Суранов А. Я.

LabVIEW 7:
СПРАВОЧНИК ПО ФУНКЦИЯМ

Москва, 2005

УДК 621.38
ББК 32.973.26-108.2
Б 28

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

ISBN 5-94074-207-6

В книге приведено описание функциональных элементов среды проектирования виртуальных приборов LabVIEW 7 Express. Описание выполнения функций сопровождается примерами их использования. Для большинства новых элементов LabVIEW 7 Express - Экспресс ВП приведены окна конфигурирования с переводом их содержимого. В справочнике большое внимание уделено функциям обработки сигналов и математическим функциям, функциям управления приложениями и функциям коммуникаций, функциям управления платами ввода/вывода данных и обмена данными по стандартным интерфейсам. Справочник может быть полезен студентам технических вузов, инженерам, а также 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

Приложение 1

Приложение 2

Приложение 3

Приложение 4

Приложение 5

Приложение 6

Приложение 7

Приложение 8

Введение

Среда графического программирования LabVIEW получает все большее распространение в промышленности и образовании, при проведении научных исследований и выполнении проектных работ. Этому способствуют ее несомненные преимущества – высокая производительность при разработке программ, называемых виртуальными приборами (ВП) и широкий набор функциональных возможностей языка и среды программирования. Однако для того, чтобы пользователь мог получить выигрыш от реализации указанных преимуществ, он должен в полной мере овладеть технологией программирования в среде LabVIEW и изучить ее функциональные возможности.

В последнее время изложению этих вопросов в отечественной литературе был посвящен ряд книг [1, 2, 3]. Книга [1] была достаточно полезна на момент издания, однако быстро устарела вследствие смены версий LabVIEW. Наиболее удачной следует признать книгу [2], в которой рассмотрены основные элементы среды LabVIEW 6i. В первой части книги излагаются основы программирования на языке G в LabVIEW, во второй части описаны дополнительные возможности среды. Естественно, что большой охват вопросов не позволил детально рассмотреть возможности функциональных элементов среды LabVIEW. Такое детальное описание, необходимое опытному пользователю, изложено в файле Помощь (LabVIEW Help), в руководстве пользователя (LabVIEW User Manual) и в большом числе заметок (LabVIEW Application Notes). Все эти документы требуют перевода и правильной интерпретации терминов и определений, что затрудняет применение изложенной в них информации. В книге [3] предпринята попытка изложения части справочного материала, однако эта часть невелика по объему, а используемая терминология существенно отличается от терминологии, принятой в книге [2].

Помимо этого, используемая в настоящее время версия LabVIEW 7 Express и анонсированная версия LabVIEW 7.1 имеют ряд дополнительных функциональных возможностей, не отраженных в перечисленных книгах. (Перечень новых возможностей LabVIEW 7.0 и 7.1 приведен в приложении 1 справочника.)

Все вышеизложенное явилось причиной подготовки данного справочника. Основное внимание в нем, как это следует из названия, уделено рассмотрению функций среды LabVIEW 7 Express, а описание других элементов среды, приведенное в главе 1, минимизировано. В главе 2 рассмотрены базовые функции

LabVIEW, к которым отнесены числовые, логические, строковые функции, функции сравнения, работы с массивами и кластерами, функции времени и диалога и функции ввода/вывода файлов.

Глава 3 посвящена функциям генерации, ввода и обработки данных LabVIEW. В первом разделе главы рассмотрены функции генерации и обработки данных в спектральной и временной области из подпалитры **Обработка сигнала** (Signal Processing), а во втором разделе – функции из подпалитры **Математика** (Mathematics). В третьем разделе данной главы описаны функции, находящиеся в подпалитрах **Расширенные** (Advanced) и **Графики и звук** (Graphics & Sound).

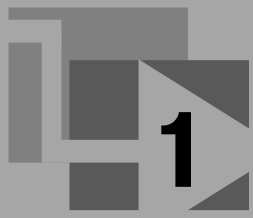
В главе 4 описаны функции интерфейса виртуальных приборов и приложений в LabVIEW. В первом разделе этой главы рассмотрены функции сервера ВП, размещенные в подпалитре **Управление приложением** (Application Control) и позволяющие программно управлять или получать информацию о выполнении приложения (LabVIEW) или виртуального прибора, а также программно управлять или получать информацию о состоянии элементов лицевой панели. Во втором разделе описаны функции связи приложений Windows и функции протоколов связи, находящиеся в подпалитре **Коммуникация** (Communication).

В главе 5 рассмотрены функции LabVIEW, обеспечивающие интерфейс с платами и каналами ввода/вывода. В первом разделе этой главы описаны функции генерации, обработки и измерения параметров осциллограмм, с помощью которых производится сбор данных в LabVIEW. В последующих разделах описаны функции DAQmx управления платами сбора данных и функции интерфейса с каналами ввода/вывода данных (GPIB, Serial, USB).

В приложение вынесена информация о новых возможностях LabVIEW 7.0 и 7.1, о синтаксисе узла Формула, о «горячих» клавишах, о методах и свойствах приложения и ВП, а также об их характеристиках.

Разделы, посвященные определенной группе функций, начинаются с кратких пояснений, после чего приводится описание функций, оформленное в виде набора таблиц, и в заключение могут быть приведены примеры использования 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, в большинстве случаев приводятся два изображения функции с подключенными элементами управления и индикаторами. При этом первое 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.

Организация среды LabVIEW и технология программирования



1.1. Панели и палитры LabVIEW

При запуске LabVIEW и выборе из меню стартового диалогового окна строки меню **Новый ⇒ Пустой ВП** (New ⇒ Blank VI) открываются два окна, содержащие лицевую панель (рис. 1.1а) и панель блок-диаграммы (рис. 1.1б) виртуального прибора (ВП). В правом верхнем углу каждой панели находится иконка, наложенная на соединительную панель ВП (последняя показана на лицевой панели). В верхней части каждого окна размещена традиционная для приложений Windows полоса главного меню с одинаковыми для обоих окон пунктами File, Edit, Operate, Tools, Browse, 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.

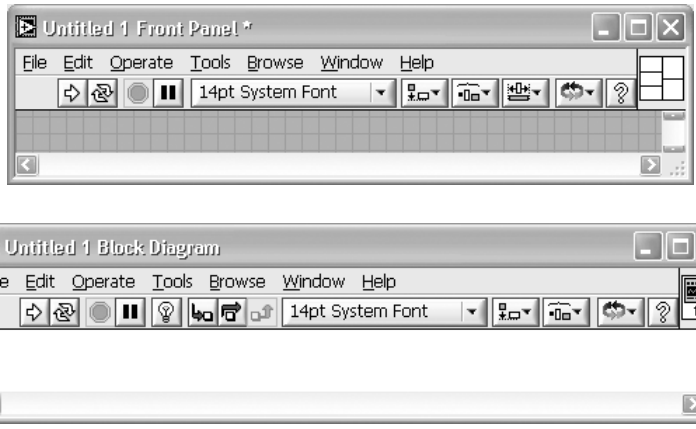


Рис. 1.1. Вид лицевой панели (а) и панели блок-диаграммы (б) виртуального прибора

Свободное пространство каждой панели образует рабочую область, снабженную горизонтальной и вертикальной полосами прокрутки. При построении ВП в рабочей области лицевой панели визуально размещаются элементы управления 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 «Т». 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 3.

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 обычно с помощью разделов меню **Окно** (Window). Так, в частности, при активном окне лицевой панели с помощью строки **Показать палитру элементов управления** (Show Controls Palette) меню **Окно** (Window) на эту панель можно вывести палитру элементов, а при активном окне панели блок-диаграммы на нее можно вывести палитру функций, пользуясь строкой **Показать палитру функций** (Show Functions Palette) этого же меню. Для вывода палитры инструментов необходимо использовать строку **Показать палитру инструментов** (Show Tools Palette) меню **Окно** (Window).

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 помощь 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, Windows, Help, и пункты, являющиеся характерными для LabVIEW,

такие как Operate, Tools, Browse. Ниже в таблице приведено краткое описание функций пунктов главного меню.

Файл (File)	Используется для открытия новых или существующих ВП, закрытия, сохранения и печати ВП
Правка (Edit)	Применяется для редактирования панелей ВП, поиска объектов и удаления неисправных проводников с блок-диаграммы
Управление (Operate)	Реализует запуск и прерывание выполнения ВП, установку значений по умолчанию, соединение с удаленной панелью и изменение других опций ВП
Инструменты (Tools)	Используется для поиска ВП на диске, управления библиотеками ВП, управления соединением с удаленными ВП, управления публикацией панелей ВП в Web, конфигурации ВП и для выполнения ряда прикладных функций
Просмотр (Browse)	Используется для просмотра иерархии ВП
Окно (Window)	Используется для отображения окон LabVIEW и его палитр
Справка (Help)	Служит для получения информации об элементах и о функциях LabVIEW

Ниже показаны полосы инструментальных панелей на лицевой панели (рис. 1.2а) и на панели блок-диаграммы (рис. 1.2б).

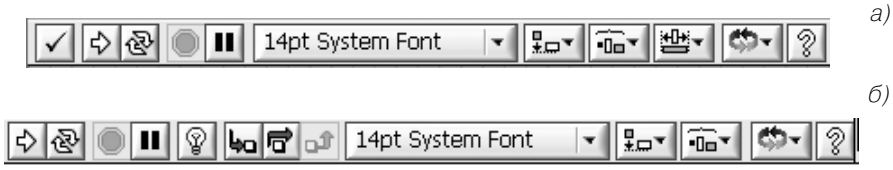


Рис. 1.2. Инструментальные панели на лицевой панели (а) и на панели блок-диаграммы (б)

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** (Abort Execution) вызывает остановку выполняющегося ВП
	Кнопка **Пауза** (Pause) временно останавливает выполнение ВП

Следующие четыре кнопки 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


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 редактировать 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 фиксировать положения 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. 1.3). При выборе определенного инструмента 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. Автома-

тический 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 инструментом **Перемещение** и **Соединение** на блок-диаграмме или между инструментом **Перемещение** и **Управление** на лицевой панели достаточно нажать пробел.

Рис. 1.3

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 7.0 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 **Установка отладочных индикаторов** (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

Палитра элементов лицевой панели по умолчанию появляется в виде палитры Экспресс, содержащей наиболее часто используемые элементы. Для перехода к более привычному по предыдущим версиям LabVIEW виду палитры необходимо вызвать диалоговое окно **Опции** (Options) с помощью выбора одноименного раздела пункта **Инструменты** (Tools) главного меню или нажав кнопку **Option**  на палитре элементов, а затем выбрать в верхнем раскрывающемся меню раздел **Палитры элементов управления или функций** (Controls/Functions Palettes) и выбрать в строке **Вид палитр** (Palette View) диалогового окна пункт **Advanced**.

В этом виде 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 и кластер (Array & 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

Лист 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, используемые для установки параметров плат ввода-вывода данных

Элементы диалога (Dialog Controls). Подпалитра содержит набор элементов, поддерживающих диалоговые функции

Классические элементы (Classic Controls). Подпалитра содержит палитры элементов, выполненных в стиле предыдущих версий LabVIEW

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

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

Выбор элемента управления (Select a Control). Подпалитра служит для выбора элемента управления, отсутствующего в палитре

Элементы пользователя (User Controls). Подпалитра служит для выбора элементов управления, созданных пользователем и помещенных им в папку LabVIEW 7.0\user.lib

Экспресс (Express). Подпалитра содержит палитры с наиболее часто применяемыми элементами

Палитра функций панели блок-диаграммы по умолчанию также появляется в виде Экспресс и содержит ряд подпалитр Экспресс-ВП. Переход к палитре с более широким набором функций на верхнем уровне производится так же, как и описанный выше переход для палитры лицевой панели. В этом виде палитра функций панели блок-диаграммы содержит следующие основные подпалитры:

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

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

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** (Cluster) включает функции формирования кластеров и массивов кластеров, выделения элементов из кластеров, а также кластерную константу. Более подробно функции работы с кластерами описаны в разделе 2.1.5

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 **Время и диалоги** (Time & Dialog) позволяют задавать или определять временные интервалы или текущее время, формировать сообщения об ошибках и контролировать активность лицевой панели. Указанные функции рассмотрены в разделе 2.1.6

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 2.1.7

Подпалитра **Измерения NI** (NI Measurements) содержит наборы функций работы с платами ввода/вывода данных, ввода изображений, управления двигателями

Подпалитра **Осциллограмма** (Waveform) включает разделы **Аналоговая осциллограмма** (Analog Waveform), **Цифровая осциллограмма** (Digital Waveform), **Файловый ввод/вывод осциллограмм** (Waveform File I/O), **Измерения параметров осциллограмм** (Waveform Measurements) и **Генерация осциллограмм** (Waveform Generation). Функции подпалитры Осциллограмма рассмотрены в разделе 5.1

