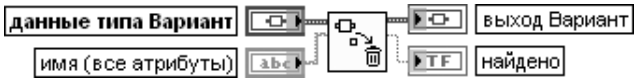
Выход Вариант (Variant out) отображает данные типа Вариант с новыми атрибутами.

Выход **заменено** (replaced) индицирует значение ИСТИНА, если атрибут и значение были заменены

Delete Variant Attribute



Удалить атрибуты данных типа Вариант



Функция удаляет атрибут(ы) и значение(я) данных типа **Вариант**, подаваемых на вход Variant.

Вход **имя** (name) содержит имя атрибута, который необходимо удалить. По умолчанию предполагается удаление всех атрибутов и связанных с ними значений данных типа **Вариант**. Если **имя** совпадает с атрибутом, то эта функция удаляет атрибут и его значение.

Выход Вариант (Variant out) отображает данные типа Вариант с удаленным(и) атрибутом(ами).

Выход **найдено** (found) имеет состояние ИСТИНА, если функция находит атрибут, определенный в **имени**

2.1.7. Функции установления времени

Функции установления времени (рис. 2.31) позволяют вводить задержки в работу ВП или измерять длительность выполнения отдельных операций, вводить текущее время и дату.

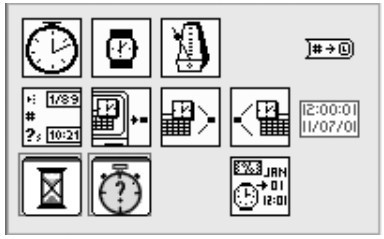
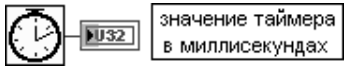


Рис. 2.31. Палитра функций установления времени

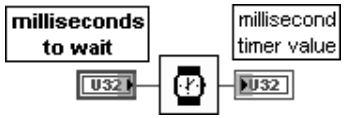
Tick Count (ms)

Счетчик времени (мс)

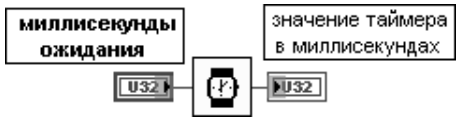


Функция возвращает значение внутренних часов операционной системы в миллисекундах. Начальное время отсчета (нуль миллисекунд) является неопределенным. Таким образом, невозможно преобразовать значение времени на выходе функции **значение таймера в миллисекундах** (millisecond timer value) в реальное время или дату. При использовании функции в операциях сравнения необходимо учитывать, что значение таймера сбрасывается в 0 после значения $2^{32}-1$

Wait (ms)

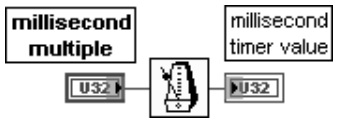


Ожидание (мс)

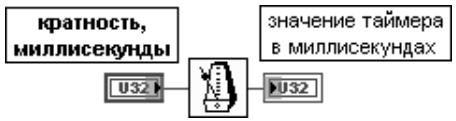


Функция ожидает заданное число миллисекунд и возвращает значение таймера в миллисекундах. Данная функция выполняет асинхронные системные вызовы, но сама работает синхронно. Следовательно, она не завершит выполнение до истечения заданного времени

Wait Until Next ms Multiple

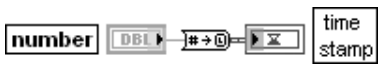


Задержка до следующего кратного интервала мс

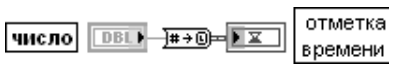


Функция заставляет ВП ожидать, пока показания внутренних часов не сравняются или не превысят кратного количества миллисекунд, поданных на вход функции **кратность, миллисекунды** (millisecond multiple). Данную функцию целесообразно использовать для синхронизации выполняемых операций. Так, например, функцию можно использовать в цикле для управления скоростью его выполнения. При этом первый цикл может быть короче остальных

To Time Stamp

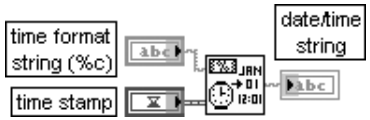


Число в отметку времени

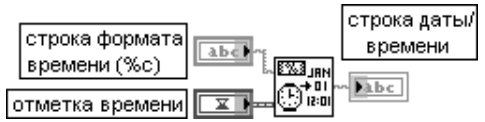


Функция преобразует число в отметку времени. Отметка времени представляет не зависящее от временного пояса число полных секунд, прошедших с 0:00 1 января 1904 года по Гринвичу

Format Date/Time String



Форматировать строку даты/времени



Функция отображает значение времени или числовое значение как время в заданном пользователем формате. Коды формата времени включают следующие параметры: %H (часы, 24-часовой интервал), %I (часы, 12-часовой интервал), %M (минуты), %S (секунды), %p (флаг до полудня/после полудня), %d (дни месяца), %m (номер месяца в году), %y (номер года в веке), %Y (номер года, включая век), %a (сокращенное название дня недели), %x (дата в локальной спецификации), %X (время в локальной спецификации), %c (дата/время в локальной спецификации) и <цифра> (дробная часть секунды с точностью, заданной значением <цифра>).

Вход **строка формата времени** (time format string) определяет формат выходной строки. Коды формата времени (начинающиеся с %) не воспринимаются функцией как код, точно возвращающий символ. По умолчанию установлен код %c, который соответствует представлению даты/времени в месте локализации компьютера. Если **строка формата времени** является пустой строкой, то функция использует значение по умолчанию.

Вход **отметка времени** (time stamp) может быть значением времени или числом. В числовом представлении этот параметр представляет не зависящее от временного пояса число полных секунд, прошедших с 0:00 1 января 1904 года по Гринвичу. По умолчанию берется текущая дата и время.

Выход **строка даты/времени** (date/time string) представляет отформатированную строку даты/времени.

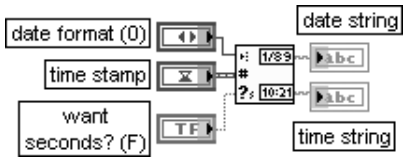
Данная функция формирует **строку даты/времени** путем копирования **строки формата времени** и замены каждого из **кодов формата времени** (time format codes) соответствующим значением. Коды формата времени дополняются нулями, которые обеспечивают постоянную ширину поля. Дополнительный модификатор # перед буквой кода формата удаляет начальные нули из следующих кодов формата:

%#d, %#H, %#I, %#j, %#m, %#M, %#s, %#S, %#U, %#w, %#W, %#X, %#y, %#Y

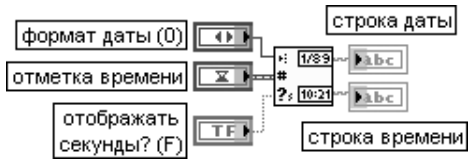
Модификатор # не изменяет поведение других кодов формата.

Коды формата %c, %x, %X и %Z зависят от локальной операционной системы

Get Date/Time String



Получить строку даты/времени



Функция преобразует значение времени или числовое значение в строку даты и строку времени, которые соответствуют временному поясу места расположения компьютера. Функция интерпретирует время или числовое значение как не зависящее от временного пояса число полных секунд, прошедших с 0:00 1 января 1904 года по Гринвичу.

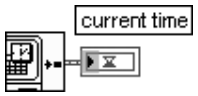
Вход **формат даты** (date format) устанавливает представление выхода **строка даты** (date string). Форматы даты изменяются в зависимости от конфигурации системы. Для получения различных форматов даты необходимо использовать функцию **Строка формата даты/времени** (Format Date/Time String). В табл. 2.12.1 приведены варианты формата даты при использовании системного формата

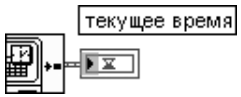
0	**Короткий** (short) – 20.01.2004
1	**Длинный** (long) – 20 января 2004 г.
2	**Сокращенный** (abbreviated) – 20 янв 2004 г.

Вход **отметка времени** (time stamp) рассмотрен выше при анализе функции **Форматировать строку даты/времени**.
Вход **отображать секунды?** (want seconds?) устанавливает отображение секунд на выходе **строка времени**.
Выход **строка даты** (date string) представляет строку, возвращаемую функцией в соответствии с форматом, определенным на входе **формат даты** (date format).
Выход **строка времени** (time string) возвращает строку, отформатированную в соответствии с временным поясом места расположения компьютера

Get Date/Time In Seconds

Получить дату/время в секундах

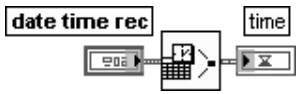


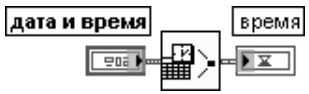


Функция возвращает на выходе **текущее время** (current time) значение текущего времени, равного системному времени LabVIEW или числу полных секунд, прошедших с 0:00 1 января 1904 года по Гринвичу

Date/Time To Seconds

Дату/время в секунды

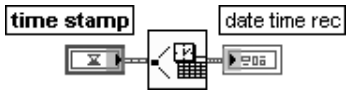


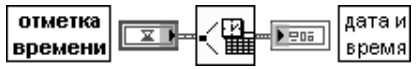


Функция преобразует кластер **дата и время** (date time rec), состоящий из 32-битовых целых чисел со знаком, представляющих время компьютера в месте его локализации, в значение текущего **времени** (time), определенное как число полных секунд, прошедших с 0:00 1 января 1904 года по Гринвичу.
Кластер **дата и время** включает следующие элементы:
секунды (second) от 0 до 59;
минуты (minute) от 0 до 59;
часы (hour) от 0 до 23;
день месяца (day of month) от 1 до 31;
месяц (month) от 1 до 12;
год (year) от 1904 до 2040;
день недели (day of week) от 1 до 7, что соответствует дням от воскресенья до субботы. Функция игнорирует данный параметр;
день года (day of year) от 1 до 366. Функция игнорирует данный параметр;
is DST устанавливает стандартное время (0) или «летнее» время (1). Функция игнорирует данный параметр

Seconds To Date/Time

Секунды в дату/время



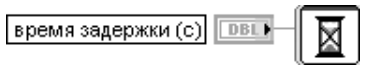


Функция преобразует значение времени или числовое значение в не зависящее от временного пояса число полных секунд, прошедших с 0:00 1 января 1904 года по Гринвичу, и возвращает кластер из девяти 32-битовых целых чисел со знаком, определяющих значение времени или числовое значение.
Компоненты кластера **дата и время** были рассмотрены выше при анализе функции Дату/время в секунды

В состав палитры функций времени входят Экспресс-ВП **Временная задержка** (Time Delay) и **Истекшее время** (Elapsed Time).

Time Delay

Временная задержка



Экспресс-ВП **Временная задержка** (Time Delay) вносит временную задержку в выполнение ВП. Величина задержки может задаваться с помощью элемента управления **временная задержка** (Time delay (seconds)) при конфигурировании ВП или с помощью элемента управления, подключаемого ко входу **время задержки (с)** (Delay Time (s)) иконки Экспресс-ВП. Значение, подаваемое на данный вход, имеет больший приоритет по сравнению с тем, что устанавливается в диалоговом окне.
Этот Экспресс-ВП использует функциональность функций **Ожидание (мс)** (Wait (ms)) и **Задержка до следующего кратного интервала мс** (Wait Until Next ms Multiple)

Истекшее время (Elapsed Time)

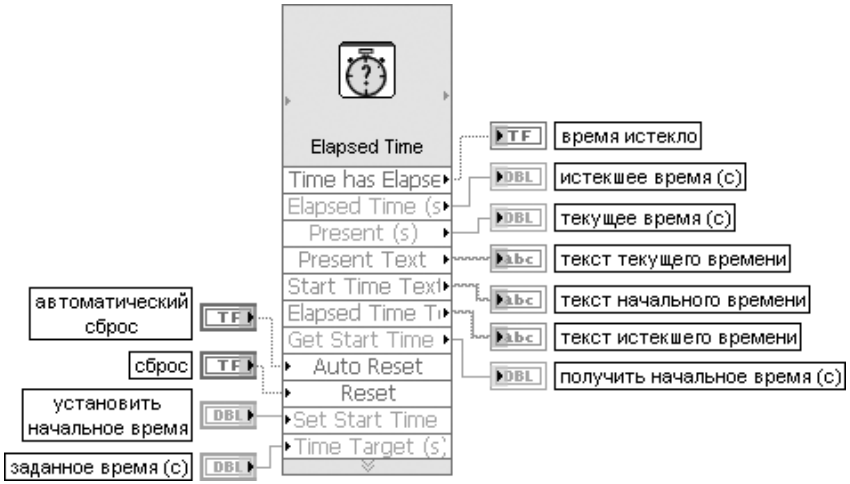


Рис. 2.32. Блок-диаграмма возможного подключения Экспресс-ВП

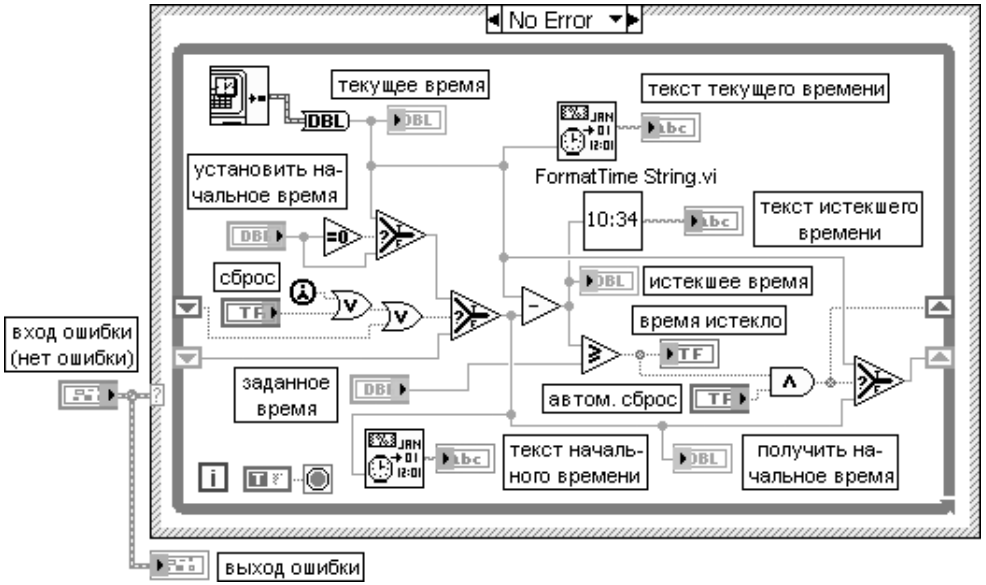


Рис. 2.33. Блок-диаграмма Экспресс-ВП ***Истекшее время*** *(Elapsed Time)*

Экспресс-ВП **Истекшее время** (Elapsed Time) сохраняет отсчеты времени для индикации момента истечения заданного интервала времени. Истекшее время определяется как разность текущего времени и заданного начального времени.

Диалоговое окно данного Экспресс-ВП имеет следующие опции:

Истекшее время (секунды) (Elapsed time (seconds)) – определяет интервал времени перед остановкой выполнения ВП. По умолчанию интервал равен 1 с.

Автоматически сбрасывать после истечения заданного времени (Automatically reset after time target) – сбрасывает маркер истекшего времени.

Входы блок-диаграммы Экспресс-ВП имеют следующие значения (рис. 2.32):

Автоматический сброс (Auto Reset) – устанавливает начальное время равным значению **текущего времени** (Present (s)), когда Экспресс-ВП достигает заданного времени (Time Target (s)).

Установить начальное время (Set Start Time (s)) – использует время, подключенное к этому входу, как начальное время вместо времени первого выполнения этого ВП.

Заданное время (Time Target (s)) – определяет заданное число секунд, которое ВП ожидает после начального времени. При достижении **Заданного времени** выход **Время истекло** (Time has Elapsed) переходит в состояние ИСТИНА.

Сброс (Reset) – управляет инициализацией внутреннего состояния ВП. По умолчанию имеет состояние ЛОЖЬ.

Выходы блок-диаграммы Экспресс-ВП имеют следующие значения:

Получить начальное время (Get Start Time (s)) – возвращает время первого выполнения ВП или время, подаваемое на вход **Установить начальное время** (Set Start Time (s)). Отображает время в секундах, прошедшее с 0:00 1 января 1904 года по Гринвичу.

Текст начального времени (Start Time Text) – отображает дату и время первого выполнения ВП или время, подаваемое на вход **Установить начальное время** (Set Start Time (s)).

Текущее время (с) (Present (s)) – отображает текущее время в секундах, прошедшее с 0:00 1 января 1904 года по Гринвичу.

Текст текущего времени (Present Text) – отображает текущие дату и время.

Истекшее время (с) (Elapsed Time (s)) – отображает время в секундах, прошедшее от **начального времени** до **текущего времени**.

Текст истекшего времени (Elapsed Time Text) – отображает время в секундах, прошедшее от **начального времени** до **текущего времени**.

Время истекло (Time has Elapsed) – включает индикатор, когда значение **Истекшее время** превышает сумму **начального времени** и **заданного времени** (Time Target (s)).

Этот Экспресс-ВП использует функции **Получить дату/время в секундах** (Get Date/Time In Seconds) и **Строка формата даты/времени** (Format Date/Time String) (рис. 2.33)

2.1.8. Функции и ВП ввода/вывода файлов

Функции и ВП ввода/вывода файлов (File I/O) выполняют файловые операции записи и считывания данных. Они размещены в основной палитре функций (рис. 2.34а) и в ряде дополнительных подпалитр: **Дополнительные файловые функции** (Advanced File Functions) (рис. 2.34б), **ВП Zip-файлов** (Zip VIs) (рис. 2.34в), **ВП хранения** (Storage VIs) (рис. 2.34г), **Потоковые TDM** (TDM Streaming) (рис. 2.34д), **Файловые константы** (File Constants) (рис. 2.34е) и **ВП файлов конфигурации** (Configuration File VIs) (рис. 2.34ж). Подпалитра **Дополнительные файловые функции** содержит, в свою очередь, подпалитру **Функции файлов протокола** (Datalog Functions) (рис. 2.34з). Необходимо отметить также, что ВП ввода/вывода файлов включены в палитры функций работы с осциллограммами и звуковыми сигналами. Далее они рассмотрены в соответствующих разделах.

Функции ввода/вывода файлов LabVIEW используют файлы различных форматов: текстовые, двоичные, файлы протокола (datalog file), текстовые (LVM) и двоичные (TDMS) файлы измерений, а также двоичные файлы с заголовком XML (TDM). Тип формата зависит от типа получаемых или формируемых данных и от приложения, в котором они будут использоваться. Так, в частности, если предполагается использовать данные в таких приложениях, как Excel, то целесообразно записывать их в виде текстовых файлов. Текстовый формат отличается большей универсальностью, однако он требует большего времени на выполнение преобразований и большего объема памяти для хранения. Если же необходимо обеспечить произвольный доступ к данным, высокие скорости записи/чтения при минимальном объеме дисковой памяти, то следует использовать двоичный формат. Текстовые и двоичные файлы известны как файлы потока данных, поскольку они сохраняют данные в виде последовательности символов или байтов. Формат файлов протокола применяется при сохранении данных сложной структуры и последующем их использовании только в рамках LabVIEW.

Среди функций ввода/вывода файлов можно выделить функции высокого уровня (High-Level), представленные в основной палитре, и функции низкого уровня (Low-Level), размещенные в подпалитре **Дополнительные файловые функции**.

Функции высокого уровня используются для выполнения общих операций ввода/вывода текстовой или числовой информации. Выполнение таких операций

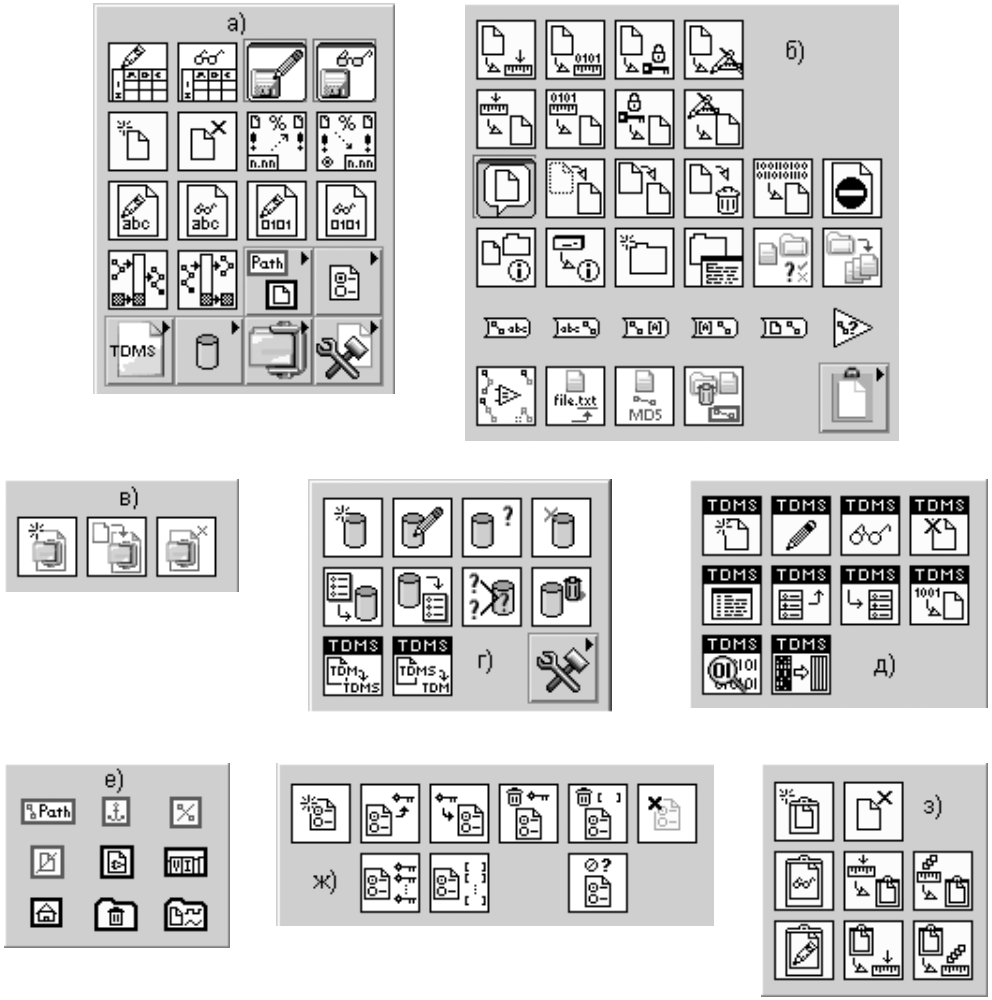
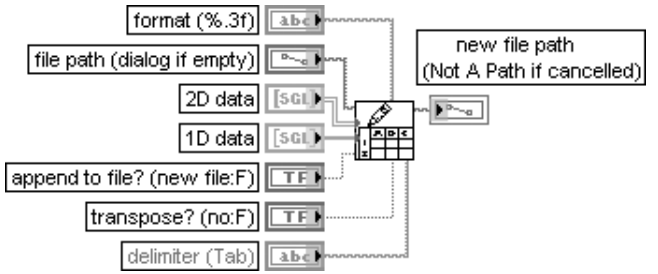


Рис. 2.34. Вид основной палитры (а)
и дополнительных подпалитр (б–з) функций ввода/вывода файлов

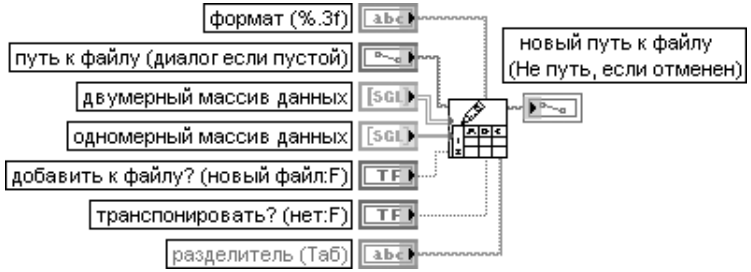
включает, как правило, три этапа: открытие уже существующих файлов или создание новых файлов; запись в файл или чтение из файла; закрытие файла. Функции низкого уровня служат для выполнения отдельных разделов функций высокого уровня и, помимо этого, таких файловых операций, как создание каталогов, перемещение, копирование или удаление файлов, вывод содержания каталогов, изменение файловых характеристик и операции с путями.

Ниже в таблицах приведено описание функций ввода/вывода файлов, находящихся в основной палитре.

Write To Spreadsheet File



Записать в файл табличного формата



ВП преобразует двух- или одномерный массив чисел с одинарной точностью в текстовую строку и записывает эту строку в виде нового байтового файла или добавляет строку к существующему файлу. Данный ВП перед началом записи открывает или создает файл, а после окончания записи закрывает его.

Этот ВП можно использовать для создания текстового файла, воспринимаемого большинством табличных приложений.

Блок-диаграмма ВП приведена на рис. 2.35. Как видно из рисунка, для преобразования массива в строку электронной таблицы ВП использует функцию **Массив в строку табличного формата** (Array To Spreadsheet String) из палитры строковых функций.

Вход **формат** (format) определяет способ преобразования входных данных в строку. По умолчанию установлен определитель формата %.3f.

Вход **путь к файлу** (file path) определяет путь к файлу, в который производится запись. Если вход не подключен, то открывается диалоговое окно для указания пути.

На вход **двумерные данные** (2D data) подаются числа с одинарной точностью, которые ВП записывает в файл, если вход **одномерные данные** (1D data) не подключен или на него ничего не поступает.

На вход **одномерные данные** (1D data) также поступают числа с одинарной точностью, записываемые в файл. ВП преобразует одномерный массив в двумерный, предварительно транспонируя и преобразуя его в строку, а затем записывает его в файл.

Вход **добавить к файлу** (append to file?) устанавливается в состояние ИСТИНА, если данные добавляются к существующему файлу, и устанавливается в состояние ЛОЖЬ (состояние по умолчанию) при записи данных в новый файл или перезаписи существующего файла.

Вход **транспонировать?** (transpose?) определяет выполнение транспонирования данных (состояние ИСТИНА) или передачу данных без транспонирования (состояние ЛОЖЬ) (по умолчанию).

Вход **разделитель** (delimiter) определяет разделитель (символ или строку символов), используемый для разделения полей в файле электронной таблицы. По умолчанию в качестве разделителя используется символ табуляции.

Выход **новый путь к файлу** (new file path) определяет путь к файлу, в который ВП произвел запись данных

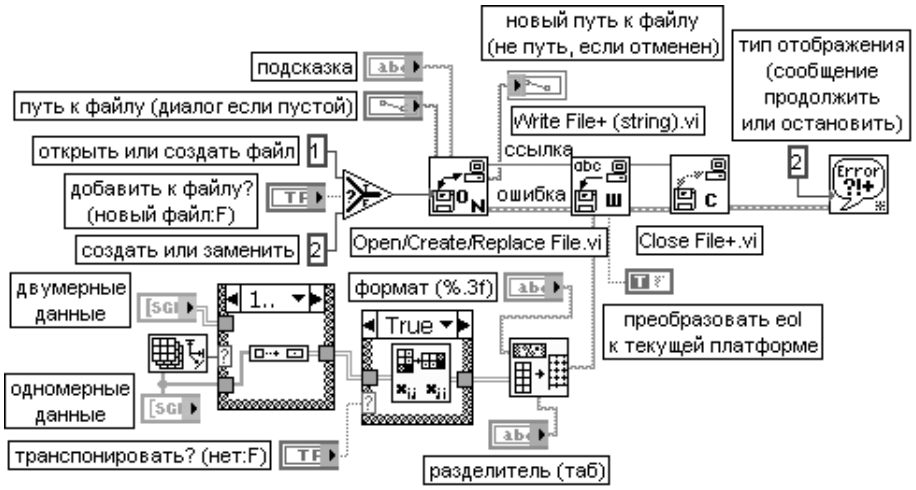
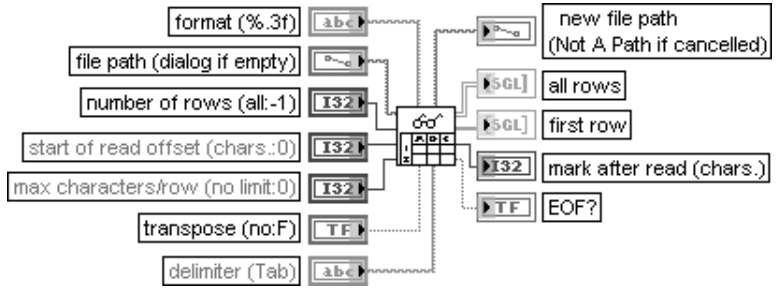
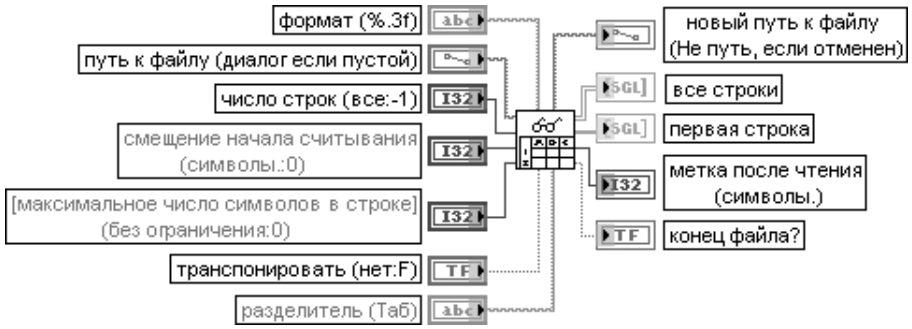


Рис. 2.35. Блок-диаграмма ВП ***Записать в файл табличного формата*** *(Write To Spreadsheet File)*

Read From Spreadsheet File



Считать из файла табличного формата



ВП считывает определенное число линий или строк из текстового файла, начиная с определённого начального символа, а затем преобразует данные в двумерный массив чисел с одинарной точностью. Данный ВП открывает файл перед выполнением операции чтения и закрывает его после завершения чтения. Этот ВП можно использовать для чтения табличного файла, сохраненного в текстовом формате.

Вход **число строк** (number of rows) определяет максимальное число строк или линий, считываемых ВП. Для данного ВП линия – это строка элементов, заканчивающаяся символами «возврат каретки», «перевод строки» или «возврат каретки», сопровождаемый символом «перевод строки».

Вход **смещение начала считывания** (start of read offset) определяет позицию в файле, задаваемую числом символов, с которой ВП начинает чтение.

Вход **максимальное число символов в строке** (max characters per row) задает максимальное число символов, считываемых ВП перед окончанием поиска конца строки или линии.

Входы **формат** (format), **путь к файлу** (file path), **транспонировать?** (transpose?) и **разделитель** (delimiter) идентичны одноименным входам рассмотренного выше ВП **Записать в файл табличного формата** (Write To Spreadsheet File).

Выход **новый путь к файлу** (new file path) определяет путь к файлу, из которого ВП считал данные.

Выход **все строки** (all rows) отображает данные, считанные из файла, в форме двумерного массива чисел с одинарной точностью.

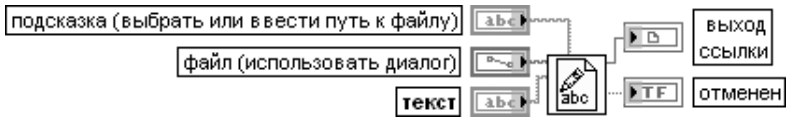
На выходе **первая строка** (first row) выводится первая строка из всего массива строк **все строки**.

Выход **метка после чтения** (mark after read) указывает на расположение метки файла после чтения.

Выход **конец файла?** (EOF?) устанавливается в состояние ИСТИНА при попытке чтения области после конца файла

Write to Text File

Записать в текстовый файл



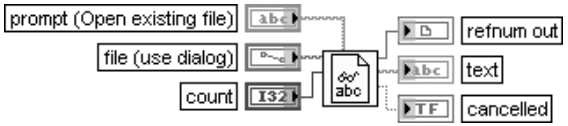
Функция записывает строку символов (string of characters) или массив строк в файл. Если ко входу **файл** (file) подключен путь (path), то перед записью функция открывает или создает файл и заменяет его предыдущее содержимое. Если к этому входу подключена ссылка, то запись начинается с текущей позиции файла. Для того чтобы произвести дозапись к существующему файлу, необходимо установить позицию файла в его конец с помощью функции **Установить позицию файла** (Set File Position). Эту же функцию можно использовать для организации произвольного доступа.

Вход **подсказка** (**выбрать или ввести путь к файлу**) (prompt (Choose or enter file path)) представляет сообщение, которое появляется в заголовке файлового диалогового окна. По умолчанию это пустая строка.

Вход **файл (использовать диалог)** (file (use dialog)) может содержать ссылку или абсолютный путь к файлу. Если на вход **файл** подается путь, то функция открывает файл, заданный этим путем.

По умолчанию отображается окно файлового диалогового окна и напоминание для выбора файла. Эта функция создает заданный файл, если он еще не существует. Если задать пустой или относительный путь, то функция вернет ошибку.
На вход **текст** (text) подаются данные, которые функция записывает в файл. Это может быть строка или массив строк. Эта функция всегда добавляет зависящие от платформы символы конца строки к элементам массива вне зависимости от наличия или отсутствия отметки в строке **Преобразовать EOL** (Convert EOL) ее контекстного меню.
Выход ссылки (refnum out) представляет ссылку на файл, в который функция производит запись. Этот выход может быть подключен к другой файловой функции в зависимости от того, что необходимо делать с этим файлом.
Выход **отменен** (cancelled) отображает значение ИСТИНА, если пользователь отменил отображение файлового диалогового окна или не выбрал замену в предупреждающем диалоговом окне

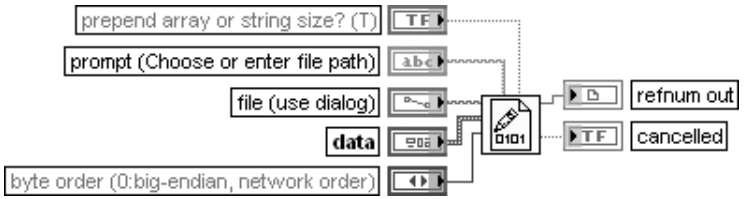
Read from Text File



Считать из текстового файла

Функция считывает определенное число символов или строк из файла, представляющего поток байтов. По умолчанию эта функция считывает все символы из текстового файла. Считывание заданного количества символов, начиная с первого, производится с помощью входа **подсчет** (count). При установке отметки в строке **Считать строки** (Read Lines) контекстного меню функции вход **подсчет** определяет количество считываемых строк. Установка значения –1 на этом входе определяет считывание всех символов в строке или всех строк текстового файла.
Выход **текст** (text) содержит текст, считанный из файла. По умолчанию это строка, содержащая символы первой строки файла. Если подключить вход подсчет, то на этот выход будет выводиться массив строк, считанных из файла. При удалении отметки **Считать строки** на этот выход будут выводиться все символы, считываемые из файла. Функция преобразует все зависящие от платформы символы конца строки в аналогичные символы LabVIEW независимо от состояния строки Convert EOL контекстного меню функции

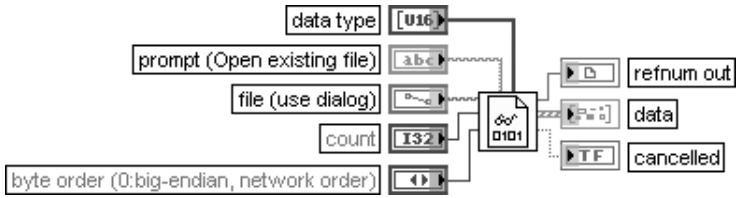
Write to Binary File — Записать в двоичный файл



Функция записывает двоичные данные в новый файл или добавляет данные к существующему файлу.
Вход **добавить в начало размер массива или строки?** (prepend array or string size?) устанавливает опцию включения информации о размере данных в начало **выхода ссылки** (refnum out) при подаче на вход **данные** массива строк. Если на этом входе установлено значение ЛОЖЬ, то LabVIEW не включает информацию о размере. По умолчанию установлено значение ИСТИНА. Этот вход только управляет информацией верхнего уровня о размере данных. Массивы и строки, входящие в состав таких иерархических типов данных, как кластеры, всегда включают информацию о размере.
Вход **данные** (data) содержит записываемые в файл данные, которые могут быть любого типа.
Вход **порядок байтов** (byte order) устанавливает endian-форму результирующих **данных**. Порядок байтов, или endian-форма, показывает, как представлены целые числа в памяти – от более значимого байта к менее значимому или наоборот. Функция должна считывать данные в том же порядке, как они были записаны.

0	**big-endian, порядок в сети (по умолчанию)** – наиболее значимый байт занимает самые младшие адреса памяти. Использовался в Mac OS и в случае, если считываемые данные записаны на другой платформе
1	**Собственный порядок главного компьютера** – использует байт-ориентированный формат главного компьютера. Увеличивает скорость чтения и записи
2	**little-endian** – наименее значимый байт занимает самые младшие адреса памяти. Использовался в Windows и Linux

Read from Binary File

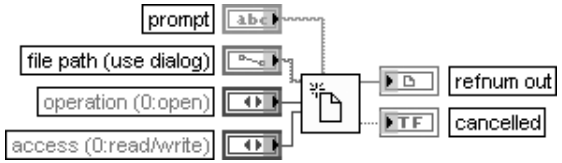


Считать из двоичного файла

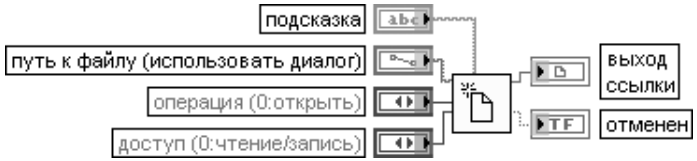
Функция считывает двоичные данные из файла и возвращает их на выходе **данные** (data). То, как считываются данные, зависит от формата конкретного файла.
Вход **тип данных** (data type) устанавливает тип данных, используемый функцией для считывания из двоичного файла. Функция интерпретирует данные, начинающиеся с текущей позиции файла, как набор экземпляров с заданным типом данных, число которых определяется входом **подсчет** (count). Если данные представляют массив, строку или кластер, содержащий массив или строку, функция принимает, что каждый экземпляр этого типа данных содержит информацию о размере. При отсутствии такой информации функция интерпретирует данные ошибочно. Если LabVIEW определяет, что данные не соответствуют типу, то устанавливается значение данных по умолчанию для заданного типа и возвращается ошибка.

Вход **подсчет** (count) задает число считываемых элементов данных. Элементами данных могут быть байты или экземпляры с заданным **типом данных**. Функция возвращает на выходе **данные** количество элементов данных, заданное на входе **подсчет**, или число всех полных элементов данных при достижении конца файла. В последнем случае выдается ошибка конца файла.
По умолчанию функция возвращает единичный элемент данных.
Если **подсчет** относится к массиву элементов и заданным типом данных является массив, функция автоматически возвращает кластер массивов или массив кластеров, поскольку LabVIEW не разрешает создание массива массивов

Open/Create/Replace File



Открыть/Создать/Заменить файл

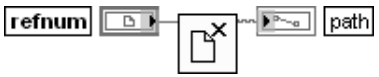


Функция открывает существующий файл, создает новый файл или заменяет существующий файл, программно или интерактивно используя файловое диалоговое окно. Пользователь может дополнительно определить **подсказку** диалога (prompt). Эта функция используется совместно с функциями записи и считывания файлов.
Вход **подсказка** (prompt) представляет сообщение, которое появляется в заголовке файлового диалогового окна. По умолчанию это пустая строка.
Вход **путь к файлу (использовать диалог)** (file path (use dialog)) содержит абсолютный путь к файлу. Если **путь к файлу** пустой (по умолчанию) или принимает значение <**Не путь**> (<Not A Path>), функция выводит диалоговое окно, которое позволяет выбрать файл. Если не определить **путь к файлу** или определить пустой или относительный путь, то функция вернет ошибку.
Вход **операция** (operation) определяет выполняемую операцию и имеет следующие варианты:

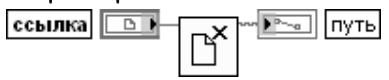
0	**Открыть** (open) – открывает существующий файл (по умолчанию). Если файл не может быть найден, происходит ошибка 7
1	**Заменить** (replace) – заменяет существующий файл путем его открытия и установки нулевого значения для конца файла
2	**Создать** (create) – создает новый файл. Если файл уже существует, то происходит ошибка 10
3	**Открыть или создать** (open or create) – открывает существующий файл или создает новый, если таковой не существует
4	**Заменить или создать** (replace or create) – создает новый файл или заменяет файл, если он существует. Эта функция заменяет файл путем его открытия и установки нулевого значения для конца файла
5	**Заменить или создать с подтверждением** (replace or create with confirmation) – создает новый файл или заменяет существующий файл при наличии прав доступа. Замена файла производится так же, как и в предыдущих операциях

Вход **доступ** (access) определяет вид доступа к файлу. По умолчанию (0) это чтение/запись. Другие варианты включают только чтение (1) и только запись (2).
Выход ссылки (refnum out) отображает ссылку на открытый файл. Если файл не может быть открыт, на выход выводится значение **Не ссылка** (Not A Refnum).
Выход **отменен** (cancelled) отображает значение ИСТИНА, если пользователь отменил отображение файлового диалогового окна или не выбрал замену в предупреждающем диалоговом окне

Close File

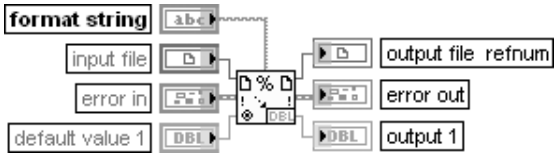


Закрыть файл

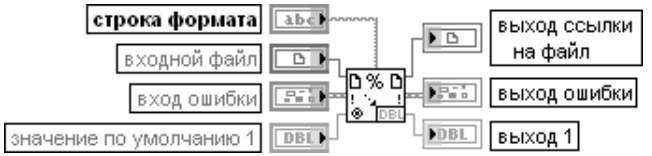


Функция закрывает открытый файл, определяемый **ссылкой** (refnum), и возвращает **путь** (path) к файлу, соответствующий ссылке. Ошибка ввода/вывода формируется только в этой функции, которая закрывает файл, несмотря на ошибки в предыдущих операциях. Это гарантирует корректное закрытие файлов

Scan From File



Просмотр файла

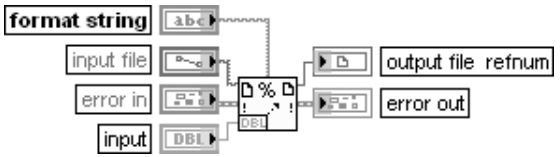


Функция просматривает текст в файле с целью обнаружения путей, строковых, числовых и логических данных, преобразует текст в данные соответствующего типа и возвращает дубль ссылки и преобразованные данные в порядке просмотра. Эта функция может быть использована для чтения всего текста в файле. Конечно, эта функция не может быть использована для определения начальной точки просмотра. Для решения такой задачи необходимо использовать ВП **Считать символы из файла** (Read Characters from File) и функцию **Просмотр строки** (Scan From String).
Вход **строка формата** (format string) определяет, как необходимо преобразовывать входную строку в выходные аргументы. По умолчанию такое преобразование осуществляется в соответствии с типом выходных выводов. Тип выходов может быть установлен или изменен с помощью опции **Редактировать строку просмотра** (Edit Scan String) контекстного меню функции.
Вход **входной файл** (input file) может быть ссылкой или путем к файлу. Если вход является ссылкой, то данная функция открывает файл, определенный этой ссылкой. По умолчанию предполагаются открытие файлового диалогового окна и вывод подсказки по выбору файла. Эта функция создает определенный файл, если он еще не существует.
Входы **по умолчанию 1..n** (default 1..n) определяют значения выходных параметров по умолчанию. Если входное значение не может быть считано из строки, функция использует значение

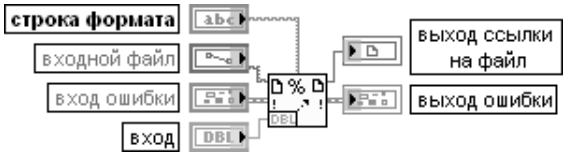
по умолчанию. Если вход **по умолчанию 1..n** не подключен, то тип данных по умолчанию определяется из **строки формата**, если **строка формата** является константой. В противном случае типом данных по умолчанию является числовой тип с плавающей запятой двойной точности. По умолчанию значение равно 0 или пустой строке в зависимости от типа данных.

Выходы 1..n (output 1..n) определяют выходные параметры. Каждый выход может быть строкой, путем, типом перечисления или каким-либо числовым типом. С этой функцией не могут быть использованы массивы и кластеры

Format Into File



Преобразовать в файл



Функция форматирует строки, пути к файлам, числовые и логические данные в текст и записывает его в файл.

Вход **строка формата** (format string) определяет способ преобразования входных аргументов. По умолчанию значение входа должно соответствовать типу данных входных аргументов. Формирование и редактирование формата входных аргументов производится с помощью опции **Редактировать строку просмотра** (Edit Scan String) контекстного меню функции.

Вход **входной файл** (input file) идентичен одноименному входу рассмотренной выше функции **Просмотр файла**.

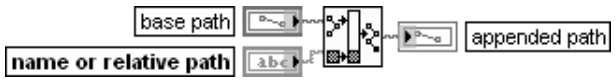
Входы 1..n (input 1..n) определяют входные преобразуемые параметры. Каждый вход может быть строкой, путем, типом перечисления или каким-либо числовым типом. С этой функцией не могут использоваться массивы и кластеры.

Выход ссылки на файл (output file refnum) представляет ссылку на файл, который ВП читает. Этот выход может быть подключен к другой файловой функции, зависящей от предполагаемых действий с файлом. По умолчанию это функция закрытия файла, если ссылка получена из файлового пути или диалогового окна. Если **входной файл** является ссылкой, то LabVIEW предполагает, что файл еще используется до закрытия пользователем.

Увеличение числа входных параметров производится с помощью выбора опции **Добавить параметр** (Add Parameter) в контекстном меню функции или увеличения размера иконки функции по вертикали с помощью инструмента перемещения.

Пользователь может применять рассмотренную функцию для определения порядка, в котором данные окажутся в файле. Эту функцию нельзя использовать для добавления данных к файлу. Чтобы сделать это, необходимо использовать функции **Преобразовать в строку** (Format Into String) и **Записать символы в файл** (Write Characters to File) или функцию **Записать файл** (Write File)

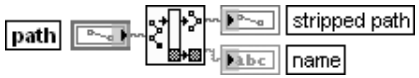
Build Path



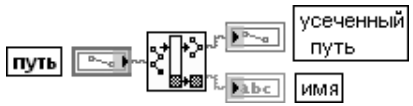
Сформировать путь

Функция формирует **добавленный путь** (appended path) с помощью добавления **имени** или **относительного пути** (name or relative path) к существующему **базовому пути** (base path)

Strip Path



Разделить путь



Функция возвращает **имя** (name) последнего компонента **пути** (path) и **усеченный путь** (stripped path), который ведет к этому компоненту

В состав палитры функций ввода/вывода файлов входят Экспресс-ВП **Записать файл результатов измерения LabVIEW** (Write LabVIEW Measurement File) и **Считать файл результатов измерения LabVIEW** (Read LabVIEW Measurement File).

Записать файл результатов измерения LabVIEW (Write LabVIEW Measurement File)

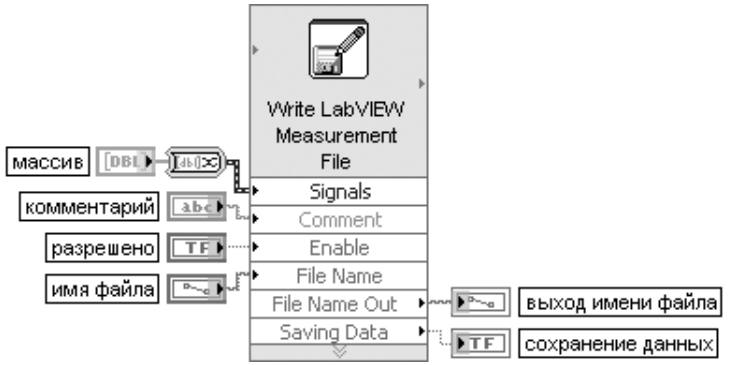


Рис. 2.36. Блок-диаграмма возможного подключения Экспресс-ВП

Экспресс-ВП записывает данные в текстовый файл результатов измерений LabVIEW с расширением .lvm или двоичный файл результатов измерений с расширением .tdm или .tdms.

Диалоговое окно данного Экспресс-ВП имеет следующие опции (рис. 2.37):

Имя файла (File name) отображает полный путь к файлу, в который записываются данные. Запись данных по этому пути производится, если одноименный вход не подключен. При подключенном входе запись производится по пути, который определяется на этом входе.

Формат файла (File Format) содержит следующие опции:

- **Текстовый (LVM)** (Text (LVM)) – устанавливает текстовый формат файла результатов измерений и расширение .lvm для файла **Имя файла**;

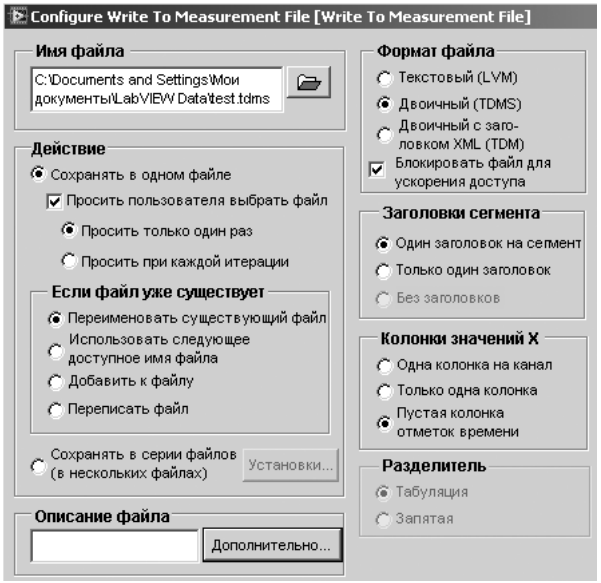


Рис. 2.37. Вид диалогового окна конфигурирования Экспресс-ВП
Записать файл результатов измерения LabVIEW
(Write LabVIEW Measurement File)

- **Двоичный (TDMS)** (Binary (TDMS)) – устанавливает двоичный формат файла результатов измерений и расширение .tdms для файла **Имя файла**. При выборе этого формата становится доступным флажок **Блокировать файл для ускорения доступа** (Lock file for faster access). Рекомендуется использовать эту опцию в большинстве случаев;
- **Двоичный с заголовком XML (TDM)** (Binary with XML Header (TDM)) – устанавливает двоичный формат файла результатов измерений и расширение .tdm для файла **Имя файла**.

Действие (Action) содержит следующие опции:
- **Сохранить в одном файле** (Save to one file) – сохраняет все данные в одном файле;
- **Просить пользователя выбрать файл** (Ask user to choose file) – отображает диалоговое окно, которое предлагает пользователю выбрать файл;
- **Просить только один раз** (Ask only once) – предлагает пользователю выбрать файл только один раз. Эта опция доступна лишь при установке флажка **Просить пользователя выбрать файл**;
- **Просить при каждой итерации** (Ask each iteration) – предлагает пользователю выбрать файл при каждом выполнении Экспресс-ВП;
- **Сохранить в серии файлов** (Save to series of files (multiple files)) – сохраняет данные в нескольких файлах;
- **Установки** (Settings) – кнопка отображает диалоговое окно **Конфигурировать установки сохранения в нескольких файлах** (Configure Multi-files Settings). Эта опция доступна лишь при установке переключателя **Сохранить в серии файлов**.

Если файл уже существует (If a file already exists) содержит следующие опции:
- **Переименовать существующий файл** (Rename existing file) – переименовывает существующий файл;
- **Использовать следующее доступное имя файла** (Use next available name) – добавляет следующее последовательное число к имени файла;
- **Добавить к файлу** (Append to file) – добавляет данные к существующему файлу;
- **Переписать файл** (Overwrite file) – заменяет данные в существующем файле.

Заголовки сегментов (Segment Headers) содержит следующие опции:
- **Один заголовок на сегмент** (One header per segment) – создает один заголовок на сегмент в файле, в который LabVIEW записывает данные;
- **Только один заголовок** (One header only) – создает только один заголовок в этом файле;
- **Без заголовков** (No headers) – не создает заголовка в файле.

Колонки значений X (X Value Columns) содержит следующие опции:
- **Одна колонка на канал** (One column per channel) – создает отдельную колонку для отсчетов времени данных, которые генерируются каждым каналом;
- **Только одна колонка** (One column only) – создает только одну колонку отсчетов времени данных, которые генерируются каждым каналом;
- **Пустая колонка отсчетов времени** (Empty time column) – создает пустую колонку для отсчетов времени данных, которые генерируются каждым каналом.

Разделитель (Delimiter) содержит следующие опции:
- **Табуляция** (Tab) – использует табуляцию для разделения полей в текстовом файле;
- **Запятая** (Comma) – использует запятые для разделения полей в текстовом файле.

Описание файла (File Description) – содержит описание файла с расширением .lvm. LabVIEW добавляет текст, введенный в это текстовое окно, к заголовку файла.

Входы блок-диаграммы Экспресс-ВП имеют следующие значения (рис. 2.36):

Сигналы (Signals) – содержит один или несколько входных сигналов;

Имя файла (File Name) – задает имя файла, в который записываются данные;

Комментарий (Comment) – добавляет комментарий к каждому набору данных, записываемому в файл с расширением .lvm;

Разрешено (Enable) – разрешает или запрещает выполнение Экспресс-ВП. По умолчанию имеет состояние ИСТИНА.

Выходы блок-диаграммы Экспресс-ВП имеют следующие значения:

Выход имени файла (File Name Out) – возвращает имя файла;

Сохранение данных (Saving Data) – показывает, что Экспресс-ВП сохранил данные.

Этот Экспресс-ВП использует функциональность следующих ВП и функций: **Открыть/Создать/Заменить файл** (Open/Create/Replace File), **Записать символы в файл** (Write Characters To File), **Записать файл** (Write File), **Записать в файл табличного формата** (Write To Spreadsheet File), **Файловый диалог** (File Dialog), **Открыть файл** (Open File), **Преобразовать в файл** (Format Into File)

Считать файл результатов измерения LabVIEW (Read LabVIEW Measurement File)

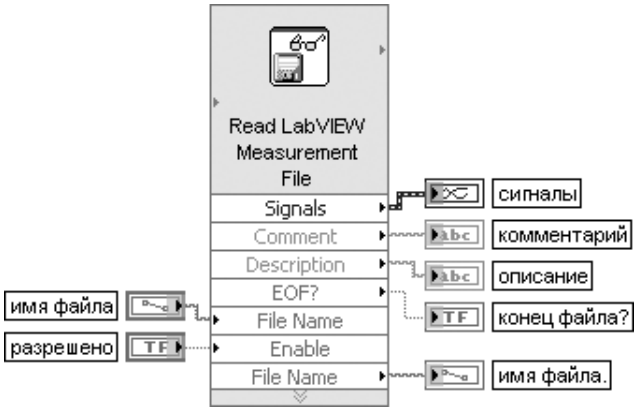


Рис. 2.38. Блок-диаграмма возможного подключения Экспресс-ВП

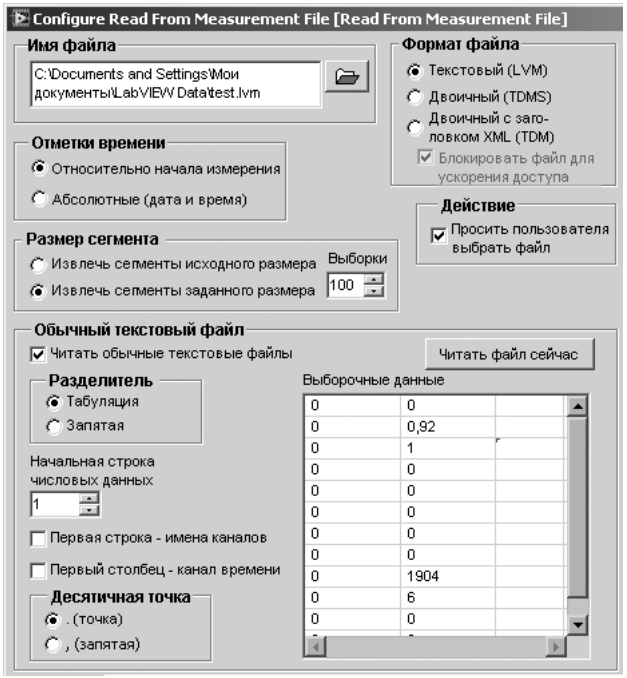


Рис. 2.39. Вид диалогового окна конфигурирования Экспресс-ВП ***Считать файл результатов измерения LabVIEW*** *(Read LabVIEW Measurement File)*

Экспресс-ВП считывает данные из файла результатов измерения LabVIEW.
Диалоговое окно данного Экспресс-ВП имеет следующие опции (рис. 2.39):
Имя файла (File name) отображает полный путь к файлу, из которого считываются данные.
Действие (Action) содержит опцию **Просить пользователя выбрать файл** (Ask user to choose file). Установка опции вызывает отображение диалогового окна, в котором пользователю предлагается выбрать файл.
Размер сегмента (Segment Size) содержит следующие опции:
- **Извлечь сегменты исходного размера** (Retrieve segments of original size) – извлекает сегменты сигнала из файла с их исходным размером;
- **Извлечь сегменты заданного размера** (Retrieve segments of specified size) – извлекает сегменты сигнала из файла, используя размер, который определяется с помощью элемента **Выборки** (Samples).

Отметки времени (Time Stamps) содержит следующие опции:
- **Относительно начала измерения** (Relative to start of measurement) – отображает числовой объект в формате часов, минут и секунд, начинающихся с нуля. Например, 100 в десятичном представлении соответствует 1:40 в относительном времени;
- **Абсолютные (дата и время)** (Absolute (date and time)) отображает числовой объект в формате времени, истекшего с 0:00 1 января 1904 года по Гринвичу.

Общий текстовый файл (Generic Text File) содержит следующие опции:
- **Читать общие текстовые файлы** (Read generic text files) – считывает данные из общих текстовых файлов;

- **Начальная строка числовых данных** (Start row of numeric data) – показывает первую строку числовых данных. Экспресс-ВП начинает чтение с этой строки. По умолчанию значение этого параметра равно 1;
- **Первая строка – имена каналов** (First row is channel names) – определяет, что имена каналов находятся в первой строке;
- **Первый столбец – канал времени** (First column is time channel) – определяет, что данные времени для каждого канала находятся в первом столбце файла данных;
- **Читать файл сейчас** (Read File Now) – импортирует данные из файла, заданного в опции **Имя файла**, в таблицу **Выборка данных**.

Входы блок-диаграммы Экспресс-ВП имеют следующие значения (рис. 2.38):
Имя файла (File Name) – задает имя файла, из которого считываются данные;
Разрешено (Enable) – разрешает или запрещает выполнение Экспресс-ВП. По умолчанию имеет состояние ИСТИНА.
Выходы блок-диаграммы Экспресс-ВП имеют следующие значения:
Сигналы (Signals) – содержит один или несколько выходных сигналов;
Имя файла (File Name) – возвращает имя файла;
Комментарий (Comment) – возвращает комментарии, добавленные к каждому набору данных в файле с расширением .lvm;
Описание (Description) – возвращает описание, находящееся в заголовке этого файла с расширением .lvm;
Конец файла (EOF?) – возвращает значение ИСТИНА, когда Экспресс-ВП достигает конца файла.
Этот Экспресс-ВП использует функциональность следующих ВП и функций: **Открыть/Создать/Заменить файл** (Open/Create/Replace File), **Считать символы из файла** (Read Characters From File), **Читать файл** (Read File), **Считать из файла табличного формата** (Read From Spreadsheet File), **Файловый диалог** (File Dialog), **Открыть файл** (Open File), **Просмотр файла** (Scan From File)

Ниже приведены блок-диаграммы и комментарии ВП **Записать двоичный файл** (Write Binary File) (рис. 2.40) и ВП **Считать двоичный файл** (Read Binary File) (рис. 2.41) из набора примеров NI Example Finder.

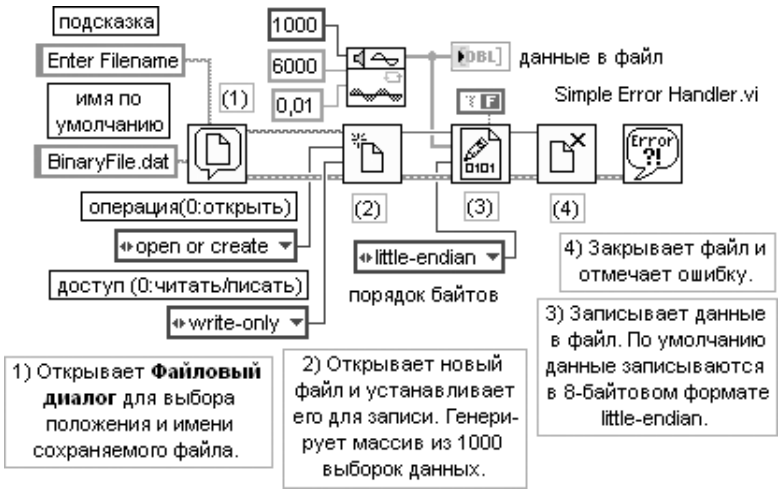


Рис. 2.40. Блок-диаграмма ВП ***Записать двоичный файл*** *(Write Binary File)*

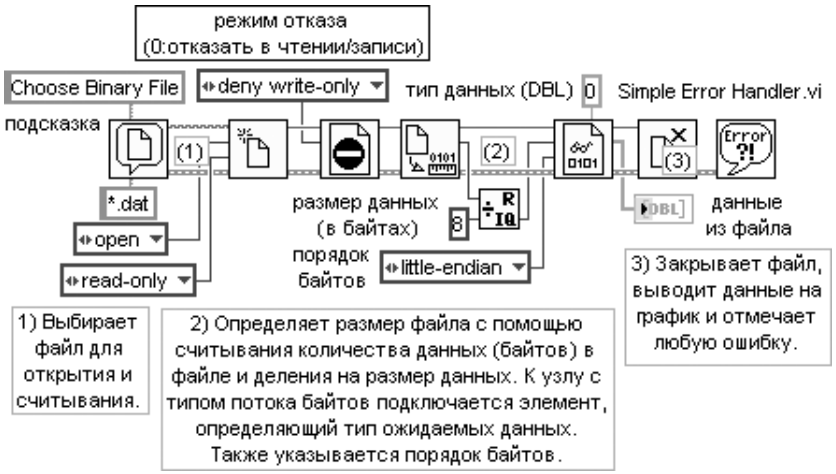


Рис. 2.41. Блок-диаграмма ВП ***Считать двоичный файл*** *(Read Binary File)*

В следующих таблицах рассмотрены функции ввода/вывода файлов из подпалитры **Дополнительные файловые функции (**Advanced File Functions**).**

Файловый диалог (File Dialog)

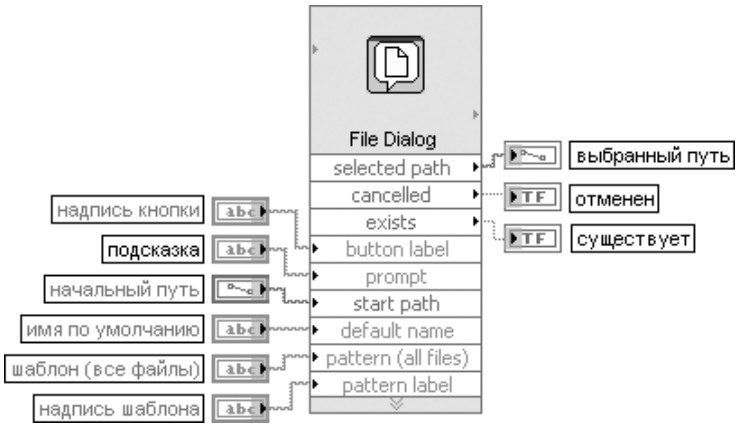


Рис. 2.42. Блок-диаграмма возможного подключения Экспресс-ВП

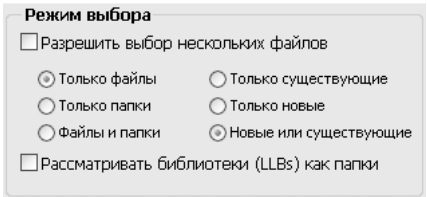


Рис. 2.43. Вид диалогового окна конфигурирования Экспресс-ВП

Экспресс-ВП отображает диалоговое окно, в котором можно определить путь к файлу или каталогу. Это диалоговое окно можно использовать для выбора уже существующего файла или каталога или для определения положения и имени нового файла или каталога.

Вид диалогового окна конфигурирования Экспресс-ВП показан на рис. 2.43.

Входы Экспресс-ВП на блок-диаграмме (рис. 2.42) имеют следующие функции.

Вход **надпись кнопки** (button label) представляет надпись, отображаемую на кнопке OK или Current Directory файлового диалогового окна. Если конфигурирование ВП позволяет пользователю выбирать каталоги, то этот вход можно использовать для определения надписи на кнопке Current Directory. Если же конфигурирование ВП не позволяет пользователю выбирать каталоги, то можно использовать этот вход для определения надписи на кнопке OK. Например, если в диалоговом окне конфигурирования выбрана опция **Только файлы** и пользователь должен выбрать существующий файл, к которому необходимо добавить данные, то на входе **надпись кнопки** можно задать строку **добавить** (Append).

Если надпись кнопки длиннее ширины кнопки, то диалоговое окно не сможет отобразить всю надпись. Например, в английской версии Windows кнопка позволяет разместить 11 символов.

Вход **подсказка** (prompt) представляет сообщение, которое появляется ниже списка файлов и каталогов в диалоговом окне. По умолчанию это пустая строка.

Вход **начальный путь** (start path) определяет путь к каталогу, содержание которого LabVIEW первоначально отображает в диалоговом окне. Если **начальный путь** достоверен, но не относится к существующему каталогу, то LabVIEW отбрасывает имя, находящееся в конце пути, до тех пор, пока укороченный путь, будет соответствовать пути каталога или станет пустым. Если **начальный путь** недостоверен или не подключен, то в диалоговом окне первоначально появится последний просмотренный каталог.

Вход **имя по умолчанию** (default name) представляет имя, которое должно появляться как начальное имя файла или каталога в диалоговом окне. По умолчанию это пустая строка.

Вход **шаблон (все файлы)** (pattern(all files)) задает шаблон имен файлов, который ограничивает перечень файлов, отображаемых в диалоговом окне. **Шаблон** не ограничивает отображаемые каталоги. Проверка соответствия шаблону в этой функции аналогична проверке шаблонов имен файлов в Windows и Linux. Если заданы символы, отличающиеся от символов (?) или (*), то функция отображает только файлы или каталоги, имена которых содержат эти символы. Использование символа (?) заменяет в проверяемом имени любой символ. Использование символа (*) заменяет последовательность из одного или большего числа символов. Так, например, шаблон, содержащий строки ***.vi;test*.llb**, установит соответствие для всех файлов с расширением **.vi** и для всех файлов, чьи имена начинаются с **test** и имеют расширение **.llb**.

Для проверки по набору шаблонов необходимо использовать символ (**;**) для разделения шаблонов. Непечатные символы, такие как пробел, табуляция и возврат каретки, воспринимаются буквально. Необходимо предотвратить использование непечатных символов, несмотря на то что они являются частью шаблона расширения. Так, например, если используется шаблон ***.html;*.doc**, диалоговое окно отобразит все файлы, которые заканчиваются **.html** и **.doc**. Если же используется ***.html;_*.doc**, будут отображены только файлы, заканчивающиеся **.html**. Здесь символ _ показывает расположение пробела.

Вход **надпись шаблона** (pattern label) определяет надпись, отображаемую в файловом диалоговом окне вслед за **шаблоном** (pattern). Если к этому входу не будет подключена строка, то по умолчанию за шаблоном пользователя будет выводиться надпись Custom Pattern. LabVIEW игнорирует этот вход, если вход шаблон остается неподключенным.

Выход **выбранный путь** (selected path) отображает полный путь к файлу или каталогу, выбираемым с помощью данного диалогового окна. При отмене диалогового окна на выходе **путь** устанавливается значение <**Не путь**> (<Not A Path>).

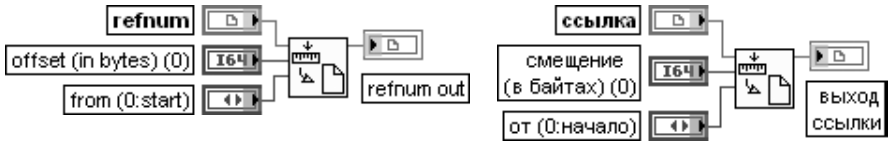
Выход **существует** (exists) устанавливается в состояние ИСТИНА, если **путь** определяет существующий файл или каталог.

Выход **отменен** (cancelled) устанавливается в состояние ИСТИНА, если файловый диалог отменен или при его использовании произошла ошибка

Get File Position / Получить положение файла

Функция возвращает положение текущей метки файла, определенного **ссылкой** (refnum), относительно его начала

Set File Position / Установить положение файла



Функция перемещает положение текущей метки файла, определенного **ссылкой** (refnum), в положение, указанное **смещением (в байтах)** (offset (in bytes)), в соответствии с режимом на входе **от** (from).

Если вход **смещение (в байтах)** подключен, **от** (from) по умолчанию устанавливается в 0 и смещение задается относительно начала файла. Если вход **смещение (в байтах)** не подключен, он по умолчанию устанавливается в 0, на входе **от** (from) по умолчанию устанавливается 2 и операция начинается с текущей метки файла.

Варианты режимов работы приведены в таблице.

0	**Начало** (start) – функция устанавливает метку файла в положение, заданное числом байтов на входе **смещение (в байтах)** (offset (in bytes)) относительно начала файла. Если на входе **от** (from) установлено нулевое значение, вход **смещение** должен быть положительным
1	**Конец** (end) – функция устанавливает метку файла в положение, заданное числом байтов на входе **смещение (в байтах)** (offset (in bytes)) относительно конца файла. Если на входе **от** (from) установлено единичное значение, вход **смещение** должен быть отрицательным
2	**Текущее** (current) – функция устанавливает метку файла в положение, заданное числом байтов на входе **смещение (в байтах)** (offset (in bytes)) относительно текущего положения метки

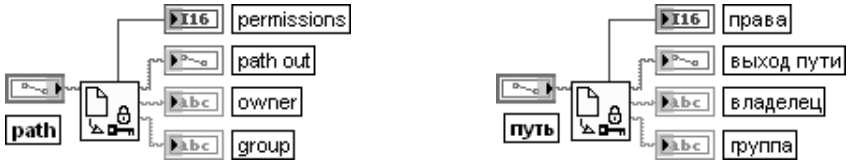
Get File Size / Получить размер файла

Функция возвращает размер файла, заданного **ссылкой** (refnum), в байтах

Set File Size / Установить размер файла

Функция устанавливает размер файла путем перемещения метки конца файла на заданное число байтов от начала файла

Get Permissions / Получить права доступа

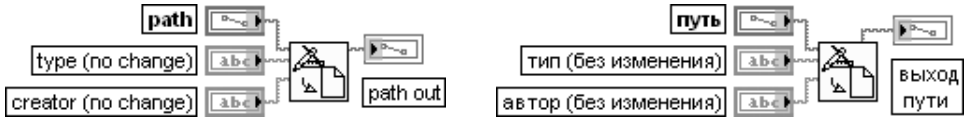


Функция возвращает текущие установки **разрешения** (permissions), **пользователя** (owner) и **группы** (group) для файла или каталога, заданных с помощью **пути** (path). **Выход пути** (path out) возвращает **путь пользователя** (path owner), содержащий текущие установки пользователя для файла или каталога после выполнения данной функции

Get Type and Creator / Получить тип и автора

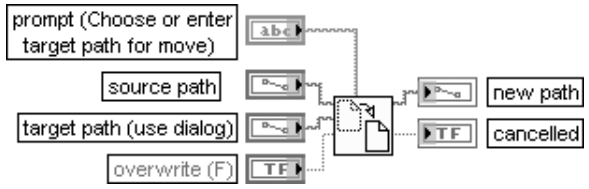
Функция считывает тип и автора файла, заданного с помощью **пути** (path). **Тип** (type) и **автор** (creator) представляют четырехсимвольные строки. Если имя указанного файла заканчивается символами, которые распознаются LabVIEW, такими как .vi для типа файла LVIN и .llb типа файла LVAR, эта функция возвращает данный тип на выходе **тип** и LBVW на выходе **автор**. Если этот тип файла не известен LabVIEW, данная функция возвращает ???? на выходах **тип** и **автор**

Set Type and Creator / Установить тип и автора



Функция устанавливает тип и автора файла, заданного с помощью **пути** (path). **Тип** (type) и **автор** (creator) представляют четырехсимвольные строки. Попытка установить тип и автора в Windows and Linux приведет к ошибке

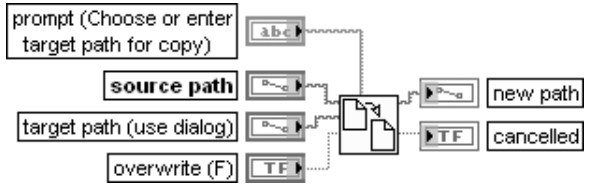
Move



Переместить

Функция перемещает файл или каталог, определенные на входе **путь источника** (source path), в место, определенное на входе **путь цели** (target path). При перемещении каталога эта функция перемещает все содержимое каталога в соответствии с вложенностью в новое место. С помощью данной функции нельзя перезаписать содержимое библиотек (LLBs)

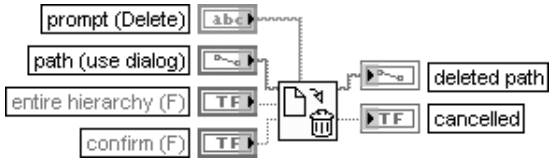
Copy



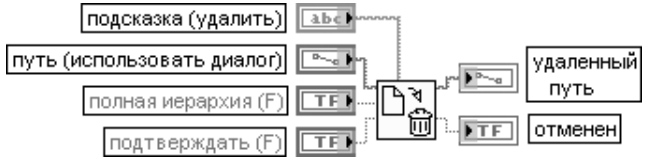
Копировать

Функция копирует файл или каталог, определенные на входе **путь источника** (source path), в место, определенное на входе **путь цели** (target path). При копировании каталога эта функция перемещает все содержимое каталога в соответствии с вложенностью в новое место

Delete

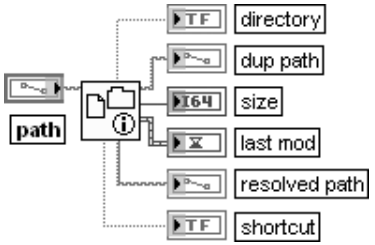


Удалить

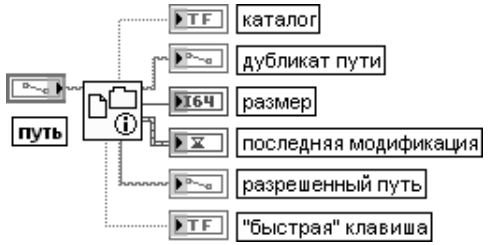


Функция удаляет файл или каталог, определенные с помощью **пути** (path). Если **путь** определяет каталог, который не является пустым, или если пользователь не имеет разрешения на запись как для файла, так и для каталога, определенных с помощью **пути** и их родительского каталога, то данная функция не удаляет каталог и возвращает ошибку

File/Directory Info



Информация о файле или каталоге



Функция возвращает информацию о файле или каталоге, определенных с помощью **пути** (path), включая **размер** (size), дату их последней модификации, информацию о том, является ли объект каталогом. Соединительная панель отображает типы данных по умолчанию для этой полиморфной функции.

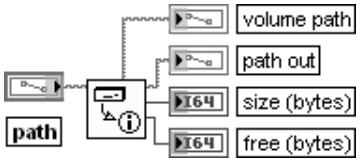
Вход **путь** (path) определяет файл или каталог, атрибуты которых должны быть определены.

Выход **каталог** (directory) устанавливается в состояние ИСТИНА, если путь указывает на каталог, и в состояние ЛОЖЬ в противном случае. Если на входе путь подключен пустой путь, то на этом выходе также выводится ИСТИНА.

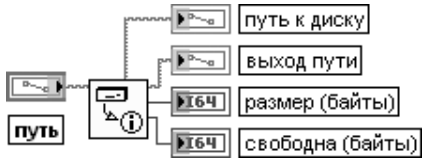
Выход **размер** (size) отображает размер файла или каталога, определенных с помощью **пути**. Если **путь** определяет каталог, то **размер** отображает число объектов в каталоге. Если **путь** является пустым путем, то **размер** отображает число драйверов в компьютере. В противном случае **размер** отображает длину в байтах определенного файла, который может быть как файлом протокола, так и файлом потока байтов.

Выход **последняя модификация** (last mod) отображает дату и время последней модификации файла или каталога. Число представляет время в секундах, прошедшее с 0:00 1 января 1904 года по Гринвичу

Get Volume Info

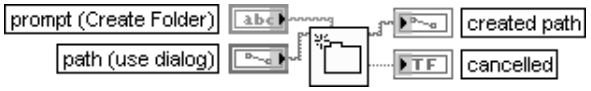


Получить информацию о томе

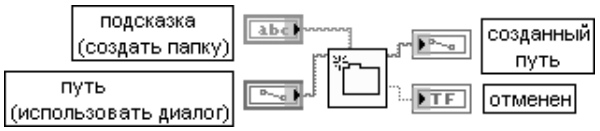


Функция возвращает информацию о диске, содержащем файл или каталог, которые заданы с помощью **пути** (path), включая общий объем памяти, предоставляемый диском, и объем свободной памяти в байтах.
Выход **путь к диску** (volume path) представляет новый путь, который определяет диск, в котором размещаются определенный файл или каталог.
Выходы **размер** (size) **свободна** (free) отображают соответственно, объем памяти, предоставляемый определенным диском, и объем свободной памяти в байтах на выбранном диске

Create Folder



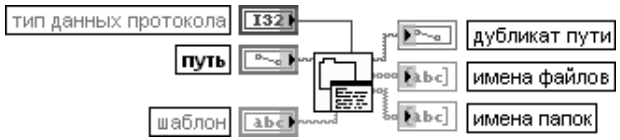
Создать папку



Функция программно создает папку, определяемую на входе **путь** (path). Если файл или папка уже существуют в заданном месте, то данная функция возвращает ошибку вместо перезаписи существующего файла или папки

List Folder

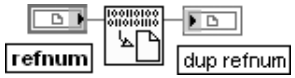
Список папки



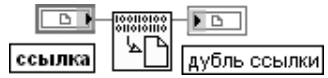
Функция возвращает два массива строк, в которых перечисляются имена всех файлов и папок, найденных по **пути** (path), имена обоих массивов фильтруются на основе **шаблона** (pattern), а массив **имен файлов** (file names) фильтруется на основе типа протокола, определенного на входе **тип протокола** (datalog type). Функция сортирует имена файлов и папок по алфавиту.
Тип протокола (datalog type) может быть любого типа и ограничивает возвращаемые **имена файлов** только файлами протокола, содержащими записи заданного типа данных. **Записи протокола данных** (Datalog records) содержат кластер отметки времени и кластер данных лицевой панели.
Вход **путь** (path) устанавливает папку, содержание которой необходимо определить. Если указана несуществующая папка, то эта функция возвращает пустые массивы имен файлов и папок и ошибку. Если путь указывает на библиотеку ВП (*.llb), выход **имена**

файлов возвращает содержимое этой библиотеки, а выход **имена папок** возвращает пустой массив

Flush File



Сбросить файл



Функция сбрасывает все буферы файла, определенного с помощью **ссылки** (refnum), на диск и корректирует вход в каталог файла, связанного со ссылкой. Данные, записываемые в файл, часто находятся в буфере до его заполнения или до закрытия файла. Эта функция заставляет операционную систему записать данные буфера в файл

Path to Array of Strings

Путь в массив строк

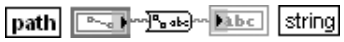
Функция преобразует **путь** (path) в **массив строк** (array of string) и индицирует тип пути. Выход **относительный** (relative) отображает значение ИСТИНА, если путь является относительным, и ЛОЖЬ, если он является абсолютным.
Выход **массив строк** содержит компоненты пути. Первый элемент представляет первый шаг в иерархии пути (имя диска для файловой системы, поддерживающей несколько дисков), а последний элемент – файл или каталог, определенные с помощью пути

Array of Strings to Path

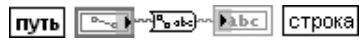
Массив строк в путь

Функция преобразует **массив строк** (array of strings) в относительный или абсолютный **путь** (path). Если в массиве строк имеется пустая строка, то положение каталога перед пустой строкой удаляется при формировании выхода пути. Такое поведение аналогично перемещению на уровень вверх в иерархии каталога

Path to Strings

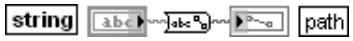


Путь в строку

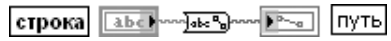


Функция преобразует **путь** (path) в **строку** (string), описывающую путь в стандартном формате платформы.
Вход **путь** может быть путем, массивом путей, кластером путей или массивом кластеров путей, которые необходимо преобразовать в строку. Если на вход **путь** подано <Not A Path>, то функция устанавливает в строке значение <Not A Path>

String To Path

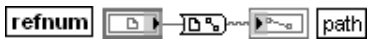


Строку в путь

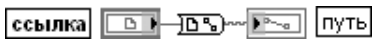


Функция преобразует **строку** (string), описывающую путь в стандартном формате платформы, в **путь** (path). Вход **строка** может быть строкой, кластером строк или массивом кластеров строк

Refnum to Path



Ссылку в путь



Функция возвращает **путь** (path), соответствующий определенной **ссылке** (refnum). Вход **ссылка** представляет ссылку открытого файла, путь к которому определяется данной функцией. Если **ссылка** не является достоверной ссылкой, то данная функция устанавливает на выходе **путь** значение <Not A Path>, означающее, что **ссылка** не связана с каким-либо открытым файлом. Ко входу **ссылка** не могут быть подключены ссылки файлов конфигурации

Path Type

Тип пути

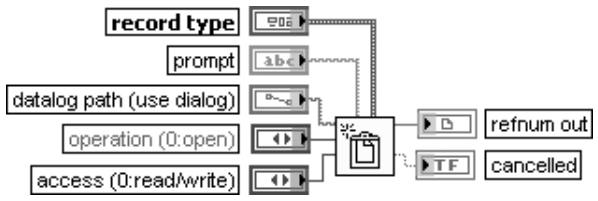
Функция возвращает **тип** определенного **пути** (path), отображая, является ли он абсолютным (0), относительным (1) путем или не является путем (2)

Файл протокола сохраняет данные в виде последовательности записей с одинаковой структурой. LabVIEW записывает каждую запись в файл как кластер, содержащий сохраняемые данные. При этом записи присваивается номер, который позволяет при необходимости осуществить к ней произвольный доступ. LabVIEW записывает запись каждый раз, когда выполняется соответствующий ВП. Пользователь не может перезаписать запись после того, как она записана в файл протокола. При считывании из файла протокола можно считать одну или несколько записей.

Файлы протокола могут создаваться как с помощью блок-диаграммы, так и при регистрации данных с лицевой панели (front panel datalogging). В последнем случае кластер записи содержит кластер отметки времени и кластер с данными элементов управления и индикаторов лицевой панели во время регистрации. Управление регистрацией, просмотром и удалением записей в интерактивном режиме осуществляется с помощью функций меню **Управление ⇒ Регистрация данных** (Operate ⇒ Data Logging). Программно зарегистрированные данные могут быть получены с помощью функций считывания данных файла протокола.

Ниже в таблице приведено описание функций ввода/вывода файлов протокола, находящихся в палитре Datalog.

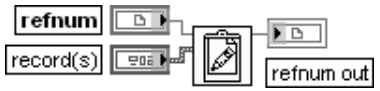
Open/Create/Replace Datalog



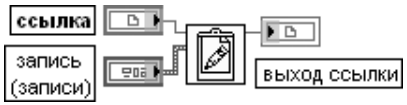
Открыть/Создать/Заменить файл протокола

Функция открывает существующий файл протокола, создает новый файл протокола или заменяет существующий файл протокола, программно или интерактивно используя файловое диалоговое окно. Пользователь может дополнительно определить **подсказку** диалога (prompt) или имя файла по умолчанию

Write Datalog

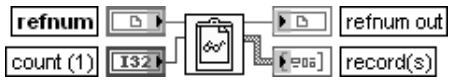


Записать в файл протокола

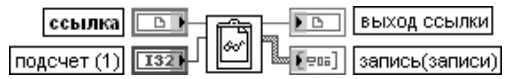


Функция записывает **запись (записи)** (record(s)) в открытый файл протокола, заданный с помощью **ссылки** (refnum). Запись начинается с конца файла.
Запись (записи) (record(s)) должна иметь тип данных, совпадающий с типом данных, заданным при открытии или создании файла или массива с таким типом данных.
В первом случае функция записывает **запись (записи)** как отдельную запись в файл протокола. При необходимости функция приводит числовые данные к представлению типа записи этого параметра. Во втором случае эта функция записывает каждую запись массива в файл протокола построчно.

Read Datalog



Считать из файла протокола



Функция считывает записи из открытого файла протокола, заданного с помощью **ссылки** (refnum), и возвращает их на выходе **запись (записи)** (record(s)). Считывание начинается с текущей позиции файла протокола.
На входе **подсчет** (count) задается количество считываемых записей. Функция возвращает на выходе **запись (записи)** количество элементов данных, установленных на входе **подсчет**, или число всех полных элементов данных при достижении конца файла. В последнем случае выдается ошибка конца файла. По умолчанию функция возвращает единичный элемент данных. Функция возвращает ошибку, если значение на входе **подсчет** отрицательно.
Если **подсчет** относится к массиву элементов и заданным типом **записи (записей)** является массив, функция автоматически возвращает кластер массивов или массив кластеров, поскольку LabVIEW не разрешает создание массива массивов

Функции **Получить (Установить) положение файла протокола** (Get (Set) Datalog Position) и **Получить (Установить) размер файла протокола** (Get (Set) Datalog Size) идентичны рассмотренным выше функциям **Получить (Установить) положение файла** (Get (Set) File Position) и **Получить (Установить) размер файла** (Get (Set) File Size).

Ниже приведены блок-диаграммы ВП **Пример записи файла протокола** (Write Datalog File Example) (рис. 2.44) и **Пример считывания файла протокола** (Read Datalog File Example) (рис. 2.45) из набора примеров NI Example Finder.

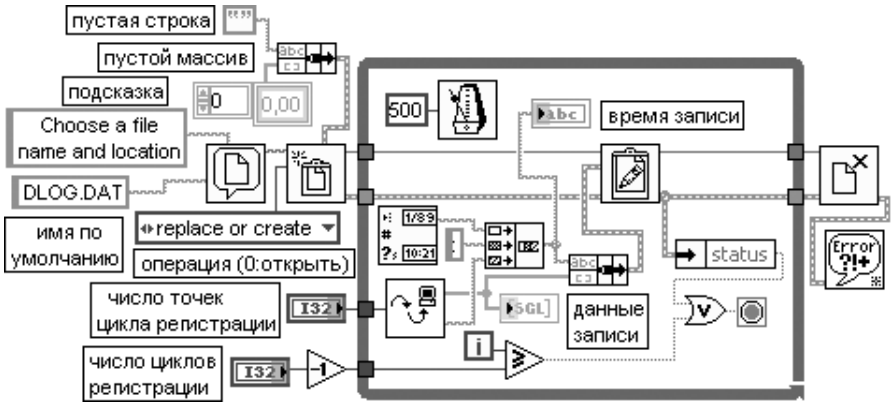


Рис. 2.44. Блок-диаграмма ВП ***Пример записи файла протокола*** *(Write Datalog File Example)*

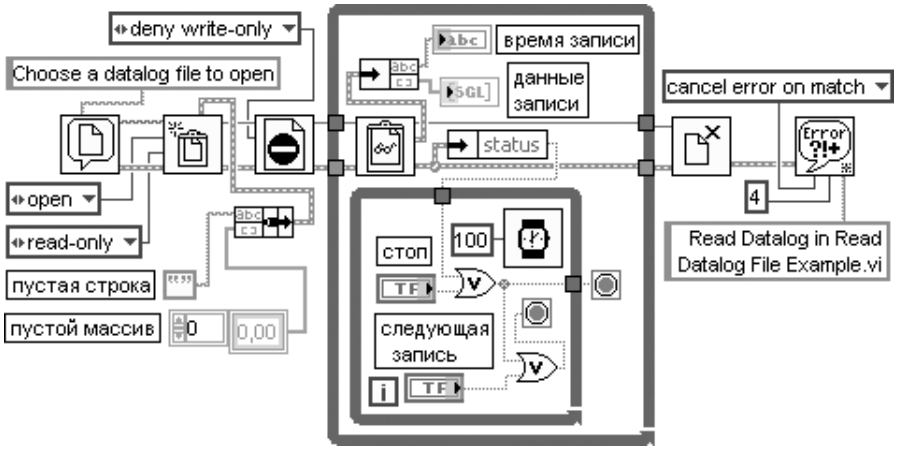
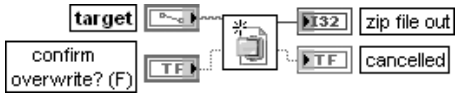


Рис. 2.45. Блок-диаграмма ВП ***Пример считывания файла протокола*** *(Read Datalog File Example)*

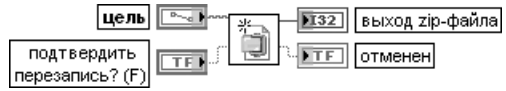
Zip-файлы используются для распространения файлов или законченных проектов LabVIEW в виде отдельного переносимого файла. Zip-файл содержит сжатые файлы, которые можно отправить другим пользователям LabVIEW.

В следующей таблице рассмотрены функции программного создания Zip-файлов из подпалитры **ВП Zip-файлов** (Zip VIs).

New Zip File



Новый Zip-файл



Функция создает новый пустой zip-файл, имеющий путь, указанный на входе **цель** (target). Новый файл удаляет и перезаписывает существующий файл или выводит подтверждающий диалог, если на входе **подтвердить перезапись?** (confirm overwrite?) установлено значение ИСТИНА.

Выход zip-файла (zip file out) возвращает открытый zip-файл. Этот выход идентичен ссылке или идентификационному номеру (ID) задачи

Add File to Zip

Добавить файл к Zip-файлу

Функция добавляет файл, заданный с помощью **пути файла источника** (source file path), к zip-файлу.

Вход **вход zip-файла** (zip file in) определяет открытый zip-файл.

Вход **путь целевого zip-файла** (destination path in zip) определяет имя файла и путь для помещения сжатого файла источника в zip-файл

Close Zip File

Закрыть Zip-файл

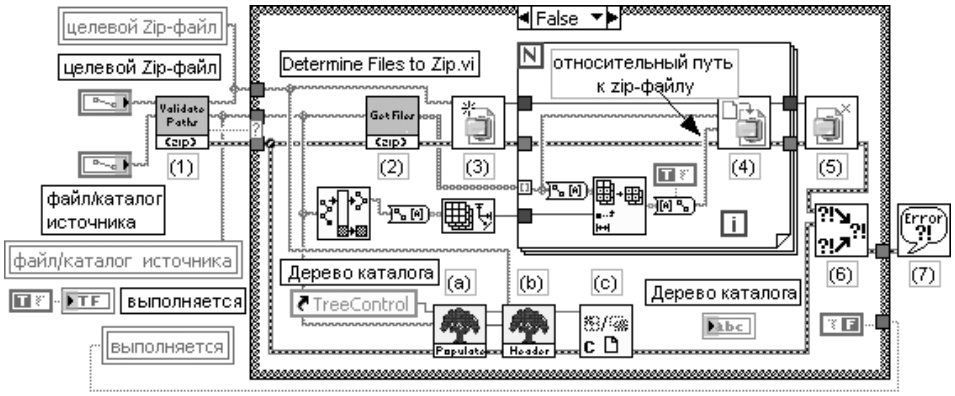
Функция закрывает zip-файл, заданный с помощью **входа zip-файла** (zip file in).

Вход **комментарии** (comments) содержит текст, предназначенный для включения в zip-файл

На рис. 2.46 и 2.47 приведены блок-диаграммы ВП **Инструмент сжатия** (Zip Tool) и подприбора этого ВП **Определить файлы для сжатия** (Determine Files to Zip) из набора примеров NI Example finder.

ВП хранения (Storage VIs) используются для считывания и записи осциллограмм и свойств осциллограмм в **двоичный файл результатов измерений** (binary measurement files (.tdm)). Файлы с расширением **.tdm** используются для обмена данными между такими программными продуктами NI, как LabVIEW и DIAdem.

ВП хранения объединяют осциллограммы и свойства осциллограмм для создания каналов. Группа каналов образует набор каналов. Файл включает набор



1) Проверяет достоверность путей к целевому Zip-файлу и к файлу/каталогу источника.
Приобнаружении недостоверности выводится подсказка пользователю с просьбой ввести корректный путь (пути).
2) Определяет файлы, которые должны быть сжаты в целевом Zip-файле.
3) Открывает ссылку к новому или существующему zip-файлу, используя путь целевого Zip-файла.
Примечание: это приводит к перезаписи существующего zip-файла.
4) Добавляет файл источника к целевому Zip-файлу.
5) Закрывает ссылку к целевому Zip-файлу.
6) Объединяет ошибки.
7) Показывает ошибки пользователю.

a) Заполняет элемент управления Дерево структурой каталога целевого Zip-файла. *
b) Создает содержимое заголовка для элемент управления Дерево.
c) Закрывает ссылку на элемент управления Дерево.
* Структура каталога, отображаемая в элементе управления Дерево, такая же, что и в файле/каталоге источника.
Пустые папки будут отображаться в элементе управления Дерево, но не будут включены в структуру каталога целевого Zip-файла.

Рис. 2.46. Блок-диаграмма ВП ***Инструмент сжатия*** *(Zip Tool)*

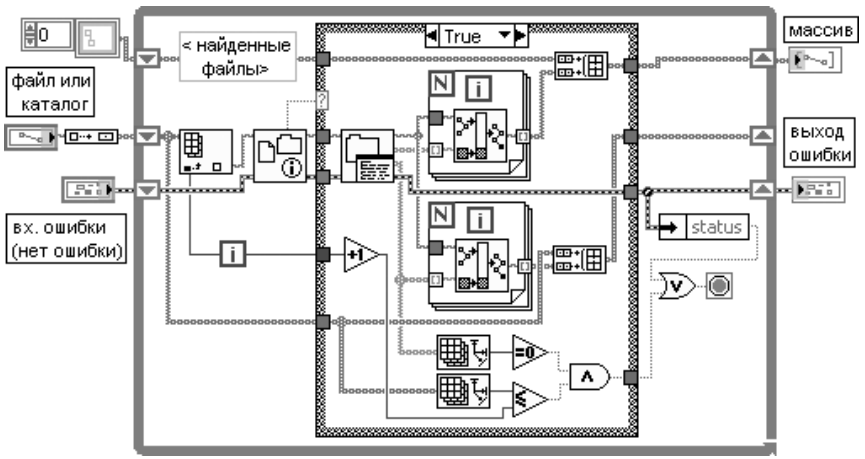
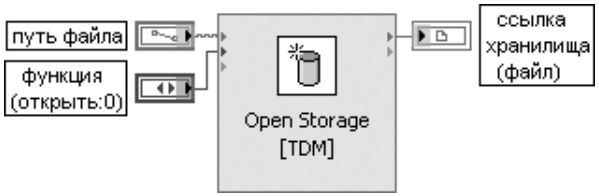


Рис. 2.47. Блок-диаграмма ВП ***Определить файлы для сжатия*** *(Determine Files to Zip)*

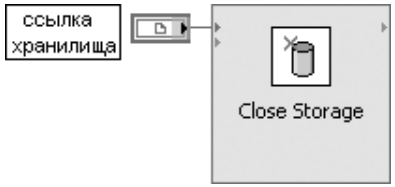
групп каналов. Если каналы сохраняются по имени, то можно быстро добавить данные к существующему каналу или получить данные из него. Помимо числовых значений, **ВП хранения** поддерживают массивы строк и массивы отметок времени. Номер ссылки представляет на блок-диаграмме файлы, группы каналов и каналы. Пользователь также может использовать **ВП хранения** для обращения к файлам с запросом для получения каналов, которые соответствуют заданным условиям.

В последующих таблицах рассмотрены параметры функций из палитры **ВП хранения** (Storage VIs).

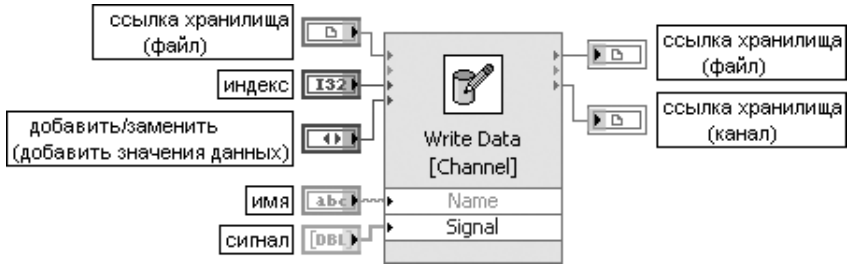
Открыть хранилище данных (Open Data Storage)



Экспресс-ВП открывает **двоичный файл результатов измерения** (binary measurement file (.tdm)) для считывания или записи. Этот Экспресс-ВП может использоваться для создания нового файла или замены существующего файла. Для закрытия ссылки к файлу после выполнения операций записи или считывания используется Экспресс-ВП
Закрыть хранилище данных (Close Data Storage)
Закрыть хранилище данных (Close Data Storage)

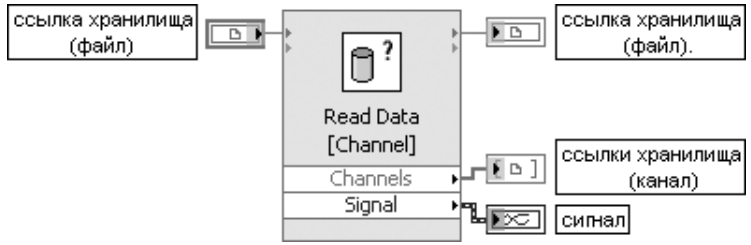


Записать данные (Write Data)



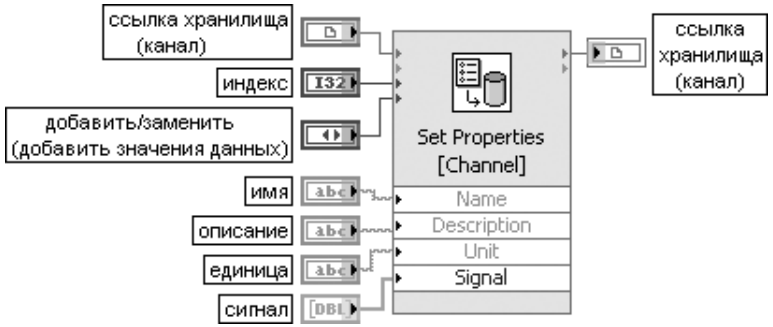
Экспресс-ВП добавляет группу каналов или канал к заданному файлу. Этот ВП также можно использовать для задания свойств добавляемых групп каналов или канала.
Диалоговое окно Экспресс-ВП имеет следующие разделы:
Установки (Settings) – позволяет выбрать **тип объекта** (Object type), добавляемого к файлу, и определить возможность добавления существующих групп каналов или канала с тем же самым именем (флажок **всегда создавать новую группу каналов/канал** (Always create new channel group / channel)). Варианты **типа объекта** включают группу каналов или канал;
Свойства (Properties) – позволяет выбрать свойства для заданного **типа объекта** из списка **доступных** (Available) и установить их как **выбранные** (Selected). Выбранные свойства появляются как выводы на блок-диаграмме. Список доступных свойств включает **имя** (Name), **описание** (Description), **единица** (Unit), **минимум** (Minimum), **максимум** (Maximum) и ряд свойств **сигнала** (Waveform);
Канал измеренных данных (Measured data channel) – позволяет установить отображение выводов на блок-диаграмме (флажок **показывать выводы для канала данных** (Show terminals for data channel) и выбрать варианты добавления или замены значений данных

Считать данные (Read Data)



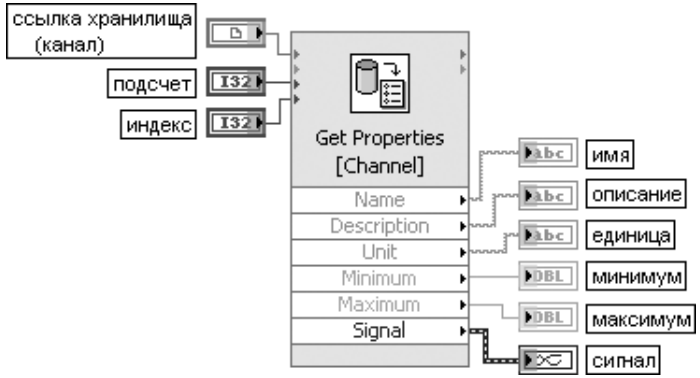
Экспресс-ВП возвращает массив ссылок, которые представляют группы каналов или каналы в файле.
Этот ВП считывает сигналы, связанные с каналом, если выбрать **Канал** (Channel) как **Тип считываемого объекта** (Object type to read) в диалоговом окне конфигурирования. Данный ВП также можно использовать для получения групп каналов или каналов, которые соответствуют условиям запроса пользователя. Для установления или получения свойств групп каналов или каналов, которые возвращаются этим ВП, необходимо использовать соответственно ВП **Установить свойства** (Set Properties) или **Получить свойства** (Get Properties).
Диалоговое окно Экспресс-ВП имеет следующие разделы:
Установки считывания (Read Settings) – позволяет выбрать **тип считываемого объекта** (Object type to read), возвращаемого при считывании. Варианты **типа объекта** включают группу каналов или канал;
Запрос (Query) – позволяет установить **свойство для сравнения** (Property to compare) в качестве условия запроса. Набор свойств идентичен по составу перечню свойств Экспресс-ВП **Записать данные**;
Канал измеренных данных (Measured data channel) – позволяет установить отображение выводов на блок-диаграмме и тип выводимых данных. Варианты вывода включают **динамический тип данных** (Dynamic data type), **массив сигналов** (Array of waveforms), **массив строк** (Array of strings), **массив отметок времени** (Array of time stamps) и массивы вещественных и целых числовых значений

Установить свойства (Set Properties)



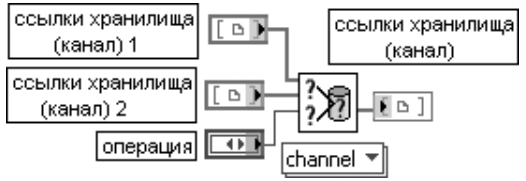
Функция определяет свойства существующего файла, канала или группы каналов. Диалоговое окно Экспресс-ВП идентично окну ЭВП **Записать данные**

Получить свойства (Get Properties)



Функция считывает значения свойств из файла, канала или группы каналов. Диалоговое окно Экспресс-ВП идентично окну ЭВП **Записать данные**

Объединить запросы (Merge Queries)



Функция объединяет результаты запроса от двух ВП **Считать данные** (Read Data). Тип данных, подключенных ко входу **ссылки хранилища** (storage refnums), определяет полиморфную реализацию используемой функции

Удалить данные (Delete Data)

Функция удаляет заданный канал или группу каналов. Тип данных, подключенных ко входу **ссылки хранилища** (storage refnums), определяет полиморфную реализацию используемой функции

На рис. 2.48 и 2.49 приведены блок-диаграммы ВП **Записать группы каналов** (Write Channel Groups) и **Считать группы каналов** (Read Channel Groups) из набора примеров NI Example finder.

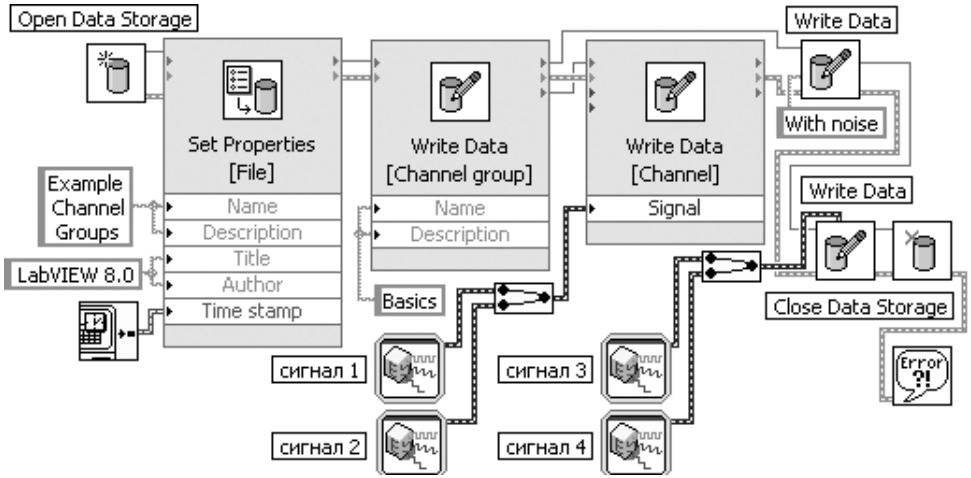
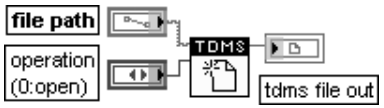


*Рис. 2.48. Блок-диаграмма ВП **Записать группы каналов** (Write Channel Groups)*

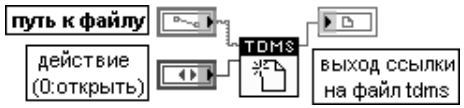
ВП **Считать группы каналов** (Read Channel Groups) циклически проверяет все содержимое файла от свойств файла до свойств каналов данных, включая свойства групп каналов. Свойства каждого из этих объектов считываются и отображаются в кластере. Данные, записанные в виде осциллограммы с динамическим типом данных, извлекаются как массивы с двойной точностью.

ВП и функции TDMS используются для чтения и записи осциллограмм и свойств осциллограмм в двоичные файлы измерений с расширением .tdms.

TDMS Open



Открыть файл TDMS



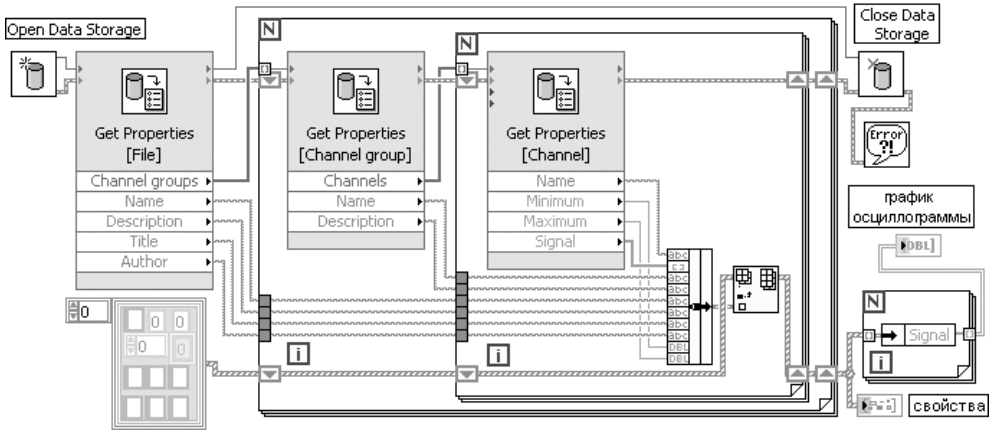
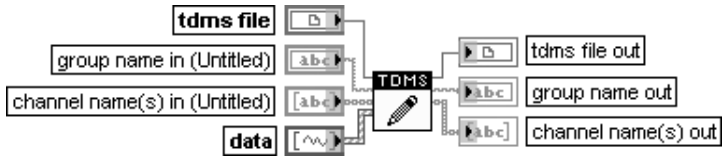


*Рис. 2.49. Блок-диаграмма ВП **Считать группы каналов** (Read Channel Groups)*

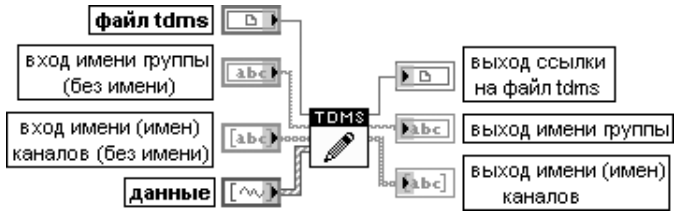
Функция открывает существующий файл с расширением .tdms для записи или считывания. С помощью этой функции также можно создать новый файл или заменить существующий. Для того чтобы закрыть ссылку к этому файлу, используется функция **Закрыть ссылку к файлу TDMS** (TDMS Close).
Вход **действие** (operation) определяет операцию, выполняемую функцией.

0	**Открыть** (open) – открывает файл с расширением .tdms для записи
1	**Открыть или создать** (open or create) – создает новый файл с расширением .tdms или открывает существующий файл .tdms для конфигурирования
2	**Создать или заменить** (create or replace) – создает новый файл с расширением .tdms или заменяет существующий файл .tdms
3	**Создать** (create) – создает новый файл с расширением .tdms
4	**Открыть (только для чтения)** (open (read-only)) – открывает версию файла .tdms только для чтения

TDMS Write

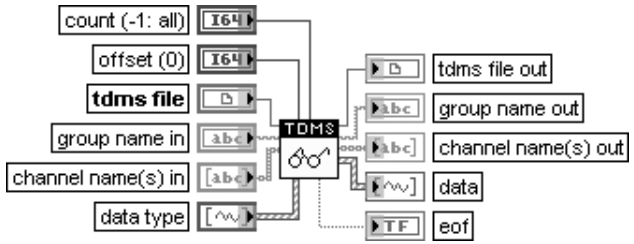


Записать в файл TDMS

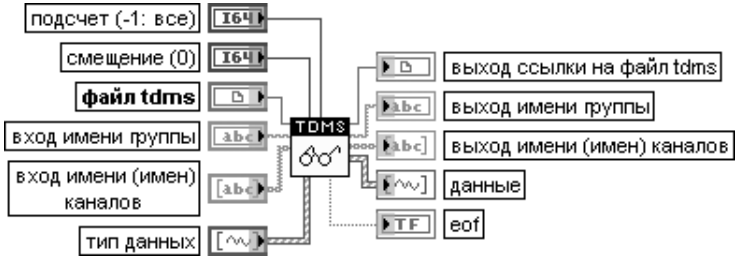


Функция производит потоковую запись в файл с расширением .tdms, заданный на входе **файл tdms** (tdms file). Набор данных, подлежащих записи, определяется значениями, установленными на входах **имя группы** (group name) и **имя (имена) каналов** (channel name(s) in)

TDMS Read



Считать из файла TDMS



Функция считывает заданный файл с расширением .tdms и возвращает **данные** в формате, указанном на входе **тип данных** (data type). С помощью входов **подсчет** (count) и **смещение** (offset) можно считать заданный фрагмент данных.

Вход **тип данных** (data type) устанавливает тип данных, которые должны быть считаны на выходе **данные** (data). Этот вход воспринимает следующие типы данных:

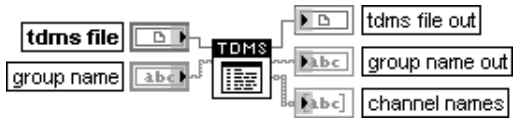
- числовой скаляр, одномерный или двумерный массив чисел
- строковый, одномерный или двумерный массив строк
- отметка времени, одномерный или двумерный массив отметок времени
- логический скаляр, одномерный или двумерный массив логических значений
- осциллограмма или одномерный массив осциллограмм
- цифровая таблица или одномерный массив цифровых таблиц

TDMS Close

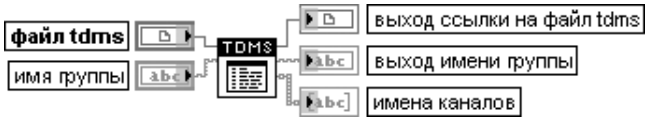
Закрыть файл TDMS

Функция закрывает ссылку к файлу с расширением .tdms

TDMS List Contents



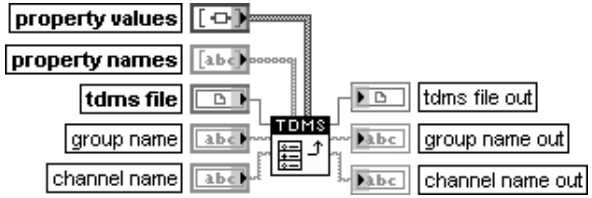
Список содержимого файла DTMS



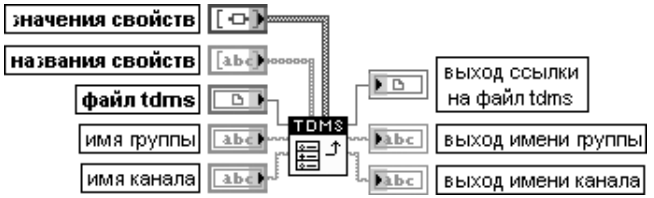
Функция предоставляет список имен групп и каналов, содержащихся в файле с расширением .tdms, заданном на входе **файл .tdms (**.tdms file).

Если вход **имя группы** (group name) подключен, то на выходе **имена групп/каналов** (group/channel names) возвращаются имена каналов заданной группы

TDMS Set Properties



Установить свойства файла DTMS

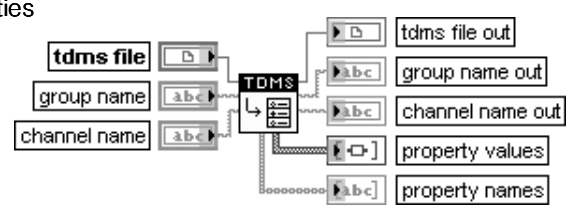


Функция устанавливает свойства заданного файла с расширением .tdms, группы каналов или канала. Свойства записываются в группу каналов или в канал при подключении и установке значений на входах **имя группы** (group name) или **имя канала** (channel name).

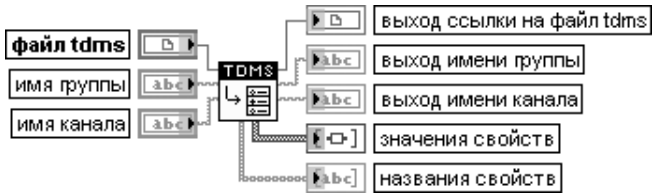
Вход **значения свойств** (property values) определяет значения свойств заданной группы каналов, канала или файла .tdms. Этот вход воспринимает следующие типы данных:

- числовой скаляр, одномерный или двумерный массив чисел
- строковый, одномерный или двумерный массив строк
- отметка времени, одномерный или двумерный массив отметок времени
- логический скаляр, одномерный или двумерный массив логических значений
- Вариант или одномерный массив данных с типом Вариант, который содержит только те типы данных, которые перечислены выше

TDMS Get Properties

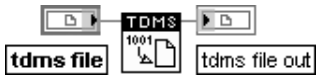


Получить свойства файла DTMS

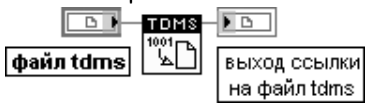


Функция возвращает свойства заданного файла с расширением .tdms, группы каналов или канала. Функция возвращает свойства группы и/или канала при подключении и установке значений на входах **имя группы** (group name) или **имя канала** (channel name)

TDMS Flush



Очистить файлы TDMS



Функция очищает системную память от всех файлов с расширением .tdms с целью защиты данных

TDMS File Viewer

Функция просмотра файла TDMS

Функция открывает файл .tdms, заданный на входе **путь к файлу** (file path), и представляет данные файла в диалоговом окне **Просмотр файла TDMS** (TDMS File Viewer). Это диалоговое окно отображает следующие элементы: **Содержимое файла** (File contents), **Свойства** (Properties), **Значения (таблица)** (Values (table)), **Аналоговые значения (график)** (Analog values (graph))

TDMS Defragment

Дефрагментировать файл TDMS

Функция дефрагментирует файл с расширением .tdms, заданный на входе **путь к файлу** (file path)

Перечень и функции файловых констант приведены в таблице.

Символ	Название
	**Путь** (Path Constant) возвращает путь, который может быть введен в рамку константы с помощью инструментов **Управление** или **Редактирование текста**. Путь также может быть указан с помощью файлового диалогового окна, вызываемого при выборе строки **Просмотреть путь** (Browse for Path..) контекстного меню константы

Символ	Название
	**Пустой путь** (Empty Path Constant) возвращает пустой путь. В отличие от константы **Не путь** (Not A Path) данная константа является действительным путем. Ее можно использовать как начальную точку для построения путей с использованием функции **Сформировать путь** (Build Path)
	**Не путь** (Not A Path) возвращает путь, имеющий значение <Не путь>. Константа с таким значением может использоваться на выходе структур и подприборов при возникновении ошибки и нежелании возвращать путь
	**Не ссылка** (Not a Refnum) возвращает ссылку, имеющую значение <Не ссылка>. Константа с таким значением может использоваться на выходе структур и подприборов при возникновении ошибки. Например, данную константу можно использовать для определения правильности ссылки перед ее передачей к следующей операции ввода/вывода файлов (см. рис.)
	**Текущий путь к ВП** (Current VI's Path) возвращает путь к файлу текущего ВП. Если ВП ни разу не был сохранен, то эта функция возвращает значение <Не путь>. Эта функция всегда возвращает текущее положение ВП. Если ВП перемещается, то возвращаемое значение изменяется
	**Библиотека ВП** (VI Library) возвращает путь к каталогу библиотеки ВП для текущей установки LabVIEW на текущем компьютере.
	**Каталог по умолчанию** (Default Directory) возвращает путь к каталогу по умолчанию. Каталогом по умолчанию является такой каталог, в котором LabVIEW автоматически сохраняет информацию, несмотря на указание других каталогов
	**Временный каталог** (Temporary Directory) возвращает путь к временному каталогу. Во временном каталоге хранится информация, которую нецелесообразно сохранять в каталоге по умолчанию
	**Каталог данных по умолчанию** (Default Data Directory) возвращает каталог, который сконфигурирован для хранения данных, формируемых ВП или функцией. Каталог данных по умолчанию может быть установлен в диалоговом окне **Опции** (Options)

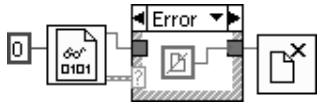


Рис. Пример применения константы пути

2.2. Дополнительные функции LabVIEW

2.2.1. Функции диалога и интерфейса пользователя

Функции диалога и интерфейса пользователя (рис. 2.50) позволяют выводить диалоговые окна с различным числом кнопок и кластеры с информацией об ошибках. При описании ВП **Очистить ошибки** (Clear Errors) приведено описание элементов такого кластера. В связи с однотипностью входного и выходного кластера ошибки в последующих таблицах с описаниями функций в большинстве случаев входные и выходные кластеры ошибок с целью экономии места не указывались, что, конечно, не отменяет необходимости их применения при построении реальных ВП.

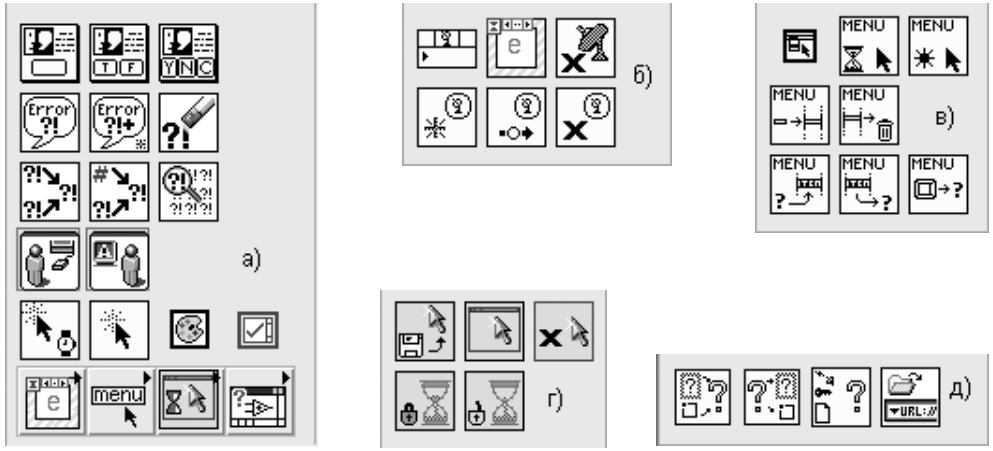


Рис. 2.50. Палитра функций диалога и интерфейса пользователя

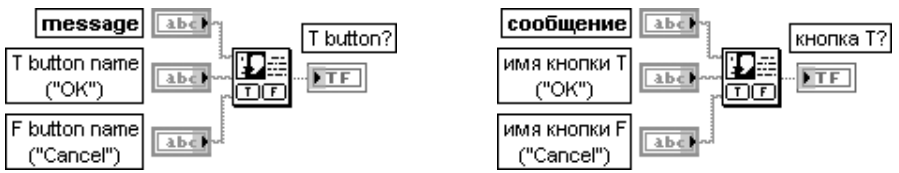
Функции диалога

Функции диалога описаны в таблице.

One Button Dialog / Диалоговое окно с одной кнопкой

Функция отображает диалоговое окно, которое содержит сообщение и одну кнопку.
Вход **сообщение** (message) содержит текст, отображаемый в диалоговом окне.
Вход **имя кнопки** (button name) содержит имя, отображаемое на кнопке диалогового окна. По умолчанию отображается OK.
На выходе **истина** (true) выводится значение ИСТИНА при нажатии на кнопку

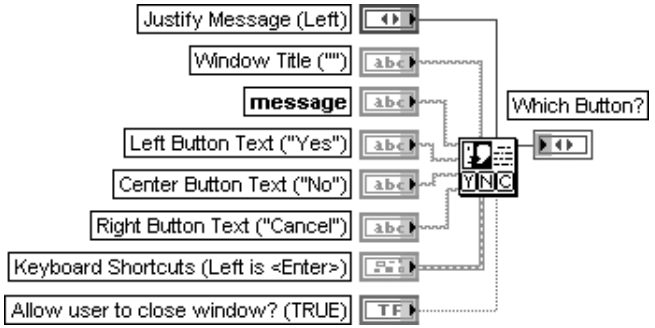
Two Button Dialog / Диалоговое окно с двумя кнопками



Функция отображает диалоговое окно, которое содержит диалоговое окно и две кнопки.
Вход **сообщение** (message) содержит текст, отображаемый в диалоговом окне.
Вход **имя кнопки T** (T button name) определяет имя, отображаемое на одной из кнопок диалогового окна. По умолчанию отображается OK.
Вход **имя кнопки F** (F button name) определяет имя, отображаемое на одной из кнопок диалогового окна. По умолчанию отображается **отмена** (Cancel).

Выход **кнопка T ?** (T button?) возвращает значение ИСТИНА при нажатии кнопки с названием **имя кнопки T**. При нажатии кнопки с названием **имя кнопки F** на этом выходе возвращается значение ЛОЖЬ

Three Button Dialog



Диалоговое окно с тремя кнопками

ВП отображает диалоговое окно, содержащее сообщение и три кнопки.
Вход **выравнивание сообщения** (Justify Message) устанавливает выравнивание отображаемого текста **влево** (Left), **по центру** (Center) или **вправо** (Right).
Вход **заголовок окна** (Window Title) представляет текст, отображаемый в строке заголовка диалогового окна.
Вход **сообщение** (message) содержит текст, отображаемый в диалоговом окне.
Вход **текст левой кнопки** (Left Button Text) определяет текст, отображаемый на левой кнопке. По умолчанию отображается **Да** (Yes).
Вход **текст центральной кнопки** (Center Button Text) определяет текст, отображаемый на центральной кнопке. По умолчанию отображается **Нет** (No).
Вход **текст правой кнопки** (Right Button Text) определяет текст, отображаемый на правой кнопке. По умолчанию отображается **Отмена** (Cancel).
Вход **клавиши быстрого выбора** (Keyboard Shortcuts) определяет клавиши быстрого выбора («горячие» клавиши) для каждой из кнопок диалогового окна. Например, можно определить клавишу <F1> для кнопки **Помощь** (Help) диалогового окна. По умолчанию определена только клавиша <**Ввод**>(<Enter>) для левой кнопки. Варианты выбора клавиш для каждой кнопки идентичны и включают следующие пункты:

- При установке кнопки Control в состояние ИСТИНА «горячая» клавиша формируется с помощью клавиши <Ctrl>;
- При установке кнопки Shift в состояние ИСТИНА «горячая» клавиша формируется с помощью клавиши <Shift>;

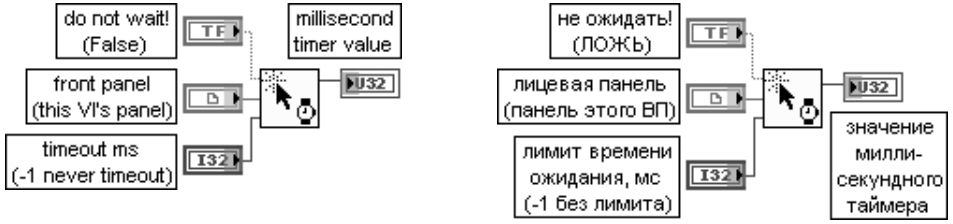
- Строковый элемент управления **клавиша** (Key) должен содержать имя «горячей» клавиши. Имя в строке **клавиша** должно соответствовать имени клавиши, выбранной в диалоговом окне **Управление с клавиатуры** (Key Navigation). Данное диалоговое окно вызывается с помощью выбора из контекстного меню элемента на лицевой панели (в данном случае кластера) пункта **Дополнительно ⇒ Управление с клавиатуры...** (Advanced ⇒ Key Navigation...).

Если вход **разрешить пользователю закрыть окно?** (Allow user to close window?) установлен в состояние ИСТИНА (по умолчанию), то окно операционной системы закрывает кнопки, появляющиеся в диалоговом окне, и пользователь может закрыть диалоговое окно без нажатия на какую-либо клавишу. В большинстве операционных систем окно закрывает клавиша, появляющаяся в правом верхнем углу окна.

Выход **Какая кнопка** (Which Button?) показывает название нажатой клавиши.

0	Left Button – нажата левая клавиша
1	Center Button – нажата центральная клавиша
2	Right Button – нажата правая клавиша
3	Window Close – пользователь закрыл окно, не нажав ни одну из клавиш

Wait For Front Panel Activity / **Ожидать активность лицевой панели**



Функция останавливает выполнение ВП до обнаружения активности лицевой панели.

Если на входе **не ожидать!** (do not wait!) установлено состояние ИСТИНА, то ВП работает без остановки выполнения.

Вход **лицевая панель** (front panel**)** является ссылкой к ВП, активность лицевой панели которого необходимо проверять. К этому входу может быть подключен ВП, лицевая панель или ссылка элемента управления. При подключении ссылки элемента управления функция контролирует активность лицевой панели, содержащей этот элемент. Если ссылка не определена, то функция контролирует активность лицевой панели ВП, в котором эта функция размещена.

Ссылка к ВП или объекту должна относиться к локальной среде LabVIEW. Не допускается ссылка к ВП или объекту удаленной среды LabVIEW.

Вход **лимит времени ожидания**, **мс** (timeout ms) определяет число миллисекунд, которое функция должна ожидать перед тем, как разрешить ВП продолжить выполнение. По умолчанию это значение равно –1, что означает отсутствие лимита времени ожидания.

Выход **значение миллисекундного таймера** (millisecond timer value) возвращает значение миллисекундного таймера.

Данная функция аналогична функции **События** (Occurrences). Функцию целесообразно использовать для блокирования выполнения диаграммы до момента изменения пользователем значений объектов лицевой панели. Использование данной функции устраняет необходимость непрерывного опроса лицевой панели с целью обнаружения изменения значений ее объектов.

Непрерывный опрос приводит к неэкономичному расходу системных ресурсов до момента взаимодействия пользователя с лицевой панелью. При использовании данной функции цикл работает только два раза: первый раз – при вызове функции, а второй – при обращении пользователя к лицевой панели.

Блок-диаграмма примера на рис. 2.51 показывает, как функция **Ожидать активность лицевой панели** формирует паузу в работе ВП до момента ввода имени или пароля или нажатия кнопки OK.

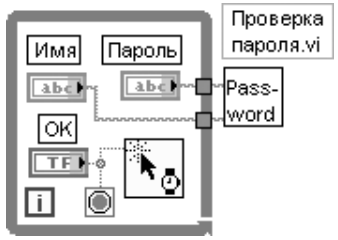


Рис. 2.51. Блок-диаграмма ВП ввода имени и пароля

Кнопка OK, подключенная к терминалу условия структуры Цикл по условию, позволяет передать имя и пароль на вход ВП **Проверка пароля** и в то же время управляет входом **не ожидать!** функции **Ожидать активность лицевой панели**

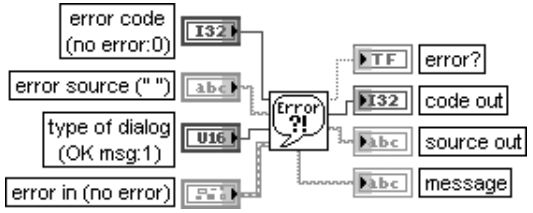
Generate Front Panel Activity / **Генерировать активность лицевой панели**

Функция программно генерирует активность лицевой панели, которая приводит к продолжению выполнения любого ВП, остановленного функцией **Ожидать активность лицевой панели** (Wait on Front Panel Activity). На лицевой панели при этом не происходит каких-либо изменений

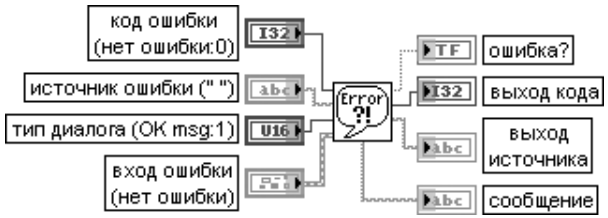
Функции обработки ошибок

В следующей таблице рассмотрены функции обработки ошибок.

Simple Error Handler



Простой обработчик ошибки



ВП показывает источник происхождения ошибки. При появлении ошибки ВП возвращает описание ошибки и дополнительно отображает диалоговое окно. Данный ВП при выполнении вызывает ВП **Общий обработчик ошибки** (General Error Handler) и имеет сходную с ним базовую функциональность, но с меньшим набором опций.

Вход **код ошибки** (error code) представляет числовой код ошибки. Если **вход ошибки** (error in) показывает ошибку, то ВП игнорирует **код ошибки**. В противном случае ВП проверяет ее. Ненулевое значение означает ошибку. К этому входу может быть подключена **кольцевая константа ошибки** (error ring constant) (раздел 2.1).

Вход **источник ошибки** (error source) представляет дополнительную строку, с помощью которой можно описать источник ошибки с принятым **кодом ошибки** (error code).

Вход **тип диалога** (type of dialog) определяет тип отображаемого диалогового окна, если таковое выводится. Независимо от этого значения ВП выводит информацию об ошибке и **сообщение** (message), описывающее ошибку. Варианты диалога приведены в таблице.

0	**Нет диалога** (No dialog) – определяет отсутствие диалогового окна. Такой вариант полезен при программном управлении обработкой ошибок
1	**Сообщение OK** (OK message) (по умолчанию) – выводится диалоговое окно с одной кнопкой OK. После подтверждения пользователем сообщения в диалоговом окне ВП возвращает управление основному ВП
2	**Сообщение «продолжить» или «остановить»** (Continue or stop message) – выводится диалоговое окно с кнопками **продолжить** (Continue) или **остановить** (Stop), которые пользователь может использовать для продолжения или остановки ВП. При выборе пользователем кнопки **остановить** ВП вызывает одноименную функцию для остановки выполнения
3	**Сообщение OK с предупреждением** (OK message with warnings) – выводится диалоговое окно с каким-либо предупреждением и одной кнопкой OK. После подтверждения пользователем сообщения в диалоговом окне ВП возвращает управление основному ВП
4	**Сообщение с предупреждением и с кнопками «продолжить» или «остановить»** (Continue or stop message with warnings) – выводится диалоговое окно с каким-либо предупреждением и кнопками, которые пользователь может использовать для продолжения или остановки ВП

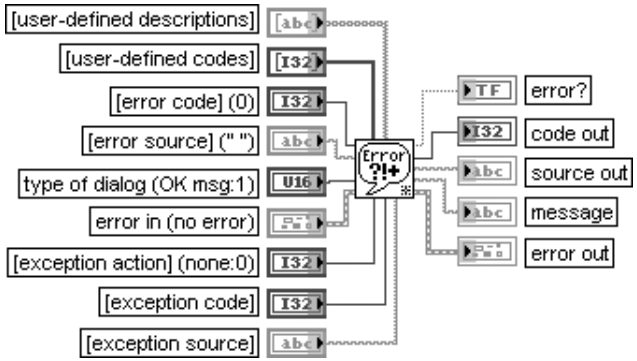
Выход **ошибка?** (error?) показывает наличие ошибки.

Выход кода (code out) выводит код ошибки, определенной на **входе ошибки** (error in) или на входе **код ошибки** (error code).

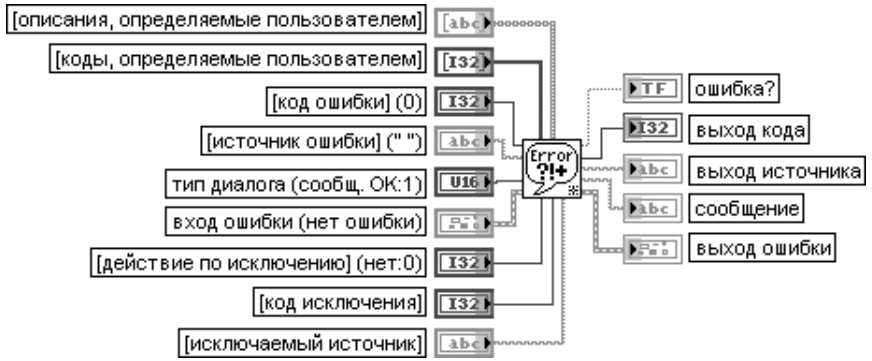
Выход источника (source out) показывает источник ошибки. Строка **выход источника** является более содержательной, чем строка **источник** в кластере **вход ошибки** (error in).

Выход **сообщение** (message) выводит код произошедшей ошибки, источник ошибки и описание ошибки

General Error Handler

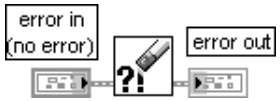


Общий обработчик ошибки

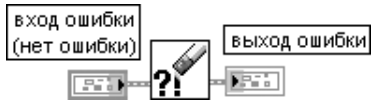


ВП показывает источник возникновения ошибки. Если ошибка возникла, то этот ВП возвращает описание ошибки и дополнительно отображает диалоговое окно. В связи со специфичностью ВП описание его функциональности ограничено переводом надписей входных и выходных терминалов

Clear Errors



Очистить ошибки

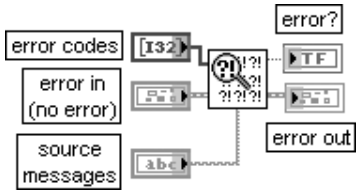


ВП сбрасывает (очищает) следующие элементы кластера **выход ошибки** (error out): **статус** (status) ошибки устанавливается в состояние **нет ошибки** (no error), **код** (code) устанавливается в 0 и **источник** (source) задается пустой строкой. Данный ВП целесообразно использовать при необходимости игнорировать ошибки. Он фактически содержит разъединенные кластеры входной и выходной ошибки.

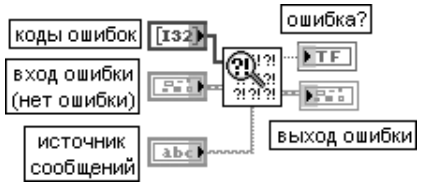
Кластер **вход ошибки** (error in) содержит аналогичные элементы, описывающие условия появления ошибок. Состав элементов и их назначение приведены в таблице.