Подпалитра **Анализ** (Analyze) включает разделы **Операции с осциллограммами** и **Измерения параметров осциллограмм**, содержание которых идентично одноименным разделам функции **Осциллограмма**, а также разделы **Подготовка осциллограмм** (Waveform Conditioning), **Контроль осциллограмм** (Waveform Monitoring), подразделы **Поточечная обработка** (Point By Point), **Обработка сигналов** (Signal Processing) и **Математика** (Mathematics)

Подпалитра **Связь с приборами** (Instrument I/O) включает разделы **Канал общего пользования** (GPIB или IEEE 488), **Последовательный интерфейс** (Serial) и библиотеку интерфейсов VISA (Virtual Instrument Software Architecture)

Функции подпалитры **Управление приложением** (Application Control) позволяют программно управлять приложением (LabVIEW) или ВП на локальном или удаленном компьютере, изменять вид меню, останавливать ВП или выгружать LabVIEW. Функции подпалитры рассмотрены в разделе 4.1

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 3.3.5. Функции работы со звуковой картой и звуковыми файлами рассмотрены в разделе 3.3.6

Подпалитра **Коммуникация** (Communication) содержит функции связи между приложениями Windows с использованием технологии .NET и ActiveX, а также функции сетевого обмена данными по протоколам TCP/IP, UDP, Data Socket, SMPT E-mail, IrDA и функцию System Exec. Функции коммуникаций рассмотрены в разделе 4.2

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

Подпалитра **Расширенные** (Advanced) включает узел **Вызов библиотечной функции** (Call Library Function Node), узел **Интерфейсный код** (Code Interface Node (CIN)) и подразделы **Управление устройством ввода** (Input Device Control), **Манипуляция данными** (Data Manipulation), **Синхронизация** (Synchronization), **Ввод/вывод данных через порты** (Port I/O) и **Доступ к реестру Windows** (Windows Registry Access Vis). Методика разработки библиотечных функций рассмотрена в разделе 3.3.1, функции манипуляции данными – в разделе 3.3.2, функции синхронизации – в разделе 3.3.3 и функции доступа к реестру – в разделе 3.3.4

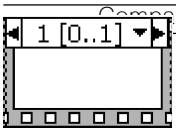
Подпалитра **Оформление** (Decorations) содержит такие графические элементы, как **Свободная метка** (Free Label), **Линия** (Line), **Ровный кадр** (Flat Frame)

Подпалитра **Выбрать ВП** (Select a VI) позволяет выбрать ВП, не содержащийся в палитре функций, с помощью диалогового окна **Choose the VI to Open**

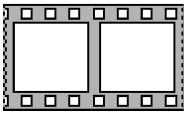
Подпалитра **Библиотеки пользователя** (User Libraries) открывает доступ к ВП из библиотек пользователя, расположенных в папке LabVIEW 7.0\user.lib

Подпалитра **Экспресс** (Express) открывает доступ к подпалитрам Экспресс-ВП

В составе подпалитры **Структуры** (Structures) входят следующие элементы: **Вход** (Input), **Анализ сигнала** (Signal Analysis), **Выход** (Output), **Управление выполнением** (Execution Control), **Арифметика и сравнение** (Arithmetic & Comparison) и **Обработка сигнала** (Signal Manipulation)

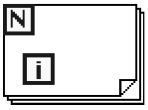


Структура **Стековая последовательности** (Stacked Sequence Structure) позволяет управлять последовательностью выполнения отдельных фрагментов кода путем их размещения в кадрах данной структуры



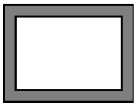
Структура **Открытая последовательности** (Flat Sequence Structure) отличается от предыдущей возможностью передачи данных между кадрами без вспомогательной переменной и возможностью вывода данных из любого кадра структуры

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**

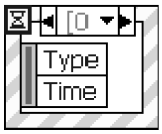


Цикл с фиксированным числом итераций (For Loop) осуществляет заданное число итераций при выполнении кода внутри данной структуры

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



Узел **Формула** (Formula Node) позволяет включить фрагмент кода в текстовом представлении



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

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

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

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

Из перечисленных выше подпалитр функций дополнительного рассмотрения заслуживает подпалитра **Анализ** (Analyze) как наиболее представительная.

Наиболее содержательными разделами этой подпалитры являются разделы **Обработка сигнала** (Signal Processing) и **Математика** (Mathematics).

Раздел функций **Обработка сигнала** включает подразделы **Генерация сигнала** (Signal Generation), **Обработка сигнала во временной области** (Time Domain) и **Обработка сигнала в частотной области** (Frequency Domain), **Фильтры** (Filters) и **Окна** (Windows). Перечисленные подразделы функций рассмотрены соответственно в разделах 3.1.1–3.1.5 справочника.

Раздел функций **Математика** включает подразделы **Формула** (Formula), **Расчет одномерных и двумерных массивов** (1D and 2D Evaluation), **Численные методы и решение дифференциальных уравнений** (Calculus) (3.2.1), **Вероятность и статистика** (Probability and Statistics) (3.2.2), **Сглаживание данных** (Curve Fitting) (3.2.3), **Линейная алгебра** (Linear Algebra) **и операции с массивами** (Array Operations) (3.2.4), **Оптимизация** (Optimization) **и поиск нулей** (Zeroes) (3.2.5), а также **Специальные и числовые функции** (Special and Numeric Functions). После названий подразделов функций указаны номера разделов справочника, в которых эти функции рассмотрены более подробно.

Помимо разработки лицевой панели и блок-диаграммы самого ВП, важное значение имеет 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 производится 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. 1.4 показана иконка и соединительная панель ВП **Записать в файл табличного формата** (Write To Spreadsheet File), находящегося в подпалитре **Файловый ввод/вывод**.

Рис. 4

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
	Наполненный прямоугольник	Рисует прямоугольник, окаймленный цветом переднего плана и наполненный цветом заднего плана
	Выбор	Выбирает область иконки для перемещения, копирования или других изменений
	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

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 индикации из палитры элементов данной панели. Аналогично при разработке блок-диаграммы производится 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 подобных 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.

В LabVIEW 7.0 введен еще один способ идентификации терминала, а именно отображение 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 

. Од-

нако представление в виде иконки приводит к потере пространства блок-диаграммы, что не всегда удобно.

Перечень и вид терминалов элементов ввода данных LabVIEW 7.0 с характеристикой типа, цвета и значения по умолчанию приведены в следующей таблице.

Элемент по управлению	Элемент индикации	Тип данных	Цвет	Значение умолча-
		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 двухбайтовый со знаком	Синий	0
		Числовой четырехбайтовый со знаком	Синий	0
		Числовой байтовый без знака	Синий	0
		Числовой двухбайтовый без знака	Синий	0
		Числовой четырехбайтовый без знака	Синий	0
		Нумерованный	Синий	–
		<64,64>-битовая отметка времени		Дата и время (локальные)
		Логический	Зеленый	FALSE
		Строковый	Розовый	Пустая строка
		Массив – заключает тип данных	Различный	–

		в квадратные скобки и принимает цвет этого типа данных		
Элемент по управлению	**Элемент индикации**	**Тип данных**	**Цвет**	**Значение умолча-**
		Кластер – заключает разные типы данных. Кластерный тип данных имеет коричневый цвет, если все элементы кластера числовые, и розовый, если элементы разных типов	Коричневый или розовый	–
		Путь	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	–
		Ссылка	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 содержат объекты, текст, кисти и другие графические элементы	Синий	–

Настройка параметров объектов лицевой панели (элементов управления и индикаторов) и терминалов блок-диаграммы производится с помощью контекстного меню опций и команд. 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 (слева) и его терминала (рис. 1.5).

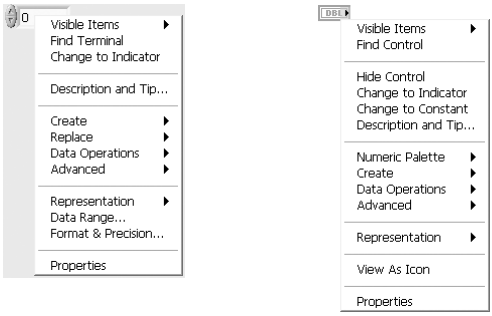


Рис. 1.5

Краткие пояснения к пунктам контекстного меню приведены в следующей таблице:

Видимые объекты (Visible Items)	Позволяет показать или скрыть 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
Описание и подсказка (Description and Tip)	Позволяет описать объект и сделать подсказку к нему
Создать (Create)	Позволяет создать локальную переменную, узел свойств или ссылку
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, устанавливать соединение DataSocket
Дополнительно (Advanced)	Позволяет производить тонкую настройку элементов управления и индикаторов
Свойства (Properties)	Позволяет выбрать диалоговое окно свойств объекта

Следующие пункты данного контекстного меню являются специфическими для числовых элементов:

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

Представление данных в LabVIEW включает следующие типы:

– повышенной точности

– двойной точности

– 32-битовый

– целочисленный 32-битовый со знаком

– целочисленный 16-битовый со знаком

– целочисленный восьмибитовый со знаком

– целочисленный 32-битовый без знака

– целочисленный 16-битовый без знака

– целочисленный восьмибитовый без знака

– комплексный повышенной точности

– комплексный двойной точности

– комплексный 32-битовый

Для логических элементов управления специфическим 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
	**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

Изображение	Режим	Действие
	**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**	Значение изменяется при щелчке и возвращается в исходное состояние после однократного считывания виртуальным прибором нового значения или после отпускания кнопки мыши

В меню настроек строковых элементов управления и индикаторов специфическим является раздел, определяющий вид отображения содержимого этих элементов. Интерпретация строк 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 восьмибитового 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 отображения 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 элемента управления строками к его терминалу на блок-диаграмме только после щелчка мышью за пределами окна строки 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 программы. 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 проводники** (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 справка** (Context Help), вызываемого с помощью опции **Справка ⇒ Показать контекстную справку** (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 справки. 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 данная подпрограмма будет доступна 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 остановки** (breakpoints).

На рис. 1.6 показан вид блок-диаграммы ВП **Использование вставляемых пробников** (Using Supplied Probes) из набора примеров NI Example Finder LabVIEW.

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

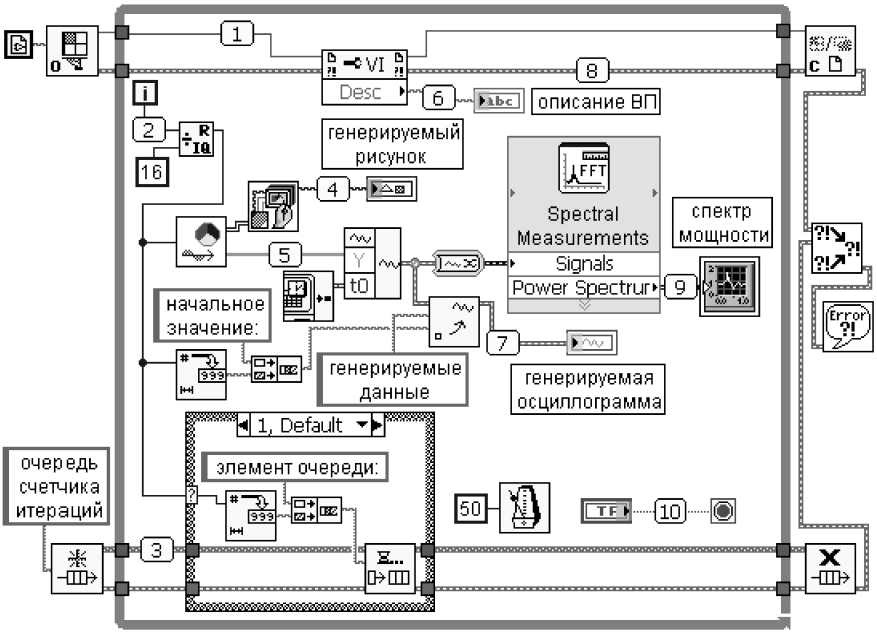


Рис. 1.6. Вид блок-диаграммы ВП ***Использование вставляемых пробников*** *(Using Supplied Probes)*

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 останов 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 останов по условию.
6. **Пробник строки с условием** (Conditional String Probe) – показывает строковые данные и обеспечивает останов 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 останов 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 останов по условию.

1.3. Структуры, массивы и графические индикаторы среды LabVIEW

В составе палитры **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 входят также структуры: узел **Формула** (Formula Node), структура **Событие** (Event Structure), **Локальная переменная** (Local), **Глобальная переменная** (Global) и узел **Обратная связь** (Feedback Node).

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. Данная опция может быть отключена для этой структуры путем снятия отметки строки Auto Grow в контекстном меню структуры или для всего приложения путем снятия отметки строки **Установить структуры с автоматическим расширением** (Place structures with Auto Grow enabled) диалогового окна **Опции** (Options) ⇒ **Новое и измененное в LabVIEW 7.0** (New and Changed in LabVIEW 7.0).

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. 1.7).

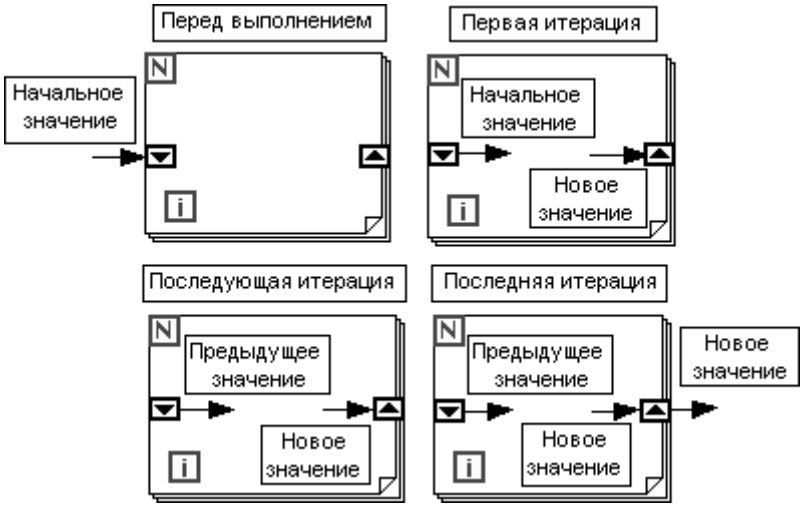


Рис. 1.7. Принцип работы сдвиговых регистров

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 с помощью 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 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 происходит 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. 1.10). Более подробно ин-

дексирование входных и выходных данных структуры цикла будет рассмотрено ниже 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. 1.8 и 1.9.

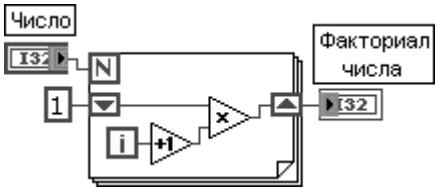


Рис. 1. 8. Блок-диаграмма ВП вычисления факториала числа

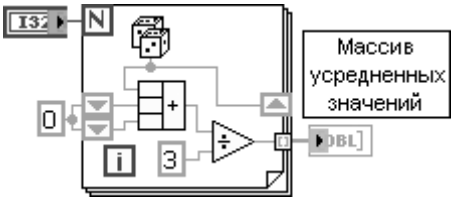


Рис. 1.9. Блок-диаграмма ВП усреднения последовательности случайных чисел

Помимо сдвигового регистра, в структуре цикла может устанавливаться структура узел **Обратная связь** (Feedback Node) (рис. 1.10). 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. 1.10.

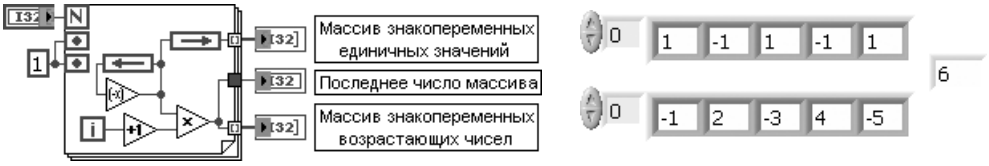


Рис. 1.10. Блок-диаграмма и лицевая панель ВП расчета знакопеременных чисел

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 тер-

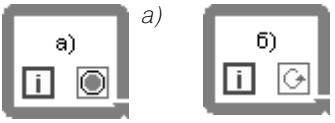


*Рис. 1.11. Вид структуры **Цикл по условию** с различными режимами*

миналом условия по умолчанию приведен на рис. 1.11а. Код программы, размещенный в структуре, выполняется до подачи на терминал условия логической переменной ИСТИНА (TRUE) в структуре на рис. 1.11а или ЛОЖЬ (FALSE) в структуре на рис. 1.11б. Изменение варианта прекращения выполнения производится с помощью строк **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 **Индексирование выключено** (Disable Indexing).

Структура **Вариант** (Case Structure) аналогична операторам case или if-then-else в текстовых языках программирования. По умолчанию структура Вариант является логической и имеет два варианта – ИСТИНА (TRUE) и ЛОЖЬ (FALSE), выбираемые 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 производится 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 данных, 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 последовательности.

В LabVIEW 7.0 структура **Последовательность** наряду с известной по предыдущим версиям структурой **Стековая последовательность** (Stacked Sequence Structure) дополнена представлением в виде структуры **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. 1.12).

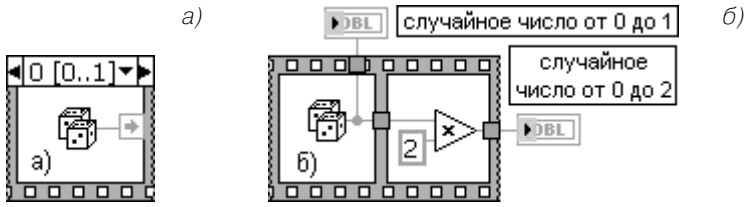


Рис. 1.12. Вид структуры Стековая последовательность (а) и структуры Открытая последовательность (б)

Так же как и в структуре Вариант, ввод и вывод данных производится с помощью входных и выходных терминалов, при этом данные из входного терминала доступны во всех кадрах. Для передачи данных внутри структуры Стековая последовательность (рис. 1.12а) используется **терминал локальной переменной** (Sequence Local). Терминал локальной переменной создается с помощью строки **Добавить локальную переменную** (Add Sequence Local) контекстного меню границы 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. 1.12б) 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 при выполнении сложных расчетов. Формулы записываются с использованием функций и операторов, перечисленных в двух последующих 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 вход** (Add Input) и **Добавить выход** (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
atanh(x)	Inverse Hyperbolic Tangent		Гиперболический арктангенс **x**, рад
ceil(x)	Round to +Infinity		Округление до большего целого
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
getman(x)	Mantissa & Exponent (Data Manipulation)		Мантисса числа **x** в случае его представления числом по основанию 2

Функция узла	Имя аналогичной функции языка G	Числовая функция	Описание функции
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**
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**
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**
sqrt(x)	Square Root		Квадратный корень **x**

Функция узла	Имя аналогичной функции языка G	Числовая функция	Описание функции
tan(x)	Tangent		Тангенс **x**
tanh(x)	Hyperbolic Tangent		Гиперболический тангенс **x**

Синтаксис узла Формула приведен в приложении 2.

Список операторов узла Формула (по приоритету):

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. 1.13.

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 4.1.

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 зна-

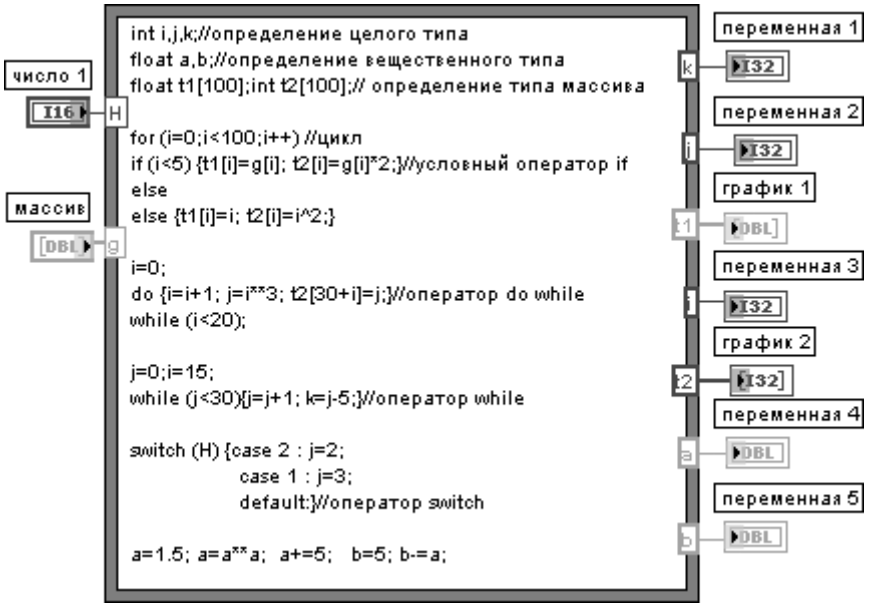


Рис. 1.13. Примеры записи операторов в узле Формула

чения 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 диаграммы. 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. 1.14а). 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. 1.14 показан вид массивов различного типа на лицевой панели и соответствующих терминалов на блок-диаграмме: массив числовых элементов управления (б), массив логических индикаторов (в), массив элементов управления строкового типа (г) и двумерный массив индикаторов целых чисел (д).

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 формируются 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. 1.15). В этом режиме терминал вывода данных представляет пустотелый двойной квадрат. Формирование массива сопровождается изменением толщины провода с данными после его выхода из структуры. Выключе-

ние 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 каждом повторении цикла называется **автоиндексированием** (auto-indexing).

При вводе массива в структуру цикла терминал ввода данных по умолчанию также находится в состоянии **Включить индексирование** (рис. 1.15). В данном режиме терминал передает в цикл по одному элементу в каждую итерацию. В этом случае структура автоматически определяет размерность 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 точку входа перевести в состояние **Отключить индексирование**, массив будет вводиться в цикл целиком (рис. 1.15). При этом, естественно, способность автоиндексирования теряется и значение числа циклов должно быть задано.

На рис. 1.15 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 из внутреннего и внешнего циклов. Увеличение размерности массива, созданного оператором на лицевой панели или в блок-диаграмме, может быть выполнено с помощью

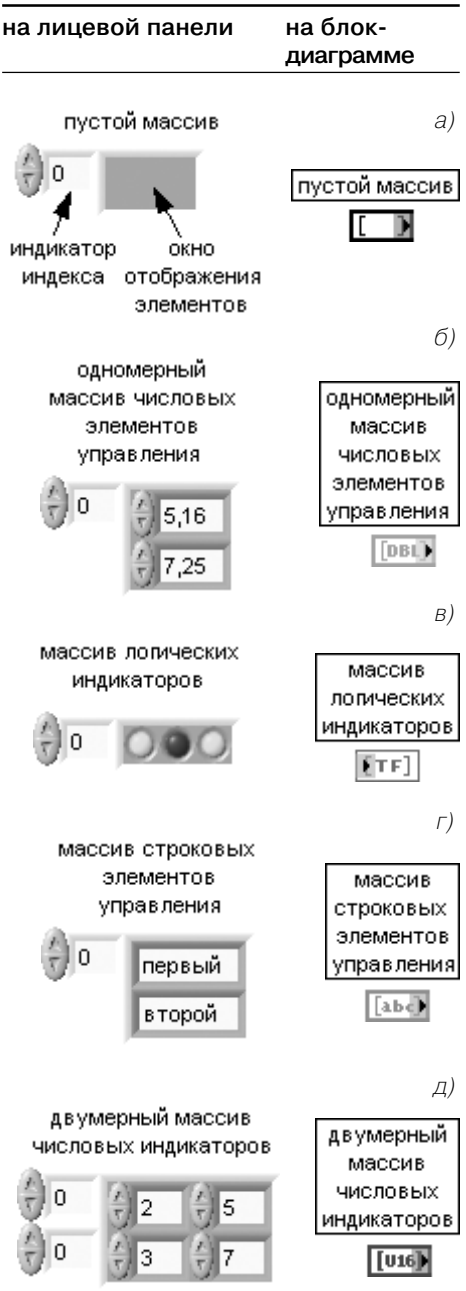


Рис. 1.14. Вид массивов на лицевой панели и соответствующих терминалов на блок-диаграмме

строки **Добавить размерность** (Add Dimension) контекстного меню элемента управления / отображения **индекса массива**. Такое же действие может быть выполнено и инструментом перемещения.

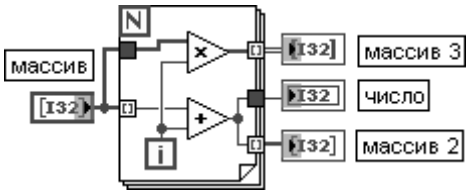


Рис. 1.15. Варианты ввода и вывода массивов в структуре цикла с фиксированным числом итераций

Рис. 1.16. 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).

В состав палитры Графики входят следующие графические индикаторы (рис. 1.16).

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. 1.17). 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** (Data Operations) ⇒ **Очистить развертку** (Clear Chart) контекстного меню графика

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. 1.17).

Рис. 1.17. Блок-диаграмма ВП контроля температуры с индикаторами Развертка температуры и График температуры

Двухкоординатный график (XY Graph) – графический индикатор, позволяющий отображать функциональные зависимости . 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 использовать функцию **Сформировать массив** (Build Array) для формирования массива кластеров (рис. 1.18).