Статус (status) имеет состояние ИСТИНА (X), если перед выполнением данного ВП произошла ошибка, и состояние ЛОЖЬ при отсутствии ошибки на входе. По умолчанию на входе установлено значение ЛОЖЬ
Код (code) является кодом ошибки или предупреждения. По умолчанию значение входа равно 0. Если на входе **статус** установлено значение ИСТИНА, то **код** является ненулевым **кодом ошибки** (error code). Если **статус** имеет состояние ЛОЖЬ, то **код** имеет значение 0 или **код предупреждения** (warning code)
Источник (source) описывает источник ошибки или предупреждения и является в большинстве случаев именем ВП или функции, которые порождают ошибку. По умолчанию данная строка является пустой

Find First Error

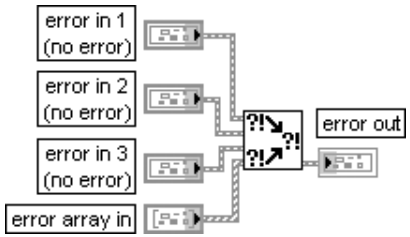


Найти первую ошибку

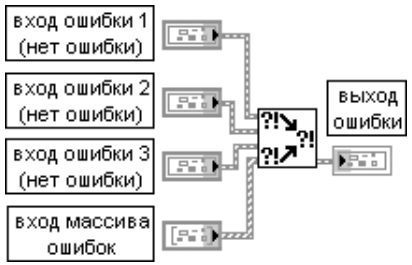


ВП проверяет **статус ошибки** (error status) одной или более функций или подприборов (subVI), которые порождают на выходе числовой код ошибки.
Вход **коды ошибок** (error codes) представляет массив числовых кодов ошибок, собранных из локальных подпрограмм или функций. Если на входе кластера **вход ошибки** (error in) ошибки отсутствуют, то ВП проверяет данные коды в порядке возрастания ненулевых значений. Если ВП находит ненулевое значение, то **выход ошибки** (error out) отражает статус ошибки этого входа.
Вход **источник сообщений** (source messages) содержит источник сообщений, которые должны появляться в кластере **выход ошибки**, если ВП находит ошибку на входе **коды ошибок**. Использование этого входа носит дополнительный характер.
Выход **ошибка?** (error?) устанавливается в состояние ИСТИНА, если кластер **вход ошибки** или какой-либо код на входе **коды ошибок** отражают ошибку

Merge Errors

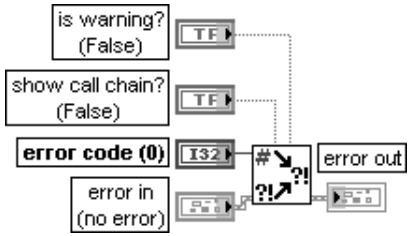


Объединить ошибки



ВП объединяет кластеры ошибок ввода/вывода различных функций.
Этот ВП сначала просматривает ошибки на входах **вход ошибки 1** (error in 1), **вход ошибки 2** (error in 2) и **вход ошибки 3** (error in 3), затем на **входе массива ошибок** (error array in) и сообщает о первой найденной ошибке. Если ВП не находит ошибку, то он просматривает предупреждения и возвращает первое найденное предупреждение. Если ВП не находит предупреждение, то он возвращает сообщение об отсутствии ошибки

Error Cluster From Error Code



Кластер ошибки из кода ошибки

ВП преобразует коды ошибки или предупреждения в кластер ошибки. Этот ВП полезен при приеме возвращаемого значения после вызова DLL или при возврате **кодов ошибок, определяемых пользователем** (user-defined error codes).
Если на входе **есть предупреждение?** (is warning?) установлено значение ИСТИНА, то элемент **статус** (status) в кластере **выход ошибки** (error out) возвращает значение ЛОЖЬ для индикации поступления предупреждения. По умолчанию на этом входе установлено значение ЛОЖЬ.
Если на входе **показывать цепь вызова?** (show call chain?) установлено значение ИСТИНА, то элемент **источник** (source) включает цепь вызовов из ВП, который порождает ошибку или предупреждение к ВП верхнего уровня. По умолчанию на этом входе установлено состояние ЛОЖЬ, которое указывает на включение только вызывающего ВП. Данный ВП использует функцию **Цепь вызова** (Call Chain) для получения цепи вызовов.
Вход **код ошибки** (error code) передает код, который пользователь хочет преобразовать в кластер ошибки. По умолчанию его значение равно 0, что показывает отсутствие ошибки

Экспресс-ВП палитры

В состав палитры функций диалога и интерфейса пользователя входят Экспресс-ВП **Подсказка пользователю для ввода** (Prompt User for Input) и **Отображение сообщения пользователю** (Display Message to User).

Подсказка пользователю для ввода (Prompt User for Input)

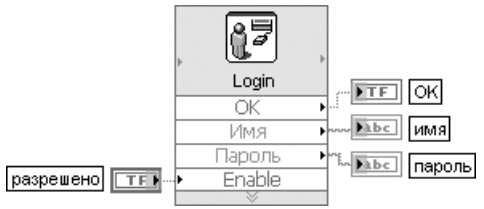


Рис. 2.52. Блок-диаграмма возможного подключения Экспресс-ВП

Экспресс-ВП **Подсказка пользователю для ввода** (Prompt User for Input) отображает стандартное диалоговое окно, которое подсказывает пользователям ввести такую информацию, как имя пользователя и пароль (рис. 2.52).
Диалоговое окно данного Экспресс-ВП имеет следующие опции (рис. 2.53).
Отображаемое сообщение (Message to Display) – содержит текст, отображаемый в диалоговом окне.

Рис. 2.53. Вид диалогового окна конфигурирования Экспресс-ВП ***Подсказка пользователю для ввода*** *(Prompt User for Input)*

Отображаемые кнопки (Buttons to Display) – содержит следующие опции:
- **Имя первой кнопки** (First button name) – определяет текст, который появляется на первой кнопке. По умолчанию на первую кнопку выводится текст **OK**;
- **Имя второй кнопки** (Second button name) – определяет текст, который появляется на второй кнопке. По умолчанию на вторую кнопку выводится текст **Cancel**. Эта опция доступна только при установке отметки **Отображать вторую кнопку** (Display second button) в соответствующем окне;
- **Отображать вторую кнопку** (Display second button) – определяет возможность отображения второй кнопки в диалоговом окне.

Входы (Inputs) – определяет имя и тип данных элементов управления, которые появляются в диалоговом окне. **Имя входа** (Input Name) определяет имя элемента управления и инструктирует пользователей – что вводить в этот элемент.

Тип входных данных (Input Data Type) – определяет тип элементов управления, которые используются в диалоговом окне. Предусмотрен выбор следующих типов: числовой, окно выбора или окно ввода текста.

Название окна (Window Title) – содержит текст, отображаемый в полосе названия диалогового окна.

Этот Экспресс-ВП использует функции **Диалоговое окно с одной кнопкой** (One Button Dialog) и **Диалоговое окно с двумя кнопками** (Two Button Dialog)

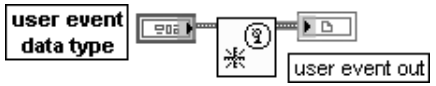
Экспресс-ВП **Отображение сообщения пользователю** (Display Message to User) отображает стандартное диалоговое окно, которое содержит предупреждение или сообщение для пользователя. Его функциональность близка к функциональности рассмотренного выше Экспресс-ВП **Подсказка пользователю для ввода**.

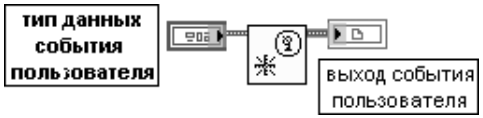
Функции подпалитры Событие

Использование функций и узлов подпалитры **Событие** (рис. 2.50б) позволяет динамически регистрировать события и создавать события пользователя.

Описание функций и узлов подпалитры **Событие** приведено в таблице.

Create User Event — Создать событие пользователя



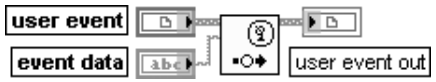


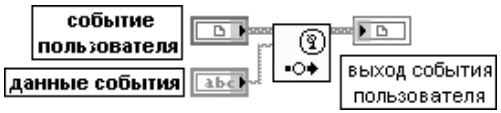
Функция возвращает ссылку к событию пользователя. LabVIEW использует **тип данных события пользователя** (user event data type) подключенного входа для определения имени события и типа данных события. Подключение **выхода события пользователя** (user event out) к узлу **Регистрация событий** позволяет выполнить регистрацию события. Подключение этого же выхода к функции **Генерировать событие пользователя** позволяет отправить событие и связанные с ним данные всем структурам **Событие**, зарегистрированным для события.

Вход **тип данных события пользователя** (user event data type) представляет кластер элементов или отдельный элемент, тип данных и имя которых определяют тип данных и имя события пользователя. При подключении кластера LabVIEW использует имя кластера как имя события пользователя. Имена и типы данных элементов кластера определяют данные события, переносимые событием пользователя. При подключении ко входу отдельного элемента данных LabVIEW использует имя элемента как имя события пользователя, а также и как имя данных, переносимых событием пользователя, которые имеют такой же тип данных, как и элемент. Действительное значение, подключенное к функции, не имеет значения, поскольку LabVIEW использует только имя и тип данных для определения события пользователя.

Выход события пользователя (user event out) возвращает ссылку события пользователя строгого типа, которая включает имя события пользователя и тип данных

Generate User Event — Генерировать событие пользователя





Функция передает **событие пользователя** (user event), подключенное к одноименному входу, и отправляет событие пользователя и связанные с ним данные каждой структуре **Событие**, зарегистрированной для обработки события.

Вход **событие пользователя** (user event) содержит ссылку к событию, созданную функцией **Создать событие пользователя** (Create User Event).

Вход **данные пользователя** (event data) содержит тип данных, определенных на входе **тип данных события пользователя** (user event data type) функции **Создать событие пользователя**.

Выход события пользователя (user event out) возвращает ссылку к событию пользователя строгого типа

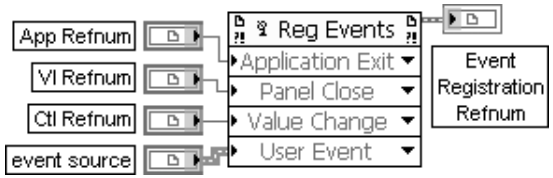
Unregister For Events — Отменить регистрацию событий

Функция отменяет регистрацию всех событий, связанных со ссылкой регистрации события. Структуры **Событие**, которые используют эту ссылку регистрации события, не могут далее принимать какие-либо динамические события. National Instruments рекомендует отменять регистрацию событий при отсутствии необходимости в их обработке. Если пользователь не отменил регистрацию событий, то LabVIEW продолжает генерировать и ставить в очередь события в течение выполнения ВП, даже если ни одна из структур **Событие** не находится в состоянии ожидания их обработки. Это приводит к захвату памяти и может привести к зависанию ВП при разрешении блокирования лицевой панели для событий

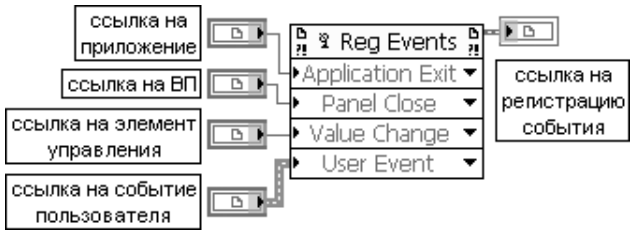
Destroy User Event / Отменить событие пользователя

Функция удаляет ссылку к событию пользователя с помощью уничтожения связанного с ней номера ссылки пользователя. Никакие структуры **Событие**, зарегистрированные для этого события пользователя, не могут после этого принимать событие

Register For Events



Регистрация событий



Функция динамически регистрирует события. События, которые можно регистрировать, зависят от типа ссылки, которая подключена к каждому входу **источник события** (event source). Выход **ссылка на регистрацию события** (event reg refnum out) следует подключать к структуре **Событие** или к другому узлу **Регистрация событий**.
Вход **ссылка на регистрацию события** (event registration refnum) содержит ссылку на существующую регистрацию события, созданную аналогичным узлом **Регистрация событий**.
Вход **источник события 1..n** (event source 1..n) может быть ссылкой на приложение, на ВП, на элемент управления или на событие пользователя (рисунок). Ссылки должны быть адресованы только к локальным объектам.
Выход **ссылка на регистрацию события** (event reg refnum out) возвращает ссылку на новую или существующую регистрацию события.

Контекстное меню каждого пункта позволяет выбрать регистрируемое событие для подключенного источника события, которым может быть приложение, ВП, элемент управления или событие пользователя. Терминал выхода **ссылка на регистрацию события** узла **Регистрация событий** может быть подключена к терминалам динамических событий на границе структуры **Событие**, к узлу **Отменить регистрацию событий** (Unregister For Events) или к верхнему левому входу другого узла **Регистрация событий**. В последнем случае узел модифицирует связанную со ссылкой информацию о существующей регистрации вместо повторной регистрации события. Зарегистрированные события остаются таковыми до явной отмены регистрации, до завершения выполнения ВП, зарегистрировавшего события, или до прерывания выполнения ВП. Если ВП, который зарегистрировал события, являлся подприбором, то события являются незарегистрированными, когда ВП верхнего уровня для этого подприбора завершает выполнение или прерывается. LabVIEW не содержит ссылку на регистрацию события в палитре элементов управления, поскольку такая ссылка является ссылкой строгого типа и общей версии для нее не существует. Ссылка на регистрацию события может быть создана путем конфигурирования узла **Регистрация событий** или создания элемента управления или индикатора с помощью контекстного меню узла

Event Structure / Структура Событие

Структура **Событие** (Event Structure) имеет одну или более поддиаграмм или вариантов событий, из которых только один выполняется при обращении к структуре. Структура Событие ожидает наступления события на лицевой панели, после чего выполняет соответствующий вариант с целью обработки события. С помощью контекстного меню структуры можно добавить новые варианты событий или определить вид обрабатываемого события. Подключение значения к терминалу **лимит времени ожидания**, находящемуся в левом верхнем углу структуры, позволяет задать лимит времени ожидания события структурой в миллисекундах. По умолчанию значение на входе этого терминала равно –1, что соответствует неограниченному времени ожидания.

В качестве примера использования функций подпалитры **Событие** на рис. 2.54 приведена блок-диаграмма модернизированного ВП **Динамическая проверка ВП** (Dynamically Monitor VI's) из библиотеки dynamicevents.llb набора примеров NI Example Finder. ВП позволяет динамически отслеживать список выполняющихся ВП с открытой лицевой панелью.

Такая задача решается в данном ВП путем создания двух событий пользователя **Добавить ВП** и **Удалить ВП** с помощью функции **Создать событие пользователя**. Созданные события регистрируются по ссылке функцией **Регистрация событий**. Также по массиву ссылок регистрируется событие **Панель закрыта?** (Panel Close?) для соответствующих ВП и по ссылке на логический элемент управления **стоп** регистрируется событие **Изменение значения** (Value Change) для этого элемента.

При выполнении структуры **Цикл по условию** с периодом 100 мс на выходе **Узла свойства** для класса **Приложение**, в котором установлено свойство **Все ВП**

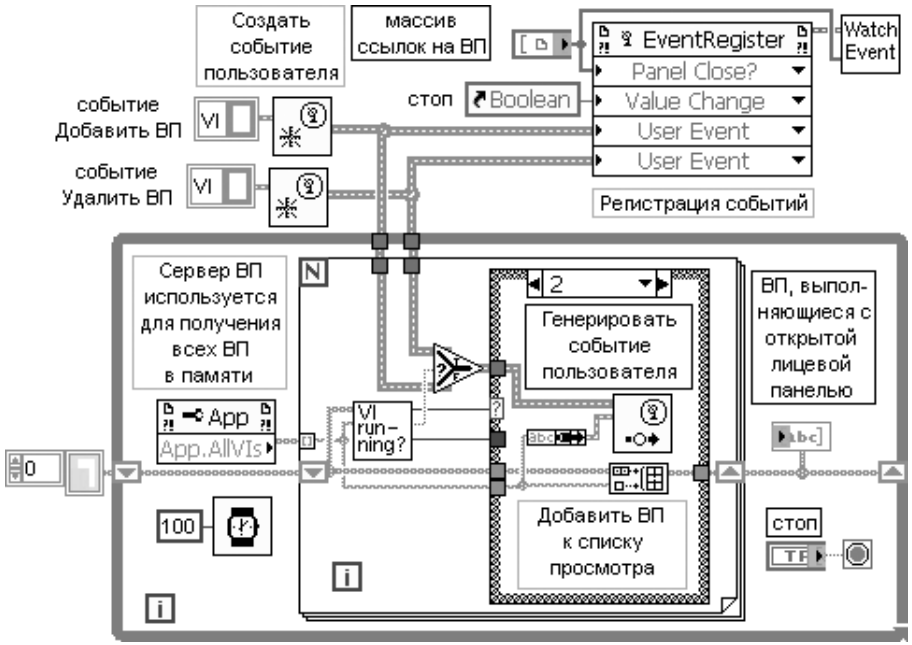


Рис. 2.54. Блок-диаграмма модернизированного ВП ***Динамическая проверка ВП***

в памяти (All VIs In Memory), выводится массив имен всех ВП, находящихся в памяти. Далее каждое имя с помощью подприбора **Выполняющиеся с открытой панелью** (Running with Panel Open) (рис. 2.55) проверяется на совпадение со списком ВП, выполняющихся на компьютере с открытой лицевой панелью (рис. 2.54). В зависимости от результата сравнения имя ВП может добавляться к списку ВП, удаляться из него или же список ВП может оставаться неизменным. На рис. 2.54 показан вариант добавления имени к списку ВП.

Подприбор **Выполняющиеся с открытой панелью** (рис. 2.55) анализирует состояние выполнения ВП с заданным именем и состояние его окна лицевой панели с помощью **Узла свойства** для класса **ВП**, в котором установлены свойства **Выполнение.Состояние** (Exec.State) и **Окно лицевой панели.Открыто** (FP.Open).

Одновременно с добавлением имени ВП к списку имен на блок-диаграмме ВП **Динамическая проверка ВП** (рис. 2.54) производится генерация события функцией **Генерировать событие пользователя**. Генерируемое событие пользователя передается структуре **Событие**, находящейся в подприборе **Ожидать событий** (Wait for Events) (рис. 2.56), и обрабатывается ею. Обработка заключается в добавлении ссылки на новый ВП, выполняющийся с открытой лицевой панелью, или удалении ссылки на ВП, лицевая панель которого закрывается. После добавления ссылки производится перерегистрация событий с помощью функции **Регистрация событий**.

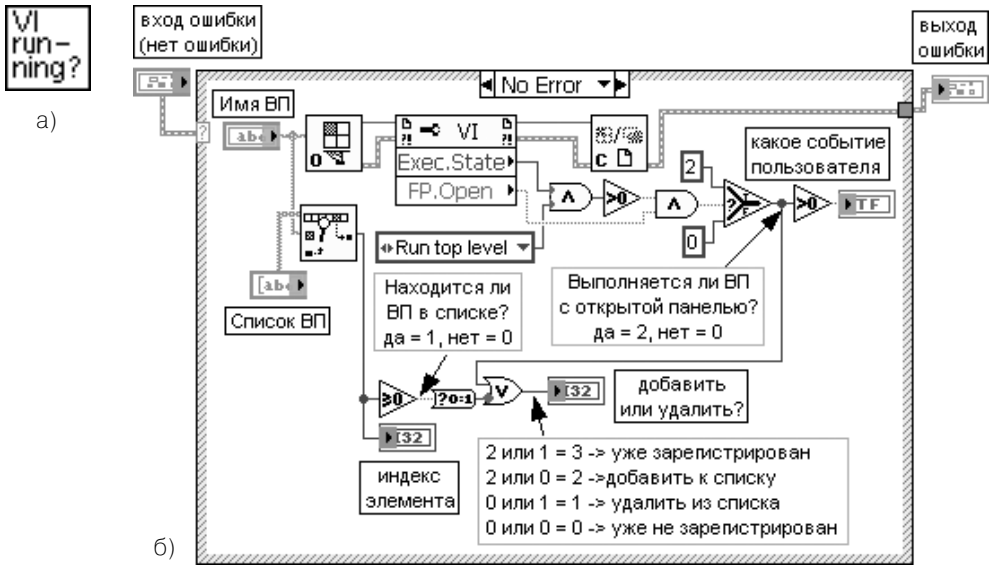


Рис. 2.55. Иконка (а) и блок-диаграмма (б) модернизированного подприбора ***Выполняющиеся с открытой панелью***

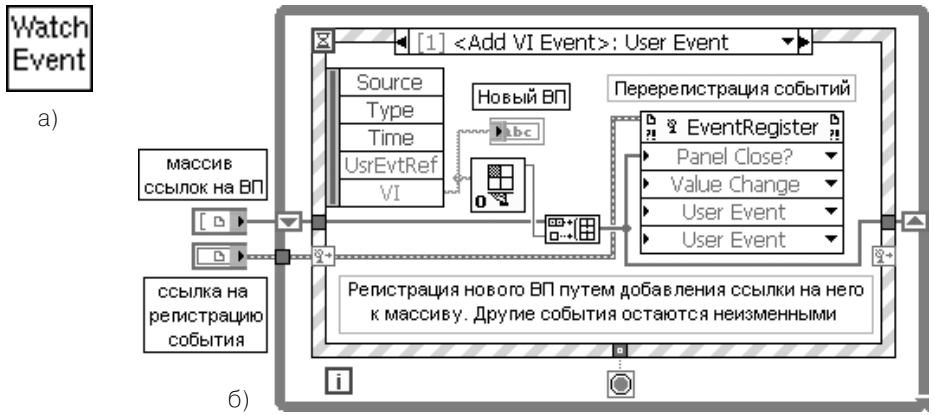


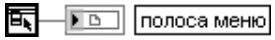
Рис. 2.56. Иконка (а) и блок-диаграмма (б) модернизированного подприбора ***Ожидать событий***

Функции меню

Использование функций подпалитры **Меню** (рис. 2.50в) позволяет модифицировать меню приложений LabVIEW. Функции, расположенные в верхнем ряду этой подпалитры, позволяют обрабатывать обращения к меню.

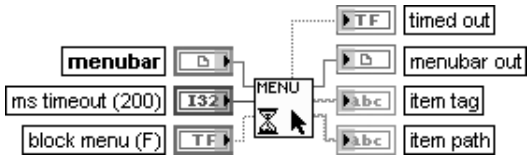
Current VI's Menubar

Полоса меню текущего ВП

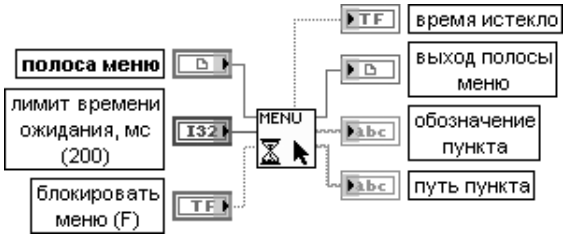


Функция возвращает ссылку на **полосу меню** (menubar) текущего ВП. Ссылка на **полосу меню** необходима для использования других функций подпалитры **Меню**

Get Menu Selection



Получить выбор меню



Функция возвращает обозначение последнего выбранного пункта меню после ожидания в течении интервала времени, заданного на входе **лимит времени ожидания, мс** (ms timeout).

Вход **полоса меню** (menubar) служит для приема ссылки на полосу меню ВП. Эта ссылка может быть получена с помощью функции **Полоса меню текущего ВП** (Current VI's Menubar).

Вход **лимит времени ожидания, мс** (ms timeout) определяет максимальный интервал времени, в течение которого функция проверяет выбор меню. По умолчанию оно равно 200 мс. Установка значения –1 приводит к неопределенно долгому времени ожидания.

Если на входе **блокировать меню** (block menu) установлено значение ИСТИНА, то LabVIEW отключает отслеживание меню после считывания обозначения его пункта. После обработки выбранного пункта необходимо использовать функцию **Разрешить отслеживание меню** (Enable Menu Tracking) для разрешения отслеживания меню. По умолчанию на входе установлено значение ЛОЖЬ.

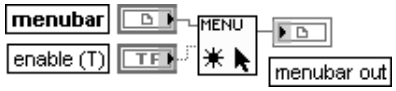
Отображение значения ИСТИНА на выходе **время истекло** (timed out) означает, что выбор меню не был произведен пользователем за время, заданное на входе **лимит времени ожидания**.

Выход полосы меню возвращает неизменное значение одноименного входа.

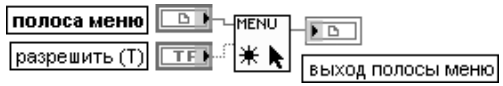
Выход **обозначение пункта** (item tag) содержит выбранный пункт меню. Для обработки каждого пункта меню следует соединить этот выход с терминалом выбора структуры **Вариант**. При создании структуры **Вариант** для обработки каждого пункта меню необходимо ввести обозначения пунктов приложения в ярлыке селектора варианта для обработки пунктов меню приложения.

Выход **путь пункта** (item path) описывает положение пункта в иерархии, которая представлена в форме списка пунктов меню, разделенных двоеточием. Например, при выборе пункта **Открыть** из меню **Файл** путь пункта будет содержать запись **Файл:Открыть**

Enable Menu Tracking



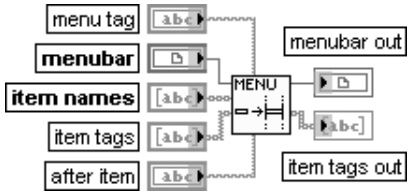
Разрешить отслеживание меню



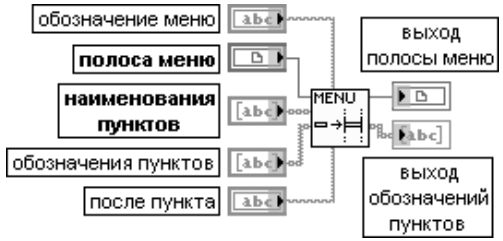
Функция разрешает или запрещает отслеживание выбора меню. Если функция **Получить выбор меню** (Get Menu Selection) используется для блокирования меню, то данная функция должна использоваться для разрешения меню.

Если на входе **разрешить** (enable) установлено значение ИСТИНА (по умолчанию), то отслеживание меню разрешено. В противном случае оно запрещено

Insert Menu Items



Вставить пункты меню



Функция вставляет пункты меню, заданные **наименованиями пунктов** (item names) или **обозначениями пунктов** (item tags), в полосу меню или подменю полосы меню.

Вход **обозначение меню** (menu tag) определяет подменю, в которое необходимо вставить пункты. Если обозначение меню не определено, то функция вставляет пункт в полосу меню.

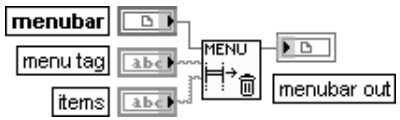
Вход **полоса меню** (menubar) содержит ссылку на полосу меню ВП. Эта ссылка может быть получена с помощью функции **Полоса меню текущего ВП** (Current VI's Menubar).

Вход **наименования пунктов** (item names) идентифицирует пункты, которые необходимо вставить в меню. **Наименования пунктов** является строкой, которая появляется в меню. Можно подключить входы **наименования пунктов** или **обозначения пунктов**, в этом случае как имена, так и обозначения имеют одни и те же значения. Если вставляется только один пункт, то необходимо подключить строку ко входу **наименования пунктов**.

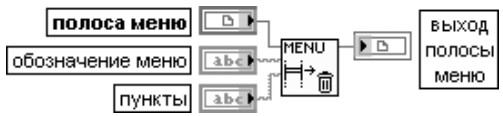
Вход **обозначения пунктов** (item tags) идентифицирует пункты, которые необходимо вставить в меню. **Обозначения пунктов** представляют строку, которая возвращается при выборе пункта меню. Можно подключить входы **наименования пунктов** или **обозначения пунктов**, в этом случае как имена, так и обозначения имеют одни и те же значения. Если вставляется только один пункт, то необходимо подключить строку ко входу **обозначения пунктов**.

Для вставки пунктов меню приложения необходимо использовать **обозначения пунктов приложения** (application item tags)

Delete Menu Items

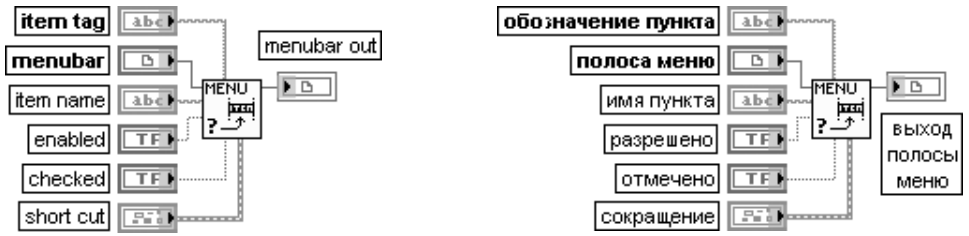


Удалить пункты меню



Функция удаляет пункты меню из полосы меню или из подменю полосы меню

Set Menu Item Info / Установить информацию о пункте меню



Функция устанавливает атрибуты пунктов меню. Неподключенные атрибуты остаются неизменными.

Вход **обозначение пункта** (item tag) представляет пункт меню, для которого устанавливаются атрибуты. Если **обозначение пункта** является недостоверным, то функция возвращает ошибку. Для установки пунктов меню приложения необходимо использовать **обозначения пунктов приложения** (application item tags).

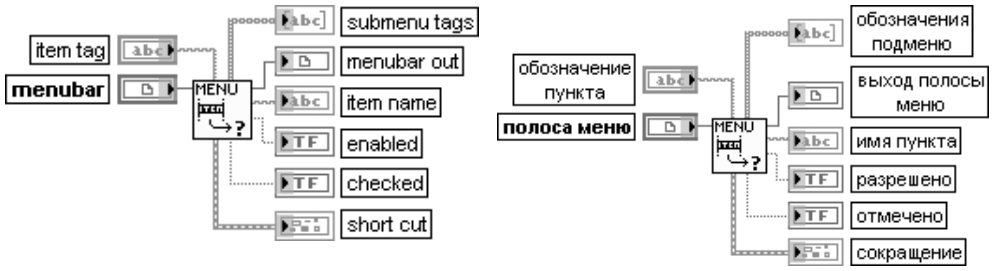
Если на вход **разрешено** (enabled) подается значение ЛОЖЬ, то пункту меню придается серый цвет, свидетельствующий о его отключении.

Если на входе **отмечено** (checked) установлено значение ИСТИНА, то пункт меню имеет отметку, расположенную перед ним.

На входе **сокращение** (short cut) задается клавишный эквивалент выбора пункта меню с помощью мыши. **Сокращение** обычно является комбинацией клавиши меню с одной из других клавиш. В Windows клавишей меню является клавиша <Ctrl>. Дополнительно может использоваться клавиша <Shift>. В состав кластера **сокращение** входят следующие элементы:

- логический элемент управления **включить клавишу Shift?** (include Shift key?). Если он находится в состоянии ИСТИНА, то сокращение включает клавишу <Shift> в дополнение к клавише меню и **«горячую» клавишу** (shortcut key);
- логический элемент управления **включить клавишу Ctrl?** (include Ctrl key?). Этот элемент может устанавливаться в состояние ЛОЖЬ только в случае выбора таких клавиш, как <F1>, <F2> и т. п. в качестве «горячих» клавиш;
- строковый элемент управления **«горячая» клавиша** (shortcut key) описывает клавишу, связанную с **сокращением**

Get Menu Item Info / Получить информацию о пункте меню



Функция возвращает атрибуты пункта меню или полосы меню.

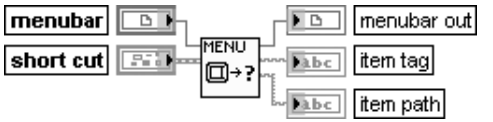
Выход **обозначения подменю** (submenu tags) содержит обозначения пунктов подменю, если пункт имеет подменю.

Выход **имя пункта** (item name) представляет строку, которая появляется в меню.

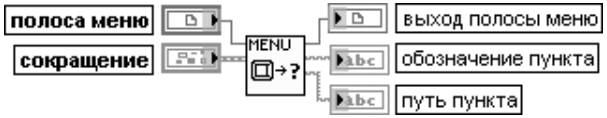
Переход выхода **разрешено** (enabled) в состояние ЛОЖЬ свидетельствует о придании пункту меню серого цвета.

Появление на выходе **отмечено** (checked) значения ИСТИНА связано с наличием отметки перед пунктом меню

Get Menu Short Cut Info



Получить информацию о сокращении меню



Функция возвращает пункт меню, который доступен через данное сокращение

На рис. 2.57 в качестве примера использования функций из подпалитры **Меню** приведена блок-диаграмма модернизированного ВП **Демонстрация динамической вставки** (Dynamic Insert Demo) из библиотеки ВП menubars.llb набора примеров NI Example Finder.

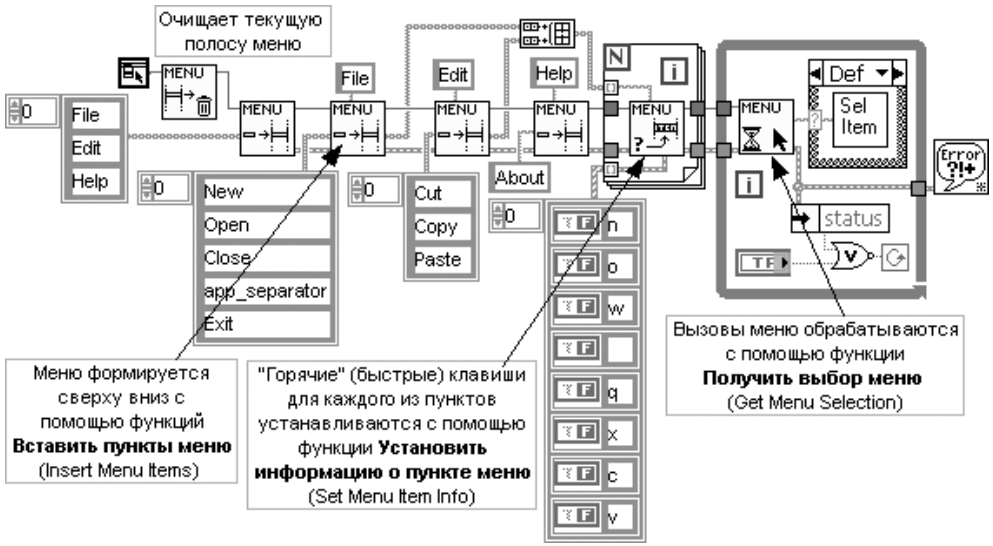
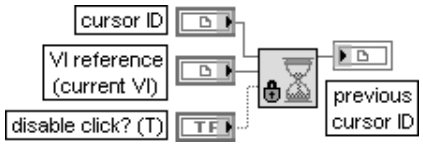


Рис. 2.57. Блок-диаграмма модернизированного ВП ***Демонстрация динамической вставки***

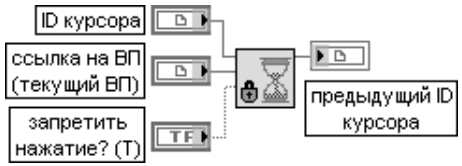
ВП из подпалитры Курсор

Использование ВП из подпалитры **Курсор** (рис. 2.50г) позволяет изменить вид курсора на лицевой панели ВП. Так, например, если ВП производит сбор или анализ данных и не может воспринимать обращения пользователя, можно изменить вид курсора так, чтобы он выглядел как наручные или песочные часы. После завершения операций сбора или анализа данных и восстановления способности восприятия обращений пользователя можно изменить вид курсора на вид курсора по умолчанию.

Set Busy



Установить занятость



ВП изменяет вид курсора на лицевой панели ВП на курсор, занятый системой. Этот ВП также может использоваться для блокирования мыши на лицевой панели. Для обратного изменения курсора к виду курсора LabVIEW по умолчанию и включения мыши необходимо использовать ВП **Снять установку занятости** (Unset Busy). Использование ВП **Установить занятость** аналогично использованию ВП **Установить курсор** (Set Cursor) при подключении 1 ко входу **иконка** (icon).

Вход **ID курсора** (cursor ID) содержит ссылку к курсору, который предполагается использовать на лицевой панели ВП. По умолчанию это курсор, занятый системой. Для получения ссылки к курсору необходимо использовать ВП **Создать курсор из файла** (Create Cursor From File). Если ссылка к курсору недостоверна, то LabVIEW изменяет курсор на курсор LabVIEW по умолчанию и возвращает управление курсором LabVIEW.

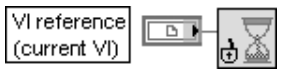
Вход **ссылка на ВП** (VI reference) определяет ссылку на ВП, для которого производится изменение курсора. По умолчанию это ссылка на ВП, который содержит данный ВП как подприбор. Для получения ссылки на другой ВП можно использовать функцию **Открыть ссылку на ВП** (Open VI Reference).

Этот вход полезен, если открыто несколько лицевых панелей и необходимо изменить курсор только на одной лицевой панели. Если лицевая панель ВП, для которого производится изменение курсора, не открыта, то ВП **Установить курсор** возвращает ошибку.

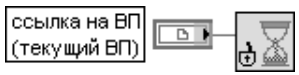
Если на входе **запретить нажатие** (disable click?) установлено значение ИСТИНА (по умолчанию), то ВП блокирует мышь на лицевой панели. Для включения мыши необходимо использовать ВП **Снять установку занятости** (Unset Busy).

При блокировании мыши на лицевой панели остается возможность нажатия кнопки **Прервать выполнение** (Abort Execution) в инструментальной панели

Unset Busy

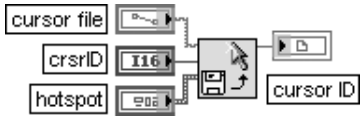


Снять установку занятости

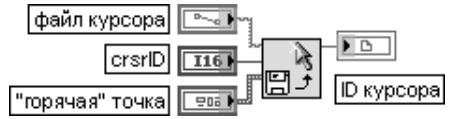


ВП изменяет вид курсора на лицевой панели ВП с курсора занятости на курсор LabVIEW и включает мышь на лицевой панели. Этот ВП должен использоваться только после ВП **Установить занятость** (Set Busy)

Create Cursor From File



Создать курсор из файла



ВП возвращает ссылку на курсор, содержащийся в файле курсора. Если ВП, распространяемый в виде автономного приложения или динамически подключаемой библиотеки, использует курсор из файла курсора, то файл курсора должен также прилагаться к распространяемой программе.

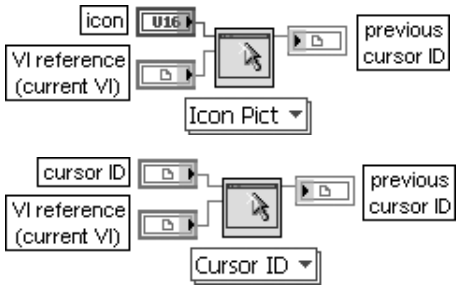
Вход **файл курсора** (cursor file) передает путь к файлу, который содержит курсор. Если файл не существует или файл не является файлом курсора, то LabVIEW возвращает код ошибки 7. В Windows файл должен иметь расширение .ani или .cur.

Вход **«горячая» точка** (hotspot) содержит координаты «горячей» точки курсора, начинающиеся от верхнего левого угла. «Горячая» точка не может быть установлена для анимированного курсора. National Instruments рекомендует устанавливать «горячую» точку только для UNIX, поскольку Windows и Mac OS уже имеют «горячую» точку.

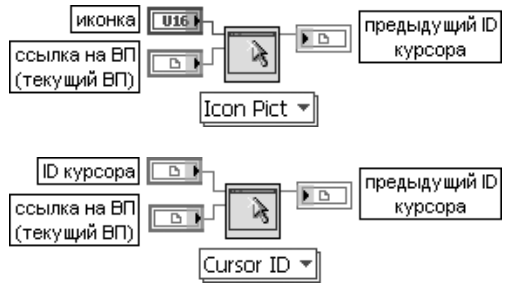
Выход **ID курсора** (cursor ID) представляет ссылку на курсор, содержащийся в **файле курсора**. Если ссылка на курсор уже существует, то ВП возвращает существующую ссылку. При необходимости изменения вида курсора на лицевой панели следует подключить эту ссылку к ВП **Установить курсор** или **Установить занятость**.

Для закрытия ссылки на курсор после его установки необходимо использовать ВП **Ликвидировать курсор**

Set Cursor



Установить курсор



ВП изменяет вид курсора на лицевой панели ВП. Этот полиморфный ВП может использоваться для установки системного курсора или курсора LabVIEW, а также для установки курсора с использованием ссылки. Тип данных, подключаемых ко входу **иконка** (icon), определяет экземпляр используемого ВП.

Вход **иконка** (icon) определяет вид курсора (системный или LabVIEW), который предполагается использовать на лицевой панели ВП. Вид палитры иконок курсора приведен на рис. 2.58.

Выход **предыдущий ID курсора** (previous cursor ID) передает ссылку на курсор лицевой панели ВП, использовавшегося перед выполнением данного ВП. Для возврата от установленного курсора к предыдущему следует подключить этот выход к еще одному экземпляру ВП **Установить курсор**.

Рис. 2.58. Вид палитры иконок курсора

Destroy Cursor

Ликвидировать курсор

ВП закрывает ссылку к курсору и изменяет курсор на курсор по умолчанию в любых ВП, использующих ссылку

В качестве примера использования функций подпалитры **Курсор** на рис. 2.59 приведена блок-диаграмма модернизированного ВП **Изменить иконку курсора** (Change Cursor Icon) из библиотеки CursorUtilities.llb набора примеров NI Example Finder.

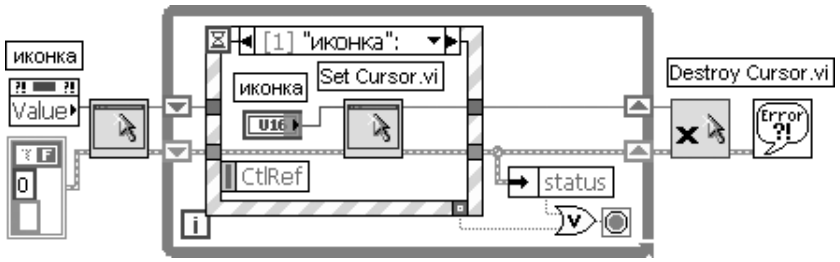


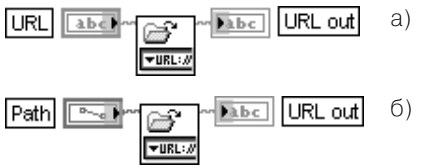
Рис. 2.59. Блок-диаграмма модернизированного ВП ***Изменить иконку курсора***

В начале работы ВП с помощью функции **Установить курсор** устанавливает исходный вид курсора. При выборе пользователем другого курсора ВП обнаруживает изменение состояния элемента управления и соответствующим образом изменяет вид курсора.

Функции из подпалитры Помощь

Из подпалитры **Помощь** (Help) (рис. 2.50д) ниже в таблице рассмотрены функции **Открыть URL в браузере по умолчанию** (Open URL in Default Browser) и **Управление оперативной помощью** (Control Online Help).

Open URL in Default Browser



Открыть URL в браузере по умолчанию

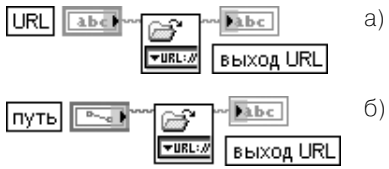
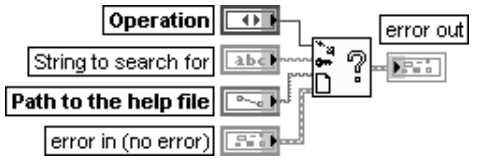


Рис. 2.60. Варианты подключения ВП ***Открыть URL в браузере по умолчанию***

Функция отображает URL или файл HTML в Web-браузере по умолчанию. Тип данных, подключенных ко входу **URL**, определяет используемый экземпляр этого полиморфного ВП (рис. 2.60). Если URL или путь, который подключен к этому ВП, содержит символ пробела, то ВП кодирует пробел как %20 перед отображением URL или файла HTML в Web-браузере

Control Online Help



Управление оперативной помощью

ВП управляет откомпилированным файлом помощи путем отображения списка содержимого, перехода к определенной теме файла или закрытия файла помощи. Пользователь может создать откомпилированный файл помощи и использовать эту функцию для установления связи между ВП и файлом помощи. Пользователь также может использовать эту функцию для отображения HTML-файла в браузере по умолчанию.
Вход **операция** (Operation) определяет операцию, выполняемую с файлом помощи.

0	**Содержание** (Contents) – отображает список содержимого файла помощи. Если эта функция используется для управления файлом с расширением *.chm, установка этого входа в 0 приводит к отображению списка содержимого, указателя или таблицы поиска, зависящей от того, какая таблица была видима при закрытии файла с указанным расширением
1	**Ключ** (Key) – отображает определенную тему в файле помощи путем использования ключевого указателя файла помощи или имени индивидуального HTML-файла в откомпилированном файле помощи
2	**Закрыть** (Close) – закрывает файл помощи

Вход **строка для поиска** (String to search for) содержит ключевой указатель или имя HTML-файла для темы файла помощи, которую необходимо отобразить. По умолчанию это пустая строка. LabVIEW игнорирует эту строку, если на входе **операция** установлены значения 0 или 2.

Если эта строка содержит ключевой указатель, то она должна совпадать с ключевым указателем, появляющимся в указателе файла помощи. Для перехода к теме с многоуровневым ключевым указателем необходимо ввести ключевой указатель первого уровня и через двоеточие ключевой указатель второго уровня.

Вход **путь к файлу помощи** (Path to the help file) содержит путь к откомпилированному файлу помощи, которым предполагается управлять. Файл помощи может иметь расширение .chm или .hlp.

На рис. 2.61 в качестве примера приведена блок-диаграмма ВП, обеспечивающего вывод файла помощи при нажатии кнопки **Помощь** (Help) в окне конфигурирования Экспресс-ВП.

*Рис. 2.61. Блок-диаграмма фрагмента Экспресс-ВП, использующего функцию **Управление оперативной помощью***

2.2.2. Функции управления приложением

LabVIEW обеспечивает широкий набор функций взаимодействия ВП между собой и с другими приложениями. Функции, размещенные в палитре **Управление приложением** (Application Control), позволяют одним локальным или удаленным ВП вызывать функции и свойства других ВП по их **ссылке**. Эти функции (рис. 2.62) позволяют программно с помощью сервера ВП управлять свойствами приложения (LabVIEW), виртуального прибора, элементов управления и индикации. Под управлением понимаются динамическое изменение свойств (атрибутов) ВП или объектов лицевой панели, динамический вызов и загрузка ВП в память с последующим запуском и закрытием.

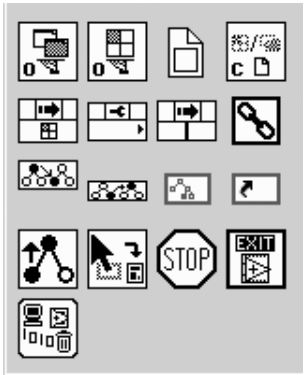


Рис. 2.62. Вид палитры функций управления приложением

Обращение к серверу может производиться с помощью технологии ActiveX или с помощью вызова ВП по ссылке (рис. 2.63). Для использования в качестве сервера ВП должен быть настроен программно или итерационно с помощью разделов **Сервер ВП: Конфигурация** (VI Server: Configuration), **Сервер ВП: Доступ TCP/IP** (VI Server: TCP/IP Access) и **Сервер ВП: Экспортируемые ВП** (VI Server: Exported VIs) диалогового окна **Опции** (Options), вызываемого из раздела **Инструменты** (Tools) главного меню.

Описание функций управления приложением приведено в таблице.

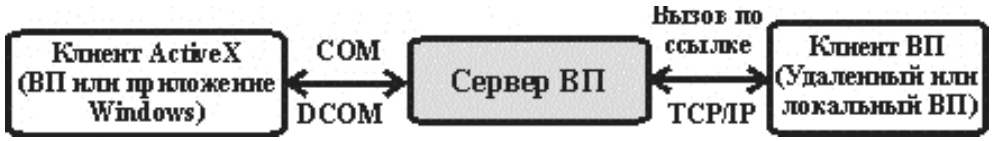
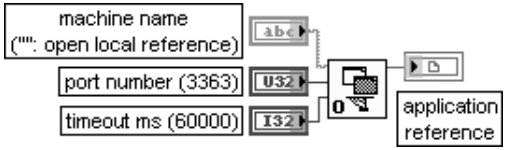
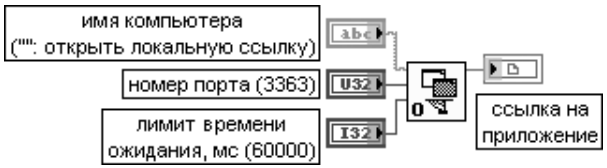


Рис. 2.63. Схема взаимодействия клиентов и сервера ВП

Open Application Reference



Открыть ссылку на приложение



Функция возвращает ссылку на приложение Сервера ВП, работающее на заданном компьютере.

Вход **имя компьютера** (machine name) задает адрес компьютера, работающего в среде LabVIEW, с которой необходимо установить соединение. Если на входе **имя компьютера** (machine name) будет подключена пустая строка, то функция возвратит ссылку на локальную среду LabVIEW, в которой она выполняется. Если на этом входе будет задано имя компьютера, то функция попытается установить связь по TCP с сервером ВП этого удаленного компьютера по заданному порту. Адрес компьютера может быть указан в точечной десятичной записи (такой как 130.164.15.250) или записи доменного имени (такой как foo.ni.com).

При осуществлении соединения между ВП и автономным приложением LabVIEW на том же компьютере к этому входу должна быть подключена строка, содержащая имя **localhost**. **localhost**, осуществляет соединение со средой выполнения LabVIEW (LabVIEW Run-Time Engine).

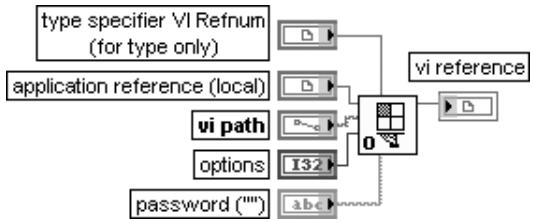
Вход **номер порта** (port number) определяет порт, который прослушивается удаленным приложением LabVIEW. По умолчанию используется номер порта слушателя сервера ВП (3363).

Вход **лимит времени ожидания, мс** (timeout ms) определяет время, отводимое на установление соединения, в миллисекундах. По умолчанию оно равно 60,000 мс (1 мин). Значение –1 на этом входе вызывает неопределенно долгое ожидание.

Выход **ссылка на приложение** (application reference) можно использовать как вход **узлов свойства** (Property Nodes) или **узлов вызова** (Invoke Nodes) для получения или установки свойств или вызова методов приложения. Использование данного выхода как входа функции **Открыть ссылку на ВП** (Open VI Reference) позволяет получить ссылку на ВП в этом приложении.

Закрытие ссылки осуществляется с помощью функции **Закрыть ссылку** (Close Reference), рассмотренной ниже

Open VI Reference



Открыть ссылку на ВП

Функция возвращает ссылку на ВП, специальный элемент управления или глобальную переменную, определенные строкой имени или путем к месту расположения ВП на диске. На вход **описатель типа ссылки на ВП** (type specifier VI Refnum) необходимо подключить элемент управления или константу ссылки на ВП для создания ссылки на ВП строгого типа, которая должна быть передана функции **Узел вызова по ссылке** (Call By Reference Node).

Вход **описатель типа ссылки на ВП** (type specifier VI Refnum) определяет тип данных выхода **ссылка на ВП** (vi reference). Значение этого входа игнорируется. Функция использует данный параметр только для определения типа данных.

По умолчанию функция возвращает **общую ссылку на ВП** (Generic VI reference). При использовании выходной ссылки с целью вызова ВП с помощью функции **Узел вызова по ссылке** (Call By Reference Node) необходимо подключить вход **описатель типа**. В этом случае становится невозможным подключение выходной ссылки к методу **Выполнить ВП** (Run VI).

Вход **ссылка на приложение** (application reference) представляет ссылку на приложение LabVIEW. По умолчанию это ссылка на локальную среду LabVIEW. Если вход подключен и ссылка указывает на удаленную среду LabVIEW, то удаленная среда LabVIEW запрашивается с целью получения ссылки на ВП.

Вход **путь к ВП** (vi path) содержит строку с именем ВП, к которому создается ссылка, или путь, содержащий полный путь к этому ВП. Функция загружает определенный таким образом ВП в память, если он еще не был открыт, и возвращает ссылку на этот ВП. Если подключается строка с именем, то ВП должен уже находиться в памяти. Если подключается путь и ВП с тем же именем уже находится в памяти, то функция возвращает ссылку к открытому ВП, независимо от того, что его путь такой же, как на входе. Если ВП не находится в памяти, то ВП должен находиться по заданному пути для успешного выполнения функции.

Если путь является относительным, то ВП интерпретирует путь относительно вызывающего ВП или каталога приложения, если вызывающий ВП не сохранен.

Вход **опции** (options) представляет набор битов, которые определяют способ обработки ссылки на ВП. Предусмотрены следующие комбинации битов на входе **опции**.

0x01	Запись модификаций. Звездочка (*) появляется в названии ВП для индикации того, что изменения были сделаны с помощью сервера ВП. ВП должен находиться в режиме редактирования, чтобы разрешить LabVIEW записать модификации
0x02	Открытие шаблонов для редактирования. Эта опция действует только на шаблонные файлы
0x04	Подсказка пользователю при закрытии. При попытке закрыть ссылку на ВП после выполнения изменений сервер ВП напоминает о необходимости сохранения перед закрытием ВП
0x08	Подготовка для повторного запуска. Резервирует целевой ВП так, что он не может редактироваться, и, если целевой ВП является реентерабельным, выделяет адресное пространство памяти для ссылки этого ВП. Если целевой ВП не является реентерабельным, то функция возвращает ошибку. При закрытии ссылки ВП LabVIEW снимает резервирование реентерабельного целевого ВП и освобождает адресное пространство. При установке данной опции и использовании метода **Выполнить ВП** (Run VI) могут одновременно выполняться несколько экземпляров реентерабельного ВП
0x10	Подсказка пользователю найти отсутствующие подприборы

Вход **пароль** (password) содержит пароль ВП. Если ВП не защищен паролем, эта функция игнорирует пароль. Если ВП защищен паролем и пользователь ввел неправильный пароль, то эта функция возвращает ошибку и недостоверную ссылку на ВП.

Выход **ссылка на ВП** (vi reference) содержит ссылку, связанную с запрошенным ВП. Если функция выполнилась с ошибкой, то выход **ссылка на ВП** содержит значение **Не ссылка** (Not A Refnum)

Call By Reference Node / Узел вызова по ссылке

Функция вызывает ВП, заданный на входе **ссылка** (reference). Ссылка ВП должна быть строгого типа.

Как функция **Узел вызова по ссылке** (Call By Reference Node), так и узел подприбора (subVI node) вызывают ВП. Функция **Узел вызова по ссылке** позволяет динамически вызывать произвольный ВП, соединительная панель которого соответствует входной ссылке строгого типа. В отличие от этого узел подприбора позволяет вызывать специфический ВП, который определен статически при размещении узла подприбора на блок-диаграмме.

В нижней части функции находится область, в которой отображается соединительная панель вызываемого ВП. К терминалам этой панели так же, как и к терминалам подприбора, могут быть подключены элементы управления и индикаторы

Property Node / Узел свойства

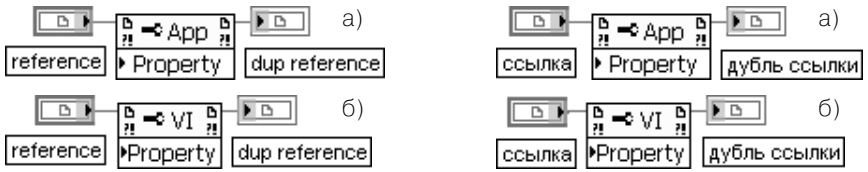
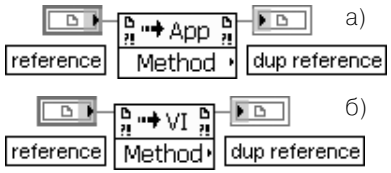


*Рис. 2.64. Варианты конфигурирования **Узла свойства***

Узел получает (читает) и/или устанавливает (записывает) свойства по **ссылке** (reference). **Узел свойства** (Property Node) автоматически адаптируется к классу объекта, задаваемого по ссылке. Для локального приложения этот узел можно также настроить и без подключения ссылки с помощью выбора пунктов **Выбрать класс** ⇒ **Сервер ВП** (Select Class ⇒ VI Server) контекстного меню. На рис. 2.64а показан вид узла при выборе класса **Приложение** (Application), а на рис. 2.64б – класса **ВП**.
Выбор конкретного свойства при установленной ссылке производится с помощью строки **свойства** (Properties) контекстного меню полоски узла с надписью **свойство** (Property). С помощью этого же меню устанавливается режим чтения или записи свойств.
Узел выполняет каждый терминал в порядке сверху вниз. Если в терминале произошла ошибка, то узел останавливается на этом терминале, возвращает ошибку и не выполняет следующие терминалы. Для того чтобы игнорировать ошибку и продолжить выполнение программы, можно установить опцию **Игнорировать ошибки внутри узла** (Ignore Errors Inside Node) контекстного меню

Invoke Node



Узел вызова

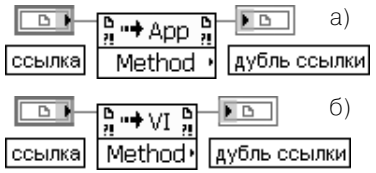
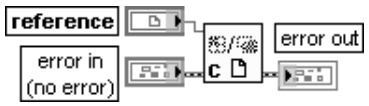


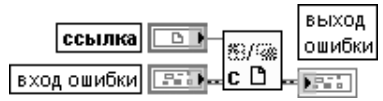
*Рис. 2.65. Варианты конфигурирования **Узла вызова***

Узел вызывает метод или действие по ссылке. Большинство методов имеют связанные с ними параметры.
Вход **ссылка** (reference) является ссылкой, связанной с объектом, который вызывает метод или выполняет действие. Ссылка может быть получена от функции **Открыть ссылку на ВП** (Open VI Reference) или от элементов управления **ссылка** (Refnum) лицевой панели. В состав элементов управления **ссылка** входят такие элементы, как ссылка на класс **Приложение** (Application Refnum), ссылка на класс **ВП** (VI Refnum), ссылка на класс **элемент управления** (Control Refnum) и ряд других. **Узел вызова** (Invoke Node) автоматически адаптируется к классу объекта, задаваемого по ссылке. На рис. 2.65 показаны варианты конфигурирования **Узла вызова** для класса **Приложение** (а) и для класса **ВП** (б).
Неподключенный **Узел вызова** может быть сконфигурирован относительно ссылки с помощью строки **выбрать класс** (Select Class) контекстного меню узла. Если узел сконфигурирован для класса **Приложение** (Application) или **виртуальный прибор** (VI), то ссылка может не подключаться. По умолчанию для класса **Приложение** ссылка относится к текущему приложению (LabVIEW), а для класса **ВП** она относится к ВП, содержащему **узел вызова**.
Узел позволяет получать (читать) и/или устанавливать (записывать) параметры методов. Параметры с белым фоном являются необходимыми входами, а параметры с серым фоном – рекомендуемыми.
Если маленькая стрелка в полосе параметра находится справа, то параметр читается, если же слева – то устанавливается (записывается). Для отображения только типа параметра необходимо выбрать строку **Формат имени** ⇒ **Без имен** (Name Format ⇒ No Names) из контекстного меню параметра

Close Reference



Закрыть ссылку



Функция закрывает ссылку, связанную с открытым ВП, объектом ВП или открытым приложением (LabVIEW)

Class Specifier Constant

Константа описателя класса

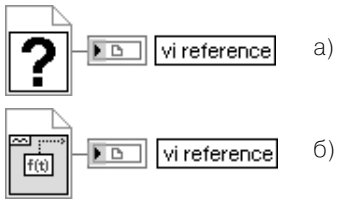
Константа ссылки заданного класса

Call Chain

Цепь вызова

Функция возвращает цепь вызовов от текущего ВП к ВП верхнего уровня. Если текущий ВП используется как подприбор, то эта функция позволяет определить все ВП, вызывающие данный подприбор

Static VI Reference



Статическая ссылка на ВП

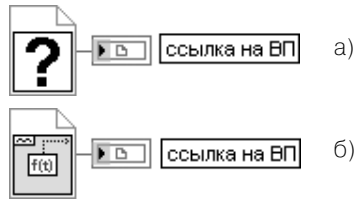
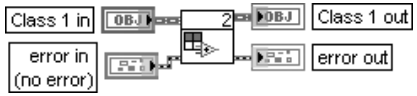


*Рис. 2.66. Варианты конфигурирования функции **Статическая ссылка на ВП***

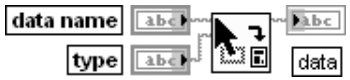
Функция обеспечивает статическую ссылку на ВП. Эту ссылку можно использовать вместе с методами и свойствами без необходимости ее явного открытия и закрытия. **Статическая ссылка на ВП** действует как подприбор и появляется в иерархии ВП верхнего уровня. Пользователь не может использовать функцию **Узел вызова по ссылке** (Call By Reference node), метод **Выполнить ВП** (Run VI) или какой-либо другой метод для выполнения ВП, содержащего узел **Статическая ссылка на ВП**. Также этот узел не может использоваться для вызова методов **Прервать ВП** (Abort VI), **Вернуть ВП** (Revert VI), **Сделать текущие значения значениями по умолчанию** (Make Current Values Default), **Экспортировать строки ВП** (Export VI Strings) или **Импортировать строки ВП**. Для предотвращения рекурсии нельзя иметь ссылку на ВП верхнего уровня, в котором находится функция **Статическая ссылка на ВП**.
На рис. 2.66 показан вид функции непосредственно после вызова (а) и после установки ссылки (б) с помощью строки Browse for Path контекстного меню

Call Parent Method / Вызвать родительский метод



Узел вызывает ближайшую реализацию метода класса. Его можно использовать только на блок-диаграмме ВП, принадлежащего классу, который наследует от ВП родительского класса. Дочерний ВП должен быть ВП с динамической отсылкой и должен иметь то же имя, что ВП родительского класса. Параметры, или входные и выходные терминалы, узла зависят от ВП родительского класса и изменяются так, чтобы полностью ему соответствовать. Узел **Вызвать родительский метод** при отображении на блок-диаграмме дочернего ВП в виде подприбора использует иконку ВП родительского класса, в левом нижнем углу которой помещен значок, указывающий на то, что подприбор вызывает родительский ВП вместо динамической отсылки, которая обычно выполняется узлом динамического подприбора

Get Drag Drop Data / Получить данные Drag Drop



Функция возвращает переносимые данные из LabVIEW при выполнении операции переноса и фиксации объекта (drag and drop). Эту функцию следует использовать только в том случае, когда необходимо получить доступ к переносимым данным. Если операция drag and drop не выполняется или если запрошенные данные недоступны, LabVIEW возвращает ошибку.

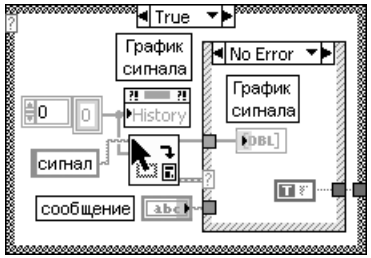


Рис. 2.67. Фрагмент блок-диаграммы ВП Drag and Drop – Built-in Source Custom Data

На вход **имя данных** (data name) подается определяемое пользователем имя переносимых данных, которые необходимо извлечь. В имени данных не может использоваться префикс LV_.

Вход **тип** (type) определяет тип переносимых данных, которые должны быть получены.

Выход **данные** (data) содержит переносимые данные, возвращаемые из LabVIEW

На рис. 2.67 приведен фрагмент ВП Drag and Drop – Built-in Source Custom Data из набора примеров NI Example Finger, в котором функция Get Drag Drop Data позволяет отображать на графике определенный сигнал с помощью переноса названия этого сигнала из элемента управления Дерево.

Stop / Стоп

Функция останавливает ВП, в котором она выполняется подобно действию кнопки **Прервать выполнение** в инструментальной панели. Остановка происходит при поступлении на вход значения ИСТИНА, после того как завершится текущее выполнение очередного узла.

Использовать данную функцию для прерывания выполнения всех ВП в иерархии из блок-диаграммы следует с особой осторожностью. Перед тем как вызвать эту функцию с установленным на входе значением ИСТИНА, необходимо убедиться в завершенности таких задач, как закрытие файлов, установка сохраняемых значений для управляемых устройств и т. д. При помещении этой функции в подприбор ее поведение должно быть известно другим пользователям ВП, поскольку она вызывает прерывание выполнения их ВП, входящих в иерархию.

В общем случае следует ограничить использование этой функции при наличии встроенных протоколов завершения ВП. Так, например, операции ввода/вывода могут выполняться в цикле по условию так, что ВП может завершить цикл по ошибке ввода/вывода. Также для завершения цикла по запросу пользователя может использоваться логический элемент управления на лицевой панели

Quit LabVIEW / Выход из LabVIEW

Функция останавливает все выполняющиеся ВП и заканчивает работу текущей реализации LabVIEW. Эта функция закрывает только LabVIEW. Функция останавливает все работающие ВП тем же способом, что и функция **Стоп** (Stop).

Если на входе **выйти?** (quit?) установлено состояние ИСТИНА (по умолчанию), то LabVIEW выходит. Если ко входу **выйти?** подключен кластер ошибки и происходит ошибка, то кластер ошибки передает значение ИСТИНА этой функции

В качестве примера использования функций управления приложением на рис. 2.68 приведен фрагмент блок-диаграммы ВП **Пример динамической загрузки** (Dynamic Load Example) из набора примеров NI Example Finger. При вызове указанного на диаграмме кадра с помощью функций **Открыть ссылку на ВП** и **Узел вызова по ссылке** производятся динамическая загрузка и запуск ВП **Создать набор данных** (Create Data Set), который формирует на выходе два одно-

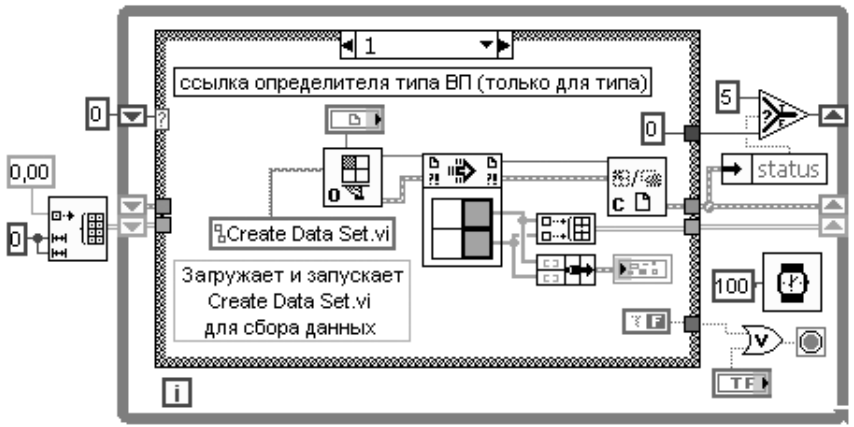


*Рис. 2.68. Блок-диаграмма ВП **Пример динамической загрузки***

мерных массива данных. После выполнения ВП с помощью функции **Закрыть ссылку** он выгружается из памяти.

2.2.3. Функции и ВП синхронизации

Функции и ВП синхронизации (рис. 2.69а) используются для синхронизации параллельно выполняющихся задач и передачи данных между такими задачами. В состав функций и ВП входят функции **Операции уведомителя** (Notifier Operations) (рис. 2.69б), **Операции очереди** (Queue Operations) (рис. 2.69в), ВП **Семафор** (Semaphore VIs) (рис. 2.69г), ВП **Встреча** (Rendezvous VIs) (рис. 2.69д) и **Функции случаев** (Occurrences Functions) (рис. 2.69е).

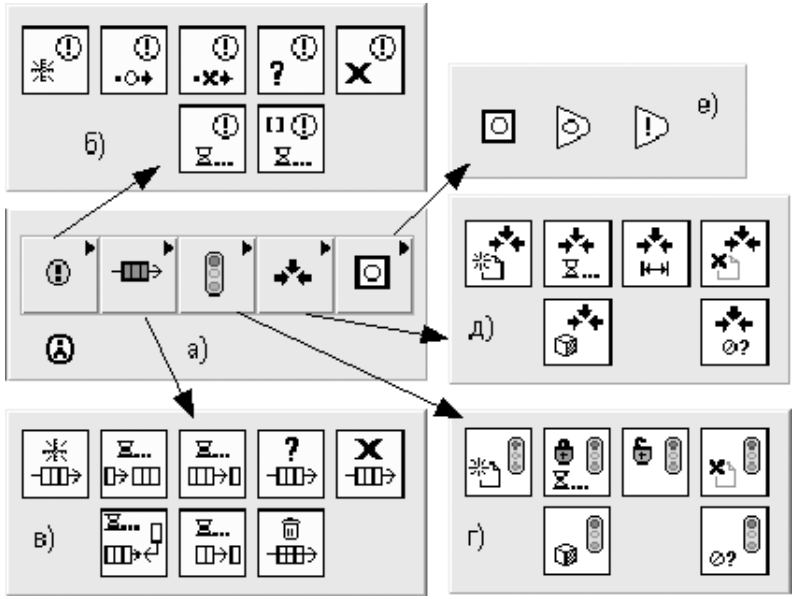


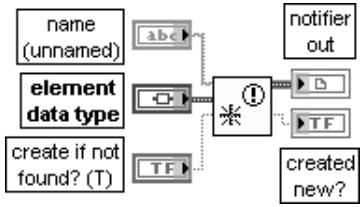
Рис. 2.69. Вид палитры (а) и подпалитр (б)–(е) функций и ВП синхронизации

Операции уведомителя

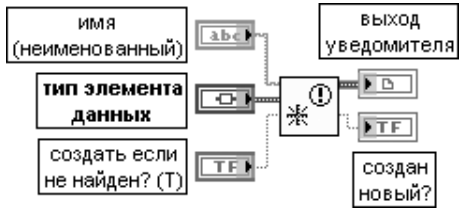
Ниже в таблице рассмотрены функции из подпалитры **Операции уведомителя** (Notifier Operations). Использование данных функций позволяет приостанавливать выполнение блок-диаграммы до получения данных из других секций блок-диаграммы или других ВП, выполняющихся на том же компьютере. Эти функции не могут использоваться для связи с ВП на других компьютерах через сеть или посредством сервера ВП.

В отличие от функций **Операции очереди** (Queue Operations) функции **Операции уведомителя** не буферизируют отправляемые сообщения. Если нет узлов, ожидающих отправляемое сообщение, то оно теряется при отправке следующего. Уведомители ведут себя как одноэлементные ограниченные очереди с потерями.

Obtain Notifier



Получить уведомителя



Функция возвращает ссылку на уведомителя. Ссылка используется для вызова других функций уведомителя. Именованные уведомители используются для передачи данных между двумя разделами блок-диаграммы или между двумя ВП.

Вход **имя** (name) содержит имя уведомителя, которого пользователь хочет получить или создать. По умолчанию на вход подается пустая строка, приводящая к созданию неименованного уведомителя.

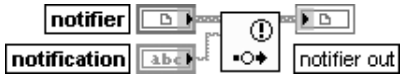
Вход **тип элемента данных** (element data type) представляет тип данных, которые должен содержать уведомитель. К этому входу могут быть подключены данные любого типа.

Вход **создать если не найден?** (create if not found?) определяет возможность создания нового уведомителя, если уведомитель с именем, заданным на входе **имя**, не существует. При установке на входе значения ИСТИНА (по умолчанию) функция создает нового уведомителя и возвращает ссылку на него.

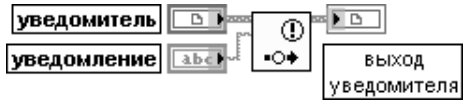
Выход уведомителя (notifier out) определяет ссылку к существующему именованному уведомителю или новому уведомителю, созданному этой функцией.

Выход **создан новый?** (created new?) отображает значение ИСТИНА, если функция создала нового уведомителя

Send Notification



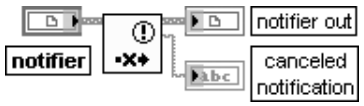
Отправить уведомление



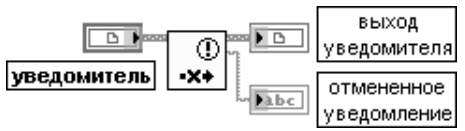
Функция отправляет сообщение всем функциям, ожидающим уведомителя. Все функции **Ожидать уведомления** (Wait on Notification) и **Ожидать уведомления от множества** (Wait on Notification from Multiple), ожидавшие уведомителя, выходят из ожидания и продолжают выполнение.

Вход **уведомление** (notification) содержит отправляемое сообщение. Его тип данных изменяется в соответствии с подтипом **уведомителя** (notifier)

Cancel Notification



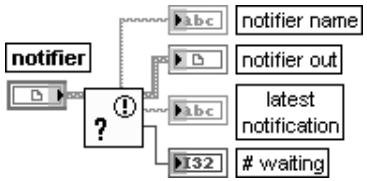
Отменить уведомление



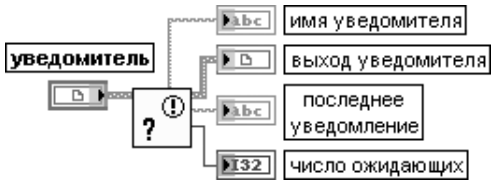
Функция стирает какие-либо сообщения, содержащиеся в уведомителе, и возвращает сообщение об отмене уведомления.

Выход **отмененное уведомление** (cancelled notification) содержит последнее уведомление, отправленное уведомителю. Тип данных выхода изменяется в соответствии с подтипом **уведомителя** (notifier).
Если функции **Ожидать уведомления** (Wait on Notification) или **Ожидать уведомления от множества** (Wait on Notification from Multiple) приняли сообщение перед вызовом этой функции, то данные функции продолжают выполняться. Эта функция не вызывает и не сбрасывает какие-либо функции ожидания. После отмены уведомления пользователем любые последующие функции ожидания ожидают принятия уведомителем другого сообщения. Отмена уведомления перед получением сообщения уведомителем не приводит к ошибке

Get Notifier Status

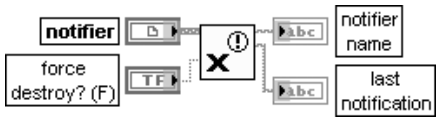


Получить статус уведомителя

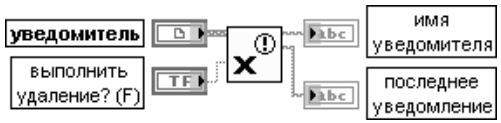


Функция возвращает информацию о текущем состоянии уведомителя, таком как последнее неотмененное уведомление, отправленное уведомителю.
Выход **последнее уведомление** (latest notification) содержит самое последнее неотмененное уведомление, отправленное уведомителю. При отсутствии доступных уведомлений функция возвращает нулевое значение для **типа данных элемента**, подключенного ко входу функции **Получить уведомителя**.
Этот тип данных изменяется в соответствии с подтипом уведомителя.
Выход **число ожидающих** (# waiting) отображает число функций, ожидающих уведомителя. К таким функциям относятся функции **Ожидать уведомления** (Wait on Notification) и **Ожидать уведомления от множества** (Wait on Notification from Multiple)

Release Notifier

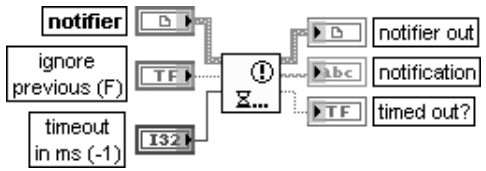


Закрыть уведомителя

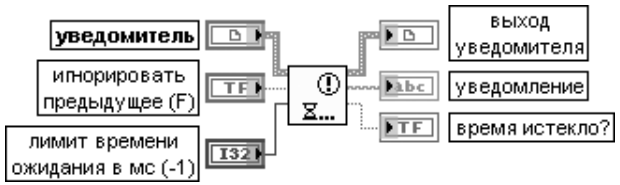


Функция закрывает ссылку к уведомителю.
Вход **выполнить удаление?** (force destroy?) определяет возможность удаления уведомителя. При установке на этом входе значения ЛОЖЬ (по умолчанию) для удаления уведомителя необходимо вызвать данную функцию столько раз, сколько раз была получена ссылка к уведомителю, или остановить все ВП, использующие ссылку к уведомителю. При установке на входе состояния ИСТИНА функция удаляет уведомителя, не требуя многократного вызова функции или остановки всех ВП, использующих ссылку к уведомителю

Wait on Notification

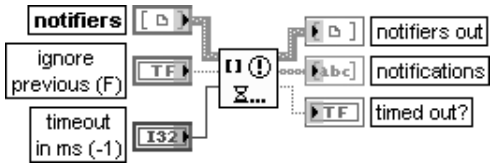


Ожидать уведомления

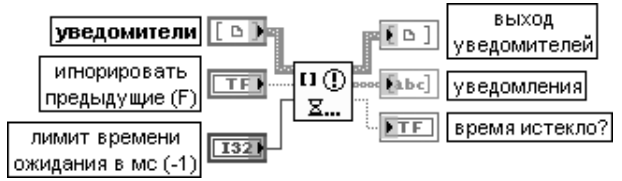


Функция ожидает получения сообщения уведомителем. После получения сообщения уведомителем функция продолжает выполнение.
Для отправки сообщения используется функция **Отправить уведомление** (Send Notification). Если ссылка к уведомителю стала недействительной (например, когда другая функция закрыла ее), то функция прекращает ожидание и возвращает ошибку с кодом 1122. Если уведомитель не содержит сообщение, то эта функция ожидает получения сообщения уведомителем.
Вход **игнорировать предыдущее** (ignore previous) определяет возможность игнорировать сообщения, отправленные уведомителю перед вызовом этой функции. Если при установке на входе состояния ИСТИНА уведомитель содержал сообщение перед вызовом этой функции, то функция ожидает принятия уведомителем другого сообщения. При установке на входе состояния ЛОЖЬ (по умолчанию) при тех же условиях функция продолжает выполнение.
Каждый экземпляр этой функции запоминает отметку времени последнего считанного сообщения. Если на входе **игнорировать предыдущее** (ignore previous) установлено состояние ЛОЖЬ, то каждый экземпляр функции **Ожидать уведомления** (Wait on Notification) ожидает, пока сообщение в уведомителе имеет ту же отметку времени, что и вновь считываемые сообщения. Если сообщение является новым, то функция возвращает сообщение.
Если на входе **игнорировать предыдущее** установлено состояние ИСТИНА, то функции **Ожидать уведомления** всегда ожидает нового сообщения, даже если сообщение, находящееся в уведомителе, никогда перед этим не встречалось.
Эта функция не удаляет сообщение из уведомителя. Последующие вызовы этой функции или функции **Ожидать уведомление от множества** (Wait on Notification from Multiple) блокируют повторные сообщения до вызова функции **Отправить уведомление** (Send Notification) с другим сообщением

Wait on Notification from Multiple



Ожидать уведомления от множества



Функция ожидает получения сообщения по крайней мере одним из заданных уведомителей. После получения сообщения одним из уведомителей функция продолжает выполнение.

Вход **уведомители** (notifiers) представляет массив ссылок к уведомителю.
Выход уведомителей (notifiers out) возвращает массив ссылок к уведомителям, от которых были приняты сообщения. Индексы этого массива соответствуют индексам выходного массива **уведомления** (notifications). Из массива **выход уведомителей** можно установить связь между сообщением и уведомителем.
Выход **уведомления** (notifications) представляет массив последних сообщений, отправленных уведомителям. Тип данных этого выхода изменяется в соответствии с подтипом **уведомителей**, за исключением подтипа массив. Если подтипом является массив, то тип данных становится кластером массивов

На рис. 2.70 приведена блок-диаграмма упрощенного ВП **Уведомитель-демультиплексор** (Notifier Demultiplexer) из набора примеров NI Example Finder. В основе его работы лежит пересылка данных, сформированных в цикле генерации данных, двум регистраторам данных с помощью функций **Операции уведомителя** (Notifier Operations).

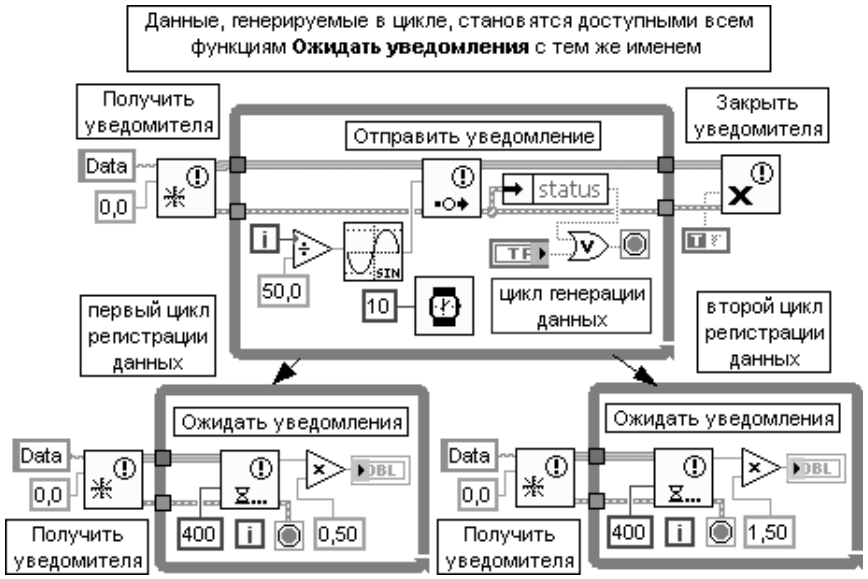
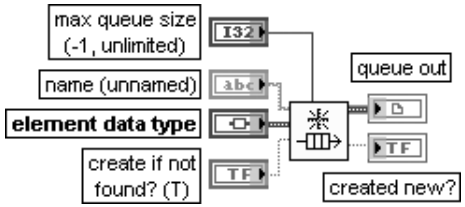


*Рис. 2.70. Блок-диаграмма ВП **Уведомитель-демультиплексор** (Notifier Demultiplexer)*

Операции очереди

В следующей таблице рассмотрены функции **Операции очереди** (Queue Operations). Эти функции используются для накопления данных в очереди с последующим извлечением данных в виде отдельных элементов или массива всех элементов.

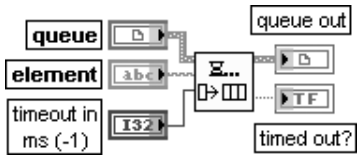
Obtain Queue



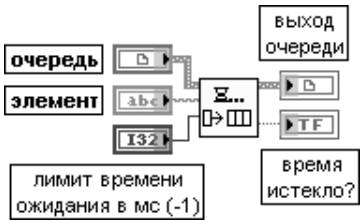
Получить очередь

Функция возвращает ссылку на очередь. Ссылка используется для вызова других функций очереди. Именованные очереди используются для передачи данных между двумя разделами блок-диаграммы или между двумя ВП.
Вход **максимальный размер очереди** (max queue size) задает максимальное число элементов, которое должно содержаться в очереди. По умолчанию это число равно –1, что соответствует неограниченному числу элементов очереди.
Если очередь достигает **максимального размера очереди**, то функции **Поставить элемент в очередь** (Enqueue Element) или **Поставить элемент в очередь на противоположном конце** (Enqueue Element at Opposite End) ожидают, пока функции **Убрать элемент из очереди** (Dequeue Element) или **Сбросить очередь** (Flush Queue) удалят элементы из очереди. Если очередь с тем же именем существует, то этот вход не оказывает влияния на выполнение функции.
Максимальный размер очереди только ограничивает число элементов в очереди.
Он не перераспределяет память. Следовательно, такие типы данных с изменяемым размером, как пути, строки, массивы и т. п., могут еще увеличивать и уменьшать общий размер очереди.
Вход **имя** (name) содержит имя очереди, которую пользователь хочет получить или создать.
По умолчанию это пустая строка, приводящая к созданию неименованной очереди.
Вход **тип элемента данных** (element data type) представляет тип данных, которые должна содержать очередь. К этому входу могут быть подключены данные любого типа.
Вход **создать если не найдена?** (create if not found?) определяет, надо ли создавать новую очередь, если очередь с именем, заданным на входе **имя**, не существует. При установке на входе значения ИСТИНА (по умолчанию) функция создает новую очередь и возвращает ссылку на нее.
Выход очереди (queue out) определяет ссылку к существующей именованной очереди или новой очереди, созданной этой функцией.
Выход **создана новая?** (created new?) отображает значение ИСТИНА, если функция создала новую очередь

Enqueue Element



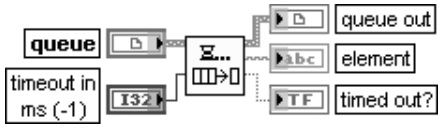
Поставить элемент в очередь



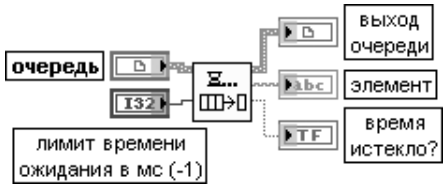
Функция добавляет элемент в конец очереди.
Если очередь заполнена, то функция ожидает в течение времени, заданного на входе **лимит времени ожидания в мс** перед выходом по времени ожидания. Если место в очереди во время ожидания становится доступным, то функция вставляет элемент и выход **время истекло?**

(timed out?) принимает значение ЛОЖЬ. Если очередь становится недостоверной (например, если ссылка к очереди закрыта), то функция прекращает ожидание и возвращает ошибку с кодом 1122. Для установки максимального размера очереди необходимо использовать функцию **Получить очередь** (Obtain Queue).
Вход **элемент** (element) определяет элемент, который добавляется в конец очереди. Его тип данных изменяется в соответствии с подтипом очереди

Preview Queue Element

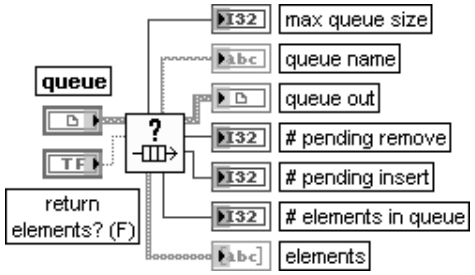


Просмотреть элемент очереди

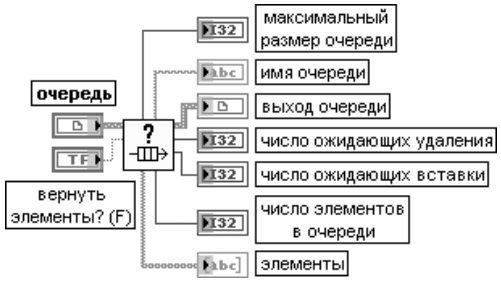


Функция возвращает элемент из начала очереди без его удаления из очереди.
Для возврата элемента из начала очереди и удаления из очереди необходимо использовать функцию **Убрать элемент из очереди** (Dequeue Element). Если очередь пустая, то функция ожидает в течение интервала времени, заданного на входе **лимит времени ожидания в мс** перед выходом по времени ожидания. Если элемент становится доступным в очереди во время ожидания, то функция возвращает элемент и устанавливает на выходе **время истекло?** (timed out?) значение ЛОЖЬ.
Выход **элемент** (element) определяет элемент из начала очереди

Get Queue Status



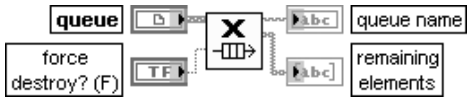
Получить статус очереди



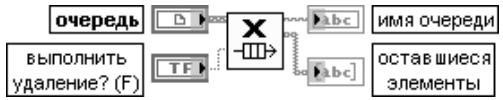
Функция возвращает информацию о текущем состоянии очереди, такую, например, как число элементов, находящихся в очереди. Эту функцию можно использовать также для проверки того, что **очередь** является достоверной ссылкой очереди. Если **очередь** не является достоверной ссылкой очереди, то функция возвращает ошибку с кодом 1.
Вход **вернуть элементы** (return elements?) показывает возможность возврата элементов очереди. При установке значения ЛОЖЬ (по умолчанию) функция не возвращает элементы очереди.
Выход **максимальный размер очереди** (max queue size) отображает максимальное число элементов, которое может содержать очередь. Если **максимальный размер очереди** равен –1, то очередь может содержать любое число элементов.
Выход **число ожидающих удаления** (# pending remove) представляет число функций **Убрать элемент из очереди** (Dequeue Element) или **Просмотреть элемент очереди** (Preview Queue Element), ожидающих удаления элемента из очереди.
Выход **число ожидающих вставки** (**#** pending insert) представляет число функций, ожидающих вставки элемента в очередь. Для вставки элемента в очередь необходимо

использовать функции **Поставить элемент в очередь** (Enqueue Element) или **Поставить элемент в очередь на противоположном конце** (Enqueue Element at Opposite End).
Если **максимальный размер очереди** равен –1, то **число ожидающих вставки** равно 0.
Выход **число элементов в очереди** (**#** elements in queue) возвращает текущее число элементов в очереди.
Выход **элементы** (elements) возвращает все элементы, находящиеся в очереди, но не удаляет их из очереди. Если вход **вернуть элементы?** находится в состоянии ЛОЖЬ, то этот массив является пустым. Тип данных этого выхода изменяется в соответствии с подтипом данных на входе **очередь**

Release Queue

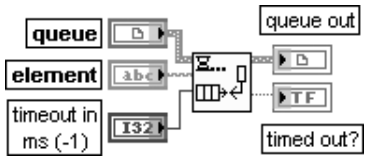


Закрыть очередь

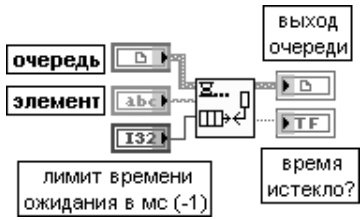


Функция закрывает ссылку к очереди.
Вход **выполнить удаление?** (force destroy?) показывает возможность удаления очереди. При установке значения ЛОЖЬ (по умолчанию) для удаления очереди необходимо вызвать данную функцию столько раз, сколько было получено ссылок к очереди, или остановить все ВП, используя ссылку к очереди. При установке значения ИСТИНА функция удаляет очередь без многократного вызова функции или остановки всех ВП с помощью ссылки к очереди

Enqueue Element At Opposite End

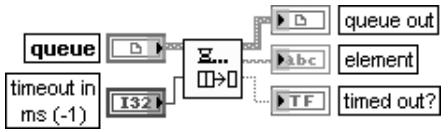


Поставить элемент в очередь на противоположном конце

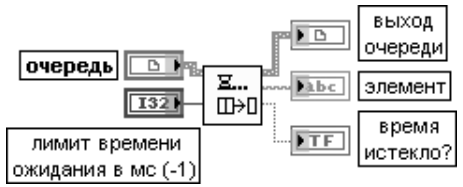


Функция добавляет элемент к началу очереди. Эта функция аналогична функции **Поставить элемент в очередь** (Enqueue Element).
Вход **лимит времени ожидания в мс** (timeout in ms) показывает, сколько миллисекунд функция ожидает доступного места в очереди, если очередь заполнена. По умолчанию значение входа равно –1, что соответствует отсутствию ожидания.
Если функция ожидает в течение заданного **лимита время ожидания в мс** и очередь остается полной, то функция возвращает значение ИСТИНА на выходе **время истекло?**

Dequeue Element

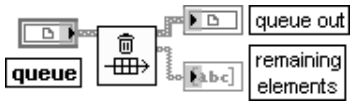


Исключить элемент из очереди

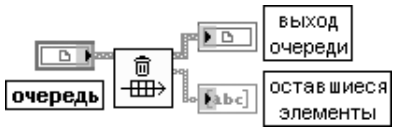


Функция удаляет элемент из начала очереди и возвращает его на выходе **элемент**.
Роль входа **лимит времени ожидания в мс** (timeout in ms) и выходов **элемент** (element) и **время истекло?** (timed out?) идентична роли одноименного входа и выходов функции **Просмотреть элемент очереди** (Preview Queue Element)

Flush Queue



Сбросить очередь



Функция удаляет все элементы из очереди и возвращает удаленные элементы в виде массива. Эта функция не освобождает ссылку к очереди.
Выход **оставшиеся элементы** (remaining elements) представляет массив элементов, удаленных из очереди. Первый элемент в массиве соответствует элементу из начала очереди, а последний элемент массива представляет элемент из конца очереди

На рис. 2.71 приведена блок-диаграмма упрощенного ВП **Очередь-мультиплексор** (Queue Multiplexer) из набора примеров NI Example Finder. В основе его работы лежит пересылка данных, сформированных в двух циклах генерации данных, одному регистратору данных с помощью функций **Операции очереди** (Queue Operations).

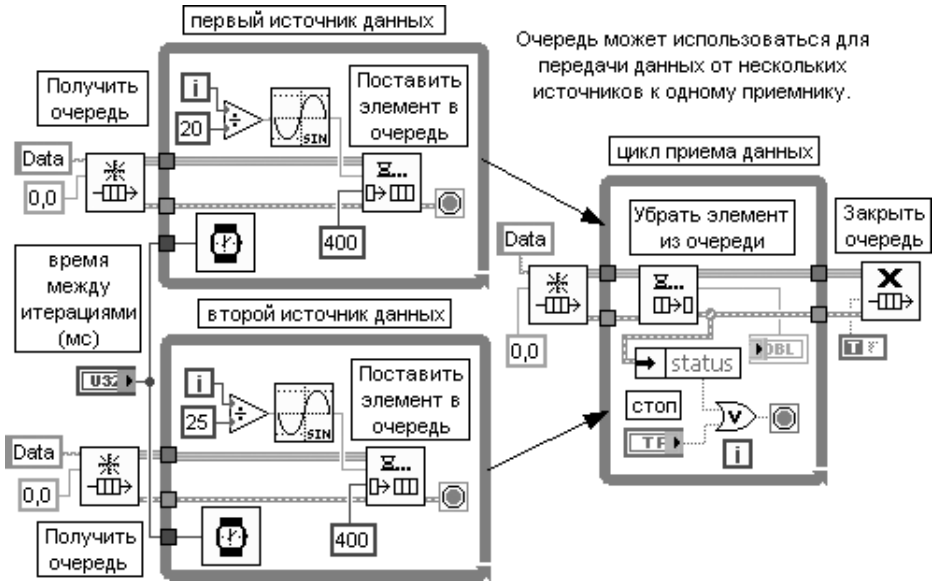


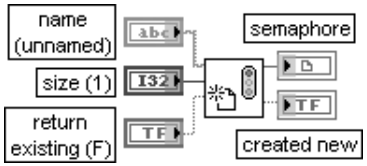
*Рис. 2.71. Блок-диаграмма ВП **Очередь-мультиплексор** (Queue Multiplexer)*

ВП Семафор

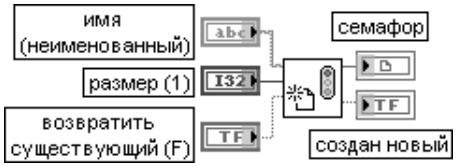
ВП из подпалитры **Семафор** (Semaphore VIs) используются для ограничения числа задач, которые могут одновременно выполняться на общем (защищенном) ресурсе. Защищенный ресурс или критическая секция могут включать запись в глобальные переменные или взаимодействие с внешними приборами.

ВП из подпалитры **Семафор** также могут использоваться для синхронизации двух или более отдельных параллельных задач так, что только одна задача выполняет критическую секцию кода, защищенную общим семафором. В частности, эти ВП необходимо использовать для того, чтобы определенные ВП или части блок-диаграммы ожидали завершения выполнения критических секций другими ВП или частями блок-диаграммы.

Create Semaphore

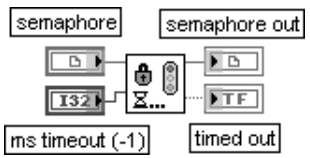


Создать семафор

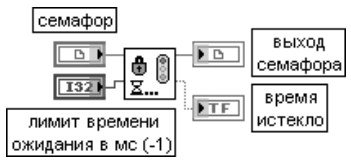


Функция просматривает существующий семафор или создает новый семафор и возвращает ссылку. Эту ссылку можно использовать для вызова других ВП из подпалитры **Семафор** (Semaphore VIs).
Вход **имя** (name) содержит имя семафора, который просматривается или создается. По умолчанию на этот вход подключается пустая строка для создания неименованного семафора. LabVIEW не освобождает автоматически именованные семафоры.
Именованный семафор существует только в течение выполнения ВП верхнего уровня, первым создавшего именованный семафор. Так, например, если ВП **Приложение** имеет подприбор **Подзадача**, то любые именованные семафоры, созданные ВП **Подзадача,** очищаются, когда ВП **Приложение** останавливает выполнение. Можно использовать тот же самый именованный семафор в любом ВП, даже не находящемся в иерархии ВП **Приложение**, но семафор прекратит существование, когда иерархия ВП **Приложение** закроется.
Вход **размер** (size) определяет максимальное число задач, которые могут одновременно получить семафор. Если именованный семафор уже существует, то подключение значения к этому входу не изменяет размера семафора. По умолчанию размер равен 1.
Вход **возвратить существующий** (return existing) определяет возможность создания нового семафора, если семафор с таким именем не существует. По умолчанию значение входа равно ЛОЖЬ, что позволяет создавать семафор, если он не существует.
Выход **создан новый** (created new) отображает значение ИСТИНА, если ВП создает новый семафор

Acquire Semaphore

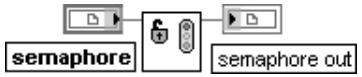


Получить семафор

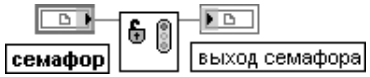


Функция обеспечивает получение доступа к семафору.
Функции входа **лимит времени ожидания в мс** (ms timeout) и выхода **время истекло** (timed out) были рассмотрены выше.
Если семафор уже получен максимальным числом задач, то ВП ожидает в течение интервала **лимит времени ожидания в мс** перед выходом по времени ожидания. Если семафор стал доступен во время ожидания, то на выходе **время истекло** формируется значение ЛОЖЬ. Если семафор не стал доступен или **семафор** не является достоверным, то на этом выходе формируется значение ИСТИНА. Счетчик семафора увеличивается на единицу при каждом выполнении ВП **Получить семафор**, даже если задача, получающая семафор, уже получила его однажды.
Получение того же семафора дважды без промежуточного вызова ВП **Закрыть семафор** (Release Semaphore) в общем случае приводит к искажению данных

Release Semaphore

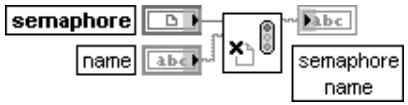


Закрыть семафор

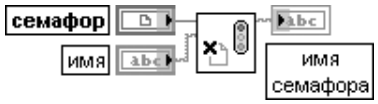


Функция закрывает доступ к семафору. Если ВП **Получить семафор** (Acquire Semaphore) ожидает семафор, который закрывается данным ВП, то он прекращает ожидание и продолжает выполнение

Destroy Semaphore

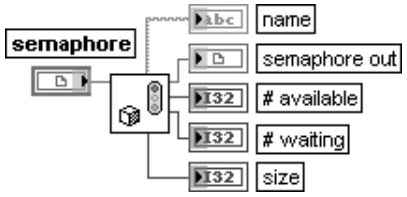


Ликвидировать семафор

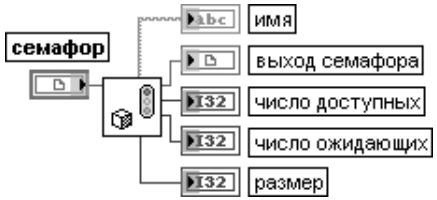


Функция ликвидирует определенный семафор. Все ВП **Получить семафор** (Acquire Semaphore), которые ожидают этот семафор, немедленно выходят из ожидания и возвращают ошибку

Get Semaphore Status



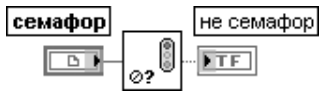
Получить статус семафора



Функция возвращает информацию о текущем статусе семафора.
Выход **имя** (name) содержит имя семафора.
Выход семафора (semaphore out) имеет то же значение, что и вход **семафор**.
Выход **число доступных** (# available) содержит число задач, которые получены семафором на текущий момент. Это число всегда меньше или равно значению на выходе **размер** (size).
Выход **число ожидающих** (# waiting) содержит число функций **Получить семафор** (Acquire Semaphore), ожидающих получения семафора

Not A Semaphore

Не семафор



Функция возвращает значение ИСТИНА, если вход **семафор** (semaphore) является недостоверной ссылкой на семафор

На рис. 2.72 приведены блок-диаграммы упрощенного ВП **Семафор с подприборами** (Semaphore with SubVIs) из набора примеров NI Example Finder. В основе его работы лежит поочередный ввод данных от одного источника данных с помощью двух подприборов (рис. 2.72а). При этом в каждом подприборе (рис. 2.72б) перед вводом данных устанавливается запрет доступа к источнику данных с помощью ВП **Получить семафор**, а после ввода такой запрет снимается с помощью ВП **Закрыть семафор**.

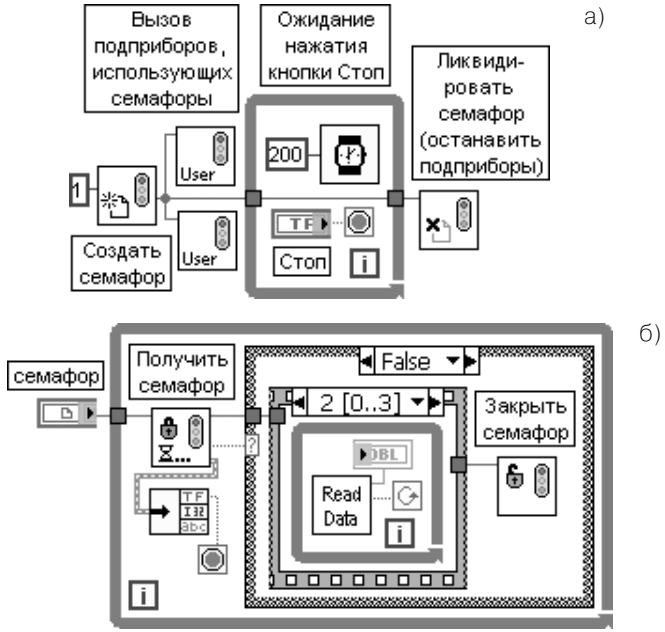


*Рис. 2.72. Блок-диаграммы ВП **Семафор с подприборами** (Semaphore with SubVIs) (а) и подприбора (б)*

ВП Встреча

ВП из подпалитры **Встреча** используются для синхронизации двух или более отдельных параллельных задач в определенной точке выполнения. Каждая задача, которая достигает **встречи**, ожидает, пока в таком же состоянии не окажется заданное число задач, после чего все задачи продолжают выполнение.

Create Rendezvous / Создать встречу

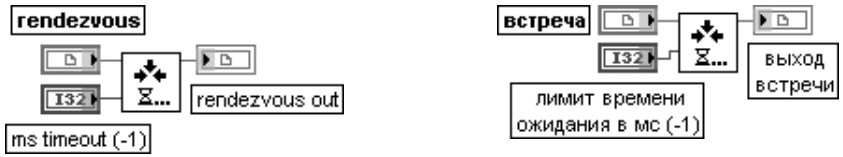
Функция просматривает существующую встречу или создает новую встречу и возвращает ссылку. Эту ссылку можно использовать для вызова других ВП из подпалитры **Встреча** (Rendezvous).

Вход **имя** (name) содержит имя встречи, которая просматривается или создается. По умолчанию на этот вход подключается пустая строка, чтобы создать неименованную встречу. LabVIEW не освобождает автоматически именованные встречи.

Вход **размер** (size) определяет минимальное число задач, которые должны встретиться, чтобы продолжить выполнение. По умолчанию число таких задач равно 2. Если именованная встреча уже существует, то подключение значения ко входу **размер** не изменяет размер встречи. Для изменения размера именованной встречи необходимо использовать ВП **Изменить размер встречи** (Resize Rendezvous).

Вход **вернуть существующую** (return existing) определяет возможность создания новой встречи, если встреча с таким именем не существует. По умолчанию значение входа равно ЛОЖЬ, что позволяет создавать встречу, если она не существует

Wait at Rendezvous / Ожидать встречу



Функция ожидает прибытия на встречу достаточного числа задач.

Вход **лимит времени ожидания в мс** (ms timeout) определяет длительность ожидания прибытия других задач на встречу. По умолчанию это значение равно –1, что показывает отсутствие лимита времени ожидания

Resize Rendezvous / Изменить размер встречи

Функция изменяет размер **встречи** (rendezvous) с помощью входа **изменить размер** (size change) и возвращает **новый размер** (new size).

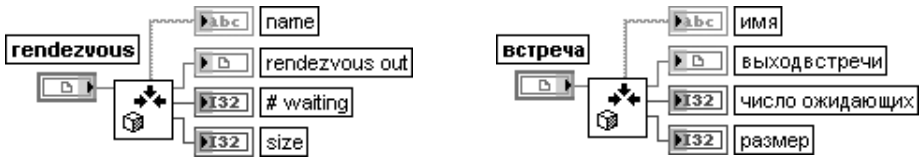
Вход **изменить размер** (size change) определяет максимальное количество задач, которые должны поступить на встречу. Отрицательный размер уменьшает число задач. По умолчанию значение входа равно 0, что не изменяет размер.

Выход **новый размер** (new size) представляет новое число задач, поступающих на встречу

Destroy Rendezvous / Ликвидировать встречу

Функция ликвидирует определенную **встречу**. Все ВП **Ожидать встречу** (Wait at Rendezvous), которые ожидают данную встречу, немедленно выходят из ожидания и возвращают ошибку

Get Rendezvous Status / Получить статус встречи



Функция возвращает информацию о текущем статусе встречи.

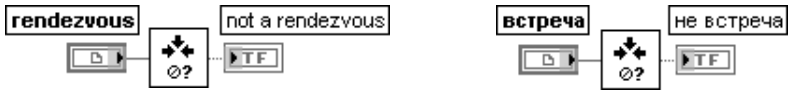
Выход **имя** (name) содержит имя встречи.

Выход встречи (rendezvous out) имеет то же значение, что и вход **встреча**.

Выход **число ожидающих** (**#** waiting) отображает число функций **Ожидать встречу** (Wait at Rendezvous), ожидающих встречи в текущий момент.

Выход **размер** (size) отображает число задач, участвующих во встрече

Not A Rendezvous / Не встреча



Функция возвращает значение ИСТИНА, если **встреча** (rendezvous) не является достоверной ссылкой к встрече

Функции случаев

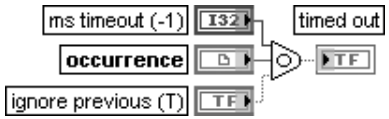
Функции случаев (Occurrences functions) используются для управления отдельными синхронными действиями. В частности, эти функции целесообразно использовать, когда один ВП или часть блок-диаграммы должны ожидать, пока другой ВП или часть блок-диаграммы закончат задачу без принудительного опроса с помощью средств LabVIEW. Эта же задача может быть выполнена с помощью глобальной переменной, когда изменение ее значения определяется с помощью опроса глобальной переменной в структуре цикла. Конечно, использование глобальных переменных с ожиданием в цикле приводит к большим потерям процессорного времени. С использованием **функций случаев** второй цикл становится холостым и не использует процессорное время. Когда первый цикл устанавливает случай, LabVIEW активизирует второй цикл и любые другие блок-диаграммы, которые ожидают определенный случай.

Generate Occurrence

Создать случай

Функция создает **случай** (occurrence), который можно передать к функциям **Ожидать случай** (Wait on Occurrence) и **Установить случай** (Set Occurrence).
Выход **случай** (occurrence) представляет ссылку на случай, которая связывает функции **Ожидать случай** и **Установить случай**.
Обычно только один узел **Создать случай** подключается к какому-либо набору функций **Ожидать случай** и **Установить случай**.
В отличие от других **ВП синхронизации** (Synchronization VIs), каждая функция **Создать случай** на блок-диаграмме представляет простой, уникальный **случай**. В этом отношении функция **Создать случай** аналогична константе. При выполнении ВП вызов функции **Создать случай** формирует одно и то же значение. Например, если функция **Создать случай** устанавливается в структуре цикла, то она выводит одинаковое значение при каждой итерации. Если функция **Создать случай** устанавливается на блок-диаграмме реентерабельного ВП, то она создает различное значение при каждом вызове.
National Instruments рекомендует использовать функции **Операции уведомителя** (Notifier Operations) вместо **случаев** для большинства операций

Wait on Occurrence



Ожидать случай

Функция ожидает функцию **Установить случай** (Set Occurrence) для установки или запуска заданного **случая** (occurrence).
Вход **лимит времени ожидания в мс** (ms timeout) определяет заданный интервал времени в мс, отводимый для случая. Если случай не происходит в отведенное **время ожидания**, то функция возвращает значение ИСТИНА. Если на входе **лимит времени ожидания** установлено значение –1, то ожидание отсутствует.
Вход **случай** (occurrence) представляет ссылку, которая связывает функции **Ожидать случай** (Wait on Occurrence) и **Установить случай** (Set Occurrence).
Если вход **игнорировать предыдущий** (ignore previous) установлен в состояние ИСТИНА и другой узел установил случай перед началом выполнения этой функции, то функция игнорирует предыдущий случай и ожидает другой случай.
Выход **время истекло** (timed out) имеет значение ИСТИНА, если случай не произошел за определенный **лимит времени ожидания**. Если **лимит времени ожидания** равен –1, то на выход **время истекло** выводится значение ЛОЖЬ

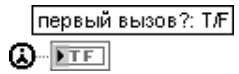
Set Occurrence

Установить случай

Функция запускает определенный **случай** (occurrence). Все блок-диаграммы, которые ожидают данный случай, прекращают ожидание.
Вход **случай** (occurrence) является ссылкой к случаю, которая связывает функции **Ожидать случай** и **Установить случай**

First Call?

Первый вызов?



Функция показывает, что подприбор (subVI) или секция блок-диаграммы выполняется в первый раз. Функция **Первый вызов?** (First Call?) возвращает значение ИСТИНА только при первом вызове, таком как вызов после нажатия кнопки **Выполнить** (Run). Функцию можно разместить в разных частях блок-диаграммы.
Эта функция полезна для однократного выполнения подприбора или секции блок-диаграммы внутри цикла или структуры **Вариант** (Case structure)

2.2.4. Функции преобразования и отображения графических файлов

Функции преобразования и отображения графических файлов размещены в подпалитрах **Функции рисунков** (Picture Functions) (рис. 2.73б) и **Графические форматы** (Graphics Formats) (рис. 2.73в), которые, в свою очередь, входят в состав палитры **Графики и звук** (Graphics & Sound) (рис. 2.73а). Они позволяют записывать и читать файлы в форматах .jpeg, .png и .bmp. Преобразование двумерных массивов данных в кластер данных изображения осуществляется с помощью ВП **Перевести массив данных в изображение** (Flatten Pixmap), обратное преобразование – с помощью ВП **Восстановить изображение из кластера** (Unflatten Pixmap). Данные изображения могут быть выведены для просмотра в виде изображения с помощью ВП **Рисовать приведенное изображение** (Draw Flattened Pixmap).

Помимо перечисленных, в состав палитры **Графики и звук** входят подпалитры **Свойства трехмерных графиков** (3D Graph Properties), **Графики рисунков** (Pic-

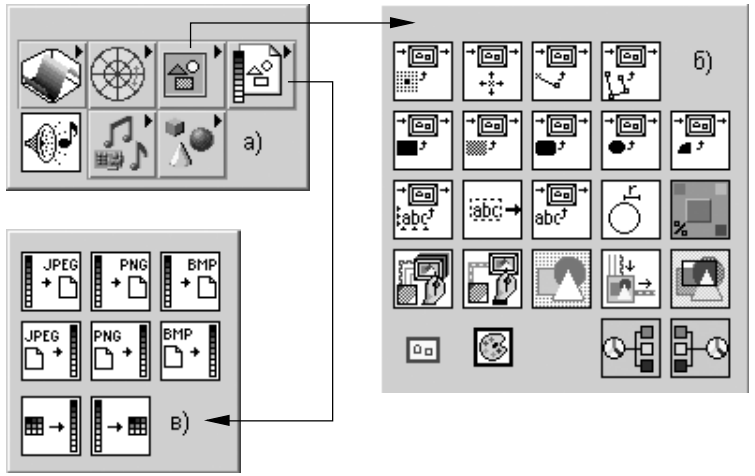
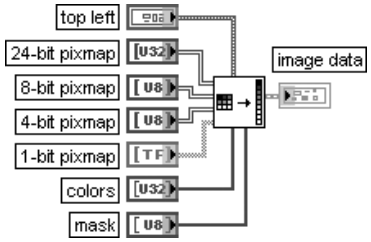


*Рис. 2.73. Вид палитры **Графики и звук** (Graphics & Sound) (а) и ее подпалитр (б), (в)*

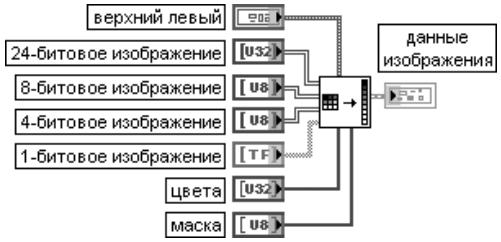
ture Plots), **Управление трехмерным рисунком** (3D Picture Control), рассмотрение которых выходит за пределы справочника.

Описание функций преобразования и отображения графических файлов приведено в таблице.

Flatten Pixmap



Перевести массив данных в изображение



ВП преобразует один из подключенных массивов данных в кластер **данные изображения** (image data), который с помощью ВП из подпалитры **Графические форматы** (Graphics Formats), описанных ниже, может быть записан в файл или с помощью ВП низкого уровня выведен для просмотра в виде изображения.

Вход **верхний левый** (top left) определяет координаты точки, в которой размещается левый верхний угол изображения. В состав кластера **верхний левый** входят следующие элементы:

- **x** представляет горизонтальную координату, которая увеличивается слева направо;
- **y** представляет вертикальную координату, которая увеличивается сверху вниз.

Входы **24-битовое изображение** (24-bit pixmap), **8-битовое изображение** (8-bit pixmap), **4-битовое изображение** (4-bit pixmap) и **1-битовое изображение** (1-bit pixmap) представляют двумерные массивы данных, которые должны быть преобразованы в **данные изображения** (image data). Размеры данных изображения должны соответствовать размерам этих массивов.

В случае преобразования **8-битового изображения** и **4-битового изображения** ВП использует их данные как индексы массива цветов. При преобразовании **1-битового изображения** элементы ЛОЖЬ отображаются элементом 0 таблицы цветов, а элементы ИСТИНА – элементами 1 этой таблицы.

Вход **таблица цветов** (colors) представляет массив значений цветов RGB, которые соответствуют значениям подключенного **входа изображения** (pixmap). Тип подключенного **входа изображения** определяет характер интерпретации LabVIEW этого входа. При подключении **24-битового изображения** LabVIEW игнорирует этот вход. В случае подключения **8-битового изображения** массив может иметь 256 элементов. При подключении **4-битового изображения** массив может иметь 16 элементов. В случае подключения **1-битового изображения** массив может иметь 2 элемента.

Вход **маска** (mask) является одномерным массивом, который описывает информацию о маскировании каждого пиксела.

Выход **данные изображения** (image data) возвращает информацию об изображении в виде кластера, в состав которого входят следующие элементы:

- **тип изображения** (image type) зарезервирован для последующих приложений;
- **глубина изображения** (image depth) задает глубину цвета изображения, которая определяется числом битов, используемых для описания цвета каждого элемента изображения. Допустимые значения включают 1, 4, 8, и 24 битов на элемент. **Глубина изображения** влияет на то, как LabVIEW интерпретирует значения массивов **изображение** (image) и **цвета** (colors);
- **изображение** (image) представляет массив байтов, которые описывают цвет каждого элемента растрового изображения.

Если **глубина изображения** равна 24, то цвет каждого элемента описывается с помощью трех байтов. Первый байт каждого элемента описывает величину красного цвета, второй байт – зеленого, третий байт – синего.

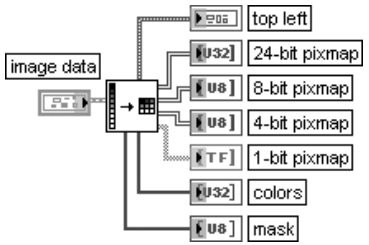
Если **глубина изображения** равна 8, то цвет каждого элемента описывается одним байтом. Значение каждого бита соответствует элементу в массиве **цветов**, который хранит 32-битовые значения RGB, где старший байт равен 0, а байты в порядке убывания соответствуют красному, зеленому и синему цветам. Допустимые значения **изображения** находятся в диапазоне от 0 до 255.

Если **глубина изображения** равна 4, то цвет каждого элемента описывается также одним байтом, однако допустимые значения **изображения** находятся в диапазоне от 0 до 15.

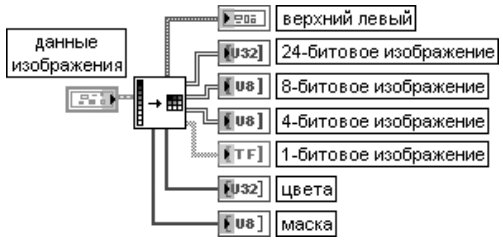
Если **глубина изображения** равна 1, нулевые значения в **изображения** соответствуют нулевому элементу в массиве **цветов**. Все другие значения соответствуют элементу 1 в массиве **цвета**. Размер массива может быть больше ожидаемого вследствие заполнения;

- **маска** (mask) представляет массив байтов, каждый бит которых описывает информацию о маскировании каждого элемента. Первый байт описывает маскирование восьми первых элементов, второй байт – восьми вторых и т. д. Если бит имеет нулевое значение, то LabVIEW рисует соответствующий элемент как прозрачный. Если массив пустой, LabVIEW рисует все элементы непрозрачными;
- **прямоугольник** (Rectangle) представляет кластер, который содержит координаты, задающие прямоугольные границы изображения. Горизонтальные координаты возрастают слева направо, вертикальные – сверху вниз. В состав кластера входят следующие элементы:
  - **left** представляет горизонтальную координату левого края прямоугольника;
  - **top** представляет вертикальную координату верхнего края прямоугольника;
  - **right** представляет горизонтальную координату правого края прямоугольника;
  - **bottom** представляет вертикальную координату нижнего края прямоугольника

Unflatten Pixmap



Восстановить изображение из кластера



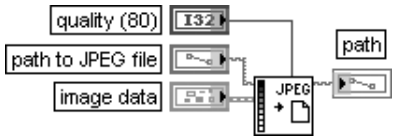
ВП преобразует кластер **данных изображения** в двумерный массив.

Данный ВП целесообразно использовать на выходе одного из ВП подпалитры **Графические форматы** (Graphics Formats), выполняющих чтение графических файлов, для преобразования данных изображения в двумерное представление.

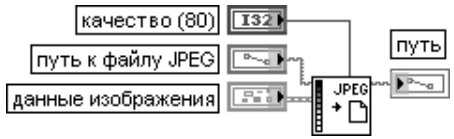
Если на вход ВП подаются 32-битовые данные изображения, то ВП удаляет альфа-канал и возвращает 24-битовое изображение.

Назначение выходов ВП идентично назначению входов описанного выше ВП **Перевести массив данных в изображение** (Flatten Pixmap)

Write JPEG File



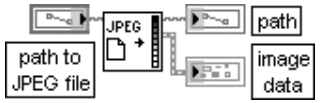
Записать в файл JPEG



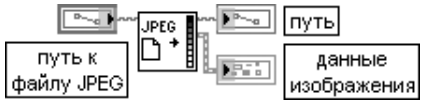
ВП производит запись **данных изображения** (image data) в файл в формате JPEG. При записи изображения в файл необходимо использовать ВП **Перевести массив данных в изображение** (Flatten Pixmap) для преобразования данных в кластер **данных изобра-**

жения перед использованием этого ВП. При записи изображения в файл необходимо использовать ВП **Рисунок в изображение** (Picture to Pixmap) для преобразования данных в кластер **данных изображения** перед использованием этого ВП.
Вход **качество** (quality) определяет уровень качества JPEG, используя шкалу библиотеки IJG JPEG, которая имеет диапазон от 0 до 100. По умолчанию значение равно 80. Шкала балансирует качество изображения и размер файла. Значение в диапазоне 75–95 создает сжатый файл с высоким качеством изображения, а значение менее 50 создает меньший по размеру файл с низким по качеству изображением.
Вход **путь к файлу JPEG** (path to JPEG file) определяет путь и имя файла JPEG, в который производится запись. Если путь не задан, то LabVIEW отображает окно файлового диалога, с помощью которого пользователь может указать путь к файлу.
Вход **данные изображения** (image data) описывает изображение, которое должно быть записано в файл.
Выход **путь** (path) определяет путь к файлу JPEG

Read JPEG File

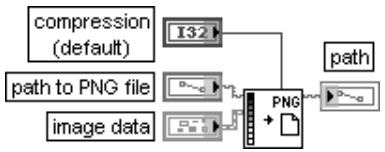


Читать файл JPEG



ВП читает файл JPEG и формирует **данные изображения** (image data), необходимые для отображения файла на индикаторе рисунка.
Функции входа **путь к файлу JPEG** (path to JPEG file) и выхода **путь** (path) идентичны функциям одноименного входа и выхода рассмотренного выше ВП **Записать в файл JPEG**

Write PNG File



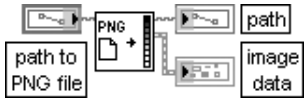
Записать в файл PNG

ВП производит запись файла в формате PNG.
Вход **сжатие** (compression) определяет требуемый уровень сжатия файла PNG. Допустимые значения изменяются в диапазоне от –1 до 9, обеспечивая баланс между сжатием файла и скоростью. Примеры значений входа **сжатие** приведены в табл. 3.30.1.

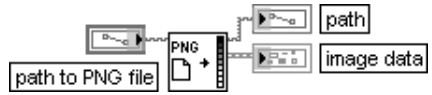
–1	Хорошее сжатие и скорость (по умолчанию)
0	Без сжатия
1	Наилучшая скорость со сжатием
9	Наилучшее сжатие

Вход **путь к файлу PNG** (path to PNG file) определяет путь и имя файла PNG, в который производится запись. Если путь не задан, то LabVIEW отображает окно файлового диалога, с помощью которого пользователь может указать путь к файлу.
Вход **данные изображения** (image data) описывает изображение, которое должно быть записано в файл PNG

Read PNG File

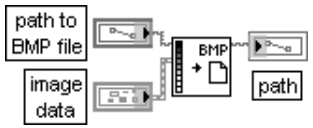


Читать файл PNG

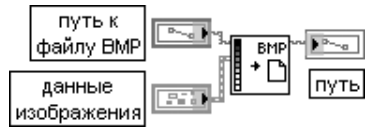


ВП читает файл PNG и формирует данные изображения, необходимые для отображения файла на индикаторе рисунка.
Функции входа **путь к файлу PNG** (path to PNG file) и выходов **путь** и **данные изображения** (image data) идентичны функциям одноименного входа и одноименных выходов рассмотренного выше ВП **Читать файл JPEG**

Write BMP File

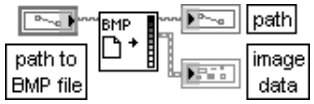


Записать в файл BMP

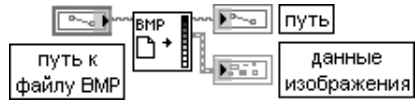


ВП производит запись файла в формате BMP.
Функции входов **путь к файлу BMP** (path to BMP file) и **данные изображения** (image data), а также выхода **путь** идентичны функциям одноименных входов и выхода рассмотренного выше ВП **Записать в файл JPEG**

Read BMP File

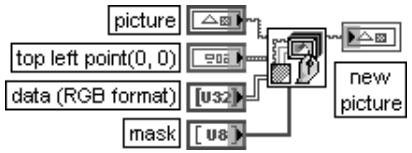


Читать файл BMP



ВП читает файл BMP и формирует данные изображения, необходимые для отображения файла на индикаторе рисунка.
Функции входа **путь к файлу BMP** (path to BMP file) и выходов **путь** и **данные изображения** (image data) идентичны функциям одноименного входа и одноименных выходов рассмотренного выше ВП **Читать файл JPEG**

Draw Unflattened Pixmap

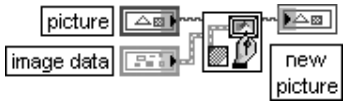


Рисовать восстановленное изображение

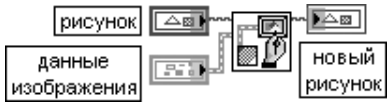
ВП преобразует изображение в рисунок, что позволяет использовать другие ВП из подпалитры **Функции рисунка** (Picture Functions) для добавления инструкций рисования элементов на изображении.
Данный полиморфный ВП позволяет преобразовывать 1-битовые, 4-битовые, 8-битовые или полноцветные изображения. Тип данных, подключенных ко входу **данные**, определяет тип выбираемого ВП из набора ВП, обеспечивающего его полиморфность. При необходимости

преобразования 4-битовых или 8-битовых изображений необходимо выбрать тип ВП с помощью строки **выбрать тип** (Select Type) контекстного меню иконки ВП.
Вход **рисунок** (picture) передает рисунок, к которому необходимо добавить изображение. По умолчанию это пустой рисунок.
Назначение входа **верхняя левая точка** (top left point) идентично назначению входа **верхний левый** (top left) ВП **Перевести массив данных в изображение** (Flatten Pixmap), рассмотренного выше. Аналогичное соответствие существует и между входами **маска** данных ВП.
Вход **данные** (data) представляет двумерный массив 32-битовых целых чисел без знака, которые описывают цвет каждого элемента (пиксела) растрового изображения. Цвет каждого пиксела описывается с помощью трех байтов. Первый байт каждого пиксела описывает величину красного цвета, второй байт – зеленого и третий байт – синего.
Выход **новый рисунок** (new picture) отображает рисунок, содержащий новое изображение. Данный выход может быть подключен к любому входу **рисунок** с целью помещения соответствующих элементов на изображение

Draw Flattened Pixmap



Рисовать приведенное изображение



ВП выводит 1-, 4- или 8-битовое изображение или 24-битовое RGB изображение на индикатор рисунка. Этот ВП берет одномерный массив байтов, предполагая, что пользователь выполнил все операции упаковки и заполнения

Picture to Pixmap

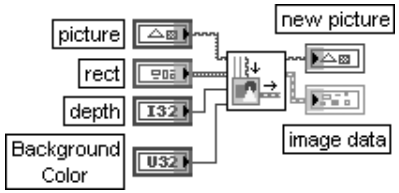
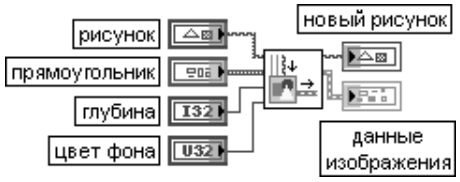
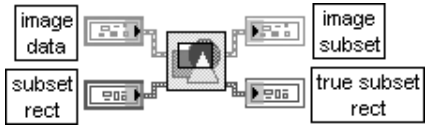


Рисунок в изображение

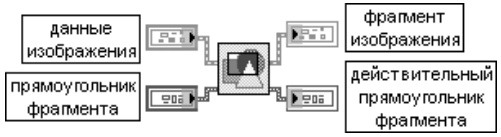


ВП преобразует рисунок в кластер данных изображения, который может быть далее сохранен в файле с помощью ВП из подпалитры **Графические форматы** (Graphics Formats).
Вход **фоновый цвет** (Background Color) устанавливает фоновый цвет изображения

Get Image Subset

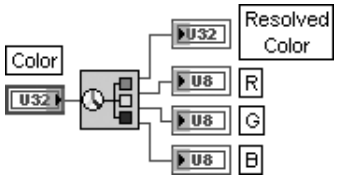


Получить фрагмент изображения

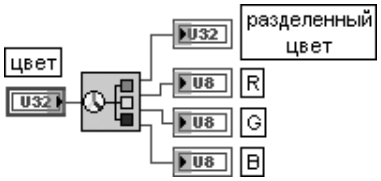


ВП возвращает часть исходного изображения.
Вход **прямоугольник фрагмента** (subset rect) представляет кластер, который содержит координаты прямоугольника, ограничивающего часть изображения. Если координаты не соответствуют допустимым значениям, то ВП преобразует их в допустимые значения и возвращает в кластере **действительный прямоугольник фрагмента** (true subset rect)

Color to RGB

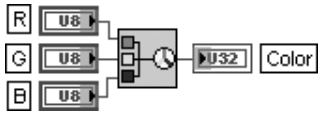


Цвет в RGB

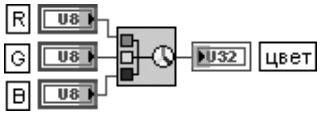


ВП разделяет цвета, в том числе системные, на их компоненты – на красный, зеленый и синий цвета. Необходимость в разделении цветов появляется при решении задач цветовой арифметики.
На вход цвет (Color) подается преобразуемый цвет. Выход **разделенный цвет** (resolved color) отображает разделяемый цвет в RGB-формате.
Выходы **R**, **G**, **B** отображают соответственно красную, зеленую и синюю компоненту значения RGB в диапазоне от 0 до 255

RGB to Color



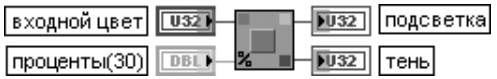
RGB в цвет



ВП преобразует значения красного, зеленого и синего цветов, находящиеся в диапазоне от 0 до 255, в соответствующий RGB цвет

Hilite Color

Изменить насыщенность цвета



ВП возвращает два новых цвета, измененных в сторону увеличения и уменьшения насыщенности относительно исходного цвета в соответствии с установленным процентным значением. Измененные цвета можно использовать для построения затененных трехмерных объектов.
Вход цвета (Color In) задает входной цвет, на базе которого возвращаются более насыщенный и менее насыщенный цвета. К этому входу может быть подключена цветовая константа.
Вход **проценты** (Percentage) определяет степень увеличения и уменьшения насыщенности новых цветов. По умолчанию значение на входе равно 30%.
Выход **подсветка** (Hilite) отображает новый цвет, отличающийся большей насыщенностью.
Выход **тень** (Shadow) отображает новый цвет, отличающийся меньшей насыщенностью

На рис. 2.74 в качестве примера использования функций преобразования и отображения графических файлов приведена блок-диаграмма упрощенного ВП **Управление рисунком – Атрибут увеличения (**Picture Control – Zoom Attribute**)** из библиотеки примеров NI Example Finder. В процессе работы данного ВП пользователь может изменять величину коэффициента увеличения изображения рисунка с помощью соответствующего числового элемента управления.

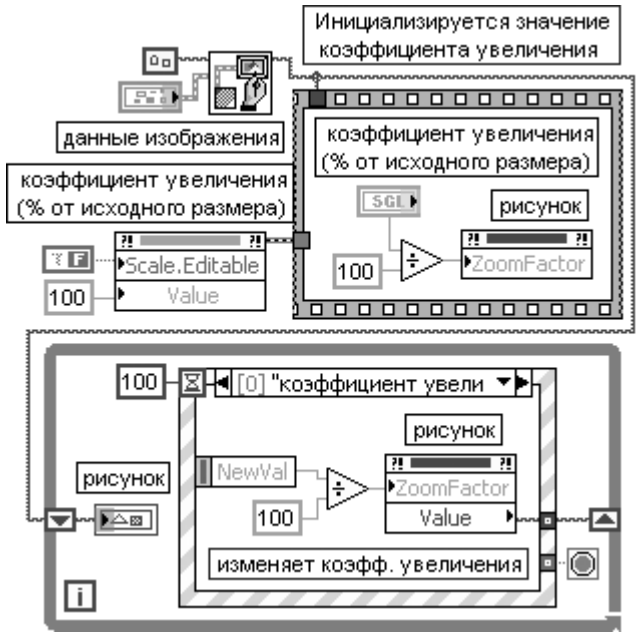
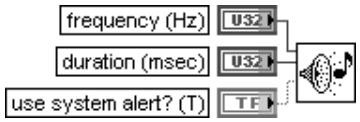


Рис. 2.74. Блок-диаграмма ВП ***Управление рисунком – Атрибут увеличения*** *(Picture Control – Zoom Attribute)*

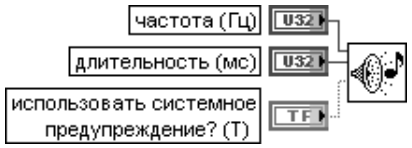
2.2.5. Функции записи и воспроизведения звуковых сигналов

Функции записи и воспроизведения звуковых сигналов позволяют считывать (вводить) сигнал со входа звуковой карты в массив данных и записывать (выводить) сигнал из массива данных на выход звуковой карты, записывать данные в звуковой файл с расширением .wav или считывать такой файл, а также воспроизводить звуковой файл с расширением .wav на выходе звуковой карты. В палитру функций (рис. 2.75а) входят подпалитры функций **вывода** (Output) (рис. 2.75б), **ввода** (Input) (рис. 2.75в) и функции записи-считывания звуковых файлов (File) (рис. 2.75г). В каждую подпалитру входят функции высокого и низкого уровней.

Beep



Звуковой сигнал



ВП вызывает воспроизведение системой звукового сигнала.
Вход **частота (Гц)** (frequency (Hz)) определяет частоту звука в герцах.

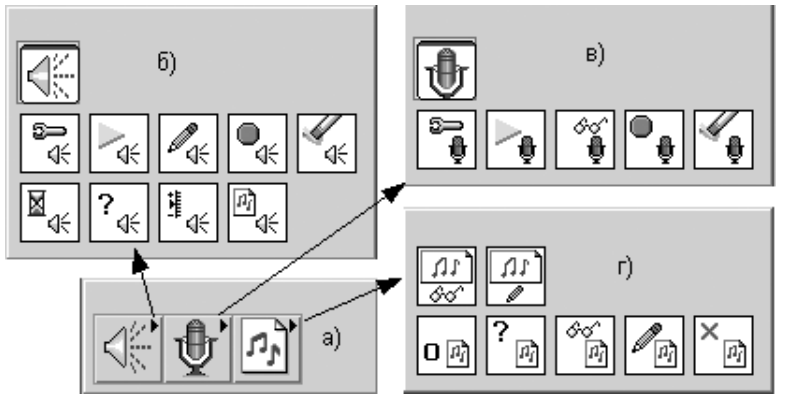


Рис. 2.75. Вид основной палитры (а) и дополнительных подпалитр (б) – (г) функций записи и воспроизведения звуковых сигналов

Вход **длительность (мс)** (duration (msec)) задает длительность звука в миллисекундах. LabVIEW игнорирует эти параметры, если на входе **использовать системное предупреждение?** (use system alert?) установлено состояние ИСТИНА (по умолчанию).
При установке на входе **использовать системное предупреждение?** значения ЛОЖЬ этот ВП формирует синхронный звуковой сигнал, используя заданные частоту и длительность

Воспроизвести осциллограмму (Play Waveform)

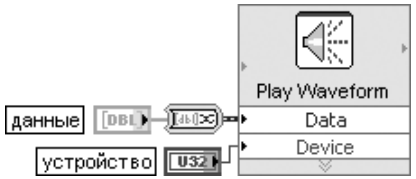


Рис. 2.76. Вариант возможного подключения Экспресс-ВП

Рис. 2.77. Вид диалогового окна конфигурирования Экспресс-ВП ***Воспроизвести осциллограмму*** *(Play Waveform)*

Экспресс-ВП воспроизводит конечную выборку данных на выходе звукового устройства. Он автоматически конфигурирует задачу вывода и освобождает ее после завершения вывода. Вид диалогового окна конфигурирования Экспресс-ВП показан на рис. 2.77. При нажатии кнопки **проверить устройство** (Test Device) на выходе звукового устройства воспроизводится сигнал с частотой 500 Гц в течение приблизительно 1/4 с.
Если с помощью контекстного меню иконки Экспресс-ВП перейти от Экспресс-ВП к стандартному подприбору (SubVI) (строка Open Front Panel), а затем, последовательно открывая лицевые

панели и переходя к блок-диаграммам подприборов, дойти до блок-диаграммы ВП Sound Output Write Simple (DBL2) (рис. 2.78), то можно увидеть, что она содержит последовательно соединенные элементарные функции управления звуковым устройством, входящие в состав подпалитры **Вывод** (Output). Ниже приведены пояснения к исполнению этих функций.

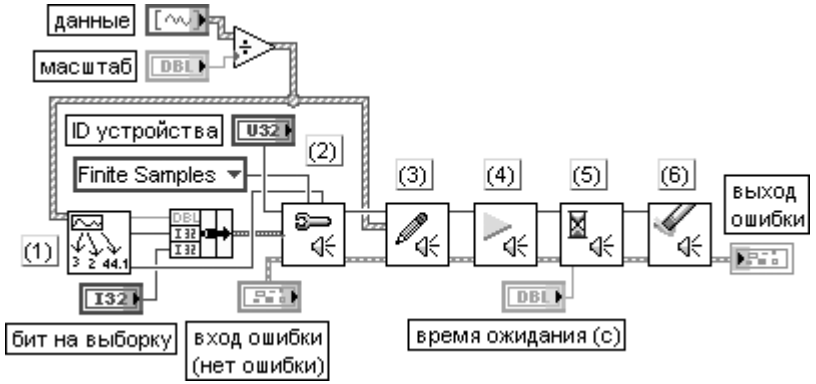


Рис. 2.78. Блок-диаграмма ВП Sound Output Write Simple (DBL2)

1) ВП FormatFromData (DBL) выделяет из массива осциллограмм первую и извлекает из нее информацию о частоте и объеме выборок, а также о числе каналов.
2) ВП **Конфигурировать выходное звуковое устройство** (Sound Output Configure) конфигурирует выходное звуковое устройство для генерации данных.
3) ВП **Записать данные в выходное звуковое устройство** (Sound Output Write (DBL)) записывает данные в звуковое устройство. ВП является полиморфным.
4) ВП **Начать вывод данных выходного звукового устройства** (Sound Output Start) начинает воспроизведение данных. Этот ВП необходим, если только перед этим был вызван ВП Sound Output Stop.
5) ВП **Ожидать завершения вывода данных** (Sound Output Wait) ожидает завершения вывода всего звукового колебания на выходе звукового устройства.
6) ВП **Освободить выходное звуковое устройство** (Sound Output Clear) останавливает воспроизведение звука устройством, очищает буфер, возвращает задачу в начальное состояние и освобождает ресурсы, связанные с задачей. Задача становится неактивной

Sound Output Info

Информация о выходном звуковом устройстве

ВП возвращает информацию о текущем состоянии выходного звукового устройства.
Выход **воспроизводится?** (is playing?) показывает состояние воспроизведения выходного звукового устройства.
Выход **громкость** (volume) возвращает громкость звуковой операции, одно значение на канал. 0 соответствует отсутствию звука, а 100 – максимальной громкости

Sound Output Set Volume

Установить громкость выходного звукового устройства

ВП устанавливает громкость на выходе звукового устройства. ВП является полиморфным и имеет опции установки громкости для одного или двух каналов.
Вход **громкость** (volume) устанавливает громкость звуковой операции, одно значение на канал. 0 соответствует отсутствию звука, а 100 – максимальной громкости

Play Sound File

Воспроизвести звуковой файл

ВП открывает файл и начинает немедленное его воспроизведение.
Вход **время ожидания (с)** (timeout (sec)) определяет максимальную длительность ожидания в секундах выполнения звуковой операции. Этот ВП возвращает ошибку при истечении времени. По умолчанию значение на этом входе равно 10. При установке –1 ВП ожидает неопределенно долго. При установке 0 ВП немедленно возвращает ID задачи, в то время как звук продолжает воспроизводиться. Для ожидания окончания воспроизведения можно использовать ВП **Ожидать завершения вывода данных** (Sound Output Wait).
Выход **ID задачи** (task ID) возвращает идентификационный номер, связанный с конфигурацией определенного устройства. Этот номер может быть передан другим ВП для выполнения звуковых операций.
Блок-диаграмма ВП приведена на рис. 2.79.

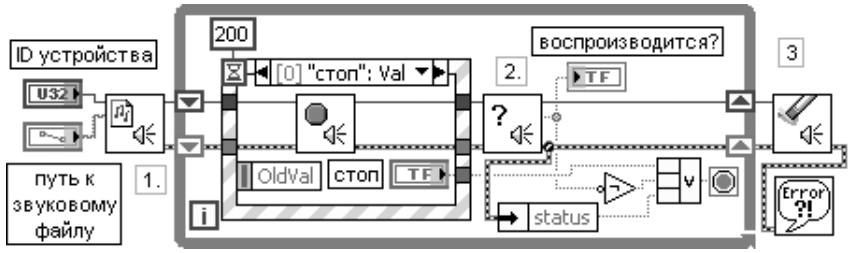


*Рис. 2.79. Блок-диаграмма ВП **Воспроизвести звуковой файл***

1. Начинает немедленное воспроизведение звукового файла.
2. Каждые 200 мс производит проверку воспроизведения и при обнаружении прекращения воспроизведения останавливает цикл
3. Очищает звуковой выход и сообщает о любой ошибке

Ниже приведены краткие описания функций из подпалитры **ввод**.

Получить звуковое колебание (Acquire Sound)

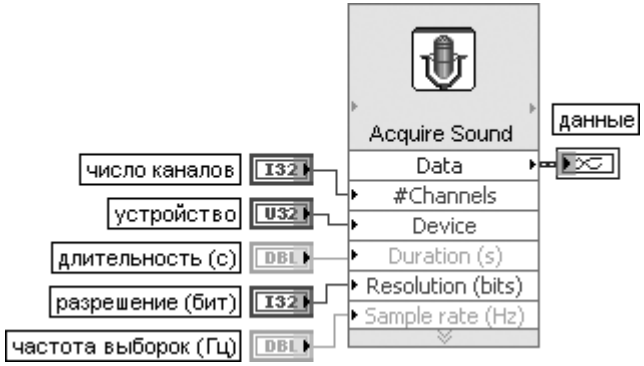


Рис. 2.80. Вариант возможного подключения Экспресс-ВП

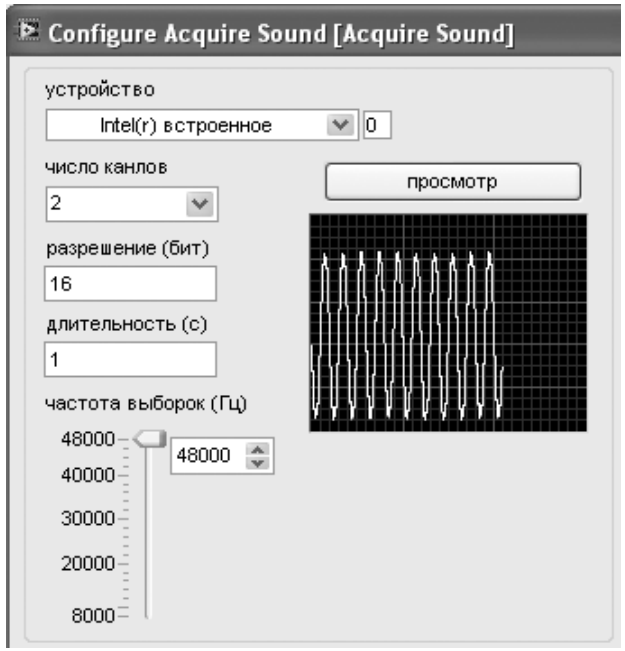


Рис. 2.81. Вид диалогового окна конфигурирования Экспресс-ВП ***Получить звуковое колебание*** *(Acquire Sound)*

Экспресс-ВП получает данные из звукового устройства. Он автоматически конфигурирует задачу ввода, получает данные и освобождает задачу после завершения ввода. Вид диалогового окна конфигурирования Экспресс-ВП показан на рис. 2.81.

Если с помощью контекстного меню иконки Экспресс-ВП перейти от Экспресс-ВП к стандартному подприбору (SubVI) (строка Open Front Panel), а затем, последовательно открывая лицевые панели и переходя к блок-диаграммам подприборов, дойти до блок-диаграммы ВП subSoundAcquire (рис. 2.82), то можно увидеть, что она содержит последовательно соединенные элементарные функции управления звуковым устройством, входящие в состав подпалитры **Ввод** (Input). Ниже приведены пояснения к исполнению этих функций.

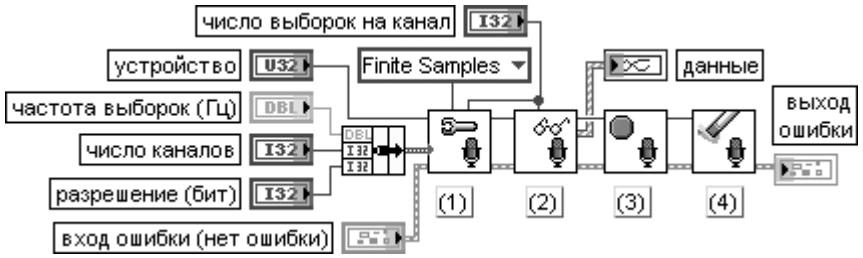


Рис. 2.82. Блок-диаграмма ВП subSoundAcquire

1) ВП **Конфигурировать входное звуковое устройство** (Sound Input Configure) конфигурирует входное звуковое устройство для сбора и передачи данных в буфер.
2) ВП **Считать данные входного звукового устройства** (Sound Input Read (DBL)) считывает данные из входного звукового устройства. Пользователь должен выбрать необходимую реализацию полиморфного ВП.
3) ВП **Остановить входное звуковое устройство** (Sound Input Stop) останавливает сбор данных из устройства. Для повторного запуска сбора данных после выполнения этого ВП необходимо использовать ВП Начать накопление данных входного звукового устройства (Sound Input Start).
4) ВП **Освободить входное звуковое устройство** (Sound Input Clear) останавливает сбор данных, очищает буфер, возвращает задачу в начальное состояние и освобождает ресурсы, связанные с задачей. Задача становится неактивной

Для создания и извлечения файлов с расширением .wav служат ВП из подпалитры Файлы (Files).

Sound File Write Simple — **Простая запись звукового файла**

ВП записывает данные из массива осциллограмм в файл с расширением .wav. Этот ВП автоматически открывает, производит запись и закрывает wav-файл.

Вход **путь** (path) определяет абсолютный путь к звуковому файлу. ВП возвращает ошибку, если путь относительный или недостоверный. По умолчанию на вход подается <**Не путь**> (<Not A Path>).

Вход **данные** (data) записывает любые звуковые данные во внутренние буферы. Каждый элемент массива осциллограмм представляет кластер, содержащий следующие элементы:

- **t0** – игнорируется;
- **dt** – интервал выборки данных звукового файла;
- **Y** – звуковые данные.

Вход **бит на выборку** (bits per sample) устанавливает качество каждой выборки в битах. Возможна установка 8 бит и 16 бит (по умолчанию).
Блок-диаграмма ВП (рис. 2.83) содержит последовательно включенные ВП **Открыть звуковой файл** (Sound File Open), **Записать звуковой файл** (Sound File Write) и **Закрыть звуковой файл** (Sound File Close).

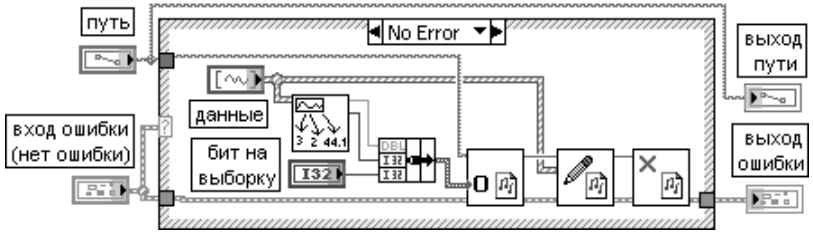
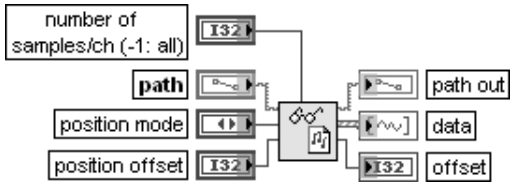
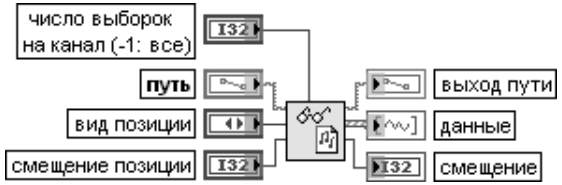


Рис. 2.83. Блок-диаграмма ВП ***Простая запись звукового файла*** *(Sound File Write Simple)*

Sound File Read Simple



Простое считывание звукового файла



ВП считывает данные из файла с расширением .wav в массив осциллограмм. Этот ВП автоматически открывает, считывает и закрывает wav-файл.
Вход число выборок на канал (**number of samples/ch**) определяет число выборок на канал, считываемых из файла. –1 делает считывание всех выборок.
Вход **режим позиции** (position mode) вместе со входом **смещение позиции** (position offset) устанавливает начало операции считывания. **Абсолютный** (Absolute) начинает операцию с начала файла и **смещения позиции**, поэтому смещение отсчитывается относительно начала файла. **Относительный** (Relative) (по умолчанию) начинает операцию с текущего положения метки файла и **смещения позиции**.
Выход **данные** (data) содержит те же компоненты, что и одноименный вход рассмотренной выше функции **Простая запись звукового файла**.
Отличие компонента **t0** заключается в том, что он содержит начальное время осциллограммы. LabVIEW устанавливает в отметке времени значение 00:00 1 января 1904 года по Гринвичу.
Выход **смещение** (offset) показывает новое положение метки файла относительно начала файла в единицах выборок. По умолчанию значение равно 0.

Блок-диаграмма ВП приведена на рис. 2.84. Она также включает ВП открытия и закрытия звукового файла и ВП **Считывание звукового файла** (Sound File Read)

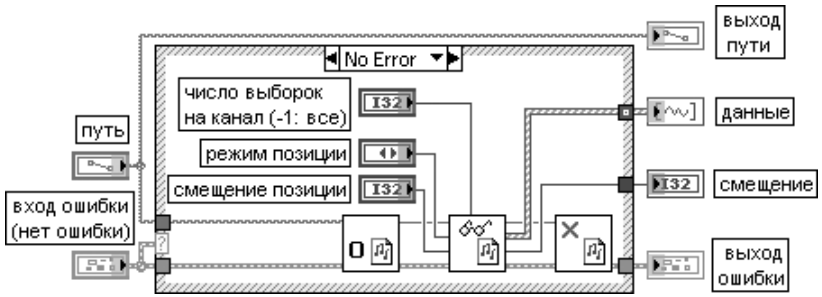


Рис. 2.84 Блок-диаграмма ВП
Простое считывание звукового файла *(Sound File Read Simple)*

Ниже на рис. 2.85 и 2.86 в качестве примеров использования функций записи и воспроизведения звуковых сигналов и файлов приведены блок-диаграммы ВП **Генерировать звук** (Generate Sound) и **Запись звукового колебания в файл** (Sound Input to File) из библиотеки примеров Sound2 NI Example Finger.

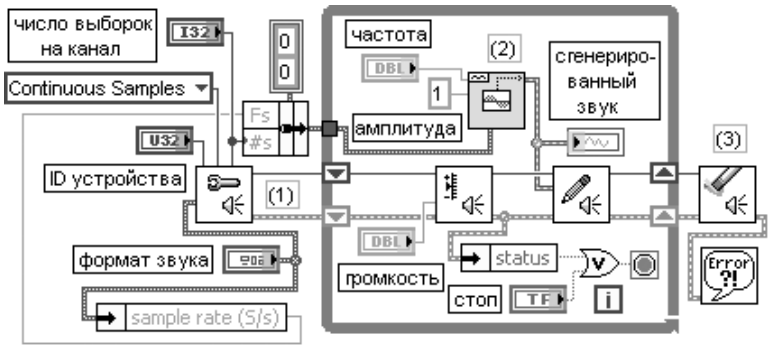


Рис. 2.85. Блок-диаграмма ВП ***Генерировать звук*** *(Generate Sound)*

Пояснения к рисунку:
1. Конфигурирует звуковое устройство для непрерывной работы.
2. До нажатия пользователем кнопки остановки ВП или прихода ошибки:
- генерирует осциллограмму с заданной частотой и амплитудой;
- отображает сгенерированную осциллограмму на графике;
- устанавливает громкость выходного устройства на заданном уровне;
- воспроизводит сгенерированную осциллограмму путем записи данных в выходное устройство.
3. Очищает выходное звуковое устройство и отображает ошибки при их появлении.

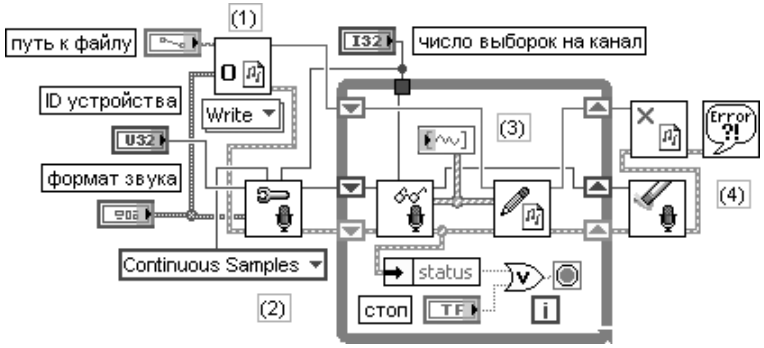


Рис. 2.86. Блок-диаграмма ВП
Запись звукового колебания в файл (Sound Input to File)

Пояснения к рисунку:

1) открывает для записи звуковой файл, заданный пользователем;
2) конфигурирует устройство для непрерывного ввода звукового сигнала;
3) до нажатия пользователем кнопки стоп или поступления ошибки:
   - принимает с микрофона количество выборок, заданное на входе число выборок и отображает их на графическом индикаторе;
   - записывает собранные данные в файл;
4) очищает входное звуковое устройство и закрывает файл. Отображает ошибки в случае их возникновения.

Математические функции LabVIEW

3

Вид подпалитр категории **математических функций** (Mathematics) в стандартном режиме показан на рисунке 3.1. В эту категорию входят следующие подпалитры (в порядке расположения слева направо и сверху вниз): **Числовые** (Numeric), **Элементарные и специальные функции** (Elementary & Special Functions and VIs), **Линейная алгебра** (Linear Algebra), **Аппроксимация данных** (Fitting), **Интерполяция и экстраполяция** (Interp & Extrap), **Интегрирование и дифференцирование** (Integ & Diff), **Вероятность и статистика** (Probability and Statistics), **Оптимизация** (Optimization), **Дифференциальные уравнения** (Differential Equations), **Геометрия** (Geometry), **Полиномы** (Polinomial), **Скрипты и формулы** (Scripts & Formulas).

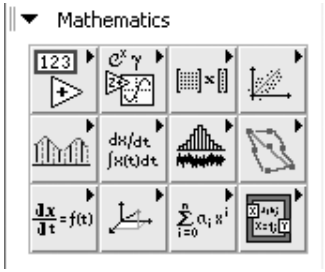


Рис. 3.1. Вид палитры математических функций

3.1. Функции линейной алгебры

Функции линейной алгебры (рис. 3.2) позволяют находить решение системы линейных уравнений, выполнять обращение матриц, рассчитывать значение определителя, находить собственные значения и собственные векторы и рассчитывать различного рода произведения матриц и векторов, позволяют выполнять различные разложения матриц, находить такие параметры матриц, как след, ранг, норму и число обусловленности. Почти все функции линейной алгебры являются полиморфными. В большинстве случаев полиморфность проявляется в возможности выбора между вещественным и комплексным результатом вычисления, однако могут быть и более сложные варианты полиморфности. При описании функций для экономии места селектор полиморфных ВП, за исключением первой функции, не показывается.

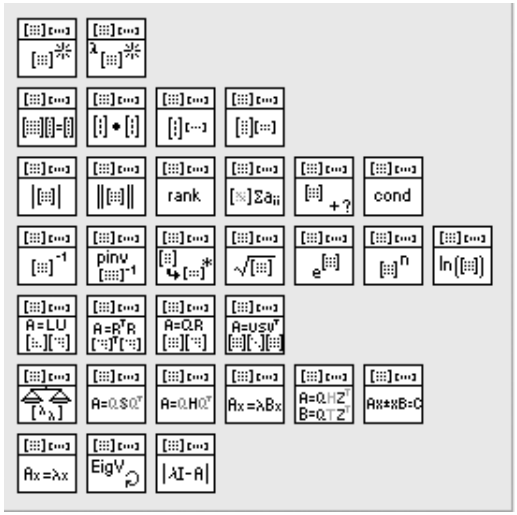
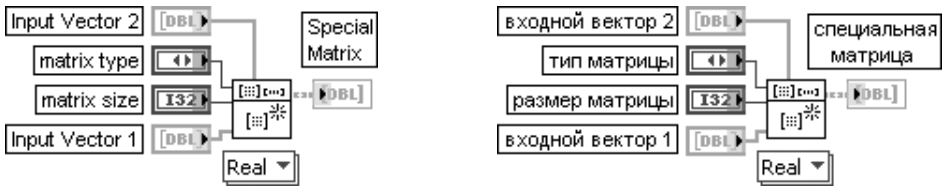


Рис. 3.2. Вид палитры функций линейной алгебры

Приоритет рассмотрения функций линейной алгебры в последующей таблице соответствует порядку их расположения в палитре слева направо и сверху вниз.

Create Special Matrix / Создать специальную матрицу



Функция порождает действительную специальную матрицу, заданную **типом матрицы** (matrix type).
Входной вектор 2 (Input Vector 2) используется для создания специальной матрицы с некоторыми опциями.
Вход **тип матрицы** (matrix type) определяет тип специальной матрицы, порождаемой на выходе **специальная матрица** (Special Matrix).
Описание различных типов матриц приведено в таблице. При этом приняты следующие обозначения: n представляет размер матрицы, **X** представляет **входной вектор 1** (Input Vector1), nx представляет размер **X**, **Y** представляет **входной вектор 2** (Input Vector2), ny представляет размер **Y**, **B** представляет выход **специальная матрица**.

0	**Единичная** (Identity) – порождает единичную матрицу размером $n \times n$
1	**Диагональная** (Diagonal) – порождает диагональную матрицу размером $nx \times nx$, диагональные элементы которой являются элементами **X**
2	**Матрица Теплица** (Toepliz) – порождает матрицу Теплица размером $nx \times nx$, у которой **X** является первым столбцом, а **Y** является первой строкой. При различии первых элементов **X** и **Y** в матрице используется первый элемент **X**

3	**Вандермонда** (Vandermonde) – порождает матрицу Вандермонда размером $nx \times nx$, у которой столбцы являются степенями **X**. Элементы матрицы Вандермонда рассчитываются следующим образом: $b_{ij} = x_i^{nx-j-1}$, где $i, j = \overline{0, nx-1}$
4	**Сопровождающая** (Companion) – порождает сопровождающую матрицу размером $nx - 1 \times nx - 1$. Если вектор **X** является вектором полиномиальных коэффициентов, то коэффициент при самой высокой степени является первым элементом **X**, а свободный член – последним элементом. При этом соответствующая сопровождающая матрица формируется следующим образом: первая строка $b_{0,j-1} = -\frac{x_j}{x_0}$; $j = \overline{1, nx-1}$. Оставшаяся часть **B**, начиная со второй строки, является единичной матрицей. Собственные значения сопровождающей матрицы содержат корни соответствующего полинома

Вход **размер матрицы** (matrix size) определяет размер выходной специальной матрицы.
Вход **входной вектор 1** (Input Vector 1) используется для создания специальной матрицы с некоторыми опциями.
Выход **специальная матрица** (Special Matrix) отображает порождаемую матрицу

Create Real Matrix From EigenValues / Создать реальную матрицу из собственных значений

Функция создает реальную матрицу из **собственных значений** (Eigenvalues)

Solve Linear Equations / Найти решение системы линейных уравнений

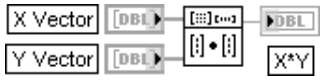
Функция находит решение действительной линейной системы уравнений **AX = Y**.
Вход **входная матрица** (Input Matrix) представляет квадратную или прямоугольную действительную матрицу. Число элементов на входе **известный вектор** (Known Vector) должно быть равно числу строк входной матрицы. При нарушении этого условия ВП устанавливает на выходе **вектор решения** (Solution Vector) пустой массив и возвращает ошибку.
Вход **известный вектор** (Known Vector) представляет массив известных значений зависимой переменной.
Вход **тип матрицы** (matrix type) определяет тип входной матрицы. Определение типа входной матрицы может ускорить расчет вектора решений и помочь избежать избыточных расчетов, которые могут внести дополнительную погрешность. Варианты типов матрицы приведены в таблице.

0	1	2	3
Общего вида (по умолчанию) (General (default))	**Положительно определенная** (Positive definite)	**Нижняя треугольная** (Lower triangular)	**Верхняя треугольная** (Upper triangular)

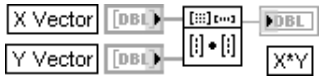
Выход **вектор решения** (Solution Vector) представляет вектор решения **X** системы ***AX = Y***.

Пусть ***A*** будет входной матрицей размером $m \times n$, ***Y*** будет набором ***m*** коэффициентов **известного вектора**, а ***X*** будет набором ***n*** элементов **вектора решения**, который является решением системы ***AX*** = ***Y***.
При $m > n$ число уравнений системы превышает число неизвестных, то есть система является переопределенной. Решение, удовлетворяющее ***AX*** = ***Y***, может не существовать, поэтому ВП находит решение ***X***, которое минимизирует $||AX - Y||$ с помощью метода наименьших квадратов.
При $m < n$ число неизвестных системы превышает число уравнений, таким образом, система является неопределенной. Она может иметь бесконечное число решений, удовлетворяющих ***AX*** = ***Y***. ВП находит одно из этих решений.
В случае, когда $m = n$, если ***A*** является невырожденной матрицей, у которой никакие строки или столбцы не являются линейной комбинацией других строк или столбцов, тогда решение системы ***X*** может быть найдено путем разложения входной матрицы ***A*** на ее нижнюю и верхнюю треугольные матрицы, ***L*** и ***U***, такие что ***AX*** = ***LZ*** = ***Y*** и ***Z*** = ***UX*** могут быть альтернативным представлением исходной системы. При этом ***Z*** также является вектором из ***n*** элементов.
Треугольные системы легко решаются с помощью рекурсивных процедур, следовательно, если из матрицы ***A*** выделены матрицы ***L*** и ***U***, то матрица ***Z*** может быть найдена из системы ***LZ*** = ***Y***, а матрица ***X*** из системы ***UX*** = ***Z***.
В случае если $m \neq n$, может быть выполнено разложение матрицы ***A*** на ортогональную матрицу ***Q*** и верхнюю треугольную матрицу ***R***, так что ***A*** = ***QR***. Линейная система после этого может быть представлена в следующем виде ***QRX*** = ***Y***, а ее решение найдено из уравнения ***RX*** = ***QTY***.
Не всегда возможно наперед оценить невырожденность матрицы, особенно для больших систем. Функция **Решение линейных уравнений** (Solve Linear Equations) обнаруживает вырожденные матрицы и возвращает ошибку, в связи с чем пользователю нет необходимости проверять правильность системы перед использованием данной функции.
Численная реализация обратимости матриц вследствие рекурсивной природы очень чувствительна к ошибкам округления, вносимым сопроцессором операций с плавающей запятой. Хотя вычисления используют максимально возможную точность, функция не гарантирует решения систем

Dot Product

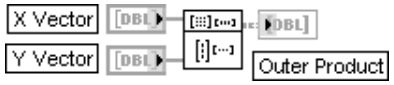


Скалярное произведение

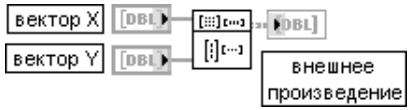


Функция рассчитывает скалярное произведение **вектора X** (X Vector) и **вектора Y** (Y Vector) используя следующее выражение $\boldsymbol{X}\,\boldsymbol{Y} = \sum_{i=0}^{n-1} x_i y_i$, где ***n*** – число элементов векторов

Outer Product



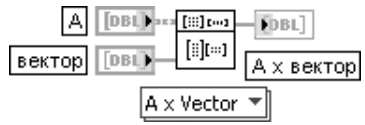
Внешнее произведение



Функция рассчитывает внешнее произведение **вектора X** (X Vector) и **вектора Y** (Y Vector), используя следующее выражение: $a_{ij} = x_i y_j$, где $\begin{cases} i = \overline{0, n-1} \\ j = \overline{0, m-1} \end{cases}$, ***A*** представляет двумерную выходную последовательность **внешнее произведение** (Outer Product), ***n*** – число элементов входной последовательности **X** и ***m*** – число элементов входной последовательности **Y**

A x B

A x B

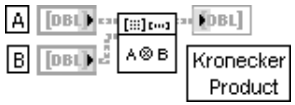


Функция выполняет матричное умножение двух входных матриц.
A является первой входной матрицей. Число столбцов матрицы **A** должно быть равно числу строк матрицы **B** и должно быть больше нуля: $k > 0$. При нарушении этих условий ВП выводит на выход **A x B** пустой массив и возвращает ошибку. **B** является второй матрицей.
Выход **A x B** представляет матрицу, содержащую результат матричного умножения **A** x **B**.
Если ***A*** представляет матрицу размером $n \times k$ и ***B*** представляет матрицу размером $k \times m$, то результатом матричного умножения ***A*** на ***B*** будет матрица $C = AB$ размером $n \times m$. ВП рассчитывает элементы матрицы ***C***, используя следующее выражение:

$c_{ij} = \sum_{l=0}^{k-1} a_{il} b_{lj}$ для $\begin{cases} i = \overline{0, n-1} \\ j = \overline{0, m-1} \end{cases}$, где ***n*** – число строк в матрице ***A***, ***k*** – число столбцов в матрице ***A*** и число строк в матрице ***B***, ***m*** – число столбцов в матрице ***B***.
ВП может выполнять умножение входной матрицы и входного вектора (справа). Если ***A*** является матрицей $n \times k$ и ***X*** является вектором с ***k*** элементами, то результатом умножения ***A*** и ***X***, $Y = AX$ является вектор ***Y*** с ***n*** элементами, которые вычисляются ВП с использованием следующего выражения:

$y_i = \sum_{j=0}^{k-1} a_{ij} x_j$, где $i = \overline{0, n-1}$, ***n*** – число строк в матрице ***A***, ***k*** – число столбцов в матрице ***A*** и число элементов вектора **X**

Kronecker Product



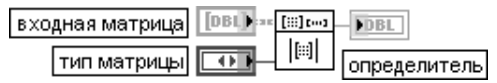
Кронекерово произведение

Функция рассчитывает **Кронекерово произведение** (Kronecker Product) двух матриц.
Кронекерово произведение двух матриц **A** и **B** размером $n \times m$ и $k \times l$ представляет матрицу **C** = **A⊗B** размером $nk \times ml$, элементы которой рассчитываются следующим образом:

$$C = \begin{bmatrix} a_{11}B & a_{12}B & \dots & a_{1m}B \\ a_{21}B & a_{22}B & & a_{11}B \\ \dots & & & \\ a_{n1}B & a_{n2}B & & a_{nm}B \end{bmatrix}$$

Determinant

Определитель

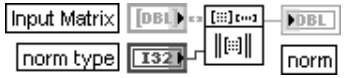


Функция рассчитывает **определитель** (determinant) действительной квадратной **входной матрицы** (Input Matrix).
Вход **тип матрицы** (matrix type) имеет те же значения, что и в рассмотренном выше ВП **Решение линейных уравнений** (Solve Linear Equations).

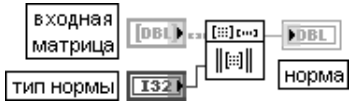
Пусть ***A*** является квадратной матрицей, которая представляет **входную матрицу**, и пусть ***L*** и ***U*** представляют ее нижнюю и верхнюю треугольные матрицы, такие что ***A = LU***, где главные диагональные элементы нижней треугольной матрицы ***L*** имеют единичные значения. При этом ВП находит определитель матрицы ***A*** с помощью произведения главных диагональных элементов верхней треугольной матрицы ***U***:

$|A| = \prod_{i=0}^{n-1} u_{ii}$, где $|A|$ – определитель матрицы ***A***, ***n*** – размер матрицы ***A***

Matrix Norm



Норма матрицы

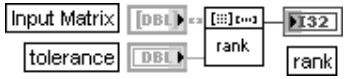


Функция находит **норму** (norm) **входной матрицы** (Input Matrix).
Входная матрица может быть квадратной или прямоугольной действительной матрицей.
Вход **тип нормы** (norm type) определяет тип используемой в расчетах нормы (таблица).

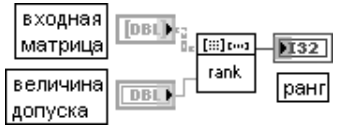
0	**2-норма** – $\|A\|_2$ равна наибольшему сингулярному значению входной матрицы **A**
1	**1-норма** – $\|A\|_1$ равна наибольшей сумме модулей элементов столбца входной матрицы **A**
2	**F-норма** – $\|A\|_f$ равна $\sqrt{\sum diag(A^T A)}$, где $diag(A^T A)$ – диагональные элементы матрицы $A^T A$, а A^T – результат транспонирования матрицы **A**
3	**inf-норма** – $\|A\|_\infty$ равна наибольшей сумме модулей элементов строки входной матрицы **A**

Выход **норма** представляет скаляр, который дает некоторую меру величины элементов входной матрицы, зависящую от выбранного типа нормы

Matrix Rank



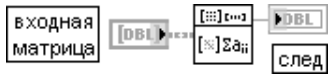
Ранг матрицы



Функция находит **ранг** (rank) действительной прямоугольной **входной матрицы** (Input Matrix).
Вход **величина допуска** (tolerance) определяет пороговый уровень, при котором число, превысившее его сингулярные значения, является рангом входной матрицы. По умолчанию значение этого параметра равно –1. Если значение **величина допуска** отрицательно, внутренний допуск, используемый для определения ранга, устанавливается в соответствии с выражением
величина допуска = $\max(m,n)*\|A\|*\varepsilon$,
где ***A*** – входная матрица, ***m*** и ***n*** представляют соответственно число строк и столбцов матрицы **A**, $\|A\|$ является нормой **2-norm** матрицы **A**, **ε** – наименьшее число, которое может быть представлено в форме с двойной точностью, а именно $\varepsilon = 2^{-52} = 2{,}22e-16$.
Выход **ранг** отображает число сингулярных значений входной матрицы, которые превышают значение **величины допуска**. **Ранг** является максимальным числом независимых строк или столбцов входной матрицы

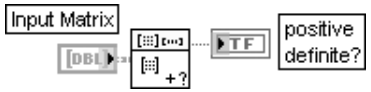
Trace

След

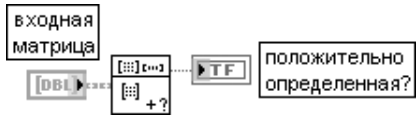


Функция находит **след** (trace) **входной матрицы** (Input Matrix).
Входная матрица должна быть квадратной, и ее размер должен быть больше 0. Если входная матрица пустая или неквадратная, то ВП устанавливает на выходе **след** значение NaN и возвращает ошибку.
След (trace) представляет сумму элементов на главной диагонали входной матрицы

Test Positive Definite

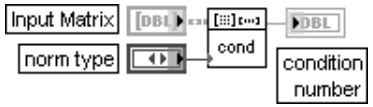


Тест на положительную определенность

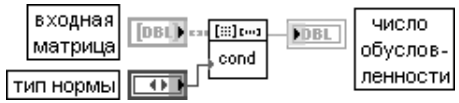


Функция проверяет **входную матрицу** (Input Matrix) на положительную определенность.
Входная матрица должна быть квадратной действительной матрицей.
Выход **положительно определенная?** (positive definite?) содержит результат тестирования. Если входная матрица является положительно определенной, то на выход **положительно определенная?** передается значение ИСТИНА, в противном случае передается значение ЛОЖЬ

Matrix Condition Number

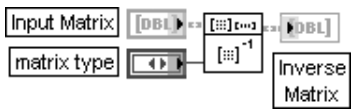


Число обусловленности матрицы

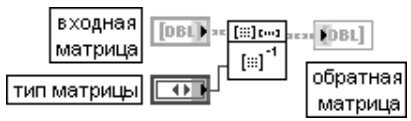


Функция находит **число обусловленности** (condition number) действительной **входной матрицы** (Input Matrix).
Входная матрица может быть прямоугольной при установке на входе **тип нормы** (norm type) нормы типа **2-norm**. При выборе другого типа нормы входная матрица должна быть квадратной.
Вход **тип нормы** отображает тип нормы, используемый для расчета числа обусловленности.
Определение типов нормы рассмотрено выше при анализе функции **Норма матрицы** (Matrix Norm).
Число обусловленности рассчитывается с помощью выражения $\mathbf{c} = \|A\|_p \|A^{-1}\|_p$, где $\|A\|_p$ является нормой входной матрицы. Различные значения **p** определяют различные типы нормы, соответственно **p** определяет и различные способы вычисления чисел обусловленности.
Для **2-norm** число обусловленности определяется как отношение наибольшего и наименьшего сингулярных значений входной матрицы **A**.
Числа обусловленности матрицы определяют чувствительность решения системы линейных уравнений к ошибкам данных

Inverse Matrix

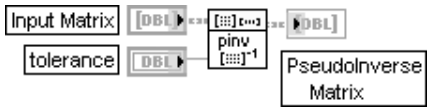


Обратная матрица

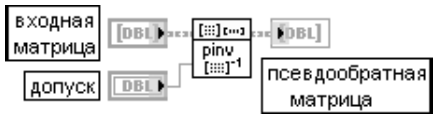


Функция находит **обратную матрицу** (Inverse Matrix), если она существует, для **входной матрицы** (Input Matrix).
Вход **тип матрицы** (matrix type) имеет те же значения, что и в рассмотренной выше функции **Решить систему линейных уравнений** (Solve Linear Equations).
Входная матрица должна быть невырожденной и квадратной. ВП определяет вырожденность матрицы и возвращает ошибку.
Если входная матрица невырожденная, то обратная матрица может быть найдена с помощью решения системы линейных уравнений $AB = I$, где A – входная матрица, B – обратная матрица, I – единичная матрица

PseudoInverse Matrix

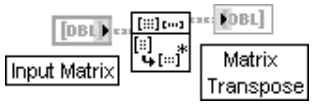


Псевдообратная матрица



Функция находит **псевдообратную матрицу** (PseudoInverse Matrix) прямоугольной действительной **входной матрицы** (Input Matrix).
Входная матрица является действительной прямоугольной матрицей. Когда входная матрица является неквадратной или вырожденной, то обратная матрица для нее не существует. В этом случае можно рассчитать псевдообратную матрицу.
Назначение входа величина допуска **(tolerance) было рассмотрено выше при анализе функции** Ранг **(Rank).**
Функция рассчитывает **псевдообратную матрицу** A^+, используя алгоритм **SVD** и любое сингулярное значение, меньшее **величины допуска**.
Матрица A^+ размером $m \times p$ называется псевдообратной для матрицы A, если A^+ удовлетворяет следующим четырем условиям Мура-Пенроуза:
1. $A A^+ A = A$;
2. $A^+ A A^+ = A^+$;
3. $A A^+$ является симметрической матрицей;
4. $A^+ A$ является симметрической матрицей

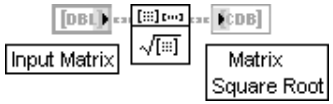
Transpose Matrix



Транспонировать матрицу

Функция транспонирует **входную матрицу** (Input Matrix). Если **входная матрица** является комплексной, ВП выполняет комплексно-сопряженное преобразование. Тип данных, подключенных ко входу **входная матрица**, определяет вид полиморфной реализации ВП

Matrix Square Root



Квадратный корень из матрицы

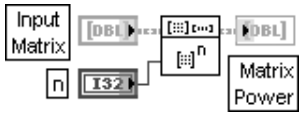
Функция рассчитывает квадратный корень из **входной матрицы** (Input Matrix). Тип данных, подключенных ко входу **входная матрица**, определяет вид полиморфной реализации ВП

Matrix Exp

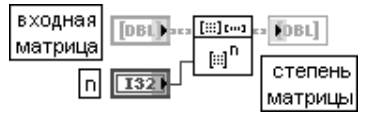
Экспонента матрицы

Функция рассчитывает экспоненту **входной матрицы** (Input Matrix). Тип данных, подключенных ко входу **входная матрица**, определяет вид полиморфной реализации ВП

Matrix Power

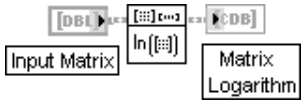


Степень матрицы

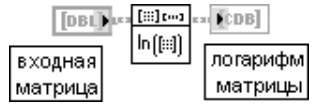


Функция рассчитывает степень **n входной матрицы** (Input Matrix). Тип данных, подключенных ко входу **входная матрица**, определяет вид полиморфной реализации ВП

Matrix Logarithm

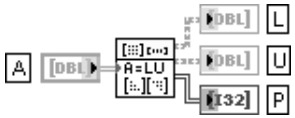


Логарифм матрицы

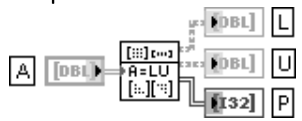


Функция рассчитывает логарифм **входной матрицы** (Input Matrix). Тип данных, подключенных ко входу **входная матрица**, определяет вид полиморфной реализации ВП

LU Factorization

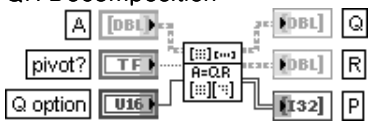


LU-разложение

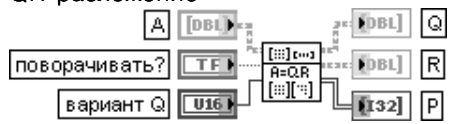


Функция выполняет **LU**-разложение действительной квадратной матрицы **A**.
Выход **L** представляет нижнюю треугольную матрицу.
Выход **U** представляет верхнюю треугольную матрицу.
Выход **P** отображает матрицу перестановки.
ВП **LU-разложение** (LU Factorization) разлагает квадратную матрицу **A** на перечисленные выше типы матриц так, что выполняется условие **PA**=**LU**.
Разложение играет роль ключевого шага для обращения матриц, вычисления определителя и решения системы линейных уравнений

QR Decomposition



QR-разложение



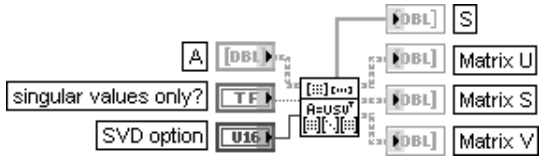
Функция выполняет **QR**-разложение действительной матрицы **A**.

A является действительной матрицей размером $m \times n$, где m – число строк и n – число столбцов матрицы **A**. Матрица **A** может быть прямоугольной или квадратной.
Вход **поворачивать?** (pivot?) определяет использование при разложении **A** поворота столбца. При установке на этом входе состояния ИСТИНА ВП производит разложение **A** в соответствии с уравнением **AP**=**QR** и возвращает абсолютные значения диагональных элементов **R**, упорядоченных по убыванию. Если на входе установлено состояние ЛОЖЬ (по умолчанию), то ВП выполняет разложение **A** в соответствии с уравнением **A**=**QR**.
Вход **вариант Q** (Q option) определяет вариант создания матрицы **Q**. **Вариант** должен принимать одно из следующих значений:

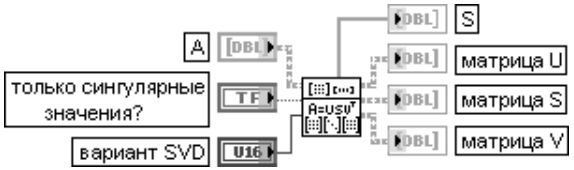
0	**Полный размер Q** (по умолчанию) – размер **Q** равен $m \times m$, а размер **R** равен $m \times n$
1	**Экономичный размер Q** – размер **Q** равен $m \times \min(m, n)$, а размер **R** равен $\min(m, n) \times n$
2	**Отсутствие Q** – ВП не создает **Q**, а размер **R** равен $\min(m, n) \times n$

Выход **Q** возвращает ортогональную матрицу размером $m \times n$.
Выход **R** возвращает верхнюю треугольную матрицу размером $m \times n$.
Выход **P** отображает матрицу перестановки размером $m \times n$.
QR-разложение иногда еще называется ортогонально-треугольным разложением.
ВП **QR-разложение** разлагает действительную матрицу **A** на перечисленные выше матрицы так, что выполняется условие **A** = **QR**.
QR-разложение может использоваться для решения системы линейных уравнений с числом уравнений большим числа неизвестных

SVD Decomposition



SVD-разложение



Функция выполняет разложение по сингулярным значениям (**SVD**) заданной действительной матрицы **A** размером $m \times n$ с $m > n$, где m – число строк, а n – число столбцов матрицы **A**. Если $m < n$, то перед использованием данного ВП такую матрицу необходимо транспонировать.
Вход **только сингулярные значения?** (singular values only?) при включении в состояние ИСТИНА устанавливает режим вычисления только сингулярных значений (без расчета матриц U и V). По умолчанию этот вход выключен (состояние ЛОЖЬ).
Вход **вариант SVD** (SVD Option) определяет вариант разложения:

0	**Ограниченный** (по умолчанию) – разлагает матрицу $m \times n$ на матричные сомножители U $(m \times \min(m,n))$, S $(\min(m,n) \times \min(m,n))$ и сопряженно-транспонированную V $(n \times \min(m,n))$
1	**Полный** – разлагает матрицу $m \times n$ на матричные сомножители U $(m \times m)$, S $(m \times n)$ и сопряженно-транспонированную V $(n \times n)$

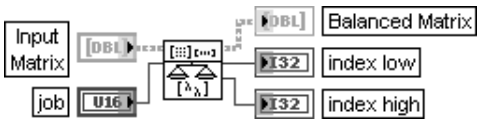
Выход вектора **S** возвращает массив сингулярных значений матрицы **A**.
Выход **матрица U** отображает матрицу размером $m \times n$, содержащую n ортогональных столбцов.
Выход **матрица S** возвращает диагональную матрицу, содержащую n значений из вектора **S**.
Выход **матрица V** представляет ортогональную матрицу размером $n \times n$.
SVD-разложение создает три матрицы, для которых выполняется следующее условие: $A = US_0V^T$, где **U** и V^T являются ортогональными матрицами, а S_0 является диагональной матрицей размером $n \times n$ с элементами вектора **S** на диагонали в порядке невозрастания

Cholesky Factorization

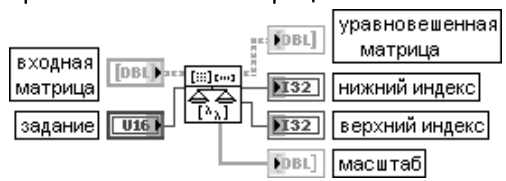
Разложение Холецкого

Функция выполняет разложение Холецкого действительной положительно определенной матрицы **A**. Выход матрицы **Холецкого** (Cholesky) содержит полученную в результате разложения верхнюю треугольную матрицу **R**.
Если действительная квадратная матрица **A** является положительно определенной, то она может быть разделена на множители в соответствии с выражением $A = R^TR$, где **R** является верхней треугольной матрицей, а R^T – результатом транспонирования матрицы **R**

Matrix Balance

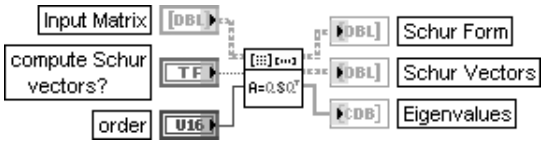


Уравновешивание матрицы

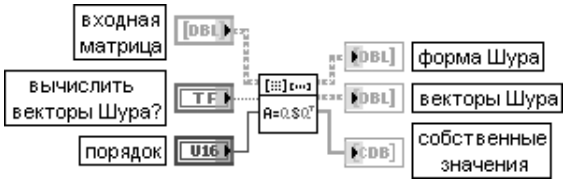


Функция уравновешивает матрицу общего вида **входная матрица** (Input Matrix) для повышения точности расчета собственных значений и собственных векторов. Тип данных, подключенных ко входу **входная матрица**, определяет вид полиморфной реализации ВП.
Вход **задание** (job) определяет вид операции уравновешивания матрицы:
0 – неперемещенная и немасштабированная;
1 – перемещенная, но не масштабированная;
2 – масштабированная, но не перемещенная;
3 – перемещенная и масштабированная (по умолчанию).
Выход **нижний индекс** (index low) отображает форму **уравновешенной матрицы** (Balanced Matrix). **Уравновешенная матрица**$(i, j) = 0$ если $i > j$ и $0<=j<$ нижний индекс. Если на входе **задание** установлен вариант 0 или 2, нижний индекс = 0.
Выход **верхний индекс** (index high) также показывает форму **уравновешенной матрицы. Уравновешенная матрица**$(i, j) = 0$, если $i > j$ и верхний индекс$< i <= n - 1$. Если на входе **задание** установлен вариант 0 или 2, верхний индекс = $n - 1$

Schur Decomposition

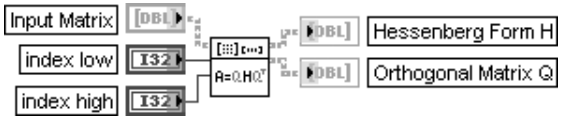


Разложение Шура

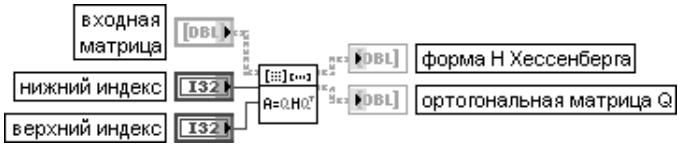


Функция выполняет разложение Шура квадратной матрицы

Hessenberg Decomposition

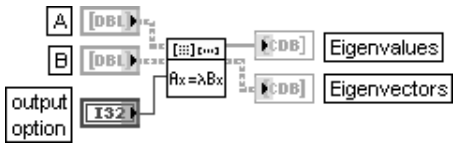


Разложение Хессенберга

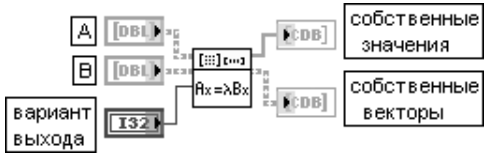


Функция выполняет разложение Хессенберга **входной матрицы**

Generalized Eigenvalues and Vectors

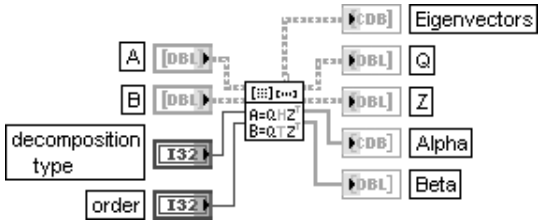


Обобщенные собственные значения и векторы

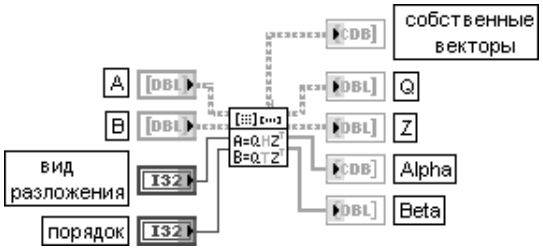


Функция рассчитывает обобщенные правые собственные значения и векторы пары матриц **A** и **B**. Типы данных, подключенных ко входам **A** и **B**, определяют полиморфную реализацию применяемого ВП

QZ Decomposition

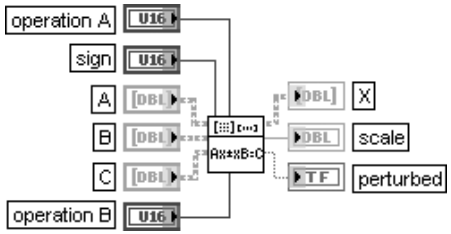


QZ-разложение

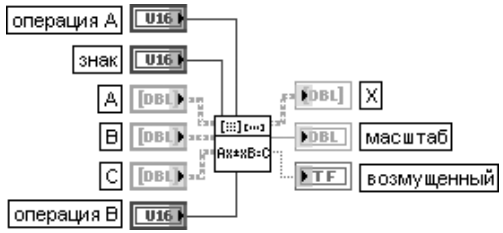


Функция выполняет QZ-разложение пары квадратных матриц. Типы данных, подключенных ко входам **A** и **B**, определяют полиморфную реализацию применяемого ВП

Sylvester Equations

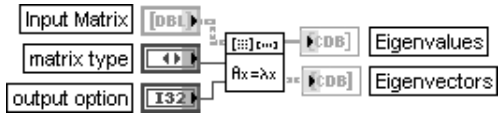


Уравнение Сильвестра

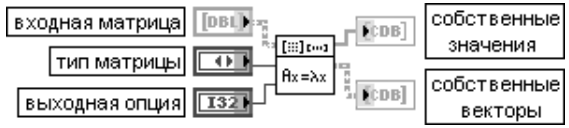


Функция решает матричное уравнение Сильвестра. Тип данных, подключенных ко входам **A**, **B** и **C**, определяет полиморфную реализацию применяемого ВП

EigenValues and Vectors

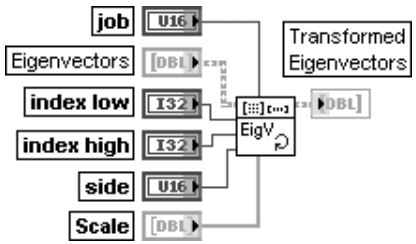


Собственные значения и векторы

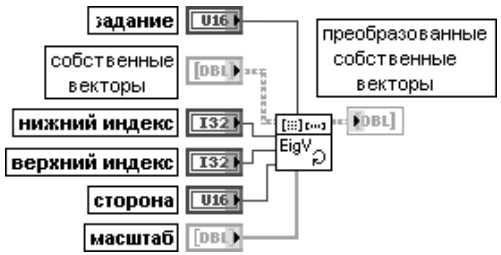


Функция находит собственные значения и правые собственные векторы квадратной действительной **входной матрицы** (Input Matrix).
Вход **тип матрицы** (matrix type) определяет тип входной матрицы и может принимать два значения. При выборе значения **общего вида** (General (0)) он определяет матрицу общего вида (по умолчанию), при установке симметрической (**Symmetric** (1)) – симметрическую матрицу. Симметрическая матрица требует меньшего количества расчетов по сравнению с несимметрической. Симметрическая матрица всегда имеет действительные собственные векторы и значения.
Вход **выходная опция** (output option) определяет состав компонентов, вычисляемых ВП. Он также может принимать два значения: значение eigenvalues (0) определяет расчет только **собственных значений**, а значение eigenvalues & vectors (1) определяет расчет **собственных значений** и **собственных векторов** (по умолчанию).
Выход **собственные значения** (Eigenvalues) отображает комплексный вектор из n элементов, который содержит все рассчитанные собственные значения входной матрицы. Входная матрица может иметь комплексные собственные значения, если она не симметрическая.
Выход **собственные векторы** (Eigenvectors) представляет комплексную матрицу размером $n \times n$, содержащую все рассчитанные собственные векторы входной матрицы. i-й столбец **собственных векторов** является собственным вектором, соответствующим i-му компоненту вектора **собственные значения**. Каждый собственный вектор нормализован так, что его наибольший компонент равен единице. Входная матрица может иметь комплексные собственные векторы, если она не симметрична.
Задачей собственных значений является нетривиальное решение уравнения $AX = \lambda X$, где A является входной матрицей размером $n \times n$, X является вектором из n элементов и λ является скаляром. n значений, которые удовлетворяют уравнению, являются собственными значениями матрицы A, а соответствующие значения X являются правыми собственными векторами матрицы A. Симметрическая действительная матрица всегда имеет действительные собственные значения и собственные векторы

Back Transform Eigenvectors



Обратное преобразование собственных векторов



Функция преобразует собственные векторы уравновешенной матрицы в аналогичные векторы исходной матрицы.
Это ВП рекомендуется использовать после уравновешивания матрицы с помощью ВП **Уравновешивание матрицы** и вычисления собственных векторов уравновешенной матрицы с помощью ВП **Собственные значения и векторы**

Matrix Characteristic Polynomial

Характеристический полином входной матрицы

Функция рассчитывает характеристический полином **входной матрицы** (Input Matrix). Выход **характеристический полином** (Characteristic Polynomial) возвращает коэффициенты характеристического полинома **входной матрицы** в порядке возрастания

3.2. Функции аппроксимации данных

Функции аппроксимации данных (рис. 3.3) позволяют выполнять аппроксимацию данных как линейной, так и рядом нелинейных зависимостей. В число последних входят экспоненциальная, степенная, гауссовская, логарифмическая, полиномиальная и нелинейная общего вида. Возможна также аппроксимация суммой функций с линейными коэффициентами. В зависимости от меры приближения могут быть выбраны метод **наименьших квадратов** (Least Square), **наименьших модулей** (Least Absolute Residual) и **биквадратный** метод (Bisquare). В состав палитры функций входит Экспресс-ВП **Аппроксимация кривой** (Curve Fitting), позволяющий реализовать основные методы аппроксимации данных.

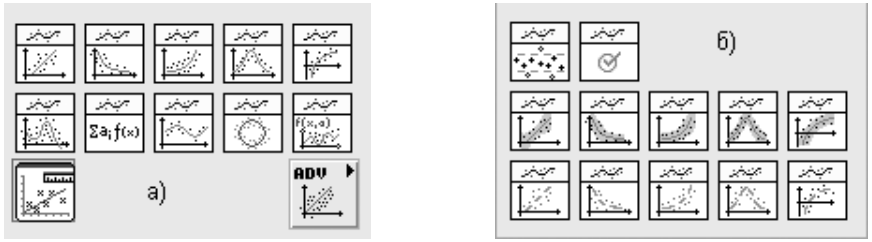
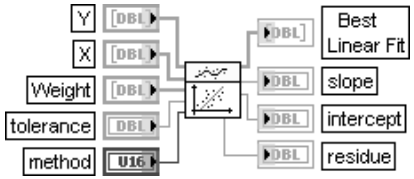


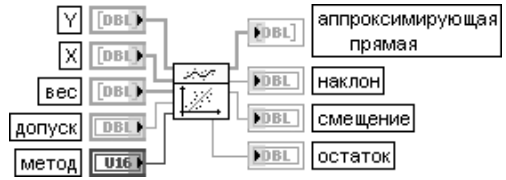
Рис. 3.3. Вид основной палитры (а) и дополнительной подпалитры (б) функций аппроксимации данных

Ниже в таблице рассмотрены все функции аппроксимации данных основной палитры и часть функций дополнительной подпалитры. Порядок рассмотрения соответствует расположению иконок функций в подпалитре справа налево и сверху вниз.

Linear Fit



Линейная аппроксимация



С помощью одного из методов ВП производит аппроксимацию набора входных данных **Y** и **X** линейной функцией, выводит значения прямой на выход **аппроксимирующая прямая** (Best Linear Fit), значения ее коэффициентов на выходы **наклон** (slope) и **смещение** (intercept), а взвешенную среднюю ошибку аппроксимации – на выход **остаток** (residue).

Входные последовательности **Y** и **X** должны содержать как минимум два значения. При нарушении этого условия ВП возвращает на выходе **аппроксимирующая прямая** пустой массив, на выходах **наклон**, **смещение** и **остаток** устанавливает значение NaN, а на выходе **ошибка** (error) возвращает ошибку.

Вход **вес** (Weight) представляет массив весовых коэффициентов для наблюдаемых значений (**X**, **Y**). **Вес** должен быть того же размера, что и **Y**. Если этот вход не подключен, то ВП устанавливает все значения **веса** равным 1. Если значение элемента **веса** отрицательно, ВП использует абсолютное значение элемента.

Вход **метод** (method) задает метод аппроксимации. Перечень методов включает метод **наименьших квадратов** (Least Square) (по умолчанию), *наименьших модулей* (Least Absolute Residual) и **биквадратный** (Bisquare). При использовании биквадратного метода коэффициенты аппроксимации получаются в результате итерационной процедуры.

Метод **наименьших квадратов** рекомендуется использовать при наличии в данных ошибок (шумов) с нормальным распределением, а метод *наименьших модулей* ***и*** **биквадратный** – при наличии выбросов данных.

Вход **допуск** (tolerance) определяет критерий остановки процедуры итерационного процесса поиска значений **наклона** и **смещения** при использовании метода ***наименьших модулей*** или биквадратного метода. Если относительная разность **остатка** в двух последовательных итерациях меньше, чем **допуск**, ВП возвращает результирующие значения **наклона** и **смещения**. Если **допуск** меньше или равен 0, ВП устанавливает его значение равным 0,0001.

Выход **остаток** (residue) возвращает взвешенную среднюю ошибку аппроксимации. Если установлен метод **наименьших модулей**, **остаток** рассчитывается как средняя взвешенная абсолютная ошибка:

$\frac{1}{N}\sum_{i=0}^{N-1} w_i \left| f_i - y_i \right|$, где w_i – ***i***-й элемент **веса**, f_i – значение выходной последовательности **аппроксимирующая прямая**, y_i – значение входной последовательности ***Y***.

В других случаях **остаток** рассчитывается как средний взвешенный квадрат ошибки:

$$\frac{1}{N}\sum_{i=0}^{N-1} w_i (f_i - y_i)^2.$$

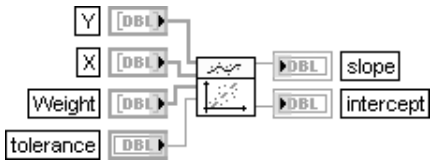
Упрощенной версией рассмотренного ВП является ВП **Коэффициенты линейной аппроксимации** (Linear Fit Coefficients), размещенный в дополнительной подпалитре. На схеме его подключения, приведенной ниже, видно, что он рассчитывает только коэффициенты прямой.

Еще одним ВП, связанным с ВП **Линейная аппроксимация,** является полиморфный ВП **Интервалы линейной аппроксимации**, который рассчитывает полосы доверительных интервалов для аппроксимирующей линии и доверительные интервалы для ее параметров.

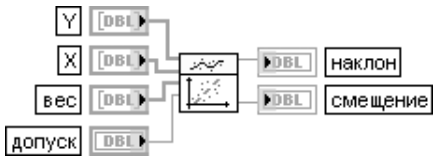
Вход **доверительная вероятность** (confidence level) определяет уровень достоверности доверительного интервала. По умолчанию установлено значение 0,95, означающее, что вероятность попадания аппроксимирующей линии между **нижней границей** (Lower Bound) и **верхней границей** (Upper Bound) равна 95%. Значение **доверительной вероятности** должно находиться между 0 и 1. Селектор этого полиморфного ВП позволяет выбрать полосу **доверительных интервалов** (Confidence) или полосу **предсказания** (Prediction).

Пример применения ВП **Интервалы линейной аппроксимации** приведен на рис. 3.4

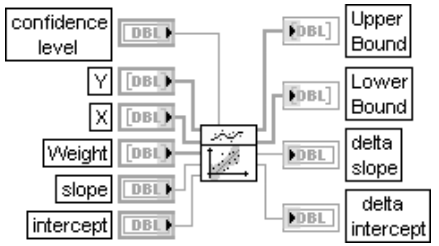
Linear Fit Coefficients



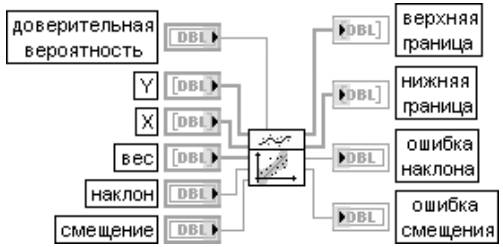
Коэффициенты линейной аппроксимации



Linear Fit Intervals



Интервалы линейной аппроксимации



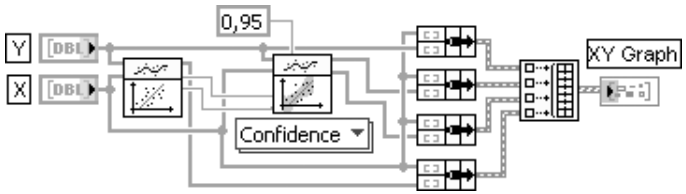
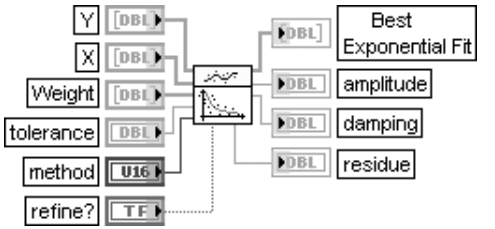
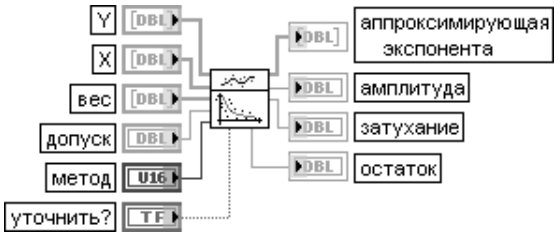


Рис. 3.4. Пример применения ВП ***Интервалы линейной аппроксимации***

Exponential Fit



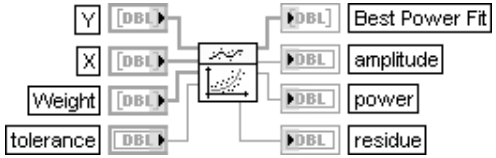
Экспоненциальная аппроксимация



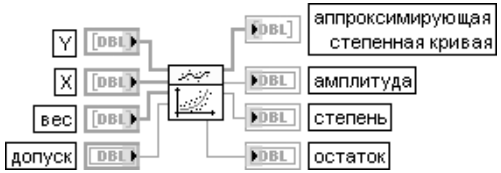
ВП производит аппроксимацию набора входных данных **Y** и **X** экспоненциальной функцией, выводит значения аппроксимирующей функции на выход **аппроксимирующая экспонента** (Best Exponential Fit), ее коэффициенты – на выходы **амплитуда** (amplitude) и **затухание** (damping), а взвешенную среднюю ошибку аппроксимации – на выход **остаток** (residue).

Значения **Y** должны быть одного знака, и их число должно быть не менее двух.
Вход **уточнить** (refine?) определяет необходимость дальнейшего уточнения **амплитуды** и **затухания**. При установке на этом входе значения ИСТИНА время вычислений для данного ВП увеличивается. По умолчанию на нем установлено значение ЛОЖЬ

Power Fit

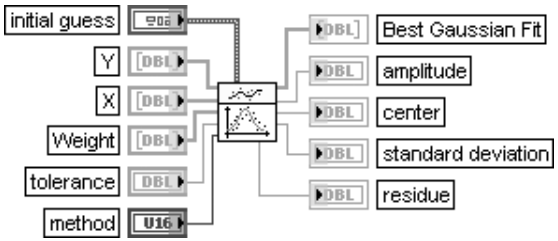


Степенная аппроксимация

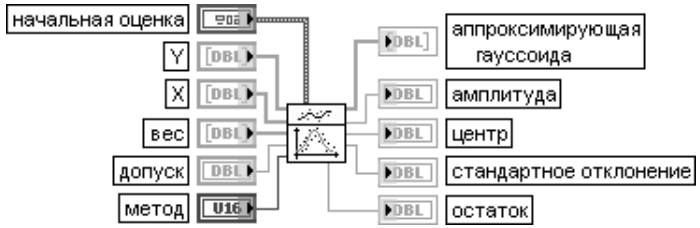


ВП производит аппроксимацию набора входных данных **Y** и **X** степенной функцией, выводит значения аппроксимирующей функции на выход **аппроксимирующая степенная функция** (Best Power Fit), ее коэффициенты – на выходы **амплитуда** (amplitude) и **степень** (power), а взвешенную среднюю ошибку аппроксимации – на выход **остаток** (residue)
Число значений **Y** должно быть не менее двух

Gaussian Peak Fit



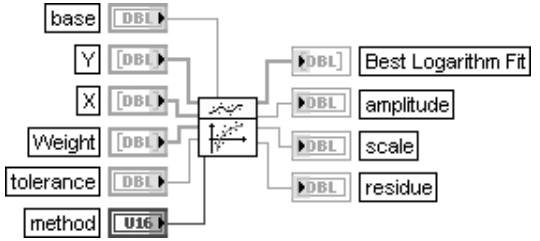
Аппроксимация гауссоидой



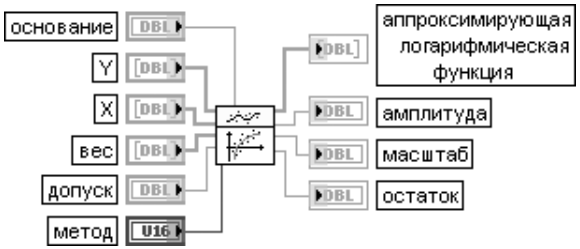
ВП производит аппроксимацию набора входных данных **Y** и **X** гауссовской функцией, выводит значения аппроксимирующей функции на выход **аппроксимирующая гауссоида** (Best Gaussian Fit), ее коэффициенты – на выходы **амплитуда** (amplitude), **центр** (center) и **стандартное отклонение** (standard deviation), а также выводит взвешенную среднюю ошибку аппроксимации на выход **остаток** (residue).
Число значений **Y** должно быть не менее трех.

Вход **начальная оценка** (initial guess) определяет начальные значения **амплитуды, центра** и **стандартного отклонения** для использования в итерационном алгоритме. Если **начальная амплитуда** (initial amplitude), **начальный центр** (initial center) или **начальное стандартное отклонение** (initial standard deviation) равны NaN, этот ВП вычисляет начальные оценки автоматически

Logarithm Fit

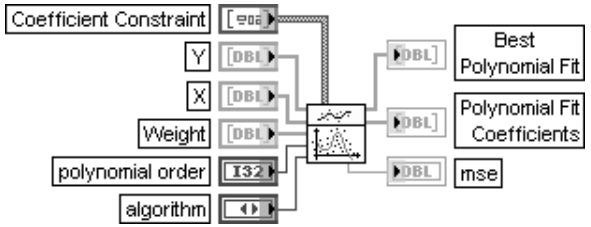


Логарифмическая аппроксимация

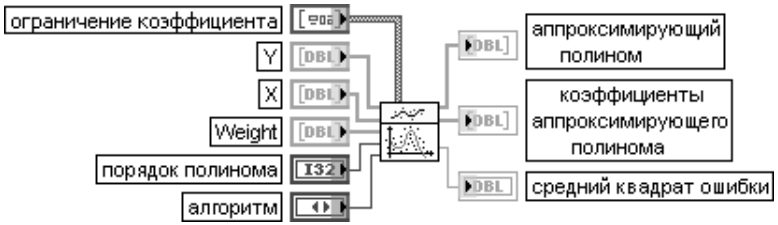


ВП производит аппроксимацию набора входных данных **Y** и **X** логарифмической функцией, выводит значения аппроксимирующей функции на выход **аппроксимирующая логарифмическая функция** (Best Logarithm Fit), ее коэффициенты – на выходы **амплитуда** (amplitude) и **масштаб** (scale), а взвешенную среднюю ошибку аппроксимации – на выход **остаток** (residue).
Число значений **Y** должны быть не менее двух.
Вход **основание** (base) определяет основание логарифма. По умолчанию используется натуральный логарифм.
Выражение для аппроксимирующей функции выглядит следующим образом:
$f = a \log_c(bx)$, где x – входная последовательность **X**, c – основание, a – амплитуда и b – масштаб

General Polynomial Fit



Полиномиальная аппроксимация общего вида



ВП находит значения **аппроксимирующего полинома** (Best Polynomial Fit) и **коэффициенты аппроксимирующего полинома** (Polynomial Fit Coefficients), которые определяют полином, наилучшим образом аппроксимирующий набор входных данных **X** и **Y**.
Число точек входных наборов данных **X** и **Y** должно быть больше **порядка полинома** (polynomial order). При нарушении этого условия ВП передает на выход **коэффициенты аппроксимирующего полинома** пустой массив и возвращает ошибку.
Вход **ограничение коэффициента** (Coefficient Constraint) в виде массива кластеров задает ограничения на коэффициенты полинома при определенных степенях с помощью установления заданного **коэффициента** (coefficient) при определенной **степени** (order). **Ограничение коэффициента** рекомендуется использовать, если точно известны значения определенных коэффициентов полинома.
Порядок полинома (polynomial order) должен быть больше или равен нулю. По умолчанию порядок полинома равен 2. Еще одно ограничение связано с числом точек n входных последовательностей: $0 \le m < n - 1$, где m – порядок полинома.
Вход **алгоритм** (algorithm) определяет алгоритм, используемый ВП для расчета **аппроксимирующего полинома**. Перечень используемых алгоритмов и их номера приведены в таблице.

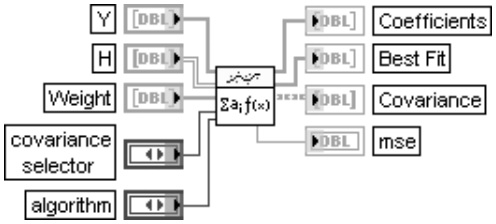
0	SVD (default)	**2**	Givens2	**4**	LU decomposition
1	Givens	**3**	Householder	**5**	Cholesky

Общее число **коэффициентов аппроксимирующего полинома** равно $m + 1$, где m – порядок полинома.
Общий вид полиномиальной аппроксимирующей функции задается выражением

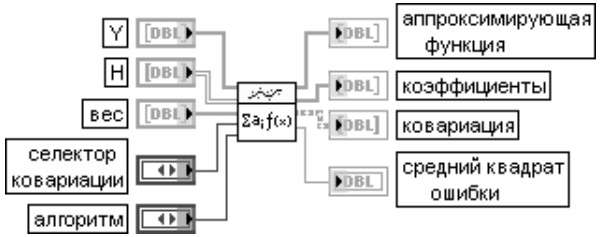
$$f_i = \sum_{j=0}^{m} a_j x_i^j,$$

где F – выходная последовательность **аппроксимирующий полином**, x – входная последовательность **X**, a – **коэффициенты аппроксимирующего полинома**, m – **порядок полинома**

General LS Linear Fit



Линейная аппроксимация общего вида методом наименьших квадратов



ВП находит значения и набор коэффициентов k-мерной линейной кривой, которая наилучшим образом аппроксимирует набор входных данных с помощью метода наименьших квадратов. Под линейной кривой понимается кривая, образованная суммой функций с линейными коэффициентами.
Вход **H** служит для передачи **матрицы наблюдений** (Observation Matrix), которая представляет выражение, используемое для аппроксимации набора данных **{X,Y}**. **H[i][j]** является функцией значений **X[i]**.
Вход **Y** представляет наблюдаемый набор данных **Y**. Число элементов **Y** должно быть равно числу строк матрицы **H**.
Вход **селектор ковариации** (covariance selector) устанавливает разрешение на вычисление **ковариационной матрицы** (Covariance matrix).
Функция входа **алгоритм** рассмотрена ранее при описании ВП **General Polynomial Fit**.
Выход **коэффициенты** (Coefficients) отображает набор коэффициентов, которые минимизируют значение суммы квадратов отклонений χ^2

$$\chi^2 = \sum_{i=0}^{n-1}\left(\frac{y_i - z_i}{\sigma_i}\right)^2 = \sum_{i=0}^{n-1}\left(\frac{y_i - \sum_{j=0}^{k-1} b_j x_{ij}}{\sigma_i}\right)^2 = \left|H_0 B - Y_0\right|.$$

Выход **аппроксимирующая функция** (Best Fit) возвращает значения аппроксимирующей функции, рассчитанной с помощью **коэффициентов** (Coefficients).
На выходе **mse** отображается **средний квадрат ошибки** аппроксимации.
Выход **ковариация** (Covariance) представляет матрицу коэффициентов ковариации **C** размером $k \times k$ элементов. c_{jk} является коэффициентом ковариации между a_j и a_k. c_{jj} является дисперсией a_j.
Применение ВП **Линейная аппроксимация общего вида методом наименьших квадратов** (General LS Linear Fit) в LabVIEW поясняется с помощью ВП **Аппроксимация методом наименьших квадратов общего вида** (General LS Fitting), находящегося в библиотеке regressn (папка примеров analysis).
Блок-диаграмма этого ВП приведена на рис. 3.5, а его лицевая панель – на рис. 3.6.

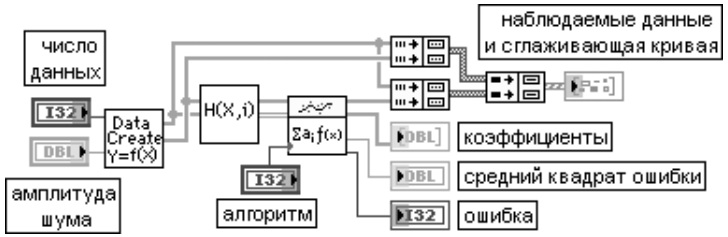


Рис. 3.5. Блок-диаграмма ВП ***Аппроксимация общего вида методом наименьших квадратов*** *(General LS Fitting)*

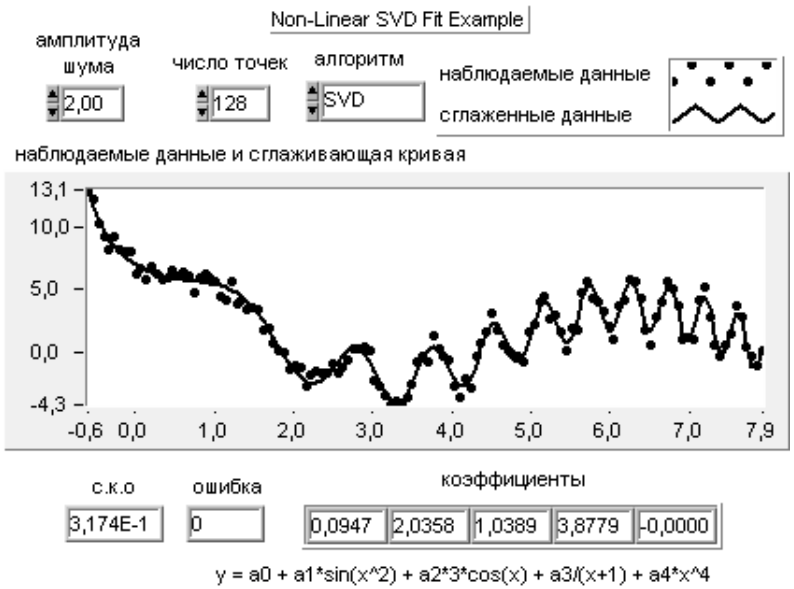


Рис. 3.6. Лицевая панель ВП ***Аппроксимация общего вида методом наименьших квадратов*** *(General LS Fitting)*

Целью ВП является нахождение с помощью метода наименьших квадратов коэффициентов $\boldsymbol{a}$, которые позволяют наилучшим образом представить набор исходных данных $(\boldsymbol{x}, \boldsymbol{y})$. Зависимость между $\boldsymbol{x}$ и $\boldsymbol{y}$ задается в виде выражения

$$y = f(a,x) = \sum_{i=0}^{m-1} a_i f_i(x) = a_0 f_0(x) + a_1 f_1(x) + ... + a_{m-1} f_{m-1}(x),$$

где $\boldsymbol{a} = \{a_0, a_1, a_2, ... a_{m-1}\}$, $\boldsymbol{m}$ – общее число функций.
Пусть данные формируются с помощью выражения
$\boldsymbol{y = 2h_0(x) + 3h_1(x) + 4h_2(x) + noise}$,

где $\boldsymbol{h_0(x) = \sin(x^2)}$, $\boldsymbol{h_1(x) = \cos(x)}$, $\boldsymbol{h_2(x) = \frac{1}{x+1}}$, $\boldsymbol{noise}$ – набор случайных чисел.

Пусть в то же время функция, используемая для аппроксимации данных, имеет вид
$\boldsymbol{y = a_0 f_0(x) + a_1 f_1(x) + a_2 f_2(x) + a_3 f_3(x) + a_4 f_4(x)}$,

где $\boldsymbol{f_0(x) = 1,0}$, $\boldsymbol{f_1(x) = \sin(x^2)}$, $\boldsymbol{f_2(x) = \cos(x)}$, $\boldsymbol{h_2(x) = \frac{1}{x+1}}$, $\boldsymbol{f_4(x) = x^4}$.

Для получения коэффициентов $\boldsymbol{a}$ необходимо на вход ВП **Аппроксимация общего вида методом наименьших квадратов** (General LS Fitting) подать совокупность данных **X** и **Y**, а также матрицу наблюдений **H(X,i)**. Блок-диаграмма подприбора LinBasFns, выполняющего формирование матрицы **H(X,i)**, приведена на рис. 3.7.
Из показаний индикаторов на лицевой панели ВП **Аппроксимация общего вида методом наименьших квадратов** (General LS Fitting) можно сделать вывод, что в результате выполнения аппроксимации вместо истинного набора коэффициентов $\boldsymbol{a} = \{0,0\ 2,0\ 1,0\ 4,0\ 0,0\}$ возвращается их достаточно близкая оценка.
В приведенном примере погрешность аппроксимации слабо зависит от выбора алгоритма. Однако это не исключает наличия оптимального алгоритма для определенного вида данных

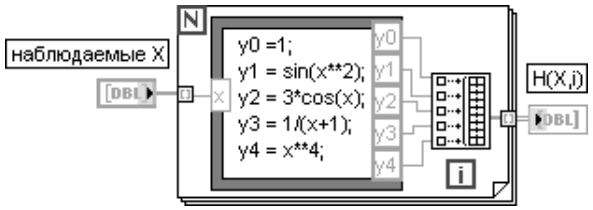
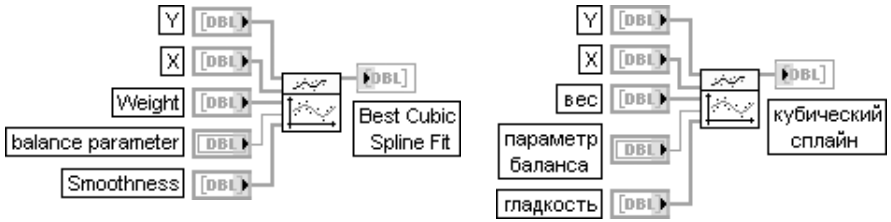


Рис. 3.7. Блок-диаграмма подприбора LinBasFns

Cubic Spline Fit / Аппроксимация кубическим сплайном



ВП использует кубический сплайн для аппроксимации набора входных данных **Y** и **X** в соответствии с величиной **параметра баланса**.
При аппроксимации минимизируется следующая функция:

$$p\sum_{i=0}^{n-1} w_i (y_i - f(x_i))^2 + (1-p)\int_{x_0}^{x_{n-1}} \lambda(x)(f''(x))^2 dx,$$

где $\boldsymbol{p}$ – параметр баланса, $\boldsymbol{w_i}$, $\boldsymbol{y_i}$, $\boldsymbol{x_i}$ – соответственно $\boldsymbol{i}$-е элементы **веса**, **Y** и **X**, $\boldsymbol{f'(x)}$ – вторая производная функции кубического сплайна, $\boldsymbol{\lambda(x)}$ – функция кусочных констант, $\boldsymbol{\lambda(x) = \lambda_i}$ при $\boldsymbol{x_i < x < x_{i+1}}$ для $\boldsymbol{i = \overline{0, n-2}}$.
Вход **параметр баланса** (balance parameter) определяет баланс между гладкостью кубического сплайна и точностью, с которой аппроксимируются наблюдения. **Параметр баланса** должен находиться в диапазоне от 0 до 1. Если значение параметра равно 0, аппроксимация кубическим сплайном эквивалентна линейной аппроксимации (наиболее гладкая аппроксимация, но расстояние до точек максимально). Если же значение **параметра баланса** равно 1, то кубический сплайн выполняет линейную интерполяцию точек (расстояние до точек минимально, но гладкость наихудшая).
Если **параметр баланса** выходит за пределы [0, 1], данный ВП рассчитывает соответствующее значение этого параметра автоматически в соответствии со значениями **X**.
Вход **гладкость** (Smoothness) управляет гладкостью каждого интервала между двумя соседними точками. Чем больше **гладкость**, тем более гладкой является кривая. Число элементов в массиве **гладкость** должно быть равным $\boldsymbol{n-1}$, где $\boldsymbol{n}$ – длина **Y**. Если вход **гладкость** не подключен, этот ВП устанавливает все элементы этого массива равными 1

Fitting on a Sphere / Аппроксимация на сфере

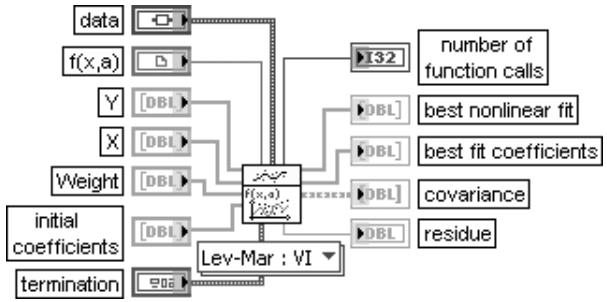
ВП определяет наилучшую сферическую аппроксимацию для области точек в трехмерном пространстве.

Входы **X**, **Y**, и **Z** задают соответственно **x**, **y** и **z** координаты точек в области.
Выходы **x_0, y_0, z_0** отображают рассчитанные координаты центра области.
Выход **r** представляет рассчитанное значение радиуса области.
Функционал минимизации выглядит следующим образом:

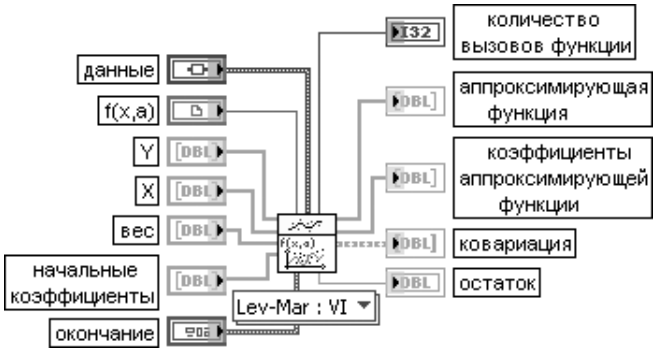
$$\sum_{i=1}^{n}((x_i-x_0)^2+(y_i-y_0)^2+(z_i-z_0)^2-r^2)^2=\min,$$

где (y_i, x_i, z_i) – заданные точки, (y_0, x_0, z_0) – неизвестные координаты центра, r – неизвестный радиус

Nonlinear Curve Fit



Нелинейная аппроксимация



Полиморфный ВП использует алгоритм Левенберга-Марквардта для определения методом наименьших квадратов набора коэффициентов, которые позволяют наилучшим образом аппроксимировать набор входных данных (**X,Y**) нелинейной функцией $y=f(x,a)$, где a – набор коэффициентов.
С помощью селектора этого полиморфного ВП пользователь может установить метод определения нелинейной функции – формулу в строковом элементе или ВП, содержащий модель.
В первом случае модельная аппроксимирующая функция задается во входном кластере **описание модели** (model description) двумя элементами – массивом строк коэффициентов аппроксимации и строковым элементом с независимой переменной.
Во втором случае используется вход $f(x,a)$, передающий ссылку строгого типа на ВП, который содержит модель аппроксимации, и вход **данные** (data), передающий статические данные, которые могут потребоваться определяемой пользователем функции во время выполнения. (Такую ссылку можно создать с помощью функции **Статическая ссылка на ВП** (Static VI Reference).)

Для создания ВП с моделью из шаблона рекомендуется использовать шаблон ВП LM model function and gradient, который находится по следующему пути: labview\vi.lib\gmath\NumericalOptimizationIlb. Соединительная панель этого ВП показана в нижней части таблицы.
Модельный ВП принимает значения независимой переменной **X** и массив коэффициентов a и возвращает функцию модели $f(X;a_1,a_2...a_M)$ и массив частных производных по коэффициентам

$$\left(\frac{df}{da_1}\right),\left(\frac{df}{da_2}\right)...\left(\frac{df}{da_M}\right).$$

Расчет частных производных не обязателен. Если их расчет в ВП не произведен и к выходу $f'(X,a)$ подключен пустой индикатор, ВП **Нелинейная аппроксимация** рассчитывает производные численно.
Вход **X** содержит массив данных, представляющих независимую переменную x. Число входных точек должно быть больше 0 и больше числа заданных коэффициентов. Данные на входе необходимо масштабировать так, чтобы интервал изменения переменной был, по крайней мере, равен 1E-2.
Вход **Y** содержит массив данных, представляющих зависимую переменную y.
На входе **начальные коэффициенты** (Initial Guess Coefficients) задаются начальные значения искомых коэффициентов. Успех использования данной функции аппроксимации во многом зависит от близости начальных коэффициентов к истинным значениям.
Вход **окончание** (termination) определяет условия остановки процесса аппроксимации. В состав кластера **окончание** входят два параметра: **максимальное число итераций** (max iteration) и **допуск** (tolerance).
Если ВП достигает максимального числа итераций без нахождения решения, то возвращается ошибка. В этом случае для получения решения пользователь должен увеличить **максимальное число итераций** или уточнить **начальные коэффициенты**.
Выход **ковариация** (Covariance) отображает матрицу коэффициентов ковариации **C**. c_{jk} является коэффициентом ковариации между a_j и a_k. c_{ij} является дисперсией a_j.
Выход **коэффициенты аппроксимирующей функции** (Best Fit Coefficients) отображает набор коэффициентов, позволяющих минимизировать сумму квадратов отклонений χ^2. Значение χ^2 определяется следующим образом:

$$\chi^2=\sum_{i=0}^{n-1}\left(\frac{y_i-f(x_i,a_1...a_m)}{\sigma_i}\right)^2.$$

Выход **аппроксимирующая функция** (Best Fit) отображает значения аппроксимирующей функции. Расчет значений производится в соответствии с выражением $s_i=f(x_i)A$, где A – набор коэффициентов **коэффициенты аппроксимирующей функции** (Best Fit Coefficients)

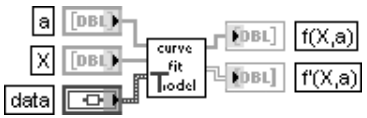
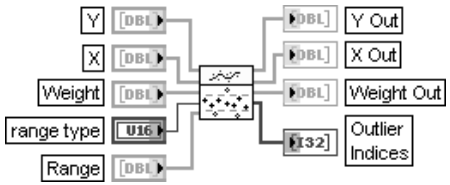
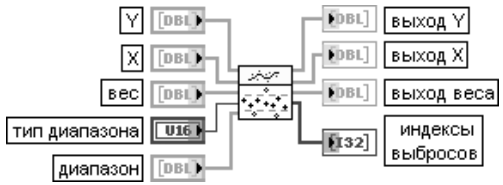


Рис. 3.8. Схема подключения шаблона ВП LM model function and gradient

Remove Outliers



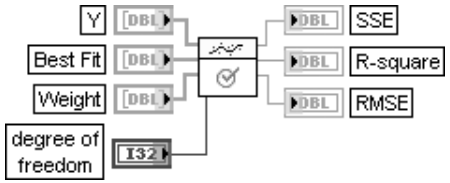
Удалить выбросы



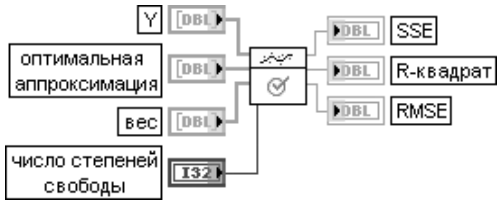
ВП удаляет точки данных, которые попадают за границы заданного диапазона или заданного массива индексов.
Вход **тип диапазона** (range type) определяет оси, к которым прикладываются границы из **диапазона**.

Вход **диапазон** (Range) определяет верхнюю и нижнюю границы диапазона. Этот ВП удаляет данные, которые лежат на границах или за границами **диапазона.** Если на входе **тип диапазона** установлен **Y** или **X,** этот ВП использует первые два элемента диапазона в качестве верхней и нижней границ соответственно для заданных осей. Если же на этом входе установлены **X** и **Y,** то этот ВП использует первый и второй элементы диапазона в качестве верхней и нижней границ оси **x**, а третий и четвертый элемент – в качестве верхней и нижней границ оси **y**.
Выходы **Y**, **X** и **выход веса** возвращают массивы соответственно зависимых и независимых переменных, а также весовых коэффициентов, из которых удалены выбросы

Goodness of Fit



Качество сглаживания



ВП вычисляет три статистических параметра: **SSE, R-квадрат** (R-square) и **RMSE**, которые описывают близость аппроксимирующей функции к исходному набору данных.
Вход **степень свободы** (degree of freedom) определяется как длина последовательности **Y** минус число коэффициентов аппроксимирующей модели. По умолчанию значение равно –1. Если **степень свободы** меньше или равна 0, этот ВП устанавливает ее значение равным длине **Y** минус 2.
Выходные параметры рассчитываются следующим образом:

$$SSE = \sum_{i=0}^{n-1} w_i (y_i - f_i)^2,\ R-square = 1 - \frac{SSE}{SST},\ RMSE = \sqrt{\frac{SSE}{DOF}},$$

$$SST = \sum_{i=0}^{n-1} w_i (y_i - \bar{y})^2,$$ где DOF – число степеней свободы, $\bar{y}$ – среднее значение **Y**.

Чем меньше значение параметров ***SSE*** и ***RMSE*** и чем ближе значение $R - square$ к 1, тем лучше аппроксимация

В состав палитры функций аппроксимации данных входит Экспресс-ВП **Аппроксимация кривой** (Curve Fitting), рассмотренный ниже.

Аппроксимация кривой (Curve Fitting)

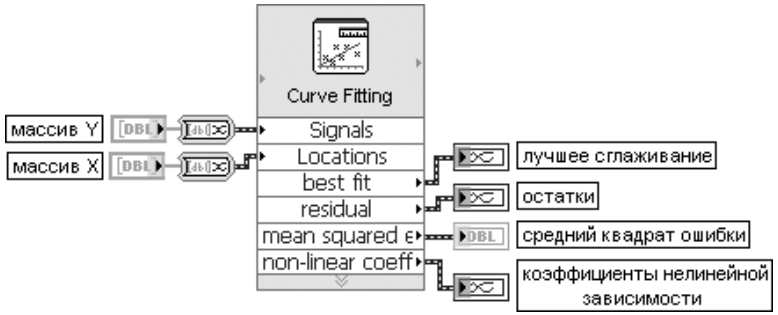


Рис. 3.9. Блок-диаграмма возможного подключения Экспресс-ВП

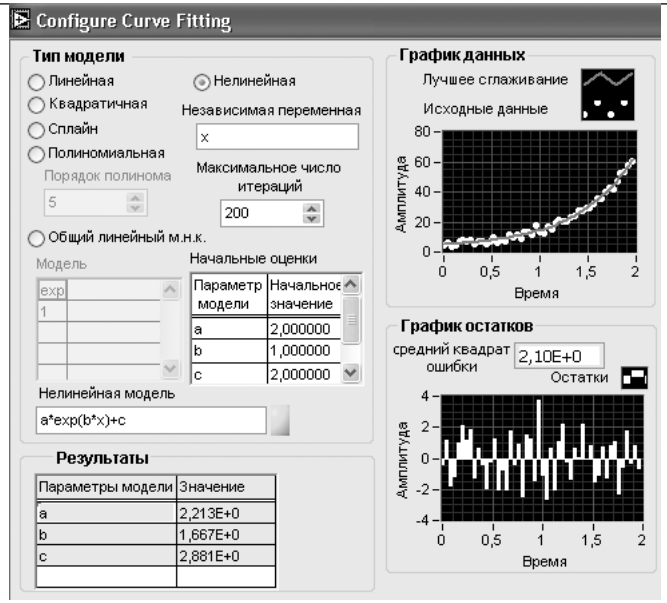


Рис. 3.10. Вид диалогового окна конфигурирования Экспресс-ВП ***Аппроксимация кривой*** *(Curve Fitting)*

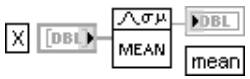
Данный Экспресс-ВП использует функциональность следующих ВП и функций: **Линейная аппроксимация** (Linear Fit), **Полиномиальная интерполяция** (Polynomial Interpolation) и **Сплайн-интерполяция** (Spline Interpolation)

3.3. Функции статистической обработки данных

Функции основной палитры статистической обработки данных (рис. 3.11а) позволяют рассчитать основные статистические параметры наборов данных: среднее, среднеквадратическое отклонение, дисперсию, среднеквадратичное значение, центральные моменты различного порядка, медиану и моду, а также рассчитать значения гистограммы набора данных. Дополнительные подпалитры (рис. 3.11б–3.11г) содержат функции расчета вероятностей (интегральных распределений) для различных законов распределения и функции расчета квантилей распределений по заданным значениям вероятности, проверки гипотез и дисперсионного анализа.

Ниже приведено описание функций основной палитры статистической обработки данных.

Mean



Среднее

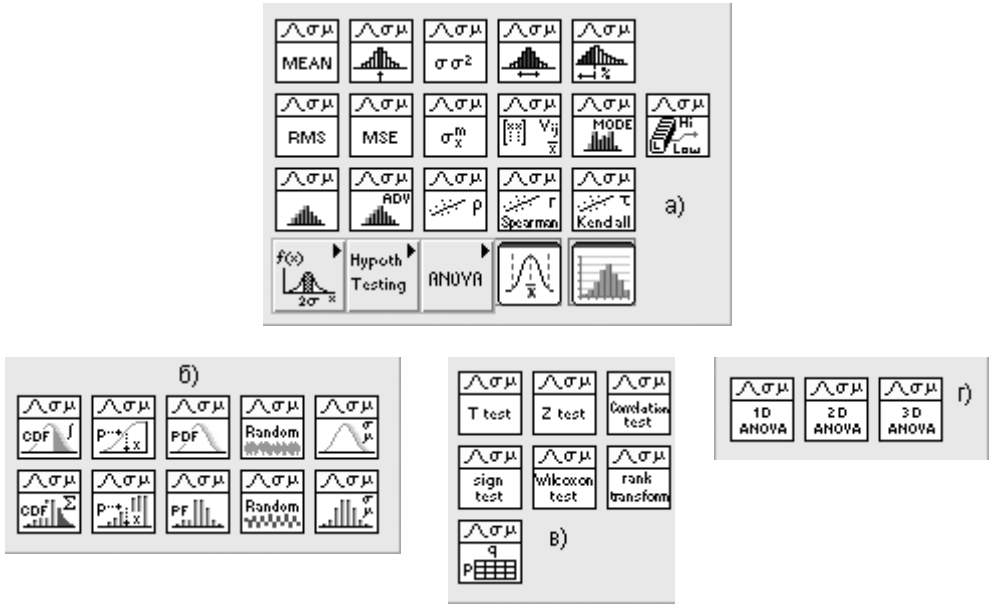
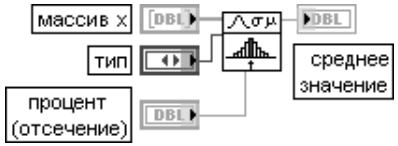


Рис. 3.11. Вид основной палитры (а) и дополнительных подпалитр (б)–(г) функций статистической обработки данных

ВП производит расчет **среднего** (mean) входной последовательности **X** в соответствии с выражением $среднее = \frac{1}{n}\sum_{i=0}^{n-1} x_i$, где n – число элементов последовательности **X**

Measures of Mean

Меры среднего



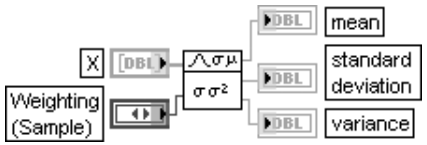
ВП рассчитывает основную тенденцию значений данных **массива x** (x array).
Вход **тип** (type) определяет тип рассчитываемой средней величины:

0	**Арифметическое** (arithmetic) – рассчитывает среднее арифметическое **массива x**
1	**Геометрическое** (geometric) – рассчитывает среднее геометрическое **массива x**
2	**Гармоническое** (harmonic) – рассчитывает среднее гармоническое **массива x**
3	**Усеченное** (trimmed) – рассчитывает среднее арифметическое **массива x** после удаления части возможных выбросов, заданной на входе **процент (отсечение)** (percent (trim))
4	**Медиана** (median) – рассчитывает центральное значение **массива x**. Медиана также представляет 50-й процентиль

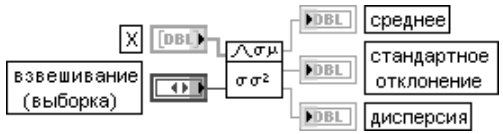
Вход **процент (отсечение)** (percent (trim)) задает общую долю отсекаемых выбросов при расчете **усеченного** (trimmed) среднего. LabVIEW отсекает половину значений, заданных на входе

процент, со стороны наименьших значений массива **x** и вторую половину – со стороны наибольших значений.
Усеченное среднее и медиана более устойчивы к выбросам данных и обычно более надежны (робастны) по сравнению с другими видами оценок среднего

Standard Deviation and Variance



Стандартное отклонение и дисперсия

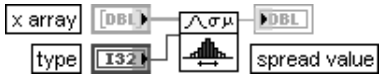


ВП рассчитывает **среднее** (mean), **стандартное отклонение** (среднеквадратичное отклонение) (standard deviation) и **дисперсию** (variance) значений входной последовательности **X**.
Вход **взвешивание** (Weighting) определяет нормирующий коэффициент при расчете стандартного отклонения и дисперсии. При установке варианта **выборка** (Sample) нормирующий коэффициент w равен $n-1$, а при установке **совокупность** (Population) – $w=n$, где n – число значений последовательности **X**. Состояние входа **взвешивание** по умолчанию – **выборка**.
Значение выхода **среднее** рассчитывается так же, как и в ВП **Среднее** (Mean).
Значение выхода **стандартное отклонение** рассчитывается в соответствии с выражением

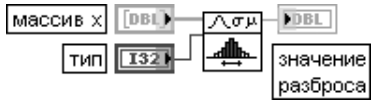
$\sigma = \sqrt{\frac{\sum_{i=0}^{n-1}(x_i - \mu)^2}{w}}$, где μ – среднее.

Значение выхода **дисперсия** определяется как σ^2

Measures of Spread



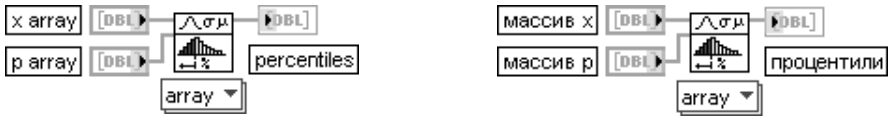
Меры разброса



ВП рассчитывает разброс данных **массива x**.
Вход **тип** (type) устанавливает вид рассчитываемого разброса:

0	**Стандартное отклонение** (standard deviation) – рассчитывает стандартное отклонение или корень квадратный из дисперсии **массива x**. Стандартное отклонение – наиболее общая мера разброса набора данных
1	**Диапазон** (range) – рассчитывает диапазон или разность между максимальным и минимальным значениями **массива x**
2	**Среднее абсолютное отклонение** (mean absolute deviation) – рассчитывает среднее абсолютное отклонение **массива x** от его среднего значения
3	**Межквартильный интервал** (interquartile rank) – рассчитывает разность между 25-м и 75-м процентилем **массива x**. Поэтому данная мера разброса более устойчива к выбросам по сравнению с другими мерами

Percentiles **Процентили**



ВП рассчитывает значение, превышающее значения ***p*** процентов данных в **массиве x**. Тип данных (скаляр или массив), подключенных ко входу ***p***, определяет используемый вариант полиморфной реализации.
На рис. 3.12 приведена блок-диаграмма ВП **Процентили** (Percentiles)

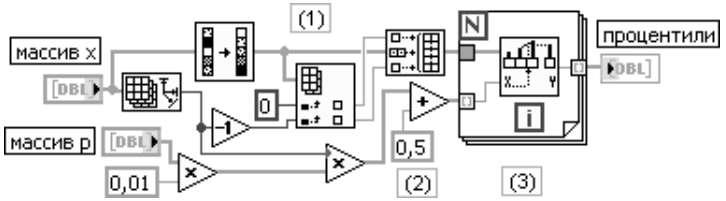
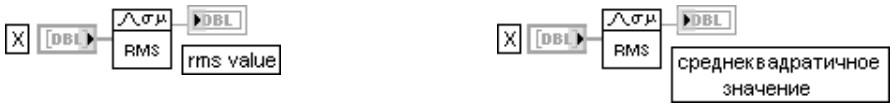


*Рис. 3.12. Блок-диаграмма ВП **Процентили** (Percentiles)*

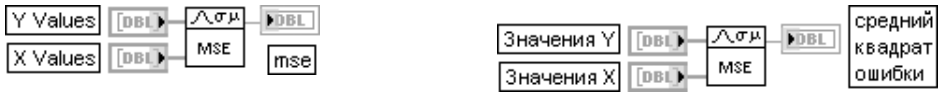
Пояснения к блок-диаграмме:
1. Сортирует элементы массива **x** по возрастанию и дублирует максимальный и минимальный элементы.
2. Рассчитывает дробные индексы элементов массива **x**, соответствующие заданным вероятностям ***p***.
3. Рассчитывает процентили путем интерполяции значений массива **x** при заданных дробных индексах

RMS **Среднеквадратичное значение**



ВП рассчитывает **среднеквадратичное значение** (RMS) входной последовательности **X** по формуле $\psi_x = \sqrt{\frac{1}{n}\sum_{i=0}^{n-1} x_i^2}$

MSE **Средний квадрат ошибки**



ВП рассчитывает **средний квадрат ошибки** (mse) по значениям входных последовательностей **X Values** и **Y Values** в соответствии с выражением $mse = \frac{1}{n}\sum_{i=0}^{n-1}(x_i - y_i)^2$.

Значения Y (Y Values) и **значения X** (X Values) являются массивами значений. Если число элементов в одном массиве отличается от числа элементов в другом, то ВП рассчитывает **средний квадрат ошибки**, ориентируясь на более короткий массив

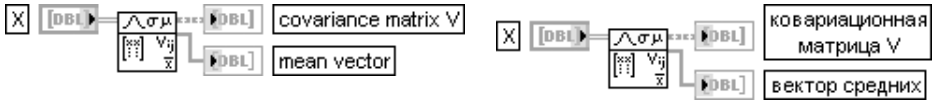
Moment about Mean **Центральный момент**

ВП рассчитывает **момент** (moment) относительно среднего входной последовательности **X**, используя заданный **порядок** (order), ***m***.
Вход **порядок** должен быть больше 0. Если вход **порядок** меньше или равен 0, то ВП выводит на выход **момент** значение NaN и возвращает ошибку. По умолчанию значение порядка равно 2.
ВП вычисляет момент ***m***-го порядка, используя следующее выражение:

$$\sigma_x^m = \frac{1}{n}\sum_{i=0}^{n-1}(x_i - \mu)^m,$$

где ***n*** – число элементов последовательности **X**

Covariance Matrix **Ковариационная матрица**



ВП рассчитывает ковариационную матрицу входной последовательности X.
Каждая колонка **X** представляет один вектор наблюдаемых выборок одной переменной. Каждая строка **X** содержит одну выборку из каждой переменной.
Выход **ковариационная матрица V** (covariance matrix V) возвращает ковариационную матрицу **X**. Если **X** представляет двумерный массив $n \times m$, то ковариационная матрица будет квадратной матрицей $m \times m$.
Вектор средних (mean vector) возвращает среднее значение каждой колонки (переменной) **X**.
Для ***m*** заданных векторов наблюдаемых выборок, где колонка ***l*** содержит случайную величину x_i, ковариационная матрица определяется следующим образом:
$V_{ij} = cov(x_i, x_j) = (x_i - \mu_i)(x_j - \mu_j)$,
где μ_i – среднее случайной величины x_i. Каждый элемент V_{ij} **ковариационной матрицы V** представляет ковариацию между величинами x_i и x_j. Диагональ **ковариационной матрицы V** содержит стандартные отклонения каждой величины x_i.
Блок-диаграмма ВП **Ковариационная матрица** приведена на рис. 3.13.

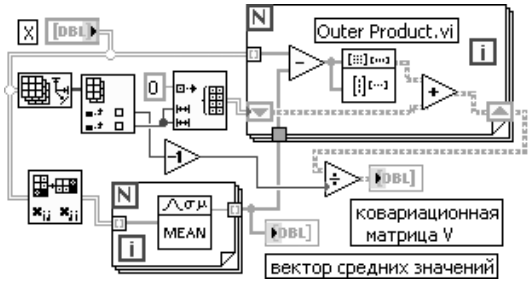
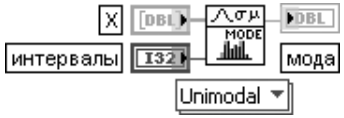


*Рис. 3.13. Блок-диаграмма ВП **Ковариационная матрица***

Mode

Мода



Этот полиморфный ВП находит оценку **моды** (mode) входной последовательности **X**.
Вход **X** должен содержать по крайней мере одну выборку. Если входная последовательность имеет постоянное значение, то ВП игнорирует число интервалов и устанавливает значение моды равным этому значению.
Вход **интервалы** (intervals) определяет число интервалов гистограммы. Число интервалов должно быть больше 0. По умолчанию оно равно 1.
Выход **мода** (mode) возвращает среднее значение интервала гистограммы, имеющего максимальную величину, или, другими словами, наиболее часто встречающееся значение последовательности

Median

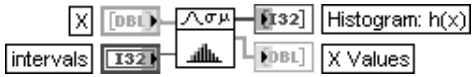
Медиана

ВП находит **медиану** (median) входной последовательности **X** путем сортировки значений **X** и выбора среднего элемента(ов) из отсортированного массива.
Значение **медианы** рассчитывается следующим образом:

$$медиана = \begin{cases} s_i & если\ n\ нечетно \\ 0{,}5(s_{k-1} + s_k) & если\ n\ четно \end{cases},$$

где n – число элементов входной последовательности **X**, s – отсортированная последовательность, $i = (n-1)/2$, $k = n/2$

Histogram



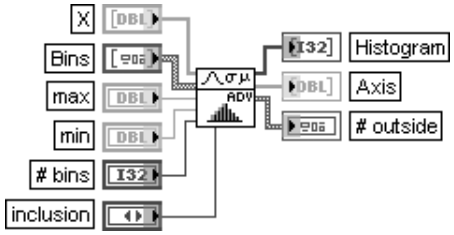
Гистограмма

ВП находит гистограмму входной последовательности **X**.
Вход **X** должен содержать по крайней мере один отсчет. Если вход **X** пустой, то ВП устанавливает на выходах **гистограмма: h(x)** (Histogram: h(x)) и **значения X** (X Values) пустые массивы и возвращает ошибку.
Вход **интервалы** (intervals) должен быть больше нуля. По умолчанию его значение равно 1. Если вход меньше или равен 0, то ВП также устанавливает на выходах **гистограмма: h(x))** и **значения X** пустые массивы и возвращает ошибку.
Выход **гистограмма: h(x)** (Histogram: h(x)) отображает массив значений гистограммы входной последовательности **X**.
Выход **значения X** (X Values) является массивом центров **интервалов** (bins) гистограммы.
Гистограмма представляет распределение частот попадания значений входной последовательности в заданные интервалы. ВП определяет **гистограмму** следующим образом. ВП просматривает входную последовательность для определения диапазона значений и затем оценивает ширину интервала Δx, исходя из заданного числа **интервалов** (intervals), с помощью следующего соотношения $\Delta x = \frac{\max - \min}{m}$, где $\max$, $\min$ – максимальное и минимальное значение входной последовательности **X**, m – требуемое число интервалов.

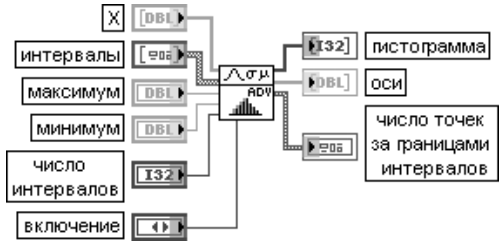
Центры интервалов рассчитываются с помощью выражения $\chi_i = \min + 0{,}5\Delta x + i\Delta x$ для $i = \overline{0, m-1}$.
ВП определяет i-й интервал в соответствии с выражением $\Delta = [*СИМВОЛ* - 0{,}5\Delta x : \chi_i + 0{,}5\Delta x)$ для $i = \overline{0, m-1}$ и определяет функцию $y_i(x) = \begin{cases} 1 & если \quad x \in \supseteq \Delta_i \\ 0 & иначе \end{cases}$. Эта функция имеет единичное значение, если x попадает в заданный интервал.

ВП оценивает последовательность значений гистограммы H, используя выражение $h_i = \sum_{j=0}^{n-1} y_i(x_j)$ для $i = \overline{0, m-1}$, n – число элементов входной последовательности **X**

General Histogram



Гистограмма общего вида



ВП находит гистограмму входной последовательности **X**, основанную на заданном определении интервалов.
Вход **интервалы** (Bins) определяет границы каждого интервала гистограммы. Вход **интервалы** является массивом кластеров, в котором каждый кластер определяет диапазон значений интервала.
- Вход **нижняя** (lower) определяет нижнюю границу интервала.
- Вход **верхняя** (upper) определяет верхнюю границу интервала.
- Вход **включение** (inclusion) определяет вид границ каждого интервала.

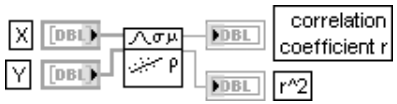
Предусмотрены следующие варианты определения границ:

0	**Нижняя** (lower) – нижняя граница входит в интервал, верхняя не входит
1	**Верхняя** (upper) – верхняя граница входит в интервал, нижняя не входит
2	**Обе** (both) – обе границы входят в интервал
3	**Никакая** (neither) – обе границы исключаются из интервала

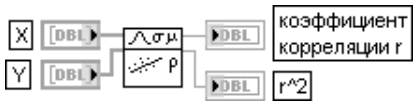
Если параметры на входе **интервалы** не заданы, то входы **max**, **min**, **# bins** и **inclusion** используются для определения равномерно распределенных интервалов.
Входы **max** и **min** определяют максимальное и минимальное значения, включаемые в гистограмму. Если **max** и **min** не определены, то ВП использует максимальное и минимальное значения входной последовательности **X**.
Вход **число интервалов** (# bins) определяет число интервалов гистограммы. Этот вход игнорируется, если входной массив **Bins** не пустой. По умолчанию число интервалов определяется с помощью формулы Старджеса $m = \log_2 n + 1 = 3{,}3\lg n + 1$, где m – число интервалов, n – размер **X**.
Вход **включение** (inclusion) определяет вид границ каждого интервала. При подаче на вход значения 0 в интервал включается нижняя граница, при подаче 1 – верхняя.
Если массив **интервалы** не пустой, то входы **max**, **min**, **# bins**, и **inclusion** игнорируются.
Выход **гистограмма** (Histogram) определяет результирующую гистограмму.
Выход **оси** (Axis) определяет центральные значения для каждого интервала гистограммы.
Центры каждого интервала определяются следующим выражением:
$center[i] = (lower + upper)/2$,
где $lower$ и $upper$ – нижняя и верхняя границы i-го интервала.

Выход **число точек за границами интервала** (**#** outside) содержит информацию о точках, не вошедших в какой-либо интервал после успешного выполнения ВП.
Элементы **выше** (above) и **ниже** (below) кластера **# outside** имеют значение, только когда **интервалы** определены так, что Bins[0].upper ≤ Bins[1].lower < Bins[1].upper, ...– < Bins[k – 1].lower < Bins[k – 1].upper, где k – число элементов на входе **интервалы**.
Элемент **общее число** (total) содержит общее число значений **X**, не попавших в какой-либо интервал.
Элемент **выше** (above) представляет число значений **X**, превышающих верхнюю границу последнего интервала. Последний интервал имеет значение Bins[размер(Bins) – 1].upper.
Элемент **ниже** (below) представляет число значений **X**, находящихся ниже нижней границы первого интервала. Первый интервал имеет значение Bins[0].lower

Correlation Coefficient



Коэффициент корреляции



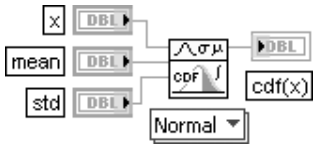
ВП рассчитывает коэффициент линейной корреляции между входными последовательностями **X** и **Y**.
Коэффициент линейной корреляции, также известный как коэффициент корреляции по смешанным моментам, или корреляция Пирсона, рассчитывается следующим образом:

$r = \frac{\sum Z_x Z_y}{n}$, где Z_x, Z_y – стандартизованные значения **X** и **Y**.

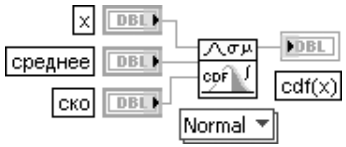
На выходе **r^2** также выводится квадрат коэффициента корреляции

В последующей таблице приведены полиморфные функции расчета вероятностей (интегральных распределений) для различных непрерывных законов распределения, расчета квантилей распределений по заданным значениям вероятности, плотности распределения вероятности, случайных значений и ожидаемых моментов из подпалитры **Вероятность** (Probability). Аналогичный набор функций в этой подпалитре имеется и для дискретных распределений.

Continuous CDF

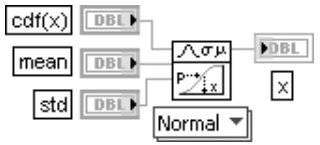


Непрерывная кумулятивная функция распределения

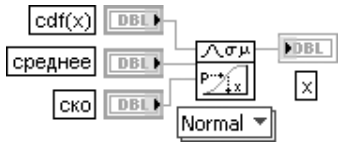


ВП рассчитывает непрерывную кумулятивную функцию распределения или вероятность того, что случайное значение **X** ≤ **x**. Пользователь должен выбрать закон распределения с помощью селектора полиморфного ВП

Continuous Inverse CDF

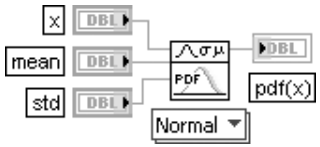


Непрерывная обратная кумулятивная функция распределения

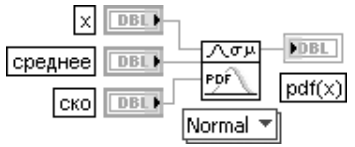


ВП рассчитывает непрерывную обратную кумулятивную функцию распределения (квантили распределения). Пользователь должен выбрать закон распределения с помощью селектора полиморфного ВП

Continuous PDF

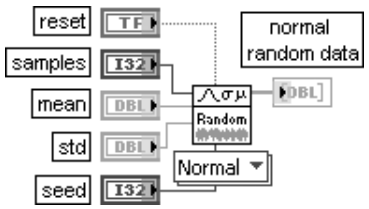


Непрерывная плотность распределения вероятности

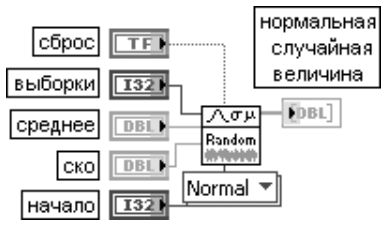


ВП рассчитывает непрерывную плотность распределения вероятности для различных законов распределения

Continuous Random

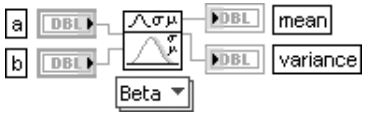


Случайные значения с непрерывным законом распределения

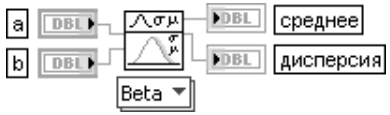


ВП генерирует случайные значения из генеральной совокупности **X** с различными непрерывными законами распределения

Continuous Moments



Моменты непрерывных распределений



ВП возвращает ожидаемые значения среднего и дисперсии случайных величин с заданными законами непрерывных распределений

Contingency Table

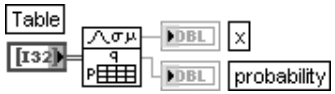
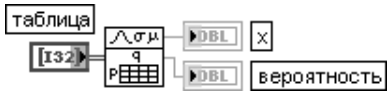


Таблица вероятностей



ВП классифицирует и подсчитывает экспериментальные объекты согласно двум схемам категорирования.

Вход **таблица** (Table) является входом таблицы вероятностей и определяется как массив событий или частот.

Выход ***x*** определяет значение, при котором необходимо интерполировать соответствующее значение ***y***.

Выход **вероятность** (probability) возвращает одностороннюю вероятность истинности гипотезы, проверяемой относительно данных таблицы.

ВП **Таблица вероятностей** (Contingency Table) использует χ^2 тест однородности и χ^2 тест независимости для проверки гипотезы. Перед проверкой гипотезы должна быть определена минимальная вероятность для каждого теста. Минимальное значение вероятности определяет порог принятия или отклонения гипотезы. Обычно минимальная вероятность выбирается на уровне 0,05. Если действительное значение вероятности, возвращаемое ВП на выходе **вероятность**, меньше установленного порога, то считается, что гипотеза должна быть отвергнута.

При выполнении χ^2 теста однородности ВП берет случайную выборку фиксированного размера из каждой категории по одной схеме категорирования. Для каждой из выборок ВП категорирует объекты в соответствии со второй схемой и подсчитывает их. ВП проверяет гипотезу о том, что две генеральные совокупности, из которых взяты выборки, распределены по одинаковому закону по отношению ко второй схеме категорирования.

При выполнении χ^2 теста независимости ВП берет только одну выборку из общей генеральной совокупности. Затем ВП категорирует каждый объект и подсчитывает их по двум схемам категорирования. ВП проверяет гипотезу о том, что схемы категорирования являются независимыми.

Пусть $y_{p,q}$ – число событий в (pq)-й ячейке таблицы вероятностей для $p = \overline{0, s-1}$ и $q = \overline{0, k-1}$, где s и q – число строк и столбцов в таблице вероятностей Table.

Пусть $y_p = \sum_{q=0}^{k-1} y_{p,q}$, $y_q = \sum_{p=0}^{s-1} y_{p,q}$, $y = \sum_{p=0}^{s-1}\sum_{q=0}^{k-1} y_{p,q}$, $e_{p,q} = \frac{y_p, y_q}{y}$, $x = \sum_{p=0}^{s-1}\sum_{q=0}^{k-1} \frac{[y_{p,q} - e_{p,q}]^2}{e_{p,q}}$.

ВП использует x для расчета **вероятности** (probability) p, исходя из выполнения следующего соотношения: $p = Prob\{X \geq x\}$, где X – случайная величина, имеющая χ^2 распределение. Если гипотеза истинна, то x считается имеющим χ^2 распределение с $(s-1)$ и $(k-1)$ степенью свободы

В состав палитры функции статистической обработки данных входят Экспресс-ВП **Статистика** (Statistics) и **Создать гистограмму** (Create Histogram), рассмотренные ниже.

Статистика (Statistics)

Экспресс-ВП **Статистика** (Statistics) возвращает выбранные параметры первого сигнала осциллограммы.

В связи с тем, что параметры, входящие в диалоговое окно конфигурирования, были рассмотрены ранее или определяются известным способом, дополнительные пояснения к блок-диаграмме и опциям окна конфигурирования данного Экспресс-ВП (рис. 3.14, рис. 3.15) далее не приводятся

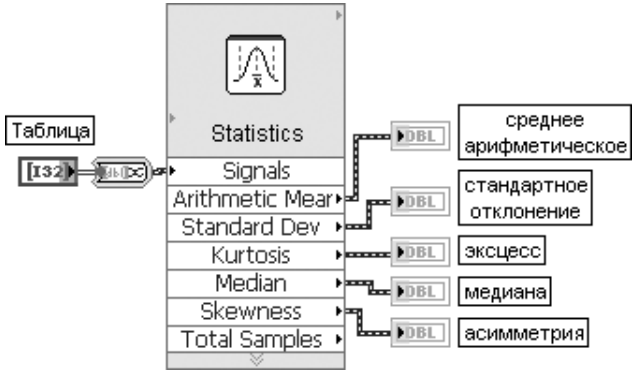


Рис. 3.14. Блок-диаграмма возможного подключения Экспресс-ВП

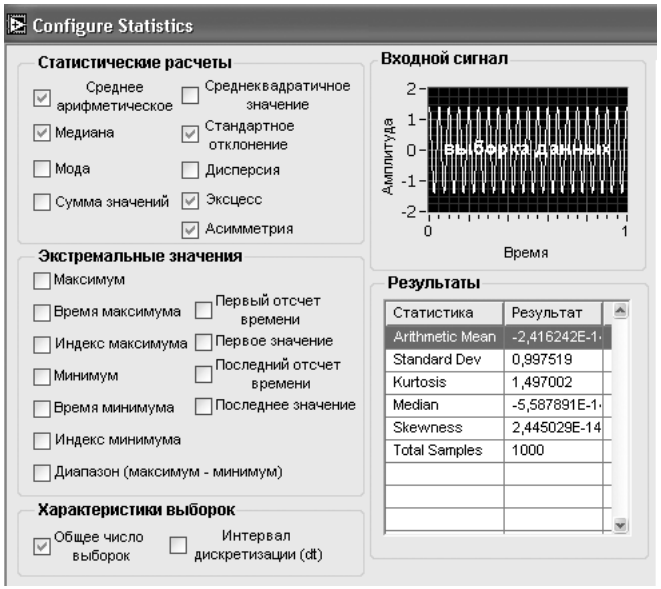


*Рис. 3.15. Вид диалогового окна конфигурирования Экспресс-ВП **Статистика** (Statistics)*

Создать гистограмму (Create Histogram)

Экспресс-ВП **Создать гистограмму** (Create Histogram) рассчитывает гистограмму входного сигнала.

Данный Экспресс-ВП использует функциональность одноименного ВП, рассмотренного выше

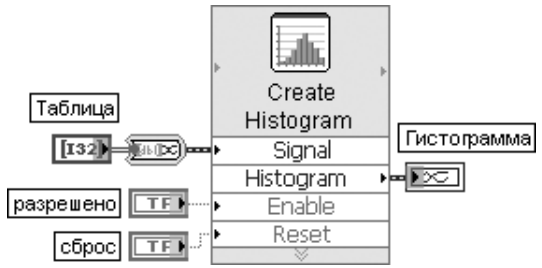


Рис. 3.16. Блок-диаграмма возможного подключения Экспресс-ВП

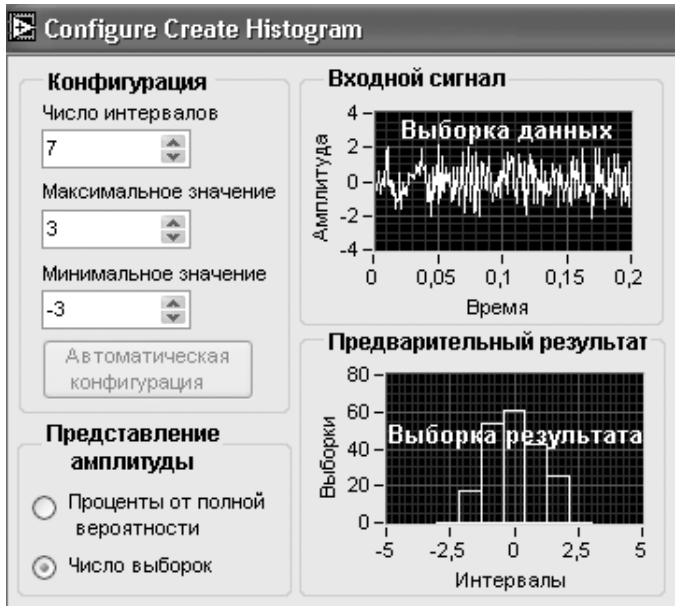


Рис. 3.17. Вид диалогового окна конфигурирования Экспресс-ВП ***Создать гистограмму*** *(Create Histogram)*

3.4. Функции численных методов

К числу функций численных методов отнесены функции, входящие в подпалитры **Интерполяция и экстраполяция** (Interp & Extrap), **Интегрирование и дифференцирование** (Integ & Diff), **Дифференциальные уравнения** (Differential Equations), и **Оптимизация** (Optimization).

3.4.1. Функции интерполяции и экстраполяции

Функции интерполяции и экстраполяции (рис. 3.18) позволяют выполнять одномерную и двумерную интерполяции, кусочно-линейную и полиномиальную интерполяции, а также Фурье-интерполяцию.

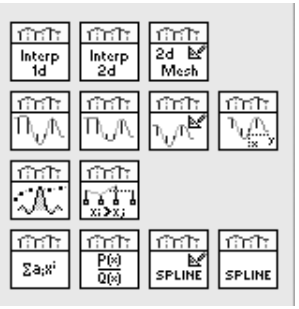
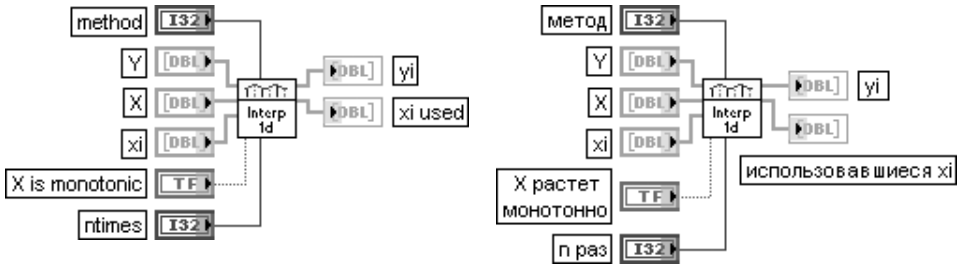


Рис. 3.18. Вид подпалитры функций интерполяции и экстраполяции

Interpolate 1D — Одномерная интерполяция



ВП выполняет одномерную интерполяцию, используя выбранный метод просмотра таблицы. Этот ВП принимает табулированные значения зависимой и независимой переменных **Y** и **X**, предоставляя для каждого значения **xi** соответствующие интерполированные значения **yi**. ВП находит каждое значение **xi** в **X** и использует его относительное положение в **X** для нахождения интерполированного значения **yi** по такому же относительному положению в **Y**.

Вход **метод** (method) определяет метод интерполяции.

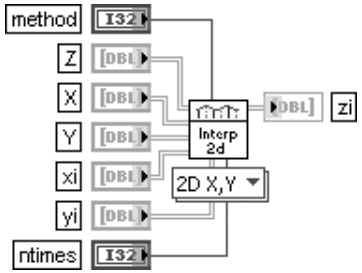
0	**Ближайший** (nearest) – выбирает значение **Y**, соответствующее ближайшему к **xi** значению **X**
1	**Линейный** (linear) – устанавливает интерполированные значения на отрезке линии, соединяющей точки данных **X** и **Y**
2	**Сплайн** (spline) – обеспечивает непрерывность первой и второй производной кубических интерполирующих полиномов, в том числе в точках данных
3	**Кубический Эрмита** (cubic Hermite) – обеспечивает непрерывность первой производной кубических интерполирующих полиномов и устанавливает определенное значение производной в крайних точках для сохранения первоначального положения и монотонности данных **Y**
4	**Лагранжа** (Lagrange) – использует барицентрический (по центру тяжести) интерполяционный полином Лагранжа

Если независимая переменная **X** пуста, LabVIEW определяет ее как **X**[i]=i для i=0...N–1, где N – число элементов в переменной **Y**.

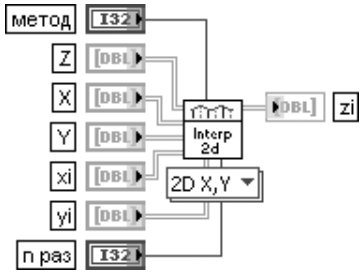
Вход **xi** определяет массив значений независимой переменой, для которых LabVIEW вычисляет интерполированные значения **yi** зависимой переменной.

На входе **X растет монотонно** (X is monotonic) устанавливается состояние ИСТИНА, если заранее известно, что значения **X** монотонно растут с увеличением индекса. Это предотвращает сортировку **X** и соответствующее переупорядочивание **Y**.
Вход **n раз** (ntimes) определяет положение точек интерполяции **xi**, **yi**, получая интерполированные значения между каждым элементом **Y,** если **xi** и **yi** являются пустыми. Интерполяция между элементами **Y** повторяется **n раз**.
Выход **yi** возвращает выходной массив интерполированных значений, которые соответствуют значениям **xi** независимой переменной.
Выход **использовавшиеся xi** (xi used) представляет одномерный массив значений, для которых производился расчет интерполированных значений зависимой переменной

Interpolate 2D



Двумерная интерполяция

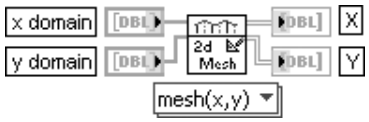


Этот полиморфный ВП выполняет двумерную интерполяцию, используя выбранный метод просмотра таблицы, когда **X**, **Y**, **xi** и **yi** являются двумерными или одномерными матрицами. ВП принимает табулированные значения **X**, **Y** и **Z** (две независимые переменные и одна зависимая) и предоставляет интерполированные значения **zi**, которые соответствуют каждой точке **xi**, **yi**
Вход **метод** (method) определяет метод интерполяции.

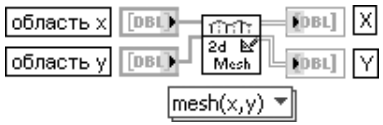
0	**Ближайший** (nearest) – выбирает значение **Z**, соответствующее ближайшему к **xi**, **yi** значению **X**, **Y**. Интерполированное значение устанавливается в ближайшей точке данных
1	**Билинейный** (bilinear) – устанавливает интерполированные значения на отрезках линий, которые соединяют поверхность **X**, **Y** и **Z**
2	**Бикубический** (bicubic) – получает интерполированную точку из бикубической поверхности, которая покрывает шестнадцать ближайших точек **X**, **Y**, **Z** и обеспечивает непрерывность первой производной интерполирующей поверхности
3	**Бикубический сплайн** (bicubic spline) – обеспечивает непрерывность первой и второй производной кубических интерполирующих полиномов, в том числе в точках данных

Селектор полиморфного ВП позволяет выбрать варианты реализации ВП с различной размерностью входных массивов

Create Mesh Grid (2D)

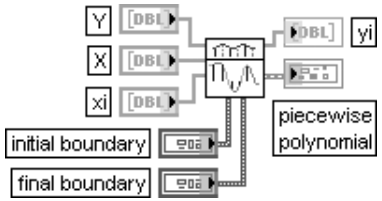


Создать прямоугольную сетку (2D)



ВП получает два массива: **область x** (x domain) и **область y** (y domain) и создает двумерные массивы **X** и **Y**, которые обычно используются для вычисления и построения графиков по двум переменным.
Массив **X** представляет двумерный массив размером N x M, который образуется путем построчного повторения N раз входного массива **область x** размером M элементов.
Массив Y имеет такой же размер, но образуется путем повторения в M столбцах входного массива **область y** размером N элементов

Spline Interpolation 1D



Одномерная сплайн-интерполяция

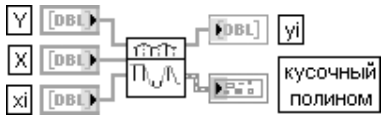
ВП выполняет одномерную интерполяцию, используя метод сплайн-интерполяции с просмотром таблицы.
Метод сплайн-интерполяции обеспечивает непрерывность первой и второй производных интерполирующего кусочного полинома, в том числе и в точках данных.
Вход **начальная граница** (initial boundary) устанавливает начальное граничное условие. Кластер **начальная граница** содержит два элемента:

- **граница** (boundary) устанавливает вид граничного условия из набора: **натуральный сплайн** (natural spline), **без пересечения** (not-a-knot), **первая производная** (first derivative) или **вторая производная** (second derivate);
- **значение производной** (derivative value) содержит значение первой производной на начальной границе.

Вход **конечная граница** (final boundary) имеет аналогичную структуру и назначение.
Помимо интерполированных значений **yi**, этот ВП также возвращает кластер **кусочный полином** (piecewise polynomial), который содержит **координаты x** концов отрезков (x locations) и соответствующие **коэффициенты** полинома (coefficients), использовавшиеся при интерполяции

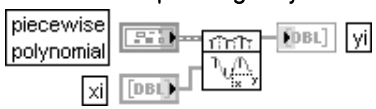
Hermite Interpolation 1D

Одномерная интерполяция Эрмита



ВП выполняет одномерную интерполяцию, используя метод кубической интерполяции Эрмита с просмотром таблицы.
Метод кубической интерполяции Эрмита обеспечивает непрерывность первой производной и устанавливает ее значения в крайних точках с целью сохранения первоначального положения и монотонности данных **Y**

Evaluate Interpolating Polynomial

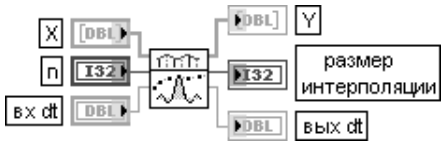


Рассчитать интерполяционный полином

ВП получает кластер **кусочный полином** (piecewise polynomial) и рассчитывает интерполяционные значения **yi** для каждого значения **xi**

Interpolate 1D Fourier

Одномерная интерполяция Фурье



ВП выполняет интерполяцию путем преобразования массива **X** в частотную область, дополнения нулями и масштабирования на соответствующий коэффициент интерполяции с целью сохранения надлежащей симметрии в частотной области. На заключительной стадии для формирования выходного интерполяционного массива **Y** (Y out) производится обратное преобразование во временную область.

На вход **X** подаются табулированные значения, по которым производится интерполяция. Предполагается, что данные в **X** представляют выборки, равномерно распределенные по оси **x**.

Параметр **n** используется как **размер интерполяции** или как **коэффициент интерполяции** в зависимости от установки **типа**.

Параметр **вх dt** (dt in) используется для расчета **вых dt** (dt out) с учетом параметров интерполяции **n** и **тип**.

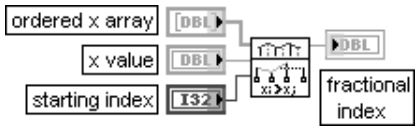
Вход **тип** (type) определяет вариант использования **n** в определении **размера интерполяции**:

0	**Размер интерполяции** (interpolation size) – размер выхода = **n**
1	**Коэффициент интерполяции** (interpolation factor) – размер выхода = **n** * размер **X**

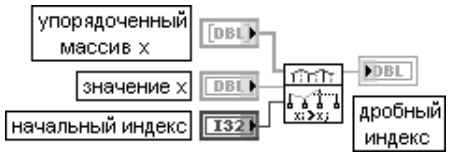
Выход **размер интерполяции** (interpolation size) возвращает размер интерполяционного выходного массива **Y** в зависимости от **n** и **типа**.

Выход **вых dt** (dt out) равен **dt in** * **N**/**m**, где **N** – размер входного массива **X**, а **m** – размер интерполяции (определяется **n** и **типом**)

Search Ordered Table



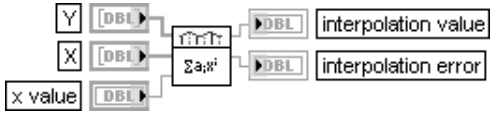
Поиск в упорядоченной таблице



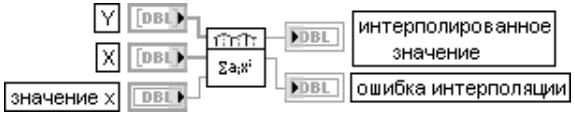
Для работы ВП **упорядоченный массив x** (ordered x array) должен быть монотонно возрастающим или убывающим. Поэтому ВП может эффективно отыскивать относительный индекс расположения **значения x** (x value) в этом массиве.

ВП запускает поиск с **начального индекса** (starting index) и начинает с быстрой фазы поиска для приблизительного определения интервала. Затем интервал сужается с помощью алгоритма двоичного поиска (деления отрезка пополам). ВП вычисляет положение **дробного индекса** (fractional index) с помощью линейной интерполяции. Чем ближе **начальный индекс** к истинному положению **значения x**, тем быстрее поиск

Polynomial Interpolation



Полиномиальная интерполяция



ВП интерполирует или экстраполирует функцию f, заданную набором n точек $(x[i],y[i])$, в точке x, заданной **значением x** (x value).

Y и **X** представляют массивы значений зависимой и независимой переменных. Если число элементов **Y** и **X** отличается, то ВП устанавливает на выходах **интерполированное значение** (interpolation value) и **ошибка интерполяции** (interpolation error) значение NaN и возвращает ошибку.