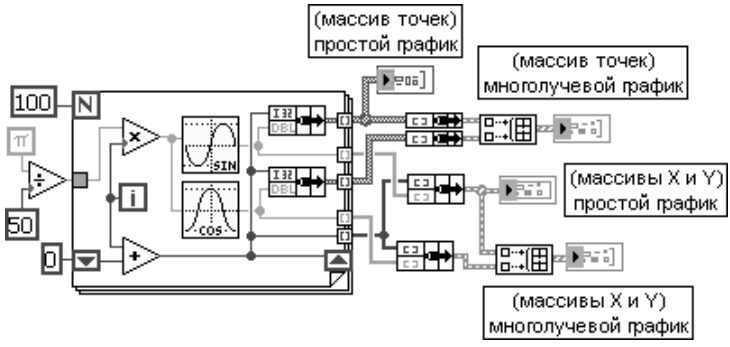


Рис. 1.18. Блок-диаграмма ВП XY Graph из набора примеров NI Example Finder

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 3 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)

Графический индикатор **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

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. 1.19): **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. 1.19. Вид графика осциллограммы с панелями и атрибутами

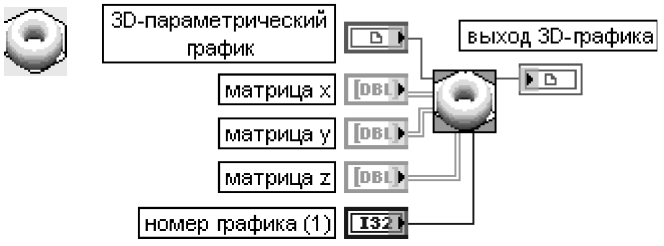
Меню **Видимые элементы** графического индикатора **Развертка осциллограммы** отличается тем, что на месте строки **Панель редактирования курсора** находится строка **Цифровой индикатор**.

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.

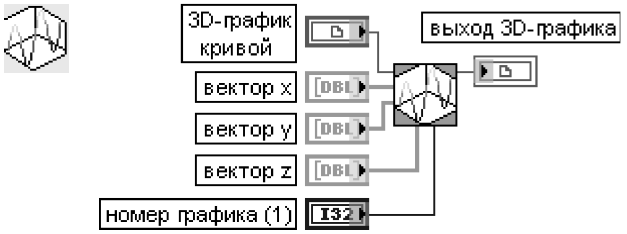
Панель редактирования шкалы позволяет фиксировать или сбрасывать в исходное состояние масштаб по осям, установленный пользователем с помощью одного из инструментов, входящих в состав **палитры элементов управления графиком**. Помимо этого, она позволяет настраивать параметры осей: формат, точность, характер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

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



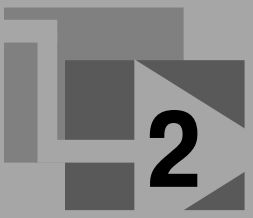
Графический индикатор **3D-параметрический график** (3D Parametric Graph) отображает данные с помощью трех двумерных массивов, характеризующих проекции графика на плоскости **x**, **y** и **z**



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.

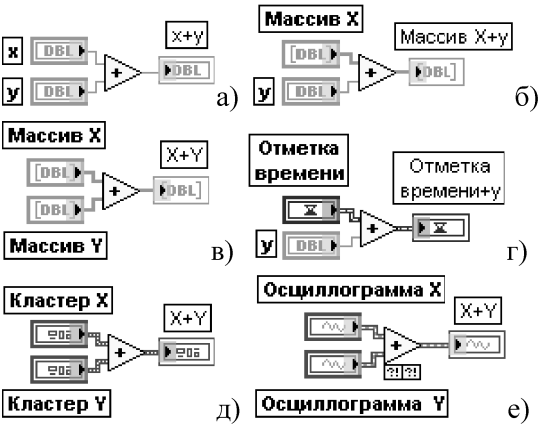
Базовые функции LabVIEW



2.1. Числовые функции

Числовые функции используются для выполнения арифметических, тригонометрических, логарифмических и комплексных операций с числовыми данными и для преобразования типов числовых данных. На рис. 2.1а показан вид основной палитры числовых функций и ряда дополнительных подпалитр: **Преобразование** (Conversion) (рис. 2.1б), **Тригонометрические функции** (Trigonometric) (рис. 2.1в), **Логарифмические функции** (Logarithmic) (рис.2.1г), **Комплексные функции** (Complex) (рис.2.1д) и **Дополнительные числовые константы** (Additional Numeric Constants) (рис. 2.1е).

Таблица основной палитры числовых функций:

Add  	**Сложить** Функция рассчитывает сумму входов. Если к 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. 2.2е, 2.2ж). Не допускается суммирование двух значений меток времени. Функция является полиморфной, поэтому входы могут быть числовыми скалярами, массивами или кластерами чисел, массивами кластеров чисел, отметками времени и т. д.

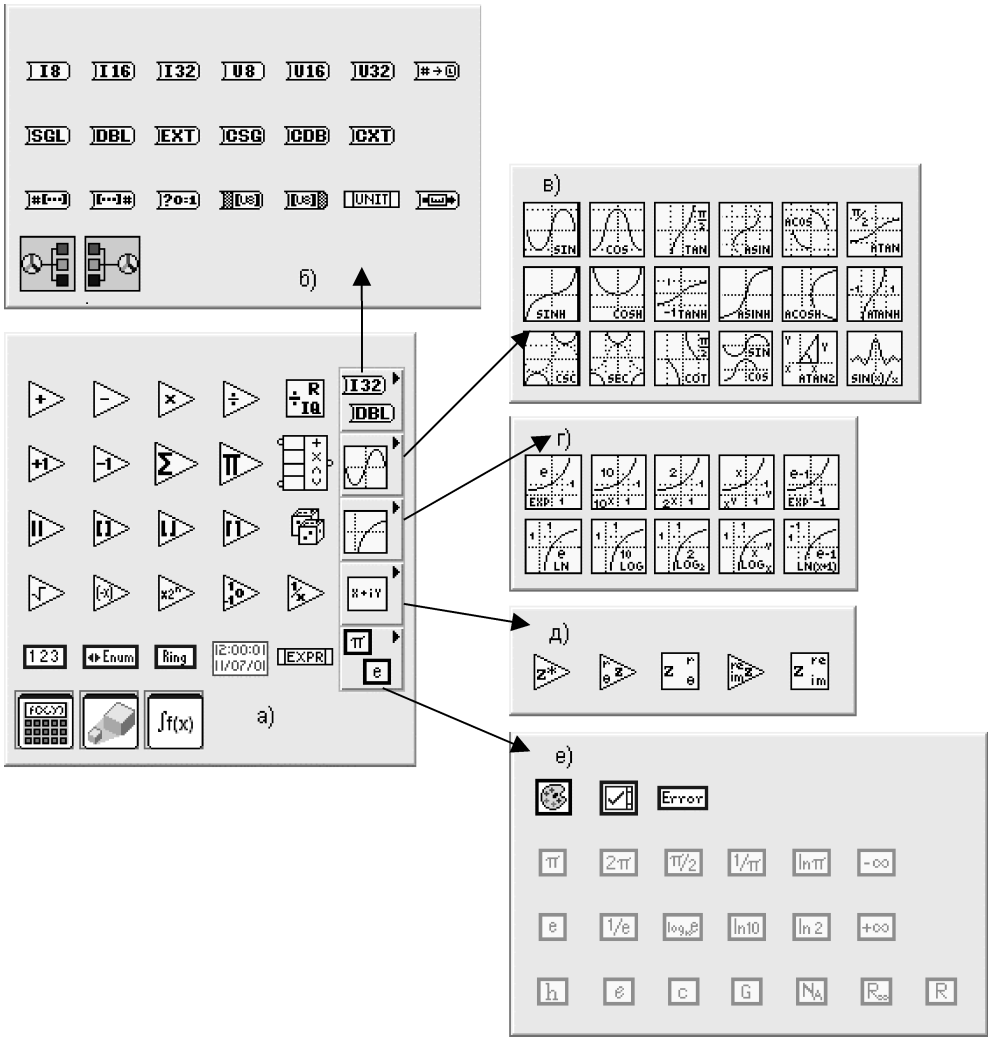


Рис. 2.1. Вид основной палитры (а)
и дополнительных подпалитр (б–е) числовых функций

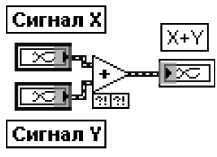
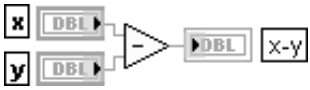


Рис. 2.2. Варианты подключения функции ***Сложить***

На рисунках показаны варианты суммирования двух скаляров (рис. 2.2а), скаляра и массива (рис. 2.2б), двух массивов (рис. 2.2в), отметки времени и скаляра (рис. 2.2г), двух кластеров (рис. 2.2д), двух осциллограмм (рис. 2.2е) и двух наборов данных динамического типа (рис. 2.2ж)

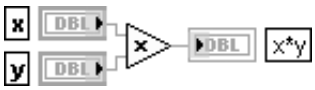
Subtract



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

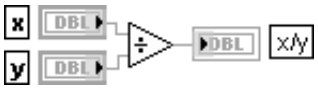
Multiply



Умножить

Функция возвращает произведение входов

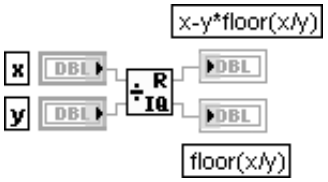
Divide



Разделить

Функция возвращает частное от деления значений на входах

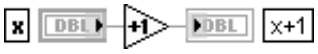
Quotient & Remainder



Частное и остаток

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

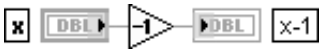
Increment



Инкремент

Функция возвращает значение входа, увеличенное на 1

Decrement



Декремент

Функция возвращает значение входа, уменьшенное на 1

Add Array Elements

Сложить элементы массива

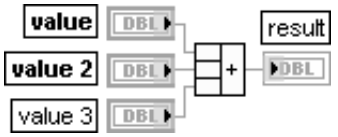
Функция возвращает сумму всех элементов входного **числового массива** (numeric array)

Multiply Array Elements

Перемножить элементы массива

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

Compound Arithmetic

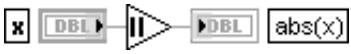


Составная арифметика

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

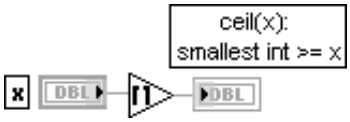
Функция округляет входное значение до ближайшего целого. Если входное значение находится посередине между двумя целыми (например, 1,5 или 2,5), то функция возвращает ближайшее четное значение (2)

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

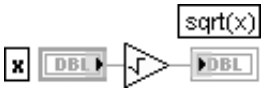
Функция округляет входное значение до большего целого. Например, если входное значение равно 3,1, то результат будет равен 4. Если на входе –3,1, то результатом будет –3

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

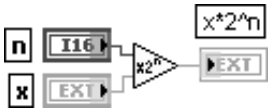
Функция рассчитывает квадратный корень входного значения. Если входное значение отрицательное, то возвращается значение NaN

Negate

Отрицание

Функция изменяет знак входной величины на противоположный

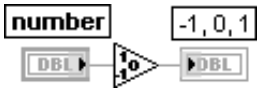
Scale By Power Of 2



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



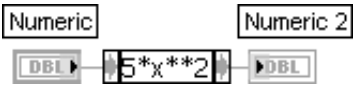
Знак

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

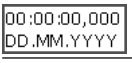


Узел Выражение
Узел **Выражение** (Expression Node) используется для расчета выражений или уравнений, которые содержат единственную переменную. В формулах могут использоваться следующие встроенные функции: abs, acos, acosh, asin, asinh, atan, atanh, ceil, cos, cosh, cot, csc, exp, expm1, floor, getexp, 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).