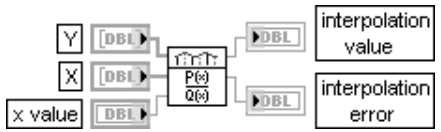
Вход **значение x** определяет точку, в которой производится интерполяция или экстраполяция. Если **значение x** лежит в диапазоне **X,** то ВП выполняет интерполяцию. В противном случае выполняется экстраполяция. Если **значение x** находится далеко за пределами диапазона **X,** то ошибка экстраполяции может быть достаточно велика. В этом случае экстраполяция будет неудовлетворительной.

Выход **интерполированное значение** (interpolation value) отображает интерполированное значение функции f при **значении x** (x value).

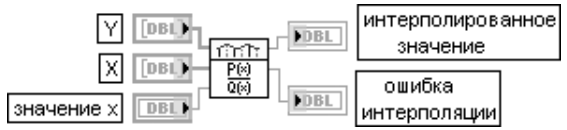
Выход **ошибка интерполяции** (interpolation error) представляет оценку ошибки интерполяции.

ВП рассчитывает **интерполированное значение** с помощью полинома $P[n-1](x)$, где $P[n-1]$ – полином степени $n-1$, проходящий через n точек $(x[i], y[i])$

Rational Interpolation

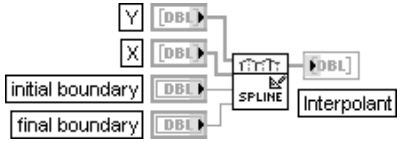


Рациональная интерполяция

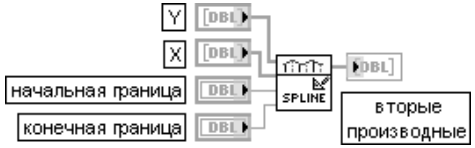


Перечень и назначение входов и выходов данного ВП идентичны описанному выше ВП **Полиномиальная интерполяция** (Polynomial Interpolation), за исключением того, что для получения **интерполированного значения** (interpolation value) применяется рациональная функция $\frac{P(x_i)}{Q(x_i)} = \frac{p_0 + p_1 x_i + \ldots + p_m x_i^m}{q_0 + q_1 x_i + \ldots + q_m x_i^m}$, которая проходит через все точки массивов значений **Y** и **X**

Spline Interpolant



Сплайн-интерполянт



ВП возвращает массив **Interpolant** длиной n, который содержит вторые производные сплайн-интерполирующей функции $g(x)$ в табулированных точках $x[i]$, где $i = \overline{0, n-1}$.
Входы **Y** и **X** представляют массивы значений зависимой и независимой переменных.
Вход **начальная граница** (initial boundary) определяет первую производную интерполирующей функции $g(x)$ в точке $x[0], g'(x[0])$. По умолчанию значение **начальной границы** равно 1,00E+30.
Вход **конечная граница** (final boundary) определяет первую производную интерполирующей функции $g(x)$ в точке $x[n-1], g'(x[n-1])$. По умолчанию значение **конечной границы** также равно 1,00E+30.
ВП рассчитывает интерполирующую функцию $g(x)$ путем интерполирования каждого интервала $[x_i, x_{i+1}]$ кубической полиномиальной функцией $p_i(x_i)$, которая удовлетворяет следующим условиям: 1) $p_i(x_i) = y_i$; 2) $p_i(x_{i+1}) = y_{i+1}$; 3) $g(x)$ имеет непрерывные первую и вторую производные в любой точке диапазона $[x_i, x_{i+1}]$: a) $p'_i(x_i) = p'_{i+1}(x_i)$; b) $p''_i(x_i) = p''_{i+1}(x_i)$.

В приведенных соотношениях $i = \overline{0, n-2}$.
Из последнего условия вытекают следующие уравнения:

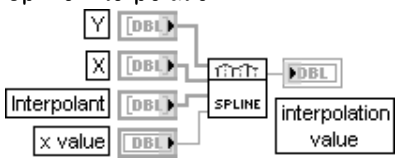
$$\frac{x_i - x_{i-1}}{6} g''(x_{i-1}) + \frac{x_{i+1} - x_{i-1}}{3} g''(x_i) + \frac{x_{i+1} - x_i}{6} g''(x_{i+1}) = \frac{y_{i+1} - y_i}{x_{i+1} - x_i} - \frac{y_i - y_{i-1}}{x_i - x_{i-1}}, \; i = \overline{0, n-2}.$$

Таким образом, имеет место система из $n-2$ линейных уравнений с n неизвестными $g''(x_i)$, $i = \overline{0, n-1}$.
ВП рассчитывает недостающие значения $g''(x_0)$, $g''(x_{n-1})$ используя **начальную границу** и **конечную границу** по формуле

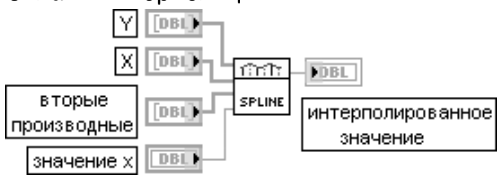
$$g'(x) = \frac{y_{i+1} - y_i}{x_{i+1} - x_i} + \frac{3A^2 - 1}{6}(x_{i+1} - x_i) g''(x_i) + \frac{3B^2 - 1}{6}(x_{i+1} - x_i) g''(x_{i+1}),$$

где $A = \frac{x_{i+1} - x}{x_{i+1} - x_i}$, $B = 1 - A = \frac{x - x_i}{x_{i+1} - x_i}$.
Решения системы уравнений – значения $g''(x_i)$ отображаются на выходе **Interpolant**. Эти значения могут быть использованы в ВП **Сплайн-интерполяция** (Spline Interpolation) для интерполяции y при всех значениях x в диапазоне $x_0 \le x \le x_{n-1}$

Spline Interpolation



Сплайн-интерполяция



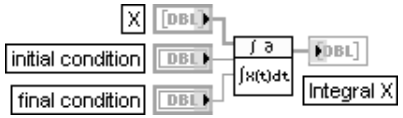
ВП возвращает **сплайн-интерполированное значение** (interpolation value) для входного **значения x** (x value).
При этом совокупность исходных значений задается табулированными значениями $(x[i], y[i])$ массивов независимой **X** и зависимой **Y** переменных, а также значениями **Interpolant,** получаемыми от ВП **Сплайн-интерполянт** (Spline Interpolant).
На интервале $[x_i, x_{i+1}]$ выход **интерполированное значение** определяется следующим выражением: $y = Ay_i + By_{i+1} + Cy''_i + Dy''_{i+1}$,
где $A = \frac{x_{i+1} - x}{x_{i+1} - x_i}$, $B = 1 - A$, $C = \frac{1}{6}(A^3 - A)(x_{i+1} - x_i)^2$, $D = \frac{1}{6}(B^3 - B)(x_{i+1} - x_i)^2$

3.4.2. Функции интегрирования и дифференцирования

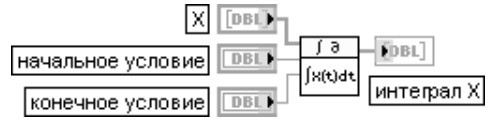
Функции интегрирования и дифференцирования позволяют выполнять численное интегрирование и дифференцирование входного сигнала.

Рис. 3.19. Вид подпалитры функций интегрирования и дифференцирования

Integral x(t)

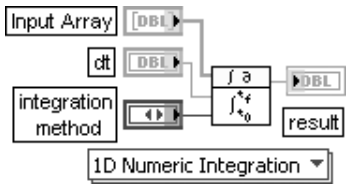


Интеграл x(t)

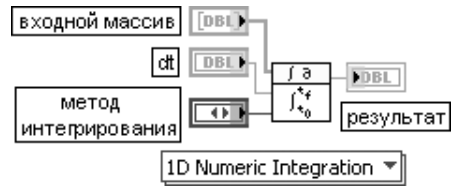


ВП выполняет дискретное интегрирование выборочного сигнала **X** в соответствии с формулой $y_i = \frac{1}{6}\sum_{j=0}^{i}(x_{j-1} + 4x_j + x_{j+1})dt$,
где $i = \overline{0, n-1}$, n – длина входной последовательности.
ВП считает определенный интеграл.
Поскольку x_{j-1} не определено при $j = 0$, а x_{j+1} – при $j = n - 1$, то для их задания служат входы **начальное условие** (initial condition) и **конечное условие** (final condition)**.** По умолчанию значения этих входов равны 0,0.
На вход dt подается величина, определяющая шаг интегрирования. По умолчанию dt =1,0

Numeric Integration



Численное интегрирование



Этот полиморфный ВП выполняет численное интегрирование массива входных данных с помощью одного из четырех наиболее распространенных числовых методов интегрирования. Селектор ВП позволяет выбирать его экземпляры для интегрирования одномерных, двумерных и трехмерных подынтегральных функций
Вход **входной массив** (Input Array) содержит интегрируемые данные, которые представляют выборки некоторой функции $f(t)$ с интервалом dt.
Вход dt представляет величину интервала выборки данных, подаваемых на вход **входной массив**.
При подаче отрицательного значения dt ВП использует его модуль. При выборе экземпляра ВП для интегрирования двумерных или трехмерных функций на этот вход подается кластер **размер интервала** (interval size), содержащий числовые элементы, которые определяют размер интервала по координатам при интегрировании этих функций.
Вход **метод интегрирования** (integration method) определяет метод, используемый для численного интегрирования.

0	1	2	3
Метод трапеций (Trapezoidal Rule)	**Метод Симпсона** (Simpsons' Rule)	**Метод Симпсона 3/8** (Simpsons' 3/8 Rule)	**Метод Боде** (Bode Rule)

Выход **результат** (result) отображает результат численного интегрирования.
Каждый из методов зависит от интервала выборки (***dt***) и вычисляет интеграл, используя последовательное применение основных методов в порядке, который зависит от числа соседних точек. Число точек, используемых для получения частичных оценок, зависит от порядка метода. Результат представляет сумму последовательных частичных оценок.

$$result = \int_{t0}^{t1} f(t)dt = \sum_{j} partial\ sum,$$

где ***j*** представляет диапазон, зависящий от числа отсчетов и метода интегрирования.
Ниже приведены основные формулы для вычисления частичных сумм по каждому методу в соответствии с возрастанием порядка метода:

- Метод трапеций: $(x[i] + x[i+1])dt/2$;
- Метод Симпсона: $(x[2i] + 4x[2i+1] + x[2i+2])dt/3$, $k=2$;
- Метод Симпсона 3/8: $(3x[3i] + 9x[3i+1] + 9x[3i+2] + 3x[3i+3])dt/8$, $k=3$;
- Метод Боде: $(14x[4i] + 64x[4i+1] + 24x[4i+2] + 64x[4i+3] + 14x[4i+4])dt/45$, $k=4$;
  для $i = 0,1,2,3\ldots$**Целая часть**$[(N-1)/k]$,

где N – число отсчетов данных, k – целое, зависящее от метода, и x – входной массив.
Если число отсчетов при выборе определенного метода не позволяет сформировать целое число частичных сумм, то этот метод применяется для точек, вошедших в частичные суммы. Для оставшихся точек применяется метод более низкого порядка. Пример применения методов различного порядка приведен в таблице:

Число отсчетов	Способ расчета частичных сумм
224	55 Боде, 1 Симпсон 3/8
225	56 Bode
226	56 Боде, Метод трапеций
227	56 Боде, 1 Симпсон
228	57 Боде, 1 Симпсон 3/8

Таким образом, если входной массив содержит 224 значения и выбран метод Боде, то ВП получает результат с помощью вычисления 55 частичных оценок по методу Боде и одной оценки по методу Симпсона 3/8.
Для выполнения двумерного численного интегрирования этот ВП сначала применяет одномерное численное интегрирование по ***x*** для расчета

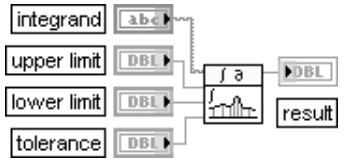
$$g(y) = \int_{x_0}^{x_1} f(x,y)dx.$$

Затем ВП применяет одномерное численное интегрирование по ***y*** для расчета результата.

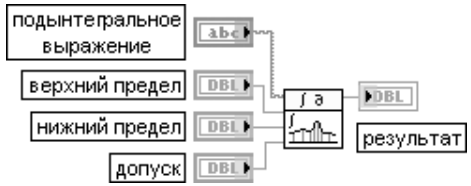
$$result = \int_{y_0}^{y_1} g(y)dy.$$

Подобным же образом производится трехмерное численное интегрирование

Lobatto Quadrature

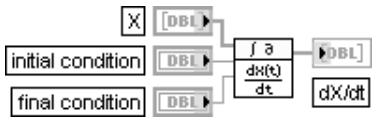


Квадратура Лобатто



ВП выполняет численное интегрирование, используя алгоритм адаптивной квадратуры Лобатто.
Вход **подынтегральное выражение** (integrand) определяет выражение, которое необходимо проинтегрировать. В качестве независимой переменной необходимо использовать **x**.
Для достижения высокой точности интегрирования при резком изменении подынтегрального выражения этот ВП делит интервал интегрирования на подынтервалы

Derivative x(t)



Производная x(t)

ВП выполняет численное дифференцирование входной последовательности **X** по формуле $y_i = \frac{1}{2dt}(x_{i+1} - x_{i-1})$, где $i = \overline{0, n-1}$, n – длина входной последовательности.
Значения, подаваемые на входы **начальное условие** (initial condition) и **конечное условие** (final condition), определяют величину x_{-1} и x_n соответственно. По умолчанию оба этих значения равны нулю.
На вход ***dt*** подается величина, определяющая шаг дифференцирования. По умолчанию ***dt***=1,0

В состав палитры функций численных методов входит Экспресс-ВП **Математическая обработка во временной области** (Time Domain Math), рассмотренный ниже.

Математическая обработка во временной области (Time Domain Math)

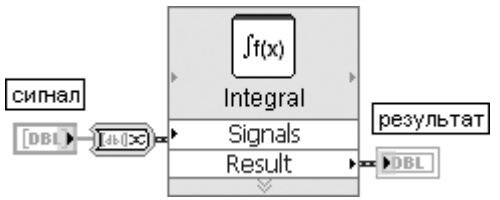


Рис. 3.20. Блок-диаграмма возможного подключения Экспресс-ВП

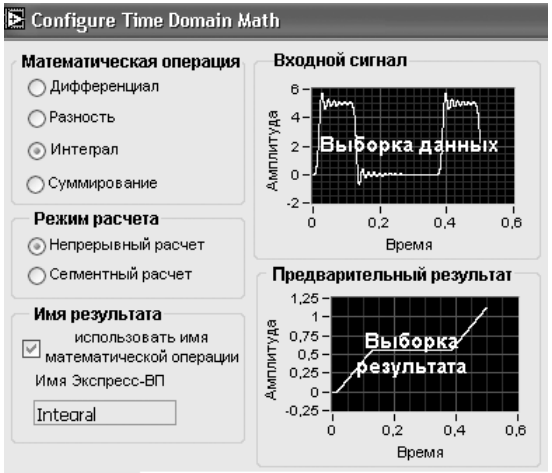


*Рис. 3.21. Вид диалогового окна конфигурирования Экспресс-ВП **Математическая обработка во временной области** (Time Domain Math)*

Экспресс-ВП **Математическая обработка во временной области** (Time Domain Math) выполняет одну из математических функций обработки сигнала во временной области. Данный Экспресс-ВП использует функциональность ВП **Производная x(t)** (Derivative x(t)) и **Интеграл x(t)** (Integral x(t))

3.4.3. Функции решения дифференциальных уравнений

Функции решения дифференциальных уравнений позволяют решать обыкновенные дифференциальные уравнения.

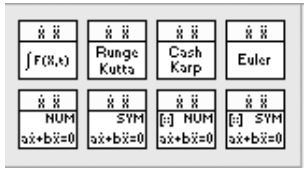
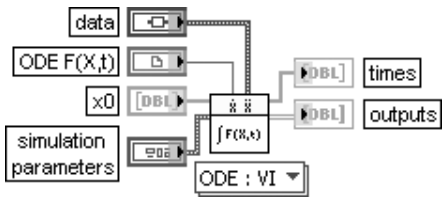
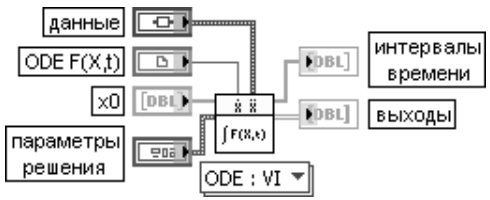


Рис. 3.22. Вид подпалитры функций решения дифференциальных уравнений

ODE Solver



Решатель ОДУ

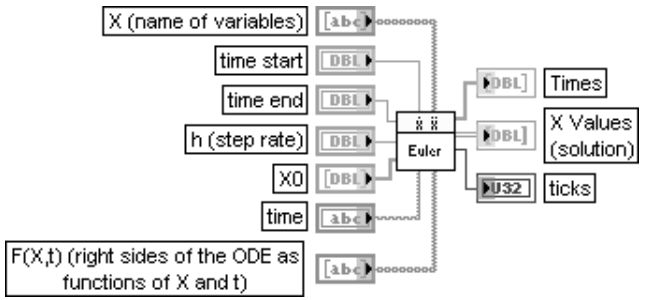


Этот полиморфный ВП находит решение обыкновенных дифференциальных уравнений (ОДУ) с начальными условиями, записанными в следующей форме: X'=F(X,t).
Селектор полиморфного ВП позволяет выбрать варианты реализации ВП с вводом правой части **F(X,t)** с помощью строкового кластера или с помощью специального ВП. На рисунке показано подключение ВП при выборе второго варианта.
Вход **данные** (data) относится к типу Вариант и может использоваться для передачи произвольных данных к ВП **ODE F(X,t)**.
Вход **ODE F(X,t)** передает ссылку строгого типа к ВП, который обеспечивает выполнение правой части ОДУ **dX/dt=F(X,t)**. Этот ВП создается из шаблона, расположенного по адресу labview\vi.lib\gmath\ode.llb\ODE rhs.vit.
При вводе правой части ОДУ с помощью строкового кластера **ODE RHS** последний содержит следующие элементы:

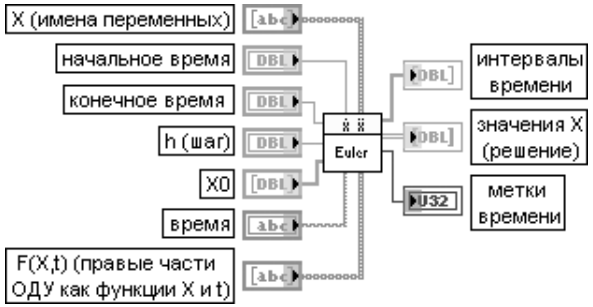
- **F(X,t)** – одномерный массив строк, представляющих правые части дифференциальных уравнений;
- **X** – массив строк переменных;
- **время** (time) – строка, обозначающая временную переменную. По умолчанию это **t**.

Вход **x0** передает вектор начальных условий ОДУ.
Вход **параметры решения** (simulation parameters) определяет набор параметров, используемых для конфигурирования числового решения дифференциального уравнения.
Выход **интервалы времени** (times) возвращает массив отметок времени, по которым ВП производил расчет.
Выходы (outputs) являются двумерным массивом значений. Каждая строка соответствует вектору значений **y**, рассчитанному в определенное время. Каждый столбец представляет временное изменение фиксированного значения **y**

ODE Euler Method



Решение ОДУ методом Эйлера



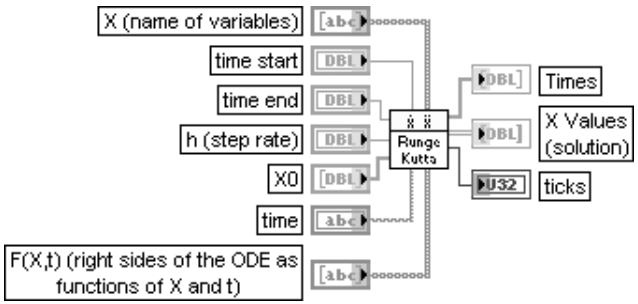
ВП находит решение обыкновенных дифференциальных уравнений (ОДУ) с начальными условиями, используя метод Эйлера.

Вход **X** представляет массив строк с именами переменных.
Вход **начальное время** (time start) задает начальную точку решения ОДУ. По умолчанию значение входа равно 0.
Вход **конечное время** (time end) определяет конечную точку временного интервала решения. По умолчанию значение входа равно 1,0.
Вход **h** задает величину фиксированного шага. По умолчанию его значение равно 0,1.
Вход **X0** определяет вектор начальных условий **x[10], ..., x[n0]**. Между компонентами **X0** и **X** существует однозначное соответствие.
Вход **время** (time) представляет строку, задающую переменную времени. По умолчанию в качестве переменной используется **t**.
Вход **F(X,t)** задает одномерный массив строк, представляющих правые части дифференциальных уравнений.
Выход **интервалы времени** (Times) отображает массив, представляющий интервалы времени. Метод Эйлера использует одинаковые временные интервалы между **начальным временем** и **конечным временем**.
Выход **значения X** (X Values) отображает двумерный массив вектора решений **x[10], ..., x[n].** Верхний индекс соответствует интервалам времени, определенным в массиве **интервалы времени**, нижний индекс – элементам **x[10], ..., x[n].**
Стандартная форма записи системы линейных дифференциальных уравнений первого порядка включает запись самой системы и ее начальных условий:

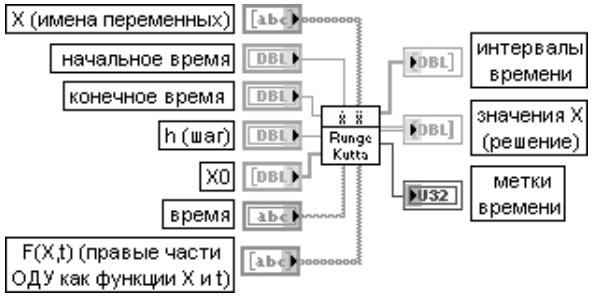
$$\begin{cases} \dot{x}_1(t) = f_1(x_1(t),\ldots x_n(t),t) \\ \quad\cdot \\ \quad\cdot \\ \dot{x}_n(t) = f_n(x_1(t),\ldots x_n(t),t) \end{cases} \qquad \begin{cases} x_1(t_0) = x_{10} \\ \quad\cdot \\ \quad\cdot \\ x_n(t_0) = x_{n0} \end{cases}$$

Предполагается, что функции $f_1,\ldots f_n$, начальные условия $x_{10},\ldots x_{n0}$ и начальный момент времени заданы. Необходимо определить вид зависимостей $x_1(t),\ldots x_n(t)$.
Используя обозначения $F = (f_1,\ldots f_n)$, $X = (x_1(t),\ldots x_n(t))$ и $X_0 = (x_{10},\ldots x_{n0})$, систему уравнений можно записать в векторной форме $\begin{cases} \dot{X}(t) = F(X(t),t) \\ X(t_0) = X_0 \end{cases}$.
Метод Эйлера является наиболее простым методом решения обычных дифференциальных уравнений (ОДУ). Его суть заключается в использовании при пошаговом интегрировании зависимой переменной двух первых членов ряда Тейлора. Таким образом, итерационная процедура вычисления значений $X(t_i)$ при начальном моменте времени t_0 и достаточно малом шаге интегрирования h может быть записана в следующем виде:
$X(t_0 + h) = X(t_0) + hF(X(t_0),t_0)$,
$X(t_0 + 2h) = X(t_0 + h) + hF(X(t_0 + h),t_0 + h)$.
Этот процесс останавливается, когда ***начальное время*** + *nh* < ***конечное время***, где ***конечное время*** представляет правую конечную точку исследуемого процесса

ODE Runge Kutta 4th Order



Решение ОДУ методом Рунге-Кутты 4-го порядка



ВП решает обыкновенные дифференциальные уравнения (ОДУ) с начальными условиями, используя метод Рунге-Кутты. Метод Рунге-Кутты работает с фиксированным шагом, обеспечивая вместе с тем более высокую точность, чем метод Эйлера. Функции одноименных входов и выходов идентичны рассмотренному выше ВП **Решение ОДУ методом Эйлера** (ODE Euler Method).
Метод Рунге-Кутты имеет 4-ый порядок и итерационные расчетные выражения выглядят следующим образом:

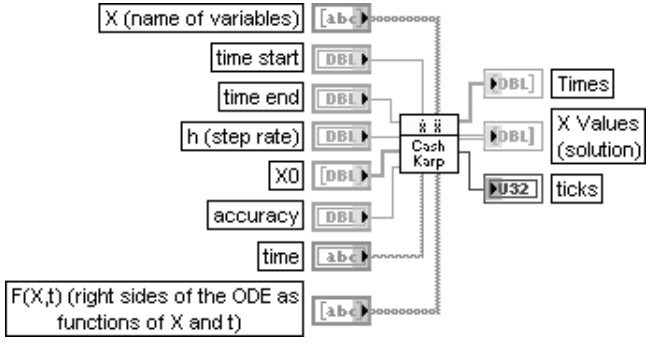
$$K_1 = hF(X(t_n),t_n); \qquad K_2 = hF\left(X(t_n) + \frac{K_1}{2}, t_n + \frac{h}{2}\right);$$

$$K_3 = hF\left(X(t_n) + \frac{K_2}{2}, t_n + \frac{h}{2}\right); \qquad K_4 = hF(X(t_n) + K_3, t_n + h);$$

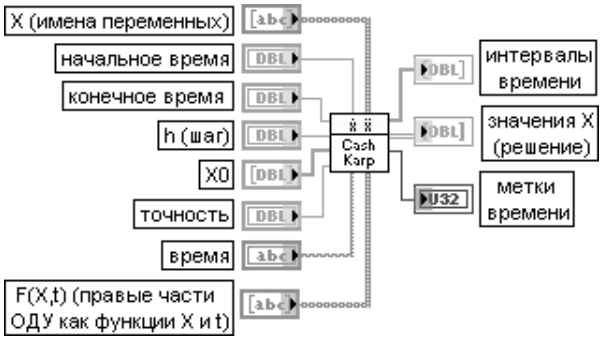
$$X(t_{n+1}) = X(t_n) + \frac{K_1}{6} + \frac{K_2}{3} + \frac{K_3}{3} + \frac{K_4}{6}.$$

Этот процесс останавливается при достижении условия ***начальное время*** + *nh* < ***конечное время***, где ***конечное время*** представляет правую конечную точку исследуемого процесса

ODE Cash Karp 5th Order



Решение ОДУ методом Кэш-Капа 5-ого порядка



ВП решает ОДУ с начальными условиями с помощью метода Кэш-Капа. Метод Кэш-Капа работает с адаптивным шагом и в вычислительном отношении более эффективен по сравнению с методами Эйлера и Рунге-Кутты.

Итерационные расчетные выражения для метода выглядят следующим образом:

$k_1 = hF(X(t_n), t_n)$;

$k_2 = hF(X(t_n) + a_2 h, t_n + b_{21}k_1)$;

$k_6 = hF(X(t_n) + a_5 h, t_n + b_{61} + \ldots b_{65}k_5)$;

и

$X(t_{n+1}) = X(t_n) + c_1k_1 + \ldots + c_6k_6$;

$\dot{X}(t_{n+1}) = X(t_n) + c_1^*k_1 + \ldots + c_6^*k_6$

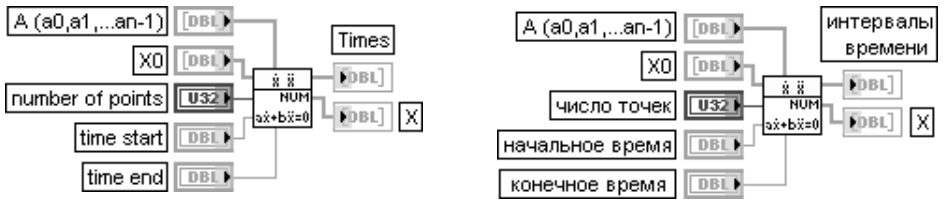
с $t_{n+1} = t_n + h$.

Действительный размер шага h_{new} определяется с помощью выражения $h_{new} = h\left|\frac{точность}{\Delta}\right|^{\frac{1}{5}}$,

где h – предыдущий размер шага, $\Delta = \left|X(t_{n+1}) - X^*(t_{n+1})\right|$

ODE Linear nth Order Numeric

Решение линейного ОДУ n-го порядка в численном виде



ВП решает однородное линейное дифференциальное уравнение n-го порядка в численном виде.

Вход **A** представляет вектор коэффициентов различных производных функции $x(t)$, начиная с коэффициента производной самого низкого порядка. Коэффициент при производной самой высокой степени предполагается равным 1 и не требует ввода.

Вход **X0** представляет вектор начальных условий $x[10], \ldots, x[n0]$. Между компонентами **X0** и **X** существует однозначное соответствие.

Вход **число точек** (number of points) задает число равноудаленных по времени точек между **начальным временем** и **конечным временем**. По умолчанию число точек равно 10.

Входы **начальное время** и **конечное время** определяют начальную и конечную точки интервала, на котором ищется решение ОДУ. По умолчанию значения точек равны соответственно 0,0 и 1,0.

Выход **интервалы времени** (Times) содержит массив значений интервалов времени.

Выход **X** содержит вектор решений **x** в равномерно расположенных на оси времени точках, заданных в массиве **интервалы времени**.

Решение линейного однородного дифференциального уравнения n-го порядка

$x^{(n)} + a_{n-1}x^{(n-1)} + \ldots + a_1x^{(1)} + a_0x = 0$

с начальными условиями $x(0) = x_{00}$; $x^{(1)}(0) = x_{10}$; ..., $x^{(n-1)}(0) = x_{n-10}$

ищется с помощью разложения выражения на простые сомножители и определения его корней.

При наличии n различных комплексных корней $\lambda_1,\ldots,\lambda_n$ общее решение дифференциального уравнения n-го порядка может быть записано в следующем виде:

$x(t) = \beta_1 \exp(\lambda_1 t) + \ldots + \beta_n \exp(\lambda_n t)$.

Неизвестные коэффициенты $\beta_1,\ldots\beta_n$ могут быть определены с помощью начальных условий

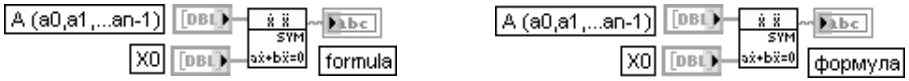
$x(0) = \beta_1 + \ldots + \beta_n$; $x^{(1)}(0) = \beta_1\lambda_1 + \ldots + \beta_n\lambda_n$; ...; $x^{(n-1)}(0) = \beta_1\lambda_1^{n-1} + \ldots + \beta n\lambda_n^{n-1}$,

где 0 соответствует **начальному времени** (time start).

Так, например, для решения дифференциального уравнения $x'' - 3x' + 2x = 0$ с начальными условиями $x(0) = 2$ и $x'(0) = 3$ необходимо ввести $A = [2, -3]$ и $X0 = [2, 3]$

ODE Linear nth Order Symbolic

Решение линейного ОДУ n-го порядка в символьном виде



ВП решает однородное линейное дифференциальное уравнение n-го порядка с постоянными коэффициентами в символьном виде.

Входы **A** и **X0** идентичны одноименным входам рассмотренного выше ВП **Решение линейного ОДУ n-го порядка в численном виде** (ODE Linear nth Order Numeric).

Выход **формула** (formula) содержит решение в символьном виде.

Общее решение дифференциального уравнения имеет следующий вид:

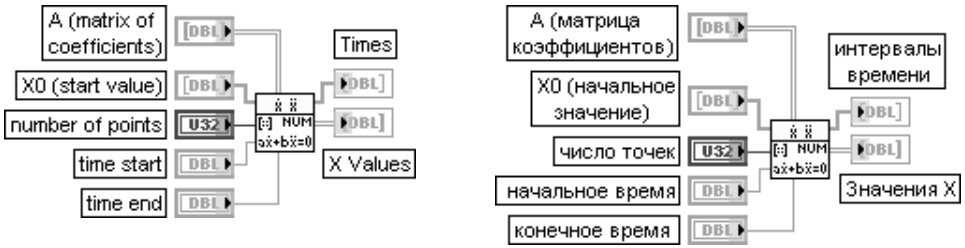
$x(t) = \beta_1 \exp(\lambda_1 t) + \ldots + \beta_n \exp(\lambda_n t)$

с комплексными $\beta_1,\ldots,\beta_n$ и $\lambda_1,\ldots,\lambda_n$.

Однако все входы ВП имеют реальные значения, соответственно и решения являются также реальными. Решение в символьном виде представляет линейную комбинацию экспоненциальных, синусоидальных и косинусоидальных функций с действительными числовыми коэффициентами

ODE Linear System Numeric

Решение системы линейных ОДУ в численном виде



ВП решает систему линейных дифференциальных уравнений n-го порядка с заданными начальными условиями. Решение основано на определении собственных значений и собственных векторов базовой матрицы. Решение дается в численном виде.

Вход **A** задает матрицу размером $n \times n$, описывающую линейную систему.

Вход **X0** определяет вектор из n элементов, описывающий начальные условия, $x[10],\ldots x[n0]$.

Между компонентами **X0** и **X** существует однозначное соответствие.

Входы **число точек**, **начальное время** и **конечное время** идентичны одноименным входам рассмотренного выше ВП **Решение линейного ОДУ n-го порядка в численном виде** (ODE Linear nth Order Numeric). Аналогичное соответствие существует и между выходами **интервалы времени** и **значения X** этих ВП.

Данный ВП позволяет получить правильные решения практически для всех случаев действительной матрицы **A**, которая может иметь повторяющиеся собственные значения, комплексные значения и т. п. Исключением является случай с сингулярными собственными векторами. Работа с такой матрицей завершается ошибкой -23016.

Линейная система дифференциальных уравнений может быть записана в векторной форме:

$$\frac{dX(t)}{dt} = AX(t);\ X(0) = X_0.$$

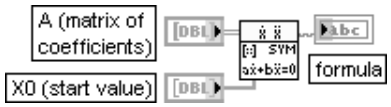
Здесь $X(t) = (x_0(t),...x_n(t))$ и A представляет реальную матрицу размером $n \times n$.

Линейная система может быть решена с помощью определения собственных значений и собственных векторов. Пусть S представляет набор всех собственных векторов, распределенных во всем n-мерном пространстве. Тогда преобразование $Y=SX(t)$ приводит запись системы к следующему виду:

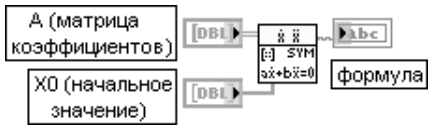
$$\frac{dY(t)}{dt} = SAS^{-1}Y(t);\ Y(0) = SX_0.$$

Матрица SAS^{-1} имеет диагональную форму, поэтому решение очевидно. Решение может быть получено путем обратного преобразования $X(t) = S^{-1}Y(t)$

ODE Linear System Symbolic



Решение системы линейных ОДУ в символьном виде



ВП решает систему линейных дифференциальных уравнений n-го порядка с заданными начальными условиями. Решение основано на определении собственных значений и собственных векторов базовой матрицы. Решение дается в символьном виде.

Вход **A** задает матрицу размером $n \times n$, описывающую линейную систему.

Вход **X0** определяет вектор из n элементов, описывающий начальные условия, $x[10],...x[n0]$.

Между компонентами **X0** и **X** существует однозначное соответствие.

Выход **формула** (formula) представляет строку с решением системы линейных уравнений в стандартной формульной записи LabVIEW. Элементы вектора решения разделены символом «возврат каретки».

Линейное дифференциальное уравнение, описанное следующей системой:

$$\begin{bmatrix} \dot{x}_1(t) \\ \dot{x}_2(t) \\ \dot{x}_3(t) \\ \dot{x}_4(t) \end{bmatrix} = \begin{bmatrix} -7 & -6 & 4 & -1 \\ -6 & 2 & 1 & -2 \\ 4 & 1 & 0 & 2 \\ -1 & -2 & 2 & -7 \end{bmatrix} \begin{bmatrix} x_1(t) \\ x_2(t) \\ x_3(t) \\ x_4(t) \end{bmatrix}$$ с начальными условиями $$\begin{cases} x_1(0) = 1 \\ x_2(0) = 2 \\ x_3(0) = 3 \\ x_4(0) = 4 \end{cases}$$

имеет решение

(+1,62*exp(-12,46*t)-1,28*exp(-6,30*t)+0,63*exp(1,34*t)+0,04*exp(5,42*t))+(0);
(+0,84*exp(-12,46*t)-0,29*exp(-6,30*t)+1,51*exp(1,34*t)-0,06*exp(5,42*t))+(0);
(-0,73*exp(-12,46*t)+0,01*exp(-6,30*t)+3,69*exp(1,34*t)+0,02*exp(5,42*t))+(0);
(+0,87*exp(-12,46*t)+2,67*exp(-6,30*t)+0,45*exp(1,34*t)+0,01*exp(5,42*t))+(0).

Для получения решения необходимо ввести следующие значения:

A : [−7,−6,4,−1;−6,2,1,−2;4,1,0,2;−1,−2,2,−7], $X0$[1,2,3,4]

3.4.4. Функции оптимизации

Функции оптимизации (рис. 3.23) позволяют определить точки минимума одномерных и многомерных функций. Выражение для исследуемой одномерной функции задается на входе **формула** (formula) строкового типа. Для многомерной функции аналогичное выражение задается на входе $f(X)$, при этом на входе **X** должен быть указан массив переменных этой функции.

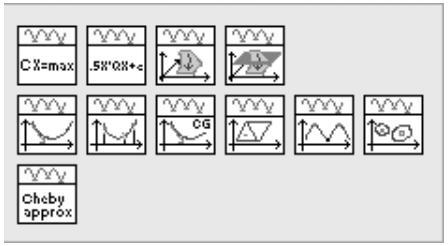
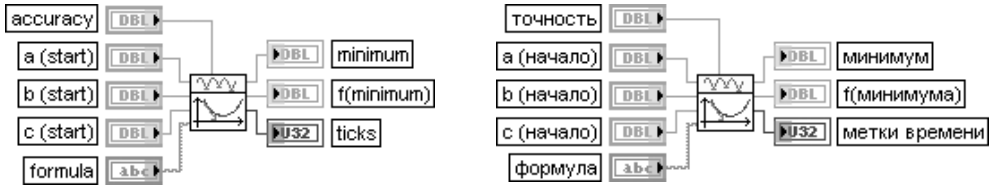


Рис. 3.23 Вид палитры функций оптимизации

Brent with Derivatives 1D

Поиск минимума одномерной функции с помощью производных



ВП определяет локальный минимум заданной одномерной функции на заданном интервале. Метод основан на расчете производных функции, заданной на входе **формула** (formula).

Вход **точность** (accuracy) устанавливает точность определения минимума выражения, заданного на входе **формула**. Метод завершает работу, если результаты двух последовательных итераций отличаются на величину, меньшую, чем **точность**.

Входы **a**, **b** и **c** представляют соответственно левую, среднюю и правую точки отрезка локализации минимума. По умолчанию значения этих входов равны 0,0.

Вход **формула** (formula) представляет строку, описывающую исследуемую функцию.

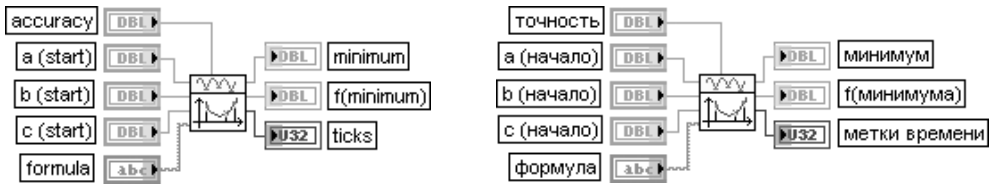
Выход **минимум** (minimum) представляет найденный локальный минимум функции, заданной ф**ормулой**.

Выход **f(минимума)** (f(minimum)) возвращает значение функции в определенном локальном минимуме.

Выход **метки времени** (ticks) определяет время выполнения расчета в миллисекундах

Golden Section 1D

Метод золотого сечения

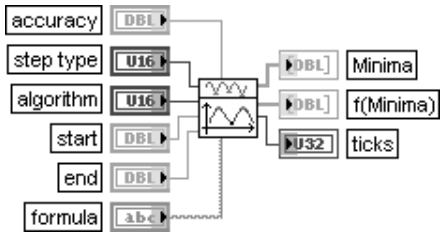


ВП определяет локальный минимум заданной одномерной функции с помощью метода золотого сечения.

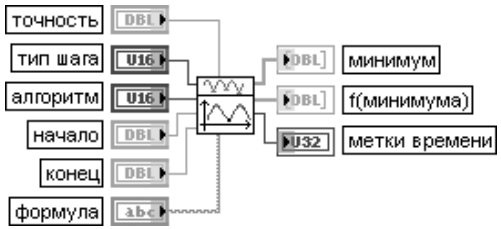
Все входы и выходы этого ВП имеют то же назначение, что и одноименные входы и выходы рассмотренного выше ВП **Поиск минимума одномерной функции с помощью производных** (Brent with Derivatives 1D).
Ограничивающий триплет (***a***, ***b***, ***c***) одномерной функции f представляет комбинацию трех точек, таких, что $f(a) > f(b)$ и $f(c) > f(b)$. Это гарантирует существование локального минимума функции f на интервале (***a, c***).
Начав с ограничивающего триплета (***a, b, c***), метод золотого сечения определяет новый ограничивающий триплет, охватывающий значительно меньший интервал. Повторение этой схемы ведет в большинстве случаев к быстрому нахождению минимума. Новая граничная точка ***x*** рассчитывается с помощью следующего выражения:

$$\left|\frac{x-b}{c-a}\right| = \sqrt{5} - 2$$

Find All Minima 1D

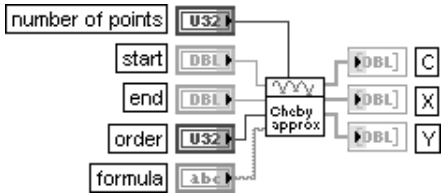


Найти все минимумы одномерной функции

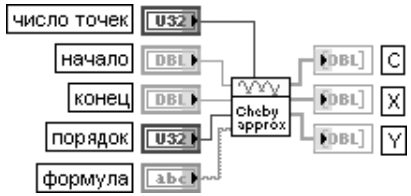


ВП определяет все локальные минимумы заданной функции на заданном интервале.
Входы **точность**, **формула** и выход **метки времени** имеют то же назначение, что и одноименные входы и выходы рассмотренного выше ВП **Поиск минимума одномерной функции с помощью производных** (Brent with Derivatives 1D).
Вход **тип шага** (step type) определяет вид расположения значений функции. При выборе варианта с **фиксированным шагом** (fixed function) (значение 0) функция имеет фиксированные равномерно расположенные значения. При выборе варианта **модифицированной функции** (modified function) (значение 1) значения модифицированной функции берутся с оптимальным шагом. В общем случае использование модифицированной функции приводит к более точному определению минимумов. По умолчанию на входе установлен первый вариант.
Вход **алгоритм** (algorithm) определяет метод поиска минимума, используемый ВП. Возможные варианты включают выбор **метода золотого сечения** (Golden Section Search) (по умолчанию) и **метода поиска с помощью производных** (Brent with Derivatives).
Входы **начало** (start) и **конец** (end) определяют начальную и конечную точки интервала. По умолчанию их значения равны соответственно 0,0 и 1,0.
Выход **минимумы** (Minima) представляет массив всех найденных минимумов выражения, заданного **формулой** на интервале (**начало**, **конец**).
Выход **f**(**минимума**) (f(Minima)) представляет значения функции в точках минимумов (Minima).
При необходимости определения максимумов функции на входе ВП необходимо перед выражением на входе **формула** дописать «–». В этом случае значения –f(Minima) будут корректно характеризовать значения максимумов функции

Chebyshev Approximation

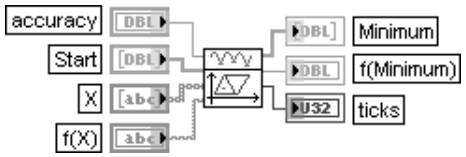


Аппроксимация полиномами Чебышева

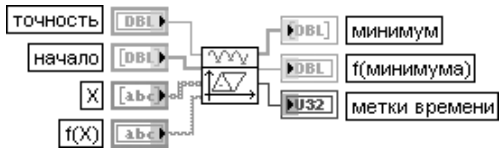


ВП аппроксимирует заданную функцию с помощью полиномов Чебышева.
Вход **число точек** (number of points) задает число равномерно расположенных точек на интервале (**начало**, **конец**). По умолчанию значение на этом входе равно 10.
Вход **начало** (start) определяет начальную точку интервала. По умолчанию значение равно 0,0.
Вход **конец** (end) определяет конечную точку интервала. По умолчанию значение равно 1,0.
Вход **порядок** (order) определяет степень аппроксимирующих полиномов Чебышева. По умолчанию степень равна 3. Степень определяет число различных полиномов Чебышева $T_0(x), T_1(x), \ldots T_m(x)$, используемых для аппроксимации выражения, заданного на входе **формула**.
Вход **формула** (formula) представляет строку, описывающую исследуемую функцию $f(x)$.
Выход **C** отображает массив коэффициентов, входящих в аппроксимирующее выражение $f(x) = c_0 T_0(x) + \ldots + c_m T_m(x)$.
Выход **X** представляет значения **x**, разделяющие интервал (**начало**, **конец**) на равновеликие подынтервалы.
Выход **Y** представляет значения **y** полинома Чебышева в точках **X** .
Коэффициенты аппроксимирующего полинома $c_0, \ldots, c_m$ могут быть рассчитаны с помощью следующего выражения: $c_j = \sum_{k=1}^{n} f(x_k) T_j(x_k)$, где $x_k = \cos\left(\frac{\pi k}{n}\right)$, $k = \overline{1,n}$

Downhill Simplex nD



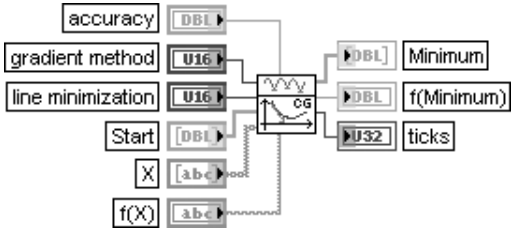
Симплекс-метод скорейшего спуска поиска минимума функции n переменных



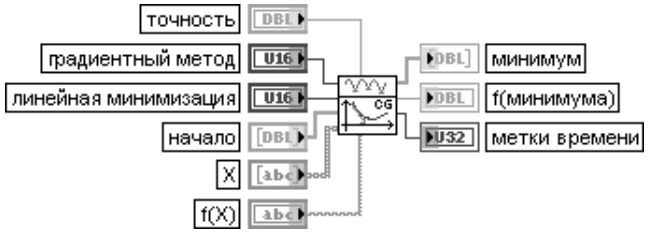
ВП ищет локальный минимум функции **n** независимых переменных с помощью симплекс-метода скорейшего спуска.
Вход **точность** (accuracy) задает точность определения минимума. Метод останавливается, если результаты двух последовательных итераций отличаются на величину меньшую, чем **точность**. По умолчанию значение входа равно 1,0E-8.
Вход **начало** (Start) представляет массив точек, которые определяют начало процесса оптимизации. Данные точки образуют симплекс в **n**-мерном пространстве.
Вход **X** задает массив строк, представляющих переменные **x**.
Вход **f(X)** задает строку, представляющую функцию переменных **x**.
Выход **минимум** (Minimum) отображает массив координат точки локального минимума в **n**-мерном пространстве.
Выход **f(минимума)** (f(Minimum)) отображает значение функции **f(X)** в найденном минимуме.
Выход **метки времени** (ticks) определяет время (в миллисекундах) выполнения расчета.
Симплекс-метод скорейшего спуска, называемый также методом Нелдера-Мида, работает без

использования частных производных. Данный метод осуществляет поиск минимума функции **f(X)** с помощью простых геометрических фигур, определяемых как симплекс.
Симплекс на плоскости представляет треугольник, симплекс в трехмерном пространстве – тетраэдр и т. д. Для задания начального симплекса необходимо иметь $(n + 1)$ начальную точку, каждая из которых имеет n координат. Необходимо ввести только одну точку из $(n + 1)$ начальных точек. Симплекс размером $(n + 1)$ будет автоматически построен.
В процессе поиска минимума новый симплекс строится на базе текущего с помощью набора таких операций, как отражение, растяжение или сжатие. В конце поиска минимум определяется с помощью симплекса очень малого размера

Conjugate Gradient nD

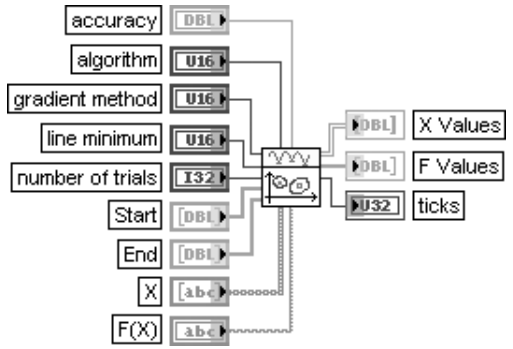


Метод сопряженных градиентов поиска минимума функции n переменных

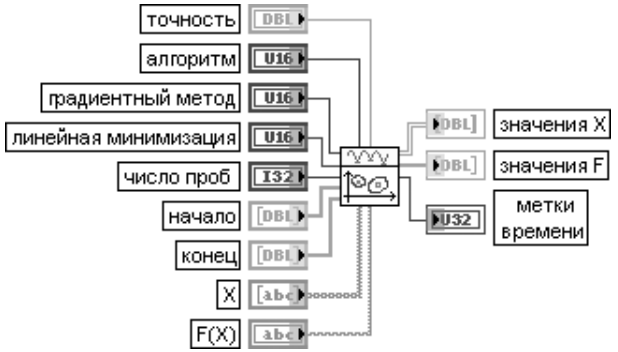


ВП определяет минимум функции **n** независимых переменных с помощью метода сопряженных градиентов.
Входы **точность**, **начало**, **X**, **f(X)** и выходы **минимум**, **f(минимума)**, **метки времени** имеют то же назначение, что и одноименные входы и выходы рассмотренного выше ВП **Симплекс-метод скорейшего спуска поиска минимума функции n переменных** (Downhill Simplex nD).
Вход **градиентный метод** (gradient method) определяет алгоритм, используемый для расчета производных. Первый вариант (значение 0) определяет выбор **метода Флетчера-Ривза** (Fletcher-Reeves). Второй вариант (значение 1) определяет **метод Полака-Рибера** (Polak-Ribiere).
По умолчанию значение входа равно 0.
Вход **линейная минимизация** (line minimization) также предоставляет два варианта выбора метода минимизации: **без использования производных** (по умолчанию) (without derivatives) и **с использованием производных** (with derivatives)

Find All Minima nD

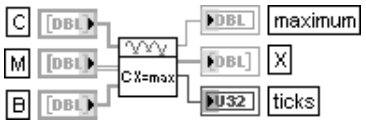


Найти все минимумы функции n переменных



ВП ищет минимум **n**-мерной функции в заданном **n**-мерном интервале.
Назначение входов **точность**, **градиентный метод**, **линейная минимизация**, **начало**, **X**, **f(X)** и выхода **метки времени** идентично назначению одноименных входов и выхода рассмотренных выше ВП оптимизации.
Вход **алгоритм** (algorithm) определяет метод, используемый в ВП. Возможные варианты включают выбор **метода сопряженных градиентов** (Conjugate Gradient) (по умолчанию) и **симплекс-метода наискорейшего спуска** (Downhill Simplex).
Вход **число проб** (number of trials) задает число случайно выбранных начальных точек процесса оптимизации. Эти точки принадлежат интервалу (**начало**, **конец**). По умолчанию число равно 5.
Входы **начало** (Start) и **конец** (End) определяют координаты начальной и конечной точек в **n**-мерном пространстве.
Выход **значения X** (X Values) представляет матрицу, описывающую все найденные локальные минимумы.
Выход **значения F** (F Values) представляет значения функции в точках минимумов **значения X**

Linear Programming Simplex Method

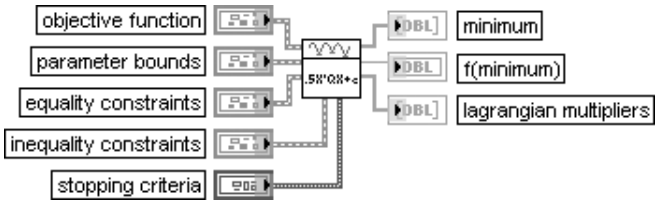


Симплекс-метод линейного программирования

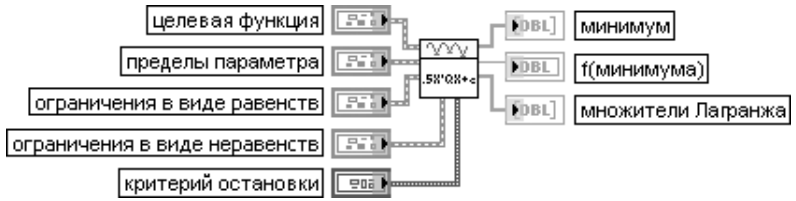
ВП определяет решение проблемы линейного программирования.
Вход **C** представляет вектор, описывающий максимизируемый линейный функционал.
Вход **M** задает матрицу, описывающую различные ограничения.
Вход **B** представляет вектор, описывающий правые части ограничений типа неравенства.
Выход **максимум** (maximum) отображает максимальное значение **x**, если оно существует при заданных ограничениях.
Выход **X** представляет вектор решения.
Проблема оптимизации, которая решается данным ВП, определяется следующими уравнениями:
$\boldsymbol{cx = max}$
с ограничениями $\boldsymbol{x \geq 0}$ и $\boldsymbol{mx \geq b}$.
Для проблемы оптимизации $\boldsymbol{cx = max}$ необходимо использовать следующие определения:
$\mathbf{X} = (x_1, ..., x_n)$; $\mathbf{C} = (c_1, ..., c_n)$; $\mathbf{B} = (b_1, ..., b_k)$; $\mathbf{M}$ – матрица $\boldsymbol{k \times n}$.
Для решения проблемы оптимизации необходимо решить, существует ли оптимальный вектор **X**. Если он существует, то его можно определить.
Решение проблемы линейного программирования состоит из двух этапов:
1. Исходную проблему необходимо преобразовать в проблему с ограничениями в нормальной форме, по существу без неравенств в формулировках.
2. Решить проблему с ограничениями в нормальной форме.

Формулировка с ограничениями в нормальной форме представляется специфичной. Однако существует ряд способов изменения условий. Так, в частности, условие $\boldsymbol{dx \leq e}$ эквивалентно условию $-\boldsymbol{dx} \geq -\boldsymbol{e}$, а $\boldsymbol{dx = e}$ эквивалентно комбинации $\boldsymbol{dx \geq e}$ и $-\boldsymbol{dx} \geq -\boldsymbol{e}$

Quadratic Programming



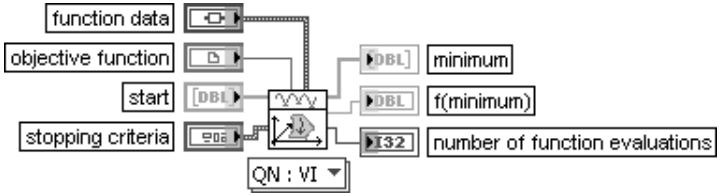
Квадратичное программирование



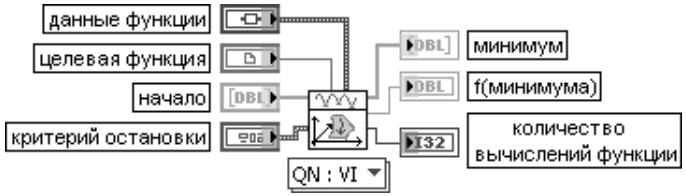
ВП решает проблему минимизации выражения 0,5x*Q*x + c*x так, что A*x=b и $l_{min} \leq D*x \leq l_{max}$.
Вход **целевая функция** (objective function) представляет ссылку к ВП, который содержит оптимизируемую функцию. В состав кластера входят квадратичный член **Q** и линейный член **c**.
Вход **пределы параметра** (parameter bounds) содержит минимальные **xmin** и максимальные **xmax** значения, которые могут принять параметры (x).
Вход **ограничения в виде равенства** (equality constraints) определяет ограничения в виде равенства, задаваемые линейным матричным уравнением Ax=b. Кластер состоит из матричного члена **A** и векторного члена **b**.
Вход **ограничения в виде неравенства** (inequality constraints) содержит пределы неравенства для линейной матрицы l_{min}<Dx<l_{max}. В состав кластера входят матричный член **D** и векторные члены l_{min} и l_{max}.

Вход **критерий остановки** (stopping criteria) содержит набор условий, которые завершают оптимизацию. Оптимизация завершается, если при очередной итерации относительное изменение любой из следующих величин: функции, параметра или градиента становится меньше допустимого или же превышены **максимальное число итераций** (max iterations) или **максимальное число вызовов функции** (max function calls).
Выход **минимум** (minimum) представляет набор значений, которые минимизируют квадратичную целевую функцию, удовлетворяющую пределам и ограничениям.
Выход **f(минимума)** f(minimum) возвращает значение квадратичной целевой функции 0,5x^TQx + cx в **минимуме**.
На выход **множители Лагранжа** (lagrangian multipliers) поступают коэффициенты лагранжиана, которые соответствуют ограничениям в виде равенства или в виде неравенства. Если имеется три ограничения в виде равенства и два ограничения в виде неравенства, первые три множителя Лагранжа соответствуют ограничениям в виде равенства, а последние два множителя Лагранжа – ограничениям в виде неравенства

Unconstrained Optimization

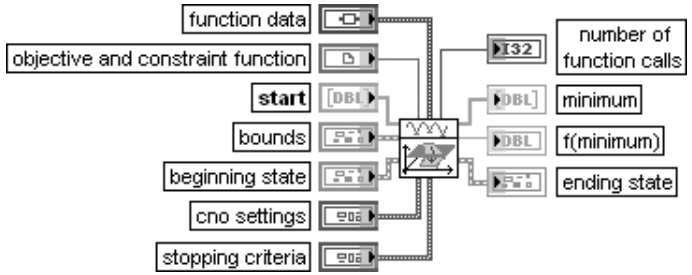


Оптимизация без ограничений

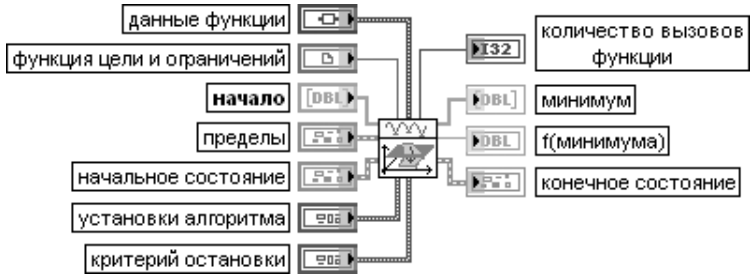


Этот полиморфный ВП решает проблему оптимизации без ограничений для произвольной нелинейной функции.
Селектор ВП позволяет выбрать определенный экземпляр из предлагаемого класса.
Для плавных функций, имеющих определенную первую и вторую производную, наиболее быструю сходимость обеспечивает квазиньютоновский алгоритм. При возникновении проблем со сходимостью квазиньютоновского алгоритма их может решить алгоритм сопряженных градиентов (Conjugate Gradient). Симплекс-метод скорейшего спуска (Downhill Simplex) опирается только на расчеты функции и часто способен найти решение для негладкой функции и другие алгоритмы не сходятся.
Вход **данные функции** (function data) содержит статические данные, которые требуются функции, определяемой пользователем, во время исполнения.
Вход **целевая функция** (objective function) представляет ссылку на ВП строгого типа, который содержит оптимизируемую функцию. Такой ВП создается из шаблона, находящегося по следующему пути: labview\vi.lib\gmath\NumericalOptimization\cno_objective function template.vit.
Вход **начало** (start) определяет точку в ***n***-мерном пространстве, с которой начинается процесс оптимизации.
Выход **количество вычислений функции** (number of function evaluations) возвращает число вызовов целевой функции в процессе оптимизации

Constrained Nonlinear Optimization



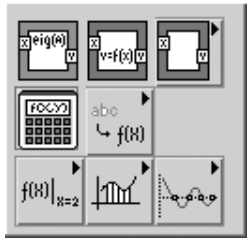
Нелинейная оптимизация с ограничениями



ВП решает общую проблему нелинейной оптимизации с нелинейными ограничениями типа равенства и нелинейными ограничениями типа неравенства, используя последовательный метод квадратичного программирования.
Вход **пределы** (bounds) является кластером, который содержит верхний и нижний пределы для оптимизируемых параметров и ограничений типа неравенства.
Вход **начальное состояние** (beginning state) содержит начальные значения ограничений функции типа неравенства, множителей Лагранжа и гессиана. **Начальное состояние** является обычно **конечным состоянием** (ending state) предыдущей оптимизации и позволяет осуществить «теплый» старт оптимизации

3.5. Окно и узел MathScript

Вид открытой подпалитры **Скрипты и формулы** (Scripts & Formulas) показан на рис. 3.24. В ее состав входят узел **MathScript** (MathScript Node), узел **Формула** (Formula Node), Экспресс-ВП **Формула** и подпалитры **Узлы скриптов** (Script Nodes), **Разбор формул** (Formula Parsing), **Оценка одномерных и двумерных зависимостей** (1D & 2D Evaluation), **Исчисление** (Calculus) и **Нули** (Zeroes).



Далее из подпалитры **Скрипты и формулы** будет рассмотрен новый и наиболее содержательный узел MathScript и одноименное окно.

Рис. 3.24. Вид подпалитры ***Скрипты и формулы*** *(Scripts & Formulas)*

LabVIEW 8.20 позволяет объединить графическое программирование потока данных с MathScript – текстовым языком математической обработки, включающим свыше 500 функций математики, обработки сигналов и анализа результатов. Перечень основных классов функций MathScript приведен в таблице:

Класс	Описание
advanced	Дополнительные математические функции
approximation	Аппроксимация, интегрирование и интерполяция
audio	Звуковые функции
basic	Основные математические функции
bitwise	Функции побитовой обработки
boolean	Логические функции
commands	Команды
comparison	Операции сравнения
constants	Константы
daq	Сбор данных
dsp	Цифровая обработка сигналов
geometry	Комбинаторная геометрия
libraries	Загрузка, выгрузка и вызовы общих библиотек
linalgebra	Линейная алгебра
matrix	Специальные матрицы
matrixops	Операции с матрицами
membership	Функции принадлежности
ode	Решение ОДУ
optimization	Оптимизация и минимизация
plots	Функции построения графиков
polynomials	Полиномиальные функции
programming	Структуры программирования
sets	Операторы работы с множествами
statistics	Статистика
string	Строковые функции
support	Функции поддержки
time	Функции даты и времени
timing	Функции синхронизации
trigonometric	Тригонометрические функции
vector	Векторный анализ

В следующей таблице приведен список членов класса функций **Цифровая обработка сигналов** (dsp), которые, в свою очередь, сами являются классами функций.

filter design	Конструирование фильтров
filter implementation	Реализация фильтров
linear systems	Линейные системы
modeling and prediction	Моделирование и прогнозирование
resampling	Функции изменения частоты выборки
spectral analysis	Спектральный анализ
transforms	Преобразования
waveform generation	Генерация осциллограмм
windows	Генерация окон

Ниже в таблице представлены члены класса **Структуры программирования** (programming).

Конструкция	Грамматика	Пример

Оператор Case-Switch	**switch** выражение
	case выражение
	группа инструкций
	[**case** выражение
	группа инструкций]
	...
	[**otherwise**
	группа инструкций]
	end
		switch mode
		case “start”
		a = 0;
		case “end”
		a = –1;
		otherwise
		a = a+1;
		end

For Loop	**for** выражение группа инструкций **end**	for i = 1:10 a = sin(2*pi*i/10) end
Оператор If-Else	**if** логическое выражение группа инструкций [**elseif** логическое выражение группа инструкций] ... [**else** группа инструкций] **end**	if b == 1 c = 3 else c = 4 end
Range	start:[step:]end	b = 2:2:20 *b* возвращает четные числа от 2 до 20. Если размер шага не задан, MathScript использует шаг, равный 1
While Loop	**while** выражение группа инструкций **end**	while i < 10 a = cos(2*pi*i/10) i = i + 1; end

Синтаксис функций и скриптов MathScript в общих чертах совместим с синтаксисом скриптов m-файлов, которые широко используются в таком пакете технических расчетов, как MATLAB [2, 3, 4]. Такая совместимость позволяет использовать в LabVIEW множество скриптов m-файлов, опубликованных в литературе или на Web-сайтах.

MathScript поддерживает как интерактивный, так и программный интерфейс с LabVIEW.

Интерактивный интерфейс поддерживается с помощью **окна MathScript** (MathScript Window) (рис. 3.25), которое запускается при помощи меню **Инструменты ⇒ Окно MathScript** (Tools ⇒ MathScript Window) и включает **окно вывода** (Output Window), **окно команд** (Command Window), **статус** (Status) и набор страниц с окнами: **переменные** (Variables), **скрипт** (Script) и **история** (History). **Окно вывода** отображает команды, которые вводятся в **окне команд**, и содержимое, ко-

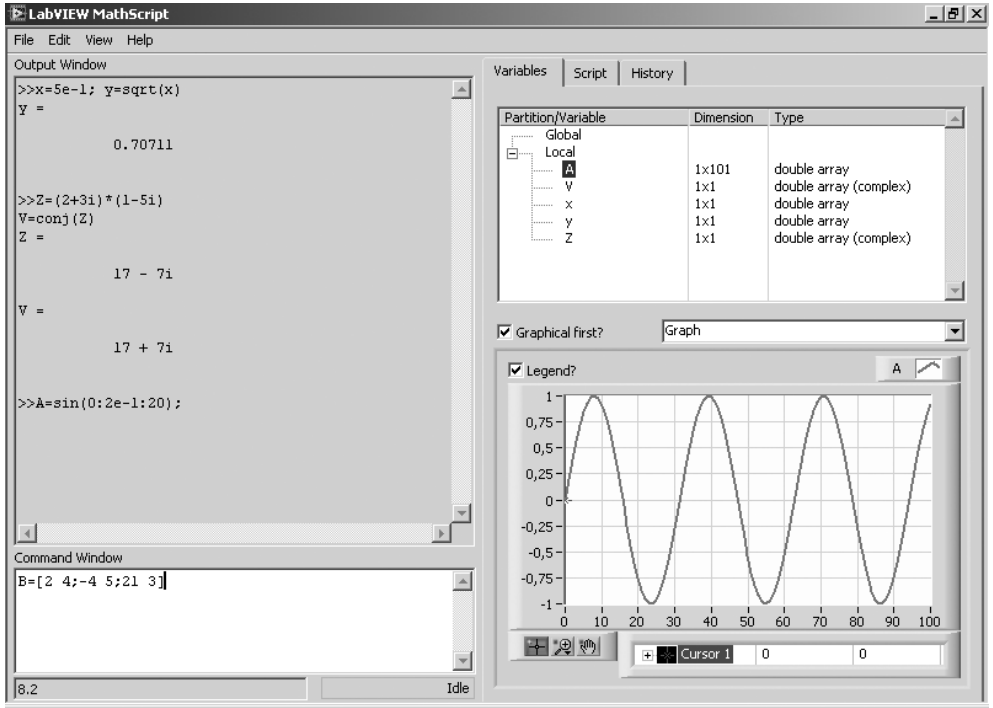


Рис. 3.25. Вид окна ***MathScript***

торое создает MathScript при выполнении этих команд. Для ввода многострочных команд необходимо нажать клавиши <Shift+Enter>. С помощью клавиш со стрелками можно выводить на индикацию в **окне команд** пункты из окна **история команд** (Command History), редактировать их и запускать на повторное выполнение.

Окно MathScript используется для редактирования и выполнения математических команд, создания математических скриптов и отображения переменных в табличном или графическом виде. В этом окне выполняется большинство скриптов, записанных в соответствии с синтаксисом языка MATLAB, но некоторые функции этого языка не поддерживаются и приводят к генерации сообщений об ошибке.

Программный интерфейс осуществляется с помощью **узла MathScript** (MathScript Node) – структуры, которая позволяет выполнять скрипты в блок-диаграмме LabVIEW. На рис. 3.26 показан вид узла MathScript из ВП MathScript Fundamentals, находящегося в библиотеке примеров NI Example Finder.

Наиболее широко в диалоговом режиме работы в окне MathScript используются данные числового типа. Они могут быть целыми или вещественными. Последние, в свою очередь, могут отображаться в формате с плавающей или фиксированной точкой. Перечень вариантов представления включает следующие опции:

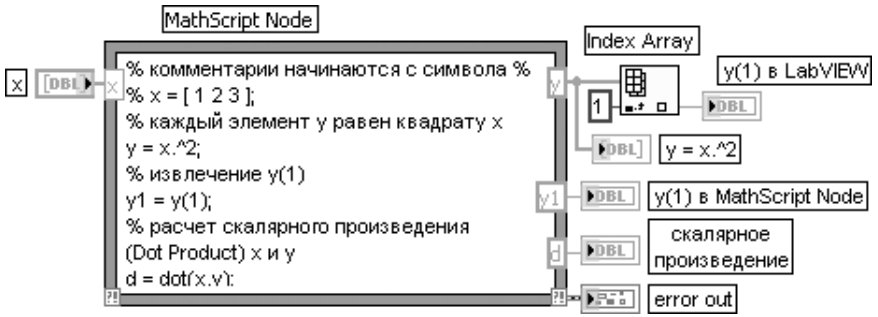


*Рис. 3.26. Вид узла **MathScript***

- **short** – отображает 5 цифр в формате с фиксированной десятичной точкой, например 314.15927;
- **long** – отображает 15 цифр в формате с фиксированной десятичной точкой, например 314.159265358979326;
- **short e** – отображает 5 цифр в формате с плавающей десятичной точкой, например 3.14159E+2;
- **long e** – отображает 15 цифр в формате с плавающей десятичной точкой, например 3.141592653589793E+2.

Формат отображения чисел может устанавливаться как интерактивно (вручную), так и программно. Ручной вариант установки реализуется в диалоговом окне MathScript Preferences, которое вызывается с помощью строки меню **Файл ⇒ Предпочтения** (File ⇒ Preferences). Для программного изменения формата отображения чисел используется функция format, входящая в класс support. Эта функция изменяет отображение всех чисел.

MathScript поддерживает выполнение арифметических операций только для чисел двойной точности. При создании число или матрица имеют по умолчанию именно этот формат. Вместе с тем для сохранения чисел с таким форматом требуется больше памяти, чем для целых чисел. Для взаимного преобразования форматов целых и вещественных числовых переменных служит ряд функции из класса support (таблица).

int64	Преобразует в 64-битовые целые со знаком
int32	Преобразует в 32-битовые целые со знаком
int16	Преобразует в 16-битовые целые со знаком
int8	Преобразует в 8-битовые целые со знаком
uint64	Преобразует в 64-битовые целые без знака
uint32	Преобразует в 32-битовые целые без знака
uint16	Преобразует в 16-битовые целые без знака
uint8	Преобразует в 8-битовые целые без знака
double	Преобразует в числа с двойной точностью
single	Преобразует в числа с одинарной точностью

Число в имени функции определяет размер памяти, отводимой для хранения числа, в битах. Вещественные данные могут быть числами **одинарной** (single – 4 байта) и **двойной** (double – 8 байт) точности.

Перечень арифметических операций, поддерживаемых MathScript, приведен в таблице.

Символ (функция)	Операция	Операнды
+ (plus())	Матричное сложение	
– (minus())	Матричное вычитание	
.* (times())	Поэлементное умножение массива	Оба операнда должны иметь одинаковые размеры или один из них должен быть скаляром. В последнем случае в операции со скаляром участвуют все элементы массива, и результат будет иметь тот же размер, что и исходный массив
.\ (ldivide())	Поэлементное левое деление массивов	
./ (rdivide())	Поэлементное правое деление массивов	
.^ (power())	Поэлементное возведение массива в степень	
* (mtimes())	Матричное умножение	Число столбцов первого операнда должно быть равно числу строк второго операнда
\ (mldivide())	Левое матричное деление	Операция **X=A\B** используется для решения системы линейных алгебраических уравнений **A*X=B**
/ (mrdivide())	Правое матричное деление	Операция **X=A/B** используется для решения системы линейных алгебраических уравнений **X*A =B**
^ (mpower())	Матричное возведение в степень	Операция **X^P** при скалярном **P** возводит квадратную матрицу **X** в степень **P**. Если **X** – скаляр, а **P** – квадратная матрица, то **X^P** возводит **X** в матричную степень **P**

Как видно из таблицы, для выполнения поэлементных мультипликативных операций (.*, ./, .\, и .^) перед соответствующим знаком должна быть установлена точка.

Для выполнения математических операций в MathScript используются следующие операторы.

Функция	Описание
abs(x)	Абсолютное значение
angle(x)	Аргументы комплексных чисел
cart2pol(x,y,z)	Прямоугольную в полярную (цилиндрическую)
cart2sph(x,y,z)	Прямоугольную в сферическую
ceil(x)	Округление до ближайшего целого справа от x
conj(x)	Комплексное сопряжение

Функция	Описание
cplxpair(x)	Сортирует комплексные числа по комплексно-сопряженным парам
cumprod(x)	Накопленные произведения
cumsum(x)	Накопленные суммы
exp(x)	Экспонента
expm(x)	Матричная экспонента
factor(x)	Простые множители
factorial(x)	N!
fix(x)	Округление до ближайшего целого в сторону нуля
floor(x)	Округление до ближайшего целого слева от x
gcd(x,y)	Наибольший общий делитель
imag(x)	Мнимая часть
lcm(x,y)	Наименьший общий множитель
log(x)	Натуральный логарифм
logm(x)	Матричный натуральный логарифм
log10(x)	Логарифм по основанию 10
log2(x)	Логарифм по основанию 2
max(x)	Наибольший элемент
min(x)	Наименьший элемент
mod(x,y)	Остаток по модулю
nextpow2(x)	Следующая степень 2
planerot(x)	Матрица поворота
pol2cart(x,y,z)	Полярную (цилиндрическую) в прямоугольную
pow2(x)	Степень 2
prod(x)	Произведение
real(x)	Действительная часть
reallog(x)	Натуральный логарифм положительных вещественных чисел
realpow(x,y)	Степени для каждого элемента
realsqrt(x)	Квадратные корни из положительных вещественных чисел
rem(x,y)	Остаток
round(x)	Округление до ближайшего целого
sign(x)	Знаки элементов матрицы
sort(x)	Сортирует элементы векторов или матриц
sortrows(x)	Сортирует строки матрицы в порядке возрастания
sph2cart(x,y,z)	Сферическую в прямоугольную
sqrt(x)	Извлечение квадратного корня
sqrtm(x)	Матричная операция извлечение квадратного корня
sum(x)	Сумма

Отдельный класс функций MathScript образуют тригонометрические функции (trigonometric), перечисленные в таблице.

Функция	Описание
sin(x), cos(x) tan(x), cot(x)	Синус, косинус, тангенс, котангенс (аргумент x – в радианах)
sec(x), csc(x)	Секанс и косеканс
asin(x), acos(x) atan(x), acot(x) asec(x), acsc(x)	Обратные тригонометрические функции (возвращают результат в радианах)
sinh(x), cosh(x) tanh(x), coth(x) sech(x), csch(x)	Гиперболические тригонометрические функции
asinh(x), acosh(x) atanh(x), acoth(x) asech(x), acsch(x)	Обратные гиперболические тригонометрические функции
sinc(x)	Функция sin(x)/x
atan2(x,y)	Арктангенс, определяемый с учетом квадрантов (возвращают результат в радианах)

В MathScript можно определять переменные, а затем использовать их в выражениях. Процедура определения заключается в присваивании переменной, заданной именем, значения выражения. Имя переменной не должно совпадать с именами встроенных процедур, функций и встроенных переменных системы. Имена переменных должны начинаться с буквы и не могут содержать математических символов и знаков пунктуации. В именах переменных должны использоваться только символы ASCII. Запись имени чувствительна к регистру.

Выражение в правой части оператора присваивания может быть числом, арифметическим выражением, строкой символов или символьным выражением. Переменные MathScript изменяются в соответствии с типом данных. Так, например, если a=sin(3*pi/2), то **a** – вещественное число двойной точности, если же a='result', то **a** становится строковой переменной.

Если символ «;» в конце выражения отсутствует, то в качестве результата выводятся имя переменной и ее значение. Наличие символа «;» передает управление следующей командной строке. Если команда не содержит знака присваивания, то по умолчанию вычисленное значение присваивается специальной системной переменной ans.

Помимо ans, в число системных переменных входят pi – число π, i,j – мнимая единица ($\sqrt{-1}$), inf – машинный символ бесконечности, namelengthmax – максимальное число символов допустимой переменной и nan – неопределенный результат (0/0, ∞/∞ и т. п.). При вводе комплексных чисел мнимая единица может быть записана до или после мнимой части. В первом случае после мнимой единицы должен находиться знак умножения «*», во втором случае он необязателен, однако мнимая часть и последующая мнимая единица должны быть записаны слитно.

Любая переменная в MathScript является матрицей, в простейшем случае это матрица, состоящая из одной строки и одного столбца. Переменную, представляющую простой список данных, называют одномерным массивом, или вектором. Для доступа к данным, хранящимся в определенном элементе массива, необходимо указать имя массива и порядковый номер этого элемента, называемый индексом.

При необходимости хранения данных в виде таблиц, содержащих строки и столбцы, используют двумерные массивы (матрицы). Для доступа к элементам такого массива указывают имя массива и два индекса: первый должен соответствовать номеру строки, а второй – номеру столбца, на пересечении которых находится искомый элемент. Индексация массивов начинается с единицы. Индексы могут быть только целыми положительными числами или нулем.

Элементы массивов могут быть введены вручную или рассчитаны программно. Так, для определения вектора-строки необходимо ввести имя массива, а затем после знака присваивания в квадратных скобках перечислить элементы массива, отделяя их друг от друга пробелами или запятыми. Элементы вектора-столбца вводятся через точку с запятой. Подобным же образом элементы в строках матрицы разделяются пробелами или запятыми, а строки – точками с запятыми.

Один из способов программного создания массива связан с использованием конструкции

[*name*] = *Xn:dX:Xk,*

где *name* – имя переменной, в которую будет записан сформированный массив, *Xn*, *Xk* – значение первого и последнего элементов массива, *dX* – шаг, с помощью которого формируется каждый следующий элемент массива.

Ограничения на создание массивов в MathScript связаны с тем, что их размерность не должна превышать 2 и их элементами не могут быть кластеры.

Наряду с встроенными функциями MathScript позволяет пользователю определять собственные функции. При записи собственной функции она должна иметь следующий синтаксис:

function *outputs=function_name(inputs)*
%documentation
script

Определение каждой функции начинается со слова **function**.

outputs представляет список выходных переменных функции. Если функция имеет более одной выходной переменной, необходимо заключить их в квадратные скобки и отделить друг от друга запятыми или пробелами.

function_name представляет имя определяемой функции.

inputs содержит список входных переменных функции, которые разделяются запятыми.

documentation представляет содержимое справочного сообщения, которое будет выводиться LabVIEW для данной функции при вызове пользователем команды помощи (help). В начале каждой строки справочной информации должен находиться символ %.

script определяет исполняемое тело функции.

В следующей таблице приведены варианты определения функций.

function foo	function a = foo	function [a b] = foo
function foo()	function a = foo()	function [a b] = foo()
function foo(g)	function a = foo(g)	function [a b] = foo(g)
function foo(g, h)	function a = foo(g, h)	function [a b] = foo(g, h)

Если в одном файле MathScript определено несколько функций, то все функции, введенные вслед за первой, становятся ее подфункциями и доступны только для главной функции. Функция может вызвать только те функции, которые находятся ниже ее. Поэтому функция не может вызвать саму себя. LabVIEW также не разрешает циклические рекурсивные вызовы функции. Например, foo не может вызвать bar, если bar вызывает foo.

Пользователь может определить для функции дополнительные входы и выходы. Функция nargin позволяет определить число входных аргументов, поддерживаемых функцией, которая вызывает nargin. Если число поддерживаемых входов меньше максимального числа входов для этой функции, то свободным входам могут быть присвоены значения по умолчанию. Подобным же образом функция nargout позволяет определить число выходных аргументов, запрошенных функцией. Если число запрошенных выходов меньше максимального числа выходов функции, то вычисление незапрошенных выходов может быть исключено.

После определения функции ее необходимо сохранить в папке, путь к которой указан в диалоговом окне MathScript Preferences. Диалоговое окно вызывается с помощью строки меню **Файл ⇒ Предпочтения** (File ⇒ Preferences). Имя файла функции должно быть таким же, как и имя самой функции, и иметь расширение .m в нижнем регистре.

Функции генерации и обработки сигналов LabVIEW

4

Данная глава содержит описание функций, размещенных в категории **Обработка сигналов** (Signal Processing) палитры функций LabVIEW (рис. 4.1).

Функции генерации и обработки сигналов, размещенные в подпалитре **Обработка сигналов**, позволяют формировать как отдельные значения, так и массивы отсчетов сигналов и шумов и производить их обработку с помощью различных операций в частотной и временной областях.

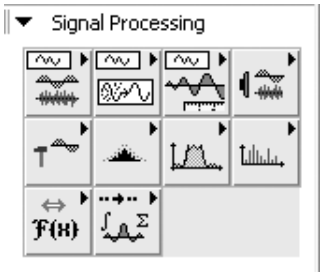
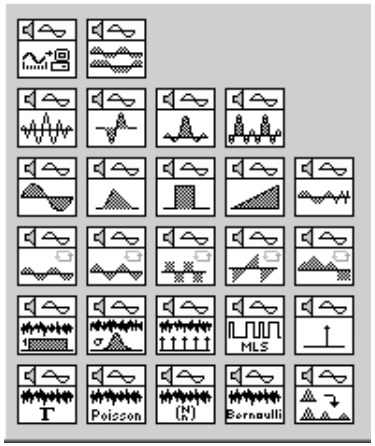


*Рис. 4.1. Вид категории функций **Обработка сигналов** (Signal Processing)*

4.1. Функции генерации сигналов и шумов



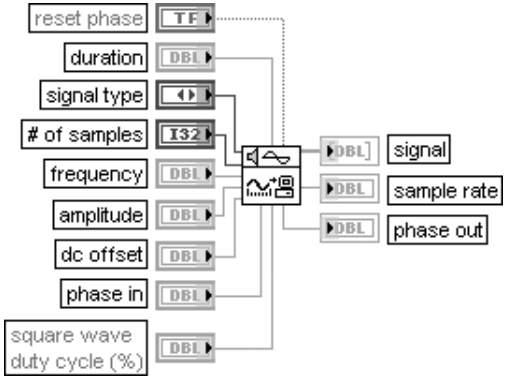
ВП из палитры генерации сигналов и шумов (рис. 4.2) используются для формирования детерминированных и случайных сигналов с заданным набором параметров. Первые два ВП в верхнем ряду представляют многофункциональные генераторы сигналов с широким набором регулируемых параметров. ВП, размещенные во второй и третьей строках, предназначены для генерации наиболее широко применяемых детерминированных периодических сигналов, а ВП, находящиеся в четвертой и пятой строках,

Рис. 4.2. Палитра функций генерации сигналов и шумов

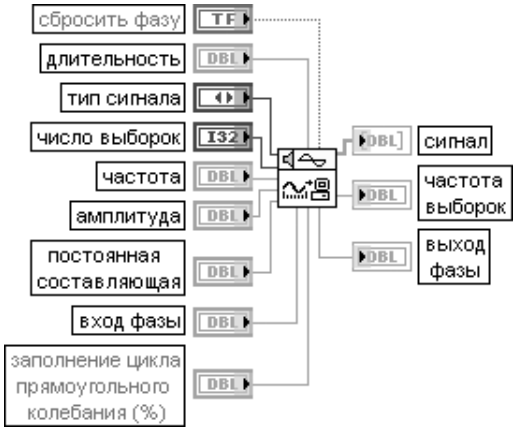
служат для расчета шумов с различными законами амплитудного и спектрального распределения.

Описание функций генерации сигналов и шумов приведено в таблице.

Signal Generator by Duration



Генератор сигналов с заданной длительностью



ВП генерирует **сигнал** (signal), имеющий форму, задаваемую на входе **тип сигнала** (signal type).

Вход **сбросить фазу** (reset phase) определяет начальную фазу выходного сигнала. При установке на входе состояния ИСТИНА (по умолчанию) начальная фаза сигнала определяется входом **вход фазы** (phase in). В противном случае начальная фаза устанавливается равной значению фазы на **выходе фазы** (phase out) при последнем выполнении этого ВП.

Вход **длительность** (duration) задает время в секундах, равное длительности генерируемого выходного сигнала. По умолчанию значение длительности равно 1,0.

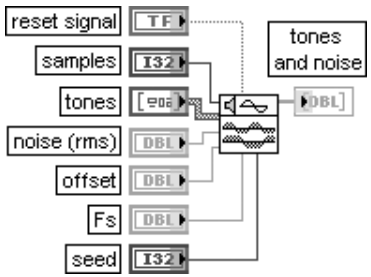
Вход **тип сигнала** (signal type) задает следующие типы генерируемого сигнала:

0	**Синусоидальный** (sine) (по умолчанию)
1	**Косинусоидальный** (cosine)

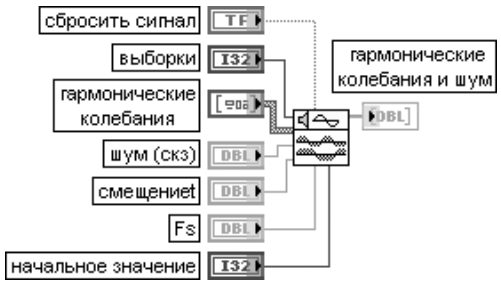
2	**Треугольный** (triangle)
3	**Прямоугольный** (square)
4	**Пилообразный** (sawtooth)
5	**Линейно нарастающий** (increasing ramp)
6	**Линейно спадающий** (decreasing ramp)

Вход **число выборок** (# of samples) задает число выборок выходного сигнала. По умолчанию это значение равно 100.
Вход **частота** (frequency) определяет частоту выходного сигнала в герцах. По умолчанию значение частоты равно 10. При задании частоты необходимо учитывать требование выполнения критерия Найквиста: **частота < число выборок/(2*длительность)**.
Вход **амплитуда** (amplitude) задает амплитуду выходного сигнала. По умолчанию значение амплитуды равно 1,0.
Вход **постоянное смещение** (dc offset) задает постоянное смещение или значение постоянной составляющей выходного сигнала. По умолчанию значение постоянной составляющей равно 0.
Вход фазы (phase in) определяет начальную фазу (в градусах) выходного сигнала при установке входа **сбросить фазу** (reset phase) в состояние ИСТИНА. По умолчанию значение на **входе фазы** равно 0.
Вход **заполнение цикла прямоугольного колебания** (square wave duty cycle) определяет время (в процентах от периода), в течение которого прямоугольный сигнал имеет высокий уровень. ВП использует данный параметр только для прямоугольного сигнала. По умолчанию значение на входе равно 50%.
Выход **сигнал** (signal) представляет сгенерированный массив выборок сигнала.
Выход **частота выборок** (sample rate) отображает частоту дискретизации выходного сигнала. **Частота выборок** равна отношению **числа выборок** к **длительности**.
Выход фазы (phase out) указывает значение фазы (в градусах) последней выборки выходного сигнала

Tones and Noise



Гармонические колебания и шум



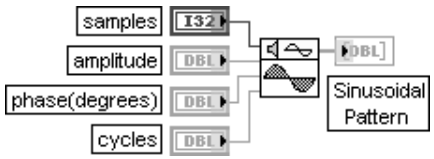
ВП генерирует массив, содержащий сумму гармонических колебаний, шум и постоянную составляющую.
Вход **сбросить сигнал** (reset signal) при подаче значения ИСТИНА устанавливает фазу каждого гармонического колебания равной значению **фазы** (phase) из массива **гармонические колебания** (tones), значение начального числа при генерации шума равным значению входа **начальное значение** (seed) и отметку времени равной нулю. По умолчанию состояние входа – ЛОЖЬ.
Вход **выборки** (samples) определяет число выборок выходного массива **гармонические колебания и шум** (tones and noise). По умолчанию это число равно 1000.
Вход **гармонические колебания** (tones), являющийся кластером, содержит параметры каждого гармонического колебания:
- **частота** (frequency) определяет частоту синусоидального колебания в герцах;
- **амплитуда** (amplitude) определяет амплитуду синусоидального колебания;

- **фаза** (phase) определяет начальную фазу синусоидального колебания. По умолчанию фаза равна 0.

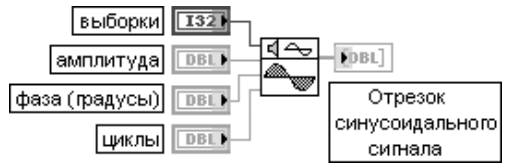
Вход **шум** (noise) определяет среднеквадратичное значение (с.к.з.) аддитивного гауссовского шума. По умолчанию значение **шума** равно 0,0.
Вход **смещение** (offset) задает уровень постоянной составляющей сигнала. По умолчанию уровень равен 0,0.
Вход **Fs** определяет частоту выборок в единицах число выборок/с. По умолчанию частота равна 1000.
Установка на входе **начальное значение** (seed) значения >0 вызывает инициализацию генератора аддитивного шума. По умолчанию значение входа равно –1. Если **начальное значение** меньше или равно 0, то генератор шума не инициализируется и продолжает генерацию шума на основе предыдущих значений.
Выход **гармонические колебания и шумы** (tones and noise) отображает сгенерированный выходной массив

В последующих ВП генерации детерминированных сигналов, рассмотренных в таблице, поясняется назначение только специфических входов, поскольку назначение таких входов, как **выборки** (samples) и **амплитуда** (amplitude), было рассмотрено выше при описании ВП **Генератор сигналов с заданной длительностью** (Signal Generator by Duration). В приводимых расчетных выражениях сигналов буквой ***a*** обозначается **амплитуда** сигнала (по умолчанию ***a*** = 1,0), буквой ***d*** – **задержка** (delay), буквой ***n*** – объем **выборки** (samples) (по умолчанию ***n*** = 128). При этом текущий индекс ***i*** в расчетных выражениях изменяется в диапазоне от 0 до ***n***–1.

Sine Pattern



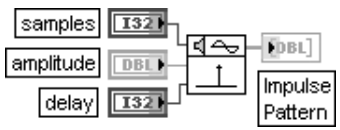
Отрезок синусоидального сигнала



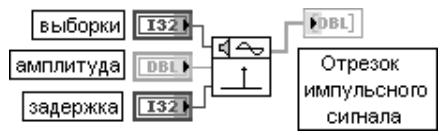
ВП генерирует массив, содержащий отрезок синусоидального сигнала.
Вход **циклы** (cycles) определяет число полных периодов **синусоидального сигнала**.
По умолчанию число циклов равно 1,0. Поскольку число циклов задается в форме с плавающей запятой, то оно может быть нецелым. Более того, оно может быть и отрицательным, что является математически корректным и полезным для понимания отрицательных частот в преобразованиях Фурье и в спектральном анализе.
Значения y_i массива **синусоидального сигнала** рассчитываются следующим образом:

$$y_i = a\sin\left(\frac{2\pi i k}{n} + \frac{\pi\varphi_0}{180}\right)$$, где k – **число периодов** (cycles), φ_0 – **начальная фаза в градусах** (phase)

Impulse Pattern

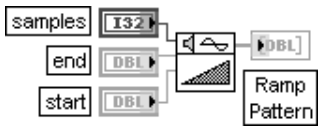


Отрезок импульсного сигнала

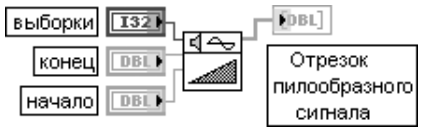


ВП генерирует массив, содержащий отрезок импульсного сигнала.
Вход **задержка** (delay) задает задержку единичного по длительности импульса относительно начала сигнала. Фактически это индекс элемента массива выходного **импульсного сигнала** (Impulse Pattern), в котором находится число, равное **амплитуде**. Должно соблюдаться условие **задержка** ≥0. Если **задержка** <0 или больше числа **выборок**, то ВП устанавливает на выходе **импульсного сигнала** нулевое значение и возвращает ошибку

Ramp Pattern

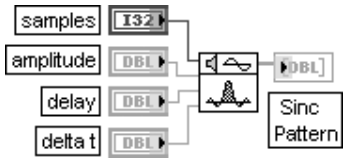


Отрезок пилообразного сигнала

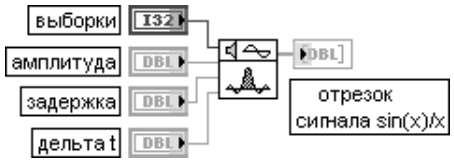


ВП рассчитывает массив, содержащий отрезок пилообразного сигнала.
Входы **начало** (start) и **конец** (end) задают начальный и конечный уровни **пилообразного сигнала.** По умолчанию значения этих параметров равны нулю.
Значения y_i массива **пилообразного сигнала** рассчитываются следующим образом: $y_i = y_0 + i\Delta y$, где $\Delta y = \frac{y_{n-1} - y_0}{n-1}$; y_0 – значение на входе **начало**, y_{n-1} – значение на входе **конец**.
Поскольку ВП не налагает ограничений на значения **начало** и **конец**, то могут быть сформированы как линейно нарастающий, так и линейно спадающий сигналы

Sinc Pattern



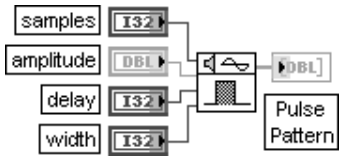
Отрезок сигнала sin(x)/x



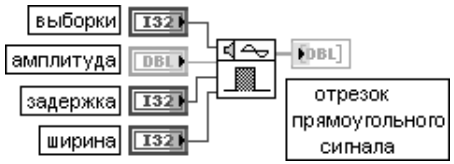
ВП генерирует массив, содержащий отрезок сигнала **sin(x)/x**.
Вход **задержка** (delay) определяет сдвиг максимума сигнала **sin(x)/x**. По умолчанию значение **задержки** равно 0,0. Максимальное значение сигнала **sin(x)/x** приходится на индекс $i = d / \Delta t$, где d – **задержка** (delay), Δt – значение на входе delta t.
Вход Δt (delta t) определяет фактически частоту следования нулей сигнала **sin(x)/x**. По умолчанию значение Δt равно 0,1, что соответствует расстоянию между нулями, равному 10 отсчетам и ширине центрального лепестка сигнала равной 20 отсчетам.
Значения y_i массива сигнала **sin(x)/x** рассчитываются следующим образом:

$$y_i = a\frac{\sin\pi(i\Delta t - d)}{\pi(i\Delta t - d)}$$

Pulse Pattern



Отрезок прямоугольного сигнала

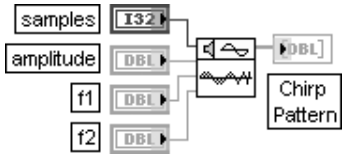


ВП генерирует массив, содержащий отрезок прямоугольного сигнала.

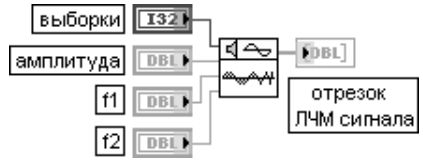
Значения y_i массива **прямоугольного сигнала** рассчитываются следующим образом:

$$y_i = \begin{cases} a \quad если \quad d \le i < (d + w) \\ 0{,}0 \ иначе \end{cases}$$, где w – **ширина** (width)

Chirp Pattern



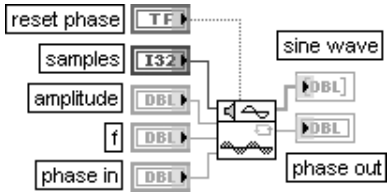
Отрезок ЛЧМ сигнала



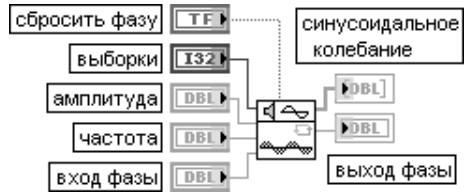
ВП генерирует массив, содержащий отрезок сигнала с линейно изменяющейся частотой (сигнал с линейной частотной модуляцией – ЛЧМ сигнал).
Входы f_1 и f_2 представляют начальную и конечную частоты **ЛЧМ сигнала**, заданные в нормализованных единицах 1/период, где период задан числом отсчетов.
Значения y_i массива **ЛЧМ сигнала** рассчитываются следующим образом:
$y_i = a\sin((\pi i(f_2 - f_1)/n + 2\pi f_1) * i)$

Для ВП генерации детерминированных сигналов, описанных в таблице, назначение входов **вход фазы** (phase in), **сбросить фазу** (reset phase) и выхода **выход фазы** (phase out) было рассмотрено ранее при анализе ВП **Генератор сигналов с заданной длительностью** (Signal Generator by Duration). Вход f определяет частоту колебания, выраженную в нормализованных единицах 1/**период**, где **период** выражен числом отсчетов. По умолчанию значение частоты при одном периоде колебания, приходящемся на 128 выборок, равно 7,8125E-3.

Sine Wave

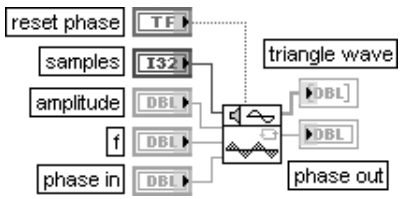


Синусоидальное колебание

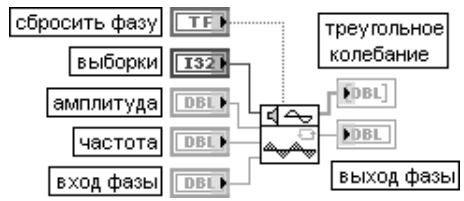


ВП генерирует массив числовых данных, представляющий синусоидальное колебание.
Значения y_i массива **синусоидального колебания** рассчитываются следующим образом:
$y_i = a\ \sin(phase[i])$, где $phase[i] = initial\ phase + f * 360 * i$

Triangle Wave



Треугольное колебание

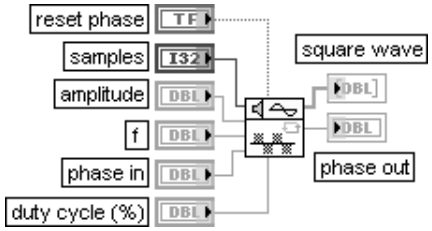


ВП генерирует массив, содержащий треугольное колебание.
Значения y_i массива **треугольного колебания** рассчитываются следующим образом:

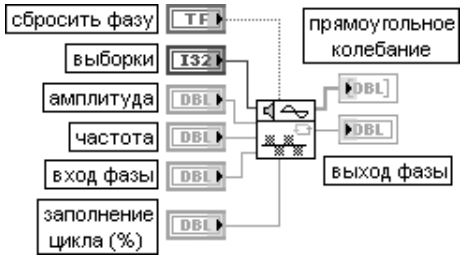
$$y_i = a\ tri(phase[i]),\ \text{где}\ tri(phase[i]) = \begin{cases} \frac{p}{90} & 0 < p < 90 \\ 2 - \frac{p}{90} & 90 < p < 270 \\ \frac{p}{90} - 4 & 270 < p < 360 \end{cases},$$

$p = (phase[i]\ по\ модулю\ 360),\ phase[i] = initial\ phase + f * 360 * i$

Square Wave



Прямоугольное колебание

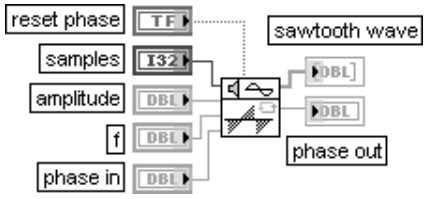


ВП генерирует массив, содержащий прямоугольное колебание.
Значения y_i массива **прямоугольного колебания** рассчитываются следующим образом:

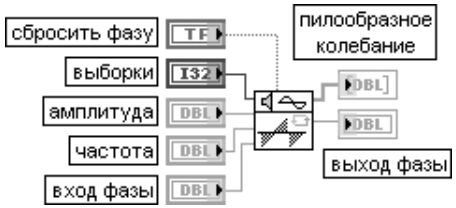
$$y_i = a\ square(phase[i]),\ \text{где}\ square(phase[i]) = \begin{cases} 1 & 0 \le p < (\frac{duty}{100}360) \\ -1 & (\frac{duty}{100}360) \le p < 360 \end{cases}$$

$p = (phase[i]\ по\ модулю\ 360)$, $duty$ – длительность импульса,
Заполнение цикла (duty cycle) определяет время (в процентах от периода), в течение которого прямоугольный сигнал имеет высокий уровень. Значение по умолчанию для этого входа равно 50%.
$phase[i] = initial\ phase + f * 360 * i$

Sawtooth Wave



Пилообразное колебание

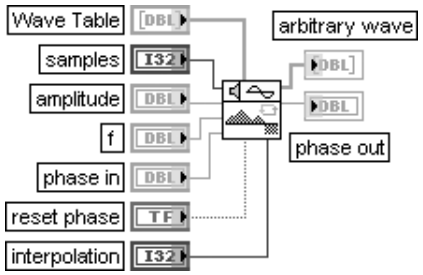


ВП генерирует массив, содержащий пилообразное колебание.
Значения y_i массива **пилообразного колебания** рассчитываются следующим образом:

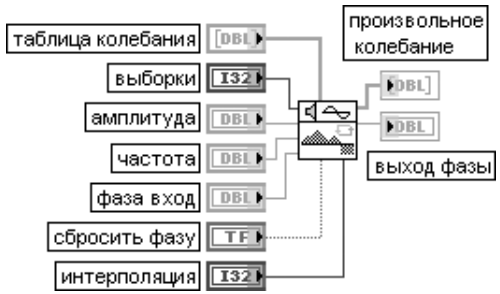
$$y_i = a\ sawtooth(phase[i]),\ \text{где}\ sawtooth(phase[i]) = \begin{cases} \frac{p}{180} & 0 \le p < 180 \\ \frac{p}{180} - 2 & 180 \le p < 360 \end{cases}$$

$p = (phase[i]\ по\ модулю\ 360),\ phase[i] = initial\ phase + f * 360 * i$

Arbitrary Wave



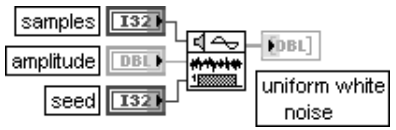
Произвольное колебание



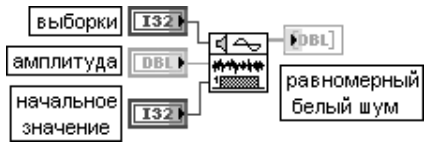
ВП генерирует массив, содержащий произвольное колебание.
Вход **таблица колебания** (Wave Table) задает форму колебания на интервале одного периода. На основе массива **таблицы колебания** ВП формирует **произвольное колебание** (arbitrary wave).
Вход **интерполяция** (interpolation) определяет вид интерполяции, которую ВП использует при генерации **произвольного колебания** из массива **таблица колебания**. По умолчанию значение входа равно 0 (отсутствие интерполяции). Если на входе **интерполяция** установлена 1, то ВП использует линейную интерполяцию.
Значения y_i массива **произвольного колебания** рассчитываются следующим образом:
$y_i = a * arb(phase[i])$,
где $arb(phase[i]) = WT((phase[i]\ по\ модулю\ 360) * m\ /\ 360)$, m – число значений в **таблице колебания** (Wave Table), $WT(x) = Wave\ Table[int(x)]$, если вход **интерполяция = 0**.
$WT(x)$ = линейно интерполированным значениям $Wave\ Table[int(x)]$
и $Wave\ Table[int(x) + 1]\ по\ модулю\ m]$, если вход **интерполяция = 1**.
$phase[i] = initial\ phase + f * 360 * i$

В следующей таблице рассмотрены ВП генерации случайных сигналов с различными законами амплитудного и спектрального распределений.

Uniform White Noise

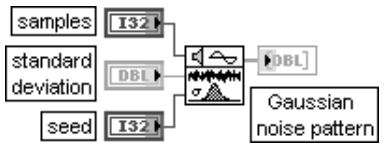


Равномерный белый шум

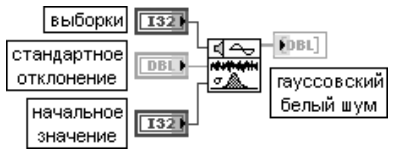


ВП генерирует псевдослучайный белый шум с равномерным законом амплитудного распределения, значения которого находятся в диапазоне **[-*a*:*a*]**, где ***a*** представляет абсолютное значение **амплитуды**.
По умолчанию значение амплитуды равно 0,0.
Псевдослучайная последовательность повторяется приблизительно через 2^{90} выборок

Gaussian White Noise



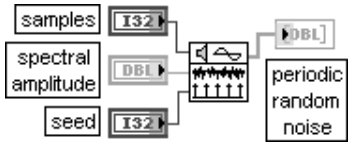
Гауссовский белый шум



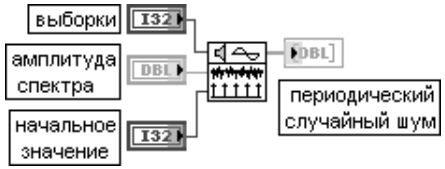
ВП генерирует псевдослучайную последовательность с гауссовским (нормальным) распределением с параметрами *(**μ,sigma**)* = (*0,**s***), где ***s*** является абсолютным значением заданного **стандартного** (среднеквадратичного) **отклонения** (standard deviation). По умолчанию значение **стандартного отклонения** равно 1,0.
Гауссовский шум описывается следующей функцией плотности вероятностей:

$$f(x) = \frac{1}{\sqrt{2\pi}s} e^{-\frac{1}{2}\left(\frac{x}{s}\right)^2}$$

Periodic Random Noise



Периодический случайный шум

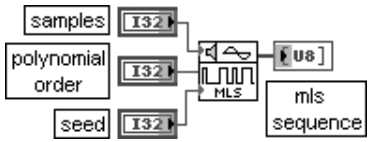


ВП генерирует массив, содержащий **периодический случайный шум** (PRN).
Вход **амплитуда спектра** (spectral amplitude) задает амплитуду частотных составляющих **периодического случайного шума**.
Выходной массив содержит все частоты, которые могут быть представлены целым числом периодов на установленном числе **выборок** (samples). Каждая частотная компонента имеет величину, заданную на входе **амплитуда спектра**, и случайную фазу. Таким образом, выходной массив можно представить как результат суммирования синусоид с одинаковыми амплитудами и случайными фазами.
Выходной массив **периодический случайный шум** ограничен по величине следующими значениями:

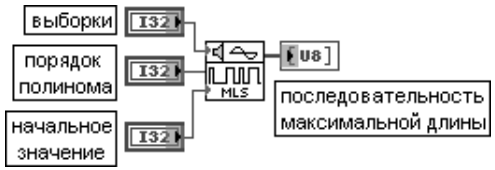
амплитуда спектра $*\left(\frac{\text{выборки}}{2} - 1\right)$, если **выборки** являются четным числом;

амплитуда спектра $*\left(\frac{\text{выборки} - 1}{2}\right)$, если **выборки** являются нечетным числом

Binary MLS

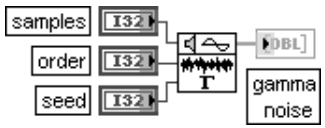


Двоичная последовательность максимальной длины

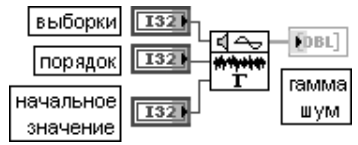


ВП генерирует двоичную **последовательность максимальной длины** (maximum length sequence – MLS), используя деление по модулю для простого полинома, имеющего порядок, заданный на входе **порядок полинома** (polynomial order). Значение по умолчанию на входе **порядок полинома** равно 31

Gamma Noise

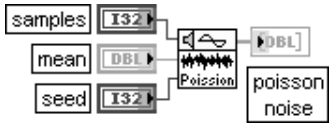


Гамма-шум

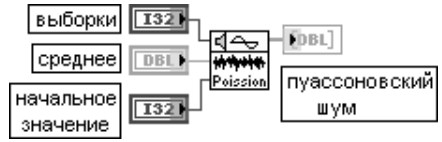


ВП генерирует псевдослучайный набор значений, которые представляют интервалы времени ожидания заданного числа событий пуассоновского процесса с единичным средним.
Вход **порядок** (order) определяет число событий. По умолчанию **порядок** равен 1

Poisson Noise

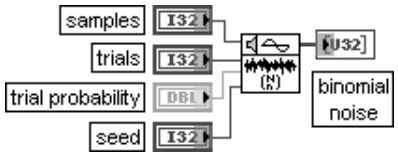


Пуассоновский шум

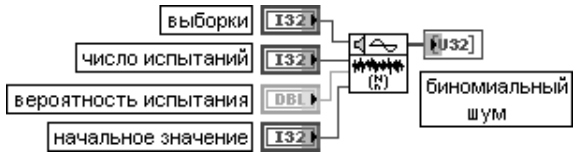


ВП генерирует псевдослучайную последовательность значений, которые представляют число событий ординарного Пуассоновского процесса, появляющихся на заданном интервале, определенном величиной на входе **среднее** (mean). По умолчанию значение **среднего** равно 1,0

Binomial Noise

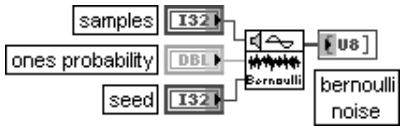


Биномиальный шум



ВП генерирует псевдослучайную последовательность с биномиальным амплитудным распределением, значения которой представляют число реализаций событий, заданных вероятностью совершения событий и числом испытаний.
Вход **число испытаний** (trials) представляет число испытаний, выполняемых для каждого элемента **биномиального шума** (binomial noise). По умолчанию это число равно 1.
Вход **вероятность испытания** (trial probability) представляет вероятность того, что данное испытание будет успешным (1). По умолчанию значение входа равно 0,5

Bernoulli Noise



Шум Бернулли

ВП генерирует псевдослучайный шум из единиц и нулей.
Каждый элемент на выходе **шум Бернулли** рассчитывается с помощью способа, эквивалентного подбрасыванию монеты с вероятностью выпадения единицы, определяемой значением на входе **вероятность единицы** (ones probability). Если значение **вероятность единицы** равно 0,7, то каждый элемент **шум Бернулли** имеет 70% вероятности быть единицей и 30% вероятности быть нулем. По умолчанию значение входа равно 0,5

4.2. Функции операций с сигналами

Функции операций с сигналами выполняют обработку сигналов во временной области (рис. 4.3) и включают функции свертки и корреляции сигналов, определения уровня постоянного и переменного напряжений в сигнале, нормирования, изменения частоты выборок и определения параметров пиков.

Набор функций операций с сигналами рассмотрен ниже в таблице.

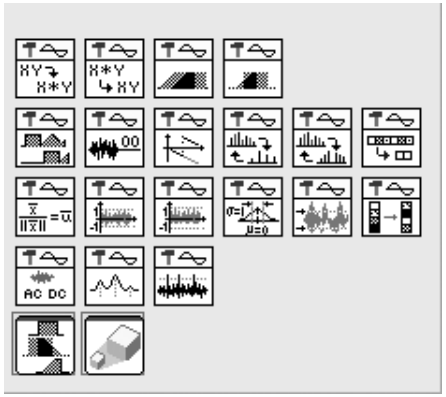


Рис. 4.3. Вид палитры функций операций с сигналами

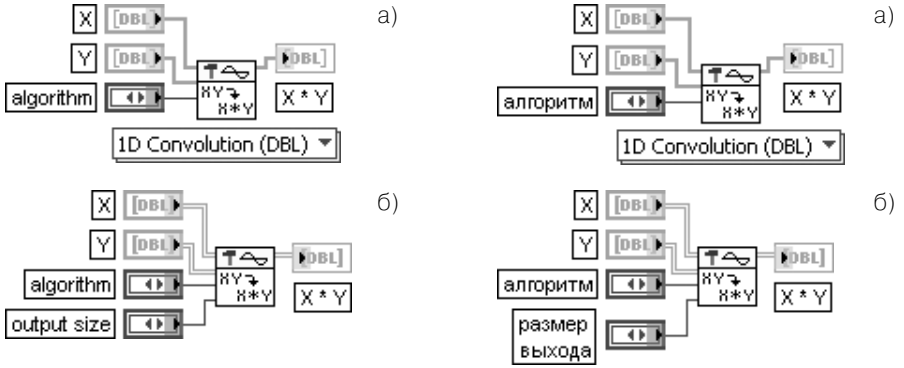


Рис. 4.4. Варианты подключения ВП Свертка (Convolution)

Это полиморфный ВП выполняет свертку двух последовательностей ***X*** и ***Y***. Входные последовательности могут быть одномерными или двумерными массивами, содержащими вещественные или комплексные значения.

Для непрерывных сигналов $x(t)$ и $y(t)$ выражение для расчета свертки выглядит следующим образом:

$$h(t) = \int_{-\infty}^{\infty} x(\tau)y(t-\tau)d\tau.$$

В данном ВП алгоритм вычисления свертки определяется значением, установленным на входе **алгоритм** (algorithm).

При выборе значения **прямой** (direct) ВП использует прямой метод линейной свертки.

При этом для дискретных одномерных последовательностей ***X*** и ***Y*** длиной n и m соответственно (рис. 4.4а) расчетное выражение для каждого элемента свертки будет иметь вид:

$$h_i = \sum_{k=0}^{n-1} x_k y_{i-k}, \text{ для } i = 0,1,2\ldots n+m-2.$$

При выборе значения **частотная область** (frequency domain) (по умолчанию) ВП рассчитывает свертку с помощью алгоритма, базирующегося на преобразовании Фурье.

Прямой метод является более быстрым для коротких последовательностей, в то время как частотный метод более эффективен для длинных последовательностей.

В случае двумерных последовательностей ***X*** и ***Y*** (рис. 4.4б) алгоритм расчета свертки выглядит следующим образом:

$$h(i,j) = \sum_{m=0}^{M_1-1}\sum_{n=0}^{N_1-1} x(m,n)y(i-m,j-n),$$

где $i = 0,1,2\ldots M_1 + M_2 - 2$ и $j = 0,1,2\ldots N_1 + N_2 - 2$, M_1, N_1 – число строк и столбцов матрицы **X**, M_2, N_2 – число строк и столбцов матрицы **Y**.

При выборе варианта двумерной свертки дополнительный вход **размер выхода** (output size) определяет размер свертки ***X*** и ***Y***. На этом входе могут быть установлены три варианта размера:

0	**Полная (по умолчанию)** (full (default)) – устанавливает ширину свертки ***X*** и ***Y*** на единицу меньше суммы ширины ***X*** и ***Y*** и устанавливает высоту свертки ***X*** и ***Y*** на единицу меньше суммы высот **X** и **Y**
1	**Размер X** (size X) – устанавливает ширину и высоту свертки ***X*** и ***Y*** по ширине и высоте ***X***. Этот вариант в большей степени подходит для обработки изображений. Если ***X*** – это изображение, подлежащее фильтрации, а ***Y*** – матрица небольшого размера, называемая **ядром свертки** (convolution kernel), то свертка ***X*** и ***Y*** представляет отфильтрованное изображение, размер которого совпадает с размером исходного изображения
2	**Компактная** (compact) – устанавливает ширину свертки ***X*** и ***Y*** на единицу больше разности ширины ***X*** и ***Y*** и высоту свертки ***X*** и ***Y*** на единицу больше разности высот ***X*** и ***Y***. Ширина и высота ***X*** должны быть больше или равны соответственно ширине и высоте ***Y***

ВП **Свертка** (Convolution) размещен в палитрах **Операции с сигналами** (Signal Operation) и **Дополнительная КИХ фильтрация** (Advanced FIR Filtering), поскольку может использоваться как при анализе данных во временной области, так и при их фильтрации

Deconvolution **Деконволюция**

ВП рассчитывает деконволюцию входной последовательности **свертка *X* и *Y*** с последовательностью ***Y***, используя преобразование Фурье.

При этом выполняются следующие шаги:

1. Производится расчет преобразования Фурье входных последовательностей ***X * Y*** и ***Y***.
2. Производится деление преобразования Фурье последовательности ***X * Y*** на преобразование Фурье последовательности ***Y***.
3. Выполняется обратное преобразование Фурье результата деления.

Длина последовательности ***X * Y*** n должна быть больше или равна длине последовательности ***Y*** m.

Длина выходной последовательности равна $n + m - 1$.

Деконволюция является неустойчивой процедурой, и ее не всегда можно рассчитать численно вследствие возможного наличия нулей в преобразовании Фурье последовательности ***Y***

AutoCorrelation **Автокорреляционная функция**

Этот полиморфный ВП вычисляет автокорреляционную функцию одномерной или двумерной входной последовательности **X** с вещественными или комплексными элементами.

Автокорреляционная функция $R_{xx}(\tau)$ непрерывного сигнала $x(t)$ определяется выражением

$$R_{xx}(\tau) = \int_{-\infty}^{\infty} x^*(t)x(t+\tau)dt.$$

В случае дискретной реализации для одномерной входной последовательности элементы выходной последовательности **Y** вычисляются в соответствии с выражением $y_j = \sum_{k=0}^{n-1} x_k^* x_{j+k}$,

где n – длина входной последовательности, $j = \overline{-(n-1),\ n-1}^*$.

При этом предполагается, что элементы входной последовательности $x_k = 0$ при $k < 0$ и $k \geq n$.

Вход **нормализация** (normalization) определяет способ нормализации, используемый для расчета автокорреляционной функции **X**. Варианты включают способ **без нормализации** (по умолчанию), с **несмещенной** (unbiased) и **смещенной** (biased) нормализациями.

При **смещенной** нормализации расчет автокорреляционной функции производится в соответствии с выражением

$y_j = \frac{1}{N}\sum_{k=0}^{n-1} x_k^* x_{j+k}$ для $j = \overline{-(n-1),\ n-1}$ и $R_{xx}(смещ)_i = y_{i-(N-1)}$ для $j = \overline{0,\ 2n-2}$.

При **несмещенной** нормализации расчетное выражение изменяется следующим образом:

$y_j = \frac{1}{N-|j|}\sum_{k=0}^{n-1} x_k^* x_{j+k}$ для $j = \overline{-(n-1),\ n-1}$ и $R_{xx}(несмещ)_i = y_{i-(N-1)}$ для $j = \overline{0,\ 2n-2}$.

В связи с тем, что в LabVIEW индексы элементов массива не могут быть отрицательными, элементы выходной последовательности R_{xx} соотносятся с элементами последовательности **Y** как $r_{xxi} = y_{i-(n-1)}$ для $i = \overline{0,2n-2}$. Таким образом, элемент выходной последовательности, соответствующий нулевому сдвигу, имеет индекс n. Для получения графика автокорреляционной функции с центром в нулевой точке целесообразно воспользоваться блок-диаграммой, приведенной на рис. 4.5. Результат выполнения ВП приведен на рис. 4.6.

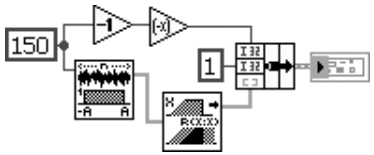


Рис. 4.5. Блок-диаграмма ВП для расчета автокорреляционной функции с центром в нулевой точке

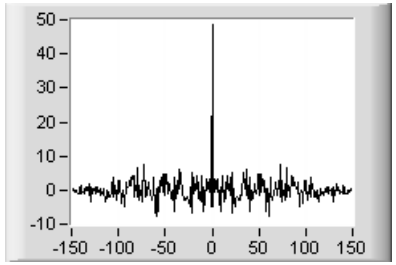


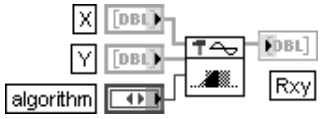
Рис. 4.6. График автокорреляционной функции с центром в нулевой точке

В случае двумерного входного массива расчетное выражение для автокорреляции выглядит следующим образом:

$$y(i,j) = \sum_{m=0}^{M_1}\sum_{n=0}^{N_1} x^*(m,n)x(i+m,j+n),$$

где $i = -(M-1),\ldots -1,0,1,\ldots(M-1)$ и $j = -(N-1),\ldots -1,0,1,\ldots(N-1)$, M, N – число строк и столбцов матрицы X

CrossCorrelation **Взаимная корреляционная функция**



Этот полиморфный ВП вычисляет взаимную корреляционную функцию одномерных или двумерных входных последовательностей **X** и **Y**, имеющих вещественные или комплексные числа.

Взаимная корреляционная функция $R_{xy}(\tau)$ сигналов $x(t)$ и $y(t)$ определяется выражением

$$R_{xy}(\tau) = \int_{-\infty}^{\infty} x^*(t)y(t+\tau)dt.$$

При дискретной реализации элементы выходной последовательности h_j вычисляются следующим образом: $h_j = \sum_{k=0}^{n-1} x_k^* y_{j+k}$, где n и m – длина одномерных входных последовательностей **X** и **Y** соответственно, $j = \overline{-(n-1),\ m-1}$.

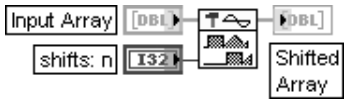
При этом предполагается, что элементы входной последовательности $x_k = 0$ при $k < 0$ и $k \geq n$, а элементы последовательности $y_k = 0$ при $k < 0$ и $k \geq m$.

В случае двумерных входных массивов расчетное выражение для взаимной корреляционной функции выглядит следующим образом:

$$h(i,j) = \sum_{m=0}^{M_1-1}\sum_{n=0}^{N_1-1} x^*(m,n)y(i+m,j+n),$$

где $i = -(M_1-1),\ldots -1,0,1,\ldots(M_2-1)$ и $j = -(N_1-1),\ldots -1,0,1,\ldots(N_2-1)$, M_1, N_1 – число строк и столбцов матрицы X, M_2, N_2 – число строк и столбцов матрицы Y

Y[i]=X[i-n] **Сдвиг Y[i]=X[i-n]**



ВП формирует на выходе последовательность в соответствии с выражением

$$y_i = \begin{cases} x_{i-shifts} & если\ \ 0 \leq i - shifts < n \\ 0 & иначе \end{cases},$$

где $i = 0, n-1$; n – количество элементов во входном массиве.

Если **сдвиг** (shifts) – положительное число, то первые **s** элементов выходного массива будут нулевыми (**s** = **сдвиг**). Следующие элементы будут равны элементам входного массива, начиная с элемента с индексом 0, а **s** последних элементов входного массива будут отброшены. Если **сдвиг** – отрицательное число, то отбрасываются первые элементы входного массива, а нулевыми будут последние элементы выходного.

Блок-диаграмма ВП **Y[i]=X[i-n]** приведена на рис. 4.7.

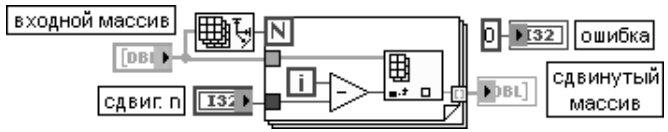
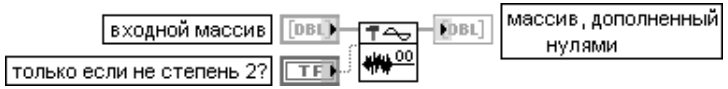


*Рис. 4.7. Блок-диаграмма ВП **Y[i]=X[i-n]***

Zero Padder

Дополнение нулями



ВП дополняет входной массив нулями так, чтобы длина выходного массива стала равна ближайшей большей степени числа 2. Если длина входного массива равна степени 2, то длина выходного массива увеличивается в два раза.

ВП целесообразно использовать при обработке данных с помощью быстрых алгоритмов (преобразования Фурье, Хартли) в тех случаях, когда длина массива данных не является степенью числа 2. Блок-диаграмма ВП приведена на рис. 4.8.

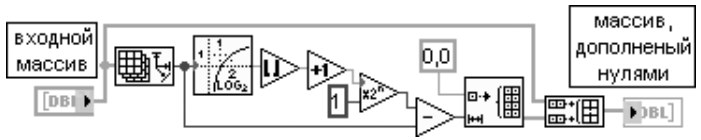


Рис. 4.8. Блок-диаграмма ВП ***Дополнение нулями*** *(Zero Padder)*

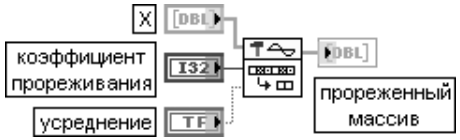
Unwrap Phase

Развертка фазы

ВП производит развертку массива значений фазы путем удаления разрывов, значения которых превышают π

Decimate (single shot)

Прореживание (однократное)



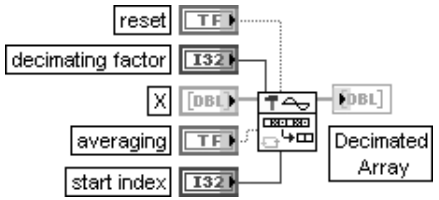
ВП формирует на выходе **прореженный массив** (Decimated Array) из входной последовательности **X** в зависимости от **коэффициента прореживания** (decimating factor) и состояния входа **усреднение** (averaging).

Элементы выходного массива y_i рассчитываются следующим образом:

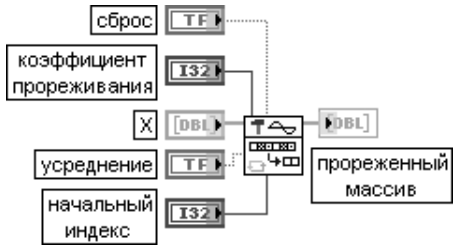
$$y_i = \begin{cases} x_{i*m} & \textit{если усреднение} = \textit{ЛОЖЬ} \\ \dfrac{1}{m}\displaystyle\sum_{k=0}^{m-1} x_{i*m+k} & \textit{если усреднение} = \textit{ИСТИНА} \end{cases},$$

где $i = \overline{0, size-1}$, $size = \left[\dfrac{n}{m}\right]$, n – число элементов входной последовательности X, m – коэффициент прореживания, $size$ – число элементов выходного массива. По умолчанию **коэффициент прореживания** равен 2, а состояние входа **усреднение** – ЛОЖЬ

Decimate (continuous)



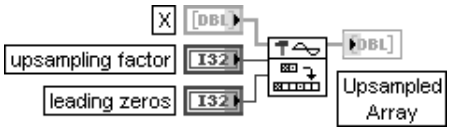
Прореживание (непрерывное)



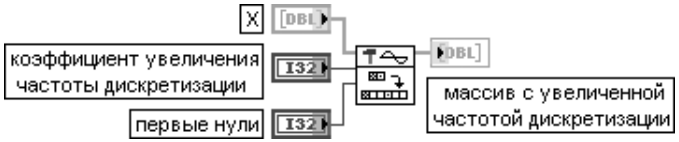
От предыдущего данный ВП отличается наличием входов **сброс** и **начальный индекс**. При установке на входе **сброс** состояния ИСТИНА или при первом выполнении ВП LabVIEW инициализирует прореживание по выборке, заданной на входе **начальный индекс**. При повторном срабатывании ВП с переводом входа сброс в состояние ЛОЖЬ LabVIEW инициализирует прореживание из финального состояния предыдущего вызова ВП.

Для обработки больших последовательностей данных, которые содержат небольшие блоки, необходимо установить сброс в состояние ИСТИНА для первого блока и в состояние ЛОЖЬ – для последующих

Upsample



Повысить частоту дискретизации



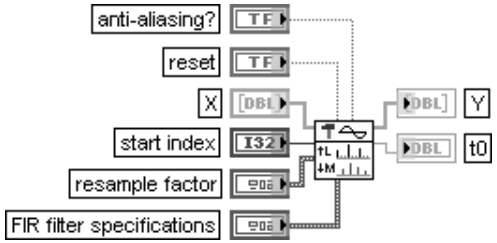
ВП вставляет нули во входную последовательность X в соответствии с **коэффициентом увеличения частоты дискретизации** (upsampling factor).

Если Y представляет выходную последовательность **массив с увеличенной частотой дискретизации** (Upsampled Array), то ВП получает элементы последовательности Y, используя следующее выражение:

$$y_i = \begin{cases} x_j \ \textit{если}\ i = j*m+k \\ 0 \quad \textit{иначе} \end{cases} \text{ для } i = 0,1,..size-1,\ size = n*m,$$

где n – число элементов в **X**, m – коэффициент увеличения частоты дискретизации, k – число **первых нулей** (leading zeros). Параметр $size$ равен числу элементов в выходной последовательности

Rational Resample



Изменение частоты дискретизации с помощью рационального коэффициента

Этот полиморфный ВП изменяет частоту дискретизации входного сигнала путем интерполяции ***X***, пропускания интерполированного сигнала через КИХ-фильтр и прореживания отфильтрованного сигнала. Варианты реализации ВП включают вещественные и комплексные числа, а также одно- и многоканальные данные.

Вход **защита от наложения спектра**? (anti-aliasing?) управляет низкочастотной фильтрацией входного сигнала при снижении частоты выборок. По умолчанию на входе установлено значение ИСТИНА. В этом случае ВП защищает сигнал с измененной частотой выборки от наложения спектра. При этом, однако, возрастают требования к вычислительным ресурсам.

Вход **сброс** (reset) управляет инициализацией внутренних состояний. По умолчанию на входе установлено значение ЛОЖЬ. При этом внутренние состояния устанавливаются в 0. При установке на входе **сброс** значения ИСТИНА внутренние состояния соответствуют последним состояниям из предыдущего вызова данного ВП. При обработке длинной последовательности, которая может быть разбита на меньшие блоки, целесообразно устанавливать данный вход в состояние ЛОЖЬ для первого блока и в состояние ИСТИНА при фильтрации остальных блоков.

Вход **X** содержит входной сигнал, у которого изменяется частота выборки. Интервал выборки **X** равен 1.

Вход **начальный индекс** определяет положение первой выборки выходного сигнала, выраженное числом интервалов, через которые вставляются нули.

Кластер **коэффициент изменения частоты дискретизации** (resample factor) содержит коэффициент **интерполяции** (interpolation) и коэффициент **прореживания** (decimation). По умолчанию значения этих коэффициентов равны 1.

Вход **параметры КИХ-фильтра** (FIR filter specifications) определяет минимум значений, которые требуются этому ВП для задания КИХ-фильтра. В этот кластер входят следующие параметры:

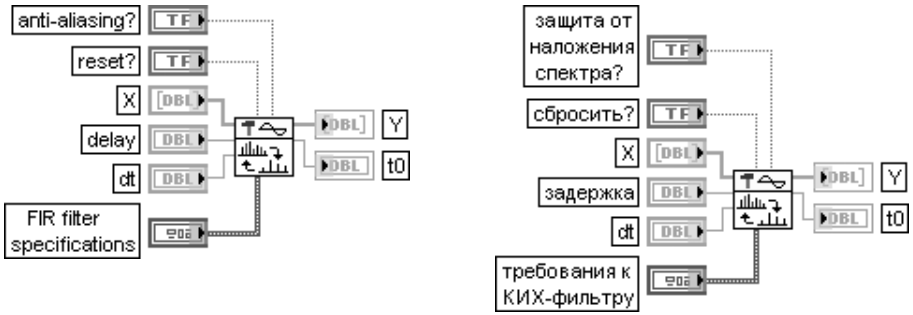
- **подавление наложения спектра (дБ)** (alias rejection (dB)) определяет минимальный уровень ослабления компонентов сигнала, вызывающих наложение спектра, после любой операции изменения частоты дискретизации;
- **нормализованная полоса пропускания** (normalized bandwidth) определяет полосу пропускания относительно новой частоты дискретизации, в которой не вносится ослабление.

Выход **Y** возвращает сигнал с измененной частотой дискретизации.

Выход **t0** возвращает время первой выборки на выходе **Y**

Resample (constant to constant)

Изменение частоты выборок (постоянную в постоянную)

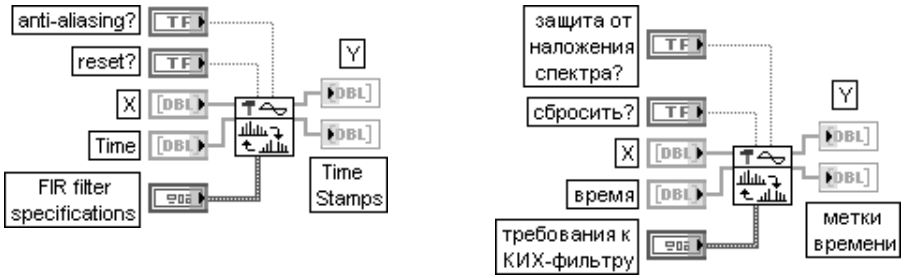


Полиморфный ВП изменяет частоту выборок входного сигнала **X** в соответствии с **задержкой** (delay) и **dt**, используя реализацию КИХ-фильтра.

Назначение большей части входов и выходов этого ВП было рассмотрено при описании предыдущего ВП. Отличие связано с входами **задержка** (delay) и **dt,** которые определяют соответственно отметку времени и интервал выборки для ***Y***

Resample (constant to variable)

Изменение частоты выборок (постоянную в переменную)

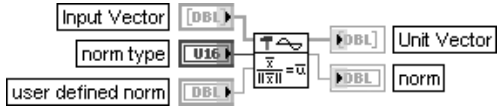


ВП изменяет частоту выборок входного сигнала **X** в соответствии со **временем** (Time), используя реализацию КИХ-фильтра.

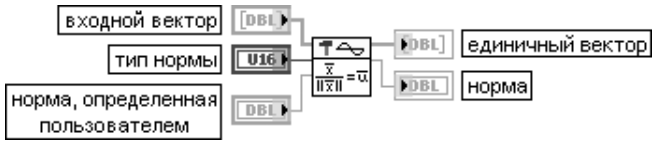
Вход **время** (Time) определяет отсчеты времени для изменения частоты дискретизации в порядке увеличения.

Выход **метки времени** (Time Stamps) возвращает временной отсчет для каждой выборки сигнала **Y** с измененной частотой дискретизации. Число элементов в массиве **метки времени** равно числу столбцов в **Y**. Содержимое входа **время** определяет содержимое выхода **метки времени**. Вследствие влияния внутреннего состояния фильтра передискретизации **метки времени** могут иметь дополнительные данные в начале массива и могут потерять часть данных в конце массива

Unit Vector



Единичный вектор



ВП находит **норму** (norm) **входного вектора** (Input Vector) и получает соответствующий ему **единичный вектор** (Unit Vector) с помощью деления исходного входного вектора на его норму.

Вход **тип нормы** (norm type) показывает, какой из типов нормы используется для вычисления нормы. По умолчанию это **2-norm**. Если в качестве **типа нормы** выбран **определяемая пользователем** (User Defined), то ВП использует в качестве типа нормы **норму, определяемую пользователем** (user defined norm).

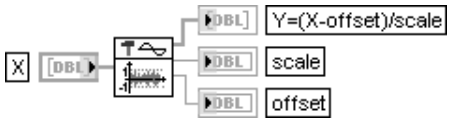
Этот ВП рассчитывает норму, используя следующие выражения:

1-norm	$\|\boldsymbol{X}\| = \|x_0\| + \|x_1\| + \ldots + \|x_{n-1}\|$
2-norm	$\|\boldsymbol{X}\| = \sqrt{(\|x_0\|^2 + \|x_1\|^2 + \ldots + \|x_{n-1}\|^2)}$
Inf-norm	$\|\boldsymbol{X}\| = \max_i(\|x_i\|)^*$
–Inf-norm	$\|\boldsymbol{X}\| = \min_i(\|x_i\|)$
Определяемая пользователем	$\|\boldsymbol{X}\| = \left\|\|x_0\|^y + \|x_1\|^y + \ldots + \|x_{n-1}\|^y\right\|^{1/y}$

где $\boldsymbol{X}$ – входной вектор, $\boldsymbol{y}$ – норма, определяемая пользователем, $||\boldsymbol{X}||$ – норма.

$$\boldsymbol{U} = \frac{\boldsymbol{X}}{\|\boldsymbol{X}\|}$$

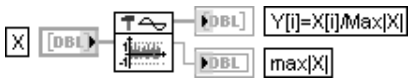
Scale



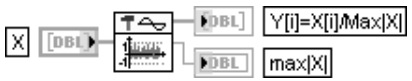
Масштабирование

ВП удаляет **смещение** (offset) из входного сигнала $\boldsymbol{X}$ и масштабирует результат так, чтобы выходная последовательность находилась в диапазоне [–1:1]

Quick Scale



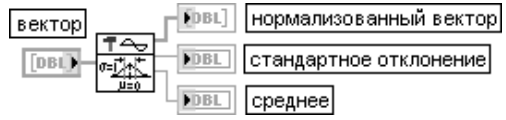
Быстрое масштабирование



ВП определяет максимальное абсолютное значение входного сигнала **X** и масштабирует **X**, используя это значение

Normalize

Нормировать



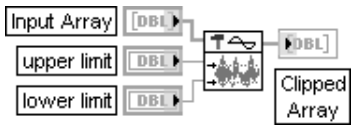
ВП нормирует входной **вектор** (Vector) или **матрицу** (Matrix), используя статистические параметры ($\boldsymbol{\mu}$,$\boldsymbol{s}$), где $\boldsymbol{\mu}$ представляет **среднее** (mean) и $\boldsymbol{s}$ представляет **стандартное отклонение** (standard deviation) вектора или матрицы, с целью получения **нормализованного вектора** (Normalized Vector), у которого эти параметры равны (0,1).

ВП рассчитывает нормализованный вектор или матрицу $\boldsymbol{Y}$, используя следующие выражения:

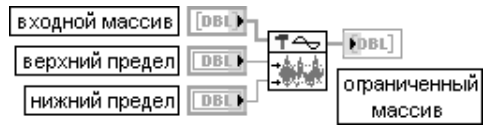
$$\boldsymbol{Y} = \frac{\boldsymbol{X} - \mu}{\sigma},$$

где $\mu = \dfrac{\sum_{i=0}^{n-1} x_i}{n}$, $\sigma = \sqrt{\dfrac{\sum_{i=0}^{n-1}(x_i - \mu)^2}{n}}$, $\boldsymbol{X}$ – входной вектор или матрица

Y[i]=Clip{X[i]}



Ограничение массива



ВП ограничивает элементы **входного массива** (Input Array) границами, заданными **верхним пределом** (upper limit) и **нижним пределом** (lower limit). Таким образом, элементы выходного ограниченного массива будут связаны с элементами входного массива соотношением: $\boldsymbol{y_i} = \begin{cases} \boldsymbol{a} & \boldsymbol{x_i} > \boldsymbol{a} \\ \boldsymbol{x_i} & \boldsymbol{b} \le \boldsymbol{x_i} \le \boldsymbol{a} \\ \boldsymbol{b} & \boldsymbol{x_i} < \boldsymbol{b} \end{cases}$, где $\boldsymbol{i} = \overline{\boldsymbol{0,n-1}}$, $\boldsymbol{a}$ и $\boldsymbol{b}$ – значения, подаваемые на входы **верхний предел** и **нижний предел**, $b \le a$.

По умолчанию верхний предел равен 1,0, а нижний – 0,0. Ниже на рис. 4.9 приведена блок-диаграмма данного ВП. Задачу ограничения сигнала можно решить и с помощью функции **Нахождение в диапазоне и ограничение** (In Range and Coerce). Блок-диаграмма соответствующего ВП приведена на рис. 4.10

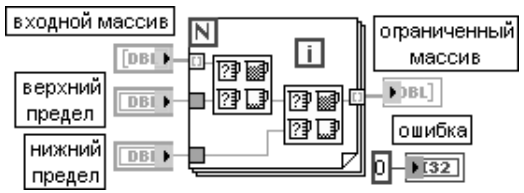


Рис. 4.9. Блок-диаграмма ВП ***Ограничение массива***

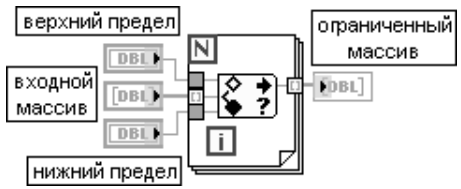
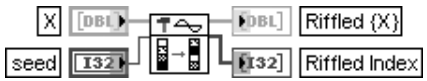


Рис. 4.10. Блок-диаграмма ВП на основе функции ***Нахождение в диапазоне и ограничение*** *(In Range and Coerce)*

Riffle

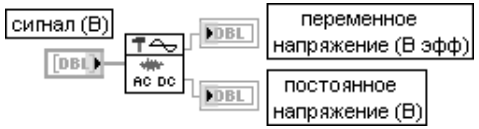


Перемешать

ВП перемешивает входной массив путем случайного выбора и перестановки двух элементов последовательности ***X*** и повторения этого процесса ***n*** раз, где ***n*** – длина последовательности ***X***

AC & DC Estimator

Оценка переменного и постоянного напряжения сигнала



ВП рассчитывает оценки переменной и постоянной составляющих напряжения входного сигнала.

В общем случае для получения оценок переменной и постоянной составляющих должен производиться спектральный анализ сигнала и выделение постоянной составляющей, расположенной на нулевой частоте, и переменной составляющей, определяемой путем среднеквадратичного суммирования значений на всех остальных частотах.

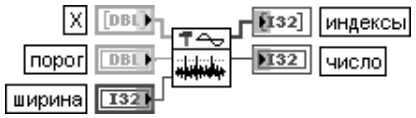
Для устойчивой оценки значений входной сигнал должен содержать как минимум три цикла переменного напряжения.

В данном ВП (рис. 4.11) для оценки постоянного и переменного напряжений используется более эффективный расчет среднего и среднеквадратичного отклонения сигнала, обработанного окном Ханна. Для компенсации влияния окна на получаемые оценки используются коэффициенты **cg** (coherent gain) и **enbw** (equivalent noise bandwidth).

Рис. 4.11. Блок-диаграмма ВП ***Оценка переменного и постоянного напряжения сигнала*** *(AC & DC Estimator)*

Threshold Peak Detector

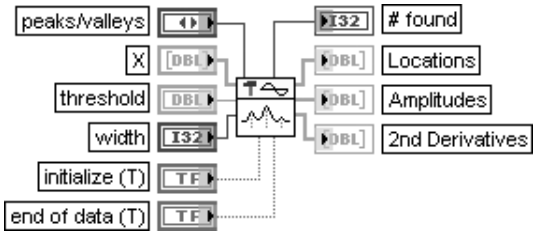
Пороговый детектор пиков



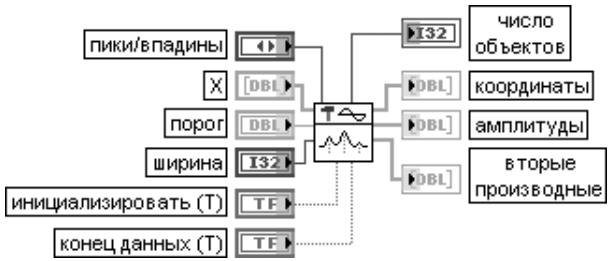
ВП анализирует входную последовательность **X** с целью определения **индексов** (Indices) и **числа** (count) пиков.

В качестве пика принимается участок последовательности, ширина которого, выраженная количеством элементов, не меньше значения, заданного на входе **ширина** (width), а уровень – не меньше значения на входе **порог** (threshold). Значение **ширина** должно быть больше нуля. Количество элементов последовательности **X** должно превышать значение **ширина**. По умолчанию значение **порог** равно нулю, а значение **ширина** – 1

Peak Detector



Детектор пиков



ВП находит **координаты** (Locations), **амплитуды** (Amplitudes) и **вторые производные** (2nd Derivatives) **пиков** (peaks) или **впадин** (valleys) во входном сигнале **X**. Входной сигнал может быть представлен в виде одного массива или набора последовательных блоков данных.

Вход **порог** (threshold) определяет уровень селекции пиков по амплитуде. При этом с порогом сравниваются **сглаженные** (fitted) значения амплитуд пиков.

Вход **ширина** (width) определяет размер области, выраженный числом отсчетов, в которой производится параболическая аппроксимация отсчетов сигнала. Выбор значения ширины зависит от уровня шума. При высоком уровне шума необходимо увеличивать ширину для уменьшения вероятности регистрации ложных пиков. Наоборот, при низком уровне шума значение ширины должно быть уменьшено до минимума (но оставаться большим 2) с целью минимизации систематических погрешностей оценок амплитуды и положения пиков.

Вход **пики/впадины** (peaks/valleys) определяет вид объектов – пики (0) или впадины (1).

Вход **инициализировать** (initialize) определяет внутренние установки ВП при обработке первого блока данных. При обработке одного блока данных этот вход можно не подключать или установить на нем значение ИСТИНА. При обработке последовательно поступающих блоков данных необходимо при обработке первого блока данных установить значение ИСТИНА, а для последующих – ЛОЖЬ.

Вход **конец данных** (end of data) определяет обработку последнего блока данных. При обработке одного блока данных этот вход можно не подключать или установить на нем значение ИСТИНА. При обработке последовательно поступающих блоков данных необходимо при обработке всех блоков, за исключением последнего, поддерживать на этом входе значение ЛОЖЬ.

Выход **число объектов** (**#** found) определяет количество найденных пиков или впадин и размер массивов **координаты** (Locations), **амплитуды** (Amplitudes) и **вторые производные** (2nd Derivatives).

Выходы **координаты** (Locations) и **амплитуды** (Amplitudes) содержат соответственно индексы (координаты) и амплитуды всех пиков/впадин, обнаруженных в текущем блоке данных. В связи с тем что для определения этих параметров используется алгоритм параболической аппроксимации, значения координат и амплитуд являются вещественными.
Значения на выходе **вторые производные** (2nd Derivatives) характеризуют «остроту» пиков или впадин. Они будут положительными для впадин и отрицательными для пиков

В состав палитры функций обработки сигналов во временной области входит Экспресс-ВП **Свертка и корреляция** (Convolution and Correlation). Этот Экспресс-ВП использует функциональность следующих ВП: **Автокорреляционная функция** (AutoCorrelation), **Свертка** (Convolution), **Взаимная корреляционная функция** (CrossCorrelation), **Деконволюция** (Deconvolution). В связи с тем что эти ВП были рассмотрены выше, их конфигурирование и выполнение в составе данного Экспресс-ВП далее не рассматриваются.

Свертка и корреляция (Convolution and Correlation)

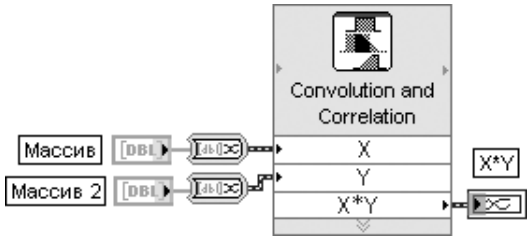


Рис. 4.12. Блок-диаграмма возможного подключения Экспресс-ВП

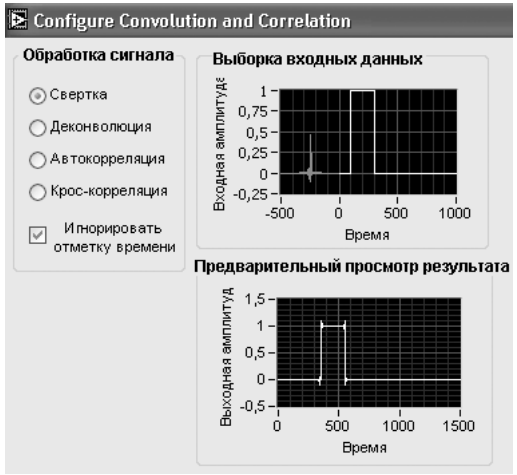


Рис. 4.13. Вид диалогового окна конфигурирования Экспресс-ВП ***Свертка и корреляция*** *(Convolution and Correlation)*

4.3. Функции преобразований сигналов

Палитра функций преобразований сигналов (рис. 4.14) содержит наборы функций, позволяющих выполнить прямое и обратное преобразования Фурье, Гильберта, Хартли и Уолша-Адамара, а также прямое и обратное вейвлет-преобразование. Методы преобразований сигналов, опирающиеся на хорошо разработанный математический аппарат преобразований Фурье [5–8], позволяют находить надежные оценки различных характеристик этих сигналов, а также характеристики цепей, через которые такие сигналы пропускаются. Реализация подобных методов в LabVIEW представлена рядом приборов высокого уровня для оценки взаимного спектра мощности, импульсной и частотной передаточной характеристик цепей, частотной функции когерентности и измерителя гармонических искажений сигнала (рис. 4.14).

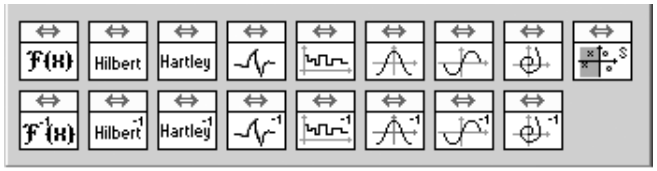


Рис. 4.14. Вид палитры функций преобразований сигналов

Связь между представлением непрерывных сигналов во временной и частотной областях устанавливает **интегральное преобразование Фурье**. Переход в частотную область представления сигнала $x(t)$ осуществляется с помощью **прямого** преобразования Фурье:

$$X(f) = F\{x(t)\} = \int_{-\infty}^{\infty} x(t) e^{-j2\pi ft} dt.$$

При этом функция $X(f)$ описывает распределение интенсивности сигнала по частоте – спектральную плотность сигнала.

Переход от частотного представления сигнала $X(f)$ к временному осуществляется с помощью **обратного** преобразования Фурье:

$$x(t) = F^{-1}\{X(f)\} = \int_{-\infty}^{\infty} X(f) e^{j2\pi ft} df.$$

Для указания того, что $x(t)$ и $X(f)$ являются парой преобразования Фурье, используется запись

$$x(t) \Leftrightarrow X(f).$$

Для дискретных (по времени) сигналов вместо интегрального применяют **дискретное** преобразование Фурье (**ДПФ** или **DFT**) [7].

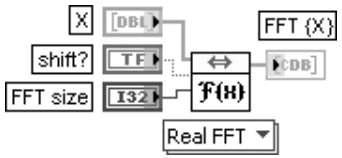
$$X_k = \sum_{i=0}^{n-1} x_i e^{-j2\pi ik/n}, \text{ для } k = \overline{0, n-1}; \qquad (4.1)$$

$$x_i = \frac{1}{n}\sum_{i=0}^{n-1} X_k e^{j2\pi ik/n}, \text{ для } i = \overline{0, n-1}; \qquad (4.2)$$

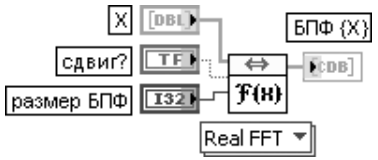
Вычисление ДПФ по формулам (4.1) и (4.2) называется **прямым методом** и в случае комплексной n-точечной последовательности требует примерно n^2 комплексных операций. В то же время разработан целый ряд алгоритмов быстрого вычисления ДПФ, получивших название **быстрого** преобразования Фурье (**БПФ**, или **FFT**). Достоинством БПФ является значительное сокращение количества арифметических операций. Так, в частности, если количество отсчетов равно степени 2, то количество арифметических операций равно примерно $n\log_2(n)$. При произвольной длине выборки в некоторых случаях целесообразно входную последовательность дополнить нулями, чтобы количество отсчетов стало равным степени 2.

Набор функций преобразований сигналов рассмотрен ниже в таблице.

FFT



Быстро преобразование Фурье

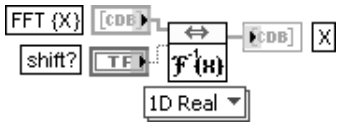


Полиморфный ВП вычисляет быстрое преобразование Фурье одномерной или двумерной входной последовательности **X**, которая также может быть вещественой или комплексной. Вариант реализации ВП определяется размерностью и типом входных данных. Он также может быть установлен помощью **селектора полиморфного ВП** (Polimorphic VI Selector), вызываемого в строке контекстного меню **видимые элементы** (Visible Items).
Вход **сдвиг?** (shift?) управляет выполнением сдвига постоянной составляющей в центр выходной последовательности **БПФ {X}** (FFT {X}). По умолчанию на входе установлено состояние ЛОЖЬ.
Вход **размер БПФ** (FFT size) определяет длину выполняемого БПФ. Если **размер БПФ** больше числа элементов в **X**, то в этом ВП к концу **X** добавляются нули для соответствия величине **размер БПФ**. Если же **размер БПФ** меньше числа элементов в **X**, то для выполнения БПФ в этом ВП используется только первая часть последовательности **X**, размер которой равен **размеру БПФ**. Если **размер БПФ** меньше или равен 0, этот ВП использует длину **X** в качестве **размера БПФ**.
При выполнении БПФ двумерной входной последовательности ВП имеет два входа для определения количества строк и столбцов преобразуемой последовательности.
Выходная последовательность **БПФ {X}** (FFT {X}) является комплексной.
Элементы комплексного спектра Y_{n-i} в соответствии со свойствами ДПФ могут быть представлены как элементы, находящиеся на отрицательных частотах Y_{-i}. Размещение составляющих комплексного спектра на относительных частотах в этом случае приведено в таблице. При четном n максимальное значение относительной частоты (частоты Найквиста) равно $k = n/2$, при нечетном n $k = (n-1)/2$. Переход к реальной частоте осуществляется путем умножения i на df, где $df = 1/(n*dt)$, dt – интервал дискретизации.

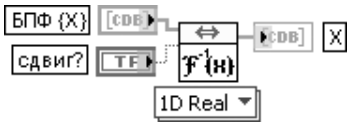
Элемент комплексного спектра	Относительная частота
Y_0	Постоянная составляющая
Y_1	1 гармоника (основная частота)
Y_2	2 гармоника

Элемент комплексного спектра	Относительная частота
Y_3	3 гармоника
…	…
Y_{k-2}	(k – 2)гармоника
Y_{k-1}	(k – 1)гармоника
Y_k	Частота Найквиста
$Y_{k+1} = Y_{n-(k-1)} = Y_{-(k-1)}$	–(k – 1) гармоника
$Y_{k+2} = Y_{n-(k-2)} = Y_{-(k-2)}$	–(k – 2) гармоника
…	…
Y_{n-3}	–3 гармоника
Y_{n-2}	–2 гармоника
Y_{n-1}	– 1 гармоника

Inverse FFT



Обратное БПФ



Полиморфный ВП вычисляет обратное преобразование Фурье комплексной входной последовательности **БПФ {X}** (FFT {X}), исходя из размерности и типа входных данных или настройки ВП

Fast Hilbert Transform

Быстрое преобразование Гильберта

ВП вычисляет быстрое преобразование Гильберта входной последовательности X.
Интегральное преобразование Гильберта непрерывной функции $x(t)$ LabVIEW определяется следующим образом:

$$h(t) = H\{x(t)\} = -\frac{1}{\pi}\int_{-\infty}^{\infty}\frac{x(\tau)}{t-\tau}d\tau.$$

Используя свойства преобразования Фурье, можно показать, что преобразование Фурье и преобразование Гильберта связаны следующим соотношением:

$$h(t) \Leftrightarrow H(f) = -j\,\mathrm{sgn}(f)X(f),$$

где $X(f)$ – преобразование Фурье $x(t)$, а $\mathrm{sgn}(f) = \begin{cases} 1 & f > 0 \\ 0 & f = 0 \\ -1 & f < 0 \end{cases}$.

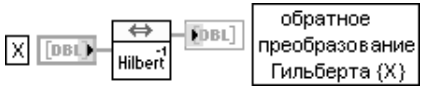
ВП выполняет дискретную реализацию преобразования Гильберта с помощью процедур БПФ, основываясь на соответствующей взаимосвязи $h(t) \Leftrightarrow H(f)$. При этом реализуются следующие шаги:

- выполняется преобразование Фурье входной последовательности X, $Y = F\{X\}$;
- постоянная составляющая приравнивается нулю, $Y_0 = 0$;
- если длина входной последовательности четна, то приравнивается нулю компонента на частоте Найквиста, $Y_{Nyq} = 0$;
- положительные гармоники в преобразовании Фурье умножаются на $-j$, а отрицательные – на j. Формируется новая последовательность $H_k = -j\,sgn(k)Y_k$;
- выполняется обратное преобразование Фурье последовательности H.

Преобразование Гильберта целесообразно использовать для решения таких задач, как извлечение информации о мгновенном значении фазы или огибающей сигнала, получения одностороннего спектра, детектирования эхо-сигнала и уменьшения частоты выборок.
Преобразование Гильберта хорошо работает с сигналами, имеющими ограниченный спектр

Inverse Fast Hilbert Transform

Обратное быстрое преобразование Гильберта



ВП вычисляет обратное быстрое преобразование Гильберта входной последовательности ***X***, используя свойства преобразования Фурье.
Обратное преобразование Гильберта функции $h(t)$ в LabVIEW определяется следующим образом:

$$x(t) = H\{h(t)\} = \frac{1}{\pi}\int_{-\infty}^{\infty}\frac{h(\tau)}{t-\tau}d\tau.$$

Используя определение преобразования Гильберта, данное выше при рассмотрении ВП **Быстрое преобразование Гильберта**, можно сделать вывод, что обратное преобразование Гильберта может быть получено из прямого путем изменения его знака
$x(t) = H^{-1}\{h(t)\} = -H\{h(t)\}$.
ВП выполняет дискретную реализацию обратного преобразования Гильберта с помощью прямого преобразования Гильберта, реализуя следующие шаги:
1. Преобразования Гильберта входной последовательности ***X***: $Y = H\{X\}$.
2. Получения обратного преобразования Гильберта как отрицательного значения *Y*: $H^{-1}\{X\} = -Y$

Fast Hartley Transform (FHT)

Быстрое преобразование Хартли

Интегральное преобразование Хартли непрерывной функции $x(t)$ определяется следующим образом: $X(f) = \int_{-\infty}^{\infty} x(t)\,cas(2\pi f t)\,dt$, где $\mathbf{cas(x) = cos(x) + sin(x)}$.

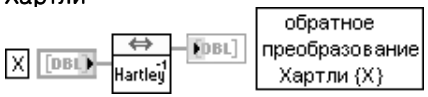
Для последовательности X_i дискретное преобразование Хартли $Y = Hartley\{X\}$ будет иметь вид

$$Y_k = \sum_{i=0}^{n-1} X_i\, cas\left(\frac{2\pi i k}{n}\right),$$ где $k = \overline{1, n-1}$, n – число элементов последовательности X_i.

Преобразование Хартли отображает действительную последовательность значений во временной области в такую же последовательность в частотной области. В отличии от этого преобразование Фурье отображает действительную последовательность в комплексную последовательность в частотной области, в которой половина данных является избыточной. Недостатком FHT является то, что размер входной последовательности должен быть равен степени числа 2

Inverse FHT

Обратное быстрое преобразование Хартли



Обратное преобразование Хартли функции $X(f)$ определяется следующим образом:

$x(t) = \int_{-\infty}^{\infty} X(f)\,cas(2\pi f t)\,df$, где $\mathbf{cas(x) = cos(x) + sin(x)}$.

При этом число элементов входной последовательности **X** должно быть кратным степени 2.
Для последовательности Y_k, представляющей выходную последовательность $Inv\ FHT\{X\}$, ВП будет вычислять значения следующим образом:

$$Y_k = \frac{1}{n}\sum_{i=0}^{n-1} X_i\, cas\left(\frac{2\pi i k}{n}\right),$$

где $k = \overline{1, n-1}$, n – число элементов последовательности X_i.
Обратное преобразование Хартли отображает действительную последовательность значений в частотной области в такую же последовательность во временной области

Wavelet Transform Daubechies4

Вейвлет-преобразование Добеши4

ВП выполняет вейвлет-преобразование входной последовательности ***X***, базирующееся на функции Добеши4. Длина последовательности должна быть кратной степени 2.
Вейвлет-преобразование Добеши4 определяется соотношением
$Wavelet\ Daubechies4\{X\} = C * X$,

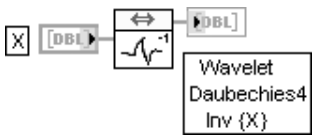
где матрица преобразования C равна $C =$

$$\begin{bmatrix}
c_0 & c_1 & c_2 & c_3 & & & & & & \\
c_3 & -c_2 & c_1 & -c_0 & & & & & & \\
 & & c_0 & c_1 & c_2 & c_3 & & & & \\
 & & c_3 & -c_2 & c_1 & -c_0 & & & & \\
 & & & \cdot & & & & & & \\
 & & & \cdot & & & & & & \\
 & & & & & & c_0 & c_1 & c_2 & c_3 \\
 & & & & & & c_3 & -c_2 & c_1 & -c_0 \\
c_2 & c_3 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & c_0 & c_1 \\
c_1 & -c_0 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & c_3 & -c_2
\end{bmatrix}$$

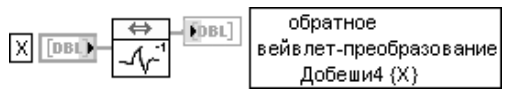
Пробелы матрицы заполнены нулями. Коэффициенты c_0, c_1, c_2, c_3 связаны определенными ортогональными свойствами:
$c_0^2 + c_1^2 + c_2^2 + c_3^2 = 1$; $c_2c_0 + c_3c_1 = 0$; $c_3 - c_2 + c_1 - c_0 = 0$; $0c_3 - 1c_2 + 2c_1 - 3c_0 = 0$,
дающими единственное решение:

$$c_0 = \frac{1+\sqrt{3}}{4\sqrt{2}} \quad c_1 = \frac{3+\sqrt{3}}{4\sqrt{2}} \quad c_2 = \frac{3-\sqrt{3}}{4\sqrt{2}} \quad c_3 = \frac{1-\sqrt{3}}{4\sqrt{2}}$$

Wavelet Transform Daubechies4 Inverse



Обратное вейвлет-преобразование Добеши4



ВП выполняет обратное вейвлет-преобразование входной последовательности ***X***, базирующееся на функции Добеши4. Длина последовательности должна быть кратной степени 2.

Обратное вейвлет-преобразование Добеши4 определяется соотношением

Wavelet Daubechies4 Inv{X} = C⁻¹ * X,

где матрица преобразования ***C*** равна

$$C = \begin{bmatrix}
c_0 & c_3 & & & & & & & c_2 & c_1 \\
c_1 & -c_2 & & & & & & & c_3 & -c_0 \\
c_2 & c_1 & c_0 & c_3 & & & & & & \\
c3 & -c0 & c_1 & -c_2 & & & & & & \\
 & & & \cdot & & & & & & \\
 & & & \cdot & & & & & & \\
 & & & & c_2 & c_1 & c_0 & c_3 & & \\
 & & & & c_3 & -c_0 & c_1 & -c_2 & & \\
 & & \cdot & \cdot & \cdot & \cdot & c_2 & c_1 & c_0 & c_3 \\
 & & \cdot & \cdot & \cdot & \cdot & c_3 & -c_0 & c_1 & -c_2
\end{bmatrix}$$

Коэффициенты c_0, c_1, c_2, c_3 имеют те же значения, что и в ВП **Вейвлет-преобразование Добеши4** (Wavelet Daubechies4)

Walsh Hadamard

Преобразование Уолша-Адамара

ВП выполняет преобразование Уолша-Адамара входной последовательности ***X***. Преобразование Уолша-Адамара аналогично по свойствам преобразованию Фурье, но требует меньших вычислительных затрат.

Преобразование Уолша-Адамара базируется на системе ортогональных функций, имеющих только два значения: –1 и +1. Например, для случая $n = 4$ преобразование Уолша-Адамара входной последовательности $X = \{x_0, x_1, x_2, x_3\}$ может быть представлено в матричной форме следующим образом:

$$WH\{X\} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix}.$$

Если WH_n и WH_{n+1} представляют матрицы преобразования Уолша-Адамара размером 2^n и 2^{n+1} соответственно, между ними имеется следующее соотношение:

$$WH_{n+1} = \begin{bmatrix} WH_n & WH_n \\ WH_n & -WH_n \end{bmatrix}$$

Walsh Hadamard Inverse

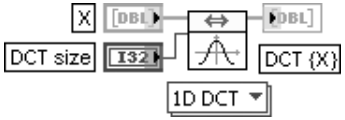
Обратное преобразование Уолша-Адамара

ВП выполняет обратное преобразование Уолша-Адамара входной последовательности X. Обратное преобразование Уолша-Адамара описывается следующим выражением:

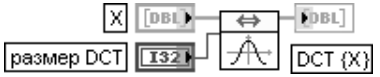
$$WHI\{X\} = \frac{1}{n} WH\{X\},$$

где n – длина входной последовательности

DCT



Дискретное косинусное преобразование



Этот полиморфный ВП вычисляет **дискретное косинусное преобразование** (DCT) входной последовательности ***X***.

Вход **размер DCT** (DCT size) определяет длину выполняемого DCT-преобразования. Если **размер DCT** больше числа элементов ***X***, ВП обеспечивает совпадение размеров путем добавления нулей к концу последовательности ***X***. Если же **размер DCT** меньше числа элементов ***X***, ВП использует для расчета столько же первых элементов ***X***. При отрицательном или равном нулю значении **размера DCT** ВП использует длину X в качестве этого параметра.

Одномерное DCT определяется следующим выражением:

$$y_k = \sqrt{\frac{2}{N}} \alpha_k \sum_{n=0}^{N-1} x_n \cos\frac{(2n+1)k\pi}{2N},$$

где $\alpha_k = \begin{cases} \frac{1}{\sqrt{2}} & k = 0 \\ 1 & k = 1, N-1 \end{cases}$, N – длина ***X***.

При реализации DCT вместо прямого вычисления приведенного выражения ВП использует алгоритм быстрого DCT, который опирается на быстрое преобразование Фурье.

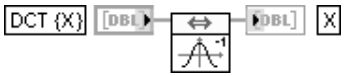
Аналогичный алгоритм для двумерного DCT выглядит следующим образом:

$$y(u,v) = \sqrt{\frac{2}{M}}\sqrt{\frac{2}{N}} \alpha_u \alpha_v \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} x(m,n) \cos\frac{(2m+1)u\pi}{2M} \cos\frac{(2n+1)v\pi}{2N},$$

где M и N – число строк и столбцов входной матрицы ***X***.

ВП выполняет двумерное DCT в два этапа: сначала рассчитывается одномерное DCT для каждой строки, а затем – одномерное DCT по столбцам промежуточного результата

Inverse DCT

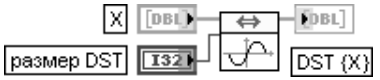


Обратное дискретное косинусное преобразование

ВП выполняет **обратное дискретное косинусное преобразование** входной последовательности **DCT {X}**. Расчетные выражения идентичны приведенным выше с соответствующей заменой переменных

DST

Дискретное синусное преобразование



Полиморфный ВП производит расчет **дискретного синусного преобразования** (DST) входной последовательности ***X***. Расчетные выражения также идентичны приведенным выше с заменой функции косинуса на синус

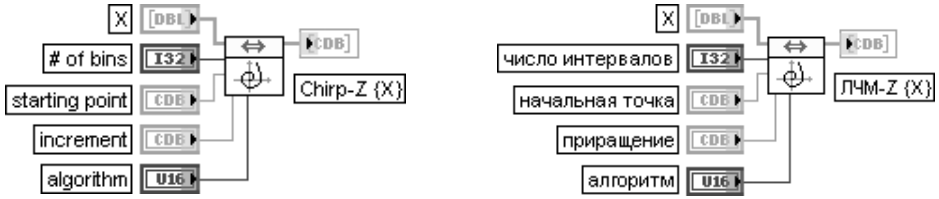
Inverse DST

Обратное дискретное синусное преобразование

ВП выполняет **обратное дискретное косинусное преобразование** входной последовательности **DST {X}**

Chirp Z Transform

Z-преобразование по спиральному контуру



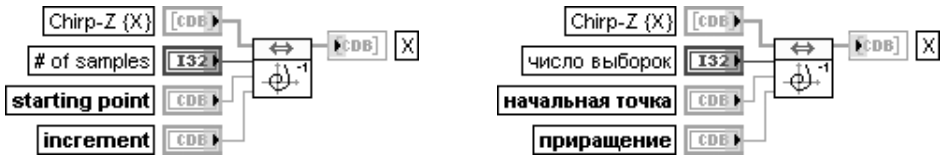
ВП вычисляет Z-преобразование по спиральному контуру входной последовательности ***X***.
Вход **число интервалов** (# of bins) определяет длину Chirp-Z {X}. Если **число интервалов** меньше 0, ВП устанавливает величину этого параметра равной длине последовательности ***X***.
Вход **начальная точка** (starting point) устанавливает точку, с которой начинается вычисление Z-преобразования.
Вход **приращение** (increment) определяет приращение между точками, в которых оценивается Z-преобразование.
Вход **алгоритм** (algorithm) определяет используемый метод преобразования. При установке на этом входе значения **прямой** (direct) этот ВП рассчитывает Z-преобразование по спиральному контуру, используя прямой метод. При установке на входе **алгоритм** варианта **частотная область** (frequency domain) этот ВП рассчитывает Z-преобразование, используя технику БПФ. Прямой метод обладает преимуществом в скорости при малом размере ***X*** или **числа интервалов**. При большой величине этих параметров большую скорость обеспечивает метод **частотной области**.
Расчетное выражение при использовании прямого метода выглядит следующим образом:

$$Y_k = \sum_{i=0}^{N-1} x_i (AW^{-k})^{-i} \text{ для } k = \overline{0, M-1},$$

где M – число интервалов, A – начальная точка, W – приращение, N – длина X

Inverse Chirp Z Transform

Обратное Z-преобразование по спиральному контуру



ВП вычисляет обратное Z-преобразование по спиральному контуру входной последовательности Chirp-Z {X}

Laplace Transform Real

Действительное преобразование Лапласа

ВП осуществляет преобразование Лапласа действительного сигнала **X**.
Для непрерывного сигнала $x(t)$ преобразование Лапласа записывается следующим образом:

$$L\{X\}(s) = \int_0^\infty x(t)\exp(-st)dt,$$

где $s \geq 0$ и s – действительное, $x(t)$ определено для всех $t \geq 0$.
Дискретное преобразование Лапласа для равномерно расположенных отсчетов является аналогом непрерывного преобразования. Для повышения эффективности расчетов дискретного преобразования Лапласа используют дробное преобразование Фурье (Fractional FFT, FFFT), определяемое следующим образом:

$$FFFT\{X\}(t) = \int_0^\infty x(s)\exp(-i\alpha st)ds,$$

где α – произвольное комплексное число

4.4. Функции спектрального анализа

Палитра функций спектрального анализа представлена рядом приборов высокого уровня для оценки взаимного спектра мощности, импульсной и частотной передаточной характеристик цепей, частотной функции когерентности и измерителя гармонических искажений сигнала (рис. 4.15).

Набор функций спектрального анализа рассмотрен ниже в таблице.

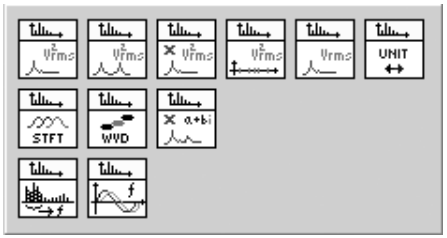


Рис. 4.15. Вид палитры функций спектрального анализа

Auto Power Spectrum

Односторонний спектр мощности

ВП вычисляет односторонний масштабированный спектр мощности действительной входной последовательности в соответствии с выражением

$$S_{xx} = \frac{1}{n^2}\left|F\{x(t)\}\right|^2,$$

где n – длина входной последовательности.
Запись во временной области должна содержать как минимум три периода сигнала для обеспечения достоверной оценки. В отличие от ВП **Спектр мощности** (Power Spectrum), входящего в его состав (рис. 4.16), диапазон частот ограничен положительными значениями, а величина спектральных составляющих увеличена в два раза.

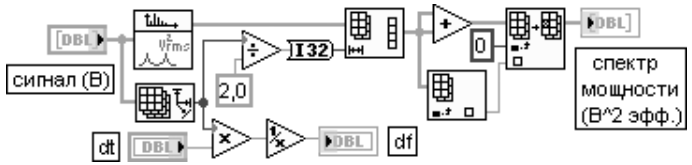
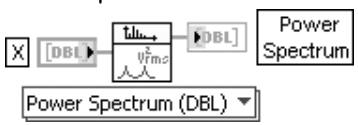


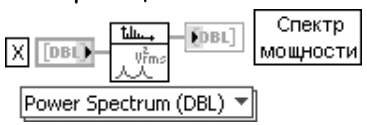
Рис. 4.16. Блок-диаграмма ВП ***Односторонний спектр мощности*** *(Auto Power Spectrum)*

Вход ***dt*** представляет период выборки временного сигнала, заданный обычно в секундах. ***dt*** определяется как $1/f_s$, где f_s – частота дискретизации временного сигнала.

Power Spectrum



Спектр мощности



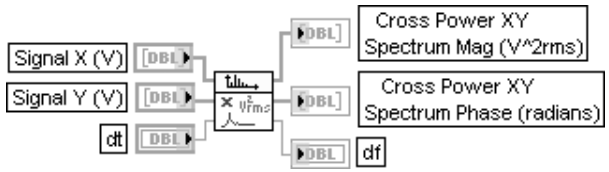
ВП вычисляет спектр мощности входной последовательности ***X*** действительных чисел в соответствии с выражением

$$S_{xx} = \frac{1}{n^2}X^*(f)X(f) = \frac{1}{n^2}\left|X(f)\right|^2 = \frac{1}{n^2}\left|F\{x(t)\}\right|^2,$$

где n – длина входной последовательности, $X^*(f)$ – комплексно-сопряженный спектр входной последовательности.
Распределение мощности по гармоникам приведено в таблице.

Элемент массива	Относительная частота
S_{xx0}	Мощность постоянной составляющей
$S_{xx1} = S_{xx(n-1)}$	Мощность 1-й гармоники (основная частота)
$S_{xx2} = S_{xx(n-2)}$	Мощность 2-й гармоники
$S_{xx3} = S_{xx(n-3)}$	Мощность 3-й гармоники
...	...
$S_{xx(k-2)} = S_{xx(n-(k-2))}$	Мощность (k – 2)-й гармоники
$S_{xx(k-1)} = S_{xx(n-(k-1))}$	Мощность (k – 1)-й гармоники
$S_{xxk} = S_{xx(n-k)}$	Мощность k-й гармоники на частоте Найквиста (n нечетно)
S_{xxk}	Мощность k-й гармоники на частоте Найквиста (n четно)

Cross Power Spectrum



Односторонний взаимный спектр мощности

ВП рассчитывает односторонний нормированный взаимный спектр двух действительных сигналов **сигнал *X*** (Signal X) и **сигнал *Y*** (Signal Y).
Выход **величина взаимного спектра мощности XY** (Cross Power XY Spectrum Mag) отображает величину одностороннего взаимного спектра мощности сигналов ***X*** и ***Y*** и имеет размерность [$В^2$ эфф.], если входные сигналы имеют размерность [В].
Выход **фаза взаимного спектра мощности XY** (Cross Power XY Spectrum Phase) показывает разность фаз частотных составляющих сигналов ***Y*** и ***X*** и имеет размерность [радианы].
Как видно из блок-диаграммы данного ВП (рис. 4.17), его основным элементом является рассмотренный ниже ВП **Взаимный спектр мощности** (Cross Power).

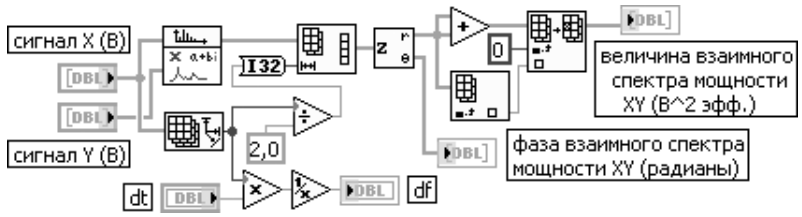
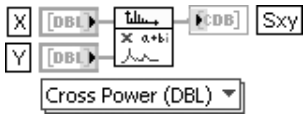
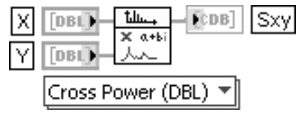


Рис. 4.17. Блок-диаграмма ВП
Односторонний взаимный спектр мощности *(Cross Power Spectrum)*

Cross Power



Взаимный спектр мощности



ВП вычисляет взаимный спектр мощности входных сигналов **X** и **Y** в соответствии с выражением

$$S_{xy} = \frac{1}{n^2}X^*(f)Y(f),$$

где n – длина входной последовательности, $X^*(f)$ – комплексно-сопряженный спектр.

Если входные последовательности имеют одинаковую длину и она равна степени 2, вычисление осуществляется через подпрограмму БПФ. При невыполнении данного условия производится дополнение более короткой последовательности нулями до выравнивания размера, затем обе последовательности дополняются нулями, пока их длина не станет равной степени 2, после чего вычисляется результат.

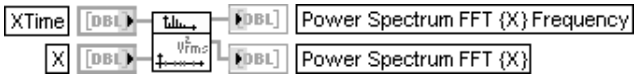
Размер N комплексной выходной последовательности s_{xy} равен

$$N = \begin{cases} \max(n,m) \text{ если } \max(n,m) = 2^k; \\ 2^i \quad \text{ если } \max(n,m) \neq 2^k; \end{cases} \text{ для } k = \overline{1,23},$$

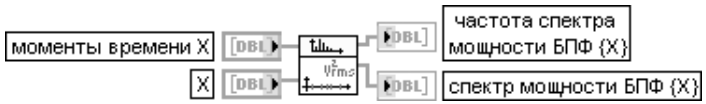
где n, m – длина входной последовательности X и Y соответственно,

$i = trunc(\log_2(\max(n,m))) + 1,\ 1 \le i \le 23$

Unevenly Sampled Signal Spectrum



Спектр сигнала с неравномерными отсчетами



ВП осуществляет расчет спектра мощности последовательности X с неравномерным по времени распределением выборок, задаваемым последовательностью **моменты времени X** (XTime). Выходы **частота спектра мощности БПФ {X}** (Power Spectrum FFT {X} Frequency) и **спектр мощности БПФ {X}** (Power Spectrum FFT {X}) представляют соответственно массивы частот и значений спектра мощности, рассчитанных с помощью нормализованной периодограммы Ломба.

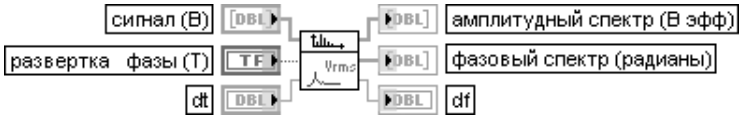
Для последовательности значений $X = \{x_0, x_1, .. x_{n-1}\}$, полученных в моменты времени $XTime = \{t_0, t_1, .. t_{n-1}\}$, нормализованная периодограмма Ломба рассчитывается следующим образом:

$$p(\omega) = \frac{1}{2\sigma^2}\left(\frac{\left[\sum_{k=0}^{n-1}(x_k - \bar{x})\cos\omega(t_k - \tau)\right]^2}{\sum_{k=0}^{n-1}\cos 2\omega(t_k - \tau)} + \frac{\left[\sum_{k=0}^{n-1}(x_k - \bar{x})\sin\omega(t_k - \tau)\right]^2}{\sum_{k=0}^{n-1}\sin 2\omega(t_k - \tau)}\right),$$

где $\tau = \frac{1}{2\sigma^2}\arctan\left(\frac{\sum_{k=0}^{n-1}\sin 2\omega t_k}{\sum_{k=0}^{n-1}\cos 2\omega t_k}\right)$, $\sigma^2 = \frac{1}{n-1}\sum_{k=0}^{n-1}(x_k - \bar{x})^2$, $\bar{x} = \frac{1}{n}\sum_{k=0}^{n-1}x_k$

Amplitude and Phase Spectrum

Амплитудный и фазовый спектр



ВП рассчитывает односторонний нормированный спектр входного **сигнала** (Signal) и представляет его в виде амплитудного и фазового спектра. Для надежной оценки сигнал должен содержать как минимум три периода.

При установке на входе **развертка фазы** (unwrap phase) значения ИСТИНА (по умолчанию) производится развертка фазы выходного **фазового спектра** (Amp Spectrum Phase). При установке значения ЛОЖЬ развертка фазы не производится.

Вход *dt* определяет интервал дискретизации входного сигнала, который, как правило, задается в секундах. Помимо этого, $dt = 1 / f_s$, где f_s – частота дискретизации.

Выход **амплитудный спектр** (Amp Spectrum Mag) отображает односторонний амплитудный спектр, имеющий размерность [В эфф.], если входной сигнал имеет размерность [В]. Значение **амплитудного спектра** рассчитывается следующим образом:

$$\text{амплитудный спектр} = \frac{|FFT\{сигнал\}|}{N}.$$

Выход **фазовый спектр** отображает односторонний фазовый спектр, выраженный в радианах. Значение **фазового спектра** рассчитывается в соответствии с выражением $фазовый\ спектр = фаза(FFT\{сигнал\})$.

Выход *df* представляет частотный интервал [Гц], если интервал *dt* задан в секундах.

Блок-диаграмма ВП **Амплитудный и фазовый спектр** приведена на рис. 4.18.

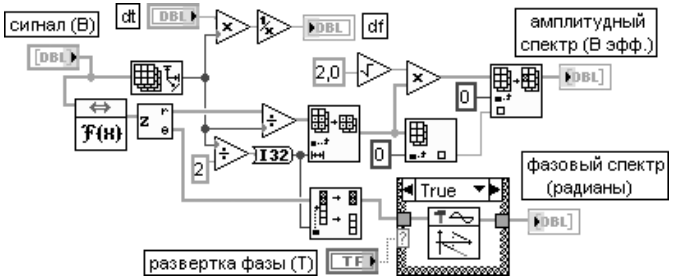
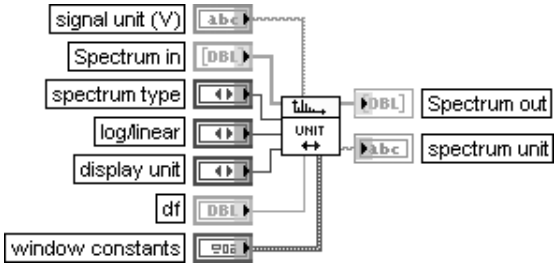
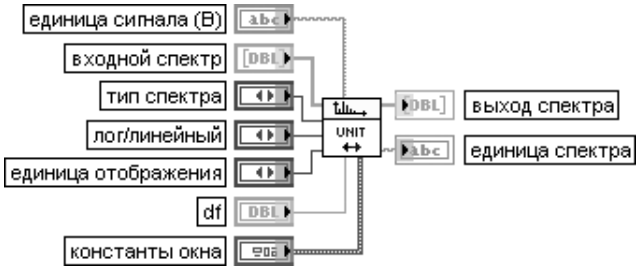


Рис. 4.18. Блок-диаграмма ВП ***Амплитудный и фазовый спектр*** *(Amplitude and Phase Spectrum)*

Spectrum Unit Conversion



Преобразование единиц спектра



ВП преобразует спектры мощности, амплитуды или коэффициента усиления (отношения амплитуд) в альтернативные форматы, включающие логарифмический (децибелы и децибелы мощности (дБм)) и спектральную плотность.

Вход **единица сигнала** (signal unit) является строкой, содержащей единицу входного сигнала, заданного во временной области. По умолчанию на этом входе установлен Вольт. Установка осуществляется с помощью записи буквы **V**.

Вход **входной спектр** (Spectrum in) представляет спектр входного сигнала. Он может иметь тип, задаваемый селектором **тип спектра** (spectrum type).

Вход **тип спектра** (spectrum type) определяет следующие типы спектров:

0 (по умолчанию)	1	2
Спектр мощности (В²эфф) (Power (Vrms^2))	**Амплитудный спектр (В эфф.)** (Amplitude (Vrms))	**Коэффициент усиления Gain** (amplitude ratio)

Вход **лог/линейный** (log/linear) определяет линейный или логарифмический вид выходного спектра.

0	1(по умолчанию)	2
Линейный	дБ	дБм

Вход **единица отображения** (display unit) определяет тип единиц для отображения спектра. Тип единиц в совокупности с номером типа приведен в таблице.

0	**В эфф.** (Vrms)	4	**В эфф./$\sqrt{Гц}$** (Vrms/sqrt(Hz))
1	**В пик** (Vpk)	5	**В пик/$\sqrt{Гц}$** (Vpk/sqrt(Hz))
2	**В² эфф.** (Vrms^2)	6	**В² эфф/Гц** (Vrms²/Hz)
3	**В² пик** (Vpk^2)	7	**В² пик/Гц** (Vpk²/Hz)

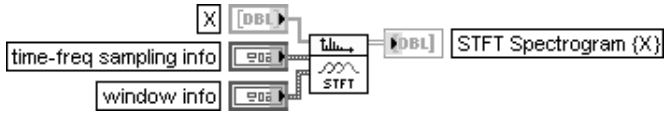
Последние четыре типа представляют спектральную плотность амплитуд (4, 5) и спектральную плотность мощности (6, 7). Выбор этих типов должен сопровождаться определением **констант окна** (window constants) и входа **df**.

Константы окна определяются в ВП **Масштабированное временное окно** (Scaled Time Domain Window). По умолчанию константы имеют значения, соответствующие равномерному окну (отсутствию окна).

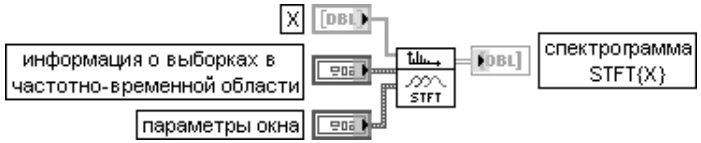
Выход спектра **(Spectrum out) содержит спектр в форме, определенной входами** лог/линейный **и** единица отображения**.**

Выход **единица спектра** (spectrum unit) представляет строку, содержащую единицу выходного спектра. Если выходной спектр выражен в децибелах, единице предшествует приставка **dB**

STFT Spectrogram



Спектрограмма STFT



ВП производит расчет распределения энергии в частотно-временной области, используя алгоритм «**локального**» **преобразования Фурье** (short-time Fourier transform (STFT)). При этом производится расчет текущего спектра в «скользящем» окне, размер которого определяется входом **ширина окна** (window length).

Вход **приращение времени** (time increment) определяет число выборок в «скользящем» окне. По умолчанию **приращение времени** равно 1. Увеличение параметра **приращение времени** уменьшает время расчетов и требования к памяти, но вместе с тем ухудшает и частотно-временное разрешение.

Вход **ширина окна** (window length) задает действительную ширину выбранного окна. По умолчанию значение параметра равно 1. **Ширина окна** определяет количество выборок, используемых для расчета преобразования Фурье.

Вход **селектор окна** (window selector) определяет тип окна, применяемого при расчете выходного двумерного массива **Спектрограмма STFT {X}**. Варианты окон приведены в таблице.

0	1	2	3	4
Прямоугольное (Rectangular)	**Блэкмана** (Blackman)	**Хэмминга** (Hamming)	**Ханна** (Hanning)	**Гауссовское** (Gaussian)

Выход **Спектрограмма STFT {X}** (STFT Spectrogram {X}) представляет двумерный массив, который описывает распределение энергии сигнала в частотно-временной области.

Число градаций спектра по времени (число строк) определяется следующим выражением:

$$\textit{число строк} = \frac{\textit{число элементов } X}{\textit{приращение времени}}.$$

Число градаций по частоте (число столбцов) рассчитывается следующим образом:

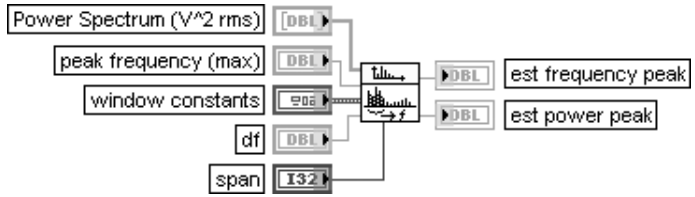
$$\textit{число столбцов} = \frac{\textit{число элементов } X}{2} + 1$$

WVD Spectrogram

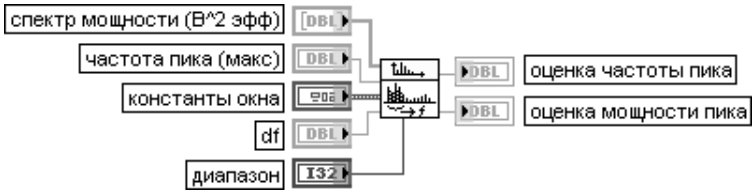
Спектрограмма WVD

ВП производит расчет распределения энергии в пространственно-временной области, используя алгоритм **Wigner-Ville**.
Вход **приращение времени** (time increment) (по умолчанию 1) задает временной интервал распределения **Wigner-Ville**. Так, например, если выборки имеют частоту f_s [Гц], то расстояние между строками спектрограммы **WVD** будет равно **приращение времени** / f_s секунд

Power & Frequency Estimate



Оценка мощности и частоты



ВП рассчитывает значения мощности и частоты в окрестности пика **спектра мощности** (Power Spectrum) сигнала.
На входе **частота пика** (peak frequency) задается частота пика спектра, обычно в герцах, в окрестности которой предполагается оценивать частоту и мощность пика. По умолчанию она равна –1. Если этот вход не подключен, то ВП автоматически ищет максимума пика в массиве спектра мощности и оценивает частоту и мощность в его окрестности.
Вход **диапазон** (span) определяет количество спектральных линий в окрестности пика, включаемых в оценки частоты и мощности. По умолчанию диапазон равен 7, что означает включение в расчетные выражения, помимо **частоты пика**, трех спектральных линий до и после этой частоты.
Выход **оценка частоты пика** (est frequency peak) отображает оценку частоты пика, рассчитанную как его центр тяжести:

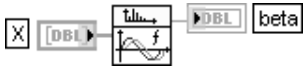
$$\textit{частота пика} = \frac{\sum \textit{спектр мощности}(j)(j * df)}{\sum \textit{спектр мощности}(j)},$$

где $j = \overline{i - \textit{диапазон}/2,\ i + \textit{диапазон}/2}$, i – индекс пика, $\textit{спектр мощности}\ (j)$ – значение спектра мощности j-ой линии спектра, df – расстояние между линиями спектра.
Выход **оценка мощности пика** (est power peak) отображает оценку мощности пика, рассчитанную в соответствии с выражением

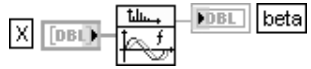
$$\textit{мощность пика} = \frac{\sum \textit{спектр мощности}(j)}{ENBW},$$

где $j = \overline{i - \textit{диапазон}/2,\ i + \textit{диапазон}/2}$, $ENBW$ – **эквивалентная шумовая полоса окна** (equivalent noise bandwidth of the window).
$ENBW$ является одним из параметров **констант окна** (window constants). По умолчанию значение $ENBW$ равно 1,0. Вторым параметром **констант окна** является coherent gain – величина, обратная масштабному коэффициенту окна.
Значения **констант окна** обычно формируются на выходе ВП **Масштабированное временное окно** (Scaled Time Domain Window)

Buneman Frequency Estimator



Оценка частоты колебания



ВП производит оценку относительной частоты **β** гармонического колебания, представленного последовательностью X, с помощью анализа амплитудного спектра этой последовательности.

$$\beta = b + \frac{n}{\pi} a\tan\left(\frac{\sin\frac{\pi}{n}}{\cos\frac{\pi}{n} + \frac{|F_b(X)|}{|F_{b+1}(X)|}}\right),$$

где b – относительная частота гармоники спектра с максимальной амплитудой, $|F_b|$, $|F_{b+1}|$ – значения максимальной по амплитуде и соседней с ней гармоник спектра, n – объем последовательности.
Действительная частота колебания рассчитывается путем умножения **β** на частоту первой гармоники
$f = \beta df = \beta^* f_s / n$,
где f_s – частота дискретизации

4.5. Функции фильтров

Вид основной палитры **функций фильтров** (Filters) и двух подпалитр **Дополнительная БИХ фильтрация** (Advanced IIR Filtering) и **Дополнительная КИХ фильтрация** (Advanced FIR Filtering) показан на рис. 4.19а, 4.19б и 4.19в соответственно. В основной палитре размещены функции, реализующие фильтры Баттерворта, Чебышева, Бесселя и эллиптические фильтры, а также набор фильтров с равномерными пульсациями (Equi-Ripple), медианный фильтр и фильтр 1/f. Необходимо отметить также, что два ВП цифровых фильтров и Экспресс-ВП **Фильтр**

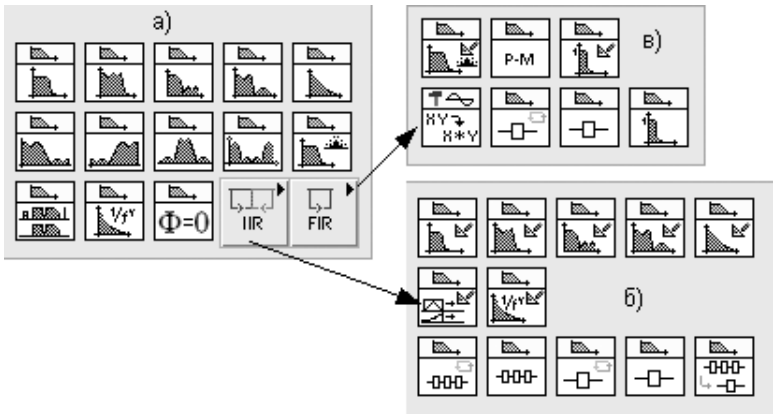


Рис. 4.19. Вид основной палитры (а) и дополнительных подпалитр (б, в) функций фильтров

Рис. 4.20. Классификация цифровых фильтров

находятся в подпалитре **Согласование осциллограмм** (Waveform Conditioning) палитры **Анализ** (Analyze).

Классификация цифровых фильтров, реализованных в LabVIEW, приведена на рис. 4.20.

Линейные фильтры характеризуются выполнением принципа суперпозиции и пропорциональности $L\{as_1(t) + bs_2(t)\} = aL\{s_1(t)\} + bL\{s_2(t)\}$, где a и b – константы, $s_1(t)$ и $s_2(t)$ – сигналы, $L\{*\}$ – линейная операция фильтрации.

Линейные фильтры с бесконечной импульсной характеристикой (БИХ-фильтры) характеризуются следующей связью входных и выходных значений [7, 8]:

$$y_i = \sum_{j=0}^{N} b_j x_{i-j} - \sum_{k=1}^{M} a_k y_{i-k},$$

где N – порядок полинома прямой ветви, M – порядок полинома обратной ветви.

У линейных фильтров с конечной импульсной характеристикой (КИХ-фильтров) коэффициенты полинома обратной ветви равны 0 и текущее значение выхода зависит только от N входных значений.

Достоинством БИХ-фильтров является большее быстродействие и меньший объем необходимой памяти, обусловленные меньшим числом коэффициентов, а недостатком – нелинейность фазовой характеристики.

Частотная характеристика фильтра Баттерворта (рис. 4.21) характеризуется гладкостью на всех частотах и монотонностью спада, начинающейся с некоторой частоты среза. Частотой среза называется частота, на которой мощность выходного сигнала уменьшается в два раза. Фильтры Баттерворта имеют максимально плоскую характеристику в полосе пропускания и ноль в полосе заграждения. При фиксированной частоте среза крутизна характеристики зависит от порядка фильтра.

Фильтры Чебышева (рис. 4.22) минимизируют амплитуду ошибки в полосе пропускания, имеют более узкую переходную полосу (большую крутизну спада) и обеспечивают максимально плоскую характеристику в полосе заграждения. Равномерная характеристика в полосе пропускания ограничивается максимальной допустимой величиной ошибки (величиной выброса).

Инверсные фильтры Чебышева (фильтры Чебышева второго типа) (рис. 4.23) минимизируют амплитуду ошибки в полосе заграждения и обеспечивают максимально плоскую характеристику в полосе пропускания. При этом крутизна характеристики в переходной полосе превышает крутизну фильтра Баттерворта при том же порядке. Это позволяет уменьшить абсолютную ошибку и повысить скорость обработки сигнала.

Эллиптические фильтры минимизируют величину ошибки, распределяя ее по полосе пропускания и по полосе заграждения. По сравнению с фильтрами Баттерворта и фильтрами Чебышева эллиптические фильтры обеспечивают самую высокую крутизну переходной области.

Фильтры Бесселя отличаются от рассмотренных выше фильтров большей линейностью фазочастотной характеристики в полосе пропускания.

ВП фильтров Баттерворта, Чебышева, Бесселя и эллиптических фильтров имеют схожий состав входов и выходов, отличаясь небольшими деталями.

Функции, размещенные в основной палитре фильтров, рассмотрены ниже в таблице.

Рис. 4.21. Амплитудно-частотная характеристика фильтра Баттерворта

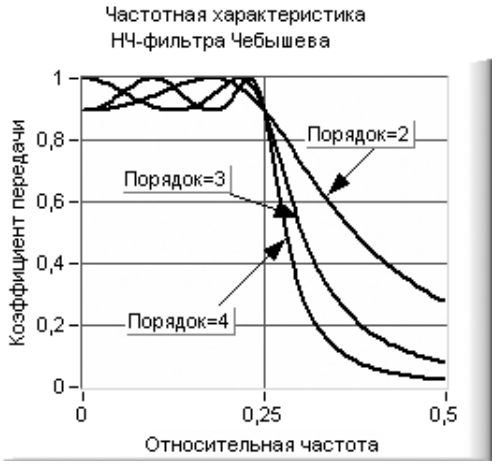
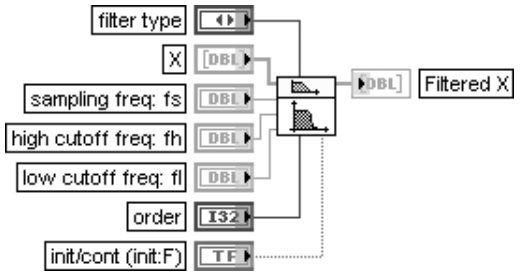


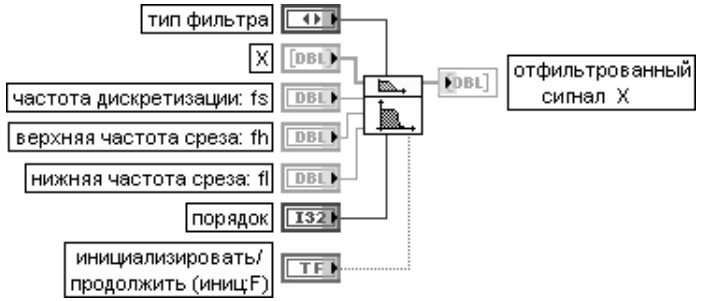
Рис. 4.22. Амплитудно-частотная характеристика фильтра Чебышева

Рис. 4.23. Амплитудно-частотная характеристика инверсного фильтра Чебышева

Butterworth Filter



Фильтр Баттерворта



ВП выполняет функцию цифрового фильтра Баттерворта путем вызова ВП **Коэффициенты Баттерворта** (Butterworth Coefficients) и ВП **Каскадный БИХ-фильтр** (IIR Cascade Filter) (рис. 4.24).

Вход **тип фильтра** (filter type) определяет тип полосы пропускания фильтра.

0	1	2	3
Фильтр нижних частот (Lowpass)	**Фильтр верхних частот** (Highpass)	**Полосовой фильтр** (Bandpass)	**Режекторный фильтр** (Bandstop)

Вход **частота дискретизации: fs** (sampling freq: fs) определяет частоту выборок. **Частота дискретизации** должна быть больше 0. По умолчанию она равна 1,0.

Вход **верхняя частота среза** (high cutoff freq: fh) игнорируется для фильтров типа 0 (фильтр нижних частот) или 1 (фильтр верхних частот). Для фильтров типа 2 (полосовой фильтр) или 3 (режекторный фильтр) **верхняя частота среза** должна быть больше, чем **нижняя частота среза**, и соответствовать критерию Найквиста $\mathbf{0 < f_h < 0{,}5 f_s}$.

Значение входа **нижняя частота среза** (low cutoff freq: fl) по умолчанию равно 0,125.

Вход **порядок** (order) определяет порядок фильтра, который должен быть больше нуля. По умолчанию его значение равно 2.

Вход **инициализировать/продолжить** (init/cont) управляет инициализацией внутренних состояний. По умолчанию на входе установлено значение ЛОЖЬ. При этом внутренние состояния устанавливаются в 0. При установке на входе **инициализировать/продолжить** значения ИСТИНА внутренние состояния соответствуют последним состояниям фильтра из предыдущего вызова данного ВП. При фильтрации длинной последовательности, которая может быть разбита на меньшие блоки, целесообразно устанавливать данный вход в состояние ЛОЖЬ при фильтрации первого блока и в состояние ИСТИНА при фильтрации остальных блоков.

ВП **Коэффициенты Баттерворта** (Butterworth Coefficients), находящийся в подпалитре **Дополнительная БИХ фильтрация** (Advanced IIR Filtering) (рис. 4.19б), принимает перечисленные выше значения входов и формирует на выходе **Кластер БИХ-фильтра** (IIR Filter Cluster), содержащий коэффициенты БИХ-фильтра Баттерворта в каскадной форме. В состав кластера входят следующие компоненты:

- выход **структура фильтра** (filter structure) – отображает второй или четвертый порядок фильтра;
- выход **коэффициенты обратной связи** (Reverse Coefficients) – отображает коэффициенты обратной связи каскадного БИХ-фильтра;
- выход **коэффициенты прямой передачи** (Forward Coefficients) – отображает коэффициенты прямой передачи каскадного БИХ-фильтра.

Исходя из блок-диаграммы ВП, второй порядок устанавливается для фильтров типа 0 или 1, а четвертый порядок – для фильтров типа 2 или 3.

ВП **Каскадный БИХ-фильтр** (IIR Cascade Filter) производит фильтрацию входной последовательности **X**, используя каскадную форму БИХ-фильтра, полученную с помощью ВП **Коэффициенты Баттерворта**.

Набор и содержание входов и выходов ВП **Коэффициенты Баттерворта** (Butterworth Coefficients) и **Каскадный БИХ-фильтр** (IIR Cascade Filter) приведены в следующих разделах данной таблицы без дополнительных пояснений.

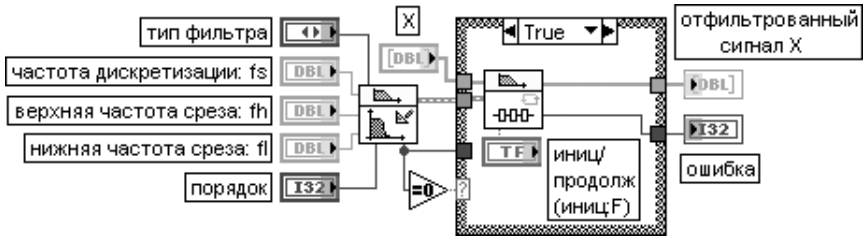
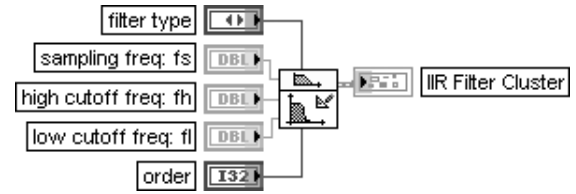
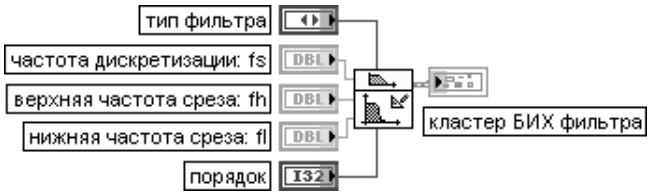


*Рис. 4.24. Блок-диаграмма ВП **Фильтр Баттерворта** (Butterworth Filter)*

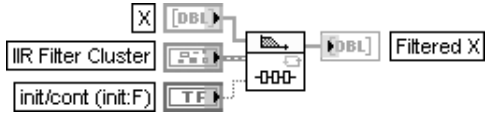
Butterworth Coefficients



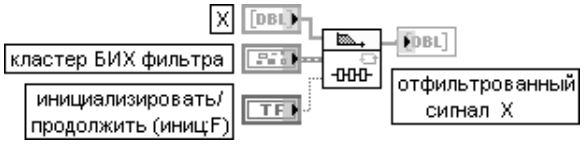
Коэффициенты Баттерворта



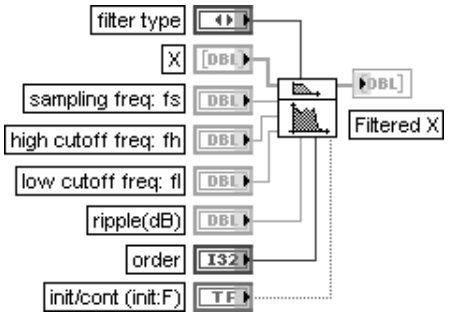
IIR Cascade Filter



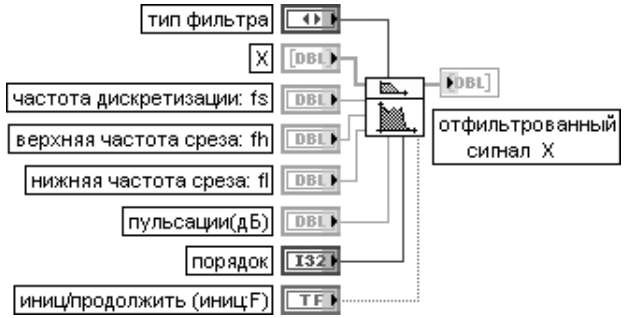
Каскадный БИХ-фильтр



Chebyshev Filter



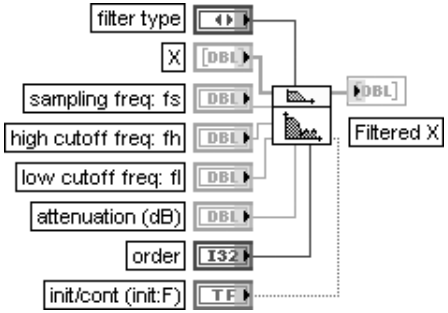
Фильтр Чебышева



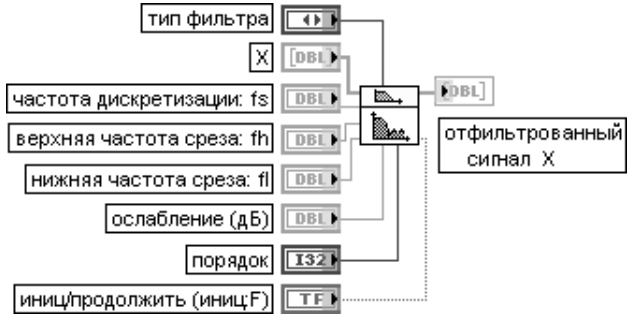
ВП выполняет функцию цифрового фильтра Чебышева путем вызова ВП **Коэффициенты Чебышева** (Chebyshev Coefficients) и ВП Каскадный БИХ-фильтр.

Перечень входов ВП идентичен рассмотренному выше ВП Фильтр Баттерворта, за исключением входа **пульсации** (ripple(dB)), с помощью которого определяется уровень пульсаций частотной характеристики фильтра в полосе пропускания. Величина пульсаций должна быть задана в децибелах. Значение по умолчанию на входе пульсации равно 0,1

Inverse Chebyshev Filter



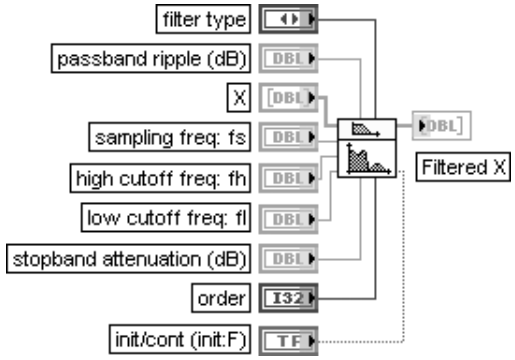
Инверсный фильтр Чебышева



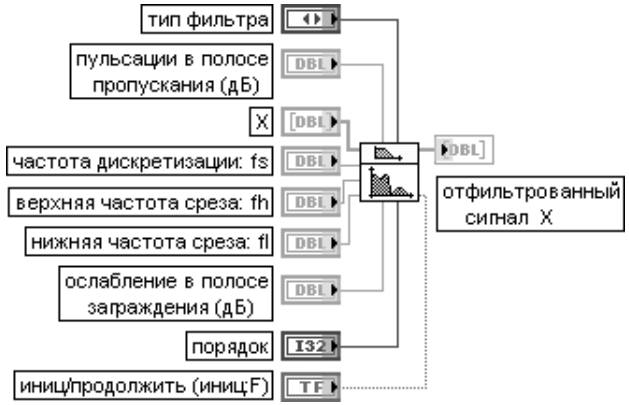
ВП выполняет функцию цифрового инверсного фильтра Чебышева путем вызова ВП **Коэффициенты инверсного фильтра Чебышева** (Inverse Chebyshev Coefficients) и ВП Каскадный БИХ-фильтр.

Перечень входов ВП идентичен рассмотренному выше ВП Фильтр Чебышева, за исключением входа **ослабление** (attenuation(dB)), с помощью которого определяется величина ослабления в полосе заграждения. Величина ослабления должна быть задана в децибелах. Значение по умолчанию на входе **ослабление** равно 60,0

Elliptic Filter

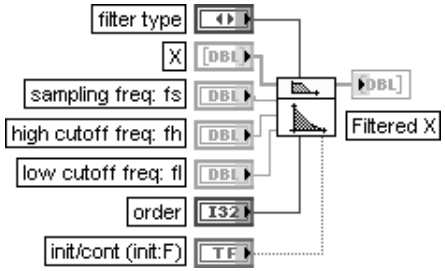


Эллиптический фильтр

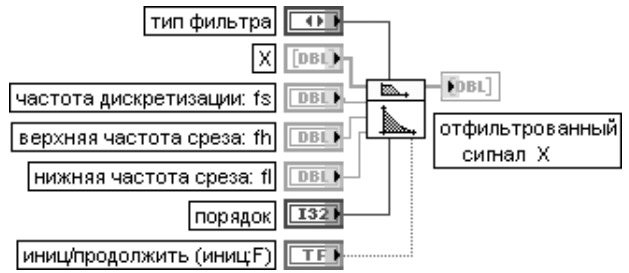


ВП выполняет функцию цифрового эллиптического фильтра путем вызова ВП **Коэффициенты эллиптического фильтра** (Elliptic Coefficients) и ВП Каскадный БИХ-фильтр.
Особенностью входов данного ВП является то, что с помощью входа **пульсации в полосе пропускания (дБ)** (passband ripple (dB)) задается уровень пульсаций в полосе пропускания, а с помощью входа **ослабление в полосе заграждения** (stopband attenuation(dB)) одновременно задается величина ослабления в полосе заграждения. Значения на перечисленных входах должны быть указаны в децибелах

Bessel Filter



Фильтр Бесселя



ВП выполняет функцию цифрового фильтра Бесселя путем вызова ВП **Коэффициенты Бесселя** (Bessel Coefficients) и ВП Каскадный БИХ-фильтр.
Перечень входов ВП идентичен рассмотренному выше ВП Фильтр Баттерворта

Equi-Ripple LowPass

Фильтр нижних частот с равномерными пульсациями

ВП реализует КИХ-фильтр нижних частот с равномерными пульсациями АЧХ, используя алгоритм Паркса – Мак-Клеллана и параметры **число отводов** (**#** of taps), **частота пропускания** (pass freq), **частота заграждения** (stop freq) и **частота дискретизации** (sampling freq: fs).
Данный ВП применяет фильтр низких частот с линейной ФЧХ к входной последовательности **X**, используя ВП **Свертка** (Convolution) для получения **отфильтрованного сигнала X** (рис. 4.25).

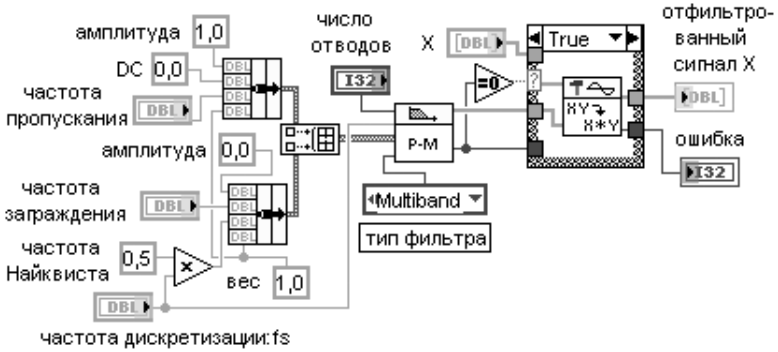
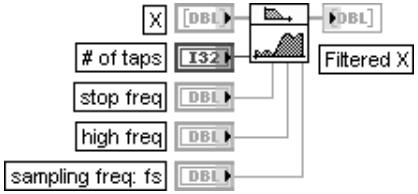


*Рис. 4.25. Блок-диаграмма ВП **Фильтр нижних частот с равномерными пульсациями** (Equi-Ripple LowPass)*

Вход **X** представляет входной сигнал фильтра.
Значение на входе **число отводов** (**#** of taps) должно быть больше нуля. По умолчанию значение этого параметра равно 32. Если **число отводов** меньше или равно 0, то ВП устанавливает на выходе **отфильтрованный X** (Filtered X) пустой массив и возвращает ошибку от ВП **Паркс – Мак-Клеллан** (Parks-McClellan).
Значение на входе **частота пропускания** (pass freq) должно быть больше 0 и соответствовать критерию Найквиста. По умолчанию значение данного параметра равно 0,2.
Значение на входе **частота заграждения** (stop freq) должно быть больше **частоты пропускания** и соответствовать критерию Найквиста. По умолчанию значение данного параметра равно 0,3.

Значение на входе **частота дискретизации: fs** (sampling freq: fs) должно быть больше 0.
По умолчанию это значение равно 1,0.
Выход **отфильтрованный сигнал X** (Filtered X) содержит результат фильтрации входной последовательности **X** с помощью свертки. Число элементов k **отфильтрованного сигнала X** определяется следующим выражением:
$k = n + m - 1$,
где n – число элементов **X**, m – число отводов.
Задержка, также связанная с выходной последовательностью, определяется следующим выражением: $задержка = (m + 1) / 2$.

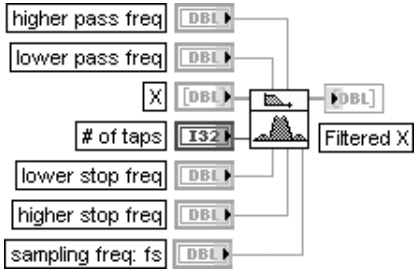
Equi-Ripple HighPass



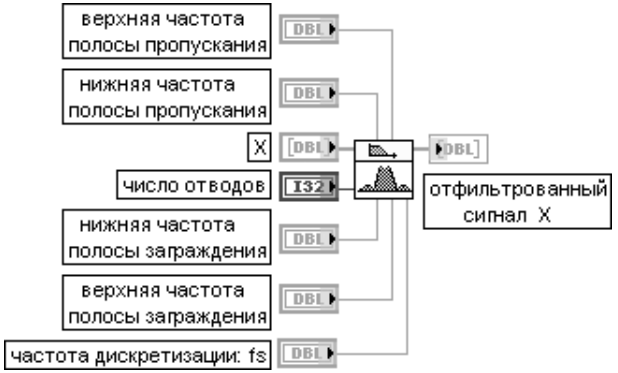
Фильтр верхних частот с равномерными пульсациями

ВП реализует КИХ-фильтр верхних частот с равномерными пульсациями АЧХ.
Перечень входов ВП идентичен рассмотренному выше ВП Фильтр нижних частот с равномерными пульсациями (Equi-Ripple LowPass)

Equi-Ripple BandPass

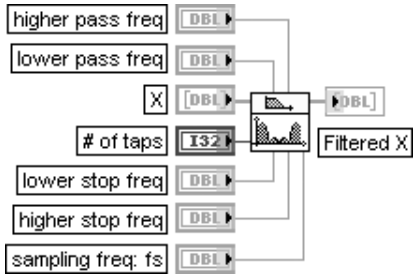


Полосовой фильтр с равномерными пульсациями

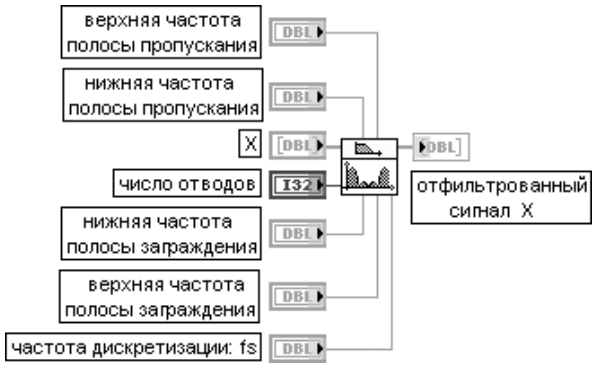


ВП реализует полосовой КИХ-фильтр с равномерными пульсациями АЧХ.
Отличие данного ВП от двух рассмотренных выше аналогичных ВП заключается в увеличении числа входов, определяющих характерные частоты фильтра.
Так, в частности, вместо одной частоты пропускания **введены** верхняя **и** нижняя частоты полосы пропускания**, а вместо одной** частоты заграждения **введены** верхняя **и** нижняя частоты полосы заграждения
Между этими частотами для полосового фильтра должны выполняться следующие соотношения:
нижняя частоты полосы заграждения < нижняя частоты полосы пропускания < верхняя частоты полосы пропускания < верхняя частоты полосы заграждения < 0,5* частота дискретизации: fs

Equi-Ripple BandStop

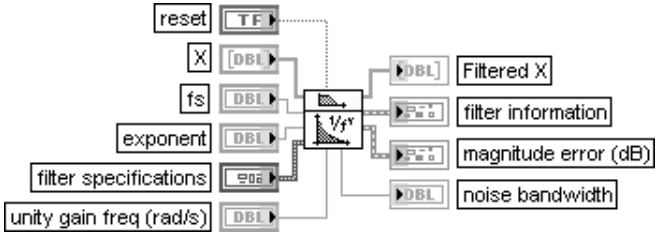


Режекторный фильтр с равномерными пульсациями

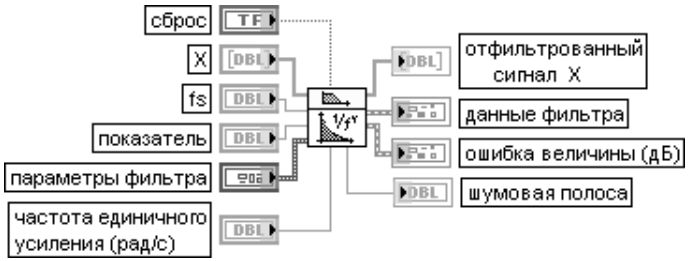


ВП реализует режекторный КИХ-фильтр с равномерными пульсациями АЧХ.
Для режекторного фильтра между характерными частотами должны выполняться следующие соотношения: **нижняя частоты полосы пропускания < нижняя частоты полосы заграждения < верхняя частоты полосы заграждения < верхняя частоты полосы пропускания < 0,5* частота дискретизации: fs**

Inverse f Filter



Фильтр 1/f



ВП рассчитывает и осуществляет БИХ-фильтр, у которого квадрат амплитудно-частотной характеристики обратно пропорционален частоте в заданном частотном диапазоне. Такой 1/f фильтр обычно используется для «окраски» белого шума с равномерной спектральной плотностью.
Вход **сброс** (reset) вызывает пересчет коэффициентов фильтра и сброс в 0 внутреннего состояния фильтра при подаче на него значения ИСТИНА. По умолчанию на этом входе установлено состояние ЛОЖЬ.
Вход **показатель** (exponent) определяет показатель в выражении для расчета квадрата частотной характеристики фильтра 1/(частота^**показатель**). По умолчанию значение показателя равно 1,0.
Вход **параметры фильтра** (filter specifications) определяет диапазон рабочих частот и порядок фильтра. В состав кластера **параметры фильтра** входят следующие элементы:

- **нижняя частота среза** (lower cutoff freq) – определяет нижнюю граничную частоту диапазона рабочих частот фильтра. По умолчанию значение параметра равно 0,1;
- **верхняя частота среза** (higher cutoff freq) – определяет верхнюю граничную частоту диапазона рабочих частот фильтра. По умолчанию значение параметра равно 100;
- **порядок** (order) – определяет число секций первого порядка фильтра 1/f. Увеличение **порядка** приводит к улучшению формы частотной характеристики фильтра 1/f, однако при этом возрастают и вычислительные затраты.

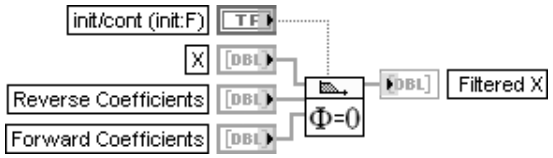
Вход **частота единичного усиления** (unity gain freq) определяет частоту в рад/с, на которой идеальный 1/f фильтр имеет единичное усиление. Действительный 1/f фильтр конструируется с целью обеспечения аппроксимации идеального фильтра в полосе частот, заданной **параметрами фильтра**. Следовательно, действительное усиление фильтра на частоте **единичного усиления** будет находиться в окрестности единицы, если сама частота **единичного усиления** находится в диапазоне, заданном **спецификацией фильтра**.
Выход **отфильтрованный сигнал X** (Filtered X) представляет выходной массив отфильтрованных выборок сигнала.

Выход **данные фильтра** (filter information) возвращает величину и фазу частотной характеристики разрабатываемого 1/f фильтра.
Выход **ошибка величины** (magnitude error) возвращает величину отклонения частотной передаточной характеристики реального 1/f фильтра от идеального в виде кластера, содержащего следующие элементы:

- **частота** (frequency) возвращает набор частот, на которых определяется величина ошибки;
- **величина** (magnitude) возвращает величину ошибки в децибелах.

Выход **шумовая полоса** (noise bandwidth) возвращает ожидаемую шумовую полосу создаваемого 1/f фильтра

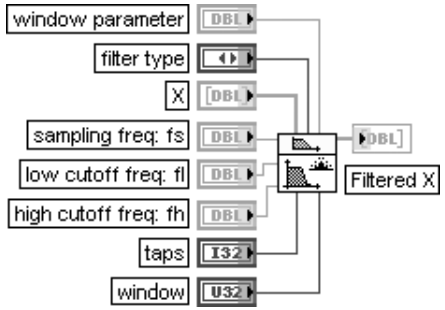
Zero Phase Filter



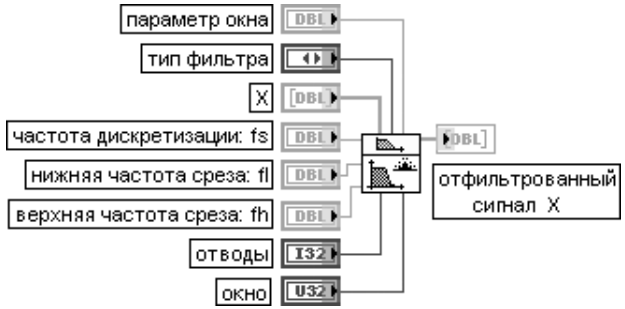
Фильтр без сдвига фазы

ВП прикладывает фильтр без сдвига фазы к входной последовательности ***X***. Нулевой сдвиг фаз достигается за счет повторной фильтрации сигнала с инверсным порядком расположения отсчетов.
Вход **иниц./продолжить** (init/cont) управляет инициализацией внутренних состояний фильтра.
Коэффициенты **прямой передачи** и **обратной связи** должны быть заданы в прямой форме

FIR Windowed Filter



Оконный КИХ-фильтр

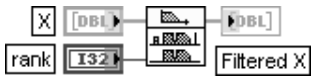


ВП фильтрует входную последовательность данных **X**, используя набор коэффициентов оконного КИХ-фильтра. Коэффициенты определяются значениями на входах **частота дискретизации: fs** (sampling freq: fs), **нижняя частота среза** (low cutoff freq: fl), **верхняя частота среза** (high cutoff freq: fh) и **число отводов** (number of taps).
Значение на входе **отводы** (taps) определяет общее число КИХ-коэффициентов и должно быть больше 0. По умолчанию оно равно 25.
Вход **окно** (window) определяет тип сглаживающего окна. **Сглаживающие окна** (Smoothing windows) уменьшают пульсации в полосе пропускания фильтра и улучшают способность фильтра подавлять частотные компоненты в полосе заграждения. Предусмотрены следующие типы окон:

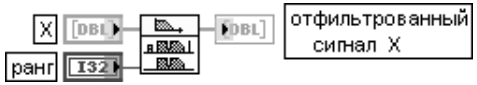
0	**Без окна** (None)
5	**Точное Блэкмана** (Exact Blackman)
1	**Ханна** (Hanning)
6	**Блэкмана-Хэрриса** (Blackman-Harris)
2	**Хэмминга (**Hamming)
7	**Кайзера-Бесселя** (Kaiser-Bessel)
3	**Треугольное** (Triangular)
8	**Плосковершинное** (Flat Top)
4	**Блэкмана** (Blackman)

Между характерными частотами оконного КИХ-фильтра должны выполняться следующие соотношения: **нижняя частота среза** < **верхняя частота среза** < **частота дискретизации: fs**

Median Filter



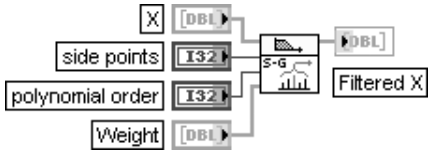
Медианный фильтр



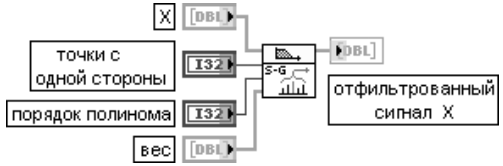
ВП выполняет медианную фильтрацию входной последовательности ***X*** фильтром с заданным **рангом** (rank).
Вход ***X*** представляет входной фильтруемый сигнал. Число элементов ***n*** последовательности ***X*** должно быть больше, чем **ранг**. Если число элементов ***X*** меньше или равно значению **ранг**, то ВП выводит на выход **отфильтрованный сигнал *X*** (Filtered *X*) пустой массив и возвращает ошибку.
Вход **ранг** (rank) должен быть больше или равен 0. По умолчанию значение входа равно 2.
Выход **отфильтрованный сигнал *X*** (Filtered *X*) отображает выходной массив отфильтрованных выборок. Размер этого массива такой же, как у входного массива ***X***.
ВП **Медианный фильтр** рассчитывает элементы выходной последовательности **отфильтрованный сигнал *X***, используя следующее выражение:
$y_i = Median(J_i)$,

где $i = \overline{0, n-1}$, J_i – фрагмент входной последовательности ***X***, центрированный относительно i-го элемента X. J_i задается следующим выражением: $J_i = \{x_{i-r}, x_{i-r+1}, \ldots, x_{i-1}, x_i, x_{i+1}, \ldots, x_{i+r-1}, x_{i+r}\}$, где r – ранг фильтра

Savitzky-Golay Filter



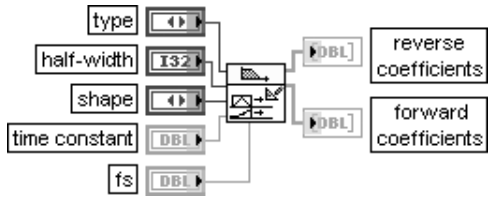
Фильтр Савицки-Голэй



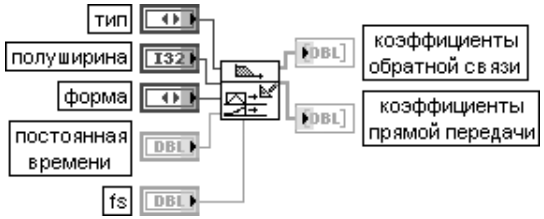
ВП производит фильтрацию входной последовательности данных **X** с помощью сглаживающего КИХ-фильтра Савицки-Голэй.
Вход **точки с одной стороны** (side points) определяет количество точек с каждой стороны от текущей точки, которые используются для минимизации с помощью метода наименьших квадратов. При этом длина всего скользящего окна равна **side points***2+1 и должна быть больше **порядка полинома** (polynomial order).
Вход **вес** (Weight) задает весовой вектор, применяемый в минимизации с помощью м.н.к. Массив должен быть пустым или иметь длину **side points***2+1.
Фильтр Савицки-Голэй производит сглаживание шума с помощью полиномиальной аппроксимации сигнала в скользящем окне

Подпалитра **Дополнительная БИХ фильтрация** (Advanced IIR Filtering) содержит в верхней строке ряд ВП, применяемых в составе ВП фильтров Баттерворта, Чебышева, Бесселя, эллиптического фильтра и фильтра 1/f для расчета коэффициентов этих фильтров в каскадной форме. Данные ВП имеют входы и выходы, являющиеся подмножеством входов и выходов использующих их ВП, не требуют отдельного рассмотрения. Аналогичное замечание относится и к ВП **Каскадный БИХ-фильтр** (IIR Cascade Filter), являющемся вторым важным элементом указанных БИХ-фильтров. Состав его входов и выходов был приведен при описании ВП **Фильтр Баттерворта**. Ниже в таблице приведены ВП, не входящие в состав других ВП и соответственно не рассмотренные ранее.

Smoothing Filter Coefficients



Коэффициенты сглаживающего фильтра



ВП рассчитывает коэффициенты сглаживающего фильтра. Данный ВП может использоваться для расчета КИХ-фильтра скользящего сглаживания или экспоненциального усредняющего БИХ-фильтра. ВП возвращает **коэффициенты обратной связи** (reverse coefficients) и **коэффициенты прямой передачи** (forward coefficients) для непосредственной передачи в ВП **БИХ-фильтр** (IIR Filter), который и используется для реализации КИХ- и БИХ-фильтров.
Вход **тип** (type) определяет тип сглаживающего фильтра.

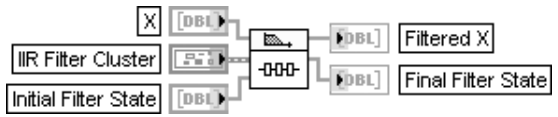
0	**Скользящее усреднение** (moving average) (по умолчанию) – рассчитывает только коэффициенты прямой передачи (КИХ-коэффициенты)
1	**Экспоненциальное усреднение** (exponential) – рассчитывает БИХ-коэффициенты первого порядка

Вход **полуширина** (half-width) определяет полуширину фильтра скользящего усреднения, выраженную числом выборок. Для **полуширины M** полная ширина фильтра скользящего сглаживания равна **N** = **1** + **2M** выборок. Таким образом, полная ширина **N** всегда является нечетным числом.
Вход **форма** (shape) определяет форму фильтра скользящего сглаживания.

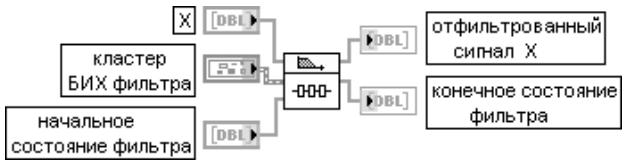
0	**Прямоугольное** (rectangular) (по умолчанию) – все выборки в скользящем окне берутся с равными весами при расчете каждого сглаженного выходного отсчета
1	**Треугольное** (triangular) – скользящее окно имеет треугольный закон распределения весовых коэффициентов, при котором максимальное значение находится в центре окна, а остальные веса линейно спадают к краям

Вход **постоянная времени** (time constant) определяет постоянную времени экспоненциального усредняющего фильтра в секундах.
Вход **fs** определяет частоту дискретизации, выраженную числом выборок в секунду.
Выход **reverse coefficients** содержит **коэффициенты обратной связи** БИХ-фильтра.
Выход **forward coefficients** содержит **коэффициенты прямой передачи** КИХ-фильтра

IIR Cascade Filter with I.C.

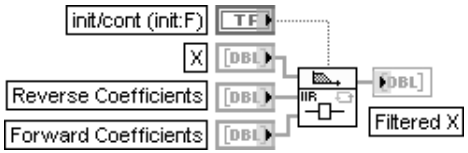


Каскадный БИХ-фильтр с начальным условием

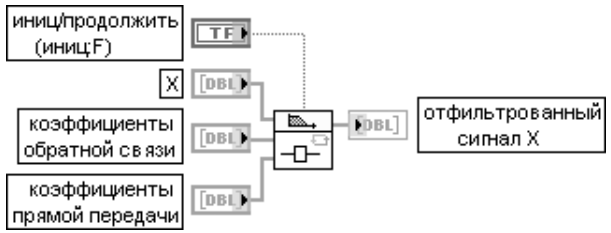


ВП производит фильтрацию входной последовательности **X**, используя каскадную форму БИХ-фильтра, определенную с помощью входа **Кластер БИХ-фильтра** (IIR Filter Cluster).
Вход **начальное состояние фильтра** (Initial Filter State) должен быть того же размера, что и массив **коэффициентов обратной связи** (Reverse Coefficients) в **кластере БИХ-фильтра**.
Отличие данного ВП от аналогичного ВП **Каскадный БИХ-фильтр** (IIR Cascade Filter) связано с возможностью непосредственного управления входом **начальное состояние фильтра** и получения на выходе значения **конечное состояние фильтра**

IIR Filter



БИХ-фильтр

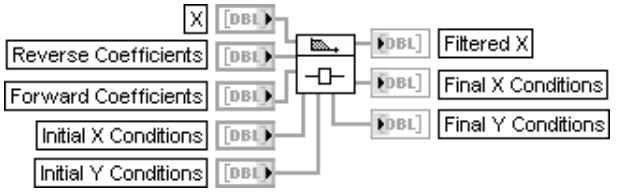


ВП производит фильтрацию входной последовательности ***X***, используя прямую форму БИХ-фильтра, определенную с помощью **коэффициентов обратной связи** (reverse coefficients) и **коэффициентов прямой передачи** (forward coefficients).
Расчет элементов **отфильтрованного сигнала *X*** производится с помощью следующего выражения:

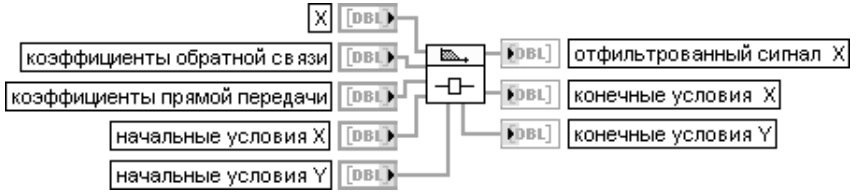
$$y_i = \frac{1}{a_0}\left(\sum_{j=0}^{N} b_j x_{i-j} - \sum_{k=1}^{M} a_k y_{i-k}\right),$$

где y_i – элемент **отфильтрованного сигнала X**, N – число **коэффициентов прямой передачи** b_j, M – число **коэффициентов обратной связи** a_k

IIR Filter with I.C.



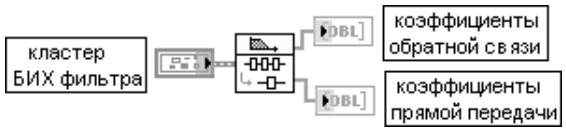
БИХ-фильтр с начальными условиями



Отличие данного ВП от аналогичного рассмотренного выше ВП **БИХ-фильтр** заключается в том, что при фильтрации блоков непрерывных данных возможно инициализировать фильтр при $I < 0$ с помощью следующих выражений:
$y_i = y_{ic}[M + i - 1]$; $x_i = x_{ic}[N + i - 1]$, где y_{ic} представляет массив **начальные условия** *Y*, а x_{ic} – массив **начальные условия** *X*

Cascade->Direct Coefficients

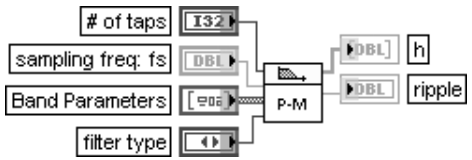
Коэффициенты из каскадной формы в прямую



ВП преобразует коэффициенты БИХ-фильтра из каскадной формы в прямую

Из подпалитры **Дополнительная КИХ фильтрация** (Advanced FIR Filtering) далее в таблице рассмотрены ВП **Паркс – Мак-Клеллан** (Parks-McClellan), включенный в состав ВП фильтров с равномерными пульсациями (Equi-Ripple Filter) (рис. 4.25), и **Узкополосный КИХ-фильтр** (FIR Narrowband Filter) с ВП **Коэффициенты узкополосного фильтра** (FIR Narrowband Coefficients).

Parks-McClellan



Паркс – Мак-Клеллан

ВП формирует набор коэффициентов цифрового многополосного КИХ-фильтра с линейной фазочастотной характеристикой, используя параметры **число отводов** (**#** of taps), **частота дискретизации: fs** (sampling freq: fs), **параметры полосы** (Band Parameters) и **тип фильтра** (filter type).

Вход **число отводов** (**#** of taps) содержит общее число коэффициентов на выходе **h**. По умолчанию значение на этом входе равно 32. Отвод соответствует умножению и сложению. При задании **n** отводов для каждой фильтруемой выборки производится **n** умножений и **n** сложений. **Число отводов** должно быть больше 0. Если **число отводов** меньше или равно 0, то ВП устанавливает на выходе **h** пустой массив, на выходе **пульсации** значение NaN и возвращает ошибку.

Вход **параметры полосы** (Band Parameters) является массивом кластеров. Каждый элемент кластера содержит необходимую информацию, связанную с каждой полосой создаваемого КИХ-фильтра. Массив кластеров **параметры полосы** должен содержать по крайней мере один элемент, который соответствует одной полосе. По умолчанию на данный вход подается пустой массив. В состав кластера **параметры полосы** входят следующие параметры:

- **амплитуда** (Amplitude) – задает соответствующую величину коэффициента передачи или усиления фильтра между **нижней частотой** (Lower Freq) и **верхней частотой** (Higher Freq). Значение 1,0 соответствует полосе пропускания, а значение 0,0 – полосе заграждения. ВП не устанавливает ограничения на данную величину;
- **нижняя частота** (Lower Freq) – представляет частоту, с которой начинается полоса;
- **верхняя частота** (Higher Freq) – представляет частоту, на которой полоса заканчивается;
- **взвешенное отклонение** (Weighted Ripple) – представляет ошибку взвешенного отклонения, которую минимизирует данный ВП. Чем больше вес, тем меньше ошибка в полосе.

Для каждой полосы **верхняя частота** должна быть больше **нижней частоты**. Для соседних полос **нижняя частота** более высокой полосы должна быть больше **верхней частоты** более низкой полосы.

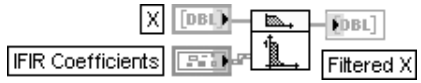
Вход **тип фильтра** (filter type) может иметь следующие значения:

0	1	2
Многополосный (Multiband) (по умолчанию)	**Дифференциатор** (Differentiator)	**Гильберт** (Hilbert)

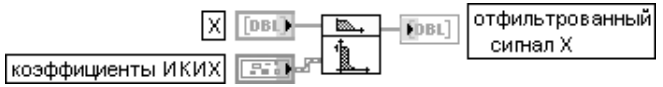
Выход **h** отображает массив коэффициентов КИХ-фильтра, которые ВП рассчитывает, используя алгоритм Пакса-Мак-Клеллана с техникой замены Ремеза.

Выход **пульсация** (ripple) отображает оптимальное отклонение, которое рассчитывается ВП и служит показателем отклонения от идеальной спецификации фильтра

FIR Narrowband Filter

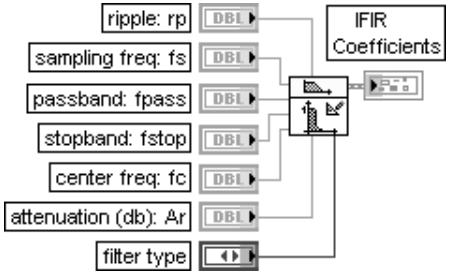


Узкополосный КИХ-фильтр

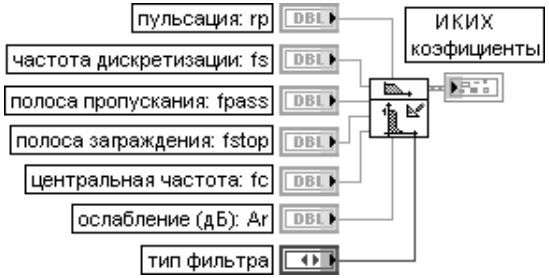


ВП производит фильтрацию входной последовательности **X** с помощью **интерполированного КИХ-фильтра** (interpolated FIR (IFIR) filter), заданного **коэффициентами ИКИХ** (IFIR Coefficients)

FIR Narrowband Coefficients



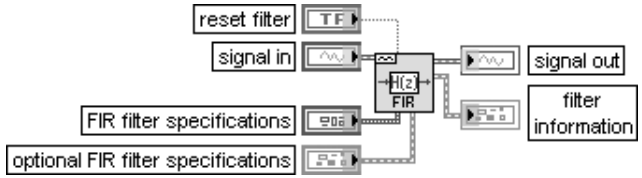
Коэффициенты узкополосного КИХ-фильтра



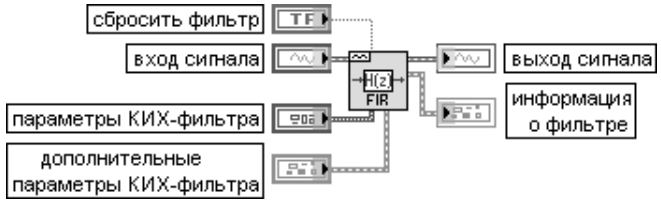
ВП формирует набор коэффициентов фильтра для реализации цифрового **интерполированного КИХ фильтра** (interpolated FIR (IFIR) filter)

Ниже в таблице рассмотрены функции фильтров, находящихся в подпалитре **Согласование осциллограмм** (Waveform Conditioning).

Digital FIR Filter



Цифровой КИХ-фильтр



ВП фильтрует сигналы одной или нескольких осциллограмм.
При фильтрации нескольких осциллограмм ВП сохраняет отдельные состояния фильтра для каждой осциллограммы. Этот полиморфный ВП можно использовать для фильтрации сигналов нескольких осциллограмм с несколькими спецификациями. Типы данных, подключенных ко входам **сигнал** и **спецификации КИХ-фильтра**, определяют используемую реализацию полиморфного ВП.
Вход **спецификации КИХ-фильтра** (FIR filter specifications) представляет кластер, содержащий параметры конструирования для КИХ-фильтра. В состав кластера входят следующие элементы:

Топология (Topology) определяет тип модели фильтра.

0	**Без фильтра** (по умолчанию) (Off (default))
1	**КИХ-фильтр, заданный спецификацией** (FIR by Specification)
2	**КИХ-фильтр с равномерными пульсациями** (Equi-ripple FIR)
3	**Оконный КИХ-фильтр** (Windowed FIR)

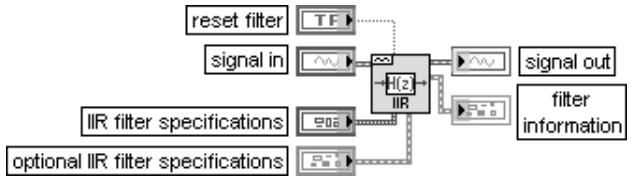
Тип (Type) определяет полосу пропускания фильтра в соответствии со следующими значениями.

0	1	2	3
Фильтр нижних частот (Lowpass)	**Фильтр верхних частот** (Highpass)	**Полосовой фильтр** (Bandpass)	**Режекторный фильтр** (Bandstop)

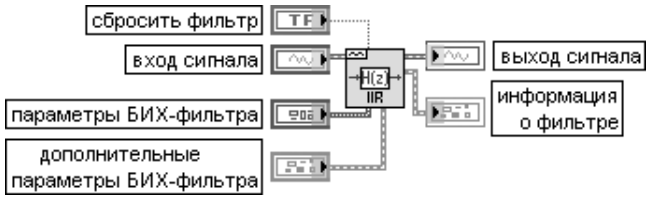
Число отводов (#Taps) задает число отводов КИХ-фильтра. По умолчанию равно 50.
Нижняя PB (Lower PB) определяет нижнюю из двух частот полосы пропускания. По умолчанию равна 100 Гц.
Верхняя PB (Upper PB) определяет верхнюю из двух частот полосы пропускания. По умолчанию равна 0.
Нижняя SB (Lower SB) определяет нижнюю из двух частот полосы заграждения. По умолчанию равна 200 Гц.
Верхняя SB (Upper SB) определяет верхнюю из двух частот полосы заграждения. По умолчанию равна 0.

Вход **дополнительные параметры КИХ-фильтра** (optional FIR filter specifications) представляет кластер дополнительных параметров, которые могут использоваться при задании КИХ-фильтра:
- **усиление PB** (PB Gain) – определяет усиление на частотах пропускания. Усиление может быть задано в линейных единицах или децибелах. По умолчанию равно –3 дБ;
- **усиление SB** (SB Gain) – определяет усиление на частотах заграждения. Усиление также может быть задано в линейных единицах или децибелах. По умолчанию равно –60 дБ;
- **масштаб** (Scale) – определяет интерпретацию параметров **Усиление PB** и **Усиление SB**;
- **окно** (Window) – задает сглаживающее окно, применяемое к округленным коэффициентам. Сглаживающие окна уменьшают пульсации в полосе пропускания и улучшают способность фильтра ослаблять частотные компоненты в полосе заграждения

Digital IIR Filter



Цифровой БИХ-фильтр



ВП фильтрует сигналы одной или нескольких осциллограмм.
При фильтрации нескольких осциллограмм ВП сохраняет отдельные состояния фильтра для каждой осциллограммы. Этот полиморфный ВП можно использовать для фильтрации сигналов нескольких осциллограмм с несколькими спецификациями. Типы данных, подключенных ко входам **сигнал** и **параметры БИХ-фильтра**, определяют используемую реализацию полиморфного ВП.
Вход **параметры БИХ-фильтра** (FIR filter specifications) представляет кластер, содержащий параметры конструирования для БИХ-фильтра. В состав кластера входят следующие элементы:

Топология (Topology) определяет тип модели фильтра.

0	**Без фильтра** (Off)
1	**Баттерворта** (Butterworth)
2	**Чебышева** (Chebyshev)
3	**Инверсный Чебышева** (Inverse Chebyshev)
4	**Эллиптический** (Elliptic)
5	**Бесселя** (Bessel)

Тип (Type) идентичен одноименному параметру ВП **Цифровой КИХ-фильтр**.

Порядок (Order) определяет порядок фильтра. Если порядок равен 0, то фильтр использует **дополнительные параметры БИХ-фильтра** (optional IIR filter specification) для расчета порядка.
Нижняя Fc (Lower Fc) является нижней частотой среза и должна соответствовать критерию Найквиста. По умолчанию равна 100.
Верхняя Fc (Upper Fc) является верхней частотой среза. Этот параметр игнорируется, если установлен тип 0 (фильтр нижних частот) или 1 (фильтр верхних частот).
Параметр **пульсации PB** (PB Ripple) должен быть больше 0 и выражен в децибелах. По умолчанию равен 1,0.
Параметр **ослабление SB** (SB Attenuation) определяет ослабление в полосе заграждения. Он должен быть больше 0 и выражен в децибелах. По умолчанию равен 60,0.

Вход **дополнительные параметры БИХ-фильтра** (optional IIR filter specifications) представляет кластер, содержащий информацию, необходимую для расчета порядка БИХ-фильтра. Параметры, входящие в состав кластера, аналогичны параметрам одноименного кластера ВП **Цифровой КИХ-фильтр**, рассмотренного выше

В палитре функций фильтров отсутствует соответствующий Экспресс-ВП, однако он включен в состав подпалитры **Анализ сигналов** (Signal Analysis) палитры Экспресс-ВП.

Фильтр (Filter)

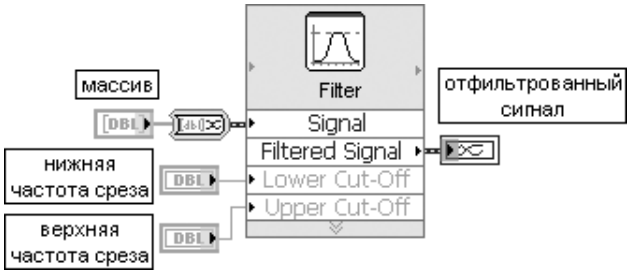


Рис. 4.26. Блок-диаграмма возможного подключения Экспресс-ВП

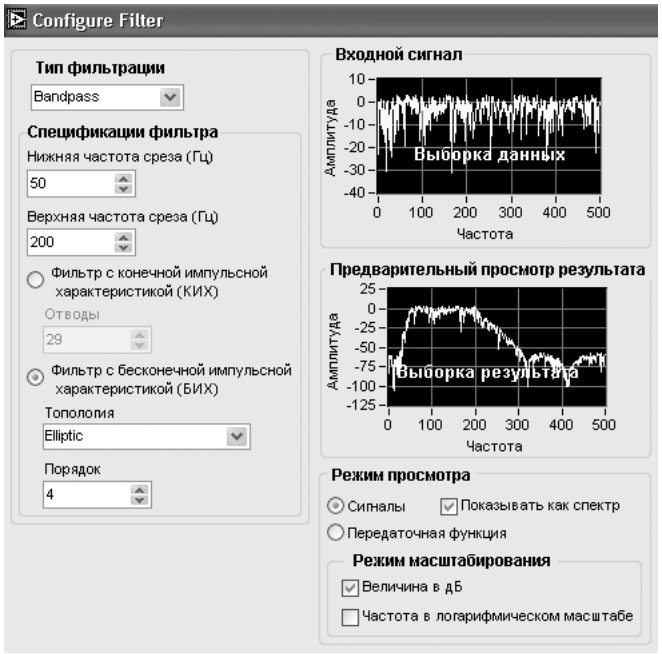


*Рис. 4.27. Вид диалогового окна конфигурирования Экспресс-ВП **Фильтр** (Filter)*

Экспресс-ВП **Фильтр** (Filter) обрабатывает сигналы, используя функции фильтров или функции весовых окон. Он использует функциональность ВП **Цифровые БИХ-фильтры** (Digital IIR Filter) и **Цифровые КИХ-фильтры** (Digital FIR Filter). В связи с тем, что все параметры этих фильтров были рассмотрены в предыдущих разделах, дополнительные пояснения к блок-диаграмме и опциям окна конфигурирования данного Экспресс-ВП (рис. 4.26, рис. 4.27) далее не приводятся

4.6. Функции обработки весовыми окнами

Функции обработки весовыми окнами (рис. 4.28) используются для предварительной обработки сигналов при их спектральном анализе с помощью дискретного преобразования Фурье (ДПФ). Обработка заключается в умножении сигнала на функцию окна. Функции окна имеют максимум в центре и плавно спадают к краям. Такая форма окон приводит к уменьшению растекания спектра сигнала, обусловленного скачками сигнала на краях интервала выборки.

Умножение сигнала на весовую функцию соответствует свертке спектров сигнала и амплитудно-частотной характеристики фильтра, соответствующего весовой функции. В результате свертки пики, содержащиеся в спектре сигнала, несколько расширяются [7]. Рассмотренные ниже окна как раз и отличаются различным соотношением степени подавления боковых лепестков и расширения центрального лепестка частотной характеристики.

В работе [6] приведен график зависимости приведенной ширины F_h основного лепестка АЧХ от уровня боковых лепестков h_L для различных весовых окон (рис. 3.29). Приведенная ширина основного лепестка равна произведению ширины основного лепестка и длительности функции окна.

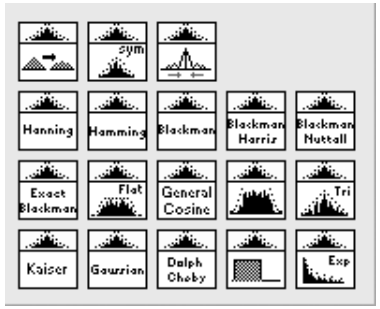


Рис. 4.28. Вид палитры функций обработки весовыми окнами

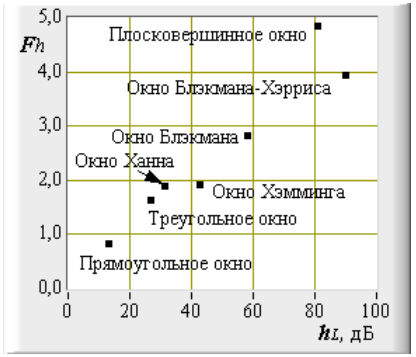
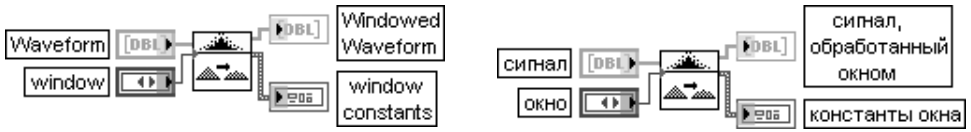


Рис. 4.29. Сравнительный анализ параметров весовых окон

В выражениях, приводимых при анализе окон, рассмотренных ниже в таблице, используется параметр $w = \frac{2\pi i}{n}$, где $i = \overline{0, n-1}$, n – число элементов последовательности X.

Scaled Time Domain Window — Масштабированное временное окно



ВП применяет выбранное **окно** (window) для обработки сигнала, заданного во временной области.
Вход **окно** определяет используемое временное окно. Виды окон перечислены в таблице.

0	**Прямоугольное** (по умолчанию) Rectangle (default)	7	**Четырехзвенное Блэкмана-Хэрриса** (4 Term B-Harris)	32	**Бохмана** (Bohman)
1	**Ханна** (Hanning)	8	**Семизвенное Блэкмана-Хэрриса** (7 Term B-Harris)	33	**Парзена** (Parzen)
2	**Хэмминга** (Hamming)	9	**С низким уровнем боковых лепестков** (Low Sidelobe)	34	**Уэлча** (Welch)
3	**Блэкмана-Хэрриса** (Blackman-Harris)	11	**Блэкмана-Нуталла** (Blackman Nuttall)	60	**Кайзера** (Kaiser)

4	**Точное Блэкмана** (Exact Blackman)	30	**Треугольное** (Triangle)	61	**Дольфа-Чебышева** (Dolph-Chebyshev)
5	**Блэкмана** (Blackman)	31	**Модифицированное Бартлетта-Ханна** (Modified Bartlett-Hanning)	62	**Гауссовское** (Gaussian)
6	**Плосковершинное** (Flat Top)				

Графики окон с номерами, соответствующими номерам в таблице, приведены на рис. 3.30 и 3.31.

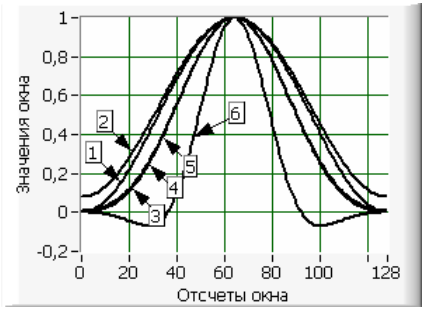


Рис. 4.30. Весовые окна Ханна (1), Хэмминга (2), Блэкмана-Хэрриса (3), Точное Блэкмана (4), Блэкмана (5), Плосковершинное (6)

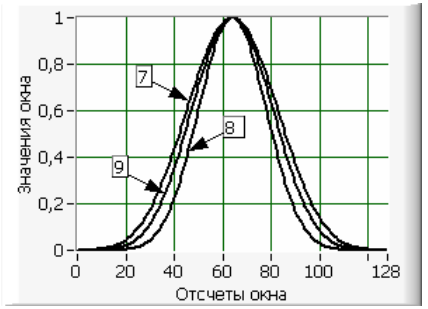


Рис. 4.31. Весовые окна Четырехзвенное Блэкмана-Хэрриса (7), Семизвенное Блэкмана-Хэрриса (8), С низким уровнем боковых лепестков (9)

Выход **оконная осциллограмма** (Windowed Waveform) отображает результат обработки **входной осциллограммы** (Waveform) заданным окном.
Выход **константы окна** (window constants) содержит константы выбранного окна.
По умолчанию значения соответствуют прямоугольному окну (отсутствию окна).
В состав кластера **константы окна** входят следующие элементы:

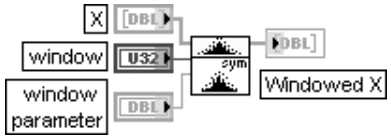
- **eq noise BW** – представляет **эквивалентную шумовую полосу** выбранного окна. Данный параметр может использоваться для расчета мощности в заданном частотном диапазоне;
- **когерентное усиление** (coherent gain) – представляет величину, обратную масштабному фактору окна.

Значения констант окна для некоторых окон приведены в таблице.

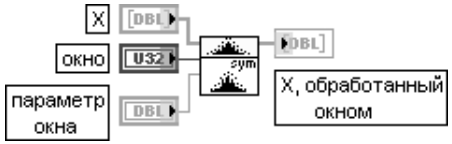
Тип окна	Свойства окна	
	CG	ENBW
Прямоугольное	1,0	1,0
Хана	0,5	1,5
Хэмминга	0,54	1,363
Блэкмана-Хэрриса	0,423	1,709
Точное окно Блэкмана	0,427	1,694
Блэкмана	0,42	1,727
Плосковершинное	0,216	3,77

Тип окна	Свойства окна	
	CG	ENBW
Четырехзвенное Блэкмана-Хэрриса	0,359	2,0
Семизвенное Блэкмана-Хэрриса	0,271	2,632
С низким уровнем боковых лепестков	0,323	2,215
Блэкмана-Нуталла	0,364	1,976

Symmetric Window



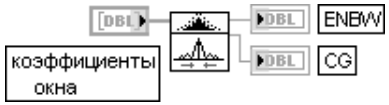
Симметричное окно



ВП применяет симметричное окно к входному сигналу ***X***. Тип данных, подключенных ко входу ***X***, определяет полиморфную реализацию ВП.
Перечень окон, которые могут быть установлены на входе **окно** (window), идентичен перечню окон рассмотренного выше ВП **Масштабированное временное окно.**
На вход **параметр окна** (window parameter) подается параметр бета для окна Кайзера, стандартное отклонение для гауссовского окна и отношение ***s*** уровня главного лепестка к уровню бокового для окна Дольфа-Чебышева. Если **окно** отличается от перечисленных, то этот вход игнорируется.
По умолчанию на входе **параметр окна** установлено значение NaN, при котором значение бета для окна Кайзера устанавливается равным 0, значение стандартного отклонения для гауссовского окна устанавливается равным 0,2, и значение ***s*** для окна Дольфа-Чебышева устанавливается равным 60

Window Properties

Свойства окна



ВП рассчитывает **эквивалентную шумовую полосу** (eq noise BW) **и когерентное усиление** (coherent gain) окна.
Выражения для расчета этих параметров выглядят следующим образом:

$$CG = \frac{\sum_{i=0}^{n-1} W_i}{n} \quad ENBW = \frac{n\sum_{i=0}^{n-1} W_i^2}{\left(\sum_{i=0}^{n-1} W_i\right)^2},$$

где W_i – коэффициенты окна, n – число коэффициентов.
Числовые значения данных параметров были приведены при описании ВП **Масштабированное временное окно**

Hanning Window

Окно Ханна

ВП применяет окно Ханна для обработки входного сигнала ***X***.
Если ***y*** представляет выходную последовательность, обработанную **окном Хана** (Hanning {X}), то ВП получает ее значения с помощью выражения $y_i = 0{,}5x_i[1 - \cos(w)]$

Hamming Window

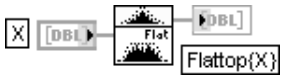
Окно Хэмминга

ВП применяет окно Хэмминга для обработки входного сигнала ***X***.
Если ***y*** представляет выходную последовательность, обработанную **окном Хэмминга** (Hamming {X}), то ВП получает ее значения с помощью выражения $y_i = x_i[0{,}54 - 0{,}46\cos(w)]$

В следующей таблице приведены коэффициенты расчетных выражений для семейства окон Блэкмана. В общем виде эти выражения выглядят следующим образом: $y_i = x_i[a_0 - a_1\cos(w) + a_2\cos(2w) - a_3\cos(3w)]$.

Окно\коэффициенты	a_0	a_1	a_2	a_3
Окно Блэкмана (Blackman Window)	0,42	0,5	0,08	–
Окно Блэкмана-Хэрриса (Blackman-Harris Window)	0,422323	0,49755	0,07922	–
Окно Блэкмана-Нуталла (Blackman-Nuttall Window)	0,3635819	0,4891775	0,1365995	0,0106411
Точное окно Блэкмана (Exact Blackman Window)	7938/18608	9240/18608	1430/18608	–

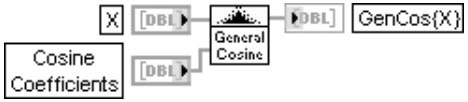
Flat Top Window



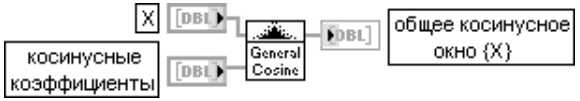
Окно с плоской вершиной

ВП применяет плосковершинное окно для обработки входного сигнала ***X***.
Если ***y*** представляет выходную последовательность, обработанную **окном с плоской вершиной** (Flat Top Window {X}), то ВП получает ее значения с помощью выражения
$y_i = x_i[a_0 - a_1\cos(w) + a_2\cos(2w) - a_3\cos(3w) + a_4\cos(4w)]$,
где $a_0 = 0{,}21557895$, $a_1 = 0{,}41663158$, $a_2 = 0{,}277263158$, $a_3 = 0{,}083578947$, $a_4 = 0{,}006947368$

General Cosine Window



Общее косинусное окно



ВП применяет общее косинусное окно для обработки входного сигнала **X**.
Если **y** представляет выходную последовательность, обработанную **общим косинусным окном** (GenCos {X}), то ВП получает ее значения с помощью выражения

$$y_i = x_i \sum_{k=0}^{m-1} (-1)^k a_k \cos(kw),$$

где a_k – косинусные коэффициенты **общего косинусного окна**, $k = \overline{0, m-1}$

Cosine Tapered Window

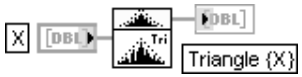
Косинусное 10% окно

ВП применяет косинусное 10% окно для обработки входного сигнала **X**.
Если **y** представляет выходную последовательность, обработанную **косинусным 10% окном** (Cosine Tapered {X}), то ВП получает ее значения с помощью выражения

$$y_i = \begin{cases} 0.5x_i(1-\cos w) & i = 0,1,2..m-1 \quad и \quad i = n-m, n-m+1, ..n-1 \\ x_i & m \le i \le n-m-1 \end{cases},$$

где $m = round\left(\frac{n}{10}\right)$, n – число элементов последовательности **X**

Triangle Window



Треугольное окно

ВП применяет треугольное окно для обработки входного сигнала **X**.
Если **y** представляет выходную последовательность, обработанную **треугольным окном** (Triangle {X}), то ВП получает ее значения с помощью выражения

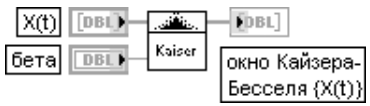
$$y_i = \begin{cases} x_i \frac{2i}{n} & 0 \le i \le \frac{n}{2} \\ x_i \frac{2(n-i)}{n} & \frac{n}{2} < i \le n-1 \end{cases} \quad для \quad четного \quad n,$$

$$y_i = \begin{cases} x_i \frac{2i}{n-1} & 0 \le i \le \frac{n-1}{2} \\ x_i \frac{2(n-i)}{n-1} & \frac{n+1}{2} \le i \le n-1 \end{cases} \quad для \quad нечетного \quad n,$$

где $i = \overline{0, n-1}$, n – число элементов последовательности **X**

Kaiser-Bessel Window

Окно Кайзера-Бесселя



ВП применяет окно Кайзера-Бесселя для обработки входного сигнала **X**.
Если **y** представляет выходную последовательность, обработанную **окном Кайзера-Бесселя** (Kaiser-Bessel {X}), ВП получает ее значения с помощью выражения

$$y_i = x_i \frac{I_0(\beta\sqrt{1.0 - a^2})}{I_0\beta},$$

где $a = \frac{i-k}{k}$, $k = \frac{n-1}{2}$, $I_0(\cdot)$ – модифицированная функция Бесселя нулевого порядка.
На рис. 4.32 приведены графики окон Кайзера-Бесселя при $\beta = 1 \div 5$

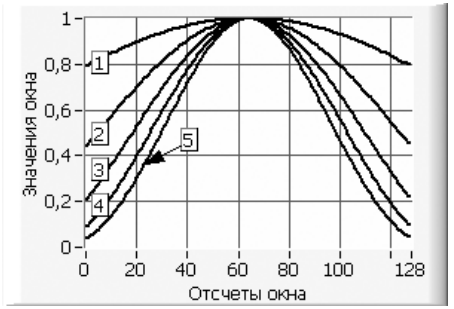


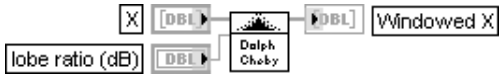
Рис. 4.32. Графики окон Кайзера-Бесселя при $\beta = 1 \div 5$

Gaussian Window

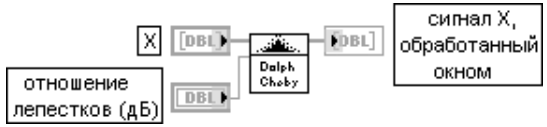
Гауссовское окно

ВП применяет асимметричное гауссовское окно для обработки входного сигнала **X**.
Вход **стандартное отклонение** (standard deviation) определяет стандартное отклонение гауссовского окна, нормированное к длине **X**. По умолчанию значение этого параметра равно 0,2

Chebyshev Window

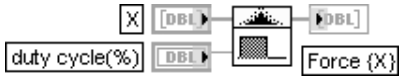


Окно Чебышева

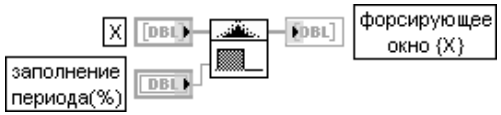


ВП применяет асимметричное окно Дольфа-Чебышева для обработки входного сигнала **X**.
Вход **отношение лепестков (дБ)** (lobe ratio (dB)) определяет отношение уровня главного лепестка к уровню бокового лепестка

Force Window



Форсирующее окно

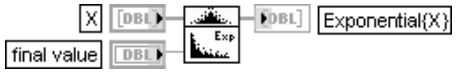


ВП применяет форсирующее окно для обработки входного сигнала **X**.
Если ***y*** представляет выходную последовательность, обработанную **форсирующим окном** (Force Window {X}), то ВП получает ее значения с помощью выражения

$$y_i = \begin{cases} x_i & \textit{если } 0 \le i \le d \\ 0 & \textit{иначе} \end{cases},$$

где $d = 0.01n(duty\ cicle)$, $i = \overline{0, n-1}$, n – число элементов последовательности ***X***

Exponential Window



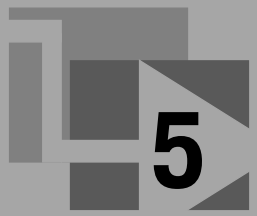
Экспоненциальное окно

ВП применяет экспоненциальное окно для обработки входного сигнала **X**.
Вход **конечное значение** (final value) ***f*** определяет постоянную времени экспоненциального окна $a = \ln(f)/(n-1)$,
где n – число отсчетов сигнала **X**.
Элементы выходной последовательности y_i, обработанные **экспоненциальным окном** (Exponential{X}), связаны с элементами входной последовательности x_i следующим соотношением: $y_i = x_i exp(ai)$,
где $i = \overline{0, n-1}$

Для четырех ВП из нижнего ряда палитры обработки весовыми окнами (рис. 4.28) вследствие однотипности их иконок и терминалов ввода/вывода далее приведены только расчетные выражения.

Название ВП	Расчетное выражение для сигнала на выходе
Модифицированное Бартлетта-Ханна (Modified Bartlett-Hanning)	$y_i = x_i\left(0{,}62 - 0{,}48\left\lvert\frac{i}{n} - 0{,}5\right\rvert + 0{,}38\cos\left(2\pi\left(\frac{1}{n} - 0{,}5\right)\right)\right)$
Бохмана (Bohman)	$y_i = x_i\left(\left(1 - \frac{\lvert i - n/2\rvert}{n/2}\right)\cos\left(\pi\frac{\lvert i - n/2\rvert}{n/2}\right) + \frac{1}{\pi}\sin\left(\pi\frac{\lvert i - n/2\rvert}{n/2}\right)\right)$
Парзена (Parzen)	$y_i = \begin{cases} x_i\left(1 - 6\left(\frac{i - n/2}{n/2}\right)^2 + 6\left(\frac{\lvert i - n/2\rvert}{n/2}\right)^3\right) & при\ \ 0 \le \left\lvert i - \frac{n}{2}\right\rvert \le \frac{n}{4} \\ 2x_i\left(\frac{1 - \lvert i - n/2\rvert}{n/2}\right)^3 & при\ \ \frac{n}{4} < \left\lvert i - \frac{n}{2}\right\rvert \le \frac{n}{2} \end{cases}$
Уэлча (Welch)	$y_i = x_i\left(1 - \left(\frac{i - n/2}{n/2}\right)^2\right)$

Функции генерации и измерения параметров осциллограмм



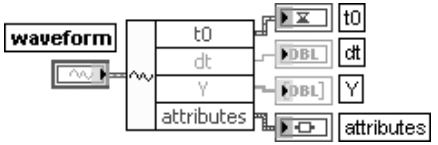
В большинстве функций сбора данных для их передачи используется специальный тип данных – осциллограмма. Тип данных **осциллограмма** (waveform) представляет особый тип кластера и используется в LabVIEW для работы с данными, зависящими от времени. Осциллограммы позволяют сохранить как значения данных, так и отметку о времени получения первого значения, значение интервала дискретизации и комментарии к данным. Подобно массивам и кластерам, осциллограммы можно складывать, вычитать, масштабировать и нормировать.

Вид основной палитры функций **Осциллограмма** (Waveform) показан на рис. 5.1а. На рисунках показаны следующие подпалитры функций: **Аналоговая осциллограмма** (Analog Waveform) (рис. 5.1б), **Цифровая осциллограмма** (Digital Waveform) (рис. 5.1в), **Цифровое преобразование** (Digital Conversion) (рис. 5.1г), **Ввод/вывод осциллограмм в/из файл(а)** (Waveform File I/O) (рис. 5.1д), **Генерация осциллограмм** (Waveform Generation) (рис. 5.1е) и **Измерения параметров осциллограмм** (Waveform Measurements) (рис. 5.1ж).

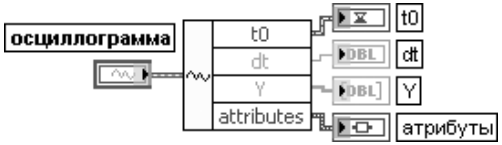
Более подробно функции работы с аналоговыми и цифровыми осциллограммами рассмотрены далее в разделе 5.1. Функциям генерации осциллограмм посвящен раздел 5.2, а функциям измерения параметров осциллограмм – раздел 5.3.

5.1. Базовые функции аналоговых и цифровых осциллограмм

Get Waveform Components



Получить компоненты осциллограммы



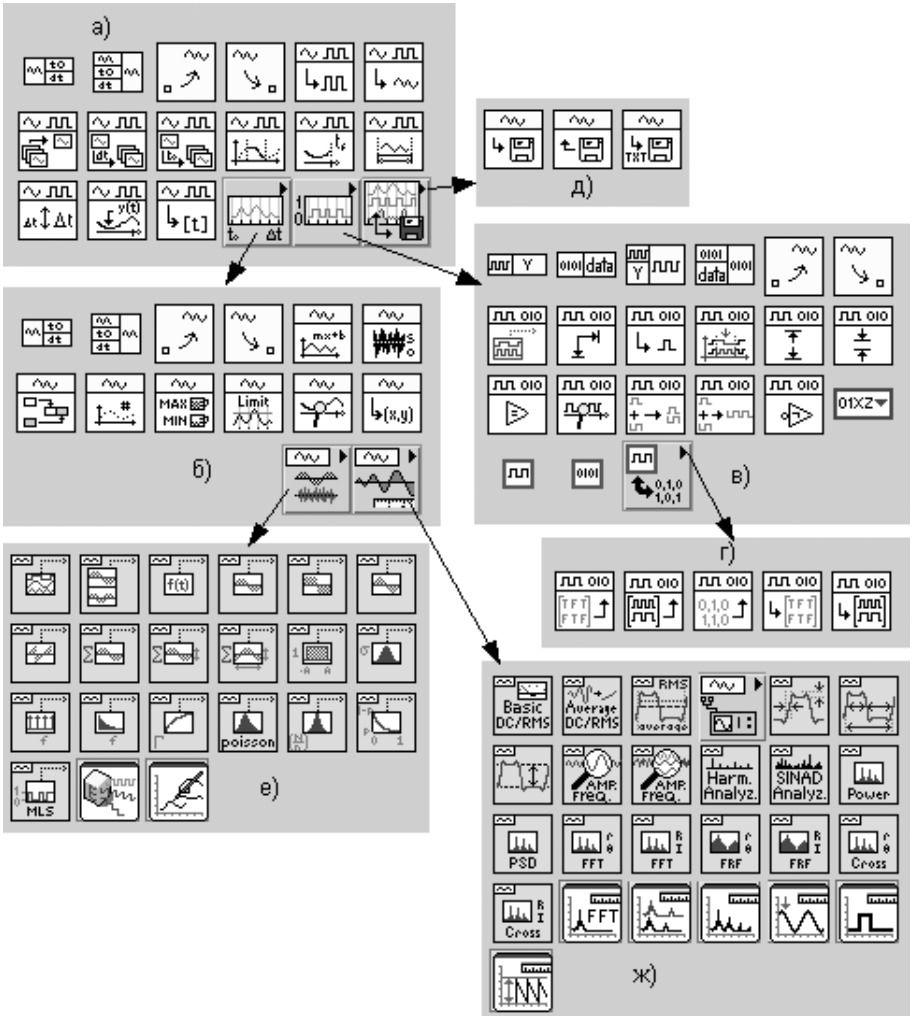


Рис. 5.1 Вид основной палитры (а) и дополнительных подпалитр (б)–(ж) функций работы с осциллограммами

Функция возвращает компоненты осциллограммы, определенные пользователем. Добавление выходов и определение компонентов осуществляется с помощью контекстного меню выходов.

Вход **осциллограмма** (waveform) представляет осциллограмму, из которой извлекаются компоненты.

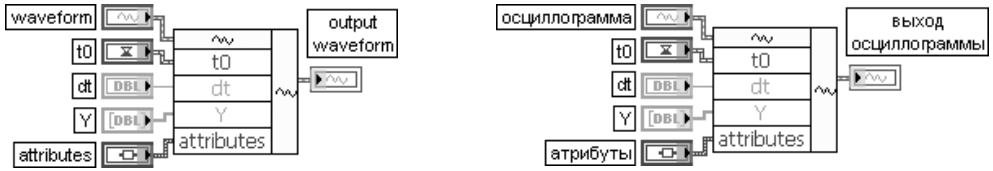
Выход **t0** представляет время (в соответствии с системными часами) получения первой точки массива **Y**.

Выход **dt** представляет интервал времени между точками массива **Y**.

Выход **Y** представляет значения данных осциллограммы.