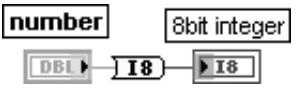
Кольцевая константа (Ring Constant)



Константа отметки времени (Time Stamp Constant)

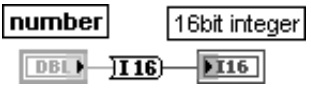
Таблица функций подпалитры **Преобразование** (Conversion):

To Byte Integer



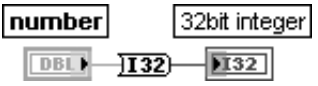
В байтовое целое число
Функция преобразует входное число в восьмибитовое целое в диапазоне от –128 до 127

To Word Integer



В целое слово
Функция преобразует входное число в 16-битовое целое в диапазоне от –32,768 до 32,767

To Long Integer

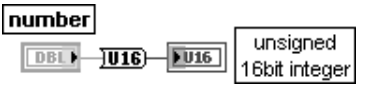


В длинное целое число
Функция преобразует входное число в 32-битовое целое в диапазоне от –(2^31) до (2^31)–1

To Unsigned Byte Integer

В байтовое целое без знака
Функция преобразует входное число в восьмибитовое целое число без знака в диапазоне от 0 до 255

To Unsigned Word Integer



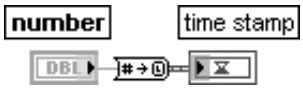
В целое слово без знака
Функция преобразует входное число в 16-битовое целое число без знака в диапазоне от 0 до 65 535

To Unsigned Long Integer

В длинное целое число без знака

Функция преобразует входное число в 16-битовое целое число без знака в диапазоне от 0 до (2^32)–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

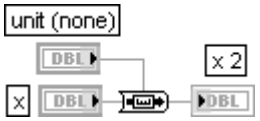
Функция преобразует входное число в комплексное число с плавающей запятой с расширенной точностью

Convert Unit

Преобразовать размерность

Функция преобразует физическое число (размерное) в безразмерное число и наоборот

Cast Unit Bases

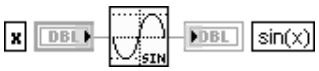


Изменить базовые единицы

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

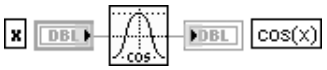
Таблица функций подпалитры **Тригонометрические функции** (Trigonometric):

Sine



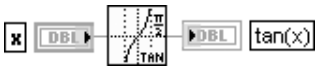
Синус
Функция рассчитывает синус входного значения **x** (рад)

Cosine



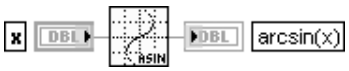
Косинус
Функция рассчитывает косинус входного значения **x** (рад)

Tangent



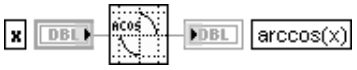
Тангенс
Функция рассчитывает тангенс входного значения **x** (рад)

Inverse Sine



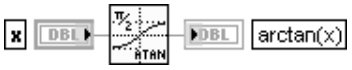
Арксинус
Функция рассчитывает значение арксинуса **x**. Результат выражен в радианах

Inverse Cosine



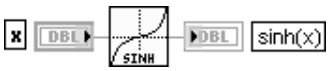
Арккосинус
Функция рассчитывает значение арккосинуса **x**. Результат выражен в радианах

Inverse Tangent



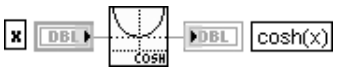
Арктангенс
Функция рассчитывает значение арктангенса **x**. Результат выражен в радианах

Hyperbolic Sine



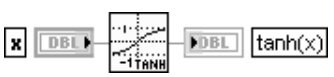
Гиперболический синус
Функция рассчитывает гиперболический синус **x**

Hyperbolic Cosine



Гиперболический косинус
Функция рассчитывает гиперболический косинус **x**

Hyperbolic Tangent



Гиперболический тангенс
Функция рассчитывает гиперболический тангенс **x**

Inverse Hyperbolic Sine

Гиперболический арксинус
Функция рассчитывает гиперболический арксинус **x**

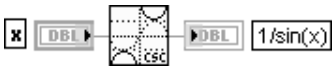
Inverse Hyperbolic Cosine

Гиперболический арккосинус
Функция рассчитывает гиперболический арккосинус **x**

Inverse Hyperbolic Tangent

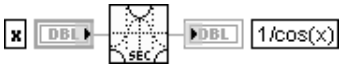
Гиперболический арктангенс
Функция рассчитывает гиперболический арктангенс **x**

Cosecant



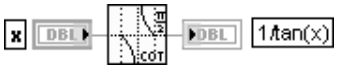
Косеканс
Функция рассчитывает косеканс входного значения **x** (рад)

Secant



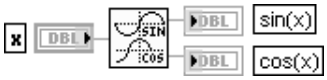
Секанс
Функция рассчитывает секанс входного значения **x** (рад)

Cotangent



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

Sine & Cosine



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

Inverse Tangent (2 Input)

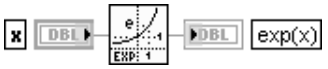
Арктангенс (2 входа)
Функция рассчитывает арктангенс отношения **y**/**x**, выраженного в радианах. Эта функция может рассчитывать арктангенс для углов в любом квадранте плоскости **x-y**, в то время как функция **Inverse Tangent** рассчитывает арктангенс только в двух квадрантах

Sinc

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

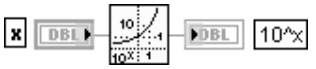
Таблица функций подпалитры **Логарифмические функции** (Logarithmic):

Exponential



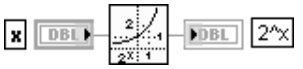
Экспонента
Функция рассчитывает значение числа **e**, возведенного в степень **x**

Power Of 10



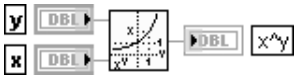
Степень числа 10
Функция рассчитывает значение числа 10, возведенного в степень **x**

Power Of 2



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

Power Of X



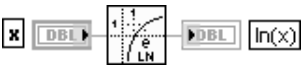
Степень числа X
Функция рассчитывает значение числа **x**, возведенного в степень **y**

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

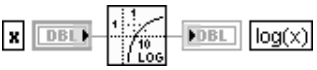
Natural Logarithm



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

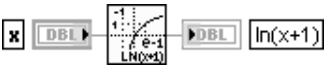
Функция рассчитывает логарифм числа **x** по основанию 2

Logarithm Base X

Логарифм по основанию X

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

Natural Logarithm (Arg +1)

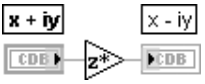


Функция Ln(x+1)

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**

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

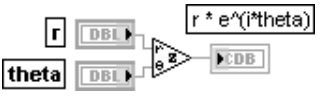
Complex Conjugate



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.

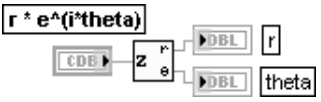
Polar To Complex



Преобразование значений в полярных координатах в комплексное значение

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

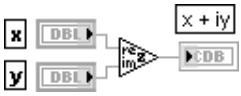
Complex To Polar



Преобразование комплексного значения в значения полярных координат

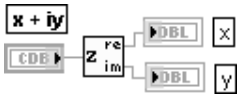
Функция разделяет комплексное значение на два значения полярных координат

Re/Im To Complex



Преобразование значений в декартовых координатах в комплексное значение
Функция формирует комплексное значение из двух значений, заданных в декартовых координатах

Complex To Re/Im



Преобразование комплексного значения в значения декартовых координат
Функция разделяет комплексное значение на два значения декартовых координат

Перечень констант подпалитры **Дополнительные числовые константы** (Additional Numeric 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)

Отрицательная бесконечность

Положительная бесконечность

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)

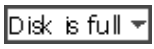
Элементарный заряд электрона (1,602176462e–19) Кл

Постоянная Ридберга (1,0973731568549e7) 1/м

Константа цвета (Color Box Constant)

Молярная газовая постоянная (8,314472) Дж/(моль · K)

Кольцевая константа символов окна списков (Listbox Symbol Ring Constant) используется для передачи символов в пункты элемента управления **Окно списков** (Listbox). Передача символа производится с помощью подключения константы к входу свойства **пункт символов** (Item Symbols) **узла свойств** (Property Node) элемента управления **Окно списков**



Кольцевая константа ошибки (Error Ring Constant) содержит список ошибок, возникающих при использовании памяти, работе в сети, печати и файловом вводе-выводе. Константа позволяет заменить код ошибки, формируемый функцией, на ее словесное описание

В состав палитры числовых функций входят Экспресс-ВП **Формула** (Formula), **Масштабирование и отображение** (Scaling and Mapping) и **Математическая обработка во временной области** (Time Domain Math).

Формула (Formula)

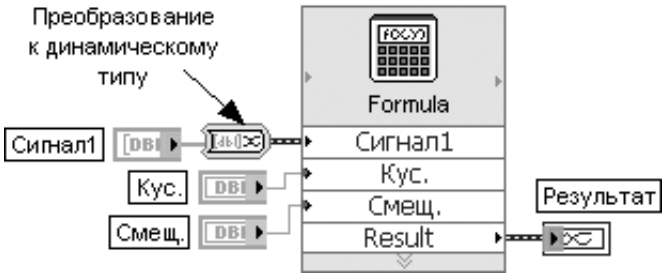


Рис. 2.3. Блок-диаграмма варианта подключения Экспресс-ВП

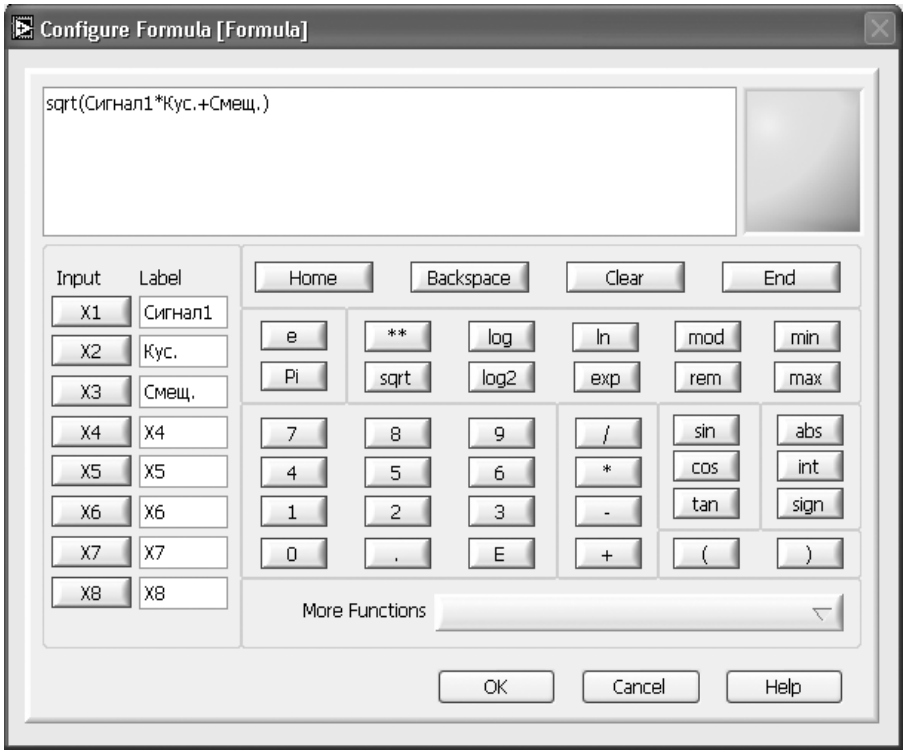


*Рис. 2.4. Вид диалогового окна конфигурирования Экспресс-ВП **Формула** (Formula)*

Экспресс-ВП **Формула** (Formula) (рис. 2.4) 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. 2.3).
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)

а) б)

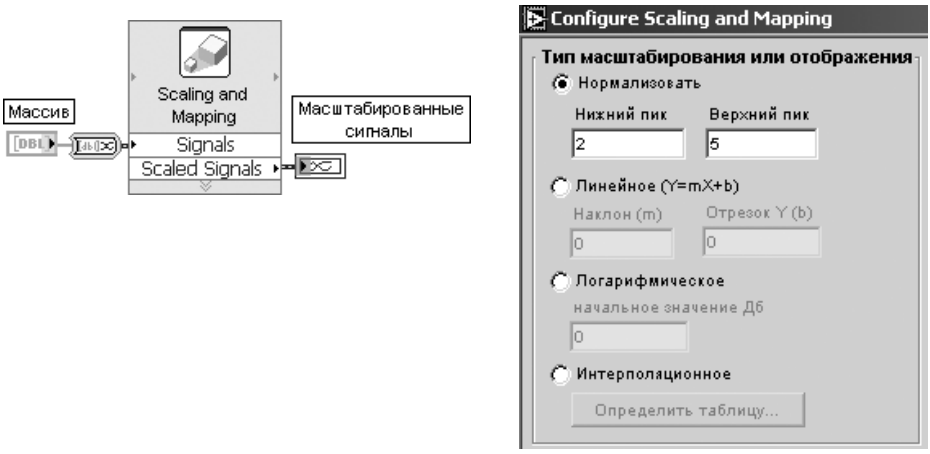


Рис. 2.5. Блок-диаграмма (а) и вид диалогового окна конфигурирования (б) Экспресс-ВП

Экспресс-ВП **Масштабирование и отображение** (Scaling and Mapping) (рис. 2.5) 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)

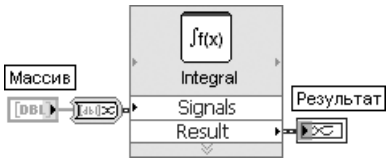


Рис. 2.6. Блок-диаграмма подключения Экспресс-ВП

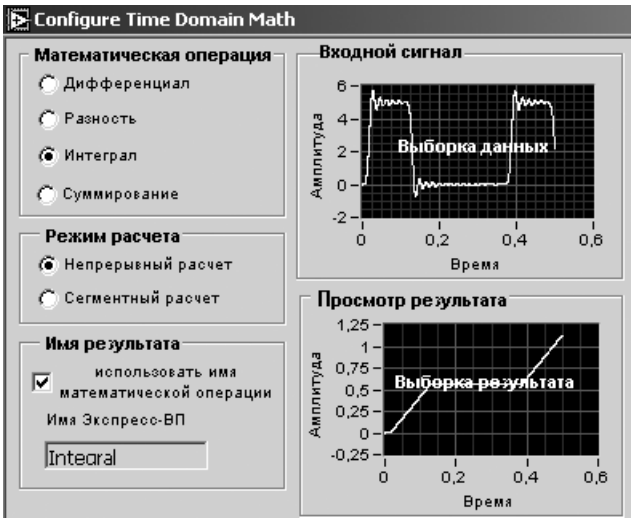


Рис. 2.7. Вид диалогового окна конфигурирования Экспресс-ВП ***Математическая обработка во временной области*** *(Time Domain Math)*

Экспресс-ВП **Математическая обработка во временной области** (Time Domain Math) (рис. 2.6) выполняет одну из операций математической обработки сигналов во временной области.