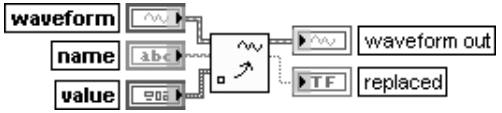
Выход **атрибуты** (attributes) позволяет передавать и выводить дополнительные данные

Build Waveform / Создать осциллограмму

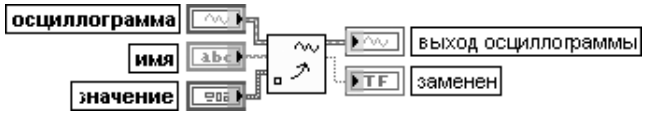


Функция создает осциллограмму или модифицирует существующую осциллограмму. ВП. Если вход **осциллограмма** (waveform) не подключен, то функция создает новую осциллограмму на основе подключенных компонентов. Если же вход **осциллограмма** подключен, то она модифицируется на основе подключенных компонентов

Set Waveform Attribute



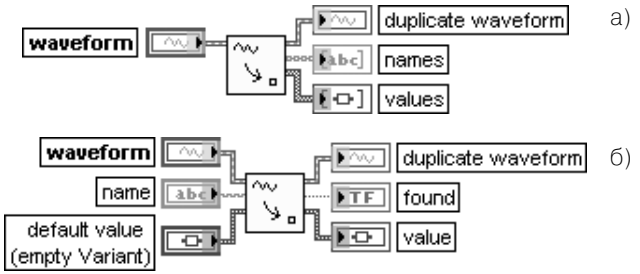
Установить атрибут осциллограммы



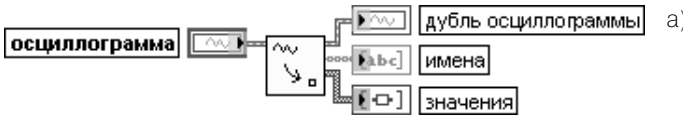
Функция добавляет или заменяет атрибут осциллограммы.

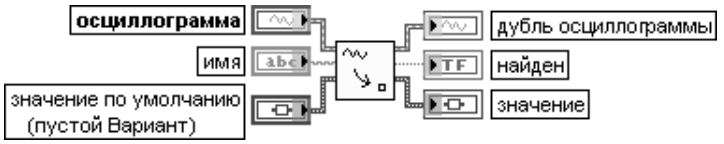
В качестве **значения** (value) атрибута могут использоваться данные произвольного типа. Если атрибут в **имени** (name) уже существует, функция заменяет его значение новым значением и устанавливает выход **заменен** (replaced) в состояние ИСТИНА. Если атрибут в имени не существует, функция создает новый атрибут

Get Waveform Attribute



Получить атрибут осциллограммы





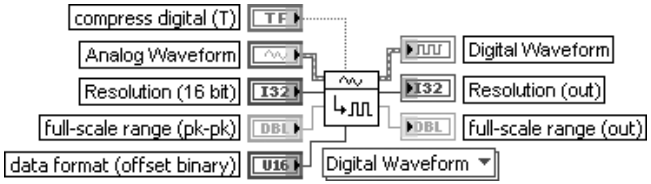
б)

Рис. 5.2. Варианты подключения функции
Получить атрибут осциллограммы

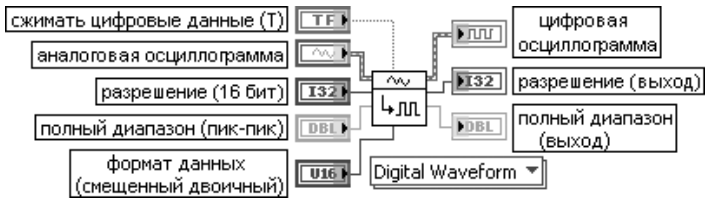
Функция извлекает **имена** (names) и **значения** (values) всех атрибутов или **значение** (value) единственного атрибута, в зависимости от способа подключения входа **имя** (name). Атрибуты могут быть именами каналов. Соединения отображают типы данных по умолчанию для этой полиморфной функции.
Эта функция имеет два режима работы в зависимости от того, подключен или нет вход **имя** (name). По умолчанию функция возвращает **имена** (names) всех атрибутов и их соответствующие **значения** (values) в виде одномерных массивов (рис. 5.2а). Если вход имя (name) подключен, то выход **имена** (names) изменяется на логический выход **найден** (found), выходы **значения** (values) изменяются на **значение** (value) (рис. 5.2б) и функция ищет только заданный атрибут. Если функция не находит заданный атрибут или она не может преобразовать атрибут в значение по умолчанию, то выход **найден** отображает значение ЛОЖЬ, а выход **значение** отображает содержимое входа **значение по умолчанию**.
В таблице приведен список атрибутов осциллограммы, устанавливаемых NI-DAQ

Имя	Атрибут	Тип данных	Допустимое значение	Описание
Номер устройства (Hardware Device Number)	NI_DeviceNumber	Строка	Любое значение	NI_DeviceNumber содержит номер устройства, формирующего осциллограмму
Имя канала	NI_ChannelName	Строка	Любое значение	NI_ChannelName содержит имя канала, расположенного в устройстве, формирующем осциллограмму
Единица данных	NI_UnitDescription	Строка	Вольты, ньютоны и подобные единицы	NI_UnitDescription содержит единицами измерения осциллограммы

Analog to Digital Waveform



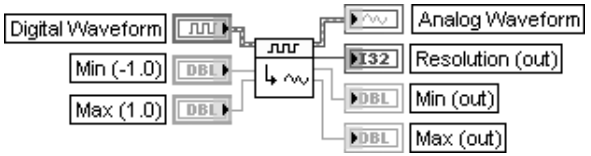
Аналоговую осциллограмму в цифровую



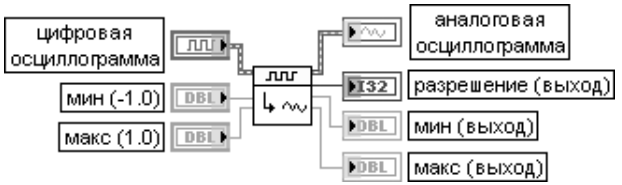
Этот полиморфный ВП преобразует аналоговую осциллограмму в цифровую осциллограмму или цифровые данные.
Вход **сжимать цифровые данные** (T) (compress digital) определяет опцию сжатия цифровой осциллограммы или цифровых данных.
Вход **разрешение** (Resolution) определяет число битов, представленных в цифровой осциллограмме или в цифровых данных.
Вход **полный диапазон (пик-пик)** (full-scale range (pk-pk)) задает полный диапазон от пика до пика или разность между минимумом и максимумом для цифровой осциллограммы или цифровых данных.
Вход **формат данных** (data format) определяет необходимое двоичное представление для цифровой осциллограммы или цифровых данных. По умолчанию устанавливается смещенный двоичный формат.

0	**Дополнительный** (2's complement) – использует формат дополнения до двух, который является общепринятым для представления двоичных чисел со знаком. Он отличается от смещенного двоичного инверсией старшего двоичного разряда
1	**Смещенный двоичный** (offset binary) – нижний предел отрицательных чисел представляется числом с нулями во всех двоичных разрядах, верхний предел положительных чисел – всеми единицами. Нулевое значение представляется единицей в старшем разряде
2	**Без знака** (Unsigned) – преобразует данные в двоичные данные без знака

Digital to Analog Waveform

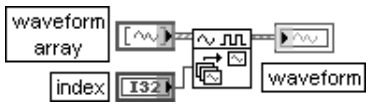


Цифровую осциллограмму в аналоговую



Полиморфный ВП преобразует цифровую осциллограмму в аналоговую.
Выход **разрешение** (Resolution) возвращает число битов, представленных в цифровой осциллограмме

Index Waveform Array



Выбрать осциллограмму из массива

ВП выбирает одну осциллограмму из массива аналоговых или цифровых осциллограмм по индексу массива или имени канала.

Данный полиморфный ВП можно использовать для выбора осциллограммы указанным способом для аналоговых осциллограмм двойной точности с плавающей запятой, целых 32-битовых или 16-битовых со знаком, комплексных двойной точности или цифровых осциллограмм. Выбор конкретной реализации зависит от типа данных, подключенных ко входам **массив осциллограмм** (waveform array) и **индекс** (index), а также от типа данных атрибута **Y** аналоговой осциллограммы

Copy Waveform dt

Копировать dt осциллограммы

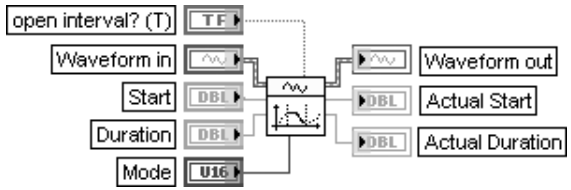
ВП заменяет все значения **dt** в массиве **входных осциллограмм** (Waveforms In) на значение **dt** осциллограммы, расположенной в этом массиве в позиции, заданной **индексом** (Index). Все типы осциллограмм, с которыми работает данная полиморфная функция, были перечислены выше

Align Waveform Timestamps

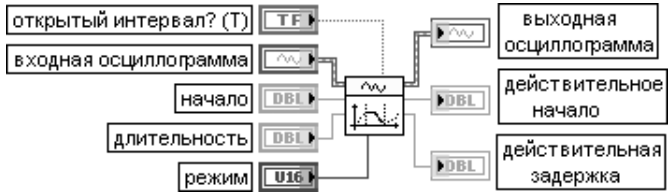
Выровнять начальное время осциллограмм

ВП заменяет все значения начального времени **t0** в массиве **входных осциллограмм** (Waveforms In) на значение **t0** осциллограммы, расположенной в этом массиве в позиции, заданной **индексом** (Index). Все типы осциллограмм, с которыми работает данная полиморфная функция, были перечислены выше

Get Waveform Subset



Получить часть осциллограммы



ВП извлекает часть осциллограммы, начинающуюся с заданного времени или индекса и имеющую заданную **длительность** (Duration). Данный полиморфный ВП можно использовать для извлечения части аналоговой или цифровой осциллограммы или набора цифровых данных. Выбор конкретной реализации зависит от типа данных, подключенных к **входной осциллограмме** (Waveform in).

Вход **открытый интервал?** (open interval?) определяет, находится ли извлекаемая часть осциллограммы в открытом или закрытом интервале. По умолчанию вход установлен в состояние ИСТИНА, что соответствует открытому интервалу. Например, если **t0** = 0, **dt** = 1, **Y** = {0, 1, 2}, **режим** (Mode) в состоянии **относительное время** (Relative Time), вход **начало** (Start) равен 0 и вход **длительность** (Duration) равен 2, то при открытом интервале на выходе вернутся значения {0, 1}. При закрытом интервале вернутся значения {0, 1, 2}.

Вход **режим** (Mode) устанавливает режим извлечения значений данных по заданному индексу или заданному времени.

0	**Индекс** (Index) (по умолчанию) – возвращает значение относительно определенного элемента данных осциллограммы
1	**Относительное время** (Relative Time) – возвращает значение по определенному времени относительно первой точки осциллограммы

Вход **начало** (Start) задает элемент данных или значение времени, с которого необходимо начать отсчет извлекаемой части осциллограммы.

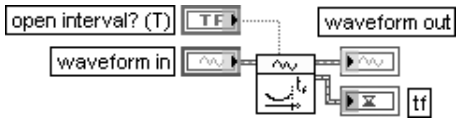
Вход **длительность** (Duration) определяет в зависимости от состояния входа **режим** (Mode) длительность извлекаемого набора данных или число элементов данных.

Выход **действительное начало** (Actual Start) отображает действительный элемент данных, с которого произошло извлечение части осциллограммы.

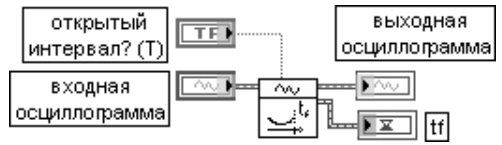
Выход **действительная длительность** (Actual Duration) отображает действительное число извлеченных элементов или действительное время сбора извлекаемых данных.

Значения на выходах **действительное начало** и **действительная длительность** зависят от величины параметра **dt** осциллограммы при установке входа **режим** в состояние **относительное время**

Get Final Time Value

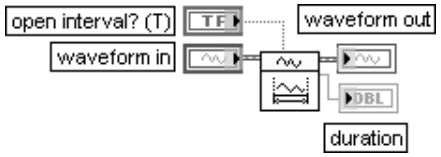


Получить время окончания осциллограммы

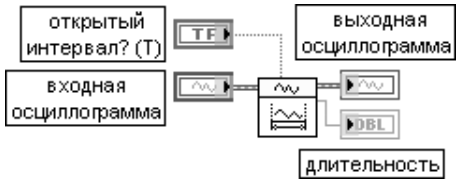


ВП возвращает время окончания **входной осциллограммы** (waveform in). Время окончания (time of final – **tf**) рассчитывается как сумма времени начала осциллограммы **t0** и ее **длительности** (duration)

Waveform Duration



Длительность осциллограммы



ВП рассчитывает **длительность** (duration) **входной осциллограммы** (waveform in), используя следующее выражение: **длительность** = (**число выборок** – 1) Ч **dt**. Блок-диаграмма ВП приведена на рис. 5.3. Число выборок осциллограммы определяется с помощью ВП **Число выборок осциллограммы**, рассмотренного ниже.

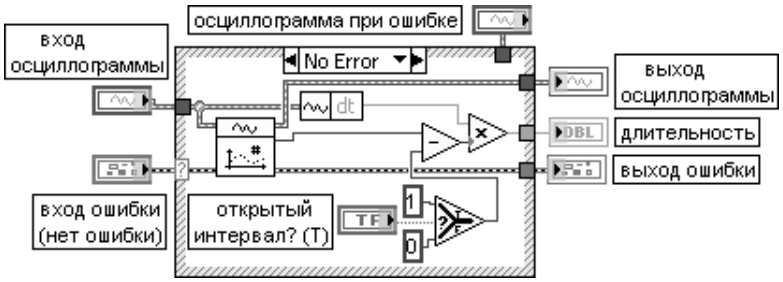
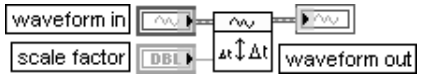
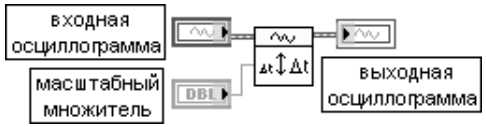


*Рис. 5.3. Блок-диаграмма ВП **Длительность осциллограммы***

Scale Delta t

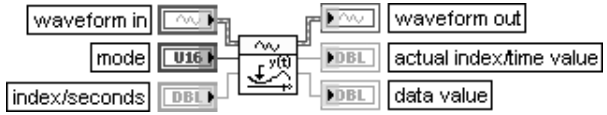


Изменить интервал дискретизации

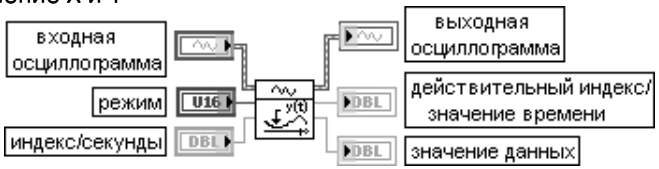


ВП умножает атрибут **dt** осциллограммы на заданный **масштабный множитель** (scale factor). С помощью такой операции изменяется частота выборок осциллограммы

Get XY Value

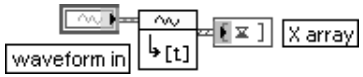


Получить значение X и Y

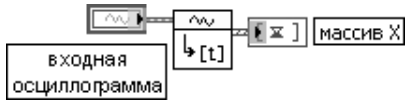


ВП возвращает значения ***X*** и ***Y*** **входной осциллограммы** или набора цифровых данных. Вход **индекс/секунды** (index/seconds) представляет номер элемента, который необходимо получить из **входной осциллограммы**, или значение времени этого элемента. По умолчанию значение входа равно 0. Если необходимо получить значение 200-го опроса (scan), необходимо ввести значение 199. Если же надо получить значение, соответствующее времени 100, то необходимо ввести значение 100.
Тип значения – индекс или время – определяется входом **режим** (mode). По умолчанию на этом входе установлено значение **индекс** (Index), что определяет выделение элемента **входной осциллограммы** по индексу. Если на входе **режим** установлено **относительное время** (Relative Time), то ВП проверяет вход **индекс/секунды** с целью определения числа содержащихся в нем целых значений **dt**. Если вход **индекс/секунды** не содержит целого числа **dt**, ВП использует ближайшее целое число **dt**. ВП возвращает ошибку, если **индекс/секунды** выходит за диапазон **входной осциллограммы**

Get Waveform Time Array



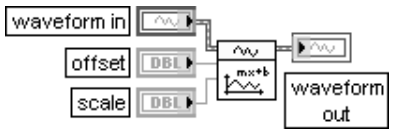
Получить массив отметок времени осциллограммы



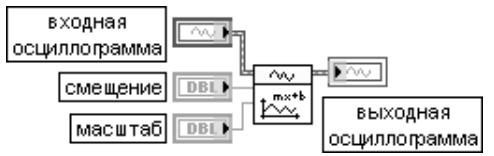
ВП создает массив отметок времени для каждого значения данных осциллограммы

Ниже в таблице приведены функции из подпалитры **Аналоговая осциллограмма** (Analog Waveform).

Waveform Scale and Offset

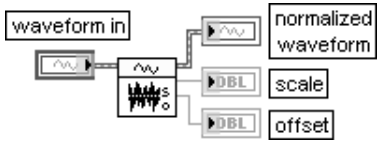


Масштабирование и смещение осциллограммы

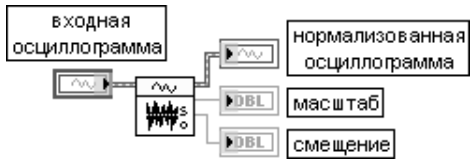


ВП масштабирует данные **входной осциллограммы**, используя выражение **выходная осциллограмма** = (**масштаб** * **входная осциллограмма** + **смещение**)

Normalize Waveform

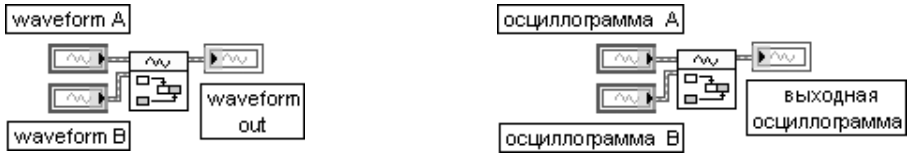


Нормализовать осциллограмму



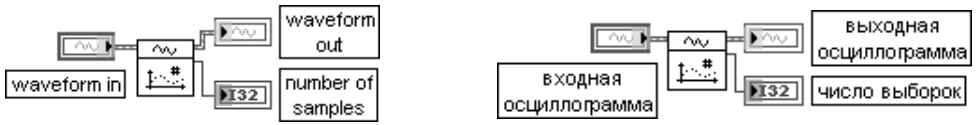
ВП определяет **масштаб** (scale) и **смещение** (offset), необходимые для преобразования данных **входной осциллограммы** к диапазону изменения от –1,0 до 1,0. Осциллограмма, приведенная к такому диапазону, передается на выход **нормализованная осциллограмма**

Append Waveforms / Добавить осциллограммы



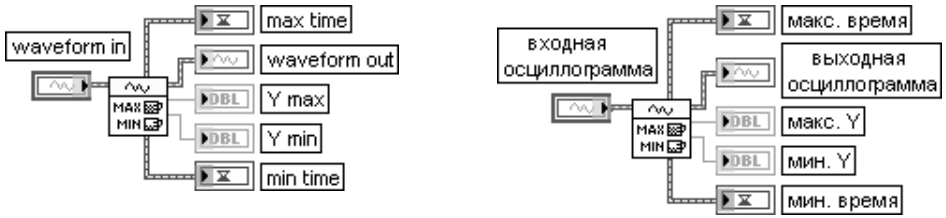
ВП добавляет осциллограмму **B** к концу осциллограммы **A**. Если частоты выборок не совпадают, то кластер ошибки возвращает ошибку. Время запуска осциллограммы **B** игнорируется. Тип данных атрибута **Y** осциллограммы **A** и осциллограммы **B** определяет используемую полиморфную реализацию ВП

Number of Waveform Samples / Число выборок осциллограммы



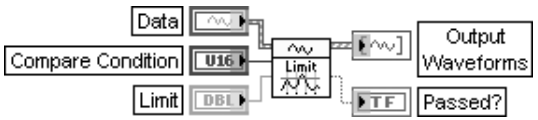
ВП возвращает число элементов осциллограммы

Waveform Min Max / Минимум и максимум осциллограммы

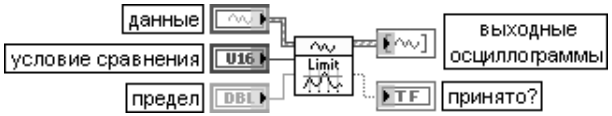


ВП определяет максимальное и минимальное значения осциллограммы и соответствующие значения времени

Waveform Scalar Limit Comparison



Сравнение осциллограммы со скалярным пределом

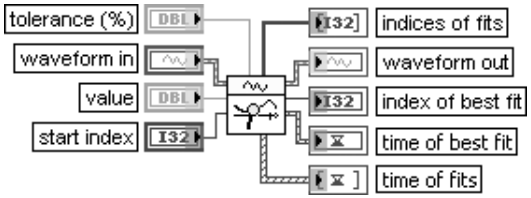


ВП сравнивает значения осциллограммы со скалярным пределом.
Вход **условие сравнения** (Compare Condition) отображает вариант сравнения максимального или минимального значений осциллограммы с числом, заданным на входе **предел** (Limit). Предусмотрены следующие варианты условия сравнения:

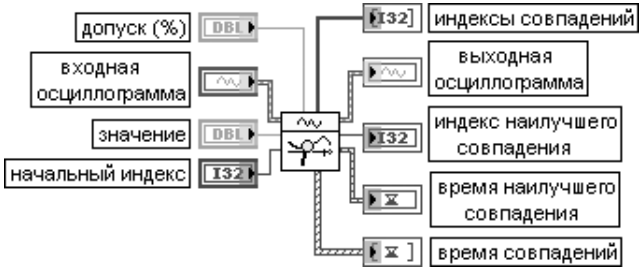
0	< – значение осциллограммы меньше **предела**
1	<= – значение в осциллограммы меньше или равно **пределу**
2	> – значение осциллограммы больше **предела**
3	>= (по умолчанию) – значение осциллограммы больше или равно **пределу**

Вход **предел** (Limit) определяет число, с которым производится сравнение наибольшего или наименьшего значения данных в осциллограмме.
Выход **принято?** (Passed?) устанавливается в состояние ЛОЖЬ, если какая-либо точка не соответствует заданному сравнению. Например, **принято?** будет иметь значение ЛОЖЬ, если точка осциллограммы меньше значения **предела** при установке на входе **условие сравнения** условия >.
Выход **выходные осциллограммы** (Output Waveforms) возвращает две осциллограммы в виде массива. Первая осциллограмма является копией входа **данные** (Data). Вторая осциллограмма содержит значения входа **данные**, если они удовлетворяют **условию сравнения**, и значение NaN в противном случае

Search Waveform



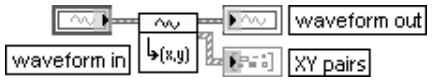
Поиск в осциллограмме



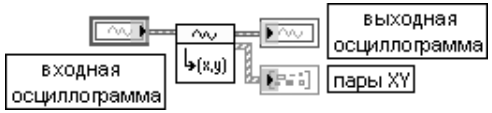
ВП возвращает значение времени элемента входной осциллограммы, соответствующего заданному **значению** (value).
Вход **допуск** (tolerance) задает величину допуска в процентах при поиске значения (value). По умолчанию значение **допуска** равно 0,01%.
Вход **значение** (value) определяет искомое значение. По умолчанию равно 0,0.
Вход **начальный индекс** (start index) определяет точку данных осциллограммы, с которой начинается поиск. По умолчанию значение начального индекса равно 0.
Выход **индексы совпадений** (indices of fits) отображает массив индексов всех значений, которые соответствуют входному **значению** (value) и критерию **допуска** (tolerance).
Выход **индекс наилучшего совпадения** (index of best fit) содержит индекс значения осциллограммы, которое в наибольшей степени соответствует входному **значению** (value).
Выход **время наилучшего совпадения** (time of best fit) содержит момент времени элемента осциллограммы, в наибольшей степени соответствующего входному **значению** (value).

Выход **время совпадений** (time of fits) отображает массив моментов времени всех значений, которые соответствуют входному **значению** (value) и критерию **допуска** (tolerance)

Waveform to XY Pairs



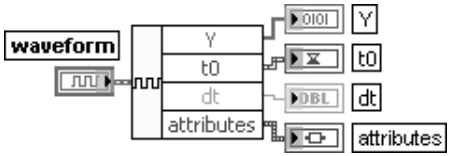
Осциллограмму в пары XY



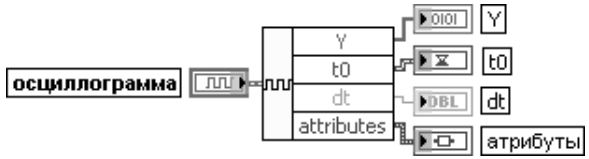
ВП создает массив значений данных и соответствующих им отметок времени.
Выход **пары XY** (XY pairs) содержит значения осциллограммы и соответствующие значения отметок времени

В следующей таблице рассмотрены функции из подпалитры **Цифровая осциллограмма** (Digital Waveform), в том числе функции из входящей в ее состав подпалитры **Цифровое преобразование** (Digital Conversion).

Get Waveform Components

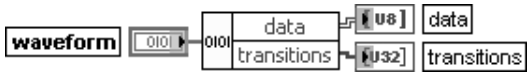


Получить компоненты осциллограммы

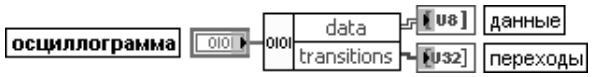


Функция возвращает компоненты цифровой осциллограммы, определенные пользователем. Добавление выходов и определение компонентов осуществляется с помощью контекстного меню выходов.
Вход **осциллограмма** (waveform) представляет цифровую осциллограмму, из которой извлекаются компоненты.
Выход **t0** представляет время (в соответствии с системными часами) получения первой точки массива **Y**.
Выход **dt** представляет интервал времени между точками массива **Y**.
Выход **Y** представляет значения данных цифровой осциллограммы.
Выход **атрибуты** (attributes) позволяет передавать и выводить дополнительные данные

Get Digital Data Components

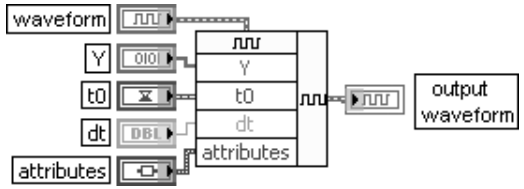


Получить компоненты цифровых данных

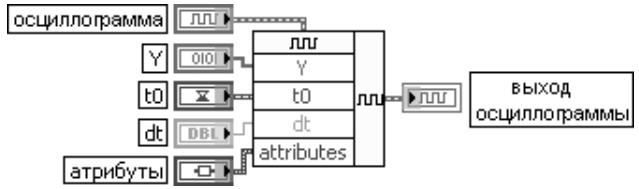


Функция возвращает компоненты цифровых данных, определенные пользователем. Добавление выходов и определение компонентов осуществляются с помощью контекстного меню выходов

Build Waveform

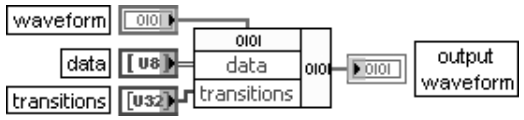


Создать осциллограмму

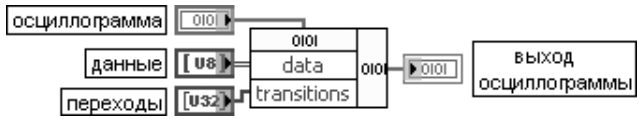


Функция создает цифровую осциллограмму или модифицирует существующую цифровую осциллограмму. Если вход **осциллограмма** (waveform) не подключен, то функция создает новую цифровую осциллограмму на основе подключенных компонентов. Если же вход **осциллограмма** подключен, то она модифицируется на основе подключенных компонентов

Build Digital Data

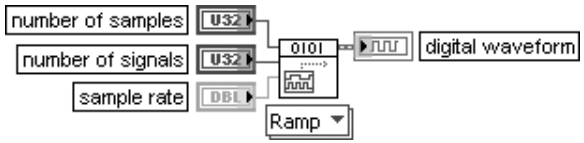


Создать цифровые данные

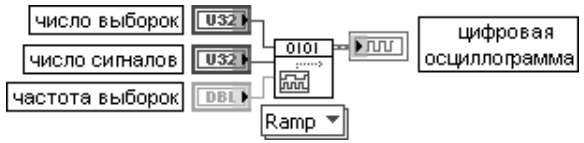


Функция создает цифровые данные или модифицирует существующие цифровые данные. Если вход **осциллограмма** (waveform) не подключен, то функция создает новые цифровые данные на основе подключенных компонентов. Если же вход **осциллограмма** подключен, то функция модифицирует существующие данные на основе подключенных компонентов

Digital Pattern Generator



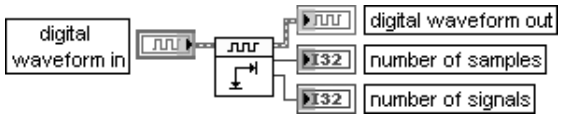
Генератор цифровых сигналов



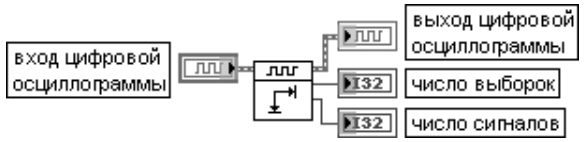
Эта полиморфная функция генерирует цифровой сигнал и возвращает его в виде цифровой осциллограммы.

В зависимости от выбора экземпляра ВП в составе осциллограммы генерируются **линейно нарастающие значения** (ramp), **«бегущие» значения** (marching values), **единственное значение** (single value), **случайные значения** (random) или **меандр** (toggle pattern). При генерации линейно нарастающих значений цифровая осциллограмма содержит набор двоичных сигналов, подобных сигналам на выходе двоичного счетчика, разрядность которого равна **числу сигналов** (number of signals)

Digital Size



Размер осциллограммы



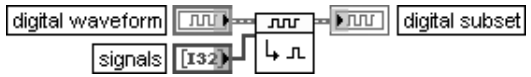
ВП возвращает число выборок и сигналов, содержащихся во входной цифровой осциллограмме.

Выход **число выборок** (number of samples) возвращает число выборок элементов данных, содержащихся в цифровой входной осциллограмме.

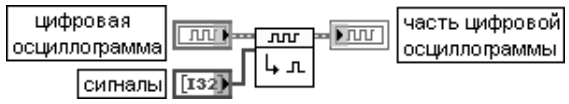
Выход **число сигналов** (number of signals) возвращает число сигнальных элементов, содержащихся в цифровой входной осциллограмме.

Например, если подключить данный ВП параллельно с индикаторами, показанными на рис. 5.7, то для входной осциллограммы он покажет, что число выборок равно 10, а число сигналов – 16. Для выходной осциллограммы число выборок также равно 10, а число сигналов – 8

Digital Signal Subset



Часть цифровой осциллограммы



ВП возвращает часть цифровой осциллограммы. Этот полиморфный ВП может использоваться для извлечения сигнала из цифровой осциллограммы или набора цифровых данных.

Вход **сигналы** (signals) передает одномерный массив чисел. Каждое число в массиве представляет номер сигнала. Если какое-либо значение на входе **сигналы** больше числа сигналов на цифровом входе, то ВП возвращает ошибку.

На выходе **часть цифровой осциллограммы** (digital subset) возвращается запрошенная часть **цифровой осциллограммы** (digital waveform).

На рис. 5.4 показан вид лицевой панели с индикаторами данных цифровых осциллограмм на входе (Y вх) и выходе (Y вых) ВП **Часть цифровой осциллограммы** при заданных значениях входа **сигналы**. Из рисунка видно, что при этом ВП фактически выделяет старшие байты входных данных.

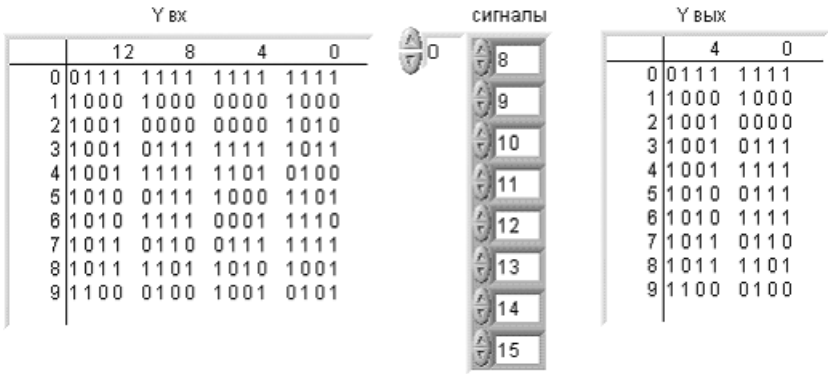
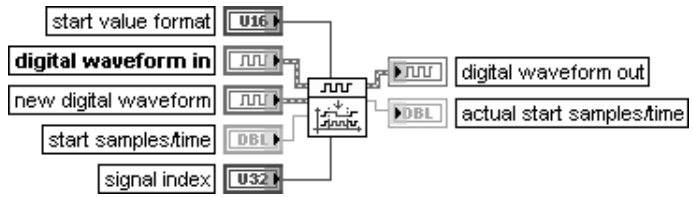
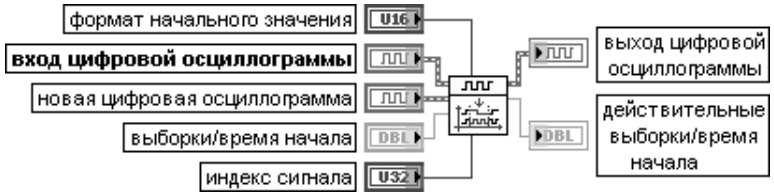


Рис. 5.4. Значения данных цифровых осциллограмм на входе (Y вх) и выходе (Y вых) ВП ***Часть цифровой осциллограммы*** *при заданных значениях входа* ***сигналы***

Replace Subset



Заменить часть осциллограммы



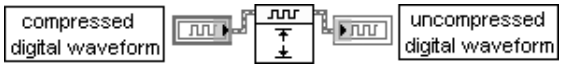
Этот полиморфный ВП заменяет часть входной цифровой осциллограммы или цифровых данных новыми данными цифровой осциллограммы (DWDT) или цифровыми данными (DTbl), начиная со значения, заданного на входе **выборки/время начала** (start samples/time).

Вход **формат начального значения** (start value format) определяет метод поиска.

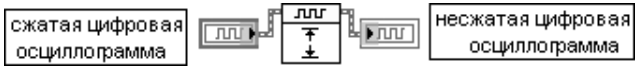
0	**Выборки** (Samples) (по умолчанию) – часть осциллограммы начинается с определенного элемента из набора данных на **входе цифровой осциллограммы**
1	**Относительное время** (Relative Time) – часть осциллограммы начинается со значения данных, имеющего заданное время

Вход **индекс сигнала** (signal index) определяет сигнал, с которого необходимо осуществлять замену данных. По умолчанию на этом входе установлено значение 0.

Uncompress Digital

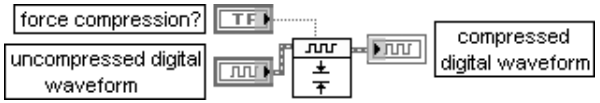


Извлечь цифровую осциллограмму

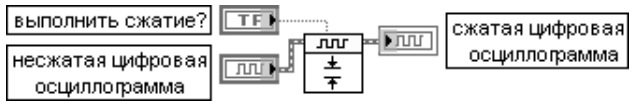


Функция извлекает цифровой сигнал, подаваемый на цифровой вход, и возвращает результат на цифровом выходе

Compress Digital



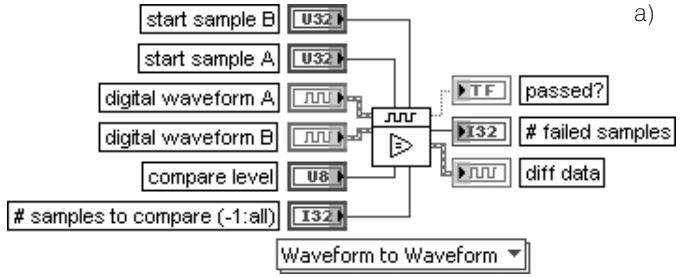
Сжать цифровую осциллограмму



Функция сжимает цифровой сигнал на цифровом входе и возвращает результат на цифровом выходе. Сжатие заключается в удалении повторяющихся значений с сохранением порядковых номеров оставшихся значений. Последнее значение сохраняется, несмотря на то что может совпадать с предыдущим.

Вход **выполнить сжатие?** (force compression?) определяет необходимость выполнения сжатия в случае, если данные оказываются сжатыми

Digital Comparison



Цифровое сравнение

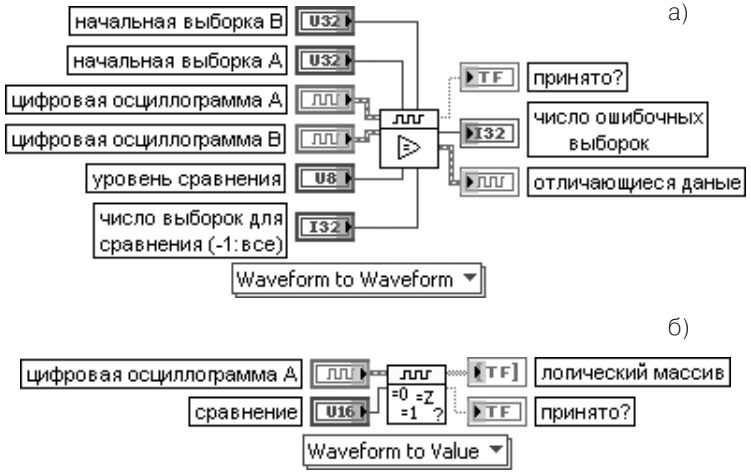


Рис. 5.5. Варианты установки функции ***Цифровое сравнение***

Этот полиморфный ВП сравнивает одну цифровую осциллограмму с другой или с определенным значением, а также набор одних цифровых данных с другим или с определенным значением. На рис. 5.5 (а) показан вариант установки функции при выборе вариантов сравнения осциллограммы с осциллограммой и таблицы с таблицей, на рис. 5.5. (б) – сравнения осциллограммы или таблицы с отдельным значением.

Для первого варианта вход **уровень сравнения** (compare level) определяет результаты сравнения, возвращаемые ВП:

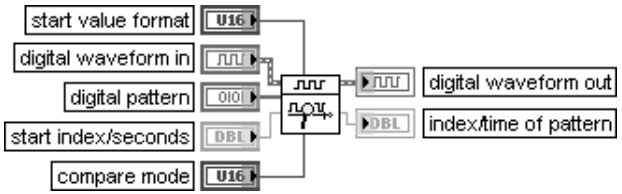
0	**Только выполнено/не выполнено (по умолчанию)** – определяет, что ВП возвращает только выход **принято?**, что улучшает скорость сравнения
1	**Выполнено/не выполнено и число ошибок** – определяет, что ВП возвращает выход **принято?** и число ошибочных выборок
2	**Выполнено/не выполнено, число ошибок и отличающиеся данные** – определяет, что ВП возвращает выход **принято?**, число ошибочных выборок и отличающиеся данные

Для второго варианта вход **сравнение** (comparison) определяет значение, с которым сравниваются цифровые данные, содержащиеся в цифровой осциллограмме:

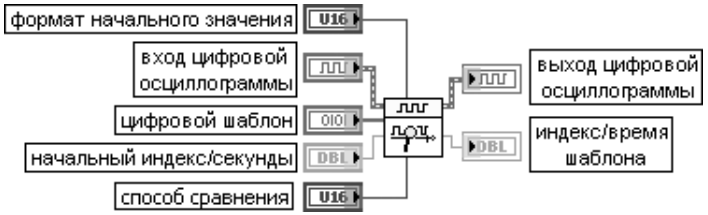
0	**Равно 0** (Equal 0) (по умолчанию)	6	**Равно T** (Equal T)
1	**Равно 1** (Equal 1)	7	**Равно V** (Equal V)
2	**Равно Z** (Equal Z)	8	**Равно 0 или L**(Equal 0 or L)
3	**Равно L** (Equal L)	9	**Равно 0 или H** (Equal 1 or H)
4	**Равно H** (Equal H)	10	**Не равно 0 или 1** (Not Equal 0 or 1)
5	**Равно X** (Equal X)	11	**Не равно 0, 1, L или H** (Not Equal 0, 1, L, or H)

Выход **число ошибочных выборок** (# failed samples) возвращает общее число несовпадающих выборок в сравниваемых осциллограммах или таблицах.
Выход **отличающиеся данные** (diff data) возвращает осциллограмму, показывающую результат сравнения осциллограмм. На этой осциллограмме нули индицируют совпадение выборок, единицы – несовпадение.
Выход **логический массив** (boolean array) служит индикатором места нахождения значений, удовлетворяющих условию **сравнения**.
Выход **принято?** (passed?) возвращает значение ИСТИНА, если для первого варианта сравнения имело место совпадение соответствующих элементов цифровых осциллограмм или данных, а для второго варианта – совпадение любого значения цифровой осциллограммы или данных с определенным значением, установленным на входе **сравнение**

Search for Digital Pattern



Искать по цифровому шаблону



ВП осуществляет поиск информации во входной цифровой осциллограмме по цифровому шаблону.
Вход **формат начального значения** (start value format) определяет метод поиска:

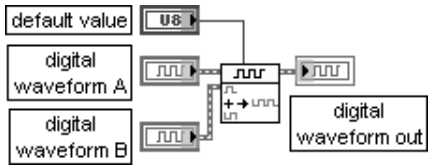
0	**Выборки** (Samples) (по умолчанию) – поиск начинается с заданного элемента из набора данных **входной цифровой осциллограммы** (digital waveform in)
1	**Относительное время** (Relative Time) – поиск начинается с заданного времени

Вход **цифровой шаблон** (digital pattern) определяет цифровой шаблон, по которому осуществляется поиск.
Вход **начальный индекс/секунды** (start index/seconds) определяет точку **входной цифровой осциллограммы**, с которой начинается поиск. Тип точки – индекс или время – определяется входом **режим**. По умолчанию на этом входе установлено значение 0, что определяет поиск с начала **входной цифровой осциллограммы**. Если на входе **режим** установлено **относительное время**, то ВП проверяет вход **начальный индекс/секунды** с целью определения содержащихся в нем числа целых значений **dt**. Если вход **начальный индекс/секунды** не содержит целого числа **dt**, то ВП использует ближайшее целое число **dt**. ВП возвращает ошибку, если **начальный индекс/секунды** выходит за диапазон **входной цифровой осциллограммы**.
Вход **способ сравнения** (compare mode) определяет способ обработки значений X для поиска:

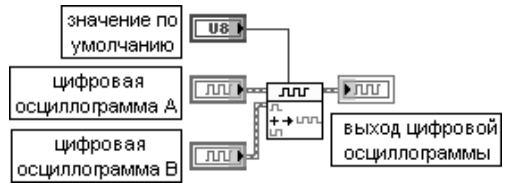
0	**Сравнивать состояния X (по умолчанию)** – значения X в **цифровом шаблоне** проверяются на совпадение с состояниями X во входных данных
1	**Игнорировать состояния X** – значения **X** в **цифровом шаблоне** являются групповыми символами и проверяются на совпадение с любым цифровым состоянием

Выход **индекс/время шаблона** (index/time of pattern) отображает первый индекс или значение времени **входной цифровой осциллограммы**, находящегося после **начального индекса/секунд** и соответствующего положению **цифрового шаблона**. Если на входе **режим** установлено **относительное время**, то **индекс/время шаблона** представляет время. Если на входе **режим** установлен **индекс**, то **индекс/время шаблона** представляет индекс

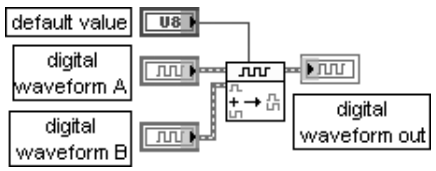
Append Digital Signals



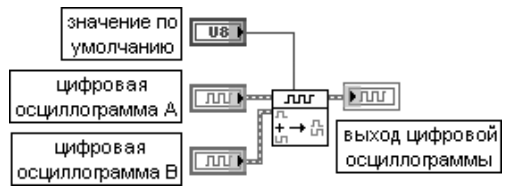
Добавить цифровые сигналы



Append Digital Samples

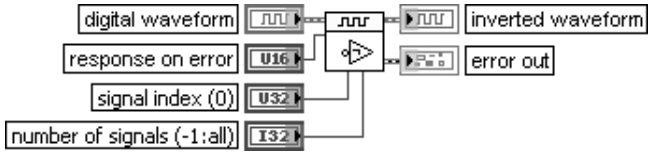


Добавить цифровые выборки

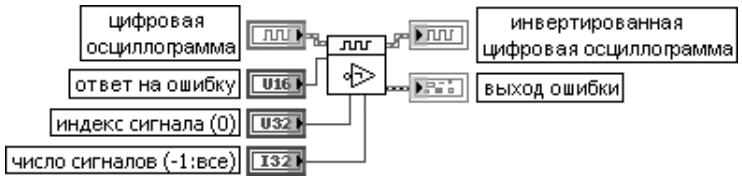


ВП добавляет **цифровую осциллограмму B** (digital waveform B) к концу **цифровой осциллограммы A** (digital waveform A). Если частоты дискретизации осциллограмм не совпадают, то на **выходе ошибки** (error out) возвращается предупреждение. Время запуска (trigger time) цифровой осциллограммы B игнорируется.
Этот полиморфный ВП может использоваться для добавления цифровой осциллограммы к другой цифровой осциллограмме или набора цифровых данных к другому набору цифровых данных

Invert Digital



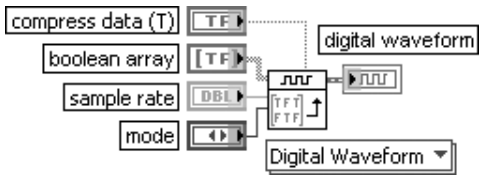
Инвертировать цифровую осциллограмму



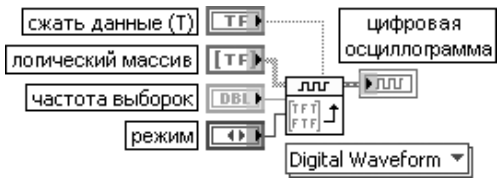
Функция инвертирует цифровые данные на цифровом входе так, что все нули становятся единицами, а все единицы – нулями.
Вход **ответ на ошибку** (response on error) определяет действие при обнаружении на цифровом входе значения, которое не является 0 или 1. Предусмотрены следующие ответы на ошибку:

0	**Предупреждать, но преобразовать** (Warn but Convert) (по умолчанию) – значение, не являющееся 0 или 1, остается неизменным, а на **выходе ошибки** (error out) возвращается предупреждение
1	**Прекратить** (Fail) – ВП возвращает пустой инвертированный массив и ошибку на **выходе ошибки**

Boolean Array to Digital

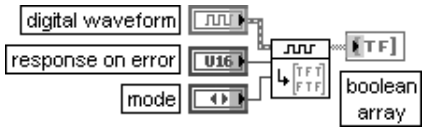


Логический массив в цифровую осциллограмму

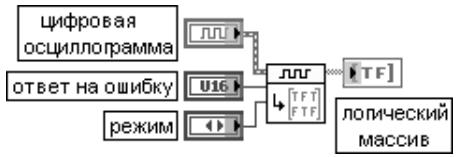


ВП преобразует двумерный логический массив в цифровую осциллограмму.
Вход **частота выборок** (sample rate) определяет частоту выборок в выходной цифровой осциллограмме.
Вход **режим** (mode) определяет, какой из битов располагается первым – самый младший (LSB) (по умолчанию) или самый старший (MSB).

Digital to Boolean Array

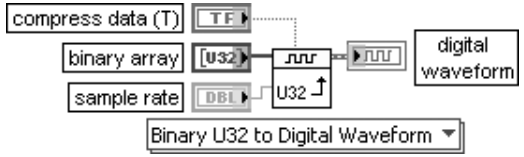


Цифровую осциллограмму в логический массив

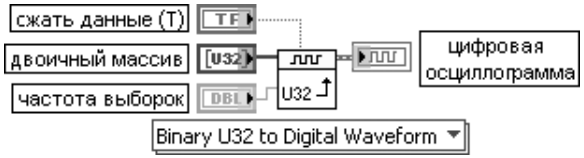


ВП преобразует цифровые данные на цифровом входе в двумерный логический массив

Binary to Digital

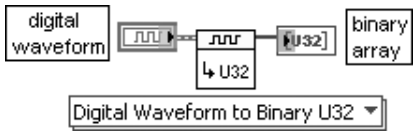


Двоичный массив в цифровую осциллограмму

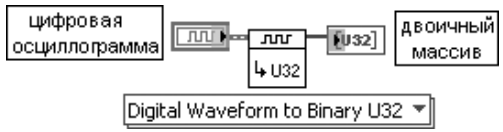


ВП преобразует двоичный массив целых чисел без знака в цифровую осциллограмму или набор цифровых данных

Digital to Binary

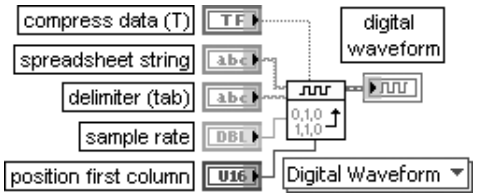


Цифровую осциллограмму в двоичный массив

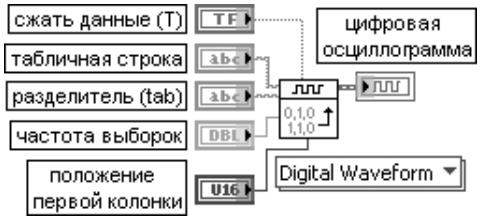


ВП преобразует цифровую осциллограмму или набор цифровых данных в двоичный массив целых чисел без знака

Spreadsheet String to Digital



Табличную строку в цифровое представление

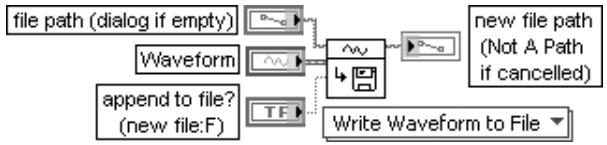


ВП преобразует табличную строку в цифровую осциллограмму или набор цифровых данных.

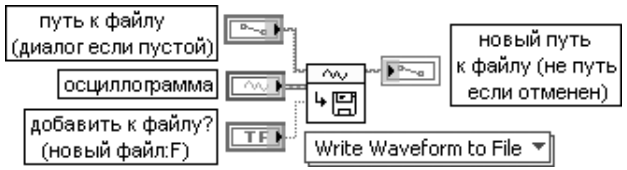
Вход **табличная строка** (spreadsheet string) содержит числовые или строковые значения, сгруппированные по строкам и столбцам. При этом столбцы (колонки) разделяются символами табуляции или запятыми, а строки – символами **конца строки** (end-of-line (EOL)). **Табличная строка** должна представлять разделенный набор ASCII символов, представляющих 8 поддерживаемых цифровых состояний: 0, 1, Z, L, H, X, T и V

Следующие три функции входят в состав подпалитры **Ввод/вывод осциллограмм в/из файл(а)** (Waveform File I/O).

Write Waveforms to File



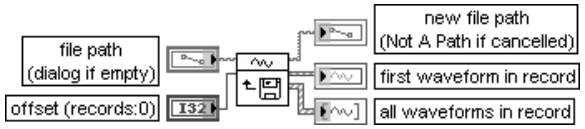
Записать осциллограмму в файл



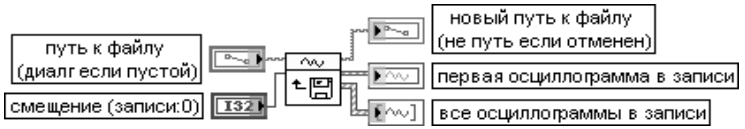
ВП создает новый файл или добавляет к существующему файлу, записывает заданное число записей в файл и закрывает файл, проверяя наличие ошибки. Каждая запись представляет массив осциллограмм. С помощью этого полиморфного ВП можно записывать в файл одномерный массив осциллограмм, двумерный массив осциллограмм или единичную осциллограмму. Тип данных, подключенных ко входу **осциллограмма** (Waveform), определяет используемую реализацию ВП.

Вход **добавить к файлу?** (append to file?) определяет возможность добавления данных к существующему файлу. При установке входа в состояние ИСТИНА ВП добавляет данные к существующему файлу. При установке в состояние ЛОЖЬ (по умолчанию) ВП заменяет данные в существующем файле. Если нет существующего файла, то ВП игнорирует значение **добавить к файлу** и создает новый файл

Read Waveform from File



Считать осциллограмму из файла



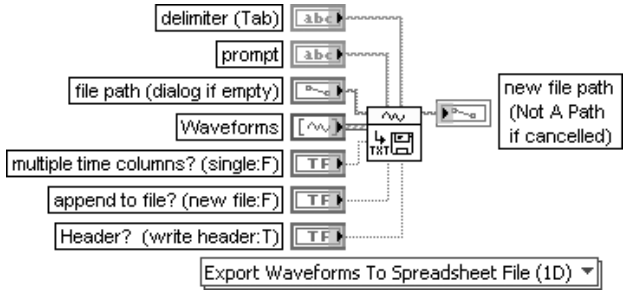
ВП открывает файл, созданный с помощью ВП **Записать осциллограмму в файл** (Write Waveforms to File), и читает одну запись файла. Каждая запись может содержать одну или более осциллограмм. ВП возвращает на отдельных выходах **все осциллограммы в записи** (all waveforms in record) и **первую осциллограммы в записи** (first waveform in record). Для получения всех записей из файла необходимо использовать данный ВП в цикле до появления конца файла.

Вход **смещение** (offset) определяет, насколько ниже начала файла будет находиться начало его считывания, выраженное числом байтов.

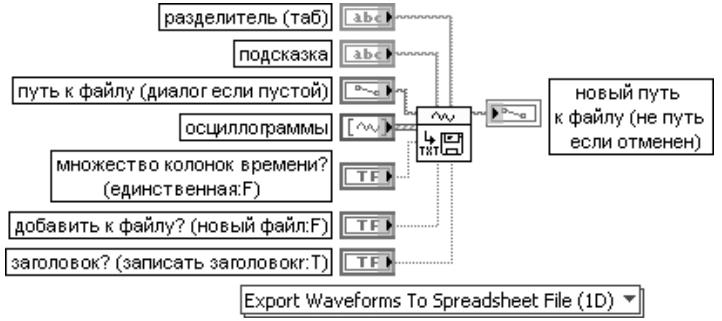
Выход **первая осциллограмма в записи** (first waveform in record) возвращает данные первой осциллограммы в записи.

Выход **все осциллограммы в записи** (all waveforms in record) возвращает данные всех осциллограмм в записи

Export Waveforms to Spreadsheet File



Экспортировать осциллограмму в табличный файл



ВП преобразует осциллограмму в текстовую строку и записывает строку в новый **файл потока байтов** или добавляет строку к существующему файлу. Дополнительно можно транспонировать данные. С помощью этого полиморфного ВП можно преобразовать одномерный массив осциллограмм, двумерный массив осциллограмм или единичную осциллограмму в текстовую строку. Тип данных, подключенных ко входу **осциллограммы** (Waveforms), определяет используемую реализацию ВП.

Вход **разделитель** (delimiter) представляет символ или строку символов, таких как табуляция (tabs), запятая (commas) и т. д., используемых для разделения полей в табличном файле. По умолчанию разделителем является символ табуляции.

Вход **подсказка** (prompt) содержит подсказку в файловом диалоговом окне в случае, если путь к файлу является пустым. По умолчанию содержит **Выбрать файл для записи** (Choose file to write).

Вход **путь к файлу** (file path) содержит имя пути к файлу. Если файл пустой (по умолчанию) или представляет <**не путь**> (<Not A Path>), то ВП отображает диалоговое окно, в котором можно выбрать файл. Ошибка 43 произойдет при отмене диалогового окна.

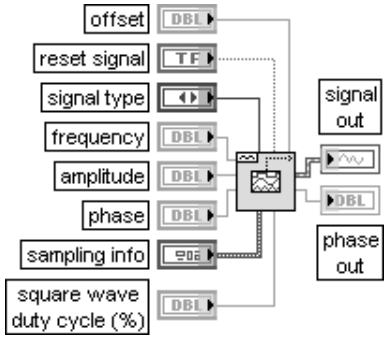
Вход **множество колонок времени?** (multiple time columns?) при установке в состояние ИСТИНА вызывает запись отдельных колонок для каждого отдельного канала, записываемого в файл. При установке в состояние ЛОЖЬ (по умолчанию) единственная колонка времени представляет один и тот же временной интервал для всех осциллограмм, записываемых в файл.

Вход **добавить к файлу?** (append to file?) определяет возможность добавления данных к существующему файлу. При установке входа в состояние ИСТИНА ВП добавляет данные к существующему файлу. При установке в состояние ЛОЖЬ (по умолчанию) ВП заменяет данные в существующем файле. Если нет существующего файла, то ВП игнорирует значение **добавить к файлу** (append to file?) и создает новый файл.

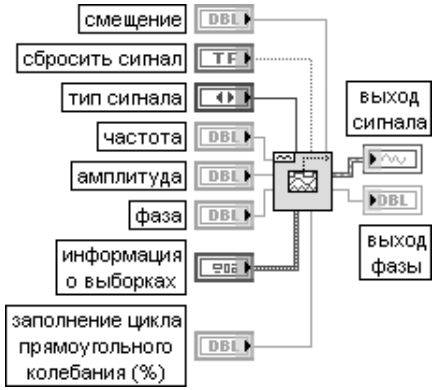
Вход **заголовок** (Header?), будучи установлен в состояние ИСТИНА (по умолчанию), вызывает транспонирование данных перед их преобразованием в строку

5.2. Функции генерации осциллограмм

Basic Function Generator



Основной генератор функций



ВП создает выходную осциллограмму, основанную на **типе сигнала** (signal type).

Вход **сбросить сигнал** (reset signal) в состоянии ИСТИНА устанавливает значение фазы равным значению на входе **фаза** и сбрасывает отметку времени в 0. По умолчанию значение входа – ЛОЖЬ.

Вход **тип сигнала** (signal type) определяет следующие варианты генерируемых осциллограмм:

0	1	2	3
Синусоидальная (Sine Wave) (по умолчанию)	**Треугольная** (Triangle Wave)	**Прямоугольная** (Square Wave)	**Пилообразная** (Sawtooth Wave)

Вход **частота** (frequency) определяет частоту осциллограммы в герцах. По умолчанию значение равно 10.

Вход **амплитуда** (amplitude) определяет амплитуду осциллограммы в вольтах. По умолчанию значение входа равно 1,0.

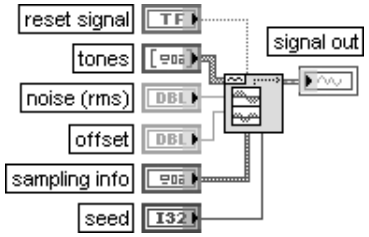
Вход **фаза** (phase) определяет начальную фазу осциллограммы в градусах. По умолчанию значение входа равно 0. ВП игнорирует **фазу** если на входе **сброс сигнала** установлено значение ЛОЖЬ.

Вход **информация о выборках** (sampling info) содержит следующую информацию о выборках:

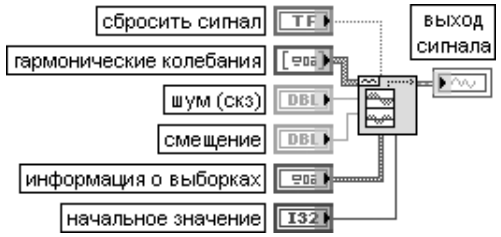
- **Fs** является частотой выборок, выраженной числом выборок в секунду. По умолчанию значение равно 1000;
- **#s** является числом выборок в осциллограмме. По умолчанию значение равно 1000.

Вход **заполнение цикла прямоугольного колебания** (square wave duty cycle) определяет в процентах время нахождения сигнала в высоком состоянии в течение одного периода. ВП использует данный параметр, только если вход **тип сигнала** задает прямоугольное колебание. По умолчанию на входе установлено 50%

Tones and Noise Waveform



Осциллограмма с гармоническими колебаниями и шумом



ВП генерирует осциллограмму, состоящую из суммы синусоидальных колебаний, шума и постоянного смещения.

Вход **сбросить сигнал** (reset signal) в состоянии ИСТИНА устанавливает фазу каждого гармонического колебания равной значению соответствующего элемента управления **фаза**, находящегося в массиве кластеров **гармонические колебания** (tones), начальное значение равно значению на соответствующем входе **начальное значение** (seed) ВП и сбрасывает отметку времени в 0. По умолчанию на входе установлено значение ЛОЖЬ.

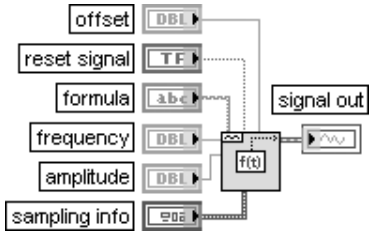
Вход **гармонические колебания** (tones) содержит следующие параметры для каждого синусоидального колебания:

- **частота** (frequency) – определяет частоту синусоидального колебания в герцах;
- **амплитуда** (amplitude) – определяет амплитуду синусоидального колебания;
- **фаза** (phase) – определяет начальную фазу синусоидального колебания в градусах.

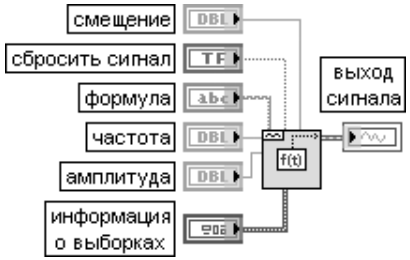
По умолчанию значение равно 0.

Вход **шум** (noise) определяет среднеквадратичное значение аддитивного гауссовского шума. По умолчанию значение равно 0,0

Formula Waveform



Расчет осциллограммы по формуле

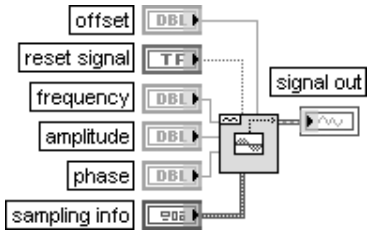


ВП создает выходную осциллограмму, используя формульную строку для определения используемой временной функции.
Вход **формула** (formula) задает представление, используемое для генерации **выходного сигнала** (signal out). По умолчанию задано выражение sin(w*t)*sin(2*pi(1)*10). В следующей таблице приведены допустимые имена переменных.

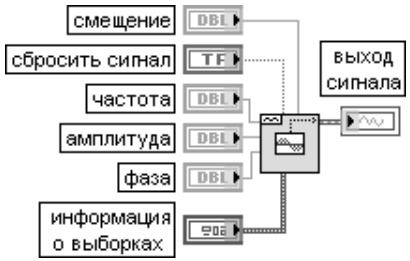
f	Частота, равная значению на входе **частота** (frequency)
a	Амплитуда, равная значению на входе **амплитуда** (amplitude)
w	2*pi*f
n	Число выборок выходного сигнала
t	Число истекших секунд
fs	Частота выборок, равная значению элемента **Fs** в кластере **информация о выборках** (sampling info)

Следующие четыре ВП сформированы на основе аналогичных ВП из палитры функций генерации сигналов и шумов, рассмотренных в разделе 4.1. В связи с этим пояснения к ним ограничены переводом наименований входов и выходов.

Sine Waveform

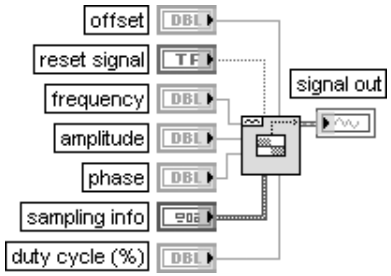


Осциллограмма с синусоидальным колебанием

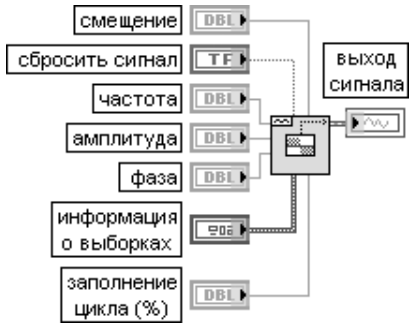


ВП генерирует осциллограмму, содержащую синусоидальное колебание. Основным элементом данного ВП является ВП **Синусоидальное колебание** (Sine Wave)

Square Waveform

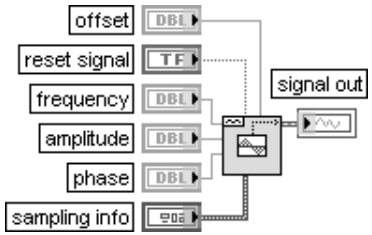


Осциллограмма с прямоугольным колебанием

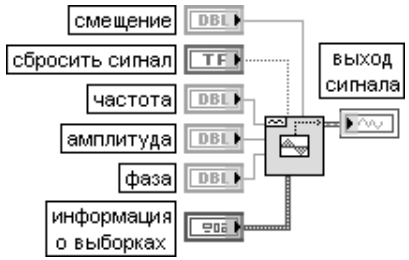


ВП генерирует осциллограмму, содержащую прямоугольное колебание. Основным элементом данного ВП является ВП **Прямоугольное колебание** (Square Wave)

Triangle Waveform

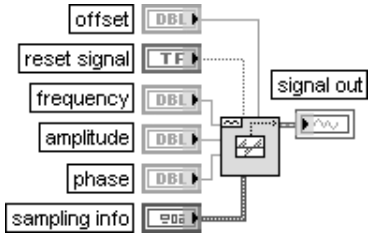


Осциллограмма с треугольным колебанием

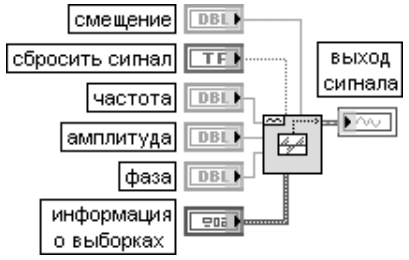


ВП генерирует осциллограмму, содержащую треугольное колебание. Основным элементом данного ВП является ВП **Треугольное колебание** (Triangle Wave)

Sawtooth Waveform



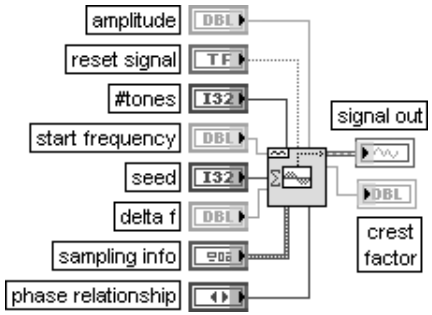
Осциллограмма с пилообразным колебанием



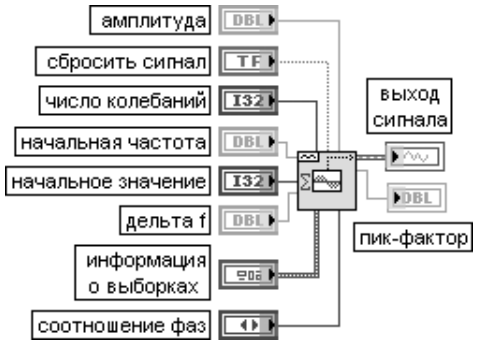
ВП генерирует осциллограмму, содержащую пилообразное колебание. Основным элементом данного ВП является ВП **Пилообразное колебание** (Sawtooth Wave)

Каждый из трех ВП, рассмотренных в следующих таблицах, выполняет формирование осциллограммы, представляющей сумму заданного числа синусоидальных колебаний с целым числом периодов и случайной начальной фазой. При этом основой первых двух ВП является третий ВП **Многотональный генератор** (Multitone Generator).

Basic Multitone



Основной многотональный генератор



ВП генерирует осциллограмму, представляющую сумму заданного числа синусоидальных колебаний с целым числом периодов и случайной начальной фазой.

Вход **амплитуда** (amplitude) определяет значение, к которому нормируется сумма всех гармонических колебаний, и соответственно наибольшее абсолютное значение, которое содержит осциллограмма. По умолчанию значение входа равно –1. Использование входа **амплитуда** полезно при передаче осциллограммы в канал аналогового выхода.

Вход **начальная частота** (start frequency) задает самую низкую частоту генерируемого гармонического колебания. Это значение должно быть целым частным от деления частоты дискретизации на число гармонических составляющих (Fs/#s). По умолчанию значение входа равно 10.

Установка на входе **начальное значение** (seed) значения > 0 вызывает инициализацию генератора шумовых выборок. **Начальное значение** игнорируется, если на входе **соотношение фаз** (phase relationship) установлено значение **линейное** (Linear).

Вход **дельта f** (delta f) задает интервал между соседними частотами гармонических колебаний. Значение **дельта f** должно быть целым частным от деления частоты дискретизации на число гармонических составляющих (Fs/#s). Если, например, **начальная частота** (start frequency) равна 100 Гц, значение **дельта f** равно десяти и число колебаний равно трем, то будет генерироваться колебание, содержащее частоты 100 Гц, 110 Гц и 120 Гц.

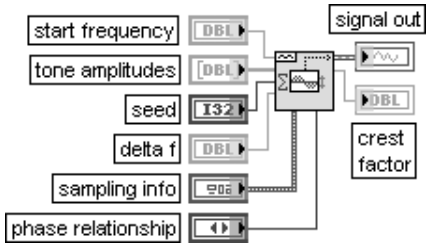
Вход **соотношение фаз** (phase relationship) определяет распределение фаз синусоидальных колебаний. Распределение фаз влияет на отношение пикового и среднеквадратичного значений всей осциллограммы. Предусмотрены следующие состояния входа:

- **Случайное** (Random) – каждое значение фазы выбирается случайно в диапазоне от 0 до 360 градусов;
- **Линейное** (Linear) – дает лучшее отношение пикового и среднеквадратичного значений, но может вызвать появление в сигнале периодических компонентов с периодом, равным длительности осциллограммы.

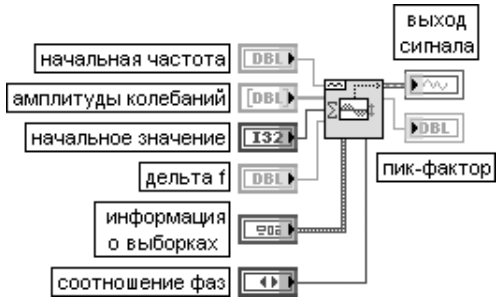
Выход **пик-фактор** (crest factor) равен отношению пикового напряжения и среднеквадратичного напряжения **выходного сигнала** (signal out).

Рассматриваемая осциллограмма в частотной области представляет последовательность импульсов, расположенных на заданных частотах. Синусоидальные колебания генерируются на основе информации о частоте и выборках. Фазы колебаний случайны, а амплитуды равны. Исходный массив масштабируется так, что наибольшее значение равно **амплитуде**. При формировании осциллограммы элемент **X0** всегда равен 0, а элемент **delta X** устанавливается равным 1/**Fs**

Basic Multitone with Amplitudes



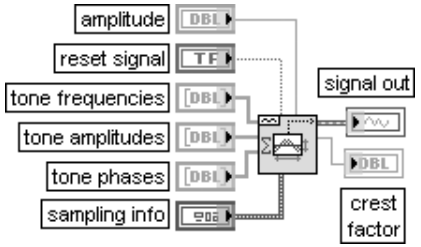
Основной многотональный генератор с заданными амплитудами



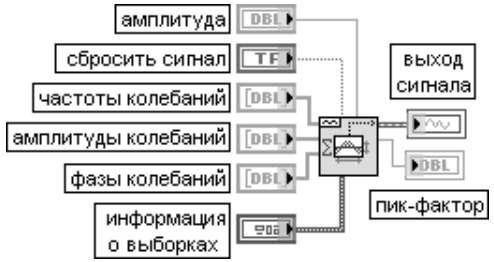
Данный ВП выполняет ту же функцию, что и рассмотренный выше ВП **Основной многотональный генератор** (Basic Multitone), отличаясь от него наличием входа **амплитуды колебаний** (tone amplitudes).

Вход **амплитуды колебаний** (tone amplitudes) представляет массив, в котором каждый элемент определяет амплитуду колебания, а размер массива определяет число генерируемых колебаний

Multitone Generator

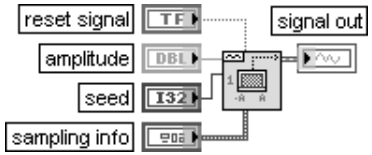


Многотональный генератор

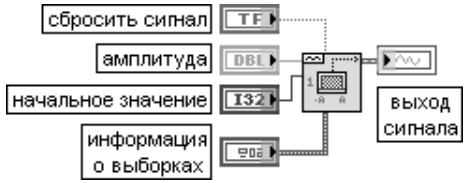


ВП генерирует осциллограмму, представляющую сумму синусоидальных колебаний с заданной частотой, амплитудой и фазой

Uniform White Noise Waveform

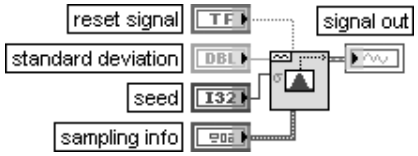


Осциллограмма с равномерным белым шумом



ВП генерирует псевдослучайный белый шум с равномерным амплитудным распределением в диапазоне [–**a**:**a**], где **a** представляет абсолютное значение входа **амплитуда** (amplitude)

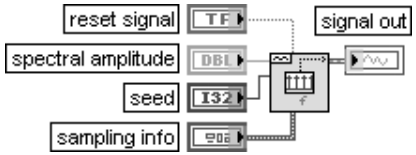
Gaussian White Noise Waveform



Осциллограмма с гауссовским белым шумом

ВП генерирует псевдослучайный гауссовский белый шум, имеющий статистические параметры (0,**s**), где **s** является абсолютным значением заданного стандартного отклонения

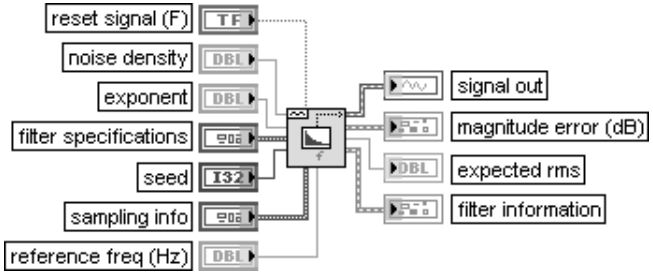
Periodic Random Noise Waveform



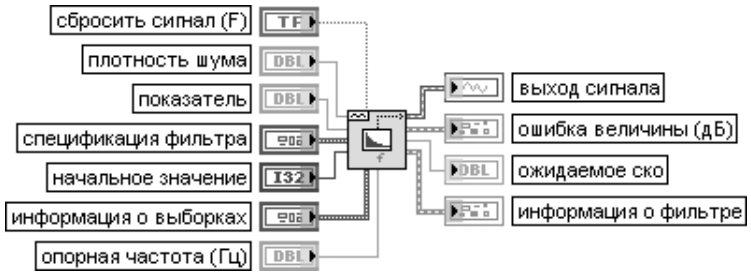
Осциллограмма с периодическим случайным шумом

ВП генерирует осциллограмму, содержащую периодический случайный шум

Inverse f Noise Waveform

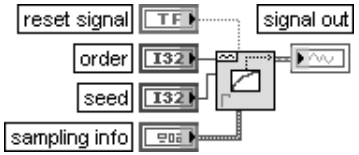


Осциллограмма шума 1/f



ВП генерирует осциллограмму шума, у которого спектральная плотность мощности обратно пропорциональна частоте в заданном спектральном диапазоне.
Генерация осуществляется путем пропускания белого гауссовского шума через цифровой фильтр, у которого квадрат частотной характеристики изменяется по закону 1/(частота^показатель). ВП **1/f фильтр** (Inverse f Filter), реализующий такой фильтр, был описан в разделе 3.1.4.
В связи с идентичностью части входов и выходов рассматриваемого ВП и ВП **1/f фильтр** далее приведено описание только отличающихся входов и выходов.
Вход **плотность шума** (noise density) определяет спектральную плотность ($В/\sqrt{Гц}$) идеального **1/f** шума на **опорной частоте** (reference freq). Действительный **1/f** шум аппроксимирует идеальный **1/f** шум в диапазоне частот, заданном **спецификацией фильтра** (filter specifications). Следовательно, действительная спектральная плотность **1/f** шума на **опорной частоте** будет находиться вблизи **плотности шума**, только если **опорная частота** находится в диапазоне частот, заданном **спецификацией фильтра**.
Выход **ожидаемое ско** (expected rms) возвращает ожидаемое среднеквадратичное значение генерируемой шумовой осциллограммы

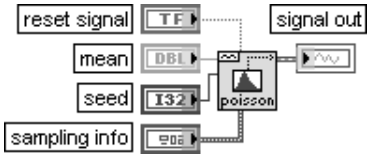
Gamma Noise Waveform



Осциллограмма гамма шума

ВП генерирует псевдослучайный набор значений, которые представляют интервалы времени ожидания заданного числа событий пуассоновского процесса с единичным средним. Вход **порядок** (order) определяет число событий. По умолчанию **порядок** равен 1

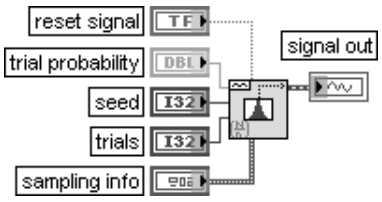
Poisson Noise Waveform



Осциллограмма пуассоновского шума

ВП генерирует псевдослучайную последовательность значений, которые представляют число событий ординарного Пуассоновского процесса, происходящих на заданном интервале, определенном величиной на входе **среднее** (mean). По умолчанию значение **среднего** равно 1,0

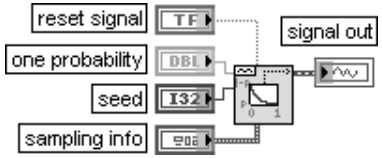
Binomial Noise Waveform



Осциллограмма биномиального шума

ВП генерирует псевдослучайную последовательность с биномиальным амплитудным распределением, значения которой представляют число реализаций событий, заданных вероятностью совершения событий и числом испытаний.
Вход **испытания** (trials) представляет число испытаний, выполняемых для каждого элемента **биномиального шума** (binomial noise). По умолчанию это число равно 1.
Вход **вероятность испытания** (trial probability) представляет вероятность того, что данное испытание будет успешным (1). По умолчанию значение входа равно 0,5

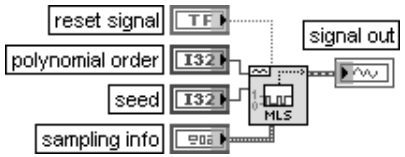
Bernoulli Noise Waveform



Осциллограмма шума Бернулли

ВП генерирует псевдослучайный шум из единиц и нулей.
Каждый элемент на выходе **шум Бернулли** рассчитывается с помощью способа, эквивалентного подбрасыванию монеты с вероятностью выпадения единицы, определяемой значением на входе **вероятность единицы** (ones probability). Если значение **вероятность единицы** равно 0,7, то каждый элемент **шум Бернулли** имеет 70% вероятности быть единицей и 30% вероятности быть нулем. По умолчанию значение входа равно 0,5.
Выход **шум Бернулли** содержит псевдослучайную последовательность с распределением Бернулли

MLS Sequence Waveform



Осциллограмма двоичной последовательности максимальной длины

ВП генерирует двоичную **последовательность максимальной длины** (maximum length sequence – MLS), используя деление по модулю для простого полинома, имеющего порядок, заданный на входе **порядок полинома** (polynomial order). Значение по умолчанию на входе **порядок полинома** равно 31

В состав палитры функций генерации осциллограмм входит Экспресс-ВП **Имитировать сигнал** (Simulate Signal), рассмотренный ниже.

Имитировать сигнал (Simulate Signal)
Экспресс-ВП имитирует синусоидальное, прямоугольное, треугольное, пилообразное колебание или шумовой сигнал.
Этот Экспресс-ВП использует функциональность следующих ВП: Осциллограмма с гауссовским белым шумом **(Gaussian White Noise Waveform),** Осциллограмма с периодическим случайным шумом **(Periodic Random Noise Waveform),** Осциллограмма с равномерным белым шумом **(Uniform White Noise Waveform),** Основной генератор

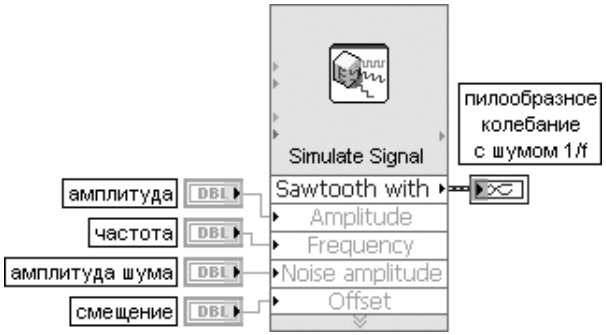


Рис. 5.6. Блок-диаграмма возможного подключения Экспресс-ВП

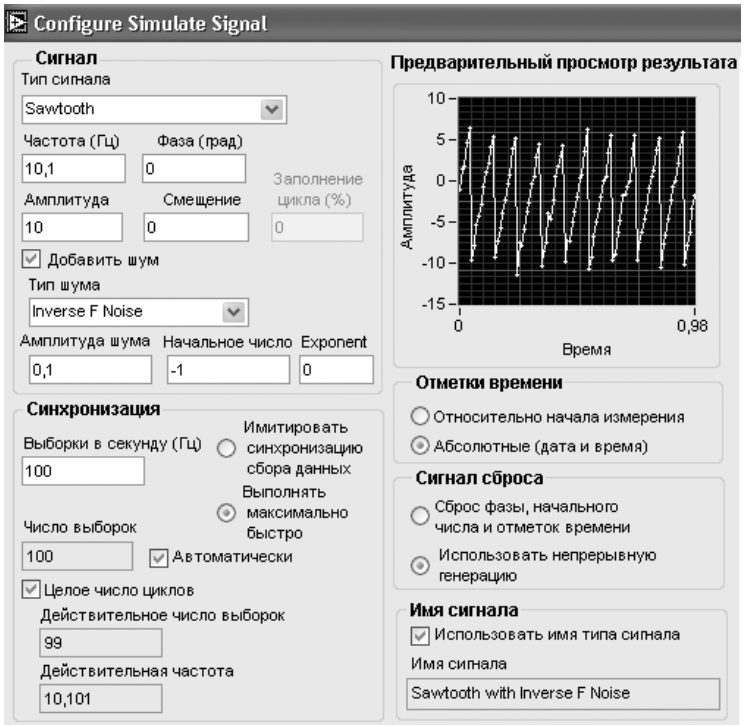
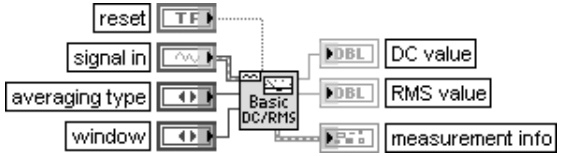


Рис. 5.7. Вид диалогового окна конфигурирования Экспресс-ВП ***Имитировать сигнал*** *(Simulate Signal)*

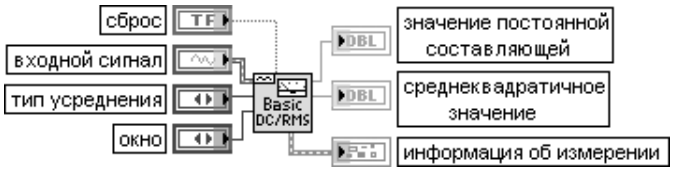
функций **(Basic Function Generator),** Осциллограмма с пилообразным колебанием **(Sawtooth Waveform),** Осциллограмма с синусоидальным колебанием **(Sine Waveform),** Осциллограмма с прямоугольным колебанием **(Square Waveform),** Осциллограмма с треугольным колебанием **(Triangle Waveform)**

5.3. Функции измерения параметров осциллограмм

Basic Averaged DC-RMS



Базовое измерение постоянной составляющей и среднеквадратичного значения с усреднением



ВП производит обработку окном входной осциллограммы или массива осциллограмм, рассчитывает значение постоянной составляющей и среднеквадратичное значение и усредняет их с аналогичными предыдущими значениями в соответствии с типом усреднения, установленным на входе **тип усреднения** (averaging type). Данный полиморфный ВП можно использовать для расчета значения постоянной составляющей и среднеквадратичного значения **единичного канала** (single channel) или **многоканальных данных** (multichannel data). Тип данных, подключенных ко входу **входной сигнал** (signal in), определяет используемую реализацию ВП.

Вход **сброс** (reset) производит сброс накапливаемых временных сигналов. Сброс обычно используется при экспоненциальном усреднении результатов измерений.

Вход **тип усреднения** (averaging type) определяет тип усреднения, используемый при измерениях. В данном ВП время интегрирования выбирается автоматически, исходя из длины записи.

0	**Линейный** (Linear) – время интегрирования равно длине записи
1	**Экспоненциальный** (Exponential) – постоянная времени равна половине длины записи

Вход **окно** (window) определяет окно, применяемое для обработки временной записи перед расчетом постоянной составляющей и среднеквадратичного значения. Набор окон соответствует перечню окон, приведенному в разделе 4.6.

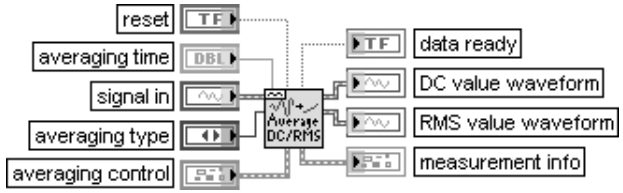
Выход **значение постоянной составляющей** (DC value) отображает измеренное значение **постоянной составляющей** в вольтах, если входной сигнал был задан в вольтах.

Выход **среднеквадратичное значение** (RMS value) отображает измеренное значение **среднеквадратичного значения** в вольтах, если входной сигнал был задан в вольтах.

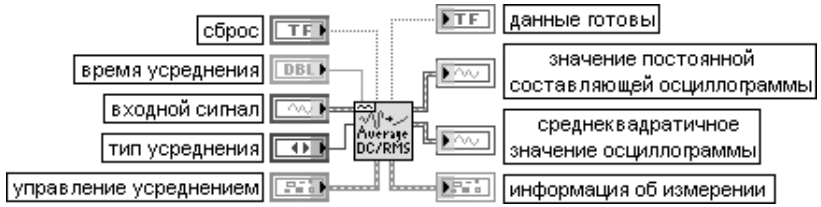
Выход **информация об измерении** (measurement info) возвращает информацию об измерении в виде кластера, в состав которого входят следующие элементы:

- **Неопределенность** (uncertainty) – зарезервировано для будущих приложений;
- **Предупреждение** (Warning) – устанавливается в состояние ИСТИНА, если во время обработки формируется предупреждение;
- **Комментарии** (comments) – содержит предупреждающее сообщение, если выход **предупреждение** устанавливается в состояние ИСТИНА

Averaged DC-RMS



Измерение постоянной составляющей и среднеквадратичного значения с усреднением



ВП рассчитывает значение постоянной составляющей и среднеквадратичное значение входной осциллограммы или массива осциллограмм.

Вход **время усреднения** (averaging time) определяет значение **dt** при оценке постоянной составляющей и среднеквадратичного значения в секундах. По умолчанию значение равно –1,00, что определяет установку времени усреднения равным длительности входного блока данных. При линейном усреднении каждая точка выходных данных является результатом усреднения входного сигнала на интервале, заданном на входе **время усреднения**. При экспоненциальном усреднении каждая точка выходных данных является результатом экспоненциального усреднения входного сигнала на интервале, заданном на входе **время усреднения** с постоянной времени, заданной на входе **постоянная времени экспоненты** (exp. time constant) в кластере **управление усреднением**.

Вход **управление усреднением** (averaging control) содержит дополнительные параметры, используемые для более полного управления измерением постоянной составляющей и среднеквадратичного значения. В состав кластера входят следующие элементы:

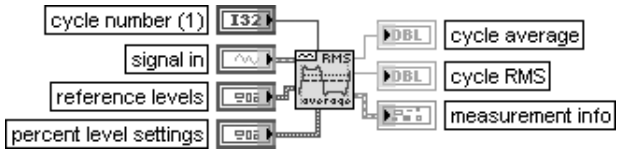
- **окно для постоянной составляющей** (window for DC) и **окно для среднеквадратичного значения** (window for RMS) идентичны входу **окно** (window) рассмотренного выше ВП;
- **выходная функция** (output function) устанавливает тип выполняемых измерений;

При необходимости измерения только постоянной составляющей или только среднеквадратичного значения установка соответствующей **выходной функции** увеличивает скорость вычислений. Предусмотрены следующие варианты данного входа:

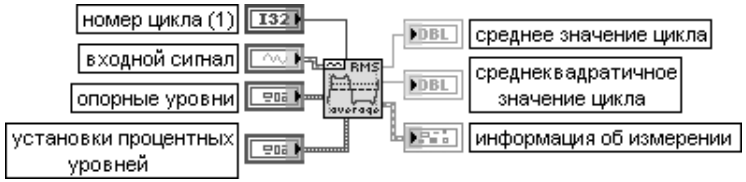
0	1	2
Только постоянная составляющая (DC only)	**Только среднеквадратичное значение** (RMS only)	**Постоянная составляющая и среднеквадратичное значение** (DC and RMS)

- **Постоянная времени экспоненты** (exp. time constant) определяет постоянную времени измерения постоянной составляющей и среднеквадратичного значения. Установка значения –1,00 на данном входе соответствует выбору постоянной времени на уровне половины длительности входного блока данных.
- **Игнорировать входную отметку времени** (Ignore input time stamp) устанавливается в состояние ИСТИНА для отключения проверки непрерывности использования значений **t0**

Cycle Average and RMS



Среднее и среднеквадратичное значение цикла



ВП возвращает среднее и среднеквадратичное значения выбранного цикла периодической осциллограммы или массива периодических осциллограмм.

Вход **номер цикла** (cycle number) определяет цикл или период периодического сигнала, по которому производится измерение.

Вход **опорные уровни** (reference levels) определяет верхний, средний и нижний опорные уровни осциллограммы. **Опорные уровни** используются для определения измерительных интервалов одного полного цикла. Кластер **опорные уровни** содержит следующие элементы:

- **верхний опорный уровень** (high ref level) – определяет верхний опорный уровень осциллограммы в процентах (по умолчанию) или абсолютных единицах. После того как сигнал пересекает **средний опорный уровень** (mid ref level) в направлении возрастания, он должен пересечь **верхний опорный уровень** перед тем, как следующее пересечение среднего уровня спадающим сигналом может быть подсчитано;
- **средний опорный уровень** (mid ref level) – определяет средний опорный уровень осциллограммы в процентах (по умолчанию) или абсолютных единицах. Интервал между последовательными пересечениями **среднего опорного уровня** в направлении возрастания определяет один цикл или период осциллограммы. По крайней мере, одно пересечение верхнего/нижнего опорного уровня должно отделять каждое пересечение **среднего опорного уровня**;
- **нижний опорный уровень** (low ref level) – определяет нижний опорный уровень осциллограммы в процентах (по умолчанию) или абсолютных единицах. После того как сигнал пересекает **средний опорный уровень** (mid ref level) в направлении спада, он должен пересечь **нижний опорный уровень** перед тем, как следующее пересечение среднего уровня возрастающим сигналом может быть подсчитано;
- **единицы опорных уровней** (ref units) – определяет выражение **верхнего**, **среднего** и **нижнего опорных уровней** в процентах от полного диапазона осциллограммы (по умолчанию) или в абсолютных единицах.

Вход **установки процентных уровней** (percent level settings) определяет метод, используемый для определения верхнего и нижнего уровней состояния осциллограммы. Вход **установки процентных уровней** определяет опорные уровни при выборе значения **проценты** в элементе управления **единицы опорных уровней** (ref units) кластера **опорные уровни**, в ином случае этот вход игнорируется. В состав кластера **установки процентных уровней** входят следующие элементы:

- **метод** (method) определяет метод вычисления верхнего и нижнего уровней состояния осциллограммы:

0	**Гистограммный** (histogram) – возвращает уровни интервалов гистограммы с максимальным числом событий в верхней и нижней областях осциллограммы. Верхняя и нижняя области осциллограммы включают значения выше и ниже 40% соответственно относительно диапазона от пика до пика
1	**Пиковый** (peak) – ищет максимальный и минимальный уровни по всей осциллограмме
2	**Автоматический выбор** (по умолчанию) – определяет интервалы гистограммы, соответствующие верхнему и нижнему уровням и содержащие каждый более 5% общего числа событий

- **размер гистограммы** (histogram size) определяет число интервалов гистограммы, используемых для определения верхнего и нижнего уровней состояния осциллограммы. **Размер гистограммы** игнорируется при выборе пикового метода;
- **метод гистограммы** (histogram method) определяет способ расчета верхнего и нижнего уровней состояния осциллограммы. В текущей версии установлен только единственный **режим** (mode).

Выход **среднее цикла** (cycle average) отображает среднее значение одного полного периода входной осциллограммы. Среднее вычисляется с помощью следующего выражения:

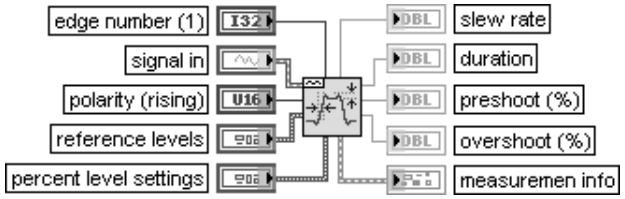
$$\textit{среднее} = \frac{1}{\textit{число точек}} \sum_{i \subset \textit{цикл}} \textit{осциллограмма}[i],$$

где i отображает выборки осциллограммы, попадающие в один период, заданный **номером цикла** (cycle number), **число точек** определяется с помощью следующего выражения: **число точек** = целая часть (**период**/**dt** + 0,5), где **dt** является интервалом дискретизации.

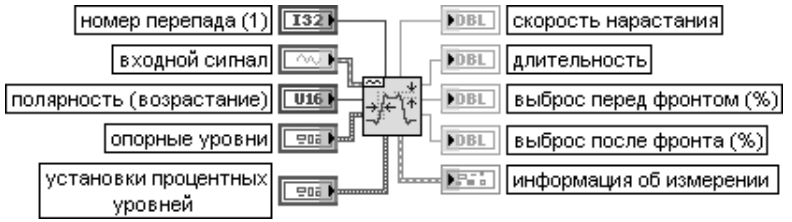
Выход **среднеквадратичное значение цикла** (cycle RMS) рассчитывается с помощью следующего выражения:

$$RMS_{\textit{цикл}} = \sqrt{\frac{1}{\textit{число точек}} \sum_{i \subset \textit{цикл}} \textit{осциллограмма}[i]^2}$$

Transition Measurements



Измерения перепадов



ВП принимает входной сигнал отдельной осциллограммы или массива осциллограмм и измеряет длительность переднего или заднего фронта, скорость нарастания, величину выброса перед и после фронта выбранного положительного или отрицательного перепада в каждой осциллограмме.

Вход **номер перепада** (edge number) определяет номер измеряемого перепада.

Входной сигнал (signal in) должен содержать число перепадов, по крайней мере равное **номеру перепада** с направлением перепада, соответствующим заданной **полярности**. Длительность положительного перепада определяется как интервал между соседними точками пересечения сигналом **нижнего** и **верхнего опорных уровней**. Длительность отрицательного перепада определяется как интервал между соседними точками пересечения сигналом **верхнего** и **нижнего опорных уровней**.
Вход **полярность** (polarity) определяет направление перепада с целью измерения его как возрастающего (по умолчанию) или как спадающего.
Выход **скорость нарастания** (slew rate) представляет оценку скорости изменения сигнала в области перепада между **верхним** и **нижним опорными уровнями**. **Скорость нарастания** определяется с помощью следующего выражения:

$$\textit{скорость нарастания} = \frac{\textit{верхний опорный уровень} - \textit{нижний опорный уровень}}{\textit{длительность перепада}},$$

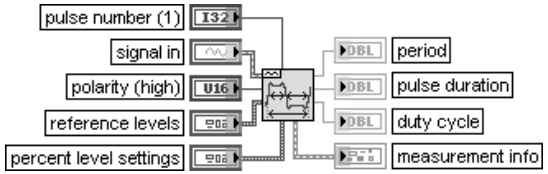
где **нижний** и **верхний опорные уровни** заданы в абсолютных единицах.
Выход **длительность** (duration) представляет интервал времени между точками пересечения осциллограммой **нижнего** и **верхнего опорных уровней** при установке на входе **полярность** варианта возрастания сигнала. При изменении полярности длительность определяется как интервал времени между точками пересечения указанных уровней в обратном порядке.
Выход **выброс перед фронтом** (**%**) (preshoot) содержит оценку локального минимума (максимума), предшествующего возрастающему (спадающему) перепаду, выраженную в процентах от амплитуды сигнала, определенной гистограммным методом. Если вход **полярность** определяет спадающий перепад, то **выброс перед фронтом** рассчитывается по следующему выражению:

$$\textit{выброс перед фронтом} = 100\frac{(\textit{локальный максимум} - \textit{верхний уровень состояния})}{\textit{амплитуда}}.$$

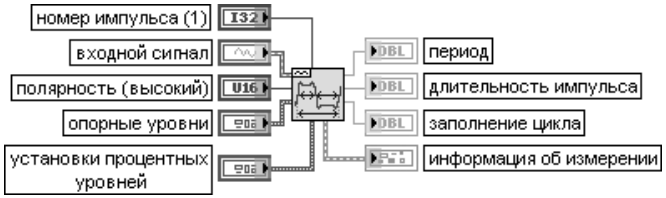
Выход **выброс после фронта** (**%**) (overshoot) содержит оценку локального максимума (минимума), следующего за возрастающим (спадающим) перепадом, выраженную в процентах от амплитуды сигнала, определенной гистограммным методом. Если вход **полярность** определяет спадающий перепад, то **выброс после фронта** рассчитывается по следующему выражению:

$$\textit{выброс после фронта} = 100\frac{(\textit{нижний уровень состояния} - \textit{локальный минимум})}{\textit{амплитуда}}$$

Pulse Measurements



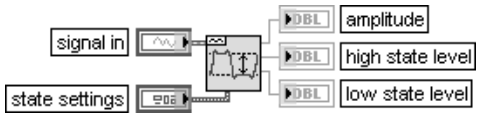
Измерения импульсов



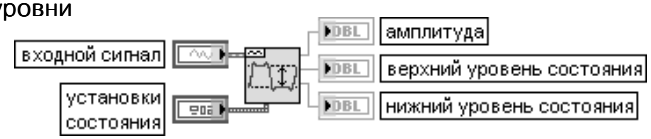
ВП принимает входной сигнал отдельной осциллограммы или массива осциллограмм и возвращает **период** (period), **длительность импульса** (pulse duration) (pulse width), **заполнение цикла** (duty cycle) (duty factor) и **центр импульса** (pulse center) выбранного импульса.
Вход **полярность** (polarity) определяет импульс как **высокий** (high) (по умолчанию) или **низкий** (low). Высокий импульс находится на интервале между соседними точками пересечения среднего опорного уровня возрастающим и спадающим сигналом.
Выход **период** (period) возвращает время между соседними пересечениями **среднего опорного уровня** (mid ref level) в одном направлении в секундах. Интервал измерения включает импульс, определенный с помощью **номера импульса** (pulse number).
Выход **длительность импульса** (pulse duration) возвращает интервал времени в секундах между первыми двумя пересечения **среднего опорного уровня** импульсом, заданным на входе **номер импульса**.
Выход **заполнение цикла** (duty cycle) рассчитывается с помощью следующего выражения:

$$\textit{заполнение цикла} = 100\frac{(\textit{длительность импульса})}{\textit{период}}$$

Amplitude and Levels

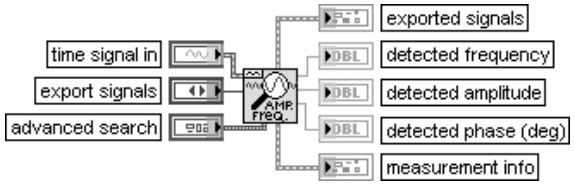


Амплитуда и уровни

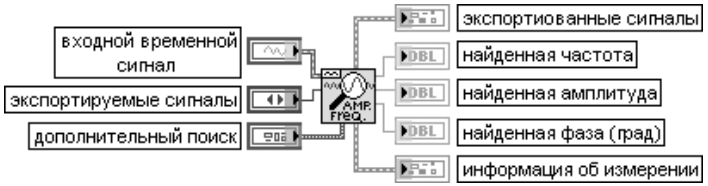


ВП возвращает **амплитуду** (amplitude), **верхний уровень состояния** (high state level) и **нижний уровень состояния** (low state level) осциллограммы или массива осциллограмм.
Выход **амплитуда** (amplitude) определяется как разность между **верхним и нижним уровнями состояния**.
Выходы **верхний уровень состояния** (high state level) и **нижний уровень состояния** (low state level) определяют уровни, на которых импульс или осциллограмма находятся соответственно в высоком или низком состоянии

Extract Single Tone Information



Извлечь информацию об единственном гармоническом колебании



ВП принимает входной сигнал, находит единственное гармоническое колебание с наибольшей амплитудой или ищет определенный диапазон частот и возвращает частоту, амплитуду и фазу этого колебания.
Вход **экспорт сигналов** (export signals) выбирает сигналы, экспортируемые на выход **экспортируемые сигналы**. Предусмотрены следующие варианты экспорта:

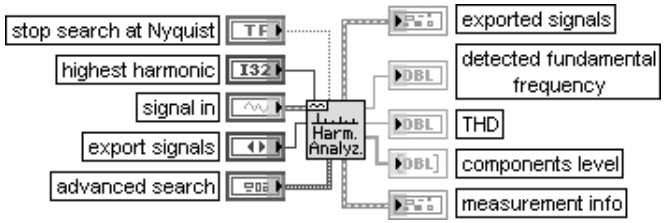
0	**Никакой** (None) (скорейшее вычисление)
1	**Входной сигнал** (Input signal)
2	**Обнаруженный сигнал** (Detected signal) (единственное синусоидальное колебание)
3	**Остаточный сигнал** (Residual signal) (сигнал минус колебание)

Вход **дополнительный поиск** (advanced search) управляет частотной областью поиска, центральной частотой и шириной. Дополнительный поиск целесообразно использовать для сужения диапазона нахождения единственного гармонического колебания. Кластер **дополнительный поиск** содержит следующие элементы:
- **приблизительная частота** (approx freq.) – задает центральную частоту, используемую при поиске синусоидального колебания в частотной области;
- **поиск** (search) – задает диапазон частот в процентах от частоты дискретизации

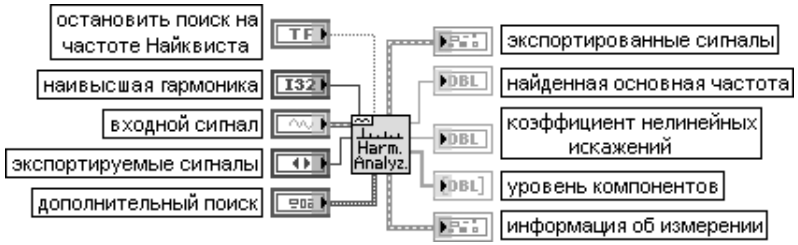
Выход **экспортируемые сигналы** (exported signals) содержит сигналы, заданные на входе **экспорт сигналов** (export signals):
- **экспортируемый временной сигнал** (exported time signal) – представляет осциллограмму, содержащую экспортируемый временной сигнал в соответствии с состоянием входа **экспорт сигналов**;
- **экспортируемый спектр** (exported spectrum) – представляет спектр экспортируемого временного сигнала в соответствии с состоянием входа **экспорт сигналов**. В состав кластера **экспортируемый спектр** входят следующие элементы:
  - **f0** – представляет начальную частоту спектра, выраженную в герцах;
  - **df** – представляет разрешение по частоте, выраженное в герцах;
  - **спектр (Ханн) дБ** (dB Spectrum (Hann)) – представляет спектр входного сигнала, выраженный в децибелах, после обработки сигнала окном Хана.

Выход **найденная частота** (detected frequency) отображает частоту найденного гармонического колебания в герцах.
Выход **найденная амплитуда** (detected amplitude) отображает амплитуду найденного гармонического колебания в вольтах.
Выход **найденная фаза** (detected phase) отображает фазу найденного гармонического колебания в градусах

Harmonic Distortion Analyzer

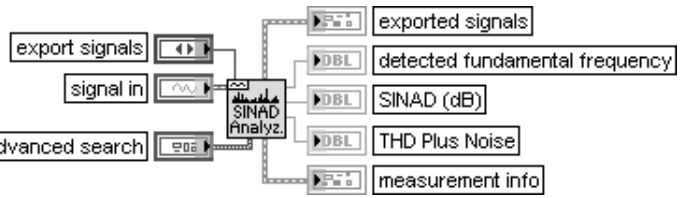


Анализатор гармонических искажений

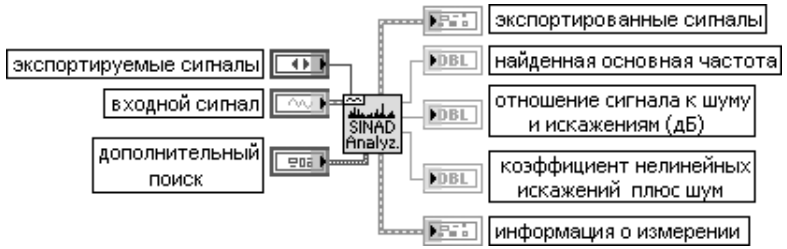


ВП принимает входной сигнал и выполняет полный гармонический анализ, включая измерение тона основной частоты и ее гармоник. В результате анализа возвращаются основная частота, все значения амплитуд гармоник и коэффициент нелинейных искажений.
Вход **остановить поиска на частоте Найквиста** (stop search at Nyquist) должен быть установлен в состояние ИСТИНА (по умолчанию) для включения в поиск гармоник частот меньших частоты Найквиста, или половины частоты дискретизации. При установке входа в состояние ЛОЖЬ этот ВП продолжает поиск в частотной области за частотой Найквиста, предполагая, что эти высокочастотные компоненты **были отражены** (have aliased) в соответствии со следующим выражением: **aliased f = Fs – (f по модулю Fs)**, где **Fs = 1/dt** – частота дискретизации.
Вход **наивысшая гармоника** (highest harmonic) устанавливает наивысшую гармонику, включая основной тон, используемую для гармонического анализа. Например, для анализа по трем гармоникам на этом входе должно быть установлено значение 3.
Выход **найденная основная частота** (detected fundamental frequency) содержит найденную основную частоту, являющуюся результатом поиска в частотной области. Для установки частотного диапазона поиска необходимо использовать вход **дополнительный поиск** (advanced search). Все гармоники измеряются на частотах, кратных этой основной частоте.
Выход **коэффициент нелинейных искажений** (**КНИ**) (THD) содержит измеренное значение коэффициента нелинейных искажений в диапазоне до **наивысшей гармоники**. **КНИ** определяется как отношение среднеквадратичного значения суммы гармоник к амплитуде основного тона. Для вычисления **КНИ** в процентах необходимо умножить значение этого выходного параметра на 100.
Выход **уровень компонентов** (components level) содержит массив амплитуд измеренных гармоник в вольтах, если **входной сигнал** задан в вольтах. Индекс массива является номером гармоники, включающим 0 (постоянную составляющую), 1 (основную частоту), 2 (вторую гармонику), ...**n** (**n**-ую гармонику) и **наивысшую гармонику** (highest harmonic)

SINAD Analyzer



Анализатор шума и искажений

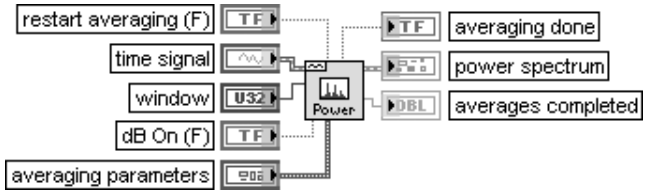


ВП принимает входной сигнал и выполняет полный **анализ отношения сигнала к шуму и искажениям** (Signal in Noise and Distortion (SINAD) analysis), включая измерение тона основной частоты. ВП возвращает основную частоту и отношение сигнала к шуму и искажениям в децибелах.

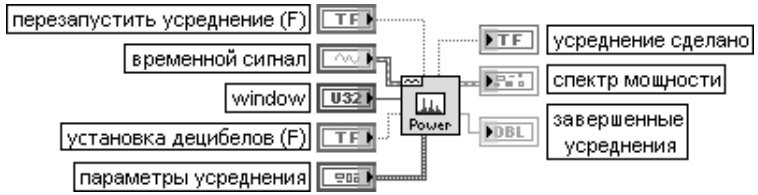
Выход **отношение сигнала к шуму и искажениям (дБ)** (SINAD (dB)) содержит измеренное значение отношения среднеквадратичного значения **входного сигнала** к разности среднеквадратичных значений **входного сигнала** и основного тона.

Выход **коэффициент нелинейных искажений плюс шум** (THD Plus Noise) содержит измеренное значение указанного параметра. Значение этого параметра определяется как отношение разности среднеквадратичных значений **входного сигнала** и основного тона к среднеквадратичному значению **входного сигнала**

FFT Power Spectrum



Спектр мощности БПФ



ВП рассчитывает усредненный спектр мощности **временного сигнала** (time signal) с помощью быстрого преобразования Фурье (БПФ).

Вход **перезапустить усреднение** (restart averaging) определяет возможность перезапуска процесса усреднения. По умолчанию на входе установлено значение ЛОЖЬ. При вызове данного ВП в первый раз процесс усреднения перезапускается автоматически. Типичный случай, когда надо перезапустить усреднение, связан с большим изменением входного сигнала в середине процесса усреднения.

Вход **установка децибелов** (dB On) определяет возможность выражения результатов в децибелах. По умолчанию на входе установлено состояние ЛОЖЬ.

Вход **параметры усреднения** (averaging parameters) представляет кластер, определяющий особенности вычисления усреднения и содержащий следующие элементы:

- **режим усреднения** (averaging mode);

0	**Без усреднения** (No averaging) (по умолчанию)
1	**Векторное усреднение** (Vector averaging)
2	**Среднеквадратичное усреднение** (RMS averaging)
3	**Пиковых значений** (Peak hold)

- **режим взвешивания** (weighting mode) – определяет режим взвешивания для **среднеквадратичного** и **векторного усреднений**. Предусмотрены **линейный** (Linear) **и экспоненциальный** (Exponential) (по умолчанию) режимы взвешивания;
- **число усреднений** (number of averages) – определяет число усреднений, которое используется при **среднеквадратичном** и **векторном усреднениях**. Если выбран режим **экспоненциального взвешивания**, то процесс усреднения выполняется непрерывно. При выборе **линейного взвешивания** процесс усреднения останавливается после вычисления установленного **числа усреднений**.

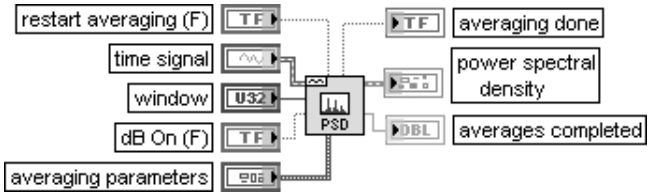
Выход **усреднение сделано** (averaging done) возвращает значение ИСТИНА, если число **выполненных усреднений** равно или превышает число усреднений, заданных на входе **параметры усреднения**. В противном случае на этом выходе возвращается значение ЛОЖЬ. На выходе **усреднение сделано** всегда будет находиться значение ИСТИНА, если выбран режим **без усреднения**.

Выход **спектр мощности** (power spectrum) возвращает усредненный спектр мощности и масштаб по частоте. В состав кластера **спектр мощности** входят следующие элементы:

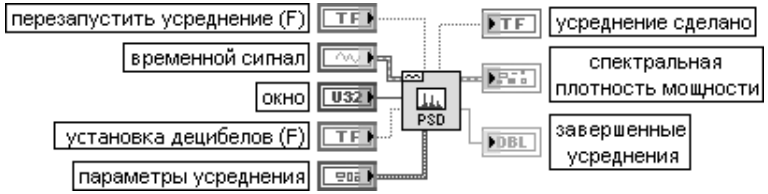
- **f0** – является начальной частотой спектра, выраженной в герцах;
- **df** – является частотным разрешением спектра, выраженным в герцах;
- **величина** (magnitude) – является величиной усредненного спектра мощности.

Выход **завершенные усреднения** (averages completed) возвращает число выполненных к текущему времени усреднений

FFT Power Spectral Density

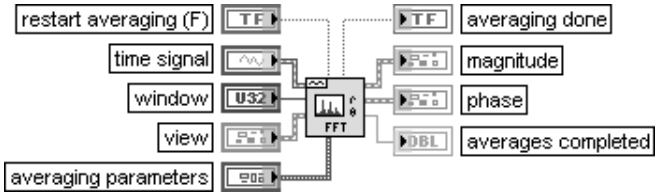


Спектральная плотность мощности БПФ

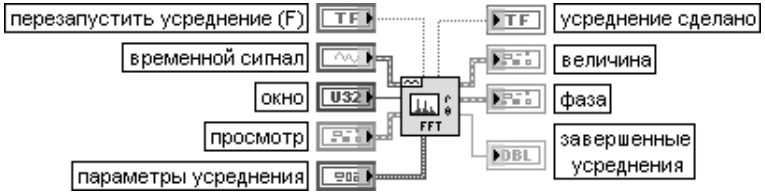


ВП рассчитывает среднюю спектральную плотность мощности **временного сигнала** (time signal) с помощью быстрого преобразования Фурье (БПФ)

FFT Spectrum (Mag-Phase)



Спектр БПФ (величина-фаза)

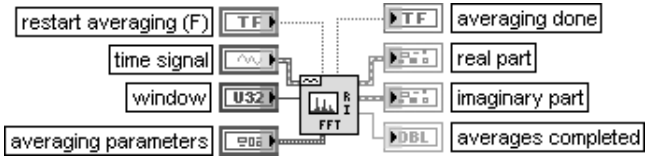


ВП рассчитывает усредненный БПФ спектр **временного сигнала** (time signal). Результаты БПФ возвращаются в виде амплитуды и фазы.

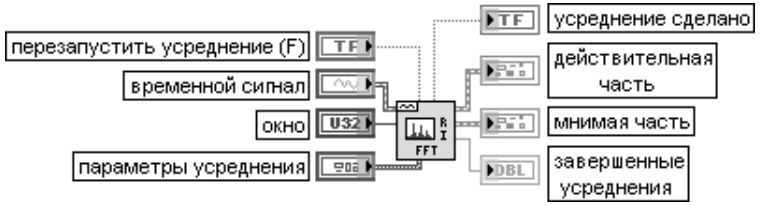
Вход **просмотр** (view) определяет параметры представления результатов, возвращаемых данным ВП:

- **установка децибелов** (dB On) – определяет отображение результатов в децибелах. По умолчанию установлено состояние ЛОЖЬ;
- **развертка фазы** (unwrap phase) – определяет развертку фазы. Развертка устраняет разрывы фазы при превышении значения **π**. По умолчанию установлено состояние ЛОЖЬ, означающее, что развертка фазы не производится;
- **преобразовать в градусы** (convert to degree) – определяет преобразование результатов из радиан в градусы. По умолчанию установлено состояние ЛОЖЬ, означающее, что результаты выражаются в радианах

FFT Spectrum (Real-Im)

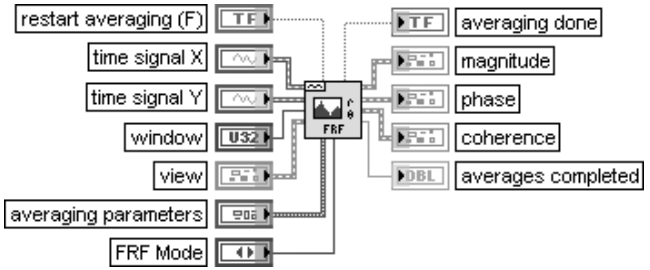


Спектр БПФ (действительная-мнимая часть)

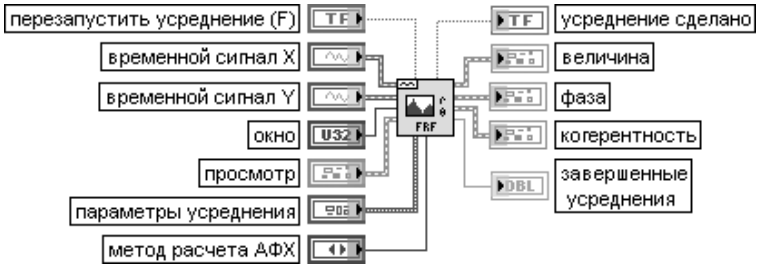


ВП рассчитывает усредненный БПФ спектр **временного сигнала** (time signal). Результаты БПФ возвращаются в виде действительной и мнимой частей

Frequency Response Function (Mag-Phase)



Функция частотного коэффициента передачи (величина-фаза)



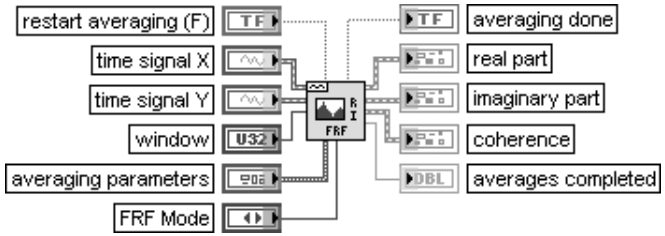
ВП рассчитывает частотный коэффициент передачи и когерентность на основе анализа входных сигналов. Результаты возвращаются в виде **величины** (magnitude), **фазы** (phase) и **когерентности** (coherence).

Как правило, **временной сигнал X** (time signal X) является стимулом, а **временной сигнал Y** – реакцией системы. Каждая временная осциллограмма соответствует единичному блоку БПФ.

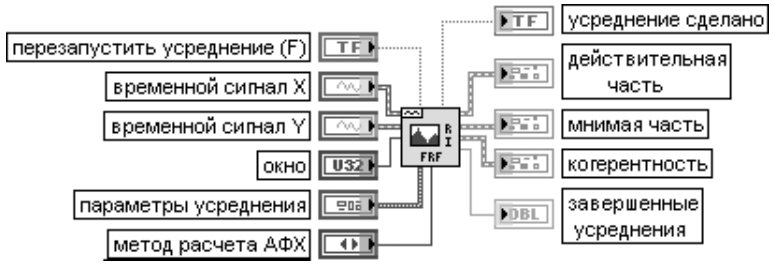
Вход **метод расчета амплитудно-фазовой частотной характеристики (АФХ)** (FRF mode) определяет способ расчета АФХ:

0	1	2
H1 (по умолчанию)	**H2**	**H3**
$A\Phi X_1(f) = \frac{\lvert X^*(f)Y(f)\rvert}{X^2(f)}$	$A\Phi X_2(f) = \frac{Y^2(f)}{\lvert X^*(f)Y(f)\rvert}$	$A\Phi X(f) = (A\Phi X_1(f) + A\Phi X_2(f))/2$

Frequency Response Function (Real-Im)

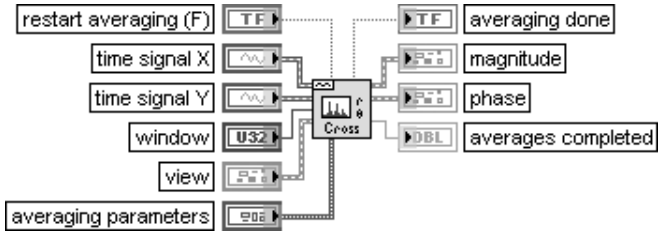


Функция частотного коэффициента передачи (действительная – мнимая часть)

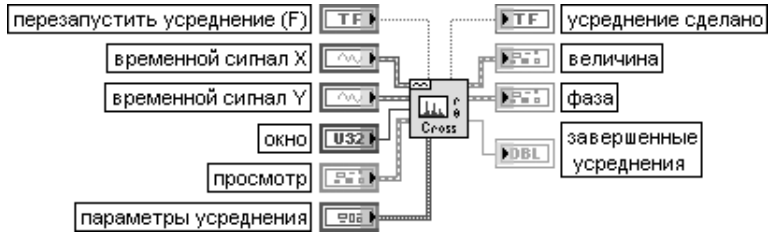


ВП рассчитывает частотный коэффициент передачи и когерентность на основе анализа входных сигналов. Результаты возвращаются в виде **действительной части** (real part), **мнимой части** (imaginary part) и **когерентности** (coherence)

Cross Spectrum (Mag-Phase)

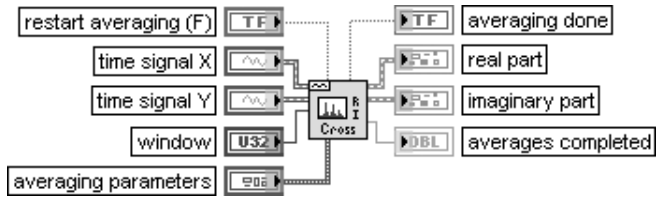


Кросс-спектр (величина-фаза)

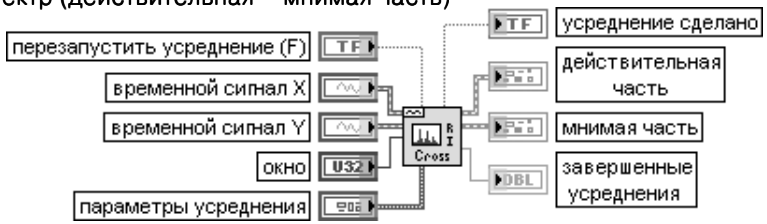


ВП рассчитывает усредненный кросс спектр мощности входных сигналов. Результаты возвращаются в виде **величины** и **фазы**

Cross Spectrum (Real-Im)



Кросс-спектр (действительная – мнимая часть)



ВП рассчитывает усредненный кросс спектр мощности входных сигналов. Результаты возвращаются в виде **действительной** и **мнимой частей**

В состав палитры функций измерения параметров осциллограмм входят Экспресс-ВП **Измерения спектра** (Spectral Measurements), **Измерения искажений** (Distortion Measurements), **Измерения гармонического колебания** (Tone Measurements), **Измерения временных и переходных параметров** (Timing and Transition Measurements) и **Измерения амплитуды и уровня** (Amplitude and Level Measurements).

Измерения спектра (Spectral Measurements)

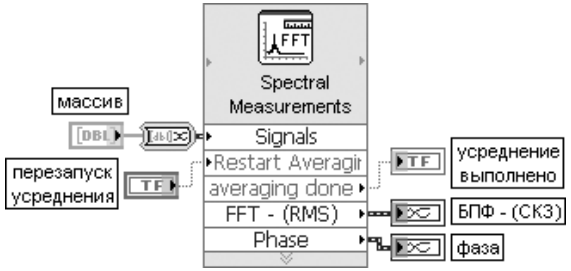


Рис. 5.8. Блок-диаграмма возможного подключения Экспресс-ВП

Экспресс-ВП выполняет измерения спектра сигнала, такие как измерения амплитудного спектра и спектра мощности.
Этот Экспресс-ВП использует функциональность следующих ВП: **Спектр мощности БПФ** (FFT Power Spectrum), **Спектр БПФ (действительная-мнимая часть)**, (FFT Spectrum (Real-Im)), **Спектральная плотность мощности БПФ** (FFT Power Spectral Density)

Измерения искажений (Distortion Measurements)

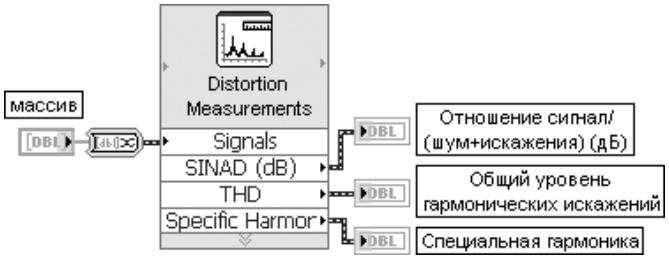


Рис. 5.10. Блок-диаграмма возможного подключения Экспресс-ВП

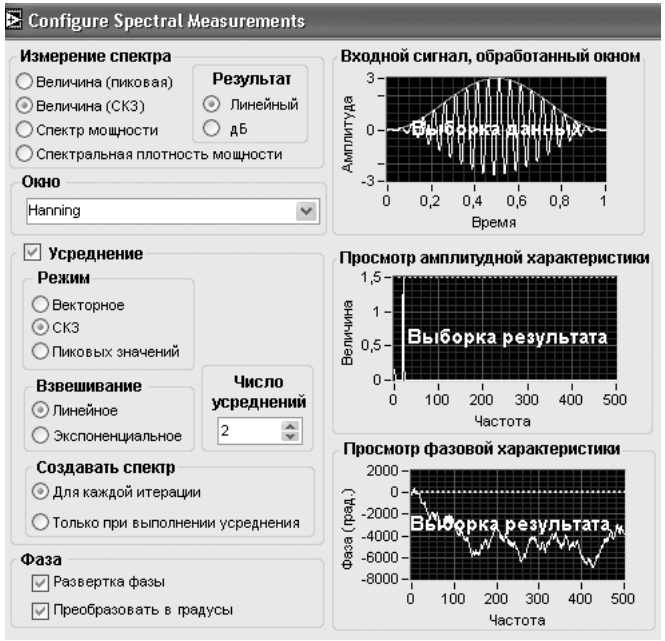


Рис. 5.9. Вид диалогового окна конфигурирования Экспресс-ВП ***Измерения спектра*** *(Spectral Measurements)*

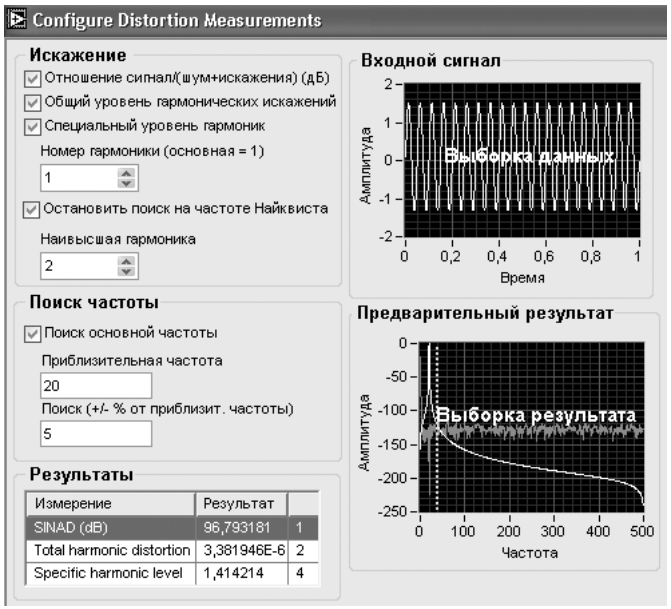


Рис. 5.11. Вид диалогового окна конфигурирования Экспресс-ВП ***Измерения искажений*** *(Distortion Measurements)*

Экспресс-ВП выполняет измерения искажений сигнала, такие как анализ гармонического колебания, коэффициента нелинейных искажений и отношения сигнала к шуму и искажениям.
Этот ВП использует функциональность следующих ВП: **Анализатор гармонических искажений** (Harmonic Distortion Analyzer), **Анализатор шума и искажений** (SINAD Analyzer)

Измерения гармонического колебания (Tone Measurements)

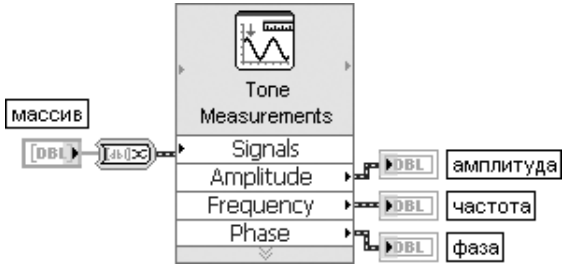


Рис. 5.12. Блок-диаграмма возможного подключения Экспресс-ВП

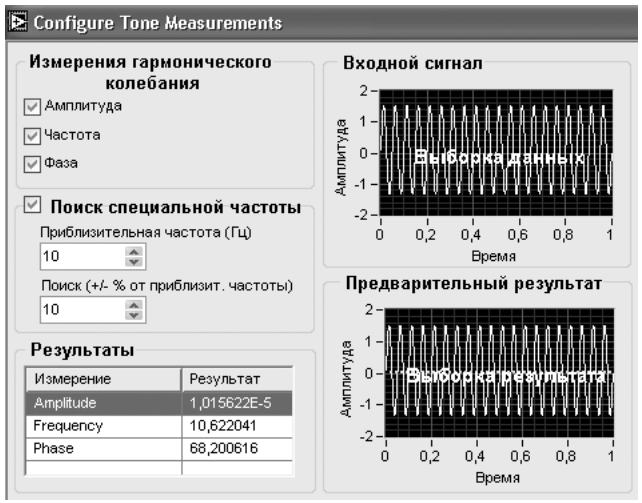


Рис. 5.13. Вид диалогового окна конфигурирования Экспресс-ВП ***Измерения гармонического колебания*** *(Tone Measurements)*

Экспресс-ВП находит единственное гармоническое колебание с наибольшей амплитудой или производит поиск такого колебания в заданном диапазоне частот. Для найденного колебания могут быть определены частота и фаза

Измерения временных и переходных параметров (Timing and Transition Measurements)

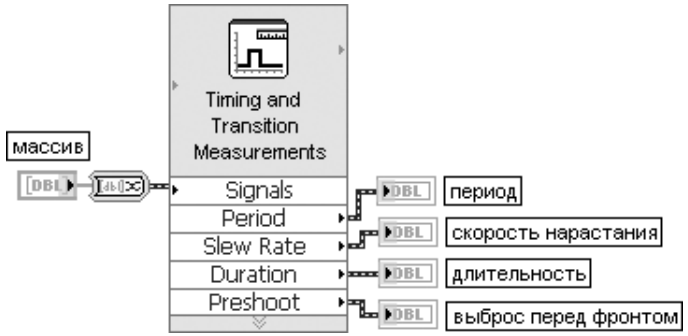


Рис. 5.14. Блок-диаграмма возможного подключения Экспресс-ВП

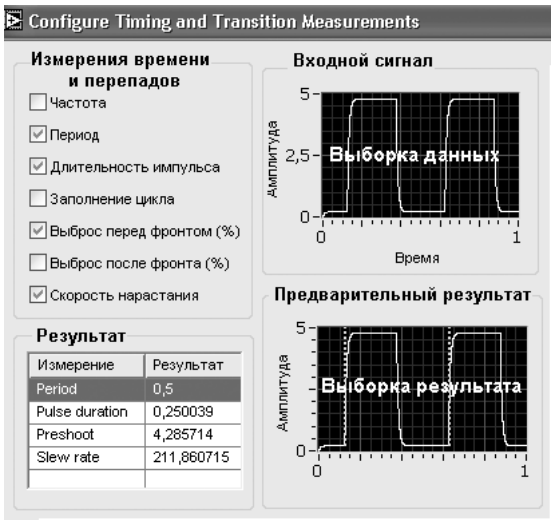


Рис. 5.15. Вид диалогового окна конфигурирования Экспресс-ВП ***Измерения временных и переходных параметров*** *(Timing and Transition Measurements)*

Экспресс-ВП выполняет измерения временных и переходных параметров импульсных сигналов.
Этот Экспресс-ВП использует функциональность следующих ВП: **Измерения импульсов** (Pulse Measurements), **Измерения перепадов** (Transition Measurements)

Измерения амплитуды и уровня (Amplitude and Level Measurements)

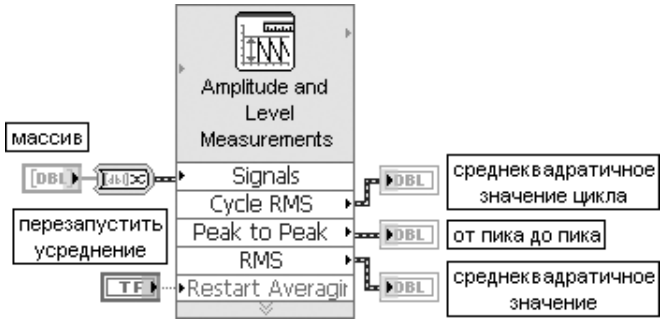


Рис. 5.16. Блок-диаграмма возможного подключения Экспресс-ВП

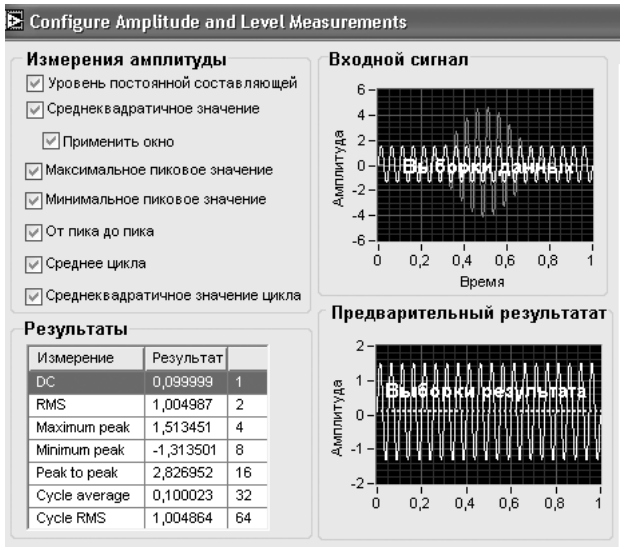


Рис. 5.17. Вид диалогового окна конфигурирования Экспресс-ВП ***Измерения амплитуды и уровня*** *(Amplitude and Level Measurements)*

Экспресс-ВП выполняет измерения напряжений сигнала.
Этот Экспресс-ВП использует функциональность следующих ВП: **Основное измерение постоянной составляющей и среднеквадратичного значения с усреднением** (Basic Averaged DC-RMS), **Измерение постоянной составляющей и среднеквадратичного значения с усреднением** (Averaged DC-RMS), **Амплитуда и уровни** (Amplitude and Levels), **Среднее и среднеквадратичное значение цикла** (Cycle Average and RMS)

Рассмотренные выше Экспресс-ВП измерения параметров осциллограмм позволяют создавать эффективные ВП анализа сигнала с компактным кодом. В качестве примера такого ВП на рис. 5.18 приведена блок-диаграмма ВП **Спектральные измерения** (Spectrum Measurements) из набора примеров NI Example Finder. В данном ВП производятся расчет и сравнение трех амплитудных спектров синусоидального колебания. Первый спектр рассчитывается для отрезка исходного сигнала, а два других – для отрезка сигнала, обработанного весовым окном. Три амплитудных спектра, представленных данными динамического типа, объединяются с помощью функции **Объединить сигналы** (Merge Signals) в один сигнал и выводятся на графические индикаторы с линейным и логарифмическим масштабом по амплитуде. Экспресс-ВП **Статистика** (Statistics) определяет частоту максимума амплитудного спектра, а Экспресс-ВП **Формула** (Formula) – начальный индекс участка амплитудного спектра, выводимого в увеличенном масштабе. Выделение участка спектра производится Экспресс-ВП **Извлечение части сигнала** (Extract Portion of Signal).

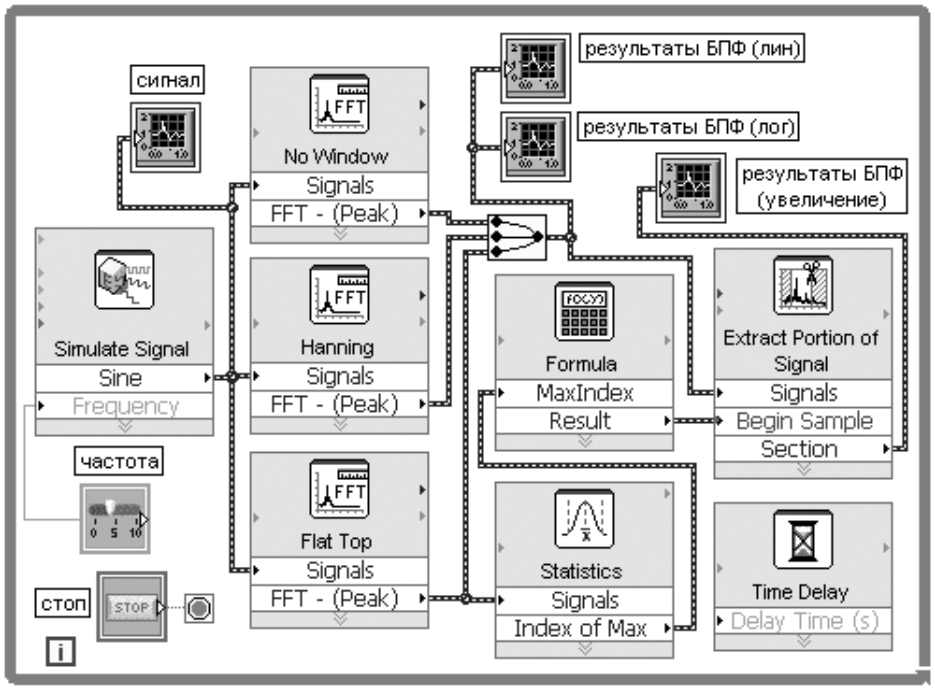


Рис. 5.18. Блок-диаграмма ВП ***Спектральные измерения*** *(Spectrum Measurements)*

Функции обмена данными

6

LabVIEW поддерживает широкий набор высокоуровневых и низкоуровневых протоколов сетевых соединений (рис. 6.1), что позволяет удовлетворить специфические потребности приложений пользователя (таблица).