Набор математических операций включает следующие (рис. 2.7): **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)).

2.2. Логические функции

Логические функции (рис. 2.8) используются для выполнения логических операций над значениями как простых логических величин, так и массивов этих 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 долж-

ны иметь логические или числовые значения, причем эти значения могут быть скалярами, массивами или кластерами. При обработке числовых значений перечисленные функции выполняют побитовую обработку чисел.

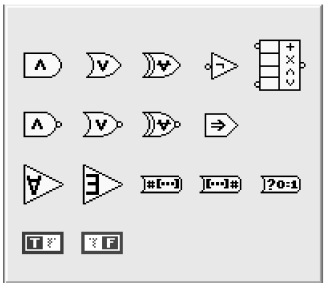


Рис. 2.8. Вид палитры логических функций

Ниже в таблице приведены пояснения к набору логических функций.

And

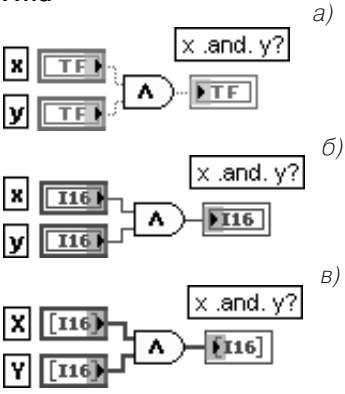


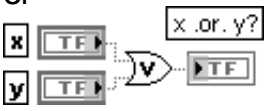
*Рис. 2.9 Варианты подключения функции **И***

Логическая функция И

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

На рис. 2.9а приведены примеры использования данной функции для обработки скаляров логического типа, на рис. 2.9б – скаляров числового типа, на рис. 2.9в – массивов числовых значений

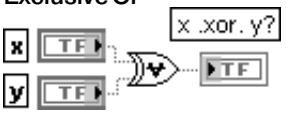
Or



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

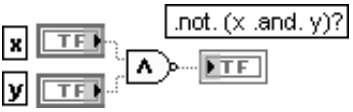
Возвращает значение ЛОЖЬ (0) только при подаче на оба входа значений ЛОЖЬ (0) или значений ИСТИНА (1), иначе возвращает значение ИСТИНА (1)

Not

Логическая функция НЕ

Возвращает значение ЛОЖЬ (0) при подаче на вход значения ИСТИНА (1) и наоборот

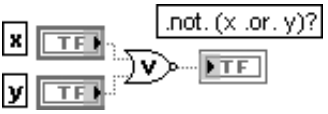
Not A nd



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

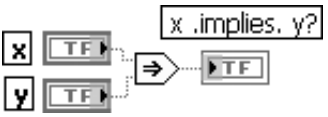
Возвращает значение ИСТИНА (1) только при подаче на оба входа значений ЛОЖЬ (0), иначе возвращает значение ЛОЖЬ (0)

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

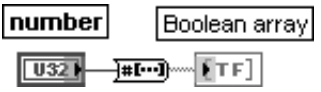
Возвращает значение ИСТИНА, если все элементы **Логического массива** (Boolean array) имеют значение ИСТИНА. Иначе возвращает ЛОЖЬ. Функция воспринимает массивы любого размера

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

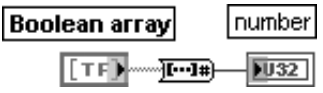
Number To Boolean Array



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

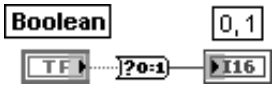


Логический массив в число

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

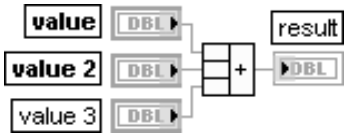
Boolean To (0,1)



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.

2.3. 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 на диске также требует их строковой организации.

На рис. 2.10а приведен вид основной палитры строковых функций с одним Экспресс-ВП и вид ряда дополнительных подпалитр: **Дополнительные строко-**

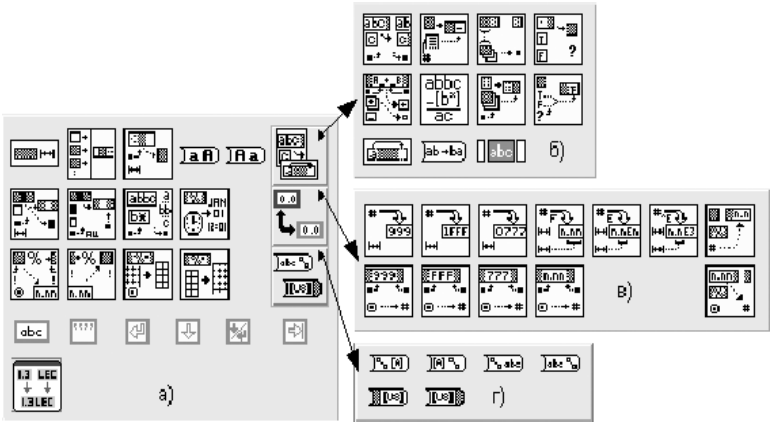


Рис. 2.10. Вид основной палитры (а) и дополнительных подпалитр (б–г) строковых функций

вые функции (Additional String Functions) (рис. 2.10б), **Функции взаимного преобразования строк и чисел** (String/Number Conversion) (рис. 2.10в) и **Функции взаимного преобразования строк, массивов байтов и путей** (String/Array/Path Conversion) (рис. 2.10г).

Ниже в таблицах 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

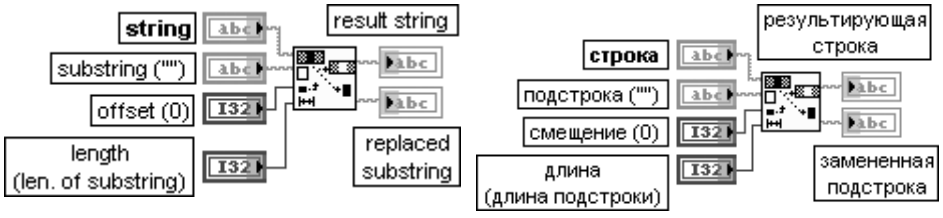
Concatenate Strings **Объединить строки**

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**

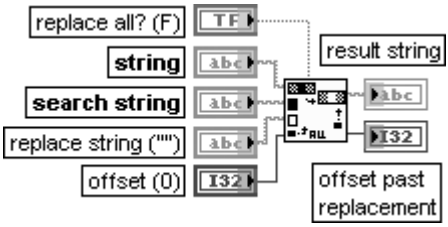
Функция возвращает часть входной **строки** (string), начинающуюся со **смещения** (offset) и содержащую число символов, заданное на входе **длина** (length). Начальный адрес первого символа в строке равен 0.
Выход **подстрока** (substring) является пустым, если смещение больше длины строки или если **длина** меньше или равна 0. Если **длина** больше или равна длине **строки** минус **смещение** (offset), то на выходе **подстрока** выводится остаток строки, начинающийся со смещения

Replace Substring **Заменить подстрокой**

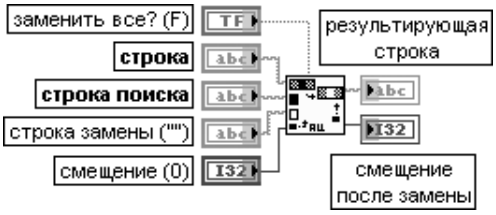


Функция удаляет часть **строки** (string), количество символов которой задано на входе **длина** (length), начиная со **смещения** (offset), и заменяет удаленную часть строки содержимым **подстроки** (substring).
Если **длина** равна 0, то функция вставляет **подстроку** начиная со **смещения**. Если **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

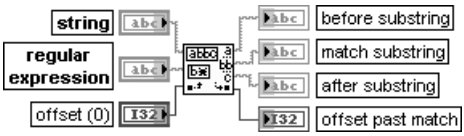


Найти и заменить строку

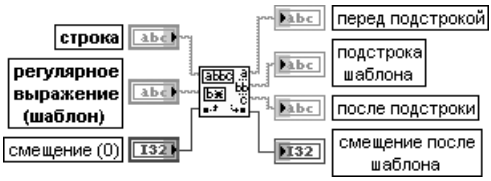


Функция заменяет одну или все образцы подстроки другой подстрокой. Данная функция проверяет **строку** (string) на наличие образцов **строки поиска** (search string), начиная с позиции, заданной величиной **смещения** (offset). Функция заменяет первый встретившийся образец искомой строки на **строку 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 регулярным выражением** (before substring), **подстрока шаблона** (match substring), **подстрока после регулярного выражения** (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 backspace** (удаление символа) **\s space** (пробел) **\f form feed** (конец стоки) **\r carriage return** (возврат каретки) **\n newline** (новая строка) **\t tab** (табуляция) **\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.
[ ]	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

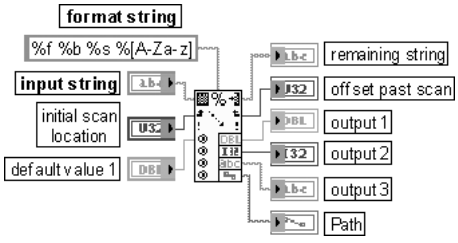
		символов цифр из диапазона от **0** до **9**.
	^	Определяет совпадение любых отображаемых символов, включая пробел и табуляцию, за исключением символов или диапазона символов, находящихся в скобках. Например, [**^0-9**] определяет совпадение любых символов, за исключением символов цифр из диапазона от **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**.

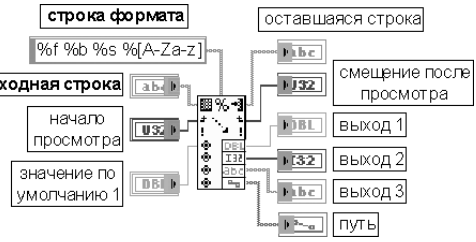
Регулярное выражение	**Обнаруженные символы**
VOLTS	VOLTS
[**Vv**][**Oo**][**Ll**][**Tt**][**Ss**]	Все варианты слова **volts**, содержащие буквы верхнего и нижнего регистров, такие как **VOLTS**, **Volts**, **volts**
[_+ –]	Пробел, знак плюса или знак минуса
[**0-9**]+	Последовательность одной или более 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 круглых скобок

Ниже на рис. 2.11 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. 2.12). При этом числа могут иметь любой из следующих форматов: 123 1.23 .123 0.123 –1.23.

Scan From String



Просмотр строки



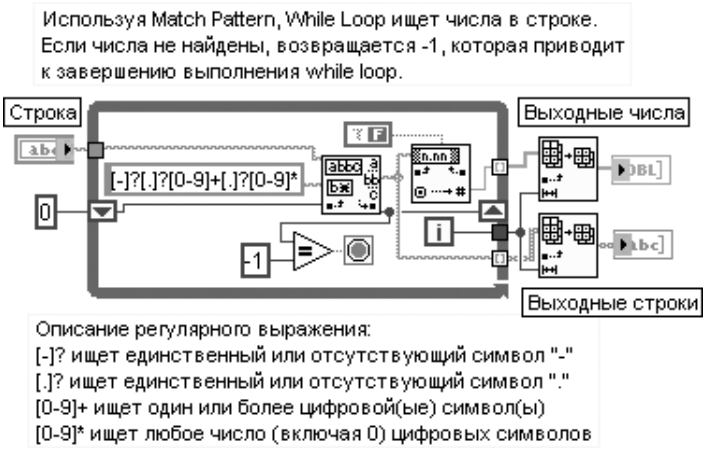


Рис. 2.11. Блок-диаграмма ВП Извлечь числа

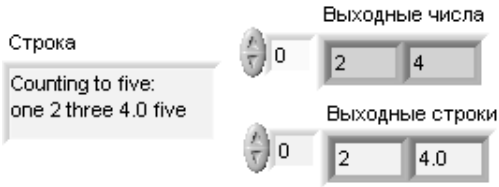


Рис. 2.12. Вид лицевой панели ВП

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 **SI** нотации. При такой нотации 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.