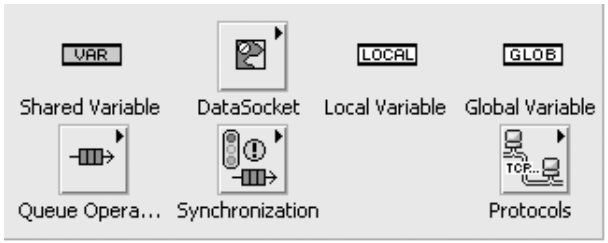


Рис. 6.1. Вид подпалитры функций обмена данными

6.1. Разделяемые переменные

Новые **Разделяемые переменные** (Shared Variables) LabVIEW 8.20 призваны решить проблемы связи между разнородными платформами за счет унификации процедур обмена данными через единый, гибкий и открытый коммуникационный протокол, поддерживающий различные процессоры, устройства реального времени, а также изделия сторонних разработчиков. Эти переменные используют функциональность таких технологий передачи данных, как DataSocket, и приспособлены для передачи сложных типов данных, характерных для расширенных приложений в распределенных системах.

В отличие от множества существующих в LabVIEW методов распределения данных, таких как UDP/TCP, очереди LabVIEW и Real-Time FIFO, привязка пользовательских элементов управления и индикации к источникам данных в уз-

Способ сетевого соединения	Характеристика применения	Необходимость программирования	Потери при передаче данных	Несколько устройств записи/считывания	Задержка передачи	Скорость передачи данных
Разделяемая переменная (Shared Variable)	Обеспечивает обмен данными с другими ВП на локальном или удаленном компьютере	Нет	Да	Многие со многими	Низкая	Высокая
Функции DataSocket с протоколом psp	–//–	Да	Да	Многие со многими	Низкая	Высокая
Функции DataSocket с протоколом dstp	–//–	Да	Нет	Многие со многими	Средняя	Высокая
LabVIEW Web Server	Публикует в сети изображения лицевой панели	Нет	Нет	Один со многими	Средняя	Низкая
ВП SMTP Email	Отправляет электронную почту с прикрепленными данными	Да	Да	Один со многими	Высокая	Высокая
ВП и функции TCP	Поддерживает связь с прибором, использующим протокол на основе TCP	Да	Да	Один с одним	Средняя	Высокая
ВП и функции UDP	Поддерживает связь с пакетом программ, использующим протокол на основе UDP	Да	Нет	Один со многими	Низкая	Высокая
Функции IrDA	Устанавливает беспроводное соединение с удаленным компьютером	Да	Да	Один с одним	Средняя	Низкая
ВП и функции Bluetooth	Устанавливает беспроводное соединение с устройством Bluetooth	Да	Да	Один с одним	Средняя	Низкая

лах распределенной системы производится с помощью конфигурирования разделяемой переменной в диалоговом окне свойств и не требует включения в приложение какого-либо кода конфигурирования. (Вместе с тем существует возможность программного конфигурирования разделяемой переменной с помощью Узла свойств.)

Разделяемая переменная позволяет распределять данные между циклами одной диаграммы или по сети между несколькими ВП. В LabVIEW 8.20 предусмотрена возможность создания разделяемой переменной трех типов: **для отдельного процесса** (Single-Process), **публикуемой в сети** (Network-Published) и запускаемой по времени. Разделяемая переменная последнего типа используется с модулем реального времени LabVIEW 8 Real-Time Module.

Разделяемая переменная создается в **открытом** проекте с помощью вызова контекстного меню таких строк в окне **Проводника проекта** (Project Explorer), как **Мой компьютер** (My Computer), **платформа реального времени** (real-time target), **библиотека проекта** (project library) или папка в библиотеке проекта, и выбора строки **Новая ⇒ Переменная** (New ⇒ Variable). При этом открывается диалоговое окно **Свойства разделяемой переменной** (Shared Variable Properties) (рис. 6.2), которое позволяет выбрать **Тип данных** (Data Type), **Тип переменной** (Variable Type), возможность использования и параметры буфера (флажок Use Buffering), а также параметры подключения к источнику (флажок Bind to Source).

Рис. 6.2. Диалоговое окно конфигурирования разделяемой переменной

На этом рисунке показан вид диалогового окна при выборе в выпадающем списке **тип переменной** (Variable Type) опции **публикуемая в сети**. При выборе варианта разделяемой переменной для **отдельного процесса** разделы меню **использовать буферизацию** (Use Buffering), **привязать к источнику** (Bind to Source) и **единственный источник записи** (Single Writer) не отображаются.

Дополнительные модули и инструменты LabVIEW, устанавливаемые пользователем, могут добавить типы, опции конфигурирования и ограничения для разделяемой переменной.

После закрытия диалогового окна конфигурирования разделяемой переменной она появляется в окне Проводника проекта под соответствующей библиотекой или папкой (рис. 6.3).

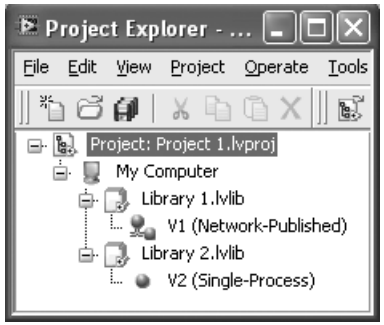


Рис. 6.3. Вид разделяемых переменных в окне проекта при различных вариантах конфигурирования

После добавления разделяемой переменной к проекту LabVIEW она может быть перенесена на **блок-диаграмму** ВП, находящегося в том же проекте, с помощью механизма «drag and drop» в виде соответствующего узла для осуществления чтения или записи данных (рис. 6.4). Также предусмотрен вариант установки узла переменной из подпалитры Структуры. В этом случае установление связи узла с самой переменой из активного проекта производится с помощью диалогового окна **Выбрать переменную** (Select Variable), вызываемого с помощью одноименной строки контекстного меню узла или двойным щелчком мыши. Контекстное меню узла также позволяет установить функцию чтения или записи и с помощью строки **Показать отметку времени** (Show timestamp) вывести отметку времени.

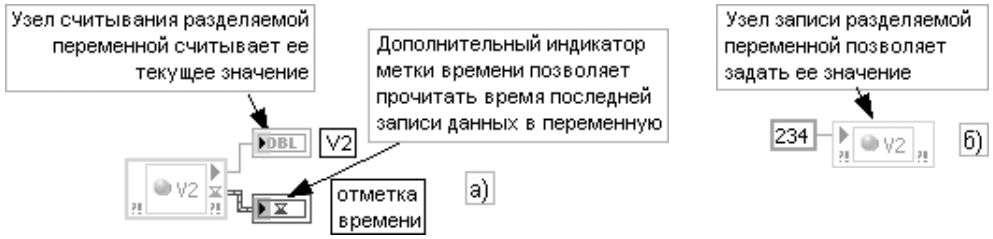


Рис. 6.4. Чтение (а) и запись (б) в разделяемую переменную через ее узлы

Связь элемента **лицевой панели** с разделяемой переменной также может устанавливаться с помощью переноса иконки переменой из окна проекта на лицевую панель или путем конфигурирования свойств этого элемента (раздел Data Binding диалогового окна свойств).

Свойства разделяемой переменной могут быть изменены в любое время, и проект LabVIEW распространит новые установки по всем ссылкам переменных, находящихся в памяти. При сохранении библиотеки с разделяемой переменной на диске эти изменения также вносятся во все экземпляры переменной, сохраненные на диске.

Переменные, сконфигурированные для отдельного процесса, рекомендуется использовать для передачи данных между двумя фрагментами одного ВП, которые нельзя соединить проводниками, например между параллельными циклами одного и того же ВП или между двумя разными ВП, находящимися в одном приложении. По своей реализации разделяемая переменная в такой конфигурации аналогична глобальной переменной. Основным ее преимуществом по сравнению с глобальной переменной является способность трансформироваться в разделяемую переменную, публикуемую в сети.

Такая переменная по сравнению с переменной для отдельного процесса обеспечивает больше возможностей и, вместе с тем, имеет более сложную внутреннюю реализацию.

Разделяемая переменная, публикуемая в сети, производит отправку и прием данных по сети с помощью **Механизма разделяемой переменной** (Shared Variable Engine – SVE), который, в свою очередь, использует фирменный протокол **публикации-подписки** NI (NI Publish-Subscribe Protocol – NI-PSP). Протокол NI-PSP создается поверх протокола UDP и поэтому получает его достоинства в отношении работы без установления соединения и фиксации состояния. LabVIEW идентифицирует разделяемую переменную, публикуемую в сети, по сетевому пути, включающему имя компьютера, имя (имена) библиотеки проекта и имя самой переменной.

Протокол NI-PSP требует меньшей пропускной способности сети и обладает большей эффективностью по сравнению с TCP/IP. Однако протокол NI-PSP, в отличие от протокола UDP, гарантирует доставку за счет осуществления дополнительного уровня функциональности поверх несовершенного протокола UDP.

Для размещения переменной в сети пользователь должен **развернуть** (deploy) разделяемую переменную, публикуемую в сети, в SVE. Такое действие выполняется либо автоматически при запуске ВП, который содержит ссылку на переменную, либо вручную с помощью контекстного меню библиотеки проекта. Последний вариант развертывания должен использоваться, например, после изменения параметров переменной в окне Свойства.

При записи данных в узел разделяемой переменной LabVIEW посылает новое значение в развернутый SVE, который и размещает переменную. После этого SVE в цикле обработки данных **публикует** значение таким образом, что **подписчики** получают обновленное значение (рис. 6.5). Если использовать клиент-серверную терминологию, то для разделяемой переменной SVE является **сервером**, а все ссылки – **клиентами**, независимо от того, выполняют они запись или считывание из переменной. Клиент SVE является частью реализации каждого узла разделяемой переменной.

Все разделяемые переменные являются частью библиотеки проекта. SVE регистрирует библиотеки проекта и содержащиеся в них разделяемые переменные, не учитывая, что LabVIEW требует только одну из этих переменных. По умолча-

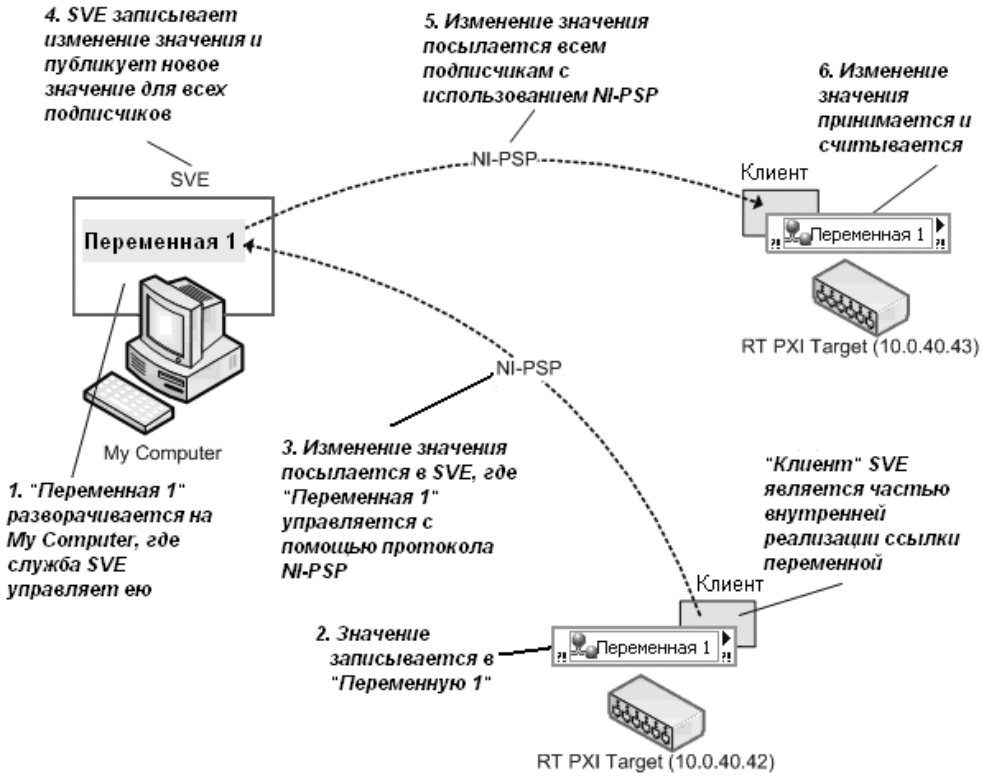


Рис. 6.5. Работа Механизма разделяемой переменной и значения переменных, публикуемых в сети

нию SVE развертывает и публикует библиотеку разделяемых переменных сразу после того, как запускается на выполнение ВП, который ссылается на любую из содержащихся в нем переменных. Так как SVE развертывает всю библиотеку, которой принадлежит разделяемая переменная, то он публикует все переменные библиотеки независимо от того – ссылается на них выполняющийся ВП или нет. В любое время можно развернуть вручную любую библиотеку пользователя в окне Проводника проекта с помощью ее контекстного меню.

Остановка ВП или перезагрузка машины, которая поддерживает переменную, не разрывает ее связь с сетью. Если необходимо удалить разделяемую переменную из сети, то надо явно **свернуть** (undeploy) библиотеку с переменной в окне Проводника проекта с помощью ее контекстного меню. Еще один вариант выполнения этой функции – выбор меню **Инструменты** ⇒ **Разделяемая переменная** ⇒ **Менеджер переменной** (Tools ⇒ Shared Variable ⇒ Variable Manager).

Разделяемая переменная может быть сконфигурирована на прием данных от разнообразных источников, в качестве которых могут выступать каналы DAQ,

элементы данных, другие переменные и элементы данных из серверов ввода/вывода, которые определены вне активного проекта. Например, это могут быть данные из модуля FieldPoint или из сервера OPC. Конфигурирование источника разделяемой переменной производится в разделе **привязать к источнику** (Bind to Source) диалогового окна **Свойства разделяемой переменной**. После конфигурирования переменной к существующему источнику LabVIEW не отслеживает изменения в конфигурации источника данных. Поэтому такие изменения могут разорвать связь с разделяемой переменной.

С переменной, публикуемой в сети, может использоваться **буферизация**. Эта опция включается в диалоговом окне конфигурирования переменной (флажок **использовать буферизацию**), после чего может быть указан размер буфера в единицах выбранного **типа данных** (Data Type).

Буферизация позволяет устранять кратковременные флуктуации скоростей чтения и записи переменной. LabVIEW использует заданный размер буфера для создания двух внутренних буферов, одного – на стороне SVE, а другого – у подписчика (рис. 6.6). Каждый подписчик разделяемой переменной, публикуемой в сети, получает свой собственный буфер, таким образом, подписчики не оказывают влияния друг на друга.

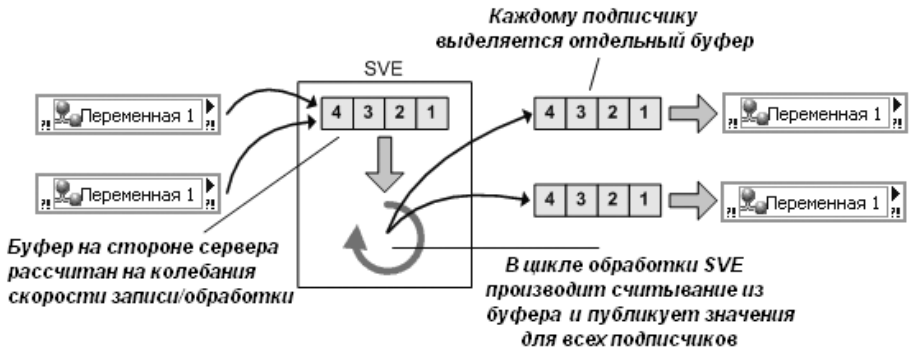


Рис. 6.6. Буферизация разделяемой переменной

Дополнительной функцией, присущей только разделяемой переменной, публикуемой в сети, является **связывание данных лицевой панели** (front panel data binding). Для создания элемента управления, связанного с разделяемой переменной, необходимо перенести ее из окна Проводника проекта на лицевую панель ВП. После установки связывания с элементом управления изменение его значения приводит к изменению значения переменной, с которой элемент управления связан. Если связывание элемента с SVE установлено успешно, во время выполнения ВП при передаче данных рядом с элементом зажигается небольшой зеленый индикатор.

Для доступа и изменения связывания любого элемента управления или индикатора служит страница **связывание данных** (Data Binding) в диалоговом окне

свойства (Properties). При использовании модулей LabVIEW Real-Time или LabVIEW DSC можно выбрать **Инструменты** ⇒ **Разделяемая переменная** ⇒ **Общая конфигурация связывания лицевой панели** (Tools ⇒ Shared Variable ⇒ Front Panel Binding Mass Configuration) для отображения диалогового окна **Общая конфигурация связывания лицевой панели** и создания интерфейса оператора, который связывает множество элементов управления и индикаторов с разделяемыми переменными.

Наряду с интерактивным созданием, конфигурированием и развертыванием разделяемой переменной с помощью проекта LabVIEW, а также чтением и записью данных в переменную через соответствующий узел на блок-диаграмме или через связывание данных лицевой панели LabVIEW также обеспечивает **программный доступ** ко всем перечисленным функциям.

Программный способ создания библиотек проекта и разделяемых переменных с помощью Сервера ВП используется в приложениях с большим числом таких переменных. Кроме того, модуль LabVIEW DSC обеспечивает всеобъемлющий набор ВП для программного создания и редактирования разделяемых переменных и библиотек проекта, так же как и для управления SVE.

При распространении автономного приложения, которое использует разделяемую переменную, не следует включать файл .lvlib, содержащий параметры ее конфигурирования, в библиотеку или исполняемый файл. Необходимо использовать страницу **Установка файлов источника** (Source Files Setting) диалогового окна **Свойства приложения** (Application Properties) для изменения **Назначения** (Destination) файла .lvlib с целью его направления за пределы библиотеки или исполняемого файла. Можно развернуть разделяемую переменную двумя способами:

- записать приложение так, что оно программно разворачивает переменную в начальный момент. Вызвать метод Deploy Library в ВП, находящемся на верхнем уровне приложения. На входе **Путь библиотеки** (Library Path) метода использовать относительный путь к файлу .lvlib, который содержит разделяемую переменную;
- вручную развернуть разделяемую переменную на компьютере или платформе с SVE перед запуском созданного приложения.

Такое же замечание относится и к варианту распространения приложения в виде библиотеки динамической компоновки (DLL), которая использует разделяемую переменную.

6.2. Технология передачи данных и функции DataSocket

Технология передачи данных DataSocket базируется на использовании функций и **сервера** DataSocket. Сервер DataSocket представляет самостоятельную программу, которая управляет подключением клиентов. Клиентские приложения могут записывать данные на сервер или считывать через сервер данные любого

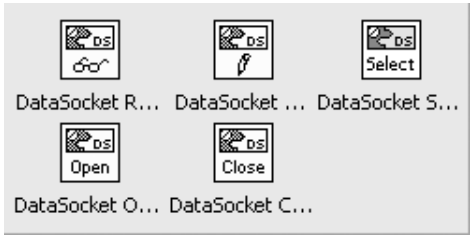


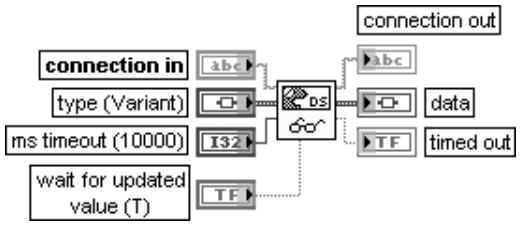
Рис. 6.7. Вид подпалитры функций и ВП DataSocket

источника. В первом случае они являются **издателями** (publishers) DataSocket, во втором – **подписчиками** (subscribers) DataSocket.

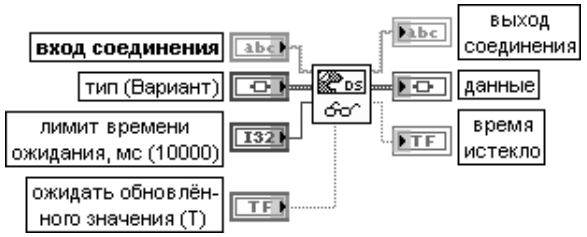
Передача данных с помощью технологии DataSocket может осуществляться как с помощью функций DataSocket (рис. 6.7), так и непосредственно между элементами лицевых панелей локальных или удаленных ВП. В последнем случае необходимо создать соединение DataSocket с помощью диалогового окна **Соединение DataSocket** (DataSocket Connection), вызываемого с помощью строки **Операции с данными** ⇒ **Соединение DataSocket** (Data Operations ⇒ DataSocket Connection) контекстного меню элемента. В окне указывается тип соединения – **Опубликовать** (Publish) или **Подписаться** (Subscribe), а также путь к данным или к файлу. Фиксация параметров производится кнопкой **Подключить** (Attach). Элементы лицевой панели, связанные протоколом Data Socket, отличаются небольшим прямоугольным индикатором, принимающим зеленый цвет при успешной передаче данных и красный цвет в случае ошибки. В процессе передачи данных передатчик и приемник данных являются клиентами сервера Data Socket.

Функции, обеспечивающие программную передачу данных с помощью соединения Data Socket, рассмотрены в таблице.

DataSocket Read



Читать из соединения DataSocket



Функция берет по очереди следующее доступное значение данных из буфера, расположенного на стороне клиента и связанного с соединением, которое определено на **входе соединения** (connection in), и возвращает это значение. Вход соединения может быть строкой DataSocket URL (по умолчанию) или ссылкой на соединение DataSocket. Для минимизации потерь данных необходимо использовать протокол DSTP, который поддерживает буферизацию, и сконфигурировать сервер и клиент.

Вход соединения (connection in) определяет источник данных для чтения.

Вход **тип** (**Вариант**) (type (Variant)) определяет тип читаемых данных и тип выходного терминала данных. Тип по умолчанию – Вариант, который может быть любого типа.

Вход **лимит времени ожидания, мс** (ms timeout) определяет необходимое время ожидания нового значения, становящегося доступным в буфере соединения. Функция игнорирует этот вход и не ожидает если на входе **ожидать обновленного значения** (wait for updated value) установлено состояние ЛОЖЬ и начальное значение поступило. По умолчанию значение входа равно 10,000 мс (10 с).

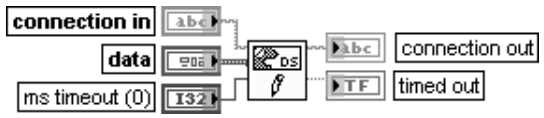
Вход **ожидать обновленного значения** (wait for updated value) заставляет функцию ожидать обновления значения при установке в состояние ИСТИНА. Если буфер соединения содержит необработанные данные, то функция возвращает следующее доступное значение немедленно. В противном случае функция ожидает обновления данных в течение интервала времени, заданного на входе **лимит времени ожидания, мс**. Если обновления не произошло за интервал таймаута, функция возвращает текущее значение и устанавливает выход **время истекло** в состояние ИСТИНА. Если на входе **ожидать обновленного значения** установлено состояние ЛОЖЬ, то функция возвращает следующее доступное значение из буфера соединения или последнее считанное значение при отсутствии доступных данных.

Выход соединения (connection out) представляет источник данных, который определяет соединение DataSocket.

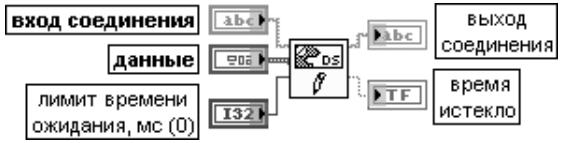
Выход **данные** (data) отображает результат чтения. Если функция закончила работу по превышению времени ожидания, на выход выводится значение, полученное при последнем считывании. Если произошел выход по превышению времени ожидания и ничего не было считано или тип данных оказался несовместимым, то в этом случае на выход выводится значение 0, пустое значение или их эквивалент.

На выходе **время истекло** (timed out) возвращается значение ИСТИНА, если функция закончила работу по истечению времени ожидания при ожидании обновления данных или при недостоверном значении

DataSocket Write



Записать в соединение DataSocket



Функция записывает данные в соединение, определяемое на **входе соединения** (connection in).

Вход соединения (connection in) определяет адрес записи данных. **Вход соединения** может быть строкой, которая описывает DataSocket URL (по умолчанию), или ссылкой соединения DataSocket.

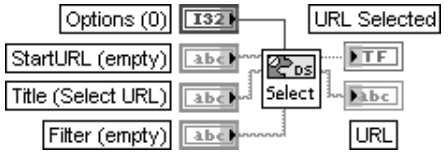
Вход **данные** (data) служит для передачи данных, записываемых в соединение DataSocket. Вход **данные** может быть любого типа.

Вход **лимит времени ожидания, мс** (ms timeout (0)) определяет интервал времени в миллисекундах, в течение которого функция ожидает выполнения продолжающейся операции. По умолчанию этот интервал равен 0. При необходимости ожидания функцией выполнения операции на этом входе необходимо установить значение –1. В настоящее время только протоколы DSTP, OPC и файловый протокол поддерживают ненулевое значение лимита времени ожидания для функции DataSocket Write.

Выход соединения (connection out) представляет источник данных, который определяет соединение DataSocket.

Выход **время истекло** (timed out) возвращает значение ЛОЖЬ, если операция завершается без ошибки в пределах лимита времени ожидания. Если **лимит времени ожидания, мс** равен 0, то выход **время истекло** принимает значение ЛОЖЬ

DataSocket Select URL



Выбрать URL DataSocket

ВП отображает диалоговое окно для выбора пользователем источника/потребителя данных и возвращает **Универсальный указатель информационного ресурса** (URL (Universal Resource Locator)) к этим данным. Данный ВП необходимо использовать, только когда URL к объектам с данными неизвестен и его надо найти с помощью диалогового окна.

Вход **опции** (Options) по умолчанию имеет значение 0.

Вход **начальный URL** (StartURL) определяет URL для открытия диалогового окна. Поле может быть пустым, отображая простое текстовое окно, или содержать протокол, например file:, отображая файловое диалоговое окно. Поле также может содержать полный URL.

Вход **заголовок** (Title) определяет заголовок диалогового окна.

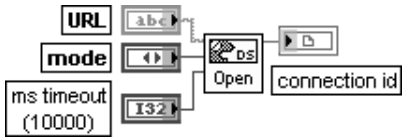
Вход **фильтр** (Filter) определяет значения фильтра, используемого в диалоге. Такая возможность используется только для файлов.

Выход **URL выбран** (URL Selected) устанавливается в состояние ИСТИНА при выборе достоверного источника/потребителя данных.

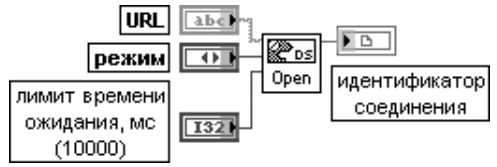
Выход URL отображает URL выбранного источника/потребителя данных. Это значение действительно, если только выход **URL выбран** имеет состояние ИСТИНА.

Блок-диаграмма ВП **Выбрать URL DataSocket** приведена на рис. 6.8. В ее основе лежит вызов функции ActiveX CWDataSocket. При этом после открытия ссылки к функции последовательно вызываются метод SelectURL и свойство URL.

DataSocket Open



Открыть соединение DataSocket



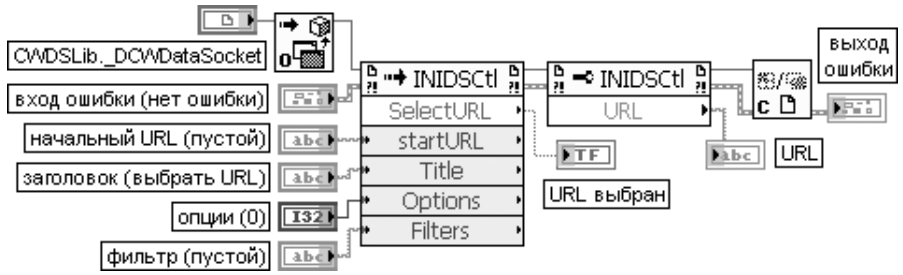


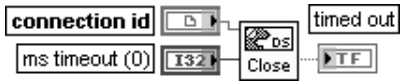
Рис. 6.8. Блок-диаграмма ВП ***Выбрать URL DataSocket***

Функция открывает соединение DataSocket, заданное с помощью URL.
Вход **URL** идентифицирует источник данных для чтения или потребителя данных для записи. URL начинается с имени протокола, используемого для чтения или записи данных, такого как dstp, opc, logos, ftp и file. Вместо хранения URL в ВП, читающем данные с сервера, можно использовать файловый протокол для чтения URL из файла DataSocket Link (DSL).
Вход **способ** (mode) определяет тип соединения DataSocket. Варианты способов приведены в таблице.

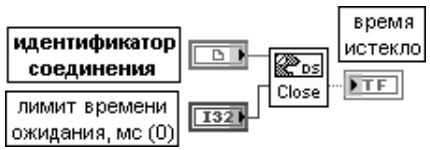
0	**Чтение** (Read) (по умолчанию)
1	**Запись** (Write)
2	**Чтение/Запись** (Read/Write)
3	**Буферизированное Чтение** (BufferedRead)
4	**Буферизированные Чтение/Запись** (BufferedRead/Write)

Буферизация применяется только при использовании функции **Читать из соединения DataSocket** для приема данных, передаваемых сервером. Буферизация не применяется при использовании соединения DataSocket для приема данных от элементов лицевой панели. Для минимизации потерь данных необходимо буферизировать данные DataSocket и на сервере.
Вход **лимит времени ожидания, мс** (ms timeout (0)) определяет время ожидания в миллисекундах, в течение которого LabVIEW должен установить соединение DataSocket. По умолчанию это время равно 10,000 мс (10 с). При установке –1 функция ожидает все время. При установке нулевого значения LabVIEW не пытается осуществить соединение DataSocket и возвращает ошибку 56.
Выход **идентификатор соединения** (connection id) однозначно определяет идентификатор соединения DataSocket

DataSocket Close



Закрыть соединение DataSocket



Функция закрывает соединение DataSocket, определенное с помощью ссылки **идентификатор соединения** (connection id).
Назначение входа **лимит время ожидания, мс** (ms timeout (0)) идентично назначению одноименного входа описанной выше функции **Запись в соединение DataSocket**

На рис. 6.9 и 6.10 приведены блок-диаграммы ВП **Устройство записи данных трехмерного графика в соединение Data Socket** (DS 3D Graph Writer) и **Устройство считывания данных трехмерного графика из соединения Data Socket** (DS 3D Graph Reader) из набора примеров NI Example Finger LabVIEW, в которых функции Data Socket используются для передачи и приема данных трехмерного графика.

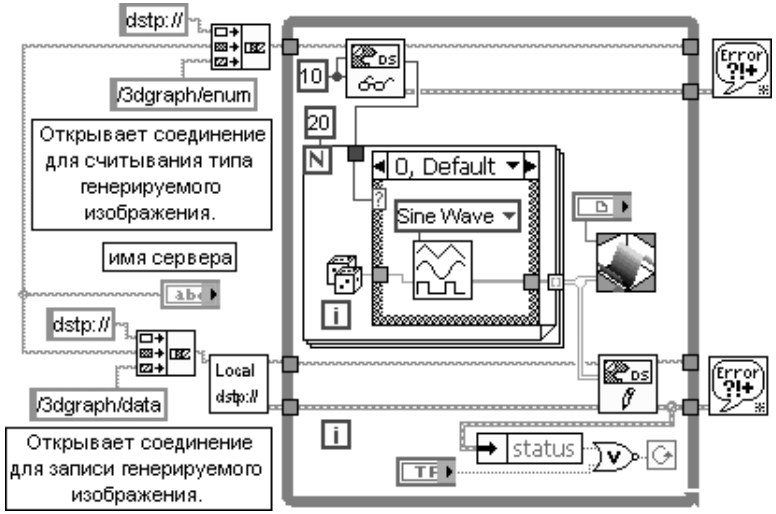


Рис. 6.9. Блок-диаграмма ***ВП Устройство записи данных трехмерного графика в соединение Data Socket*** *(DS 3D Graph Writer)*

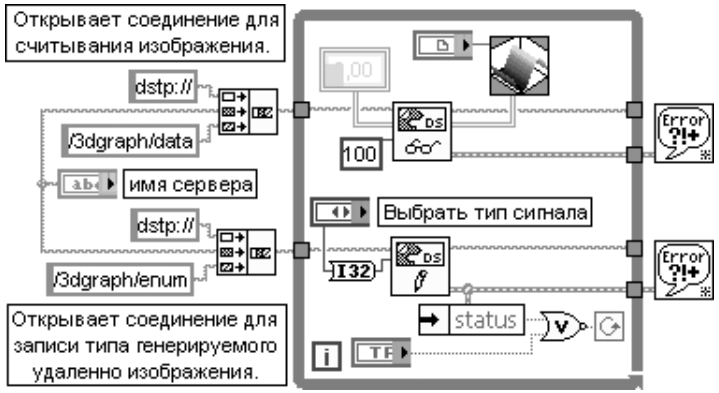


Рис. 6.10. Блок-диаграмма ***ВП Устройство считывания данных трехмерного графика из соединения Data Socket*** *(DS 3D Graph Reader)*

6.3. Функции протоколов передачи данных

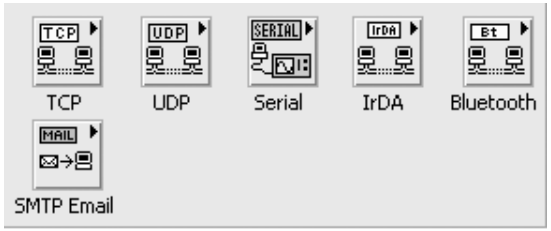


Рис. 6.11. Вид подпалитры функций протоколов передачи данных

6.3.1. Функции протоколов TCP/IP

Протокол межсетевого взаимодействия (Internet Protocol – IP), **доставки пользовательских дейтаграмм** (User Datagram Protocol – UDP) и **контроля передачи** (Transmission Control Protocol – TCP) входят в число основных инструментов для построения сетевых соединений. Название стека протоколов TCP/IP образуется из названий двух хорошо известных протоколов.

Стек TCP/IP может использоваться для связи компьютеров как в отдельных, так и во взаимосвязанных сетях. Основным в TCP/IP является транспортный уровень, который предназначен для надежной доставки данных прикладным программам. Протокол TCP обеспечивает надежную передачу сообщений между удаленными прикладными процессами за счет образования виртуальных соединений. Он на порядок сложнее протокола UDP и позволяет строить более безопасные сервисы. Однако для простых сервисов с низкими требованиями к надежности и безопасности удобнее использовать более простой в реализации протокол UDP. Он также может использоваться, когда высокий уровень надежности обеспечивается высоким качеством каналов связи.

ВП, входящие в подпалитры TCP (рис. 6.12) и UDP, позволяют создавать клиентские и серверные приложения.

В следующей таблице рассмотрены функции протоколов TCP/IP.

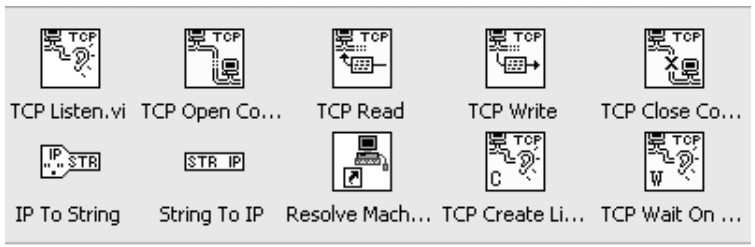
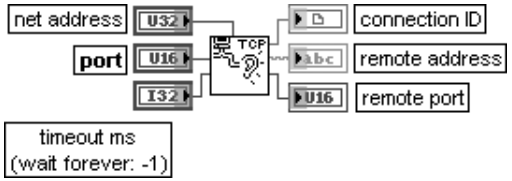
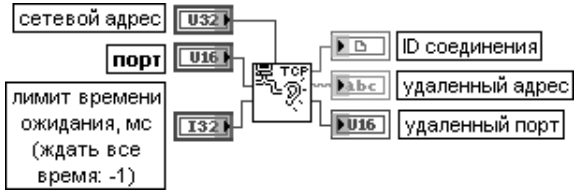


Рис. 6.12. Вид подпалитры функций TCP/IP

TCP Listen



Слушать TCP



ВП создает слушателя и ожидает принимаемое сетевое соединение по протоколу TCP с заданным **портом** (port).

Вход **сетевой адрес** (net address) определяет сетевой адрес. Определение адреса полезно, если имеется более одной сетевой карты и желательно прослушать только одну карту с заданным адресом. Если сетевой адрес не определен, то LabVIEW прослушивает все сетевые адреса. Для получения IP-адреса текущего компьютера необходимо использовать функцию **Строку в IP** (String to IP).

Вход **порт** (port) определяет номер порта, который прослушивается для установления соединения.

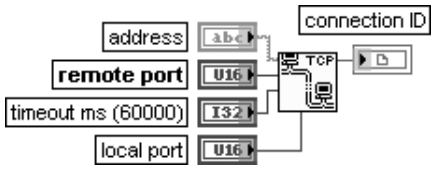
Вход **лимит времени ожидания, мс** (timeout ms) определяет время в миллисекундах, в течение которого должно быть установлено соединение. Если этого не произошло, то ВП завершается и возвращает ошибку. По умолчанию значение входа равно –1, что определяет неограниченное ожидание.

Выход **ID-соединения** (connection ID) представляет идентификатор сетевого соединения, который однозначно определяет TCP-соединение. Это значение используется в качестве ссылки при вызовах последующих ВП.

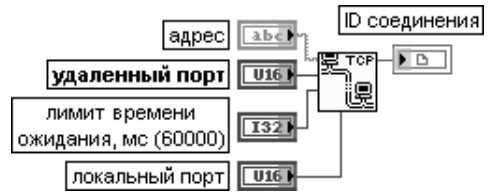
Выход **удаленный адрес** (remote address) отображает адрес удаленной машины, связанной с TCP-соединением. Этот адрес представляется в формате точечной записи IP-адреса.

Выход **удаленный порт** (remote port) отображает порт удаленной системы, используемый для соединения

TCP Open Connection



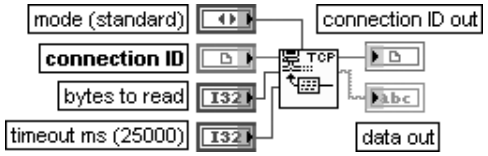
Открыть соединение TCP



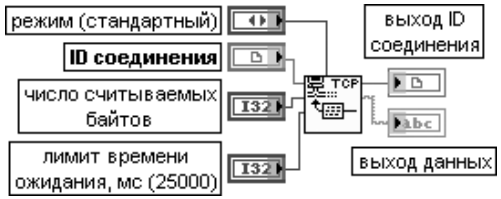
Функция открывает сетевое TCP-соединение с заданным **адресом** (address) и **портом** (port). Закрытие соединения производится с помощью функции **Закрыть соединение TCP** (TCP Close Connection).

Вход **адрес** (address) задает адрес, с которым должно быть установлено соединение. Этот адрес может быть IP-адресом, заданным в точечном формате, или DNS-именем машины. Если адрес не задан, то LabVIEW осуществляет соединение с локальным компьютером.
Вход **удаленный порт** (remote port) определяет порт для заданного **адреса** (address), с которым осуществляется соединение.
Вход **лимит времени ожидания, мс** (timeout ms) задает время в миллисекундах, в течение которого должно осуществиться соединение. Если соединения за указанное время не произошло, то функция завершается и возвращает ошибку. По умолчанию значение времени равно 60,000 мс (1 мин). Установка значения – 1 приводит к неограниченному ожиданию.
Вход **локальный порт** (local port) задает локальный порт соединения. Некоторые серверы разрешают соединения клиентам, которые используют номера портов из определенного диапазона, зависящего от сервера. Если значение на этом входе равно 0, то операционная система выбирает неиспользуемый порт.
Функция выхода **ID-соединения** (connection ID) была рассмотрена в предыдущем разделе

TCP Read



Читать из TCP

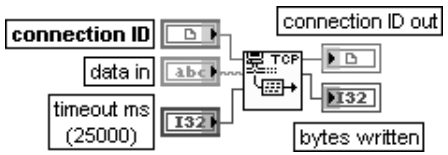


Функция читает заданное число байтов из сетевого соединения по протоколу TCP, возвращая результат на **выходе данных** (data out).
Вход **режим** (mode) устанавливает следующие режимы операции чтения:

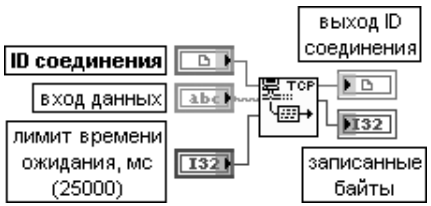
0	**Стандартный** (standard) (по умолчанию) – ожидается поступление всех запрошенных байтов или истечение времени, заданного на входе **лимит времени ожидания, мс**. Возвращаются все полученные байты. Если число поступивших байтов оказалось меньше запрошенного, то возвращается частичное количество байтов и сообщается об ошибке по времени ожидания
1	**Буферизированный** (Buffered) – ожидается поступление всех запрошенных байтов или истечение времени ожидания. Если число поступивших байтов оказалось меньше запрошенного, то байты не возвращаются и сообщается об ошибке по времени ожидания
2	**CRLF** – ожидается прием CR (возврат каретки) с последующим LF (перевод строки) с числом запрошенных байтов или истечение времени ожидания. Возвращаются все принятые байты и символы CR и LF. Если CR и LF не найдены, то байты не возвращаются и сообщается об ошибке по времени ожидания
3	**Немедленный** (Immediate) – Ожидается прием любого числа байтов. Заданное время ожидается если только отсутствуют принятые байты. Возвращается число принятых байтов. Сообщается об ошибке по времени ожидания при отсутствии принятых байтов

Вход **число считываемых байтов** (bytes to read) определяет число считываемых байтов.
Вход **лимит времени ожидания, мс** (timeout ms) определяет время в миллисекундах, которое вход **режим** использует как максимальное время перед сообщением об ошибке по истечению времени ожидания. По умолчанию это время равно 25,000 мс.
Выход данных (data out) содержит данные, считанные из TCP-соединения

TCP Write

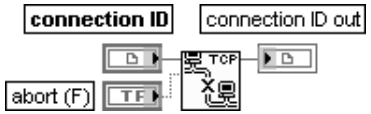


Записать в TCP

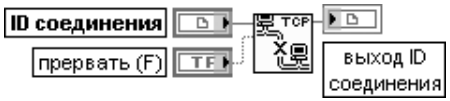


Функция записывает данные в сетевое TCP-соединение.
Вход данных (data in) содержит данные, записываемые в соединение.
Вход **лимит времени ожидания, мс** (timeout ms) определяет время, в течение которого должна быть выполнена функция записи. Если за отведенное время не произошло записи байтов, функция завершается и возвращается сообщение об ошибке. Значение по умолчанию равно 25,000 мс.
Выход **записанные байты** (bytes written) отображает число байтов, записанных функцией в заданное соединение

TCP Close Connection

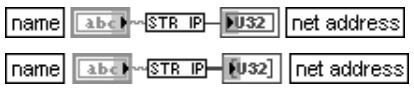


Закрыть соединение TCP



Функция закрывает сетевое TCP соединение

String To IP



Строку в IP-адрес

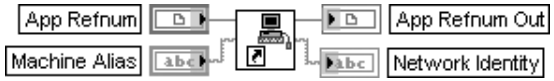
Функция преобразует строку в IP-адрес или массив IP-адресов.
Если функция находится в режиме простого выхода, **сетевой адрес** (net address) содержит первый результат, возвращаемый распознавателем адреса по DNS-имени. Если эта функция находится в режиме **множественного выхода** (Multiple Output), то результат представляет массив всех IP-адресов, возвращаемых распознавателем адреса по DNS-имени. Если узел не может преобразовать строку, результат принимает значение 0 в режиме простого выхода или пустой массив в режиме множественного выхода.
Для переключения между режимами простого и множественного выхода необходимо выбрать из контекстного меню узла строку **множественный выход** (Multiple output).
Вход **имя** (name) представляет преобразуемую строку. Если строка пустая, то **сетевой адрес** содержит IP-адрес текущей машины.
Выход **сетевой адрес** (net address) отображает сетевой IP-адрес, соответствующий **имени** (name); он является числовым представлением записанного в точечной нотации IP-адреса

IP To String

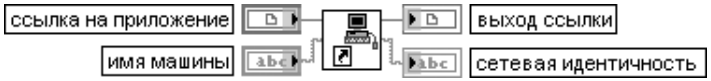
IP в строку

Функция преобразует сетевой адрес в строку.
Вход **сетевой адрес** (net address) представляет сетевой IP-адрес в виде набора целых чисел без знака, имеющих формат точечной записи.
Вход **точечная запись** (dot notation) определяет представление **имени** (name) в формате точечной записи. По умолчанию на входе установлено значение ЛОЖЬ.
Выход **имя** (name) представляет строковый эквивалент **сетевого адреса**.

Resolve Machine Alias

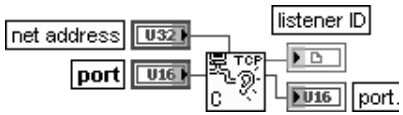


Преобразовать имя машины

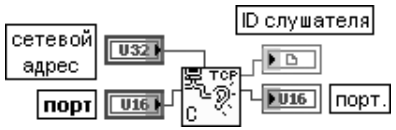


ВП возвращает IP-адрес машины, который может использоваться с сетевыми функциями и функциями Сервера ВП.
На входе **ссылка на приложение** (App Refnum) указывается ссылка на приложение LabVIEW.
На выход **сетевая идентичность** (Network Identity) выводится IP-адрес машины. Если ВП не может преобразовать имя машины, то он возвращает ошибку, а на выходе **сетевая идентичность** – сетевое имя машины

TCP Create Listener

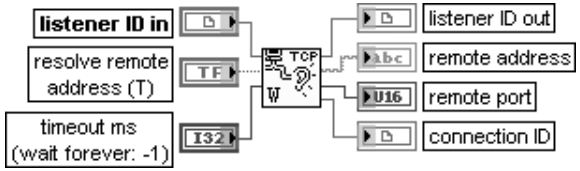


Создать слушателя TCP

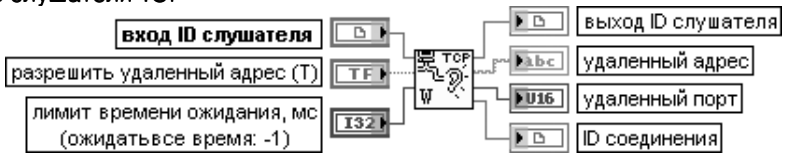


Функция создает слушателя для сетевого TCP-соединения. Подключение 0 ко входу порта позволяет операционной системе динамически выбирать доступный порт для установки TCP-соединения.
Назначение входов и выходов функции идентично одноименным входам и выходам рассмотренной выше функции **TCP Listen**

TCP Wait On Listener



Ожидать слушателя TCP



Функция ожидает приема по сетевому TCP-соединению.
Вход ID слушателя (listener ID in) представляет ссылку сетевого соединения, которая однозначно определяет слушателя.
Вход **разрешить удаленный адрес** (resolve remote address) определяет необходимость вызова функции **IP to String** для удаленного адреса. По умолчанию состояние входа – ИСТИНА.
Выход **удаленный адрес** (remote address) отображает адрес удаленного компьютера, связанного с TCP-соединением. Этот адрес указывается в формате точечной записи.
Выход **удаленный порт** (remote port) представляет порт удаленной системы, используемый для соединения

На рис. 6.13 и 6.14 приведены блок-диаграммы ВП **Простой клиент данных** (Simple Data Client) и **Простой сервер данных** (Simple Data Server) из набора примеров NI Example Finder, использующих функции TCP. Первым должен запускаться ВП Простой сервер данных, который устанавливает порт с заданным номером на прослушивание соединения и ожидает обращения клиента в течение 5 с. При отсутствии такого обращения ВП завершает работу. При обнаружении обращения клиента ВП сервера непрерывно передает данные, а ВП клиента принимает их и отображает на графическом индикаторе.

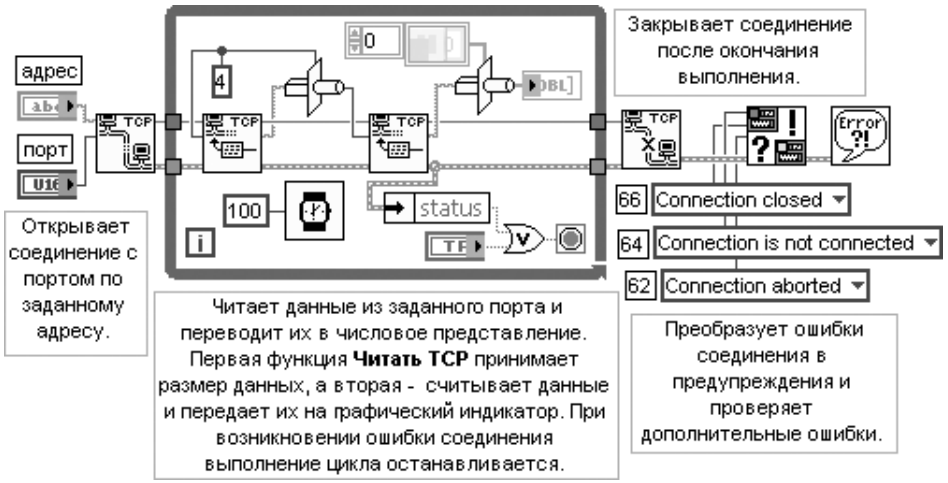


*Рис. 6.13. Блок-диаграмма ВП **Простой клиент данных** (Simple Data Client)*

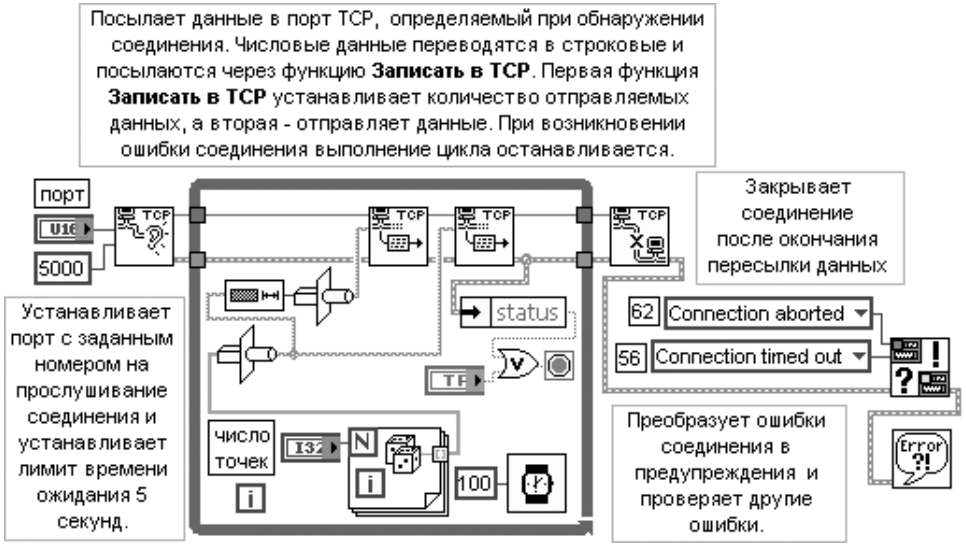


Рис. 6.14. Блок-диаграмма ВП ***Простой сервер данных*** *(Simple Data Server)*

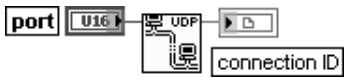
6.3.2. Функции протоколов UDP

Протокол UDP отличается от TCP тем, что не производится проверки ошибок в принятых пакетах информации, вследствие чего отсутствует гарантия получения предполагаемым получателем передаваемого фрагмента данных.

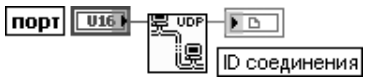
UDP может произвести широковещательную передачу данных по сети с помощью определения IP-адреса 255.255.255.255. Однако при этом необходимо учитывать, что маршрутизаторы и другое сетевое оборудование обычно отказывают в широковещательной передаче за определенные точки, поэтому протяженность широковещания на практике ограничена. Функции протокола UDP из соответствующей подпалитры (рис. 6.15) рассмотрены в таблице.

Рис. 6.15. Вид подпалитры функций UDP

UDP Open

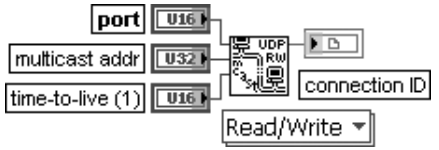


Открыть UDP

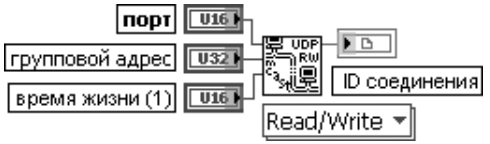


Функция открывает UDP-соединение, используя заданный **порт** (port). Закрытие соединения осуществляется функцией **UDP Close**. Для открытия соединения, позволяющего производить чтение, запись или чтение и запись, необходимо использовать ВП **Открыть групповое UDP-соединение** (UDP Multicast Open) вместо данной функции.
Вход **порт** задает локальный порт, с которым устанавливается UDP-соединение.
Выход **connection ID** представляет ссылку сетевого соединения, которая однозначно определяет UDP-соединение. Данный выход используется для передачи ссылки на соединение в последующие ВП

UDP Multicast Open



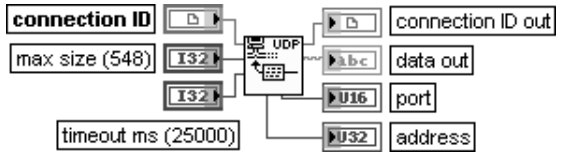
Открыть групповое UDP-соединение



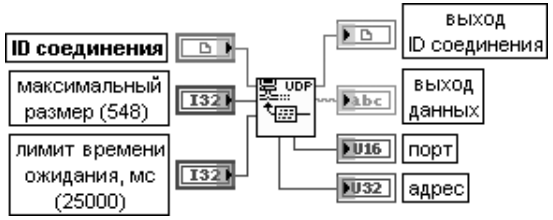
ВП открывает групповое UDP-соединение для заданного **порта** (port). Данный полиморфный ВП позволяет производить чтение, запись или чтение и запись данных по протоколу UDP. Выбор функции производится вручную с помощью меню.
Вход **порт** (port) определяет локальный порт, с которым необходимо создать UDP-соединение.
Вход **групповой адрес** (multicast addr) определяет IP-адрес группы, с которой необходимо установить соединение. Если адрес на данном входе не установлен, то соединение с группой не производится и возвращаемое соединение является соединением только для записи. Групповые адреса находятся в диапазоне с 224.0.0.0 по 239.255.255.255.
Вход **время жизни** (time-to-live (TTL)) определяет число маршрутов минус 1 для пересылки дейтаграммы. Значение TTL применяется для всех дейтаграмм, посылаемых с использованием данного соединения. Таблица показывает действия, происходящие в групповой дейтаграмме при определении значения параметра TTL. При определении значения больше 1 дейтаграмма посылается и маршрутизаторы пересылают ее через TTL-1 слой.

0	Дейтаграмма остается на хост-компьютере
1	Дейтаграмма посылается каждому клиенту в той же локальной подсети, в которой находится отправитель. Хабы/повторители и мосты/переключатели пересылают дейтаграмму. Маршрутизаторы не пересылают дейтаграмму, если **время жизни** равно 1

UDP Read

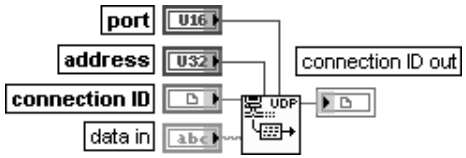


Читать из UDP

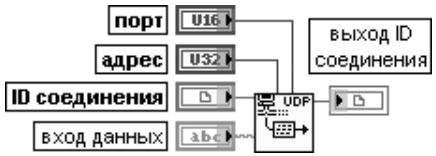


Функция читает дейтаграмму из UDP-соединения, сохраняя результат на **выходе данных** (data out). Функция возвращает данные при приеме какого-либо числа байтов и ожидает полное **время ожидания** (timeout ms), если прием байтов отсутствует.
Вход **максимальный размер** (max size) определяет максимальное число считываемых байтов. По умолчанию оно равно 548. При работе в системе Windows установка на этом входе иного числа может вызвать ошибку.
Вход **лимит времени ожидания, мс** (timeout ms) задает интервал времени в миллисекундах, в течение которого функция ожидает поступления байтов. При их отсутствии по истечении заданного времени функция завершается и возвращает ошибку. По умолчанию значение входа равно 25,000 мс.
Выход ID-соединения (connection ID out) имеет то же значение, что и **ID-соединения**.
Выход данных (data out) содержит данные, считываемые из дейтаграммы UDP.
Выход **порт** (port) отображает порт UDP-соединения, отправивший дейтаграмму.
Выход **адрес** (address) отображает адрес компьютера, в котором была сформирована дейтаграмма

UDP Write

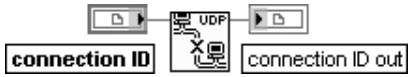


Записать в UDP

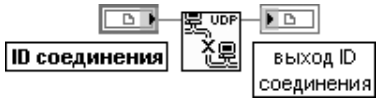


Функция записывает данные в удаленное UDP-соединение.
Вход **порт** (port) определяет порт адреса, в который передается дейтаграмма.
Вход **адрес** (address) определяет адрес компьютера, которому передается дейтаграмма.
Вход **ID-соединения** (connection ID) представляет ссылку на сетевое соединение, однозначно определяющую UDP-соединение.
Вход данных (data in) содержит данные, записываемые в другое UDP-соединение. В сети Ethernet объем данных ограничен 8192 байтами

UDP Close



Закрыть UDP



Функция закрывает UDP-соединение.
Вход **ID-соединения** (connection ID) содержит ссылку на сетевое соединение, однозначно определяющую UDP-соединение, которое необходимо закрыть.
Выход ID-соединения (connection ID out) имеет то же значение, что и **ID-соединения**. Этот выход не должен подключаться к другим функциям UDP

На рис. 6.16 и 6.17 приведены блок-диаграммы ВП **Отправитель UDP** (UDP Sender) и **Получатель UDP** (UDP Receiver) из набора примеров NI Example Finder, использующих функции UDP. Первым должен запускаться ВП **Получатель UDP**.

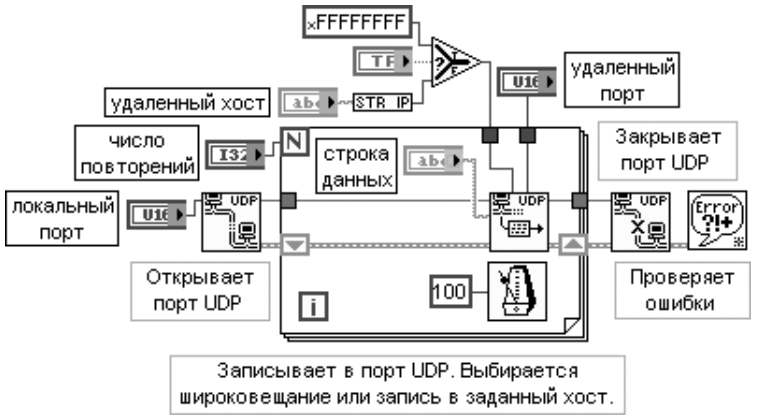


*Рис. 6.16. Блок-диаграмма ВП **Отправитель UDP** (UDP Sender)*

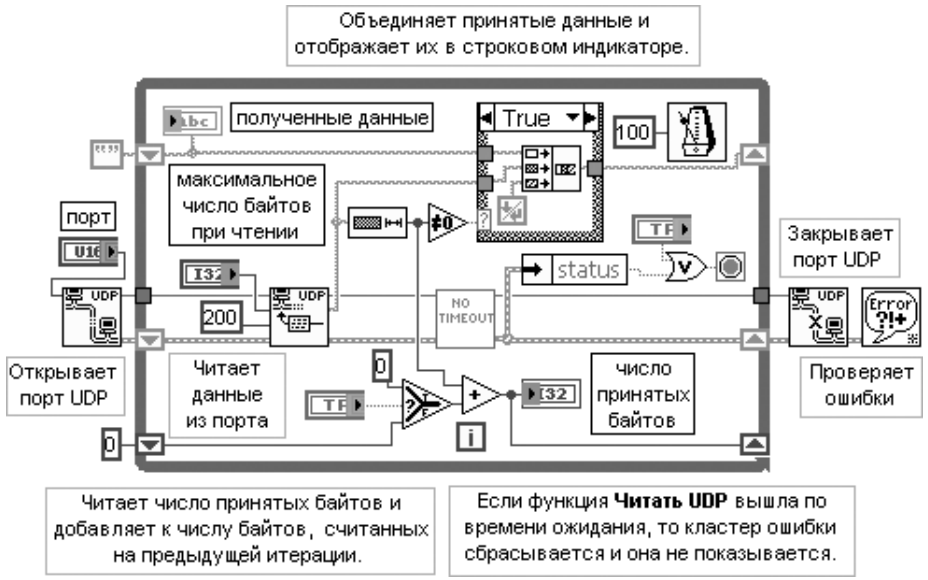


*Рис. 6.17. Блок-диаграмма ВП **Получатель UDP** (UDP Receiver)*

6.3.3. Функции протокола Bluetooth

Технология Bluetooth предназначена для унификации возможностей ближней радиосвязи между несколькими пользователями и опирается на постоянно включенные радиоустройства на базе микрочипа, работающие на частоте 2,4 ГГц. Два устройства Bluetooth, находящиеся на расстоянии 10 м, могут обеспечить передачу данных со скоростью до 720 Кбит/с. В настоящее время максимальная дальность работы устройств Bluetooth на открытой местности составляет 300 м.

Функции и ВП из подпалитры (рис. 6. 18) используют протокол замещения кабеля RFCOMM, который представляет виртуальный последовательный порт, предназначенный для максимально незаметной для пользователей и программ замены кабельных технологий. RFCOMM обеспечивает транспортировку двоичных данных и эмулирует сигналы управления EIA-232 над узкополосным уровнем Bluetooth. EIA-232 (ранее известный как RS-232) представляет широко используемый стандарт интерфейса последовательного порта. С помощью интерфейса RFCOMM создаются серверы и клиенты Bluetooth.

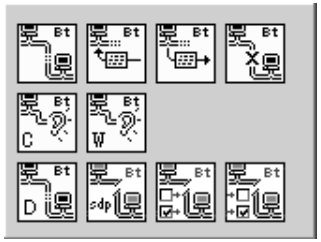


Рис. 6. 18. Вид подпалитры ВП и функций протокола Bluetooth

Создание клиентских и серверных приложений Bluetooth аналогично созданию таких приложений для TCP. Сервер Bluetooth использует протокол SDP (Service Discovery Protocol) для широковещательного оповещения о сервисах, предоставляемых сервером, и прослушивает возвращаемые соединения. Клиент создает исходящее RFCOMM соединение с сервером. После того как клиент и сервер соединились друг с другом, они обмениваются данными до прерывания соединения клиентом или сервером или до потери соединения.

LabVIEW поддерживает устройства Bluetooth, которые используют только драйвер Microsoft Bluetooth из Windows XP Service Pack 1 или более позднего. Информацию об устройствах Bluetooth, которые поддерживают драйвер Microsoft Bluetooth, можно получить на сайте Microsoft. Большинство устройств Bluetooth используют по умолчанию фирменные драйверы. Для использования устройства с LabVIEW необходимо переключиться на драйвер Microsoft Bluetooth. В Windows XP Service Pack 2 или в более поздней версии такой драйвер уже установлен.

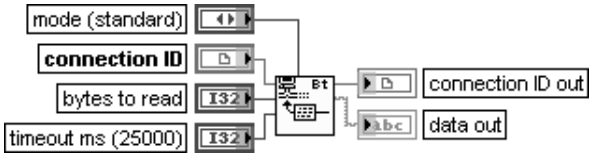
Перечень функций протокола Bluetooth рассмотрен в следующих таблицах.

Bluetooth Open Connection / Открыть соединение Bluetooth

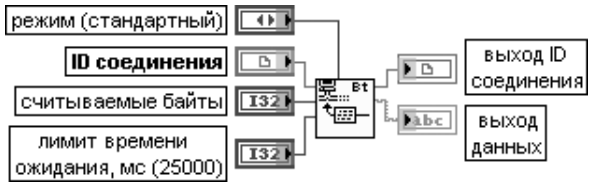
Функция запрашивает соединение с сервером Bluetooth.
Вход **адрес** (address) задает адрес сервера Bluetooth. Пример записи адреса выглядит следующим образом: 00:07:E0:07:D7:50.
Вход **канал** (channel) определяет номер канала сервера Bluetooth. Если значение на данном входе равно нулю, то эта функция использует UUID для определения устройства.
Вход **лимит времени ожидания, мс** (timeout ms) определяет время ожидания до выхода по ошибке. По умолчанию это время равно 60 000 мс.

Вход **uuid** представляет уникальный идентификатор для службы. Если значение на входе **канал** равно нулю, то для определения службы функция использует UUID. UUID должен быть представлен в формате GUID. Пример записи UUID в формате GUID выглядит следующим образом: B62C4E8D-62CC-404b-BBBF-BF3E3BBB1374.
Выход **ID-соединения** (connection ID) представляет ссылку на сетевое соединение, которая однозначно устанавливает соединение Bluetooth

Bluetooth Read

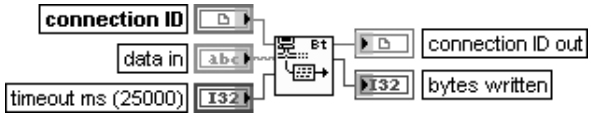


Читать из Bluetooth

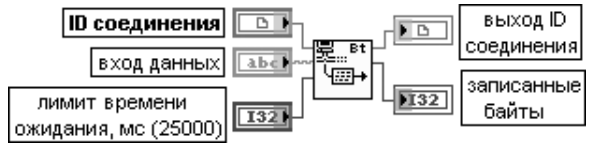


Функция считывает заданное число байтов из сетевого соединения Bluetooth и возвращает результат на выходе данных.
Вход **режим** (mode) идентичен одноименному входу рассмотренного выше ВП **Читать из TCP**.
Выход данных (data out) содержит данные, считанные функцией из соединения Bluetooth

Bluetooth Write

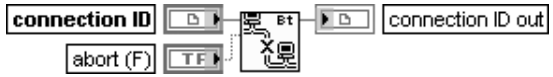


Записать в Bluetooth

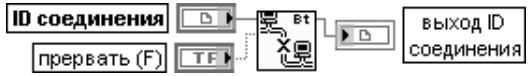


Функция записывает данные в соединение Bluetooth.
Вход данных (data in) содержит данные, которые должны быть записаны в соединение.
Выход **записанные байты** (bytes written) показывает число байтов, записанных функцией в соединение

Bluetooth Close Connection

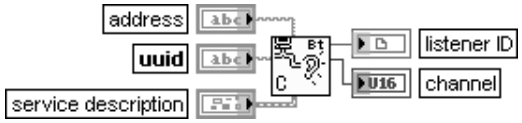


Закрыть соединение Bluetooth

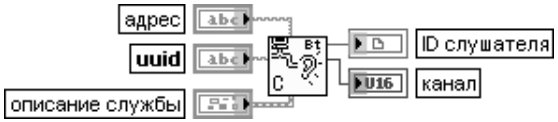


Функция закрывает ссылку, связанную с открытым соединением Bluetooth

Bluetooth Create Listener

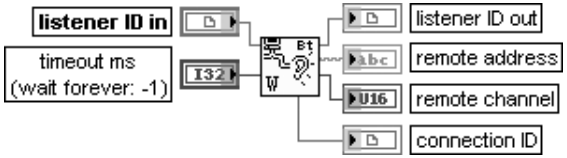


Создать слушателя Bluetooth

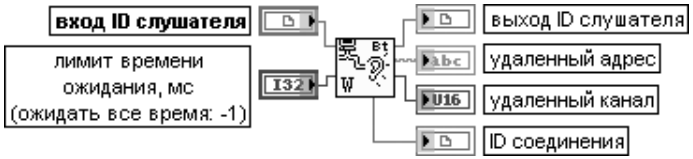


Функция создает службу для сервера Bluetooth и возвращает канал Bluetooth, который сервер может использовать для прослушивания возвращающегося соединения.
Вход **описание службы** (service description) содержит имя и описание службы, которая указана на входе **uuid**.
Выход **канал** (channel) возвращает канал, присвоенный слушателю

Bluetooth Wait On Listener

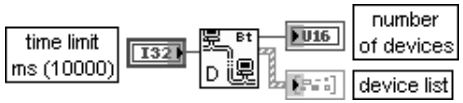


Ожидать слушателя Bluetooth

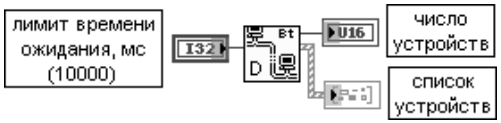


Функция ожидает слушателя для приема сетевого запроса.
Выход **удаленный адрес** (remote address) представляет адрес клиента. Пример записи адреса Bluetooth выглядит следующим образом: 00:07:E0:07:D7:50.
Выход **удаленный канал** (remote channel) возвращает канал клиента Bluetooth

Bluetooth Discover

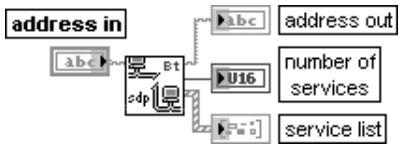


Обнаружить Bluetooth

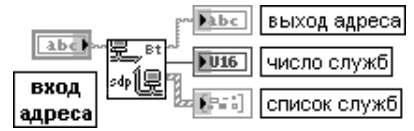


Функция производит поиск всех локально установленных или других устройств Bluetooth в радиусе действия сети Bluetooth.
Процесс обнаружения устройств Bluetooth может быть достаточно медленной операцией. Если известен адрес устройства Bluetooth, с которым необходимо установить соединение, то можно пропустить процесс обнаружения и использовать функцию **Открыть соединение Bluetooth** (Bluetooth Open Connection) для непосредственного соединения с устройством.
Выход **число устройств** (number of devices) показывает число устройств Bluetooth, обнаруженных в сети.
Выход **список устройств** (device list) возвращает список устройств, которые обнаружены функцией в сети. Выход является массивом кластеров. Каждый кластер содержит следующие элементы:
- **id устройства** (device id) – устанавливает устройство в сети Bluetooth. Этот ID используется для открытия соединения с сервером;
- **имя устройства** (device name) – отображает имя устройства Bluetooth

Bluetooth RFCOMM Service Discovery

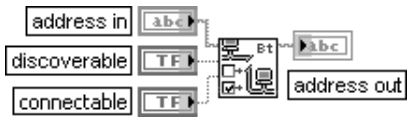


Обнаружить сервис RFCOMM Bluetooth



ВП возвращает список служб, доступных по адресу Bluetooth. Этот ВП выполняет запрос по протоколу Service Discovery Protocol (SDP) для поиска доступных служб на локальном или удаленном устройстве Bluetooth, заданном на **входе адреса** (address in).
Выход **число служб** (number of services) возвращает число служб, доступных по заданному входу адреса.
Выход **список служб** (service list) представляет массив кластеров, которые определяют каждый сервис, доступный в устройстве Bluetooth:
- **канал** (channel) – возвращает канал службы;
- **uuid** – устанавливает службу;
- **имя службы** (service name) – возвращает имя службы;
- **описание службы** (service description) – возвращает описание службы

Bluetooth Set Mode

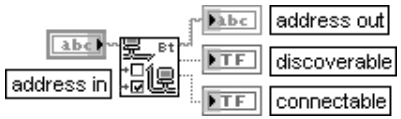


Установить режим Bluetooth

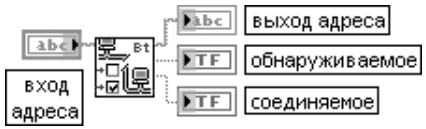
ВП устанавливает статус обнаружения и соединения локального устройства Bluetooth.
Вход адреса (address in) определяет адрес локального устройства Bluetooth. Если адрес не задан, то этот ВП пытается установить режимы обнаружения и соединения для всех локальных устройств.

Вход **обнаруживаемое** (discoverable) определяет разрешение для других устройств Bluetooth обнаруживать данное устройство. По умолчанию на данном входе установлено значение ИСТИНА (TRUE).
Вход **соединяемое** (connectable) определяет разрешение для других устройств Bluetooth соединяться с данным устройством. По умолчанию значение входа равно ИСТИНА. Если вход **обнаруживаемое** установлен в состояние ИСТИНА, а вход **соединяемое** – в состояние ЛОЖЬ, то ВП возвращает ошибку

Bluetooth Get Mode



Получить режим Bluetooth



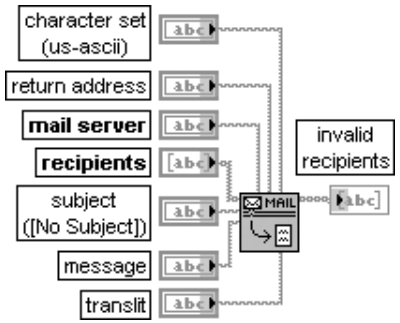
ВП возвращает статус обнаружения и соединения локального устройства Bluetooth

6.3.4. Функции электронной почты

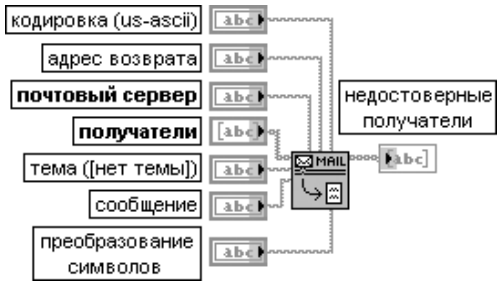
ВП электронной почты используются для отправки электронных писем, включая прикрепленные данные и файлы, с помощью протокола Simple Mail Transfer Protocol (SMTP). При этом не поддерживается идентификация для SMTP. Также не предусмотрен прием электронных писем. Для кодирования сообщений ВП электронной почты используют формат MIME (Multipurpose Internet Mail Extensions), в котором можно отправлять различные документы, в том числе файлы с двоичными данными.

Во всех ВП, за исключением SMTP Email Send Message (Small), LabVIEW поддерживает 8-битовое кодирование символов в формате UTF-8.

SMTP Email Send Message



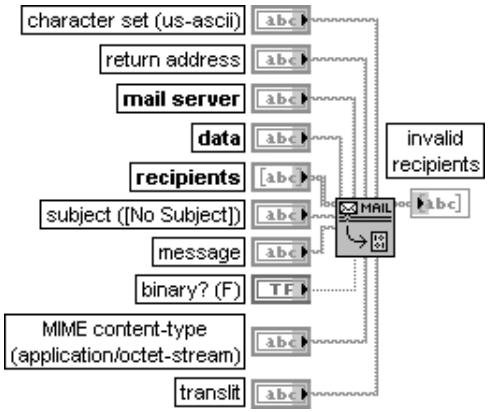
Отправить сообщение по электронной почте



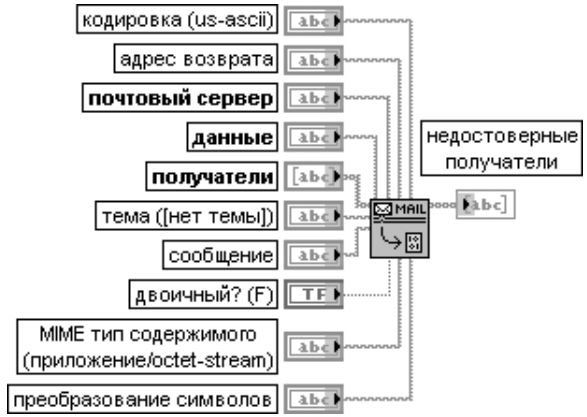
ВП отправляет текст электронного письма по списку получателей. При необходимости минимизации использования памяти целесообразно использовать ВП **SMTP Send Message**.
Вход **кодировка** (character set) устанавливает кодировку, используемую в сообщении, такую как us-ascii, iso latin-1 и macintosh.

Вход **адрес возврата** (return address) передает электронный адрес отправителя сообщения.
Вход **почтовый сервер** (mail server) представляет имя или IP-адрес сервера SMTP.
Вход **получатели** (recipients) представляет массив строк, которые содержат электронные адреса получателей сообщений. Каждый адрес может быть отдельным элементом массива.
Вход **тема** (subject) задает строку темы сообщения. По умолчанию в этой строке тема не задана [No Subject].
Вход **сообщение** (message) содержит текстовое сообщение, передаваемое по электронной почте.
Вход **преобразование символов** (translit) определяет соглашение о преобразовании символов между виртуальным набором символов и набором символов получателя. LabVIEW хранит информацию о преобразовании в списке, содержащем три элемента, разделенных запятыми: <виртуальный набор символов>, <набор символов>, <файл преобразования>. <Виртуальный набор символов> представляет результирующую строку; <набор символов> относится к набору символов получателя, а <файл преобразования> определяет информацию о преобразовании, такую как преобразование **a** в **A**.
Выход **недостоверные получатели** (invalid recipients) выводит список получателей, не воспринятых почтовым сервером

SMTP Email Send Data



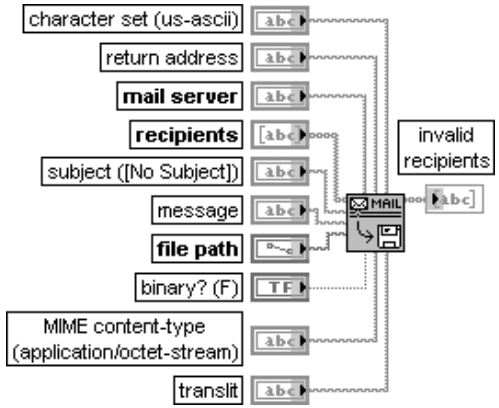
Отправить данные по электронной почте



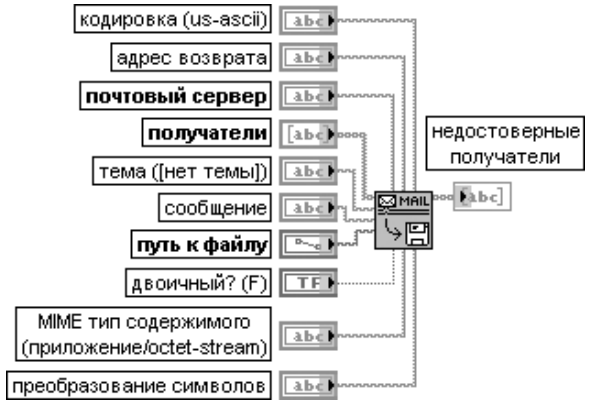
ВП отправляет электронное письмо с присоединенными данными по списку получателей. Большая часть входов идентична входам рассмотренного выше ВП **Отправить сообщение по электронной почте** (SMTP Email Send Message).
Отличие обусловлено следующими входами:
Вход **данные** (data) содержит информацию, которая присоединяется к электронному письму. Если данные представлены в двоичном формате, то необходимо установить на входе **двоичный?** (binary?) значение ИСТИНА и выбрать тип содержимого на входе **тип содержимого MIME**.
Вход **двоичный?** (binary?) определяет тип вложения – текстовое, описываемое параметром **набор символов** (character set), или же двоичным, описываемым параметром **тип содержимого MIME**. По умолчанию данный вход установлен в состояние ЛОЖЬ, которое определяет текстовое вложение.
Вход **тип содержимого MIME** (MIME content-type) определяет тип содержимого двоичного вложения к сообщению. Могут быть определены ряд типов, поддерживаемых стандартом RFC 2045, в частности следующие наиболее используемые типы:

application/octet-stream	(По умолчанию) определяет общий двоичный файл
image/jpeg	Определяет изображение в формате JPEG
text/html	Определяет документы в формате HTML

SMTP Email Send File

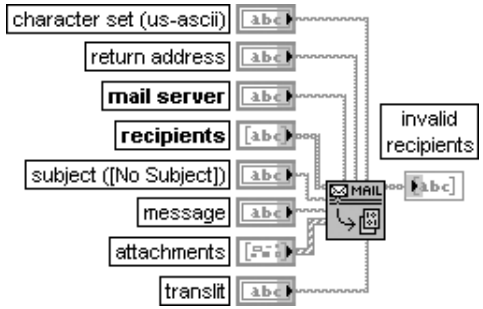


Отправить файл по электронной почте

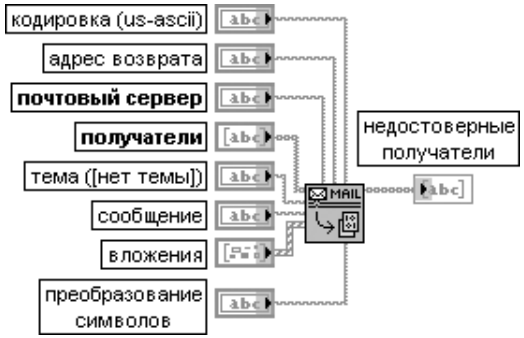


ВП отправляет электронное письмо с вложенным файлом по списку получателей. Вход **путь к файлу** (file path) определяет путь к вложенному файлу. Вложенной может быть строка или файл на локальном компьютере

SMTP Email Send Multiple Attachments



Отправить набор вложений по электронной почте



ВП отправляет электронное письмо с набором вложенных файлов и данных по списку получателей

SMTP Email Send Message (Small)

Отправить простое сообщение по электронной почте

Отправляет текстовое электронное письмо единственному получателю. В письме должны использоваться только ASCII символы. LabVIEW отправляет электронное письмо, используя ВП TCP/IP. Данный ВП целесообразно использовать при отправке писем без вложения и при необходимости минимизации использования памяти

Функции поддержки взаимодействия приложений

7

Функции поддержки взаимодействия приложений среды LabVIEW входят в категорию **Средства взаимодействия** (Connectivity) (рис. 7.1) и включают подпалитры функций обмена данными между приложениями с использованием технологии .NET и ActiveX, а также подпалитры функций создания **библиотек динамической компоновки и узлов интерфейсного кода** (Libraries & Executables), **доступа к реестру Windows** (Windows Registry Access), **управления устройствами ввода** (Input Device Control), **контроля версий** (Source Control) и **портов ввода/вывода** (Port I/O). В подпалитре Libraries & Executables размещена функция **Системная командная строка** (System Exec).

Рис. 7.1. Вид основной палитры функций поддержки взаимодействия приложений

7.1. Технология и функции ActiveX

ActiveX – это набор технологий, которые позволяют программным компонентам взаимодействовать друг с другом по сети или на локальной машине вне зависимости от того, на каком языке они написаны. В основе ActiveX лежит модель COM.

COM (Component Object Model) – модель многокомпонентных объектов, которая определяет и реализует стандартный механизм, с помощью которого одна

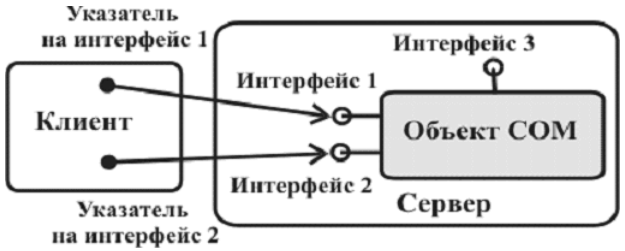


Рис. 7.2. Схема взаимодействия клиента и сервера в технологии обмена ActiveX

часть программного обеспечения предоставляет свои сервисы другой. В COM любая часть программного обеспечения реализует свои сервисы как один или несколько **объектов COM** (рис. 7.2). Каждый такой объект поддерживает один или несколько **интерфейсов**, состоящих из **методов**. **Метод** – это функция или процедура, которая выполняет некоторое действие и может быть вызвана программным обеспечением, использующим данный объект (**клиентом** объекта).

Пользователь (клиент) COM-объекта получает доступ к нему через **указатели** на интерфейсы. Объект, называемый **сервером**, организует доступ к COM-объекту, реализуя один или несколько интерфейсов. Клиент может иметь свободный доступ к объекту вне зависимости от языка реализации объекта. Объект будет вести себя в соответствии с его интерфейсами, даже если он выполняется в другом процессе или на другой машине, на другой операционной системе, написан на любом языке программирования или у него изменилась версия и он более новый или старый, чем тот, который вызывается клиентом.

Определяя интерфейсы как способ связи между объектами и их клиентами, COM эффективно решает проблему версии. При создании новой версии элемента новый интерфейс просто добавляется к объекту, оставляя старые без изменения. Поэтому клиенты, использующие старые интерфейсы, будут спокойно работать с более новыми объектами, не вызывая новых интерфейсов.

COM – это платформонезависимая, распределенная, объектно-ориентированная система для создания двоичных программных компонентов, которые могут взаимодействовать между собой.

Так как COM-компоненты являются независимыми от языка, то они могут взаимодействовать с любыми программами, реализованными на других языках. Кроме того, они могут выполняться в любом адресном пространстве: как в том, где запущен клиент, так и в другом процессе на той же машине или даже на другой машине.

COM – это основа для построения составных документов (OLE), ActiveX и т. д.

Для того чтобы понять COM, надо иметь в виду, что это не объектно ориентированный язык, а двоичный стандарт, определяющий, как COM-объекты взаимодействуют с другими объектами. Язык для реализации COM-объектов должен поддерживать **указатели** (ссылки) и вызывать функции через **указатели**.

Более подробно материал по технологии ActiveX изложен в книгах [9, 10].

С помощью ActiveX одно приложение Windows, такое, например, как LabVIEW, обеспечивает публичный набор объектов, команд и функций, к которым могут иметь доступ другие приложения Windows. При этом LabVIEW выступает в роли сервера ActiveX. LabVIEW может использоваться и как клиент ActiveX, имеющий доступ к объектам, свойствам, методам и событиям, связанным с другими поддерживающими ActiveX-приложениями. В такой роли LabVIEW может использоваться для решения следующих задач:

- открытие приложения, такого как Microsoft Excel, формирование документа и добавление данных в документ;
- внедрение документа, такого как Microsoft Word или таблицы Microsoft Excel на лицевую панель;
- размещение кнопки или другого объекта из другого приложения на лицевой панели;
- установление связи с элементом управления ActiveX, разработанным с помощью другого приложения.

Объекты ActiveX могут быть видимыми для пользователя, такими, например, как кнопки, окна, картины, документы и диалоговые окна, или невидимыми, такими как реестровые объекты приложения. В LabVIEW отображаемые объекты ActiveX, с помощью которых формируется пользовательский интерфейс, устанавливаются на лицевой панели в **контейнер ActiveX**, находящийся в палитре **Контейнеры** (Containers). Установка производится с помощью строки **Вставить объект ActiveX** (Insert ActiveX Object) контекстного меню контейнера. Методы и свойства отображаемых объектов могут устанавливаться программно с помощью **узлов методов** и **свойств**.

С помощью этих же узлов могут определяться методы и свойства неотображаемых объектов ActiveX, связанных с вызываемым приложением. Для обеспечения доступа к приложению, поддерживающему ActiveX, необходимо установить на блок-диаграмме функцию **Открыть автоматизацию** (Automation Open) и с помощью константы или элемента управления установить ссылку к приложению. Установка ссылки производится с помощью строки **Выбрать класс ActiveX** (Select ActiveX Class) контекстного меню константы или элемента управления. К выходу ссылки функции Открыть автоматизацию могут подключаться входы ссылки узлов свойств и методов. После завершения выполнения ссылка на объект ActiveX должна быть закрыта с помощью функции **Закрыть автоматизацию** (Automation Close). Закрытие ссылки удаляет объект из памяти.

При использовании контейнера ActiveX открывать и закрывать ссылку на объект ActiveX не требуется.

Еще одним элементом ActiveX являются **события** (events), которые включают такие действия над объектами, как щелчок мыши, нажатие клавиши или работа за пределами памяти. Совершение этих действий

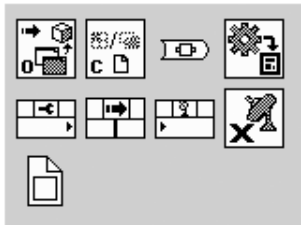


Рис. 7.3. Вид подпалитры функций ActiveX

с объектом вызывает посылку объектом соответствующего извещения и данных в контейнер ActiveX.

Перечень функций и узлов технологии ActiveX (рис. 7.3) рассмотрен в таблице.

Automation Open — **Открыть автоматизацию**

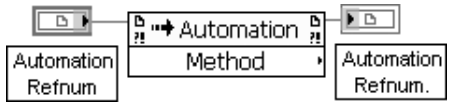
Функция возвращает ссылку автоматизации, которая указывает на определенный объект ActiveX.

Вход **ссылка автоматизации** (Automation Refnum) является ссылкой на объект ActiveX. Связь с объектом ActiveX устанавливается с помощью выбора из контекстного меню терминала на блок-диаграмме или элемента управления на лицевой панели строки **выбрать класс ActiveX** (Select ActiveX Class). В появляющемся при этом диалоговом окне **выбрать объект из библиотеки классов** (Select Object From Type Library) необходимо выбрать требуемый объект.

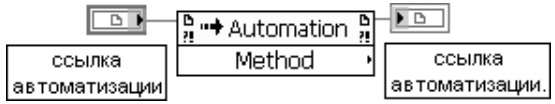
Вход **имя компьютера** (machine name) определяет машину, на которой ВП открывает **ссылку автоматизации**. Если имя компьютера не задано, то объект будет открыт на локальном компьютере. Если вход **открыть новый экземпляр** (open new instance) находится в состоянии ИСТИНА, то LabVIEW создает новый экземпляр **ссылки автоматизации**. Если вход находится в состоянии ЛОЖЬ (по умолчанию), то LabVIEW пытается соединиться с экземпляром ссылки, который уже открыт. Если попытка оказывается безуспешной, то LabVIEW открывает новый экземпляр.

Закрытие ссылки осуществляется с помощью функции **Закрыть ссылку** (Close Reference), рассмотренной ниже

Invoke Node



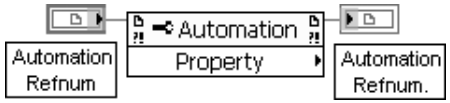
Узел вызова



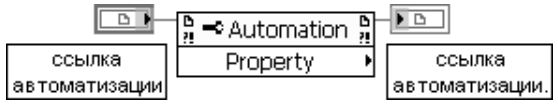
Узел вызывает метод или действие по ссылке. Большинство методов имеют связанные с ними параметры.

Вход **ссылка** (reference) является ссылкой, связанной с объектом, который вызывает метод или выполняет действие. Ссылка может быть получена от функции **Открыть автоматизацию** (Automation Open) или от элемента управления **ссылка** (Refnum) лицевой панели. **Узел вызова** (Invoke Node) автоматически адаптируется к классу объекта, задаваемого по ссылке

Property Node

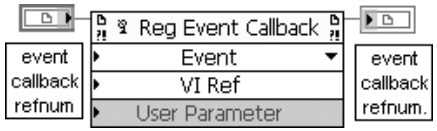


Узел свойства

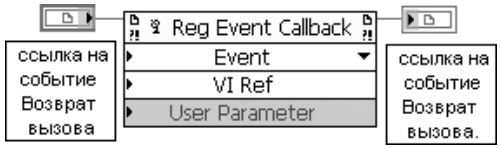


Узел получает (читает) и/или устанавливает (записывает) свойства по **ссылке** (reference). **Узел свойства** (Property Node) автоматически адаптируется к классу объекта, задаваемого по ссылке.
Выбор конкретного свойства при установленной ссылке производится с помощью строки **свойства** (Properties) контекстного меню полоски узла с надписью **свойство** (Property). С помощью этого же меню устанавливается режим чтения или записи свойств

Register Event Callback



Регистрация события Возврат вызова



Узел регистрирует ВП, который вызывается при наступлении события. LabVIEW использует тип входной ссылки, подключенной к каждому пункту, для определения событий, которые могут быть зарегистрированы. Вход ссылки ВП должен быть строгого типа, чтобы соответствовать данным, которые предоставляются регистрируемыми событиями. При наступлении события LabVIEW передает дополнительные параметры вызываемому ВП. Размер узла может быть увеличен для регистрации сразу нескольких вызовов событий.
Вход **ссылка на событие возврат вызова** (event callback ref) принимает ссылку к существующей регистрации события.
Вход **ссылка к источнику события** (event source ref) принимает ссылку на автоматизацию ActiveX. Стрелка в правой части полоски Event, направленная вниз, позволяет выбрать тип события, которое пользователь хочет сгенерировать. Ссылка должна указывать на локальные объекты. Нельзя подключать ссылку к удаленному объекту.
Вход **ссылка ВП** (VI reference) является ссылкой ВП строгого типа к ВП, который вызывается из приложения, сгенерировавшего событие. National Instruments рекомендует, чтобы используемый при вызове ВП был реентерабельным. Если ВП не является реентерабельным, то он не может быть вызван одновременно при многократном повторении события.
Входы **ссылка к источнику события** и **ссылка ВП** являются обязательными.
Вход **параметр пользователя** (User Parameter) содержит данные, которые LabVIEW передает пользователю через вызываемый ВП, когда объект ActiveX генерирует событие.
Выход **ссылка на событие возврат вызова** (event callback ref out) возвращает ссылку к новой или существующей регистрации события.

В качестве примера использования функции **Регистрация события Возврат вызова** на рис. 7.4 приведена блок-диаграмма ВП **Событие Возврат вызова ActiveX для Excel** (ActiveX Event Callback for Excel) из набора примеров NI Example Finder.
ВП использует функцию **Регистрация события Возврат вызова** вместе с константой ссылки на приложение Excel. При запуске ВП открывает Excel и, после того как пользователь последовательно выберет в панели меню Excel строки **Файл** Ю **Создать...** и **Новая книга** (New Workbook), то есть создаст событие, регистрация которого предусмотрена в рассматриваемом ВП, произойдет заполнение данными (названиями дней недели) колонки книги Excel. Само заполнение данными производится ВП NewWorkbookCallback.vi, ссылка на который подключена ко входу VI Ref функции **Регистрация события Возврат вызова**.

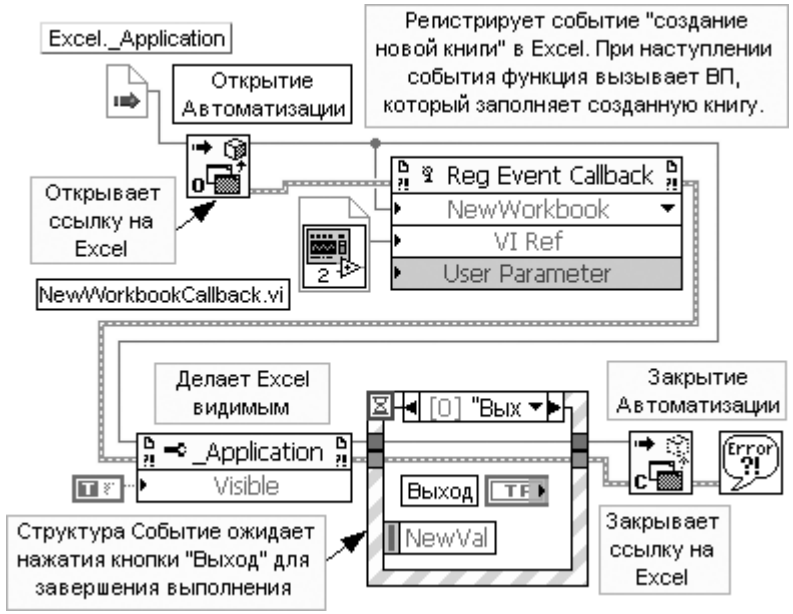
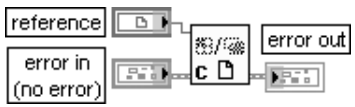
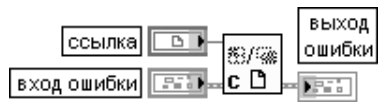


*Рис. 7.4. Блок-диаграмма ВП **Событие Возврат вызова ActiveX для Excel***

Close Reference



Закрыть ссылку



Функция закрывает ссылку, связанную с открытым ВП, объектом ВП, открытым экземпляром LabVIEW, объектом ActiveX или объектом .NET. LabVIEW закрывает ссылку к элементу управления, когда она уже больше не нужна

В LabVIEW с помощью контейнеров ActiveX реализованы трехмерные графики. Поэтому при просмотре набора примеров NI Example Finger по **задачам** (task) большая часть примеров в разделе ActiveX ⇒ General посвящена именно этим графикам. В качестве примера использования функций ActiveX для управления при-

ложением можно привести размещенный в этом же разделе ВП **Слайд-шоу** (Slideshow), блок-диаграмма которого приведена на рис. 7.5. В этом ВП функции ActiveX используются для управления приложением Power Point. Во втором разделе ActiveX ⇒ Excel, помимо рассмотренного выше ВП **Событие Возврат вызова ActiveX для Excel** приведен ряд ВП, использующих технологию ActiveX для передачи данных из LabVIEW в Excel. На рис. 7.6 приведен еще один простой пример использования технологии ActiveX для программного вызова сервера DataSocket.

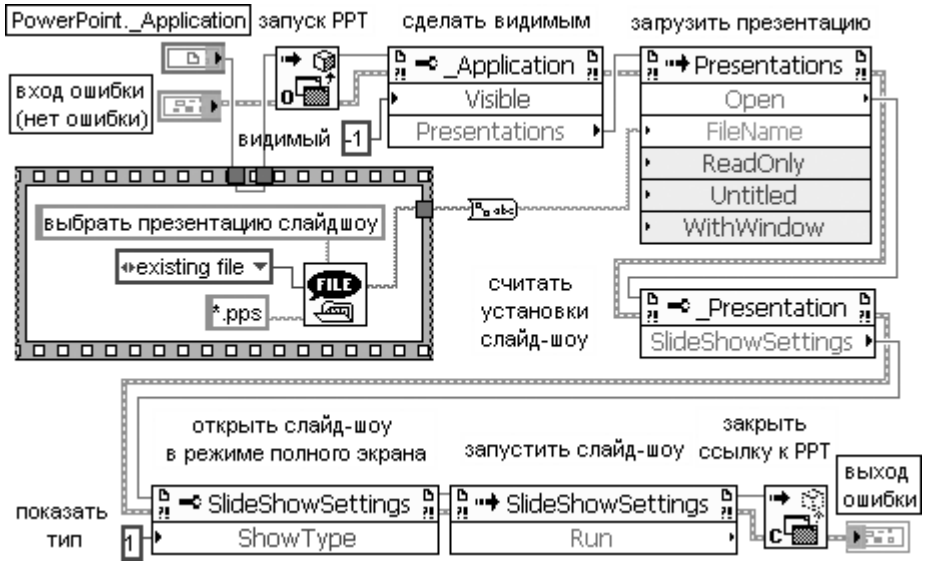


*Рис. 7.5. Блок-диаграмма ВП **Слайд-шоу** (Slideshow)*

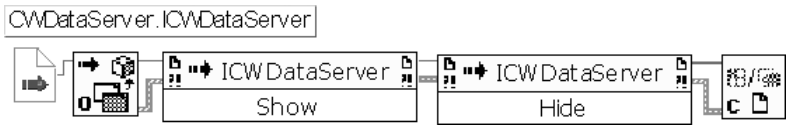


Рис. 7.6. Блок-диаграмма ВП программного вызова сервера DataSocket

7.2. Технология и функции .NET

Microsoft .NET [11] является дальнейшим развитием технологии ActiveX. Так же как и ActiveX, .NET используется в LabVIEW для обеспечения доступа к другим приложениям Windows. LabVIEW может использоваться как клиент .NET для доступа к объектам, свойствам и методам, связанным с серверами .NET. В то же время LabVIEW не является сервером .NET, то есть другие приложения не могут непосредственно взаимодействовать с LabVIEW через .NET.

С помощью ВП, использующих .NET, можно получить доступ к сервисам Windows и API. Среда периода выполнения .NET framework включает сервисы компонентов COM+, среду периода выполнения для Web ASP.NET и поддержку ряда протоколов сервисов Web, таких как SOAP, WSDL и UDDI.

.NET framework является программной базой для среды .NET при построении, развертывании и работе Web-приложений и XML Web-сервисов.

К числу основных элементов среды .NET относятся **общий язык периода выполнения** (Common Language Runtime (CLR)), **библиотеки классов** (Class Libraries) и **сборки** (Assemblies).

Общий язык периода выполнения (CLR) представляет набор библиотек, отвечающих за сервисы периода выполнения, такие как языковая интеграция, обеспечение безопасности доступа к программам, управление памятью, сборка мусора, управление процессами и потоками. CLR образует основу .NET и использует промежуточный язык (intermediate language – IL) для облегчения взаимодействия программ, разработанных в среде .NET, с другими программами.

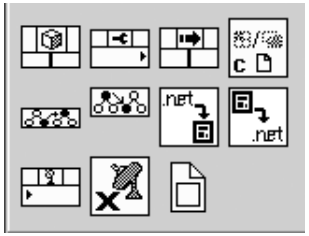
Для помощи во взаимодействии .NET с различными программами CLR обеспечивает систему типов данных, которая разделяет области языков программирования и операционной системы. Пользователи могут использовать CLR, чтобы видеть систему как набор типов данных и объектов, а не набор памяти и потоков. CLR требует от трансляторов формировать информацию в формате метаданных CLR IL. В системе Win32 все программные языковые компиляторы генерируют код CLR IL, а не ассемблерный код.

Библиотеки классов (Class Libraries) представляют набор классов, которые обеспечивают стандартную функциональность, такую как ввод и вывод, обработку строк, управление безопасностью, сетевое взаимодействие, управление потоками, управление текстом, средства конструирования пользовательского интерфейса и т. д. Эти классы содержат такие же функции, что и система Win32/COM. В .NET framework пользователь может применять классы, созданные на одном языке, в другом языке .NET.

Сборки (Assemblies) представляют единицы развертывания приложения, аналогичные DLL, OCX или EXE-компонентам COM. Сборки являются DLL- или EXE-файлами, которые разработаны с помощью .NET CLR. Сборки могут состоять из одного или множества файлов. Сборка включает **декларацию** (manifest), которая содержит информацию о имени и версии сборки, локальную информацию, список файлов, составляющих сборку, список зависимых сборок, ресурсов и экспортируемых типов данных. Однофайловые сборки содержат все данные в одном файле, включая декларацию и любые ресурсы, которые ей необходимы. Многофайловые сборки могут иметь внешние ресурсы, такие как файлы изображений, звуковые файлы и т. п., или иметь один файл для кода ядра и другие для библиотек помощников.

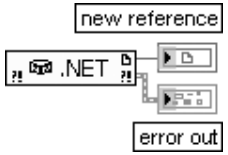
Сборки могут быть публичными (открытыми) или приватными (закрытыми). .NET требует, чтобы закрытые сборки находились в одном каталоге с каталогом приложения, а открытые сборки – в каталоге, который называется глобальный кэш сборок (Global Assembly Cache (GAC)). Глобальный кэш сборок аналогичен реестру для COM-объектов. Он может находиться в разделе \\winnt\assembly.

Рис. 7.7. Вид подпалитры функций .NET



Функции подпалитры .NET (рис. 7.7) рассмотрены ниже в таблице.

Constructor Node

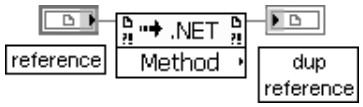


Узел конструктора

Узел формирует экземпляр объекта .NET. Этот узел определяет конструктора, который должен создать объект .NET. При помещении узла на блок-диаграмму LabVIEW отображает диалоговое окно **Выбрать конструктора .NET** (Select .NET Constructor). Это же окно можно вызвать и двойным щелчком мыши на данном узле. Некоторые конструкторы могут содержать параметры инициализации, которые можно использовать для создания объектов .NET с конкретным состоянием. Не все конструкторы имеют параметры инициализации.

Выход **новая ссылка** (new reference) возвращает ссылку на объект .NET. Эта ссылка может передаваться далее для определения методов и свойств объектов

Invoke Node

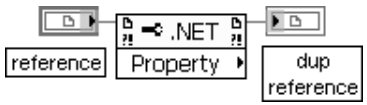


Узел метода

Узел вызывает метод или действие по ссылке. Большинство методов имеют связанные с ними параметры. Ссылка может быть получена с выхода функции **Узел конструктора** (Constructor Node). Узел метода автоматически адаптируется к классу объекта, задаваемого по ссылке.

Выбор конкретного метода при установленной ссылке производится с помощью строки **методы** (Methods) контекстного меню полоски узла с надписью **метод** (Method)

Property Node

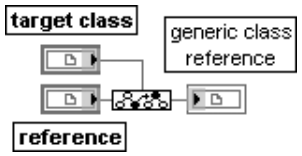


Узел свойства

Узел получает (читает) и/или устанавливает (записывает) свойство по **ссылке** (reference). **Узел свойства** (Property Node) автоматически адаптируется к классу объекта, задаваемого по ссылке.

Выбор конкретного свойства при установленной ссылке производится с помощью строки **свойства** (Properties) контекстного меню полоски узла с надписью **свойство** (Property). С помощью этого же меню устанавливается режим чтения или записи свойств

To More Generic Class



К более общему классу

Функция формирует ссылку к более общему классу в иерархии наследования.

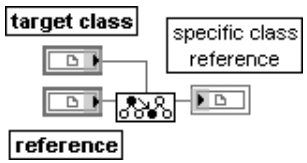
Вход **целевой класс** (target class) определяет класс, до которого повышается **ссылка** (reference). К этому входу можно подключить константу описателя класса или ссылку .NET.

Вход **ссылка** (reference) задает ссылку элемента управления или ссылку .NET, которая преобразуется к более общему классу.

Выход **ссылка общего класса** (generic class reference) содержит ссылку на элемент управления более общего класса. Если выполнение функции привело к ошибке, то на выходе **ссылка общего класса** возвращается значение **Не ссылка** (Not A Refnum).

Каждое свойство и метод принадлежат к определенному классу. Классы расположены в иерархии, в которой каждый класс наследует свойства и методы, связанные с классом предыдущего уровня. Так, например, кнопка является членом логического класса, который имеет набор присущих только ему свойств, таких как ширина и высота кнопки. Кроме того, все логические элементы являются членами класса элементов управления, который включает свойства, обнаруживаемые в большинстве других элементов управления и индикации лицевой панели, таких как видимость, надпись и свойства по умолчанию

To More Specific Class



К более определенному классу

Функция формирует ссылку к более определенному классу в иерархии наследования.

Целевой класс (target class) представляет класс, до которого предполагается уменьшить уровень ссылки

Для иллюстрации использования технологии .NET в разделе **Связь с внешними приложениями ⇒ .NET** (Communicating with External Applications ⇒ .NET) набора примеров NI Example Finger приведены ВП **Калькулятор** (Calculator) и **Простой монитор задач** (SimpleTaskMonitor). На рис. 7.8 приведена блок-диаграмма упрощенного ВП **Калькулятор**, позволяющего складывать два числа и отображать результат.

Для обеспечения работоспособности этого ВП и выбора других методов целесообразно сохранить его в каталоге вместе с копией DLL Calculator.dll, которая хранится вместе с указанными выше примерами, а при установке конструктора в диалоговом окне **Выбрать конструктор .NET** с помощью кнопки **Просмотреть** (Browse) указать путь к этой DLL.

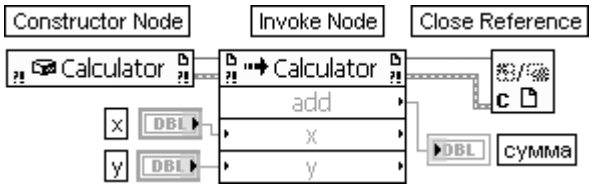


Рис. 7.8. Блок-диаграмма упрощенного ВП ***Калькулятор*** *(Calculator)*

7.3. Разработка библиотек динамической компоновки

Одним из достоинств пакета LabVIEW является возможность вызова процедур, написанных на других языках, с помощью подключения библиотек динамической компоновки (Dynamic Link Library, DLL). Для создания таких библиотек можно использовать любой компилятор процедурного языка, позволяющий создавать библиотеки динамического связывания для Windows. Необходимость применения DLL, в частности, возникает при разработке драйверов оригинальных устройств ввода/вывода или передачи данных, в которых используются комплектующие, имеющие сложную логику программирования. В ряде случаев базовые драйверы для таких микросхем или плат поставляются их производителями в виде набора DLL, рассчитанных на применение в текстовых языках.

В LabVIEW для вызова DLL служит узел **Вызов библиотечной функции** (Call Library Function Node)  , находящийся в подпалитре **Библиотеки и исполняемые программы** (Libraries & Executables), которая входит в состав категории **Средства взаимодействия** (Connectivity). Первоначально функция не имеет параметров и возвращает тип void. Настройка атрибутов функции производится с помощью диалогового окна Вызов библиотечной функции (рис. 7.9), вызываемого двойным щелчком ЛКМ на иконке функции или выбором пункта **Конфигурировать** (Configure) в ее контекстном меню.

В LabVIEW 8.20 это диалоговое окно содержит три страницы: **Функция** (Function) (рис. 7.9), **Параметры** (Parameters) и **Обратные вызовы** (Callbacks).

На странице **Функция** поле **имя или путь библиотеки** (Library Name or Path) служит для указания полного пути к двоичному файлу откомпилированной внешней библиотеки. При установке флажка **определять путь на диаграмме** (Specify path on diagram) путь может быть задан на входе узла.

В поле **имя функции** (Function Name) указывается имя функции, находящейся в вызываемой библиотеке. Имя должно быть соблюдено с точностью до регистра, так как система Windows и компиляторы языка **C** различают строчные и прописные буквы.

В правой части страницы находятся две группы переключателей: **поток** (Thread) и **соглашения вызова** (Calling Conventions). В первую группу входят переключатели **Выполняться в потоке пользовательского интерфейса** (Run in UI Thread) и

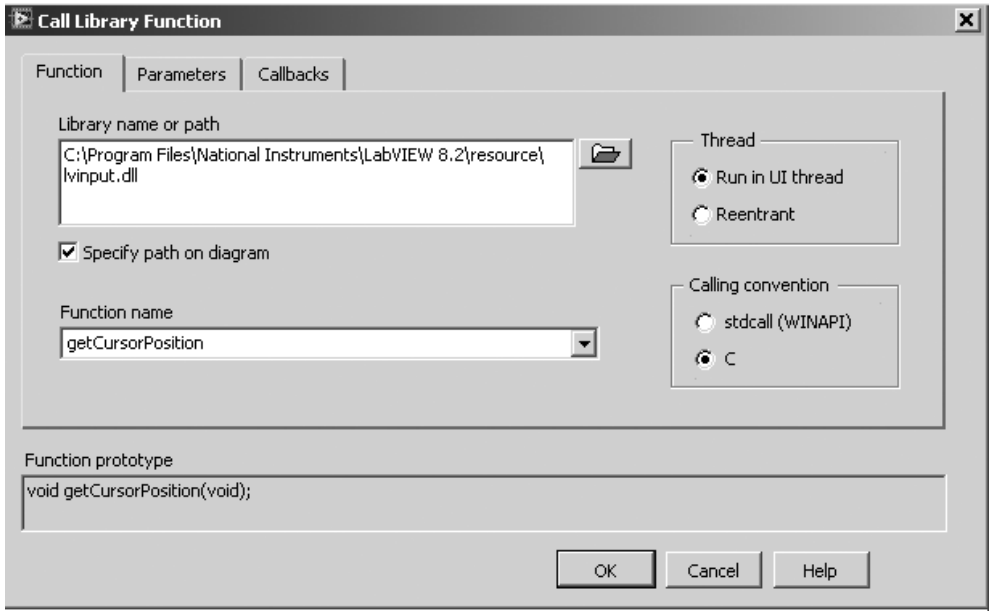


Рис. 7.9. Вид страницы ***Функция*** *диалогового окна узла* ***Вызов библиотечной функции***

Выполняться с возможностью повторного вызова (Reentrant). Они определяют способность функции работать в многопоточной среде. Так, например, если функция может выполняться одновременно только в одном потоке, то следует выбрать пункт Run in UI Thread. Если же функция способна работать в нескольких потоках выполнения, то есть является **реентерабельной**, то выбирается пункт Reentrant. Хотя такие функции и дают некоторые преимущества по сравнению с нереентерабельными, их написание является непростой задачей.

Переключатели группы **соглашения вызова** определяют стандарт вызова функции. LabVIEW поддерживает два стандарта:

- **Stdcall** – стандарт, принятый для системных вызовов и библиотек динамической компоновки Windows. Также этот стандарт принят как основной для языка Pascal;
- **C** – стандарт вызовов языка **C** (по умолчанию).

Следующая страница **Параметры** (рис. 7.10) позволяет конфигурировать параметры функции. На ней размещено окно с перечнем параметров, набор кнопок для добавления или удаления параметров в этом окне и набор окон и выпадающих списков для установления конкретных значений для **текущего параметра** (Current parameter). Следует быть внимательным при настройке данных опций. LabVIEW не может проверить соответствие указанного количества и типов аргументов реальным параметрам. Ошибка, допущенная при описании прототипа вызова функции, может приводить к краху среды LabVIEW и ее выгрузке из памяти.

Сразу после вызова функция имеет только один аргумент **возвращаемый тип** (return type), который описывает возвращаемое функцией значение и может при-

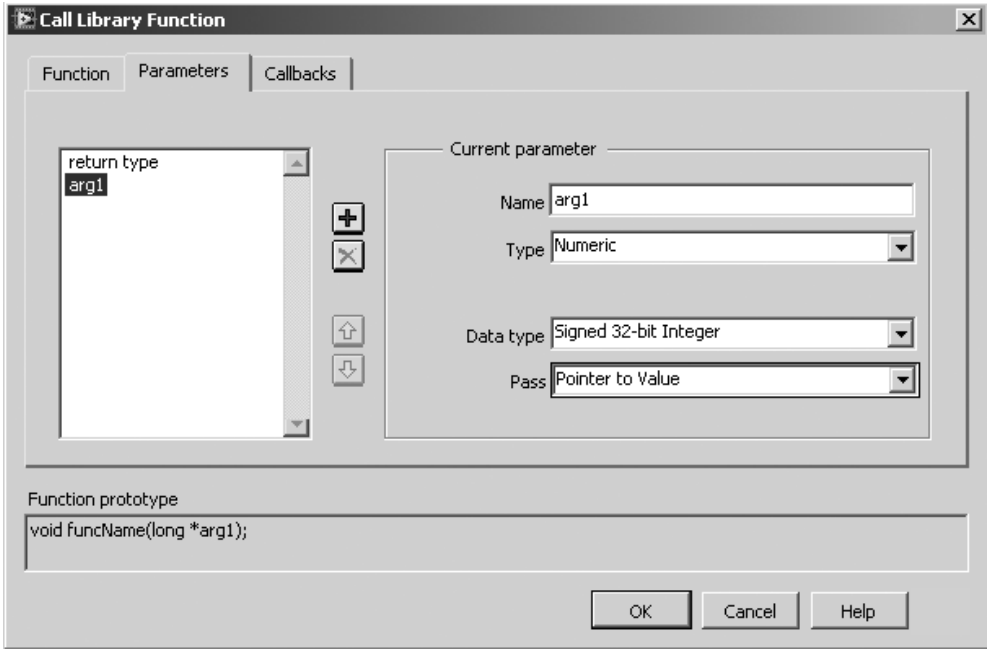


Рис. 7.10. Вид страницы ***Параметры*** *диалогового окна* ***Вызов библиотечной функции***

надлежать только к типам **пустой** (Void), **числовой** (Numeric) и **строковый** (String). Для добавляемых параметров в качестве аргументов LabVIEW позволяет передавать данные следующих типов: **числовой** (Numeric), **массив** (Array) **строковый** (String), **осциллограмма** (Waveform), **цифровая осциллограмма** (Digital Waveform), **цифровые данные** (Digital Data), ActiveX, **адаптивный к типу** (Adapt to Type) и **экземпляр указателя данных** (Instance Data Pointer).

Тип **пустой** (Void) является чисто абстрактным типом и указывает на отсутствие факта передачи данных. Так, например, функция, имеющая параметр **возвращаемый тип** типа Void, не будет возвращать никаких данных и будет являться процедурой. В этом случае верхняя пара выводов функции не используется.

Типу **числовой** (Numeric) в LabVIEW принадлежат 10 различных подтипов, 8 из которых – целочисленные и 2 – вещественные. Так как внешние функции не могут получать аргументы обобщенного типа **числовой**, то для каждого из них должен быть указан конкретный подтип, который и будет передан в качестве аргумента. Далее в таблице приведено соответствие числовых подтипов и типов языка **C**.

Тип числа	Тип LabVIEW (поле Data Type)	Тип языка C
8-разрядное целое со знаком	Signed 8-bit Integer	signed char
16-разрядное целое со знаком	Signed 16-bit Integer	signed short int

Тип числа	Тип LabVIEW (поле Data Type)	Тип языка C
32-разрядное целое со знаком	Signed 32-bit Integer	signed long int
64-разрядное целое со знаком	Signed 64-bit Integer	
8-разрядное целое без знака	Unsigned 8-bit Integer	unsigned char
16-разрядное целое без знака	Unsigned 16-bit Integer	unsigned short int
32-разрядное целое без знака	Unsigned 32-bit Integer	unsigned long int
64-разрядное целое без знака	Unsigned 64-bit Integer	
Вещественное одинарной точности	4-byte Single	float
Вещественное двойной точности	8-byte Double	double

При добавлении числового аргумента наряду с полем **тип данных** (Data Type) также появляется поле **передавать** (Pass), определяющее способ передачи параметра. Это поле имеет две опции:

- **Значение** (Value) – в стек вызовов помещается значение параметра, поэтому даже если внешняя функция изменит значение своего аргумента, то значение аргумента, хранящееся в памяти LabVIEW, не изменится, и значения на выходных терминалах функции **Вызов библиотечной функции** будет равно соответствующим входным терминалам/аргументам;
- **Указатель (ссылка) на значение** (Pointer to Value) – в стек вызовов помещается указатель на значение параметра. Это позволяет изменять значение параметра, хранящееся в памяти LabVIEW, и возвращать из функции более одного значения.

При выборе типа данных **массив** (Array) необходимо определить тип данных элементов массива (используя типы данных числовых величин), его **размерность** (Dimensions) и **формат массива** (Array Format), используемый при передаче массива. Предусмотрены следующие форматы массива: Array Data Pointer, Array Handle, Array Handle Pointer.

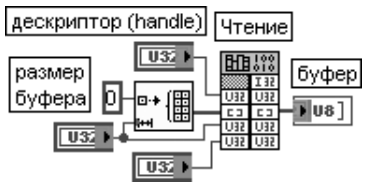


Рис. 7.11. Блок-диаграмма ВП чтения данных по шине USB

При выборе формата **указатель на данные массива** (Array Data Pointer) передается указатель на данные массива. При этом размеры массива по каждой размерности должны быть переданы в DLL в виде отдельных переменных. На рис. 7.11 показана блок-диаграмма ВП чтения данных по шине USB с помощью интерфейсной микросхемы FT232B [12], в котором для передачи массива используется формат **указатель на данные массива**.

При передаче массива с помощью формата **указатель на дескрипторы массива** (Array Handle Pointer) нет необходимости вводить отдельный параметр для передачи размера массива.

При выборе **строкового** (String) типа параметра появляющееся поле **формат строки** (String Format) определяет вид, в котором строка будет передана внешней функции. Предусмотрено 4 возможных варианта:

- **Указатель на строку C** (C String Pointer) – в функцию передается указатель на начало так называемой ASCIIZ строки. Символы в строках такого типа идут последовательно друг за другом, начиная с адреса, на который ссылается указатель. Длина строк этого типа ограничена лишь объемом памяти компьютера, однако эти строки не могут содержать символы с кодом 0, так как этот символ зарезервирован для обозначения конца строки. Поэтому передача двоичных данных через строки данного типа невозможна. Указатели этого типа используются в большинстве приложений Win32 API;
- **Указатель на строку Pascal** (Pascal String Pointer) – строки этого типа имеют структуру, несколько отличную от строк языка **C**. На первом месте в строке стоит байт, указывающий длину строки, за которым следуют байты строки. Таким образом, строка такого типа имеет следующий вид:

n	C_1	C_2	C_3		C_{n-1}	C_n

Хотя строки такого формата и не имеют недостатков ASCIIZ строк, они ограничены в длину 255 символами (максимальным значением байта длины строки);

- **Дескриптор строки LabVIEW** (LabVIEW String Handle) – передает указатель на указатель строки, в начале которого расположены четыре байта с информацией о длине строки;
- **Ссылка на дескриптор строки LabVIEW** (LabVIEW String Handle Pointer) – передает указатель на дескриптор строки в виде 4-байтного значения.

При вызове общей библиотеки, содержащей тип данных **осциллограмма**, по умолчанию устанавливается числовой подтип **Вещественное двойной точности** (8-byte Double). Однако параметр **размерность** (Dimensions) должен быть обязательно установлен. Его значение равно 0 для отдельной осциллограммы и 1 – для массива осциллограмм. Это же соотношение справедливо для **цифровой осциллограммы** (Digital Waveform) и для **цифровых данных** (Digital Data).

Для типа **ActiveX** предусмотрены следующие опции в меню **тип данных** (Data Type):

- **Указатель на данные типа Вариант** (ActiveX Variant Pointer) – передает указатель на данные типа Вариант ActiveX;
- **Указатель Idispatch*** (Idispatch* Pointer) – передает указатель на интерфейс Idispatch для сервера автоматизации ActiveX;
- **Указатель IUncnown** (IUncnown Pointer) – передает указатель на интерфейс IUncnown для сервера автоматизации ActiveX.

В нижней части диалогового окна находится поле **прототип функции** (Function Prototype), которое содержит прототип описанной в диалоговом окне функции для языка **C**.

Ниже приведен фрагмент листинга программы, которая использовалась для формирования DLL считывания данных по шине USB (рис. 7.11).

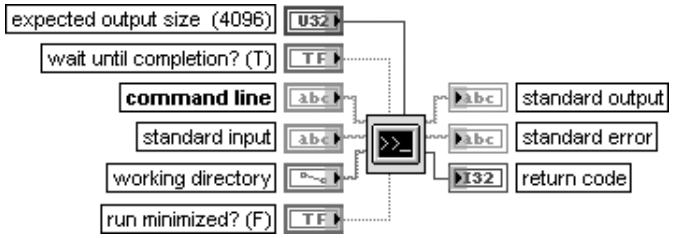
```
#include «stdafx.h»          //Стандартный заголовочный файл
#include «ftd2xx.h»          //Заголовочный файл ftd2xx.lib
```



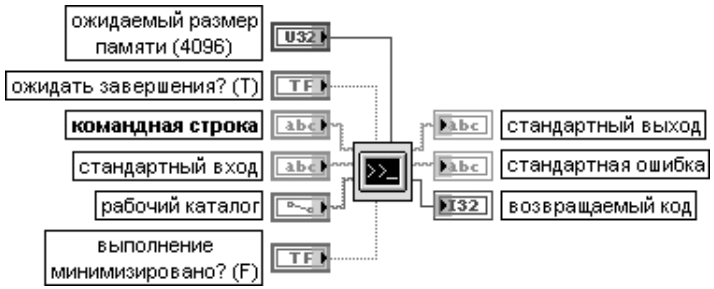
```
#include «ft2xxLV.h»          //Заголовочный файл библиотеки пользователя

//Вызывается при загрузке данной DLL функцией LoadLibrary()
BOOL APIENTRY DllMain( HANDLE hModule, DWORD ul_reason_for_call, LPVOID
lpReserved )
{
 return TRUE;
}
//Экспортируемая функция
//Указатель на данную функцию можно получить, используя функцию
GetProcAddress()
//При вызове из LabVIEW вызов GetProcAddress() происходит неявно
_declspec (dllexport) int Read(unsigned long handle, BYTE *buffer, DWORD
RxBytes, DWORD *BytesReceived) {
        FT_STATUS ftStatus;
        ftStatus = FT_Read(handle, buffer, RxBytes, &*BytesReceived);
        if (ftStatus == FT_OK) {
                return 1;
        }
        return 0;
}
```

В подпалитре функций создания **библиотек динамической компоновки и узлов интерфейсного кода** (Libraries & Executables) находится также ВП **Системная командная строка** (System Exec), рассмотренный ниже.

System Exec



Системная командная строка



ВП выполняет системную команду.
Для запуска программы с опциями необходимо использовать синтаксис имяфайла.exe -опция1 -опция2.
Необходимо также создать файл с расширением *.bat, который будет вызывать исполняемый файл с опциями и использовать данный ВП для вызова этого файла.
Вход **ожидаемый размер памяти** (expected output size) используется для повышения эффективности использования памяти. Необходимо устанавливать размер памяти немного больше ожидаемого. Команда будет выполняться и при превышении размера, однако LabVIEW будет использовать память менее эффективно. По умолчанию размер памяти равен 4096.
Если на входе **ожидать завершения** (wait until completion?) установлено значение ИСТИНА, то **стандартный вход** (standard input) становится доступным для ввода команды и выходы **стандартный выход** (standard output) и **стандартная ошибка** (standard error) становятся доступными после завершения выполнения команды. Если на этом входе установлено значение ЛОЖЬ, то команда выполняется в фоновом режиме и ее вход и выход недоступны.
Вход **командная строка** (command line) содержит команду, передаваемую в LabVIEW для вызова исполняемой программы. В системе Windows при использовании команд DOS необходимо перед командой вставить command.com /C.
Вход **стандартный вход** (standard input) содержит текст, передаваемый в командную строку как стандартный вход.
Вход **рабочий каталог** (working directory) содержит путь каталога, из которого должна выполняться команда.
Если вход **выполнение минимизировано?** (run minimized?) установлен в состояние ИСТИНА, то ВП минимизирует выполнение программы. По умолчанию значение входа равно ЛОЖЬ. **Выполнение минимизировано?** не используется на платформах UNIX.
Выход **возвращаемый код** (return code) отображает системно-зависимый выходной код, возвращаемый командой

7.4. ВП доступа к реестру Windows

Реестр системы Windows представляет специальную базу данных, в которой находится почти вся информация, необходимая для загрузки и конфигурирования системы и настройки ее под конкретного пользователя [13]. Реестр похож на файловую систему, поскольку состоит из набора каталогов, каждый из которых содержит либо подкаталоги, либо записи. Отличие заключается в том, что корпорация Microsoft называет каталог реестра **ключом** (key), при этом все каталоги верхнего уровня начинаются со строки HKEY, что означает «дескриптор ключа».

В нижней части иерархической структуры реестра располагаются записи, называемые **значениями**. Каждое значение имеет три части: имя, тип и данные. Имя представляет строку формата Unicode. Тип может быть одним из стандартных типов. Наиболее часто используются строки формата Unicode, 32-разрядные целые числа, двоичные числа произвольной длины и символьные ссылки на каталог или запись реестра. На верхнем уровне в реестре Windows имеются шесть ключей, называемых **корневыми ключами**.

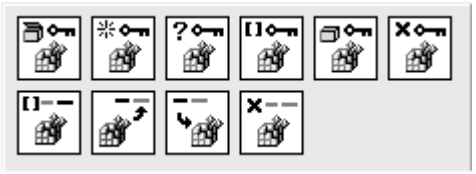
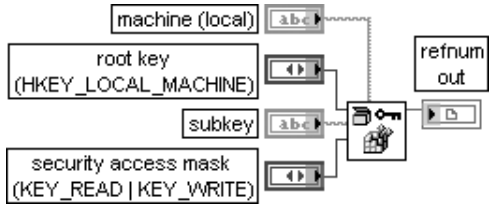


Рис. 7.12. Вид подпалитры ВП доступа к реестру Windows

С помощью ВП доступа к реестру Windows (рис. 7.12) можно создавать, открывать, запрашивать, перечислять, закрывать и удалять ключи реестра Windows. Также можно перечислять, считывать, записывать и удалять значение ключей реестра Windows. Необходимо отметить, что описание всех ВП доступа к реестру Windows в разделе LabVIEW Help сопровождается предупреждающим сообщением о том, что некорректное изменение реестра может повредить Windows или затруднить ее запуск.

Open Registry Key



Открыть ключ реестра

ВП открывает ссылку к ключу или подключу в реестре Windows.
Вход **машина** (machine) определяет имя сетевой машины. По умолчанию задается локальная машина.
Вход **корневой ключ** (root key) определяет корневой ключ Windows. Предусмотрены следующие варианты установки этого входа:

Ключ	Описание
HKEY_CLASSES_ROOT	Ссылка на подключ HKEY_LOCAL_MACHINE\SOFTWARE\CLASSES
HKEY_CURRENT_USER	Ссылка на текущий профиль пользователя
HKEY_LOCAL_MACHINE	Свойства аппаратного и программного обеспечения
HKEY_USERS	Информация о пользователях
HKEY_CURRENT_CONFIG	Ссылка на текущий профиль аппаратного обеспечения

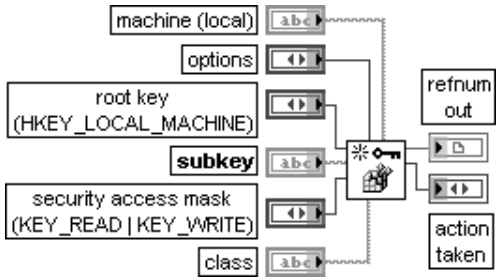
Вход **подключ** (subkey) представляет имя подключа корневого ключа. Начальный символ обратного слэша «\» может вызвать ошибку.
Вход **маска безопасности доступа** (security access mask) определяет права доступа к назначению ключа. Предусмотрены следующие варианты установки этого входа:

Ключ	Описание
KEY_QUERY_VALUE	Запросить значение ключа
KEY_SET_VALUE	Установить значение ключа
KEY_CREATE_SUB_KEY	Создать подключи ключа

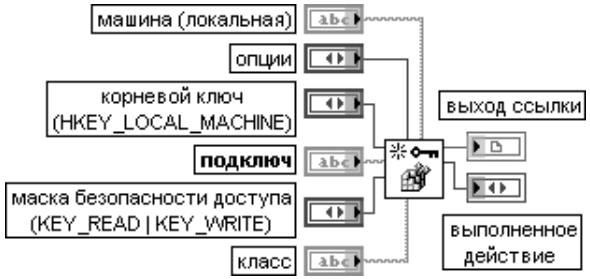
Ключ	Описание
KEY_ENUMERATE_SUB_KEYS	Перечислить подключи ключа
KEY_NOTIFY	Оповестить ключ
KEY_CREATE_LINK	Создать связь ключа
KEY_READ	Считать ключ
KEY_WRITE	Записать ключ
KEY_ALL_ACCESS	Общий доступ к ключу

Выход ссылки (refnum out) содержит ссылку к открытому ключу

Create Registry Key



Создать ключ реестра



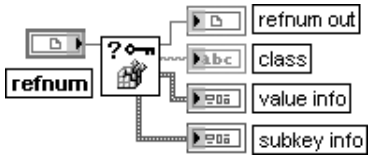
ВП создает ключ в реестре Windows или открывает его, если он уже существует.
Вход **опции** (options) определяет специальные опции для ключа:

Опция ключа	Описание
REG_OPTION_NON_VOLATILE	Ключ с присвоенным значением
REG_OPTION_VOLATILE	Ключ без присвоенного значения
REG_OPTION_BACKUP_RESTORE	Дублирование ключа для восстановления

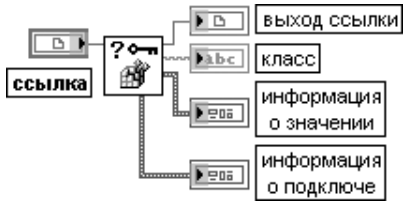
Вход **класс** (class) определяет класс (тип объекта) ключа.
Выход **выполненное действие** (action taken) показывает действие, выполненное ВП:

Выполненное действие	Описание
Unknown	Неизвестное
REG_CREATED_NEW_KEY	Создание нового ключа
REG_OPENED_EXISTING_KEY	Открытие существующего ключа

Query Registry Key Info



Запросить информацию о ключе реестра



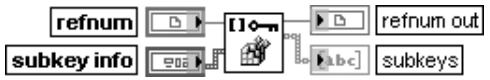
ВП извлекает информацию о ключах.
Выход **информация о значении** (value info) содержит набор параметров, описывающих набор значений и данных:

- **число значений** (numValues) – содержит число значений в ключе, определенном с помощью hKey;
- **максимальная длина имени значения** (maxValueNameLen) – представляет длину самого длинного имени в ключе, определенном с помощью hKey;
- **максимальная длина данных значения** (maxValueDataLen) – представляет длину самого длинного значения в ключе, определенном с помощью hKey.

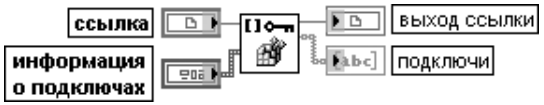
Выход **информация о подключе** (subkey info) содержит набор параметров, описывающих подключи:

- **число подключей** (numSubKeys) – представляет число подключей в ключе, определенном с помощью hKey;
- **максимальная длина имени подключа** (maxSubKeyLen) – представляет длину наиболее длинного имени подключа в ключе, определенном с помощью hKey;
- **максимальная длина имени класса** (maxSubKeyClassLen) – представляет длину наиболее длинного имени класса ключа, определенного с помощью hKey

Enum Registry Keys



Перечислить ключи реестра



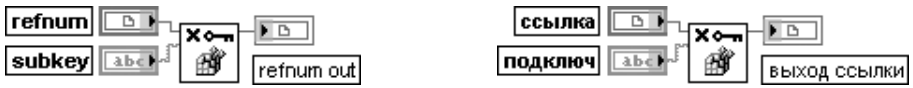
ВП отображает подключи определенного ключа или подключа. Этот ВП необходимо использовать вместе с ВП **Запросить информацию о ключе реестра** (Query Registry Key Info)

Close Registry Key

Закрыть ключ реестра

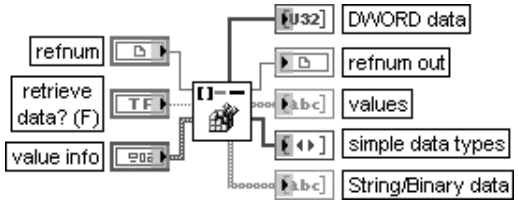
ВП закрывает ключ в реестра Windows

Delete Registry Key / Удалить ключ реестра

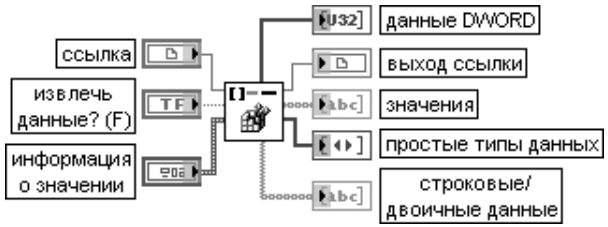


ВП удаляет определенный ключ или подключ

Enum Registry Values Simple

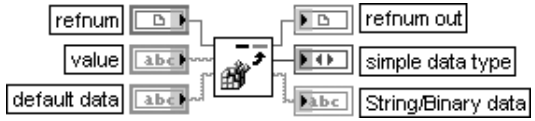


Перечислить значения реестра

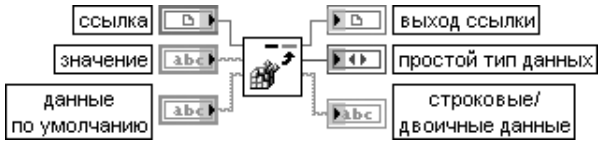


ВП отображает значения определенного ключа или подключа.
Если на входе **извлечь данные?** (retrieve data?) установлено состояние ИСТИНА, то ВП возвращает **32-разрядные целые числа** (DWORD data) и **строковые/двоичные данные** (String/Binary data) в дополнение к **значениям** (values) и **простым типам данных** (simple data types). Этот ВП необходимо использовать вместе с ВП **Запросить информацию о ключе реестра** (Query Registry Key Info)

Read Registry Value Simple

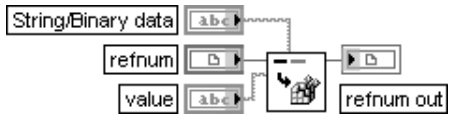


Читать значение реестра

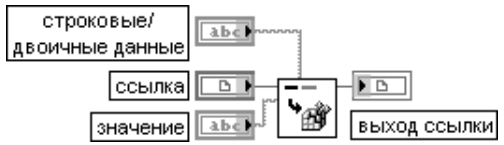


ВП читает данные и упрощенный тип данных из значения реестра.
Если ключ или значение не существует или возникает неисправимая ошибка, ВП возвращает данные по умолчанию. Этот полиморфный ВП может использоваться для записи строковых или числовых данных. Тип данных, подключенных ко входу **данные по умолчанию** (default data), определяет реализацию используемого полиморфного ВП

Write Registry Value Simple



Записать значение реестра



ВП записывает данные в **значение** (value) ключа реестра, определенного с помощью **ссылки** (refnum). Этот полиморфный ВП может использоваться для записи строковых или числовых данных. Тип данных, подключенных ко входу **строковые/двоичные данные** (String/Binary data), определяет реализацию используемого полиморфного ВП

Delete Registry Value / Удалить значение реестра

ВП удаляет значение реестра

На рис. 7.13 в качестве примера приведена блок-диаграмма ВП управления параметрами Web-камеры с помощью ВП доступа к реестру.

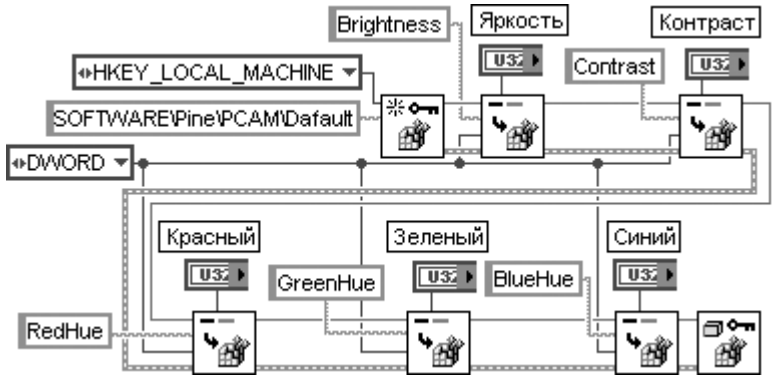


Рис. 7.13. Блок-диаграмма ВП управления параметрами Web-камеры

7.5. ВП управления устройствами ввода и портами ввода/вывода

ВП управления устройствами ввода (рис. 7.14) используются для получения информации о джойстике, клавиатуре и мыши.

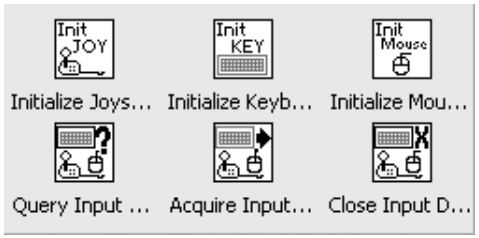
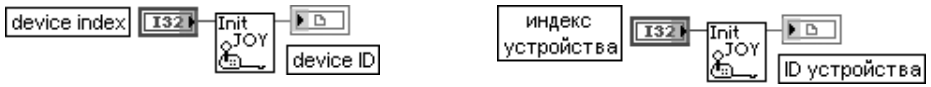


Рис. 7.14. Вид подпалитры ВП управления устройствами ввода

Initialize Joystick — **Инициализировать джойстик**



ВП открывает ссылку и инициализирует джойстик по заданному **индексу устройства** (device index). По умолчанию на этом входе установлено значение 0, что соответствует одному джойстику.
Выход **ID устройства** (device ID) содержит ссылку к открытому джойстику

Initialize Keyboard — **Инициализировать клавиатуру**

ВП открывает ссылку и инициализирует клавиатуру, подключенную к компьютеру

Initialize Mouse — **Инициализировать мышь**

ВП открывает ссылку и инициализирует мышь, подключенную к компьютеру

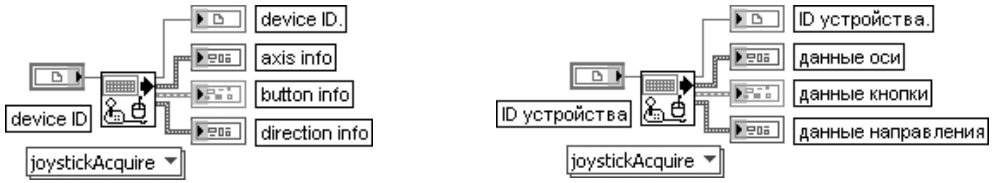
Query Input Devices — **Запросить устройства ввода**

ВП получает информацию об устройствах, подключенных к компьютеру.
Кластер **данные джойстика** (joystick info) содержит информацию о джойстике, подключенном к компьютеру:
- **общее число осей** (axes total);
- **общее число кнопок** (buttons total);
- **общее число точек обзора** (pov total);
- **имя устройства** (device name).

Кластер **данные клавиатуры** (key info) содержит информацию о клавиатуре, подключенной к компьютеру:
- **общее число клавишей** (buttons total);
- **наименование устройства** (device name).

Кластер **данные мыши** (mouse info) содержит информацию о мыши, подключенной к компьютеру:
- **общее число осей** (axes total);
- **общее число кнопок** (buttons total);
- **имя устройства** (device name)

Acquire Input Data — **Получить данные ввода**



ВП возвращает данные об устройстве, подключенном к компьютеру. Тип данных, подключенных к входу **ID устройства** (device ID), определяет вариант реализации полиморфного ВП

Close Input Device — **Закрыть устройство ввода**

ВП закрывает устройство ввода, определенное ссылкой на входе **ID устройства** (device ID)

На рис. 7.15 в качестве примера использования ВП управления устройствами ввода приведена блок-диаграмма ВП Basic Input Demo из библиотеки примеров NI Example Finder. ВП позволяет определять координаты джойстика и координаты мыши на экране, а также обнаруживать нажатие клавиши мыши или клавиатуры и определять тип нажатой клавиши клавиатуры.

Два следующих ВП используются для считывания или записи данных в порты, заданные 16-битовым адресом.

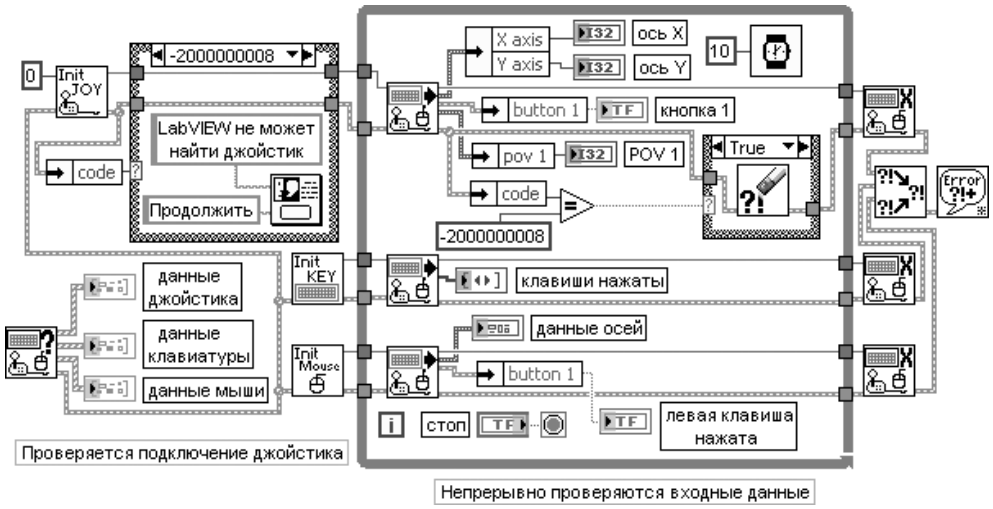
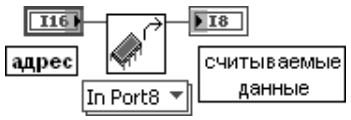


Рис. 7.15. Блок-диаграмма ВП Basic Input Demo

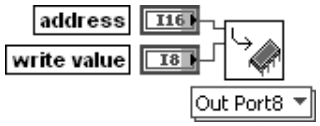
In Port

Порт ввода



ВП считывает целое число со знаком из порта, заданного 16-битовым **адресом**. Вариант реализации этого полиморфного ВП, определяющий разрядность считываемого числа, выбирается вручную

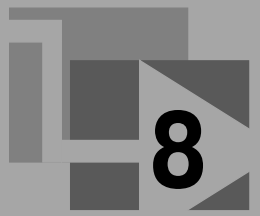
Out Port



Порт вывода

ВП записывает целое число со знаком в порт, заданный 16-битовым **адресом**. Вариант реализации этого полиморфного ВП, определяющий разрядность записываемого числа, выбирается вручную

Функции поддержки ввода/вывода данных и стандартных интерфейсов



Поддержка работы плат ввода/вывода данных и стандартных интерфейсов в LabVIEW 8.20 обеспечивается функциями, входящими в категории **Контроль ввода/вывода** (Measurement I/O), **Связь с приборами** (Instrument I/O) и **Зрение и движение** (Vision&Motion). В данном справочном пособии более подробно рассмотрены функции подпалитры **Сбор данных DAQmx**, входящей в категорию **Контроль ввода/вывода**, и функции из подпалитр **Канал общего пользования (КОП)** (General Purpose Interface Bus – GPIB) и **Последовательный канал** (Serial), входящие в состав категории **Связь с приборами** (рис. 8.1).

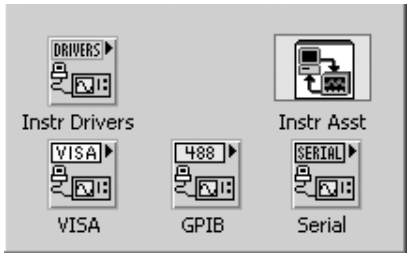


Рис. 8.1. Вид категории функций Связь с приборами

Начальным этапом работы с встроенными платами и стандартными интерфейсами ввода/вывода данных после установки плат и драйверов являются их конфигурирование и тестирование с помощью программы **Проводник измерений и автоматизации** (Measurement and Automation Explorer (MAX)). MAX является программным интерфейсом Windows, обеспечивающим доступ ко всем платам NI. MAX устанавливается по умолчанию во время установки LabVIEW. При запуске MAX открывается диалоговое окно (рис. 8.2), которое позволяет выбирать и устанавливать различные режимы функционирования плат и стандартных интерфейсов.

В левом окне **Конфигурация** (Configuration) показываются разделы конфигурации локальной и удаленных систем. Перечень разделов включает **Окружение данных** (Data Neighborhood), **Устройства и интерфейсы** (Devices and Interfaces), **Шкалы** (Scales), **Программное обеспечение** (Software) и **Драйверы IVI** (IVI Drivers).

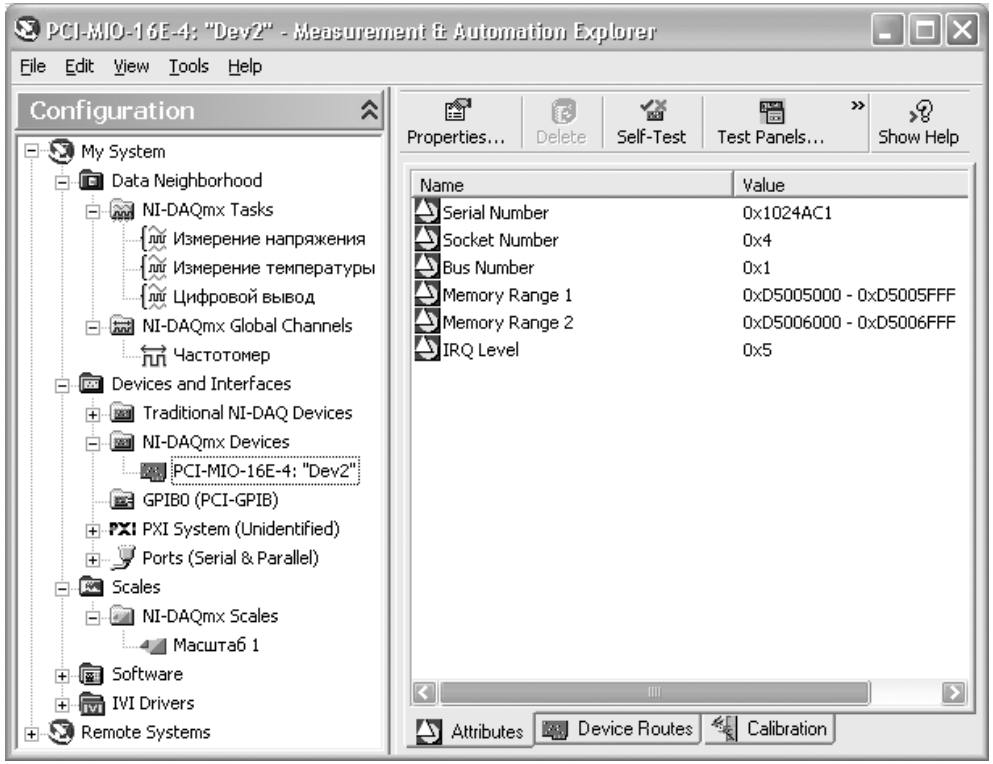


Рис. 8.2. Вид диалогового окна программы MAX

В разделе **Окружение данных** отображаются все настроенные **задачи NI-DAQmx** и **глобальные каналы NI-DAQmx**. **Задача** – это новая важная концепция NI-DAQmx, которая представляет совокупность одного или нескольких виртуальных каналов с синхронизацией, запуском и другими свойствами. Задача описывает измерение или генерацию сигнала, которые необходимо выполнить. Входящие в состав задачи виртуальные каналы включают в себя совокупность настроек физического канала DAQ, тип измерения и информацию о нормировке значений. В NI-DAQmx можно конфигурировать виртуальные каналы как часть задачи или отдельно от задачи. Виртуальные каналы, созданные внутри задачи, являются локальными. Виртуальные каналы, созданные вне задачи, являются глобальными и могут использоваться отдельно. Глобальные каналы могут быть созданы в MAX или в LabVIEW. Задачи также могут быть созданы как в MAX, так и в LabVIEW.

Создание новой задачи в программе MAX производится с помощью кнопки **Создать новую задачу NI-DAQmx** (Create New NI-DAQmx Task) в окне **Задачи NI-DAQmx** (NI-DAQmx Tasks) (рис. 8.3), которое, в свою очередь, выводится при выборе одноименного пункта в окне конфигурации MAX. Выбор конкретной задачи в этом же окне позволяет открыть в правой части окно настройки параметров задачи.

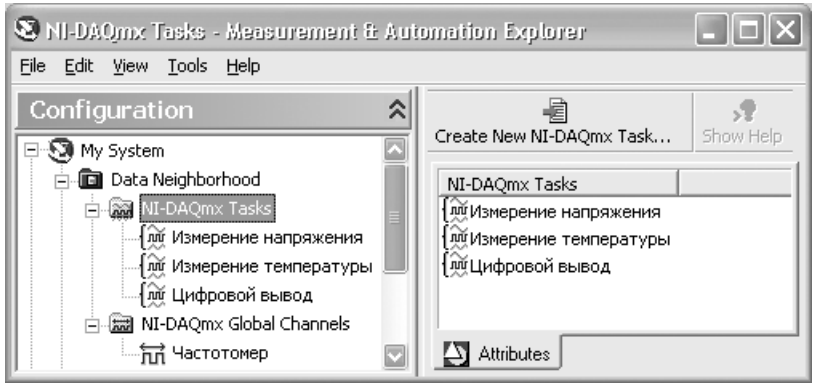


*Рис. 8.3. Вид диалогового окна MAX при выборе пункта **Задачи NI-DAQmx***

Раздел **Устройства и интерфейсы** окна **Конфигурация** (рис. 8.2) позволяет просмотреть параметры установленных в системе устройств и выполнить ряд действий по их тестированию и калибровке.

Раздел **Шкалы** позволяет создать, просмотреть и при необходимости настроить шкалу, определяющую характер преобразования измеряемой величины.

Раздел **Программное обеспечение** позволяет просмотреть описание, расположение и номер версии приложений NI.

Программа MAX позволяет также произвести тестирование и настройку аппаратных средств GPIB, последовательного и параллельного интерфейса.

8.1. Функции сбора данных DAQmx

ВП и функции сбора данных NI-DAQmx (рис. 8.4) относятся к следующему поколению драйверов NI-DAQ.

Выше было отмечено, что основным элементом NI-DAQmx является задача. Также было отмечено, что задача может быть создана как в программе MAX, так и в LabVIEW. В LabVIEW задача может быть создана несколькими способами:

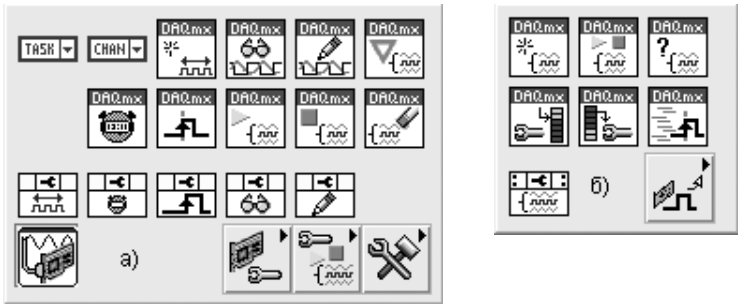


Рис. 8.4. Вид основной (а) и одной из дополнительных подпалитр (б) функций сбора данных DAQmx

1) путем размещения элемента управления **Имя задачи DAQmx** (DAQmx Task Name) на лицевой панели или константы с таким же именем (рис. 8.4а) на блок-диаграмме и выбора пункта **Новая задача** (New Task) из контекстного меню элемента или константы. При этом LabVIEW загружает данную задачу в память только один раз, даже если константа или элемент управления находятся в структуре цикла;
2) путем размещения Экспресс-ВП **Помощник DAQ** (DAQ Assistant) на блок-диаграмме. Задача, созданная с помощью Экспресс-ВП, является локальной по отношению к приложению и не может быть сохранена в MAX для использования в других приложениях. Для использования задачи в других приложениях или для генерации кода Экспресс-ВП Помощник DAQ должен быть преобразован в константу **Имя задачи DAQmx**. Преобразование осуществляется с помощью пункта **Преобразовать в константу имени задачи** (Convert to Task Name Constant) контекстного меню иконки Экспресс-ВП;
3) путем выбора строки **ВП из шаблона** (VI from Template) в окне первоначального запуска и последующего выбора ВП **Сбор данных с помощью NI-DAQmx** (Data Acquisition with NI-DAQmx).

При создании задачи как в MAX, так и в LabVIEW открывается диалоговое окно **Создать новую** (Create New). На рис. 8.5 показан вид диалогового окна при его открытии из MAX. С помощью набора кнопок в правой части окна производится выбор класса измерения или генерации сигнала. Набор основных классов задач включает **Аналоговый ввод** (Analog Input), **Аналоговый вывод** (Analog Output), **Подсчет на входе** (Counter Input), **Подсчет на выходе** (Counter Output) и **Цифровой ввод/вывод** (Digital I/O). Более подробная иерархия видов задач измерения и генерации сигналов приведена в таблице.

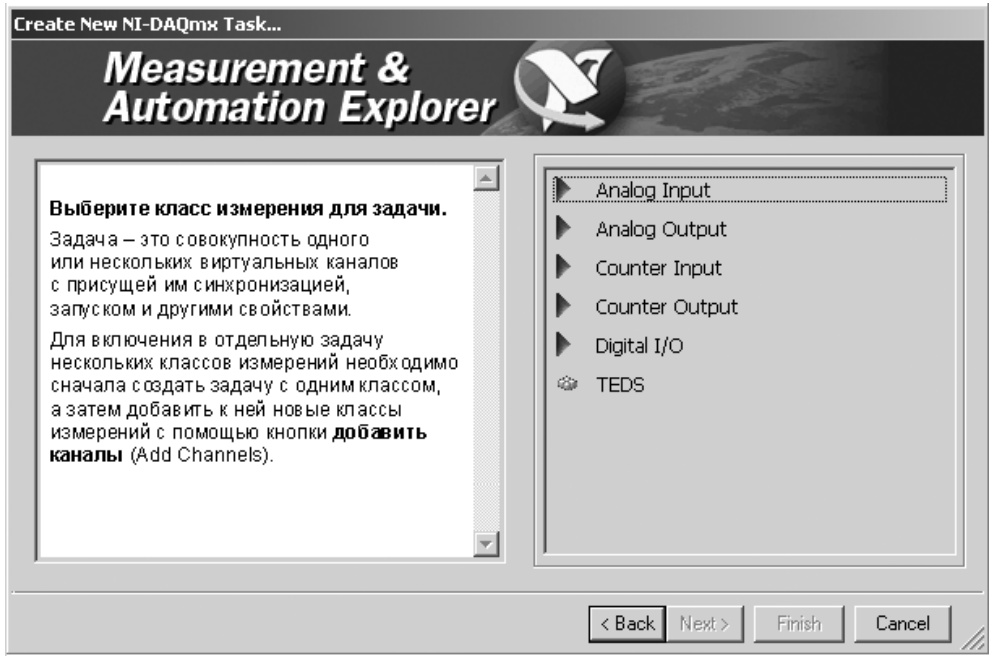


Рис. 8.5. Вид диалогового окна создания новой задачи

Класс измерений	Вид измерений
Аналоговый ввод (Analog Input)	Напряжение (Voltage) Температура (Temperature) • терморезистор с подачей тока (I_{ex} Termistor) • терморезистор с подачей напряжения (V_{ex} Termistor) • термометр сопротивления (RTD) • термопара (Thermocouple) Деформация (Strain) Ток (Current) Сопротивление (Resistance) Частота (Frequency) Положение (Position) • датчик перемещения на основе линейного дифференциального трансформатора (LVDT) • датчик углового перемещения на основе вращающегося дифференциального трансформатора (RVDT) Ускорение (Acceleration) Специальное напряжение с возбуждением (Custom Voltage with Excitation) Звуковое давление (Sound Pressure)
Аналоговый вывод (Analog Output)	Напряжение (Voltage) Ток (Current)
Подсчет на входе (Counter Input)	Подсчет перепадов (Edge Count) Частота (Frequency) Период (Period) Ширина импульса (Pulse Width) Полупериод (Semi Period) Интервал между двумя перепадами (Two Edge Separation)
Подсчет на выходе (Counter Output)	Вывод импульса (Pulse Output)
Цифровой ввод/вывод (Digital I/O)	Ввод линии (Line Input) Ввод порта (Port Input) Вывод линии (Line Output) Вывод порта (Port Output)
Датчики с электронными таблицами (TEDS)	Те же виды измерений, что и в п. Аналоговый ввод

После выбора задачи производится выбор физического канала (каналов) из списка установленных (рис. 8.6).

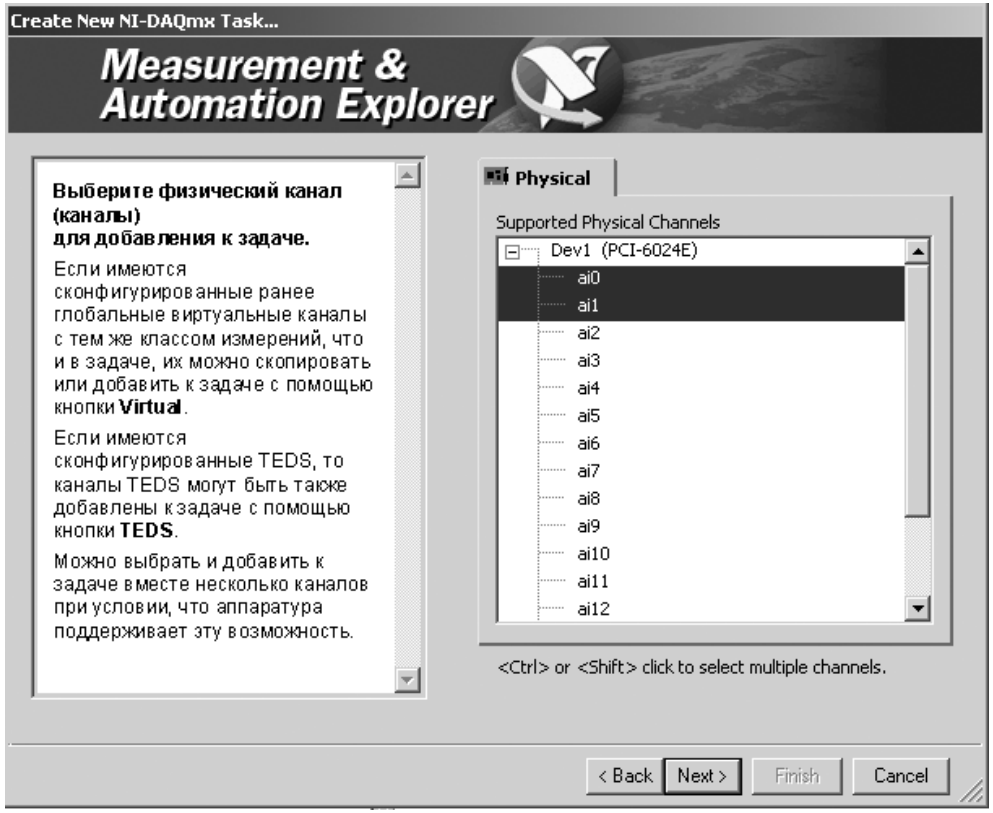


Рис. 8.6. Вид окна выбора физического канала

После выбора физического канала и ввода имени задачи в следующем диалоговом окне производится настройка параметров физического канала. Настройка в зависимости от вида измерения включает установку параметров в следующих разделах окна:

1) раздел **Установки** (Settings). На рис. 8.7 показан вид данного раздела для задачи измерения температуры с помощью **термометра сопротивления** (Resistance temperature detector – RTD);
2) раздел **Синхронизация задачи** (Task Timing). На рис. 8.8 показан вид раздела с выбранными опциями **Получение N выборок** (Acquire N Samples), **Внешний** (External) в окне **Тип тактирования** (Clock Type), **Нарастающий** (Rising) в окне **Активный фронт** (Active Edge) и PFI1 в окне **Источник тактирования** (Clock Source);
3) раздел **Запуск задачи** (Task Triggering). Раздел содержит две группы параметров: **Начало** (Start) и **Реперная точка** (Reference). Параметры группы **Начало** определяют сигнал, запускающий сбор данных, а параметры группы **Реперная точка** – сигнал, устанавливающий реперную точку в наборе входных значений. На рис. 8.9 показан возможный вид раздела с парамет-

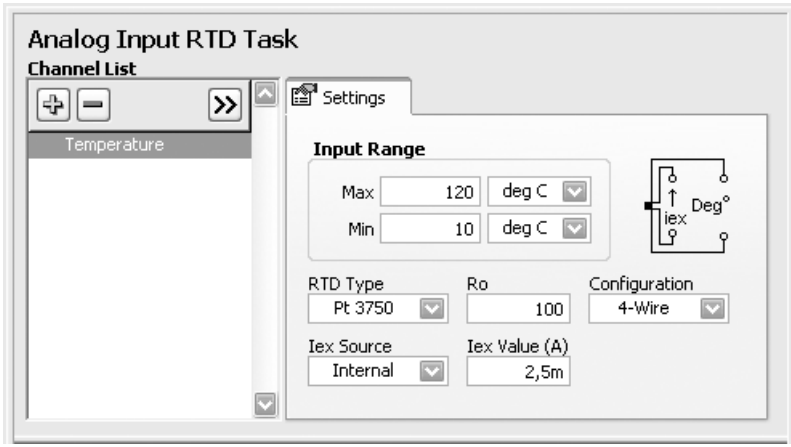


*Рис. 8.7. Вид раздела **Установки** диалогового окна настройки параметров физического канала*

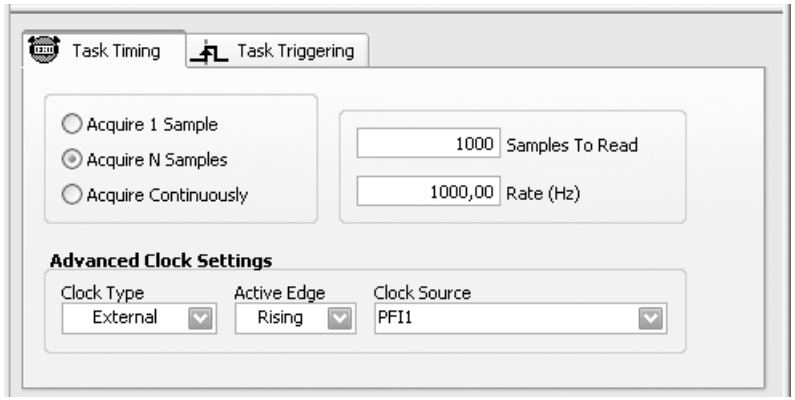


*Рис. 8.8. Вид раздела **Синхронизация задачи***

рами групп **Начало** и **Реперная точка**. В состав первой группы входят параметры **Цифровой фронт** (Digital Edge) в окне **Тип запуска** (Trigger Type), **Нарастающий** (Rising) в окне **Фронт** (Edge) и **Источник** (Source). Набор параметров второй группы включает **Аналоговый фронт** в окне **Тип запуска**, **Выборки перед запуском** (Pretrigger Samples), **Спадающий** (Falling) в окне **Наклон** (Slope) и **Уровень** (Level).

После завершения конфигурирования задачи она может быть протестирована с помощью кнопки **Тест** (Test). Корректно сконфигурированная задача, открытая из MAX, может быть сохранена с помощью кнопки **Сохранить задачу** (Save Task).

При установке на панели блок-диаграммы Экспресс-ВП **Помощник DAQ** (DAQ Assistant) выводится такая же последовательность диалоговых окон (рис. 8.5–8.9), только вместо кнопки **Сохранить задачу** после завершения ее конфигуриро-

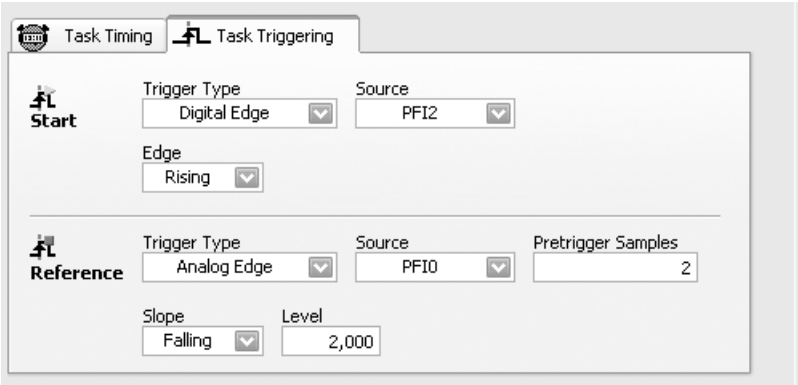


Рис. 8.9. Вид раздела ***Запуск задачи***

вания нажимается кнопка **OK**. После **проверки задачи** (Verifying Task) Помощник производит формирование ВП (Building Assistant VI). Сформированный таким образом Экспресс-ВП **Помощник DAQ** может быть включен в блок-диаграмму виртуального прибора или на его основе может быть сформирован подприбор (subVI), обеспечивающий аналогичную функциональность. Переход от Экспресс-ВП к подприбору осуществляется с помощью выбора опции **Открыть лицевую панель** (Open Front Panel) в контекстном меню иконки Экспресс-ВП. Пример блок-диаграммы такого подприбора, сформированного из Экспресс-ВП измерения переменного напряжения, представлен на рис. 8.10.

Из блок-диаграммы видно, что каждая из функций создания и конфигурирования задачи, заключенная в структуру **Вариант** и в структуру **Цикл по условию**, выполняются только один раз при первом запуске ВП. В последующих запусках будет выполняться только ВП **Читать DAQmx** (Read DAQmx). Такой характер выполнения функций обеспечивается с помощью подключения функции **Первый вызов?** (First Call?) к терминалу селектора структуры **Вариант** и подключения константы ИСТИНА к терминалу условия выхода из цикла структуры **Цикл по условию**, установленному в состояние **Остановить если истина** (Stop If True).

Полученная блок-диаграмма обладает определенной избыточностью. Более компактный код можно создать, преобразовав Экспресс-ВП Помощник DAQ в константу **Имя задачи DAQmx**. В контекстном меню данной константы предусмотрены три варианта формирования кода при выборе опции **Генерировать код** (Generate Code) из контекстного меню константы (рис. 8.11):

а) **Пример** (Example) – генерирует весь код, необходимый для выполнения задачи или канала, такой как ВП чтения или записи выборок, ВП для запуска и остановки задачи, циклы и графики. Эта опция выбирается, если задача или канал являются специфическими для системы и не используются в других системах;

б) **Структура** (Configuration) – генерирует код, связанный со структурой. Константа/элемент управления ввода/вывода заменяются подприбором, который содержит ВП и узлы свойств, используемые для создания и конфигурирования каналов, конфигурирования синхронизации и запуска, используемых в задаче или канале. Эта опция выбирается, если необходима портативная структура, которая может быть перемещена в другую систему;

в) **Структура** и **Пример** (Configuration and Example) – генерирует как код структуры, так и код примера для задачи или канала в одном шаге.

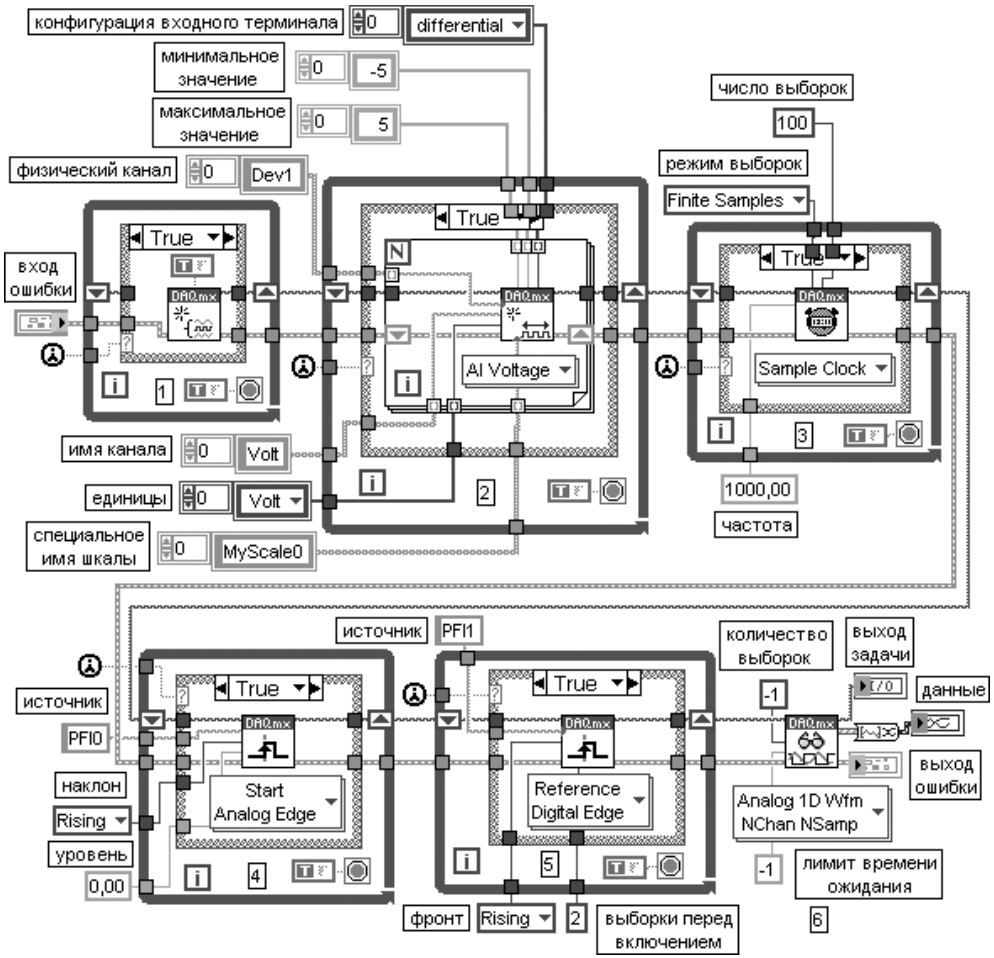


Рис. 8.10. Блок-диаграмма ВП измерения переменного напряжения

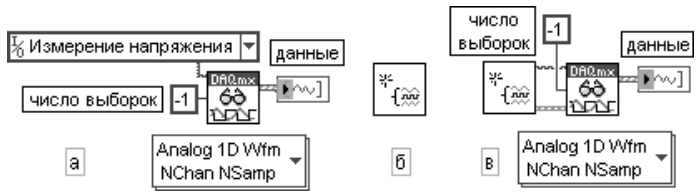


Рис. 8.11. Варианты кода ВП измерения напряжения

На рис. 8.12 приведена блок-диаграмма подприбора, изображенного на рис. 8.11 б.

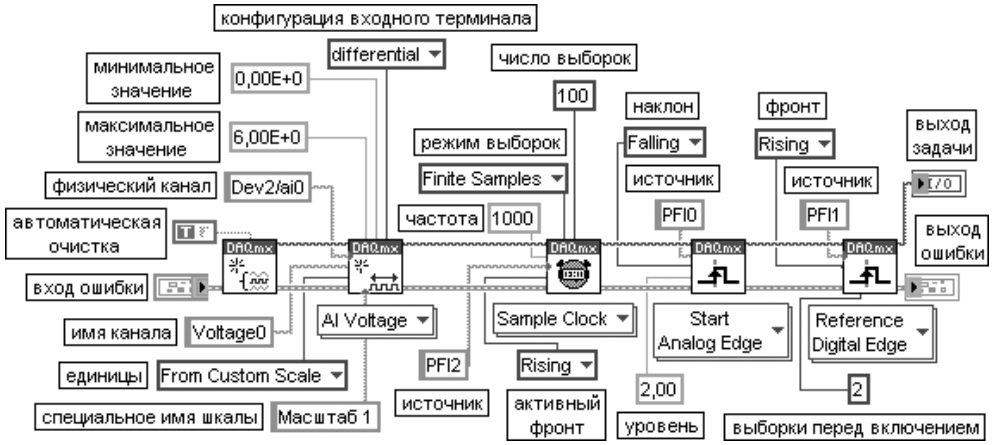


Рис. 8.12. Блок-диаграмма подприбора, изображенного на рис. 8.11б

Аналогичную структуру в виде цепочки ВП из палитры DAQmx имеют ВП из набора примеров NI Example Finder LabVIEW. На рис. 8.13 приведена блок-диаграмма модернизированного ВП регистрации и отображения напряжения с внутренней синхронизацией, цифровым фронтом запуска начала сбора данных и реперной точки Acq&Graph Voltage-Int Clk-Dig Start&Ref из набора примеров NI Example Finder (для однообразия с приведенными выше ВП цифровой фронт начала заменен на аналоговый).

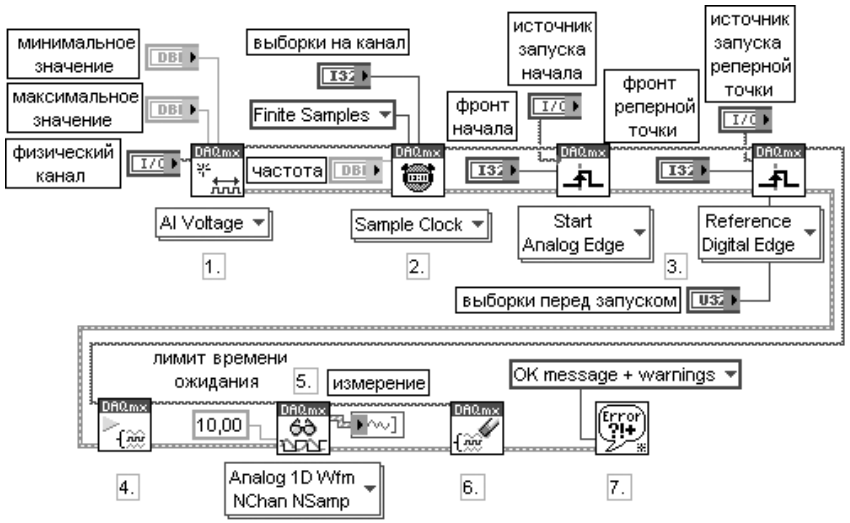


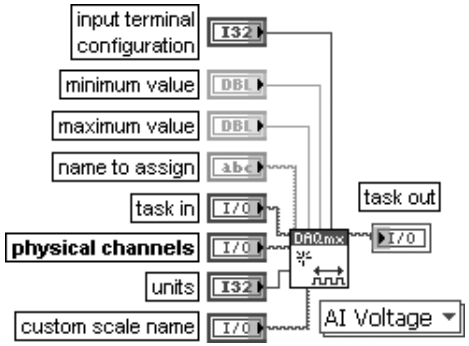
Рис. 8.13. Блок-диаграмма модернизированного ВП Acq&Graph Voltage-Int Clk-Dig Start&Ref

В процессе выполнения этого ВП реализуются следующие шаги:

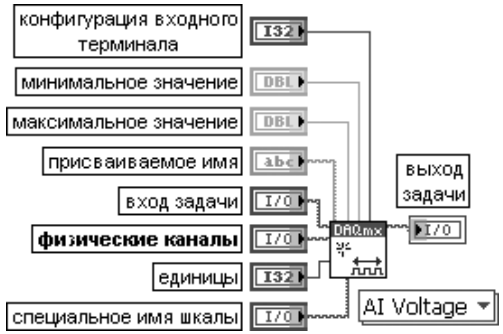
1) создается канал аналогового ввода напряжения;
2) определяются параметры внутреннего источника тактовых импульсов. Дополнительно определяется режим конечного числа выборок;
3) определяются параметры для ВП **Запуск DAQmx** в вариантах **Аналоговый фронт запуска начала** и **Цифровой фронт запуска реперной точки**;
4) вызывается ВП **Начать задачу** для начала сбора данных;
5) используется ВП **Читать DAQmx** для измерения совокупности выборок из N каналов платы сбора данных. Лимит времени ожидания устанавливается так, что в случае его превышения будет возвращаться ошибка;
6) вызывается ВП **Очистить задачу**;
7) для отображения возможной ошибки используется диалоговое окно.

Блок-диаграммы, приведенные на рис. 8.10–8.13, показывают программный способ создания задачи с помощью ВП **Создать виртуальный канал DAQmx** (DAQmx Create Virtual Channel) или ВП **Создать задачу DAQmx** (DAQmx Create Task). Ниже в таблице более подробно рассмотрены функции из основной палитры **Сбор данных DAQmx** (DAQmx – Data Acquisition) и подпалитры **Дополнительные опции задачи DAQmx** (DAQmx Advanced Task Options) (рис. 8.4б).

DAQmx Create Virtual Channel



Создать виртуальный канал DAQmx



ВП создает **виртуальный канал** (virtual channel) или набор виртуальных каналов и добавляет их к **задаче** (task). При конфигурации этого полиморфного ВП с помощью селектора выбирается тип канала ввода или вывода: аналоговый, цифровой или счетный. В зависимости от вида измерения или генерации сигнала может выбираться измерение температуры, генерация напряжения или подсчет событий. В некоторых случаях выбирается также и тип датчика, такой, например, как термопара или терморезистор при измерениях температуры.

Если при использовании ВП **Создать виртуальный канал DAQmx** не определена задача, к которой добавляются созданные каналы, то NI-DAQmx создает новую задачу и выделяет для нее ресурсы. LabVIEW не освобождает эти ресурсы автоматически до завершения приложения. При использовании ВП **Создать виртуальный канал DAQmx** в цикле без определения **входа задачи** (task in) NI-DAQmx создает новую задачу при каждой итерации цикла и не закрывает их до завершения приложения. Для предотвращения увеличения объема занятой памяти при завершении задачи необходимо использовать в цикле ВП **Очистить задачу DAQmx** (DAQmx Clear Task).

Блок-диаграмма ВП **Создать виртуальный канал DAQmx** для варианта измерения напряжения приведена на рис. 8.14. Как видно из блок-диаграммы, для выполнения конфигурирования канала используется узел свойства **Канал DAQmx** (DAQmx Channel). Перечень свойств изменяется в зависимости от вида измерения. Свойства **Канал DAQmx** являются подклассом свойств **DAQmx**.

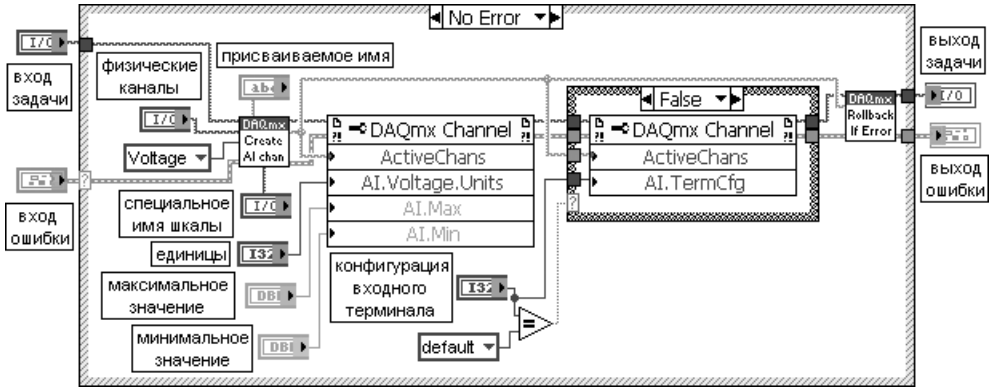
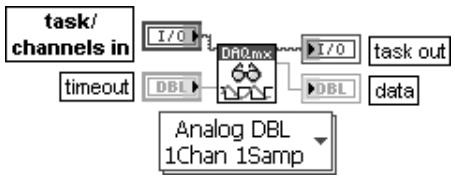
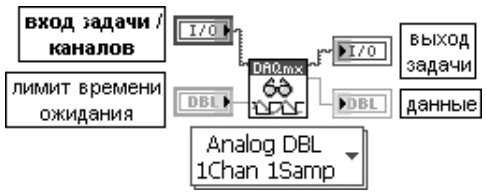


*Рис. 8.14. Блок-диаграмма ВП **Создать виртуальный канал DAQmx***

DAQmx Read



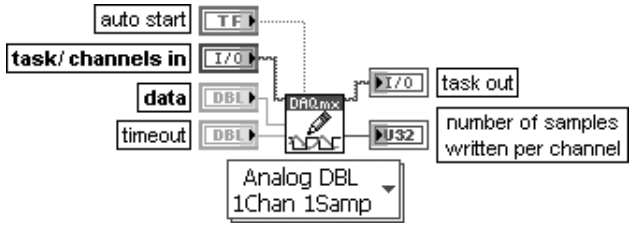
Читать DAQmx



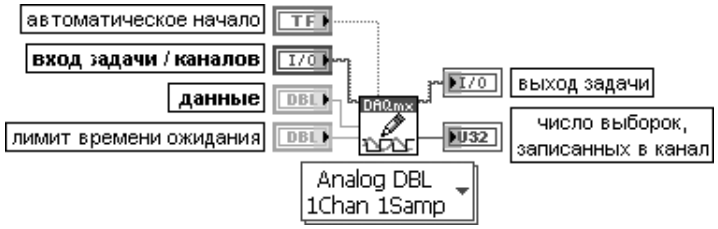
ВП считывает выборки из задачи или каналов, определенных пользователем. Реализация этого полиморфного ВП предусматривает выбор формата возвращаемых выборок, выбор считывания единичной выборки или совокупности выборок, а также выбор считывания из одного или нескольких каналов.

Узел свойства **Читать DAQmx** (DAQmx Read) включает дополнительные опции конфигурации операций чтения

DAQmx Write



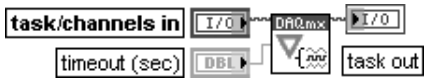
Записать в DAQmx



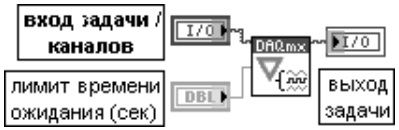
ВП записывает выборки в задачи или каналы, определенные пользователем. Реализация этого полиморфного ВП предусматривает выбор формата записываемых выборок, выбор записи единичной выборки или совокупности выборок, а также выбор записи в один или несколько каналов.

Если задача использует синхронизацию по запросу, по умолчанию ВП **Синхронизация DAQmx** (DAQmx Timing) не используется, этот ВП возвращает только после того, как устройство сгенерирует все выборки. Если задача использует типы синхронизации, отличающиеся от синхронизации по запросу, этот ВП возвращает немедленно и не ожидает завершения генерации всех выборок. Используемое приложение должно определять выполнение задачи, чтобы гарантировать, что устройство сгенерировало все выборки

DAQmx Wait Until Done



Ожидать выполнения DAQmx

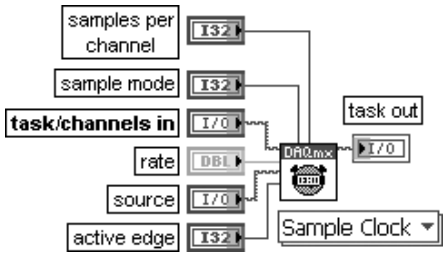


ВП ожидает завершения измерения или генерации. Данный ВП необходимо использовать для того, чтобы гарантировать завершение заданной операции перед остановкой задачи.

Вход задачи /каналов (task/channels in) содержит имя задачи или список виртуальных каналов, к которым применяется данная операция. При задании списка каналов NI-DAQmx создает задачу автоматически.

Вход **лимит времени ожидания (сек)** (timeout (sec)) определяет максимальную длительность в секундах ожидания завершения измерения или генерации. Этот ВП возвращает ошибку, если заданное время истекло. По умолчанию значение равно 10. При установке на входе значения –1 ВП ожидает неопределенно долго. Если на входе установлено значение 0, то ВП проверяет один раз и возвращает ошибку, если задача не выполнена

DAQmx Timing



Синхронизация DAQmx

ВП конфигурирует число собираемых или генерируемых выборок и создает буфер, если это необходимо. Реализация этого полиморфного ВП соответствует типу синхронизации, используемой в задаче.
Данный ВП реализован на основе узла свойств **Синхронизация DAQmx** (DAQmx Timing), также являющегося подклассом узла свойств **DAQmx**

DAQmx Trigger / Запуск DAQmx

ВП конфигурирует запуск задачи. Реализация этого полиморфного ВП соответствует наличию и типу запуска задачи.
Узел свойства **Запуск DAQmx** (DAQmx Trigger) содержит все опции запуска, включенные в этот ВП, а также дополнительные опции запуска

DAQmx Start Task / Начать задачу DAQmx

ВП переводит задачу в состояние выполнения для начала измерения или генерации. Использование этого ВП необходимо для некоторых приложений и является дополнительным для других.
Если этот ВП не используется, то измерительная задача начинается автоматически при выполнении ВП **Читать DAQmx**. Вход **автоматическое начало** ВП **Записать в DAQmx** определяет автоматическое начало выполнения этого ВП.
Если ВП **Начать задачу DAQmx** и **Остановить задачу DAQmx** не используются при многократном применении ВП **Читать DAQmx** и **Записать в DAQmx**, то задача начинается и останавливается многократно, что приводит к ухудшению работы приложения

DAQmx Stop Task / Остановить задачу DAQmx

ВП останавливает задачу и возвращает ее к состоянию, в котором задача находилась перед использованием ВП **Начать задачу DAQmx** или перед использованием ВП **Записать в DAQmx** с установленным в состояние ИСТИНА входом **автоматическое начало**

DAQmx Clear Task / Очистить задачу DAQmx

ВП очищает задачу. Перед очисткой этот ВП останавливает задачу, если это необходимо, и освобождает все ресурсы, отведенные задаче. После очистки задача не может использоваться, несмотря на ее воссоздание

В следующей таблице рассмотрены функции из подпалитры **Дополнительные опции задачи DAQmx** (DAQmx Advanced Task Options).

DAQmx Create Task / Создать задачу DAQmx

ВП создает **задачу** (task) и добавляет **виртуальные каналы** (virtual channels) к этой задаче, если они определены на входе **глобальные виртуальные каналы** (global virtual channels). Если определена **задача для копирования** (task to copy), то этот ВП добавляет любые виртуальные каналы к копии этой задачи, а не к самой задаче. В противном случае ВП создает новую задачу и добавляет каналы к этой задаче.
Если этот ВП используется в структуре цикла, то NI-DAQmx создает новую задачу при каждой итерации цикла. Для использования ВП **Создать задачу DAQmx** в цикле без выделения дополнительной памяти при каждой итерации необходимо использовать ВП **Очистить задачу DAQmx** (DAQmx Clear Task) в том же цикле после завершения задачи.
Вход **автоматическая очистка** (auto cleanup) ВП **Создать задачу DAQmx** определяет длительность пребывания этих задач в памяти. Если на входе **автоматическая очистка** установлено состояние ИСТИНА, то созданные задачи остаются в памяти до завершения выполнения приложения. При установке состояния ЛОЖЬ (по умолчанию) задачи остаются в памяти до выхода из LabVIEW, а не до завершения выполнения приложения. Такое поведение позволяет создавать задачу в одном приложении, а затем использовать ее в другом

DAQmx Control Task / Управление задачей DAQmx

ВП изменяет состояние задачи в соответствии с **действием** (action), заданным пользователем.
Вход **действие** (action) определяет вариант изменения состояние задачи:

Действие	Описание
Прервать (abort) (6)	Прерывает выполнение задачи. Прерывание задачи немедленно прекращает выполнение таких активных операций, как считывание или запись. Прерывание переводит задачу в нестабильное, но восстановимое состояние. Для восстановления задачи путем ее перезапуска следует использовать ВП **Начать задачу DAQmx** или **Остановить DAQmx** для сброса задачи без ее запуска

Действие	Описание
Совершить (commit) (3)	Программирует аппаратуру насколько возможно в соответствии с конфигурацией задачи
Резервировать (reserve) (4)	Резервирует ресурсы аппаратуры, необходимые для задачи. Никакие другие задачи не могут резервировать эти же ресурсы
Снять резервирование (unreserved) (5)	Освобождает все зарезервированные до этого ресурсы
Проверить (verify) (2)	Проверяет соответствие всех параметров задачи возможностям аппаратуры

Если **вход ошибки** (error in) показывает, что ошибка произошла перед выполнением данного ВП, то ВП будет выполняться нормально, если на входе **действие** установлено состояние **снять резервирование** (unreserved) или **прервать** (abort)

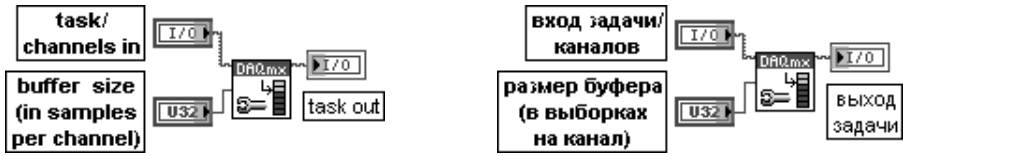
DAQmx Is Task Done

Задача DAQmx выполнена?

ВП запрашивает статус задачи и возвращает информацию о завершении ее выполнения. Этот ВП используется для обеспечения гарантии того, что заданная операция выполнена перед остановкой задачи

DAQmx Configure Input Buffer

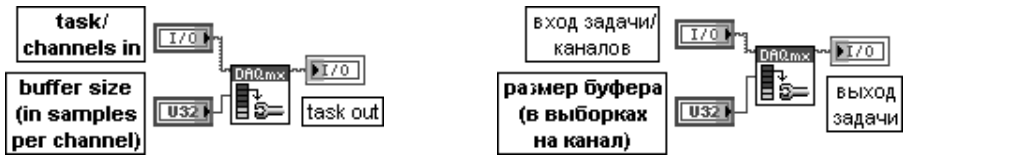
Конфигурировать входной буфер DAQmx



ВП аннулирует автоматическое назначение входного буфера, выполняемое NI-DAQmx. Вход **размер буфера (в выборках на канал)** (buffer size (in samples per channel)) определяет число выборок, которые буфер может содержать для каждого канала в задаче. При нулевом значении буфер не создается. Такое значение позволяет выполнить операцию с аппаратной синхронизацией без использования буфера

DAQmx Configure Output Buffer

Конфигурировать выходной буфер DAQmx



ВП аннулирует автоматическое назначение выходного буфера, выполняемое NI-DAQmx

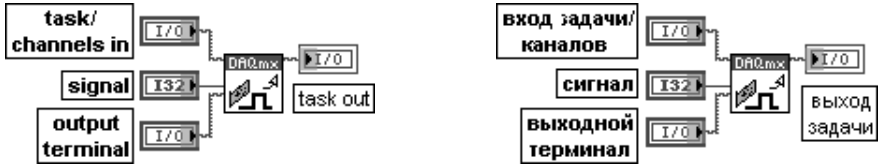
DAQmx Send Software Trigger

Послать программный запуск DAQmx

ВП генерирует программный запуск, определяемый пользователем. Пользователь должен сконфигурировать программный запуск с помощью узла свойства **Запуск DAQmx** перед использованием этого ВП

DAQmx Export Signal

Экспортировать сигнал DAQmx



ВП направляет управляющие сигналы к терминалу, заданному пользователем. Выходной терминал может находиться в устройстве, генерирующем управляющий сигнал, или в другом устройстве. Этот ВП может использоваться для распространения синхронизирующих и запускающих сигналов по множеству задач и устройств.
Вход **сигнал** (signal) передает имя экспортируемого запуска, синхронизации или события. Предусмотрены следующие варианты состояния данного входа:

20Mhz Timebase Clock	Выход генератора, который является встроенным источником основной синхронизации. Другие синхросигналы являются производными от этого
Наступление события завершения (Advance Complete Event)	Сигнал, генерируемый ключом после того, как он выполнит команду (команды) на входе списка сканирования и дождется истечения времени установления сигнала
Наступление запуска (Advance Trigger)	Запуск, который передвигает ключ к следующему входу списка сканирования
Синхронизация преобразования (AI Convert Clock)	Синхронизация, которая инициирует аналого-цифровое преобразование в **устройствах серии E** (E-Series device). Одно преобразование соответствует единичной выборке из одного канала
Событие завершения хранения (AI Hold Complete Event)	Сигнал, генерируемый устройством серии E, когда оно блокирует аналоговый вход данных (АЦП переходит в режим хранения) и предохраняет его от любого внешнего аппаратного переключения с целью удаления текущего сигнала и его замены следующим. Это событие не показывает завершения фактического аналого-цифрового преобразования
Событие выхода счетчика (Counter Output Event)	Сигнал, генерируемый счетчиком при достижении предельного состояния
Запуск реперной точки (Reference Trigger)	Запуск, который устанавливает реперную точку между выборками перед запуском и выборками после запуска
Синхронизация выборки (Sample Clock)	Синхросигналы, которые устройство использует для тактирования каждой выборки
Запуск начала (Start Trigger)	Запуск, который начинает измерение или генерацию

Ниже на рис. 8.15–8.17 приведены блок-диаграммы ВП из набора примеров NI Example Finder, реализующих основные типы задач измерения и генерации сигналов с помощью рассмотренных выше функций.

Так, в частности, на рис. 8.15 приведена блок-диаграмма ВП непрерывного вывода аналогового напряжения с внешним тактированием и запуском по цифровому фронту Cont Gen Voltage Wfm-Ext Clk-Dig Start из библиотеки **Генерировать напряжение** (Generate Voltage).

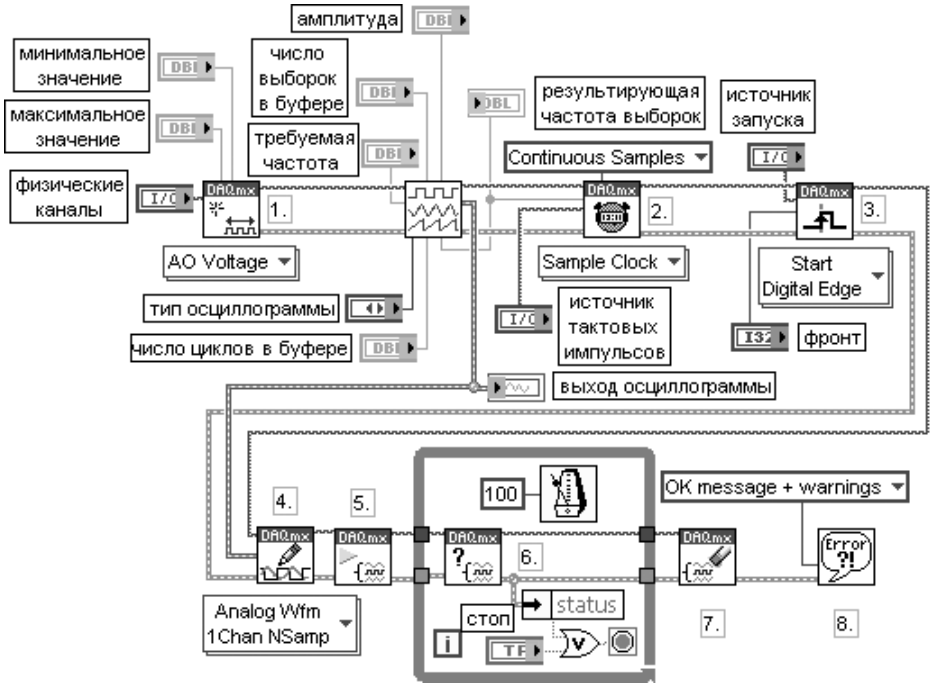


Рис. 8.15. Блок-диаграмма модернизированного ВП Cont Gen Voltage Wfm-Ext Clk-Dig Start

В процессе выполнения данного ВП реализуются следующие шаги:

1) создается канал аналогового вывода напряжения;
2) вызывается ВП **Синхронизация DAQmx**, который устанавливает частоту выборок, рассчитанную в подприборе **Генерация осциллограммы из буфера** (Waveform Buffer Generation). Дополнительно устанавливается режим непрерывной генерации;
3) определяются **источник** и **фронт** для ВП **Запуск DAQmx**;
4) производится запись осциллограммы в выходной буфер;
5) вызывается ВП **Начать задачу**;
6) циклически с интервалом 100 мс выполняется вывод сигнала, а также проверка ошибки с помощью ВП **Задача DAQmx выполнена?**;

7) после нажатия кнопки **стоп** выполнение цикла заканчивается и вызывается ВП **Очистить задачу**;
8) для отображения возможной ошибки используется диалоговое окно.

На рис. 8.16 приведена блок-диаграмма ВП измерения частоты цифрового сигнала Meas Dig Freq-Buffered-Cont-Large Range 2 Ctr из библиотеки **Измерить частоту цифрового сигнала** (Measure Digital Frequency) набора примеров NI Example Finder.

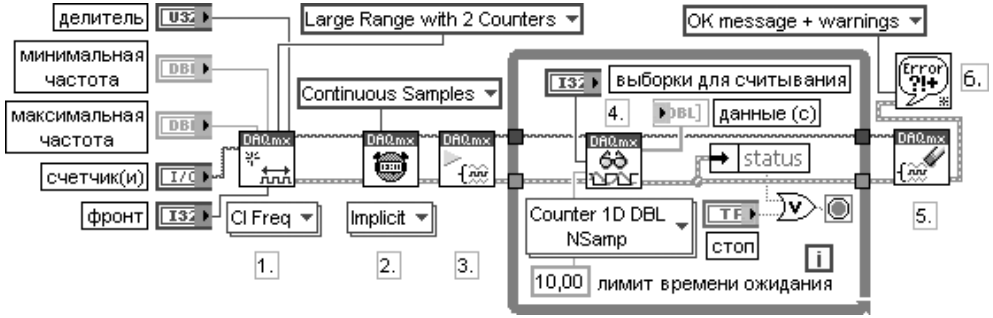


Рис. 8.16. Блок-диаграмма ВП Meas Dig Freq-Buffered-Cont-Large Range 2

В процессе выполнения данного ВП реализуются следующие шаги:

1) создается канал **Вход счетчика** для измерения частоты. Параметр **фронт** используется для выбора для измерения положительного или отрицательного фронта. Вход **делитель** определяет число периодов сигнала, используется для расчета частоты. Увеличение этого параметра повышает точность измерения, но вместе с тем увеличивает его длительность. Важно наиболее точно установить значения **максимальной** и **минимальной частоты**, так как это определяет выбор внутренней частоты синхронизации, минимизирующей ошибки. По умолчанию значения граничных частот определяют диапазон, который может быть измерен с использованием внутренней частоты 20 мГц;
2) вызывается ВП **Синхронизация DAQmx** для конфигурирования **режима выборок** и **числа выборок на канал**. При этом устанавливается **адаптивная** (implicit) синхронизация, поскольку частота синхронизации определяется самим измеряемым сигналом;
3) вызывается ВП **Начать задачу** для стробирования счетчика и начала измерения;
4) для непрерывных измерений счетчик будет непрерывно считывать все доступные данные до нажатия кнопки стоп на лицевой панели;
5) вызывается ВП **Очистить задачу**;
6) для отображения возможной ошибки используется диалоговое окно.

На рис. 8.17 приведена блок-диаграмма ВП **Считать цифровой канал** (Read Dig Chan) из библиотеки **Считать значения** (Read Values) набора примеров NI Example Finder.

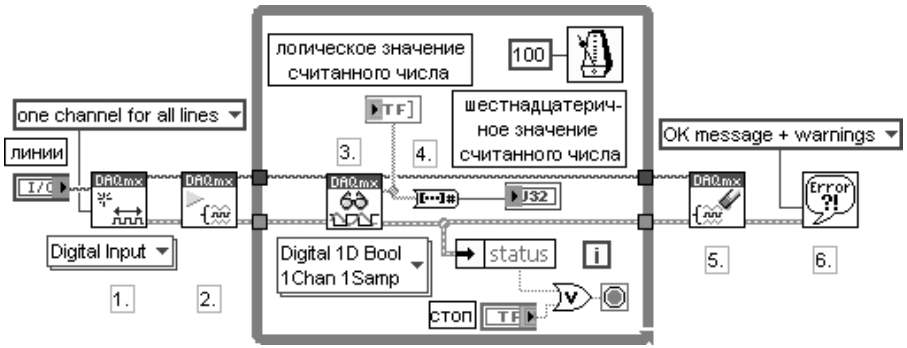


*Рис. 8.17. Блок-диаграмма ВП **Считать цифровой канал** (Read Dig Chan)*

В процессе выполнения данного ВП реализуются следующие шаги:

1) создается канал цифрового ввода. Один канал используется для всех линий. В качестве альтернативы можно использовать для каждой линии, выбрав соответствующую версию полиморфного ВП **Читать DAQmx**;
2) вызывается ВП **Начать задачу**;
3) производится чтение цифровых данных в цикле до нажатия пользователем кнопки **стоп** или прихода ошибки. ВП Читать DAQmx считывает единичную выборку цифровых данных по запросу, поэтому нет необходимости в тайм-ауте;
4) для просмотра считываемых данных в виде единственного числа используется ВП **Логический массив в число**;
5) вызывается ВП **Очистить задачу**;
6) для отображения возможной ошибки используется диалоговое окно.

В LabVIEW 8.20 появилась возможность копировать данные конфигурации из файла в MAX и обратно. Эту функцию выполняет рассмотренный ниже ВП **Копировать данные конфигурации MAX** (MAX Copy Configuration), размещенный в палитре функций **Контроль ввода/вывода** (Measurement I/O).

MAX Copy Configuration

Копировать данные конфигурации MAX

ВП копирует данные конфигурации. Этот ВП можно использовать для импорта данных конфигурации из файла в MAX, экспорта этих данных из MAX в файл или копирования данных конфигурации непосредственно из системы в систему.

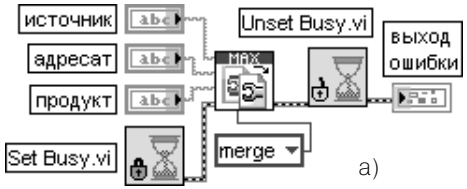
Вход **источник** (source) определяет источник копируемой конфигурации. На этом входе может быть задан файл конфигурации или конфигурация системы MAX. В первом случае на входе **источник** задается путь к файлу, во втором случае указывается «system://<host>», где <host> – системное имя компьютера или IP-адрес.

Вход **адресат** (destination) устанавливает получателя копируемых данных. Формат **адресата** должен быть таким же, что и формат **источника**.

Вход **продукт** (product) определяет название продукта, данные которого должны быть скопированы. Если этот вход оставлен неподключенным или задано пустое имя продукта, ВП копирует из источника все данные конфигурации. Название продукта не чувствительно к регистру. В качестве примеров названий продуктов можно указать **daqmx** и **serial**.

Вход **режим замены** (replace mode) задает вариант обработки существующих данных в адресате. При выборе на этом входе опции **объединить** (merge) ВП объединяет поступающие данные источника с любыми существующими данными адресата. При выборе опции заменить (replace) ВП заменяет все данные конфигурации адресата данными конфигурации источника.

На рис. 8.18 приведен пример использования ВП **Копировать данные конфигурации MAX**.



Пример использования ВП **Копировать данные конфигурации MAX**
1. Укажите источник и адресата в виде пути к файлу или system://<ip адрес>
2. Выберите продукт (например daqmx) или оставьте поле пустым для всех поддерживаемых продуктов
3. Запустите ВП. Возврат кода 0 свидетельствует об успешном копировании данных.

б)

*Рис. 8.18. Блок-диаграмма (а) и лицевая панель (б) ВП **Копировать данные конфигурации** (Copying Configurations)*

8.2. Функции интерфейса канала общего пользования (GPIB)

Канал общего пользования (КОП), или GPIB (General Purpose Interface Bus), представляет стандартный интерфейс для связи между измерительными и управляющими приборами различных производителей. Интерфейс используется, как правило, для специализированных настольных измерительных приборов с ручным управлением.

В интерфейсе GPIB используется побайтовая асинхронная схема передачи данных, то есть байты целиком последовательно передаются через шину на скорости, определяемой скоростью самого медленного участника передачи. Передача сообщений производится с помощью строк, содержащих символы ASCII. В связи с этим для формирования и декодирования сообщений используются строковые функции, рассмотренные в разделе 2.1.3.

Каждый GPIB-измерительный прибор и GPIB-интерфейсная плата имеют уникальный GPIB-адрес в диапазоне от 0 до 30. Адрес 0 обычно присваивается

GPIB-интерфейсу. Измерительные приборы, связанные с GPIB-интерфейсом, могут иметь адреса от 1 до 30. Каждое GPIB-устройство может быть **передатчиком** (talker) – источником сообщения, **слушателем** (listener) – устройством, принимающим данные, или **контроллером** (controller). Контроллер, обычно компьютер, управляет потоком информации, передаваемой по шине. Он определяет коммуникационные связи и посылает GPIB-команды измерительным приборам. ВП из палитры GPIB автоматически оперируют адресацией и большинством других управляющих команд шины.

Магистраль приборного интерфейса GPIB представляет 24-проводную параллельную шину, состоящую из восьми линий данных, пяти линий управления шиной, трех линий синхронизации и восьми заземляющих линий. Обозначения линий управления и синхронизации и краткие пояснения к ним приведены в таблице.

Обозначение	Пояснение
	Линии управления
ATN (ATTENTION) – УП (Управление)	Определяет способ интерпретации данных, поступающих по шине данных DIO
IFC (INTERFACE CLEAR) – ОИ (Очистка интерфейса)	Вызывает установку узлов приборов в исходное состояние
SRQ (SERVICE REQUEST) – ЗО (Запрос обслуживания)	Вырабатывается источником или приемником и указывает на необходимость организации с ним связи для обмена информацией
REN (REMOTE ENABLE) – ДУ (Дистанционное управление)	Переводит устройство, подключенное к шине, в режим исполнения команд с шины (а не с контрольной панели) и обратно
EOI (END OF IDENTIFY) – КП (Конец передачи)	Вырабатывается источником для обозначения конца многобайтового сообщения. Контроллер выставляет этот сигнал для инициализации параллельного опроса подключенных к шине устройств
	Линии синхронизации
DAV (DATA VALID) – СД (Синхронизация данных)	Используется передатчиком для оповещения слушателей о наличии и достоверности выставленной им информации
NRED (NOT READY FOR DATA) – ГП (Готовность к приему)	Используется слушателями для сообщения передатчику о своей неготовности принять данные
NDAC (NOT DATA ACCEPTED) – ДП (Данные приняты)	Используется слушателями для сообщения передатчику об успешном приеме данных

Вид палитры функций GPIB и подпалитры функций протокола 488.2 приведен на рис. 8.19.

Рассмотрение функций интерфейса канала общего пользования ограничено функциями из палитры GPIB.

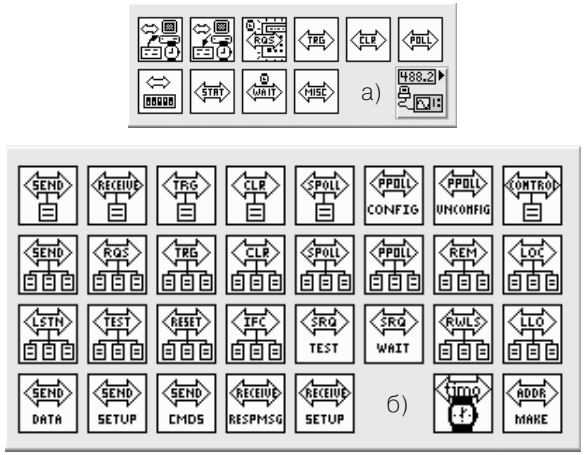
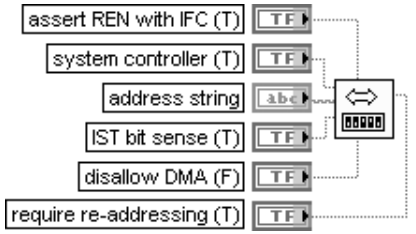


Рис. 8.19. Вид палитры функций GPIB и подпалитры функций 488.2

GPIB Initialization



Инициализация GPIB

Функция конфигурирует интерфейс GPIB в соответствии со **строкой адреса** (address string).

Если на входе **установить REN с IFC** (assert REN with IFC) установлено состояние ИСТИНА и если этот контроллер, определяемый идентификационным номером (ID) в адресной строке, является **системным контроллером** (System Controller), то функция устанавливает линию **дистанционное управление** (Remote Enable).

Если на входе **системный контроллер** (system controller) установлено состояние ИСТИНА, то этот контроллер является системным.

Вход **строка адреса** (address string) устанавливает адрес GPIB самого контроллера GPIB. Значением по умолчанию для строки адреса является сконфигурированный адрес для основного контроллера GPIB системы. Сконфигурированный адрес обычно равен 0. Как правило, этот вход не подключается.

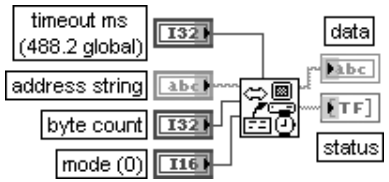
Если LabVIEW использует несколько контроллеров GPIB, то для указания используемого функцией контроллера перед строкой адреса указывается префикс в виде **ID:адрес** (или **ID:** если нет необходимости указания адреса). Если ID контроллера отсутствует, то функция по умолчанию относится к контроллеру (или шине) 0.

Если на входе **значение бита IST** (IST bit sense) установлено значение ИСТИНА, то **индивидуальный бит статуса** (Individual Status bit) устройства имеет такое же значение при параллельном опросе.

Если вход **запретить прямой доступ к памяти** (disallow DMA) установлен в состояние ИСТИНА, то данное устройство использует программный ввод/вывод для передачи данных.

Если на входе **требует переадресации** (require re-addressing) установлено состояние ИСТИНА, то функция адресует устройство перед каждым чтением или записью. При установке состояния ЛОЖЬ устройство должно сохранять адресацию от одного чтения или записи до другого

GPIB Read



Читать из GPIB

Функция считывает из GPIB-устройства с адресом, заданным в **строке адреса** (address string), число байтов, определенное на входе **подсчет байтов** (byte count). Для установки синхронного чтения необходимо в контекстном меню функции выбрать опцию **выполнять ввод/вывод синхронно** (Do I/O Synchronously).

Вход **лимит времени ожидания, мс** (timeout ms) определяет интервал времени, в течение которого должна завершиться операция чтения. При превышении лимита операция прерывается и устанавливается бит 14 **статуса** (status). Для блокирования тайм-аута необходимо установить на этом входе значение 0. Для использования общего тайм-аута 488.2 этот вход не подключается. Изменение значения по умолчанию общего тайм-аута 488.2 производится с помощью функции **Установить тайм-аут** (SetTimeOut). Первоначальное значение по умолчанию этого входа равно 10,000.

Вход **строка адреса** (address string) содержит адрес GPIB-устройства, с которым функция поддерживает связь. В строку адреса можно ввести одновременно первичный и вторичный адреса, используя форму записи **первичный+вторичный**.

Если адрес не определен, то функции не выполняют адресацию перед попыткой чтения или записи строки. Они предполагают, что эти команды отправлены другим путем или что другой контроллер назначен ответственным за адресацию. Если контроллер, обязанный адресовать устройство, не выполнил адресацию до истечения лимита времени, то функция завершается с ошибкой 6 GPIB (тайм-аут) и устанавливает бит 14 **статуса**. Если GPIB не является ответственным контроллером (Controller-In-Charge), то строка адреса не должна определяться.

Вход **режим** (mode) определяет условия завершения чтения, отличающиеся от достижения предельного значения счетчиком байтов. В таблице указаны достоверные значения и соответствующие символы **конца строки** (end-of-string) (EOS). Любой режим, не приведенный в таблице, определяется десятичным числом символа конца строки, выбираемым пользователем.

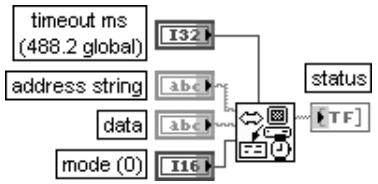
0	Отсутствие символа EOS. Режим завершения по концу строки отключен
1	Символом EOS является CR (Возврат каретки). Чтение завершается по сигналу линии EOI, по счетчику байтов или по символу CR
2	Символом EOS является LF (Перевод строки). Чтение завершается по сигналу линии EOI, по счетчику байтов или по символу LF

Выход **статус** (status) описывает состояние GPIB-контроллера. Функция **Чтение GPIB** завершается при выполнении одной из следующих задач:

- считывается заданное число байтов;
- обнаруживается ошибка;
- превышается лимит времени;
- обнаруживается сообщение КОНЕЦ (сигнал линии EOI установлен);
- обнаруживается символ EOS (если эта опция разрешена значением, установленным на входе **режим**).

Функция сравнивает все восемь битов при проверке символа EOS

GPIB Write



Записать в GPIB

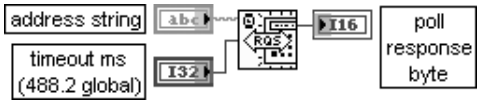
Функция записывает **данные** (data) в GPIB-устройство, определяемое **строкой адреса** (address string).

Вход **режим** (mode) определяет вариант завершения функции **Записать в GPIB**.

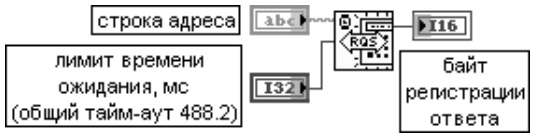
0	Послать EOI с последним символом строки
1	Добавить CR к строке и послать EOI с CR
2	Добавить LF к строке и послать EOI с LF
3	Добавить CR LF к строке и послать EOI с LF
4	Добавить CR к строке, но не посылать EOI
5	Добавить LF к строке, но не посылать EOI
6	Добавить CR LF к строке, но не посылать EOI
7	Не посылать EOI

Выход **статус** (status) является логическим массивом, в котором каждый бит описывает состояние GPIB-контроллера. При возникновении ошибки функция GPIB устанавливает бит 15. **Ошибка GPIB** (GPIB error) достоверна только при установке бита 15 статуса. Более подробная информация о битах статуса и кодах ошибок приведена при рассмотрении функции **Статус GPIB** (GPIB Status)

Wait for GPIB RQS



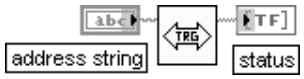
Ожидать GPIB RQS



Функция ожидает, пока устройство, указанное в **строке адреса** (address string), установит сигнал RQS.

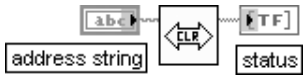
Выход **байт регистрации ответа** (poll response byte). Если линия SRQI установлена, то функция опрашивает устройство по заданному адресу для обнаружения запроса на обслуживание. Когда заданное устройство запрашивает обслуживание (установлен бит 6 в байте регистрации ответа), то функция возвращает последовательную регистрацию ответа. Если устройство, указанное в строке адреса, не отвечает за отведенное время, то **байт регистрации ответа** содержит ?1

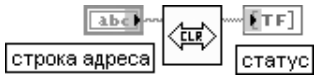
GPIB Trigger / Запуск GPIB



Функция посылает команду **Запуск прибора** (GET – Group Execute Trigger) устройству, указанному в **строке адреса** (address string)

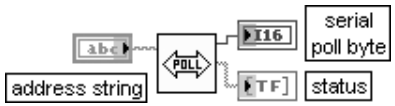
GPIB Clear / Очистка GPIB

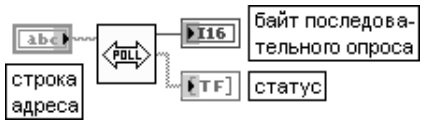




Функция посылает команды **Сброс адресный** (сброс приборов, назначенных слушателями (SDC – Selected Device Clear) или **Сброс универсальный** (DCL – Device Clear))

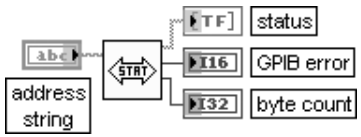
GPIB Serial Poll / Последовательный опрос GPIB

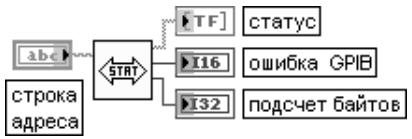




Функция выполняет последовательный опрос устройства, указанного в строке адреса. Выход **байт последовательного опроса** (serial poll byte) представляет ответ устройства. Если адресуемое устройство не ответило за отведенное время ожидания, то на данном выходе выводится значение –1

GPIB Status / Статус GPIB





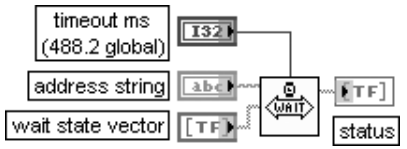
Функция показывает текущий статус GPIB-контроллера, заданного в строке адреса. Таблица показывает числовое значение и символический статус каждого бита выходного логического массива **статус** (status). Таблица также включает описание каждого бита.

Бит статуса	Числовое значение	Символический статус	Описание
0	1	DCAS	**Состояние очистки устройства** (Device Clear state)
1	2	DTAS	**Состояние запуска устройства** (Device Trigger State)
2	4	LACS	**Слушатель активен** (Listener Active)
3	8	TACS	**Передатчик активен** (Talker Active)
4	16	ATN	**Объявлено внимание** (Attention Asserted)
5	32	CIC	**Ответственный контроллер** (Controller-In-Charge)
6	64	REM	**Дистанционное состояние** (Remote State)
7	128	LOK	**Блокированное состояние** (Lockout State)
8	256	CMPL	**Операция выполнена** (Operation Completed)

Бит статуса	Числовое значение	Символический статус	Описание
12	4096	SRQI	**Обнаружен SRQ пока CIC** (SRQ Detected while CIC)
13	8192	END	**Обнаружен EOI или EOS** (Detected EOI or EOS Detected)
14	16384	TIMO	**Истечение времени ожидания** (Timeout)
15	–32768	ERR	**Обнаружена ошибка** (Error Detected)

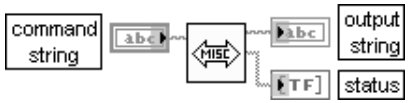
Выход **подсчет байтов** (byte count) содержит число байтов, посланных предыдущей операцией GPIB

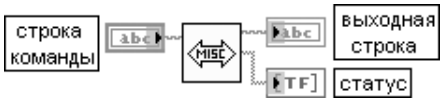
GPIB Wait / Ожидать GPIB



Функция ожидает состояния (состояний), заданных на входе **ожидать вектор состояния** (wait state vector), для устройства, определенного строкой адреса.
Каждый логический элемент в массиве соответствует состоянию, которое может ожидаться в устройстве. Если в состояние ИСТИНА установлено более одного элемента, то функция завершается при наличии любого состояния из набора установленных.
Значения, которые могут быть заданы на входе **ожидать вектор состояния**, по большей части идентичны по составу значениям, перечисленным в таблице **статуса** функции **Ожидать GPIB**, рассмотренной выше. Отличие связано с отсутствием значений с символическим обозначением CMPL и ERR

GPIB Misc / Разные функции GPIB





Функция выполняет операции GPIB, отображаемые в **строке команды** (command string). Эта функция низкого уровня используется, когда другие функции высокого уровня неприемлемы.
Следующая таблица показывает функции устройства или контроллера, которые могут быть определены в **строке команды** (command string). (Дополнение *address* показывает, что пользователь должен добавить адрес в строке команды.)

Функции устройства	Описание
loc *address*	Временно переводит устройство из режима дистанционного программного режима в локальный режим. ***Адрес*** *(address)* является GPIB-адресом устройства. Этот аргумент показывает оба адреса – первичный и вторичный, если используется форма *первичный+вторичный*, где *первичный* и *вторичный* являются десятичными значениями первичного и вторичного адресов

Функции устройства	Описание
off *address*	Переводит устройство с заданным адресом в автономное состояние. Эта функция необходима только при разделении применения устройства с другим приложением, которое использует библиотеку NI-488
pct *address*	Передает полномочия **ответственного контроллера** (Controller-In-Charge (CIC)) устройству с заданным адресом. GPIB-контроллер становится нерабочим автоматически. Функция предполагает, что устройство, к которому **pct** передает управление, имеет возможности контроллера
ppc byte *address*	Разрешает устройству отвечать на параллельный опрос. **Байт** (byte) является 0 или достоверной командой **включения параллельного опроса** (parallel poll enable (PPE)). Если байт является 0, то посылается байт **выключения параллельного опроса** (parallel poll disable (PPD)) 0x70 для выключения ответа устройства на параллельный опрос. Каждое из 16 сообщений PPE выбирает линию данных GPIB (DIO1 – DIO8) и значение (1 или 0), которые устройство должно использовать при ответе на сообщение **идентификации** (Identify (IDY)) во время параллельного опроса. Устройство сравнивает значение **индивидуального бита статуса** (individual status bit (ist)) и переводит отображаемую линию DIO в состояние ИСТИНА или ЛОЖЬ
cac 0/1	Берет управление синхронно (0) или немедленно (1), а в некоторых случаях и асинхронно. Обычно нет необходимости использовать эту функцию, поскольку другие функции, такие как **cmd** и **rpp**, берут управление автоматически.
cmd string	Посылает сообщения команд GPIB. Эти сообщения команд включают адреса устройств-передатчиков и устройств-слушателей, вторичные адреса, сообщения конфигурации последовательного и параллельного опроса, сообщения очистки устройства и запуска. **cmd** не используется для передачи программных инструкций устройствам. Функции **Чтение GPIB** (GPIB Read) и **Запись GPIB** (GPIB Write) передают программные инструкции и другую приборную информацию. ***Строка*** *(string)* содержит командные байты, отправляемые контроллером. ASCII-символы представляют эти байты в строке **cmd**
dma 0/1	Устанавливает режим ПДП (1) или режим программируемого ввода/вывода (0). Некоторые платы GPIB не имеют ПДП. В этом случае при попытке выполнения **dma** 1 функция вернет ошибку 11 GPIB
gts 0/1	Устанавливает GPIB-контроллер в резервное состояние и снимает сигнал ATN, если он является активным контроллером. Обычно GPIB-контроллер участвует в передаче данных. **gts** разрешает GPIB-устройствам передавать данные без участия GPIB-контроллера. После посылки команды **gts** необходимо ожидать END перед вводом другой команды GPIB. Это можно сделать с помощью функции **Ожидать GPIB** (GPIB Wait)
ist 0/1	Устанавливает индивидуальный бит статуса
llo	Запрещает местное управление, то есть восприятие сигналов от лицевой панели устройства. **llo** посылает команду **Запрет местного управления** (Local Lockout) (LLO))
loc	Переводит GPIB-контроллер в локальное состояние с помощью отправки локального сообщения **Вернуться в локальное** (Return To Local (RTL)), если он не зафиксирован в режиме дистанционного управления (индицируемом битом LOK статуса). Функция **loc** применяется для имитации переключателя RTL лицевой панели при использовании компьютера для имитации прибора
off	Переводит контроллер в автономное состояние. Эта функция необходима только при разделении применения контроллера с другим приложением, которое использует библиотеку NI-488

Функции устройства	Описание
ppc byte	Конфигурирует GPIB-контроллер на участие в параллельном опросе путем присвоения его сообщению **Локальный опрос разрешен** (Local Poll Enable (LPE)) значения ***байта*** *(byte)*. Если значение байта равно 0, то GPIB-контроллер устанавливается в исходное состояние.
ppu	Блокирует ответ всех устройств на параллельный опрос. **ppu** посылает команду **Отмена конфигурации параллельного опроса** (Parallel Poll Unconfigure (PPU))
rpp	Проводит параллельный опрос предварительно сконфигурированных устройств с помощью установки сигналов ATN и EOI, которые посылают сообщение IDY. **rpp** размещает ответ параллельного опроса в выходной строке как ASCII символы
rsc 0/1	Отменяет или запрашивает возможность GPIB-контроллера посылать сообщения **Сброс интерфейса** (Interface Clear (IFC)) и **Дистанционное управление** (Remote Enable (REN)) GPIB-устройству, используя функции **sic** и **sre**. Чтобы GPIB-контроллер отвечал на IFC, посланное другим контроллером, GPIB-контроллер не должен быть системным контроллером. В большинстве приложений GPIB-контроллер всегда является системным контроллером. **rsc** используется, если только компьютер не является системным контроллером во время выполнения программы
rsv byte	Запрашивает обслуживание и/или устанавливает байт статуса последовательного опроса равным значению *байта*. Если в байте установлен бит 0x40, то GPIB-контроллер также запрашивает обслуживание из контроллера с помощью установки линии RQS GPIB.
sic	Инициирует формирование в контроллере сигнала IFC длительностью не менее 100 мс, если контроллер имеет полномочия системного контроллера. Это действие инициализирует GPIB и делает порт контроллера ответственным контроллером (CIC). Сигнал IFC сбрасывает только функции GPIB шинного устройства, он не сбрасывает внутренние функции устройства. Команды **Сброс универсальный** (Device Clear (DCL)) и **Сброс адресный** (Selected Device Clear (SDC)) сбрасывают функции устройства
sre 0/1	Устанавливает (1) или очищает (0) линию дистанционного управления (REN). Устройства контролируют состояние линии REN, когда они выбирают между местным и дистанционным режимами функционирования. Устройство не войдет в дистанционный режим, пока оно принимает собственный адрес слушателя

В качестве примера использования функций интерфейса канала общего пользования на рисунке 8.20 приведена блок-диаграмма ВП LabVIEW <–> GPIB из набора примеров NI Example Finder LabVIEW. ВП производит запись набора символов в GPIB-устройство, определяемое строкой адреса GPIB, и считывание заданного числа байтов из того же устройства. После выполнения операций записи или считывания производится анализ статуса GPIB-устройства (рис. 8.21). При этом с помощью структуры **Цикл с фиксированным числом итераций** производится формирование строки статуса, содержащей символический статус и описание каждого ненулевого бита статуса. С помощью структуры **Вариант** производится формирование строки ошибки и при необходимости вывод диалогового окна.

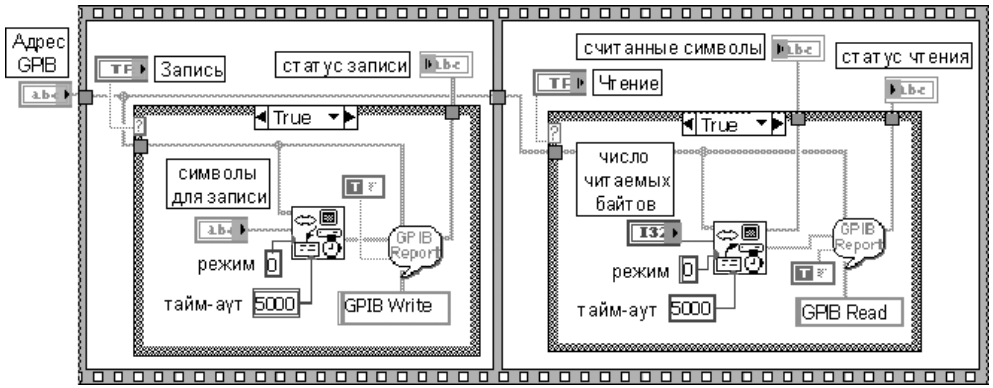


Рис. 8.20. Блок-диаграмма ВП LabVIEW <–> GPIB

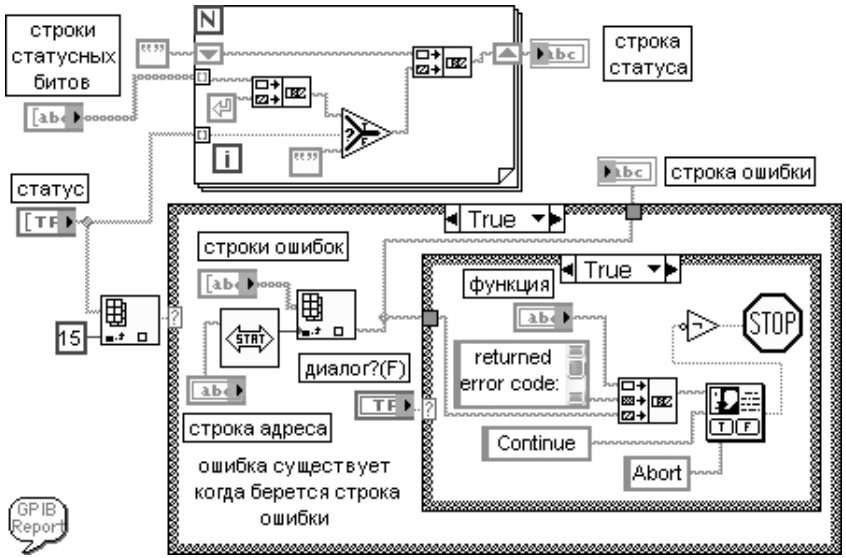


*Рис. 8.21. Иконка и блок-диаграмма ВП **Сообщение об ошибке GPIB** (GPIB Error Report)*

8.3. Функции последовательной коммуникации

Передача данных по каналам последовательной связи была и остается наиболее распространенным способом взаимодействия компьютеров и периферийных устройств. При этом в качестве основных протоколов обмена данными до недавнего времени использовались протоколы RS-232 и RS-485. В основе их работы лежит последовательная бит за битом передача данных от передатчика к приемнику по двум проводам. Каждый передаваемый символ упаковывается в кадр символа, состоящий из одиночного **стартового бита** (start bit), **битов данных** (data bits), **бита четности** (parity bit) и заданного числа **стоповых битов** (stop bits).

Вид палитры функций последовательной коммуникации (Serial) приведен на рис. 8.22а. Данные функции являются специализированными функциями VISA для последовательных портов. В состав палитры входят функции конфигурирование последовательного порта VISA, чтения, записи и закрытия VISA.

В настоящее время все большую популярность приобретает канал USB (Universal Serial Bus). Функции управления передачей по каналу USB располагаются в подпалитре VISA USB (рис. 8.22б).

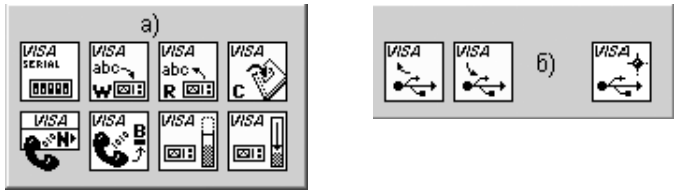
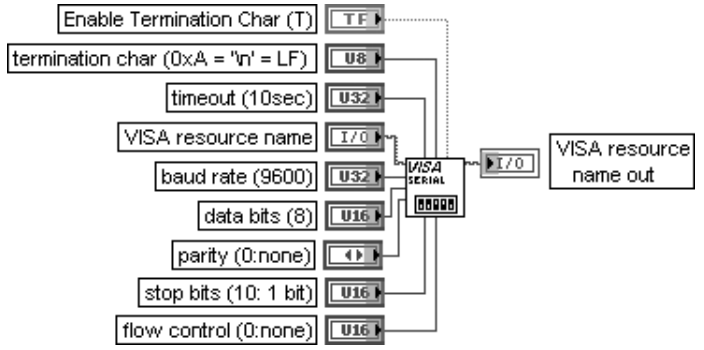
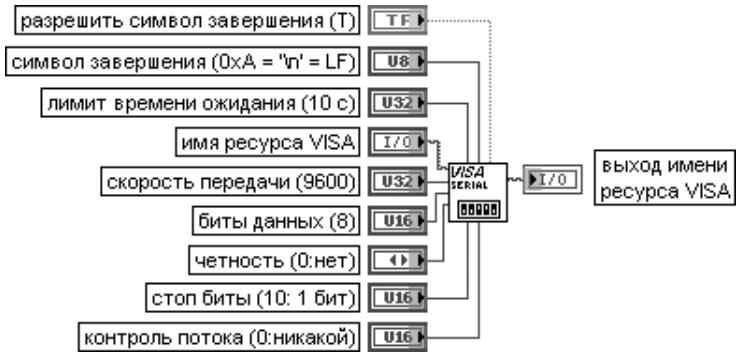


Рис. 8.22. Вид палитры функций последовательной коммуникации (а) и функций управления передачей по каналу USB (б)

VISA Configure Serial Port



Конфигурировать последовательный порт VISA



Функция инициализирует последовательный порт, определяемый с помощью входа **имя ресурса VISA** (VISA resource name), производя определенные установки. Этот полиморфный ВП может использоваться для инициализации последовательного порта с помощью узла свойств класса **Instr VISA** или класса **Serial Instr VISA**, являющегося основным элементом блок-диаграммы ВП. Класс VISA, заданный **именем ресурса VISA**, определяет используемую реализацию полиморфного ВП.

При подаче на вход **разрешить символ завершения** (Enable Termination Char) значения ИСТИНА (по умолчанию) последовательное устройство подготавливается к распознаванию **символа завершения** (termination char). В противном случае оно не распознает такой символ.

Символ с входа **символ завершения** (termination char) вызывается для завершения операции чтения. Операция чтения завершается при считывании **символа завершения** из последовательного устройства. 0xA является шестнадцатеричным эквивалентом символа **перевод строки** (linefeed character (\n)). Для строк сообщений, которые завершаются символом **возврат каретки** (carriage return (\r)), на этом входе необходимо установить значение 0xD.

Значение на входе **лимит времени ожидания** (timeout) устанавливает допустимое значение интервала ожидания выполнения операций записи или чтения.

Вход **имя ресурса VISA** (VISA resource name) определяет открываемый ресурс. Этот элемент управления также определяет сессию и класс.

Вход **биты данных** (data bits) определяет число битов в поступающих данных. Значение изменяется в диапазоне от 5 до 8 битов. По умолчанию число битов равно 8.

Вход **четность** (parity) определяет способ проверки четности, используемый для каждого передаваемого или принимаемого кадра:

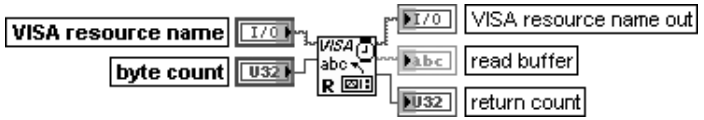
0	1	2
Нет проверки четности (по умолчанию) (no parity)	**Проверка на нечетность** (odd parity)	**Проверка на четность** (even parity)

Вход **стоп-биты** (stop bits) определяет число стоповых битов, используемых для индикации конца кадра:

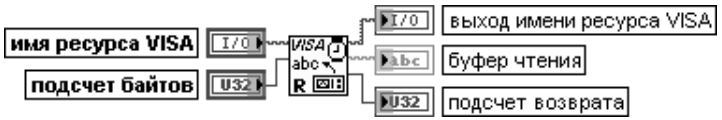
10	15	20
1 стоп-бит	1.5 стоп-бита	2 стоп-бита

Вход **контроль потока** (flow control) устанавливает тип контроля, используемого механизмом передачи

VISA Read



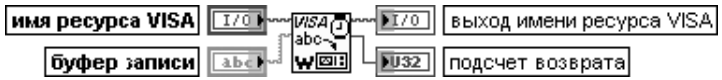
Читать VISA



Функция считывает заданное число байтов из устройства или интерфейса, определенного с помощью **имени ресурса VISA** (VISA resource name), и возвращает данные в **буфере чтения** (read buffer). Синхронность или асинхронность считывания данных зависит от платформы. Для установки синхронного чтения необходимо в контекстном меню функции выбрать опцию **выполнять ввод/вывод синхронно** (Do I/O Synchronously). Операция возвращается только после завершения передачи

VISA Write

Записать VISA



Функция записывает данные из **буфера записи** (write buffer) в устройство или интерфейс, определенные с помощью **имени ресурса VISA** (VISA resource name).

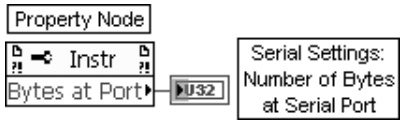
Выход **подсчет возврата** (return count) содержит действительное число записанных байтов

VISA Close

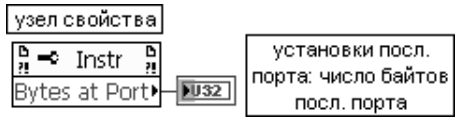
Закрыть VISA

Функция закрывает сессию устройства или объект события, определенные с помощью **имени ресурса VISA** (VISA resource name). Каждая открытая сессии VISA должна быть закрыта при завершении работы с ней. Эта функция воспринимает все доступные классы

VISA Bytes at Serial Port



Байты последовательного порта VISA

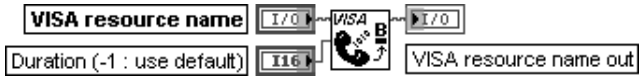


Функция возвращает число байтов входного буфера заданного последовательного порта.

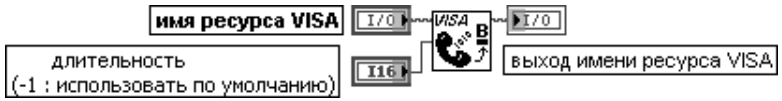
Свойство **число байтов последовательного порта** (Number of Bytes at Serial Port) определяет число байтов, доступных в текущее время в последовательном порту, используемом в данной сессии. Функция **Байты последовательного порта VISA** является **Узлом свойств** (Property Node) класса **VISA:I/O Session:Instr**.

С помощью узла свойств указанного класса могут быть программно установлены или считаны все параметры последовательного порта. Именно такая установка параметров последовательного порта лежит в основе рассмотренной выше функции его конфигурирования

VISA Serial Break

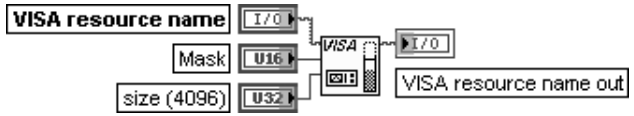


Пауза последовательного порта VISA

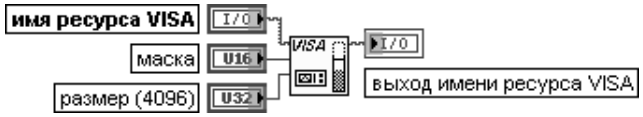


Функция посылает паузу (по умолчанию 250 мс) в заданный выходной порт

VISA Set I/O Buffer Size

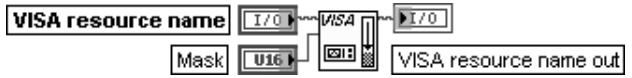


Установить размер буфера ввода/вывода VISA



Функция устанавливает размер буфера ввода/вывода.
Вход **маска** (mask) определяет, для какого из буферов должен устанавливаться размер. Допустимые варианты включают установку размеров приемного, передающего или обоих буферов.
Вход **размер** (size) определяет размер буфера (по умолчанию 4096). Если этот вход не подключен, значение по умолчанию зависит как от VISA, так и от операционной системы

VISA Flush I/O Buffer



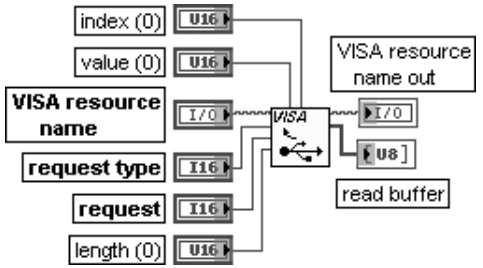
Очистить буфер ввода/вывода VISA

Функция очищает буфер ввода/вывода.
Вход **маска** (mask) определяет буфер, подлежащий очистке. Варианты значений приведены в таблице. Для одновременного выполнения операций можно объединить значения по ИЛИ, однако следует использовать только одну маску для приемного и передающего буфера.

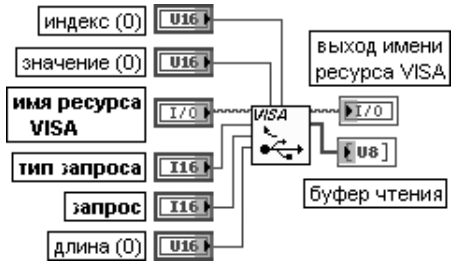
Значения маски	Шестнадца-теричные коды	Описание
16	0x10	Очищает и отбрасывает содержимое приемного буфера (то же самое, что и 64)

Значения маски	Шестнадца-теричные коды	Описание
32	0x20	Очищает и отбрасывает содержимое передающего буфера с помощью записи всех буферизированных данных в устройство
64	0x40	Очищает и отбрасывает содержимое приемного буфера (не производит какой-либо ввод/вывод в устройство)
128	0x80	Очищает и отбрасывает содержимое передающего буфера (не производит какой-либо ввод/вывод в устройство)

VISA USB Control In

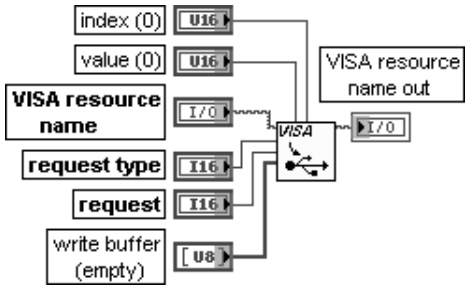


Управление входной точкой VISA USB

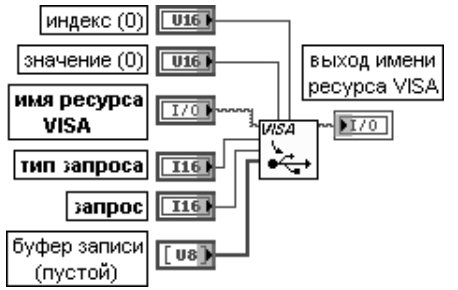


Функция выполняет управление передачей по каналу USB от устройства USB. Значения данных, загружаемые на стадии установки управления передачей, берутся как параметры и включают bmRequestType, bRequest, wValue, wIndex и wLength. Дополнительный буфер данных также считывается, если для этой передачи требуется этап передачи данных.
Вход **индекс** (index) передает параметр устройству. Значение, которое вводится здесь, зависит от значения, введенного на входе **запрос** (request). Индекс часто используется в запросах для определения **конечной точки** (endpoint) или интерфейса.
Вход **значение** (value) передает параметр устройству. Значение, которое вводится здесь, зависит от значения, введенного на входе **запрос** (request).
Вход **тип запроса** (request type) является числовым представлением запроса, который посылается устройству. Этот параметр является полем с побитовым отображением, которое позволяет распознать характеристики заданного запроса. Бит, определяющий направление, должен быть установлен в 1 (от устройства к хосту).
Вход **запрос** (request) определяет отдельный запрос. Запрос, который может быть введен, зависит от значения, введенного на входе **тип запроса** (request type).
Вход **длина** (length) определяет длину передаваемых данных во время второй фазы управления передачей. При этом направление установлено от устройства к хосту.
Выход **буфер чтения** (read buffer) содержит данные, считываемые из устройства

VISA USB Control Out

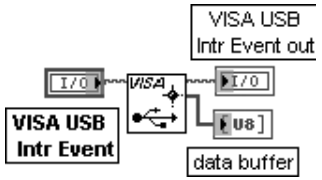


Управление выходной точкой VISA USB

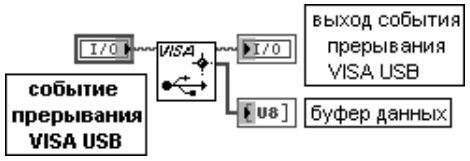


Функция выполняет управление передачей по каналу USB к устройству USB. Большая часть входов описана при рассмотрении предыдущей функции.
Вход **буфер записи** (write buffer) содержит данные, записываемые в устройство

VISA Get USB Interrupt Data



Получить данные прерывания VISA USB



Функция извлекает данные прерывания, которые сохраняются в событии прерывания VISA USB.
Вход **VISA USB Intr Event** содержит уникальный логический идентификатор к событию прерывания VISA USB.
Выход **буфер данных** (data buffer) является буфером данных прерывания USB

В качестве примера применения функции последовательной коммуникации на рис. 8.23 приведена блок-диаграмма модернизированного ВП LabVIEW«Serial из набора примеров NI Example Finder LabVIEW. При выполнении ВП производится последовательная инициализация порта, запись байтов строки в порт, считывание заданного числа байтов из порта и закрытие сессии с портом.

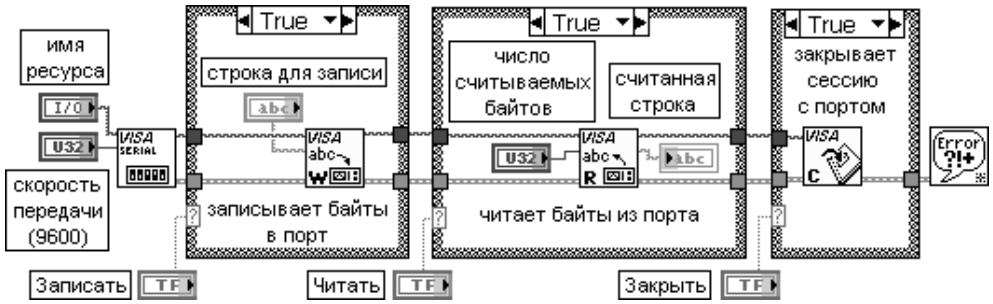
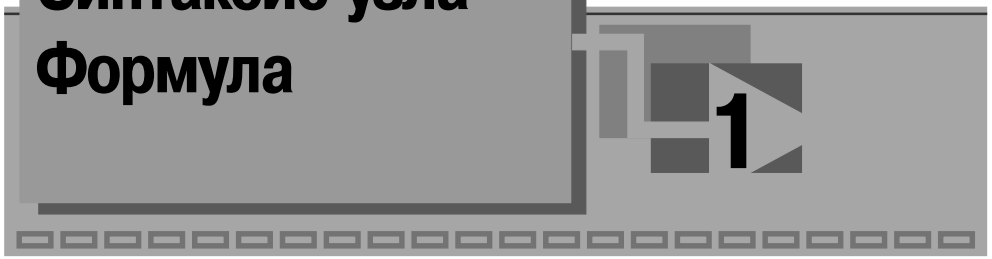


Рис. 8.23. Блок-диаграмма модернизированного ВП LabVIEW«Serial

Приложение
Синтаксис узла Формула



Синтаксис узла **Формула** аналогичен синтаксису текстовых языков программирования.

Все переменные, объявленные в блоках программы, ограниченных скобками, доступны только в этих блоках. Все входные терминалы считаются переменными внешнего (самого крайнего) блока (не заключенного в скобки) и не могут быть объявлены еще раз в этом блоке. LabVIEW старается сопоставить переменные, объявленные во внешнем блоке (не заключенном в скобки), с выходным терминалом с тем же именем.

оператор (statement):

1. **объявление переменной** (variable-declaration).
2. **присваивание** (assignment).
3. **составной оператор** (compound-statement).
4. **условный оператор** (conditional-statement)
5. **итеративный оператор** (iterative-statement)
6. **оператор переключения** (switch-statement).
7. **оператор управления** (control-statement).
   1. **объявление переменной** (variable-declaration):
      1.1. **идентификатор определителя типа** (type-specifier identifier),
      1.2. **идентификатор определителя типа** (type-specifier identifier) **список индексов массива** (array-index-list),
      1.3. **идентификатор определителя типа** (type-spec identifier) **= присваивание** (assignment).
         1.1. **определитель типа** (type-specifier):
            1.1.1. **тип с плавающей точкой** (floating-point- type): float, float32, float64,
            1.1.2. **целый тип** (integer-type): int, int8, int16, int32, uInt8, uInt16, uInt32.
         1.2. **список индексов массива** (array-index-list):
            1.2.1. **[целая константа** (integer-constant)**]**,
            1.2.2. **[целая константа** (integer-constant)**] список индексов массива** (array-index-list).

2. **присваивание** (assignment):
   2.1. **выражение** (expression),
   2.2. **левосторонний оператор присваивания** (left-hand-side assignment-operator) **присваивание** (assignment).
      2.1. **выражение** (expression):
         2.1.1. **выражение** (expression) **двоичный оператор** (binary-operator) **выражение** (expression),
         2.1.2. **унарный оператор** (unary-operator) **выражение** (expression),
         2.1.3. **выражение** (expression) **унарный оператор** (unary-operator),
         2.1.4. **выражение** (expression) **? выражение** (expression) **: выражение** (expression),
         2.1.5. **(выражение** (expression)**),**
         2.1.6. **идентификатор** (identifier),
         2.1.7. **константа** (constant),
         2.1.8. **имя функции** (function-name) **(список аргументов** (argumentlist)**)**.
      2.2. left-hand-side:
         2.2.1. **идентификатор** (identifier),
         2.2.2. **идентификатор** (identifier) **описание массива** (array-subscription).
            2.2.2. **описание массива** (array-subscription):
            2.2.2.1. **[присваивание** (assignment)**]**
            2.2.2.2. **[присваивание** (assignment)**] описание массива** (array-subscription).
               - **оператор присваивания** (assignment-operator):
                 =, +=, –=, *=, /=, >>=, <<=, &=, ^=, |=, %=, **=.
               - **двоичный оператор** (binary-operator):
                 +, –, *, /, ^, !=, ==, >, <, >=, <=, &&, ||, &, |, %, **.
               - **унарный оператор** (unary-operator):
                 +. –, !, ++, ––, ~.

               список аргументов (argument-list):
               выражение (expression),
               выражение (expression) , **список аргументов** (argument-list).
               константа (constant): **pi**, **число** (number).
3. **составной оператор** (compound-statement):
   { список операторов (statement-list) **}**.
4. **условный оператор** (conditional-statement):
   4.1. **оператор if** (if-statement),
   4.2. **оператор if-else** (if-else-statement).

   4.1. **оператор if** (if-statement):
      if (присваивание) оператор (if (assignment**)** statement),
   4.2. **оператор if-else** (if-else-statement):
      if оператор else оператор (if-statement **else** statement).
5. **итеративный оператор** (iterative-statement):
   5.1. **оператор do** (do-statement):
      do оператор while (присваивание) (**do** statement **while (**assignment**)**).
   5.2. **оператор for** (for-statement):
      for ([**присваивание**] ; [**присваивание**] ; [**присваивание**] **) оператор (for (** [assignment] ; [assignment] ; [assignment] **)** statement).
   5.3. **оператор while** (while-statement):
      while (присваивание) оператор (**while (** assignment **)** statement).
6. **оператор управления** (control-statement):
   6.1. **прервать (break)**,
   6.2. **продолжить (continue)**.
7. **оператор переключения** (switch-statement):
   switch (присваивание) { список операторов варианта }
   (**switch (** assignment **) {** case-statement-list **}**)
      список операторов варианта (case-statement-list):
         оператор варианта (case-statement),
         список операторов варианта (case-statement-list) **оператор варианта** (case-statement).
      оператор варианта (case-statement):
      номер варианта (case number**): список операторов** (statement-list),
      по умолчанию (default): список операторов (statement-list).

не цифра (non-digit):
   один из символов **a~z A~Z**.
цифра (digit):
   один из символов **0 1 2 3 4 5 6 7 8 9**.
ненулевая цифра (non-zero-digit):
   один из символов **1 2 3 4 5 6 7 8 9**.
двоичная цифра (binary-digit):
   один из символов **0 1**.
восьмеричная цифра (octal-digit):
   0 1 2 3 4 5 6 7.
шестнадцатеричная цифра (hex-digit):
   0 1 2 3 4 5 6 7 8 9 a b c d e f A B C D E F.
идентификатор (identifier):
   не цифра (non-digit) [**не первый символ** (non-first-character)].
не первый символ (non-first-character):
   не цифра (non-digit) [**не первый символ** (non-first-character)],
   цифра (digit) [**не первый символ** (non-first-character)].

число (number):
целая константа (integer-constant)
константа с плавающей точкой (float-constant)
целая константа (integer-constant):
десятичная константа (decimal-constant),
двоичная константа (binary-constant),
восьмеричная константа (octal-constant),
шестнадцатеричная константа (hex-constant).
десятичная константа (decimal-constant):
ненулевая цифра (non-zero-digit) **набор цифр** (#digit).
двоичная константа (binary-constant):
0b набор двоичных цифр (#binary-digit),
0B набор двоичных цифр (#binary-digit).
восьмеричная константа (octal-constant):
0 набор восьмеричных цифр (#octal-digit).
шестнадцатеричная константа (hex-constant):
0x набор шестнадцатеричных цифр (#hex-digit),
0X набор шестнадцатеричных цифр (#hex-digit).
константа с плавающей точкой (float-constant):
дробь (fraction) **экспоненциальная часть** (exponent-part),
десятичная константа (decimal-constant) **экспоненциальная часть** (exponent-part).
дробь (fraction):
набор цифр (#digit) . **цифра** (digit) **набор цифр** (#digit).
экспоненциальная часть (exponent-part):
e [**знак** (sign)] **набор цифр** (#digit),
E [**знак** (sign)] **набор цифр** (#digit).
знак (sign):
+ или **–**.

Приложение 2 Перечень «горячих» клавиш (Keyboard Shortcuts)

«Горячая» клавиша	Действие
Объект/Движение	
Shift+щелчок	Выбирает несколько объектов; добавляет объекты к текущему выбору
Клавиши со стрелками (Arrow keys)	Перемещает выбранный объект на один элемент (пиксел)
Shift+клавиши со стрелками	Перемещает выбранный объект на несколько элементов
Shift+щелчок (перенос)	Перемещает выбранный объект по одной оси
Ctrl+щелчок (перенос)	Дублирует выбранный объект
Ctrl+Shift+щелчок (перенос)	Дублирует выбранный объект и перемещает его по одной оси
Shift+изменение размера	Изменяет размеры выбранного объекта, сохраняя пропорции
Ctrl+изменение размера	Изменяет размеры выбранного объекта, сохраняя положение центра
Ctrl+перетаскивание прямоугольника	Вставляет пустое рабочее пространство на лицевую панель или на блок-диаграмму
Ctrl+A	Повторяет последнюю операцию выравнивания объектов
Ctrl+D	Повторяет последнюю операцию распределения объектов
Двойной щелчок на открытом пространстве	Размещает свободную метку на лицевой панели или на блок-диаграмме
Ctrl+колесо мыши	Выполняет прокрутку через поддиаграммы структур Вариант, Событие или Стековая последовательность
Навигация по среде LabVIEW	
Ctrl+E	Отображает блок-диаграмму или лицевую панель
Ctrl+#	Включает или выключает выравнивающую сетку
Ctrl+/	Увеличивает размер окна до максимального
Ctrl+T	Размещает рядом лицевую панель и панель блок-диаграммы
Ctrl+F	Выполняет поиск объекта или текста
Ctrl+G	Выполняет поиск в ВП следующего экземпляра объекта или текста
Ctrl+Shift+G	Выполняет поиск в ВП предыдущего экземпляра объекта или текста
Ctrl+Shift+F	Отображает окно **Результаты поиска** (Search Results)
Ctrl+Tab	Циклически перемещается между окнами LabVIEW
Ctrl+I	Отображает диалоговое окно **Свойства ВП**
Ctrl+L	Отображает диалоговое окно **Список ошибок** (Error List)
Ctrl+Y	Отображает диалоговое окно **История** (History)
Ctrl+D	Перерисовывает окно иерархии (Hierarchy)
Ctrl+A	Показывает все ВП в окне иерархии
Enter	Находит следующий узел, идентичный искомой строке, после инициализации поиска в окне иерархии

«Горячая» клавиша	Действие
Shift-Enter	Находит предыдущий узел, идентичный искомой строке, после инициализации поиска в окне иерархии
Отладка	
Ctrl+стрелка вниз	Выполняет шаг в узел
Ctrl+стрелка вправо	Выполняет шаг через узел
Ctrl+стрелка вверх	Выполняет шаг из узла
Файловые операции	
Ctrl+N	Создает новый ВП
Ctrl+O	Открывает существующий ВП
Ctrl+W	Закрывает ВП
Ctrl+S	Сохраняет ВП
Ctrl+P	Печатает окно
Ctrl+Q	Выходит из LabVIEW
Основное редактирование	
Ctrl+Z	Отменяет действие
Ctrl+Shift+Z	Возвращает отмененное действие
Ctrl+X	Вырезает объект
Ctrl+C	Копирует объект
Ctrl+V	Вставляет объект
Помощь	
Ctrl+H	Отображает окно контекстной помощи
Ctrl+Shift+L	Блокирует окно контекстной помощи
Ctrl+? или <F1>	Отображает окно помощи LabVIEW
Инструменты и палитры	
Ctrl	Переключает на следующий наиболее полезный инструмент
Shift	Переключает на инструмент позиционирования
Ctrl+Shift через открытое пространство	Переключает на инструмент прокрутки
Пробел	Переключается между двумя наиболее общими инструментами если автоматический выбор инструментов отключен
Shift+табуляция	Включает автоматический выбор инструмента
Табуляция	Циклически переключает четыре наиболее общих инструмента в случае если автоматический выбор инструмента отключен. В противном случае осуществляется автоматический выбор инструмента
Клавиши стрелок	Перемещается по временным палитрам элементов управления или функций
Enter	Перемещается внутри временной палитры
Esc	Перемещается из временной палитры
Подприборы	
Двойной щелчок на подприборе	Отображает лицевую панель подприбора
Ctrl+двойной щелчок на подприборе	Отображает блок-диаграмму подприбора
Перетаскивание иконки ВП на блок-диаграмму	Размещает этот ВП как подприбор на блок-диаграмме
Shift+перетаскивание иконки ВП на блок-диаграмму	Размещает этот ВП как подприбор на блок-диаграмме с константами, подключенными на входах элементов управления и имеющими значения, отличающиеся от значений по умолчанию
Ctrl+щелчок ПКМ на блок-диаграмме и выбор ВП	Открывает лицевую панель этого ВП из палитры функций
Выполнение	
Ctrl+R	Запускает ВП
Ctrl+.	Останавливает ВП при его выполнении
Ctrl+M	Изменяет режим выполнения или редактирования
Ctrl+кнопка Выполнение	Перекомпилирует текущий ВП
Ctrl+Shift+кнопка Выполнение	Перекомпилирует все ВП в памяти

«Горячая» клавиша	Действие
Ctrl+стрелка вниз	Перемещает фокус в массиве или кластере при выполнении ВП
Ctrl+стрелка вверх	Перемещает фокус за пределами массива или кластера при выполнении ВП
Табуляция	Переключает управление элементами управления или индикаторами в соответствии с их порядковыми номерами во время выполнения ВП
Shift+табуляция	Переключает управление элементами управления или индикаторами в обратном порядке во время выполнения ВП
Подключение	
Ctrl+B	Удаляет все разорванные провода
Esc, щелчок ПКМ или щелчок на терминале	Отменяет начатое соединение
Единичный щелчок на проводе	Выбирает один сегмент провода
Двойной щелчок на проводе	Выбирает ветвь
Тройной щелчок на проводе	Выбирает весь провод
A	Временно отключает автоматическое соединение терминалов
Двойной щелчок (при выполнении соединения)	Закрепляет провод без соединения
Пробел	Включает автоматическое соединение при передвижении объектов
Shift+щелчок	Отменяет последнюю точку, с которой было начато соединение
Ctrl+щелчок на входе функции с двумя входами	Меняет местами два входных провода
Пробел	Переключает направление провода между горизонтальным и вертикальным
Текст	
Двойной щелчок	Выбирает отдельное слово в строке
Тройной щелчок	Выбирает всю строку
Ctrl+стрелка вправо	Перемещается вперед на одно слово в строке
Ctrl+стрелка влево	Перемещается назад на одно слово в строке
Home	Перемещается в начало текущей линии строки
End	Перемещается в конец текущей линии строки
Ctrl+Home	Перемещается в начало всей строки
Ctrl+End	Перемещается в конец всей строки
Shift+Enter	Добавляет новые объекты при вводе объектов в элементы управления или константы типа перечней или кольцевого типа, а также структуры Вариант
Esc	Отменяет текущее редактирование в строке
Ctrl+Enter	Заканчивает ввод текста
Ctrl+=	Увеличивает текущий размер шрифта
Ctrl+-	Уменьшает текущий размер шрифта
Ctrl+0	Отображает диалоговое окно **Шрифт**
Ctrl+1	Отображает шрифт **приложения** (Application font) в диалоговом окне **Шрифт**
Ctrl+2	Отображает шрифт **системы** (System) в диалоговом окне **Шрифт**
Ctrl+3	Отображает шрифт диалога в диалоговом окне **Шрифт**
Ctrl+4	Отображает текущий шрифт в диалоговом окне **Шрифт**

Приложение 3 Алфавитный указатель функций

Функции программирования (Programming)

Application Control VIs and Functions, 192

Call By Reference Node, 195
Call Chain, 197
Call Parent Method, 198
Class Specifier Constant, 197
Close Reference, 197
Get Drag Drop Data, 198
Invoke Node, 196
Open Application Reference, 193
Open VI Reference, 194
Property Node, 195
Quit LabVIEW, 199
Request Deallocation
Static VI Reference, 197
Stop, 198
To More Generic Class, 463
To More Specific Class, 463
VI Server Reference

Array Functions, 109

Array Constant
Array Max & Min, 116
Array Size, 110
Array Subset, 114
Array To Cluster, 121
Array To Matrix, 121
Build Array, 113
Cluster To Array, 121
Decimate 1D Array, 119
Delete From Array, 113
Index Array, 111
Initialize Array, 110
Insert Into Array, 112
Interleave 1D Arrays, 119
Interpolate 1D Array, 117
Matrix To Array, 121
Replace Array Subset, 112
Reshape Array, 120
Reverse 1D Array, 115
Rotate 1D Array, 115
Search 1D Array, 115
Sort 1D Array, 116
Split 1D Array, 116
Threshold 1D Array, 118
Transpose 2D Array, 117

Boolean Functions, 74

Cluster & Variant Functions, 121

Array To Cluster, 121
Build Cluster Array, 123
Bundle By Name, 123
Bundle, 122
Cluster Constant
Cluster To Array, 121
Flattened String To Variant, 125
Index & Bundle Cluster Array, 124
To Variant, 124
Unbundle By Name, 123
Unbundle, 123
Variant To Data, 124
Variant To Flattened String, 125

Comparison Functions, 103

Comparison, 109
Decimal Digit?, 108
Empty Array?, 107
Empty String/Path?, 107
Equal To 0?, 105
Equal?, 104
Greater Or Equal To 0?, 106
Greater Or Equal?, 105
Greater Than 0?, 105
Greater?, 105
Hex Digit?, 108
In Range and Coerce, 107
Less Or Equal To 0?, 106
Less Or Equal?, 105
Less Than 0?, 106
Less?, 105
Lexical Class, 108
Max & Min, 106
Not A Number/Path/Refnum?, 107
Not Equal To 0?, 105
Not Equal?, 105
Octal Digit?, 108
Printable?, 108
Select, 106
White Space?, 108

Dialog & User Interface VIs and Functions, 169

Clear Errors, 175
Color Box Constant
Display Message to User
Error Cluster From Error Code, 177
Find First Error, 176
General Error Handler, 175
Generate Front Panel Activity, 173
Listbox Symbol Ring Constant
Merge Errors, 176
One Button Dialog, 170
Prompt User for Input, 177
Simple Error Handler, 173
Three Button Dialog, 171
Two Button Dialog, 170
Wait For Front Panel Activity, 172

Подпалитры:

Cursor VIs, 188
- Create Cursor From File, 189
- Destroy Cursor, 190
- Set Busy, 188
- Set Cursor, 189
- Unset Busy, 188

Events Functions, 178
- Create User Event, 179
- Destroy User Event, 180
- Event Structure, 181
- Generate User Event, 179
- Register For Events, 180
- Unregister For Events, 179

Help VI and Functions, 191
- Control Help Window
- Control Online Help, 191
- Get Help Window Status
- Open URL in Default Browser, 191

Menu Functions, 183
- Current VI's Menubar, 184
- Delete Menu Items, 185
- Enable Menu Tracking, 185
- Get Menu Item Info, 186
- Get Menu Selection, 184
- Get Menu Short Cut Info, 187
- Insert Menu Items, 185
- Set Menu Item Info, 186

File I/O VIs and Functions, 133

Build Path, 142
Close File, 141
Default Data Directory
Format Into File, 142
Open/Create/Replace File, 140
Read from Binary File, 139
Read From Measurement File, 145
Read From Spreadsheet File, 136
Read from Text File, 138
Scan From File, 141
Strip Path, 143
Write to Binary File, 138
Write To Measurement File, 143
Write To Spreadsheet File, 135
Write to Text File, 137

Подпалитры:
Advanced File Functions, 148
- Array of Strings to Path, 155
- Copy, 152
- Create Folder, 154
- Delete, 152
- Deny Access
- File Dialog, 148
- File/Directory Info, 153
- Flush File, 155
- Get File Position, 150
- Get File Size, 150
- Get Permissions, 151
- Get Type and Creator, 151
- Get Volume Info, 153
- List Folder, 154
- Move, 151
- Path to Array of Strings, 155
- Path To String, 155
- Path Type, 156
- Refnum to Path, 156
- Set File Position, 150
- Set File Size, 150
- Set Permissions
- Set Type and Creator, 151
- String To Path, 155

Datalog Functions, 156
- Close File
- Get Datalog Position
- Get Number of Records
- Open/Create/Replace Datalog, 156
- Read Datalog, 157
- Set Datalog Position
- Set Number of Records
- Write Datalog, 157

File Constants, 169
Storage VIs, 159
- Close Data Storage, 161
- Delete Data, 164
- Get Properties, 163
- Merge Queries, 163
- Open Data Storage, 161
- Read Data, 162
- Set Properties, 163
- Write Data, 161

Zip VIs, 158
- Add File to Zip, 159
- Close Zip File, 159
- New Zip File, 159

Graphics & Sound VIs, 215

Beep, 222

Подпалитры:
Graphics Formats VIs, 216
- Flatten Pixmap, 216
- Read BMP File, 219
- Read JPEG File, 218
- Read PNG File, 219
- Unflatten Pixmap, 217
- Write BMP File, 219
- Write JPEG File, 217
- Write PNG File, 218

Picture Functions VIs
- Color Box Constant
- Color to RGB, 221
- Create Mask

- Draw Arc
- Draw Circle by Radius
- Draw Flattened Pixmap, 220
- Draw Grayed Out Rect
- Draw Line
- Draw Multiple Lines
- Draw Oval
- Draw Point
- Draw Rect
- Draw Round Rect
- Draw Text at Point
- Draw Text in Rect
- Draw Unflattened Pixmap, 219
- Empty Picture
- Get Image Subset, 220
- Get Text Rect
- Hilite Color, 221
- Move Pen
- Picture to Pixmap, 220
- RGB to Color, 221

Sound VIs, 222
- Sound File Close
- Sound File Info
- Sound File Open
- Sound File Read Simple, 228
- Sound File Read
- Sound File Write Simple, 227
- Sound File Write

Numeric Functions, 57

Подпалитры:
Complex Functions, 66
Conversion VIs and Functions, 61
Data Manipulation Functions, 66
- Flatten To String, 67
- Join Numbers, 71
- Logical Shift, 70
- Mantissa & Exponent, 69
- Rotate Left With Carry, 69
- Rotate Right With Carry, 69
- Rotate, 70
- Split Number, 70
- Swap Bytes, 71
- Swap Words, 71
- Type Cast, 67
- Unflatten From String, 68

Math & Scientific Constants, 72

String Functions, 77

Array To Spreadsheet String, 86
Build Text, 102
Carriage Return Constant, 88
Concatenate Strings, 78
Empty String Constant, 88
End of Line Constant, 88
Format Date/Time String, 129
Format Into String, 85
Line Feed Constant, 88
Match Pattern, 80
Match Regular Expression, 82
Replace Substring, 79
Scan From String, 83
Search and Replace String, 79
Space Constant, 88
Spreadsheet String To Array, 86
String Constant, 88
String Length, 77
String Subset, 78
Tab Constant, 88
To Lower Case, 87
To Upper Case, 87
Trim Whitespace, 87

Подпалитры:
Additional String VIs and Functions, 88
- Append True/False String, 92
- Index String Array, 92
- Match First String, 89
- Match True/False String, 89
- Pick Line, 88
- Reverse String, 92
- Rotate String, 92
- Scan String For Tokens, 90
- Search and Replace Pattern, 91

Structures, 36

Synchronization VIs and Functions, 200

Timing VIs and Functions, 127

Waveform VIs and Functions, 371

Математические функции (Mathematics), 231

Differential Equations VIs, 278

Elementary & Special Functions

Подпалитры:
Exponential Functions, 64
Trigonometric Functions, 63

Fitting VIs, 245

Подпалитра:
Advanced Curve Fitting VIs

Linear Algebra VIs, 231

Integ & Diff VIs, 275

Interp & Extrap VIs, 269

Optimization VIs, 285

Probability and Statistics VIs, 257

Подпалитры:

Функции обработки сигналов (Signal Processing VIs), 302

Filters VIs, 341

Подпалитры:

Signal Generation VIs, 302

Signal Operation VIs, 312

Spectral Analysis, 333

Transforms, 325

Windows, 363

Функции обмена данными (Data Communication VIs and Functions), 423

DataSocket VI and Functions, 430

Protocols VIs and Functions, 436

Функции поддержки взаимодействия приложений (Connectivity VIs and Functions), 454

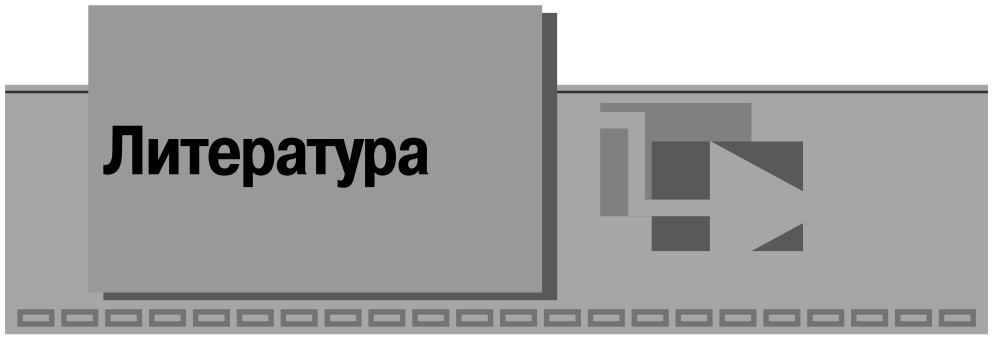
.NET Functions, 460

ActiveX Functions, 454

Input Device Control VIs, 476

Libraries & Executables VIs and Functions, 464

Функции поддержки ввода/вывода данных и стандартных интерфейсов (Measurement I/O, Instrument I/O), 479



1. Суранов А. Я. LabVIEW 7: справочник по функциям. – М.: ДМК Пресс, 2005. – 512 с.
2. Кривлёв А. В. Основы компьютерной математики с использованием системы MATLAB. – М.: Лекс-Книга, 2005. – 496 с.: ил.
3. MATLAB 7 / Е. Н. Алексеев, О. В. Чеснокова. – М.: НТ Пресс, 2006. – 464 с.: ил.
4. Новгородцев А. Б. Расчет электрических цепей в MATLAB: учебный курс. – СПб.: Питер, 2004. – 250 с.: ил.
5. Цифровая обработка сигналов / А. Б. Сергиенко. – СПб.: Питер, 2003. – 608 с.: ил.
6. Гутников В. С. Фильтрация измерительных сигналов. – Л.: Энергоатомиздат, 1990. – 192 с.
7. Отнес Р., Эноксон Л. Прикладной анализ временных рядов: Основные методы. – М.: Мир, 1982. – 428 с.
8. Рабинер Л., Гоулд Б. Теория и применение цифровой обработки сигналов / пер. с англ.; под. ред. Ю. И. Александрова. – М.: Мир, 1978.
9. Дэвид Чеппел Технологии ActiveX и OLE / пер. с англ. – М.: Издательский отдел «Русская редакция», 1997. – 320 с.: ил.
10. Дайл Роджерсон Основы COM / пер. с англ. – М.: Издательский отдел «Русская редакция», 1997. – 375 с.: ил.
11. Дэвид С. Плат Знакомство с Microsoft .NET / пер. с англ. – М.: Издательско-торговый дом «Русская редакция», 2001. – 240 с.: ил.
12. Суранов А. Я., Белых С. В. Микропроцессорный регистратор одномерных изображений на базе фотодиодного приемника // ПТЭ. – 2003. – № 6. – С. 140–142.
13. Таненбаум Э. Современные операционные системы. 2-е изд. – СПб.: Питер, 2002. – 1040 с.: ил.

Книги издательства «ДМК Пресс» можно заказать в торгово-издательском холдинге «АЛЬЯНС-КНИГА» наложенным платежом, выслав открытку или письмо по почтовому адресу: **123242, Москва, а/я 20** или по электронному адресу: **post@abook.ru**.

При оформлении заказа следует указать адрес (полностью), по которому должны быть высланы книги; фамилию, имя и отчество получателя. Желательно также указать свой телефон и электронный адрес.

Эти книги вы можете заказать и в Internet-магазине: **www.abook.ru**.

Оптовые закупки: тел. **(095) 258-91-94, 258-91-95**; электронный адрес **abook@abook.ru**.

Суранов Александр Яковлевич

LabVIEW 8
Справочник по функциям

Главный редактор *Мовчан Д. А.*
dm@dmk-press.ru
Корректор *Синяева Г. И.*
Верстка *Чаннова А. А.*
Дизайн обложки *Мовчан А. Г.*

Подписано в печать __.10.2007. Формат 70×100 $^{1}/_{16}$.
Гарнитура «Петербург». Печать офсетная.
Усл. печ. л. __. Тираж _____ экз.
№

Электронный адрес издательства: www.dmkpress.ru