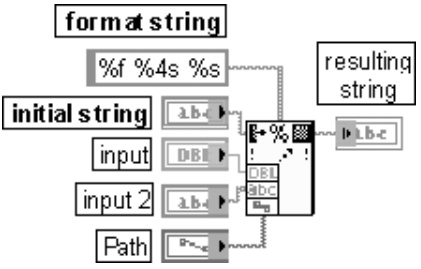
T и **t** могут использоваться только в элементах управления, константах и индикаторах

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.

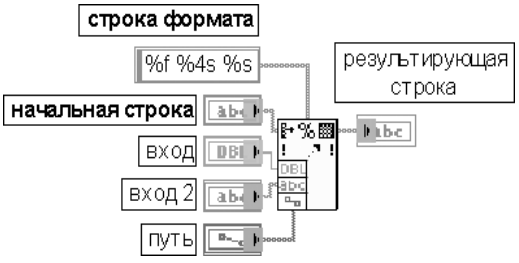
Выход **смещение после просмотра** (offset past scan) отображает смещение во **входной строке** (input string) после выполнения сканирования.
Выходы **выход 1…n** (output 1...n) определяют выходные параметры. Каждый выход может быть строкой, путем, типом перечисления или любым числовым типом. С этой функцией не могут использоваться массивы и кластеры.
В таблице приведены примеры использования функции **Просмотр строки**.

входная строка	строка формата	по умолчанию	выходы	оставшаяся строка
abc, xyz	%3s, %s%f%2d	–	abc	00
12,3+56i 7200		–	xyz	
		0,00 + 0,00 i	12,30 + 56,00 i	
		–	72	
Q+1.27E-3 tail	Q%f t	–	1,27E-3	ail
0123456789	%3d%3d	–	12	6789
			345	
X:9,860 Z:3,450	X:%fY:%f	100(I32)	10	Z: 3,450
		100,00(DBL)	100,00	
set49,4,2	set%d	–	49	,4,2
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 LabWIEW	Smith
			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 ранее при анализе функции **Просмотр строки**.

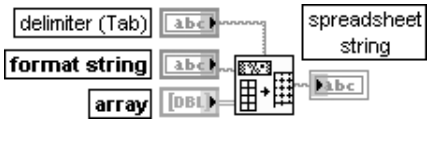
Далее в таблице приведены только элементы, не рассмотренные ранее.

Синтаксический элемент	Описание
– (тире)	Указывает на выравнивание по ширине влево
+ (плюс)	Используется с числовыми параметрами, вызывает вывод знака числа
.(точка)	Символ, разделяющий значение **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

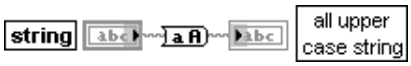
Array To Spreadsheet String **Массив в строку табличного формата**



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**.

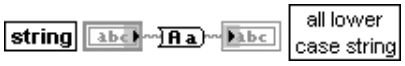
Для преобразования массива строк в **строку табличного формата** можно указать формат строки **%s**, а для преобразования массива данных – формат **%d** или **%f**.
Для трехмерных (и более) массивов каждой странице предшествует серия индексов следующего формата: [**n,m,…0,0**],
где **n** – самый большой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

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.

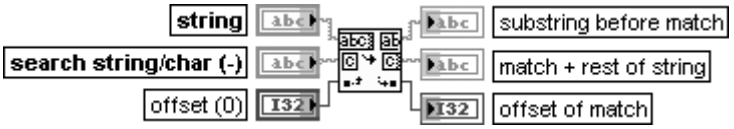
To Lower Case **К нижнему регистру**



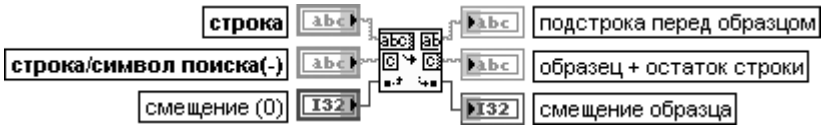
Функция преобразует буквенные символы **строки** (string) в символы нижнего регистра. Воспринимает все числа в **строке** как ASCII коды символов. Эта функция не действует на символы, не являющиеся буквенными.

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

Search/Split Siring

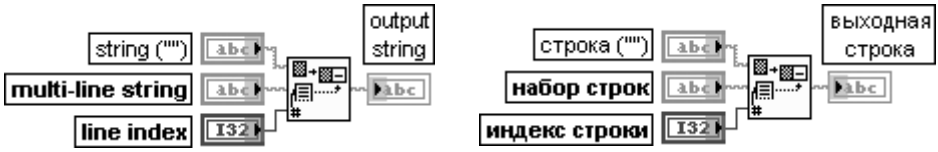


Найти/Разделить строку



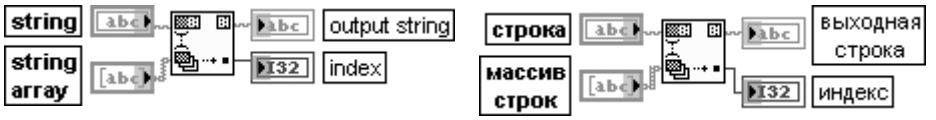
Функция разделяет строку на две части по **строке или символу поиска** (search string/char) начиная от **смещения** (offset). Если функция не находит символ или строку поиска, то на выходе **смещение образца** (offset of match) будет установлена константа –1, на выход **подстрока перед образцом** (substring 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** (offset)

Pick Line **Выбрать строку**



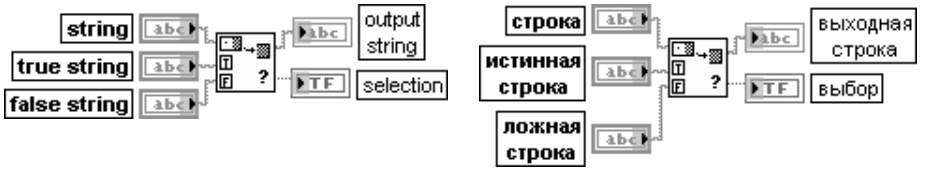
Функция выбирает строку из **набора строк** (multi-line string) по **индексу строки** (line index), присоединяет ее к **строке** (string) и вновь образованную строку передает на выход **выходная строка** (output string).
По умолчанию на вход **строка** подается пустая строка.
Набор строк состоит из одной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

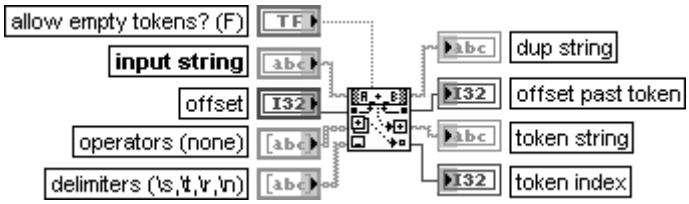
Match True/False String **Сопоставить истинную или ложную строки**



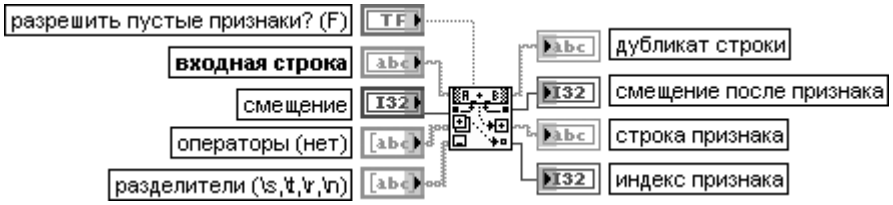
Функция сопоставляет начальную часть **строки** (string) со строками **истинная строка** (true string) или **ложная строка** (false string). Функция возвращает логическое значение

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** (string) 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



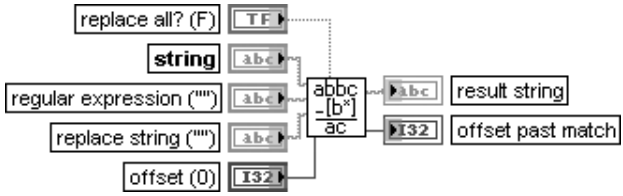
Функция проверяет **входную строку** (input string) начиная от **смещения** (offset) на наличие строки **признаков** (token), перечисленных в массиве **операторы** (operators) или окруженных разделителями, перечисленными в массиве **разделители** (delimiters). Как правило, искомые строки признаков представляют 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 признаков** (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 в научном формате;
- %% – поиск символа %.

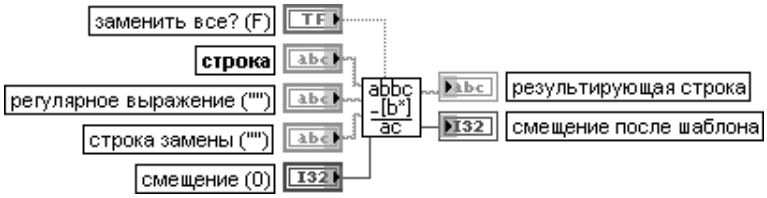
Если вход **разделители** (delimiters) не подключен, то в качестве разделителей используются неотображаемые символы – **пробел** (space), **табуляция** (tab), **перевод строки** (new line), **возврат каретки** (carriage return).
На выход **дубликат строки** (dup string) передается неизменная входная 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.

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.

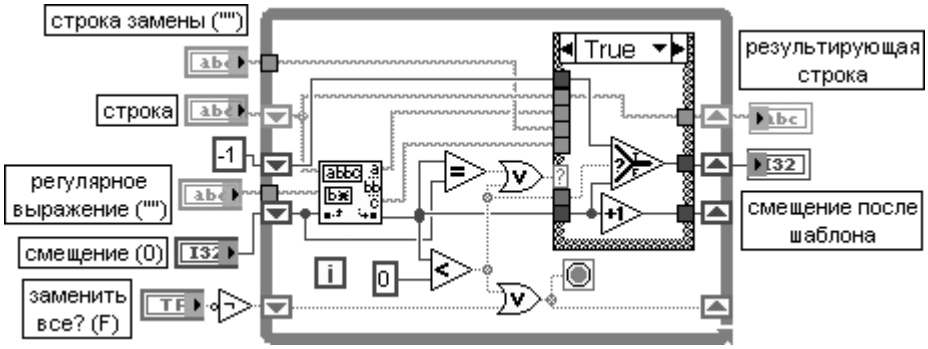
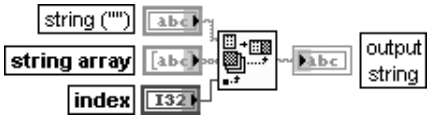
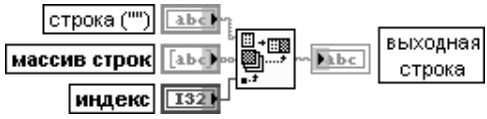


*Рис. 2.13. Блок-диаграмма ВП **Найти и заменить шаблон** (Search and Replace Pattern)*

Index String Array

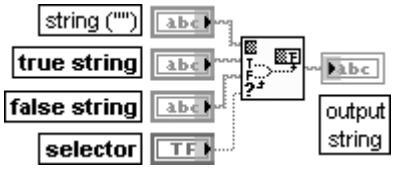


Выбрать строку по индексу

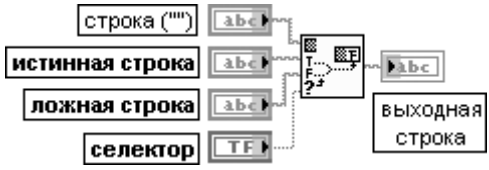


Функция выбирает строку из **массива строк** (string array) по **индексу** и присоединяет ее к **строке** (string)

Append True/False String



Добавить строку по выбору



Функция выбирает **истинную строку** (true string) или **ложную строку** (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)

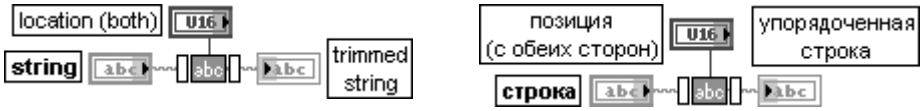
Rotate String **Циклически сдвинуть символы строки**

Функция перемещает первый символ **строки** (string) в последнюю позицию выходной строки **первый символ последним** (first char last), сдвигая все другие символы на одну позицию вперед. 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

Trim Whitespace **Упорядочить 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**.

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.

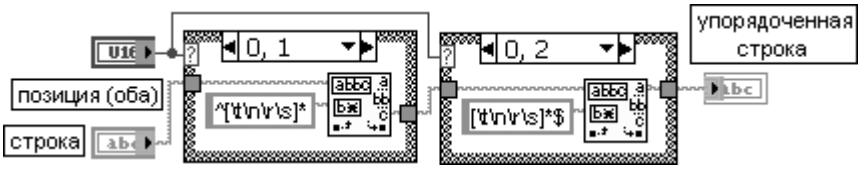
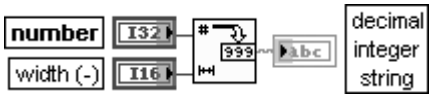


*Рис. 2.14. Блок-диаграмма ВП **Упорядочить пробелы** (Trim Whitespace)*

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

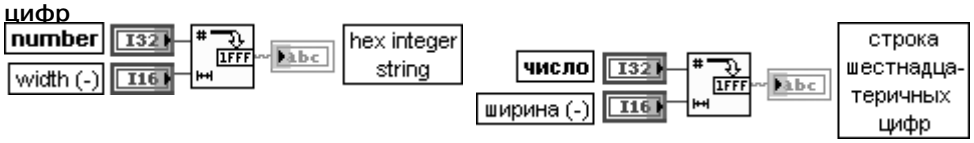
Number To Decimal String **Число в строку десятичных цифр**



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 такую ширину, какая необходима

Number To Hexadecimal String **Число в строку шестнадцатеричных цифр**



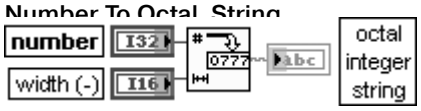
Функция преобразует **число** (number) в **строку шестнадцатеричных цифр** (hex integer string) с шириной, равной или большей значения, установленного на входе **ширина** (width). Цифры A–F всегда отображаются в выходной строке в верхнем регистре. Если **число** дробное, то оно округляется до 32-битового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

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

строку **строка шестнадцатеричных цифр**.

Number To Octal String



Число в строку восьмеричных цифр

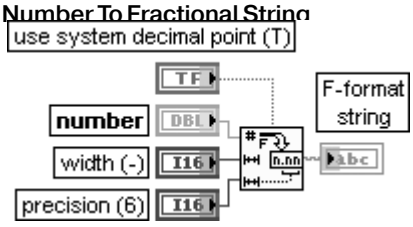
Функция преобразует **число** (number) в **строку восьмеричных цифр** (octal integer string) с шириной, равной или большей значения, установленного на входе **ширина** (width). Если **число** дробное, то оно округляется до 32-битового целого перед преобразованием.

Таблица показывает, как числовые параметры на входах **число** и **ширина** влияют на выходную

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

строку **строка восьмеричных цифр**.

Number To Fractional String



Число в строку с дробным форматом

Функция преобразует **число** (number) в **строку в F-формате** (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,

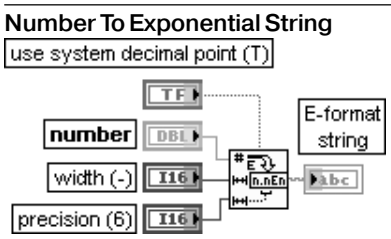
сколько необходимо для представления числа без излишнего дополнения.

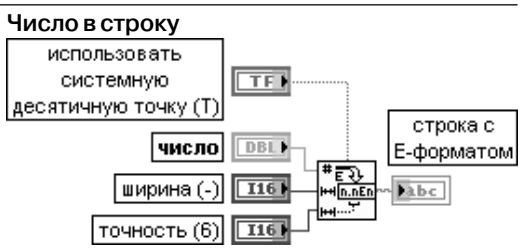
Вход **точность** (precision) должен быть числовым. Функция округляет число цифр после десятичной точки в выходной строке до числа, задаваемого на входе **точность**. Если значение **точность** равно 0, то выходная строка не содержит десятичную точку и содержит по крайней мере три цифры мантиссы. По умолчанию значение **точность** равно 6.

Выход **строка в F-формате** (F-format string) представляет результирующую дробную строку. **Строка в F-формате** может быть **Inf**, **-Inf** или **NaN**, если значение, которое подключено ко входу **число**, является бесконечным или не является числом.

Число	Ширина	Точность	Строка в F-формате	Комментарии
4,911	6	2	_ _4,91	Число округляется и дополняется пробелами слева
,003926	8	4	_ _0,0039	Число округляется и дополняется пробелами слева
–287,3	5	0	_–287	Число округляется и дополняется пробелами слева

Следующая таблица показывает, как числовые параметры на входах **число**, **ширина** и **точность** влияют на выходную строку **строка в F-формате**.



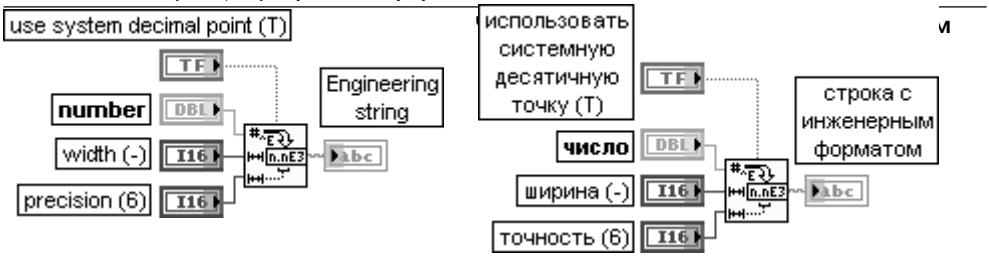


Функция преобразует **число** (number) в **строку с E-форматом** (экспоненциальная запись) представления числа с плавающей запятой, имеющую количество символов, равное или большее значения, заданного на входе **ширина** (width).

Назначение входов идентично назначению одноименных входов рассмотренной выше функции **Число в строку с дробным форматом** (Number To Fractional String).

Число	Ширина	Точность	Строка в E-формате	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

Следующая таблица показывает, как числовые параметры на входах **число**, **ширина** и **точность** влияют на выходную строку **строка в E-формате**.



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**.

Число	Ширина	Точность	Строка в инженерном формате	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

Следующая таблица показывает, как числовые параметры на входах **число**, **ширина** и **точность** влияют на выходную строку **строка в инженерном формате**.

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**

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

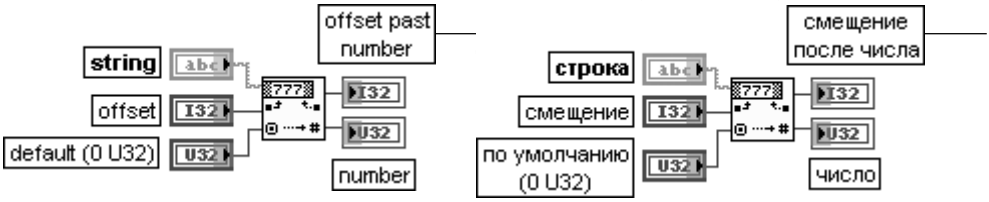
(offset past number) определяет индекс первого элемента строки, следующего за числом.

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

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

выход **число** (number).

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

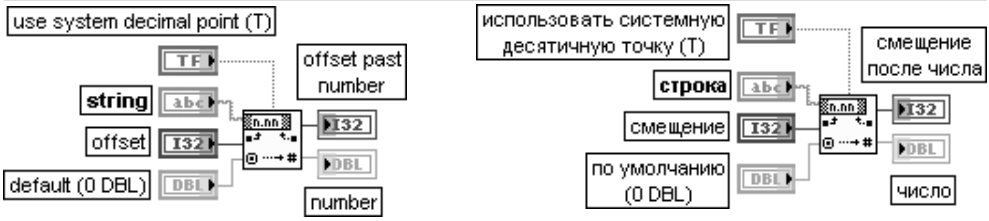


Функция преобразует символы от 0 до 7 **строки** (string), начиная от **смещения** (offset),

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

в целое восьмеричное число и передает его на выход **число** (number).

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

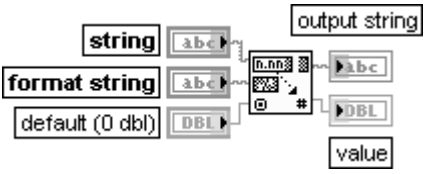


Функция интерпретирует символы от 0 до 9, плюс, минус, **e**, **E** и десятичную точку **строки** (string), начиная от **смещения** (offset) как число с плавающей запятой в инженерной записи, экспоненциальном или дробном формате и передает его на выход

Строка	Смещение	По умолчанию	Смещение после числа	Число	Комментарии
–4,7e–3x	0	0	7	–0,0047	Символ **x** не допустим, поэтому

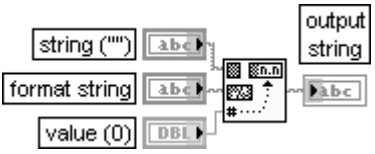
Строка	Смещение	По умолчанию	Смещение после числа	Число	Комментарии
+5,3,2	0	0	4	5,3	преобразование заканчивается на нем Второй десятичный разделитель не допустим, поэтому преобразование заканчивается на нем

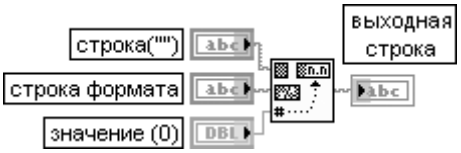
Scan Value **Просмотреть значение**



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**



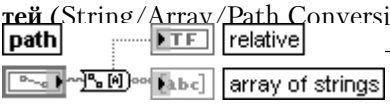


Функция преобразует число, подключенное ко входу **значение** (value), в упорядоченную строку в соответствии с форматом, определенным в **строке формата** (format string), и добавляет ее к входной **строке** (string), образуя **выходную строку** (output string). Данная функция аналогична функции **Преобразовать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

число (number).

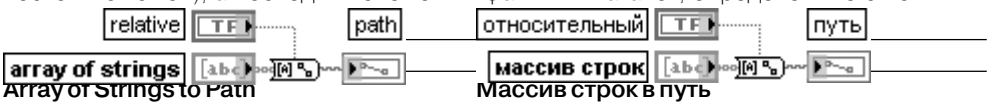
Назначение входов и выходов идентично рассмотренной выше функции **Строку десятичных цифр в число**.

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



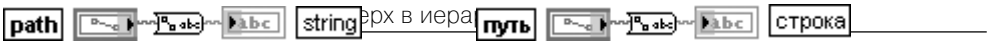
Функция преобразует **путь** (path) в **массив строк** (array of string) и индицирует тип пути.
Выход **относительный** (relative) отображает значение ИСТИНА, если путь является относительным, и ЛОЖЬ, если он является абсолютным.

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 по-



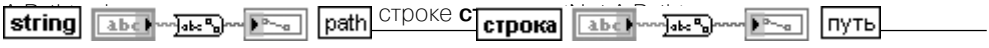
Array of Strings to Path **Массив строк в путь**

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 аналогич-



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

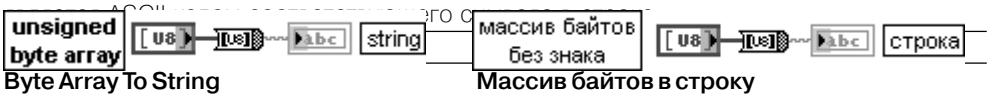


String To Path **Строку в путь**

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

String To Byte Array **Строку в массив байтов**

Функция преобразует строку в массив байтов без знака. Каждый байт в массиве



Byte Array To String **Массив байтов в строку**

Символ	ASCII код (десятичный)	Название
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))

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

Сформировать текст **(Build Text)**

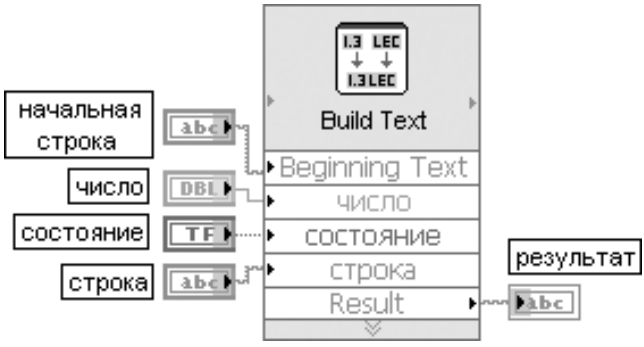


Рис. 2.15. Блок-диаграмма возможного подключения Экспресс-ВП

Экспресс-ВП **Сформировать текст** (Build Text) (рис. 2.15) объединяет входные параметры в строку. Если вход не является строкой, то Экспресс-ВП преобразует вход в строку, опираясь на конфигурацию ВП (рис. 2.16)**.**
Экспресс-ВП использует функции **Объединить строки** (Concatenate Strings) и **Преобразовать в строку** (Format Into String).

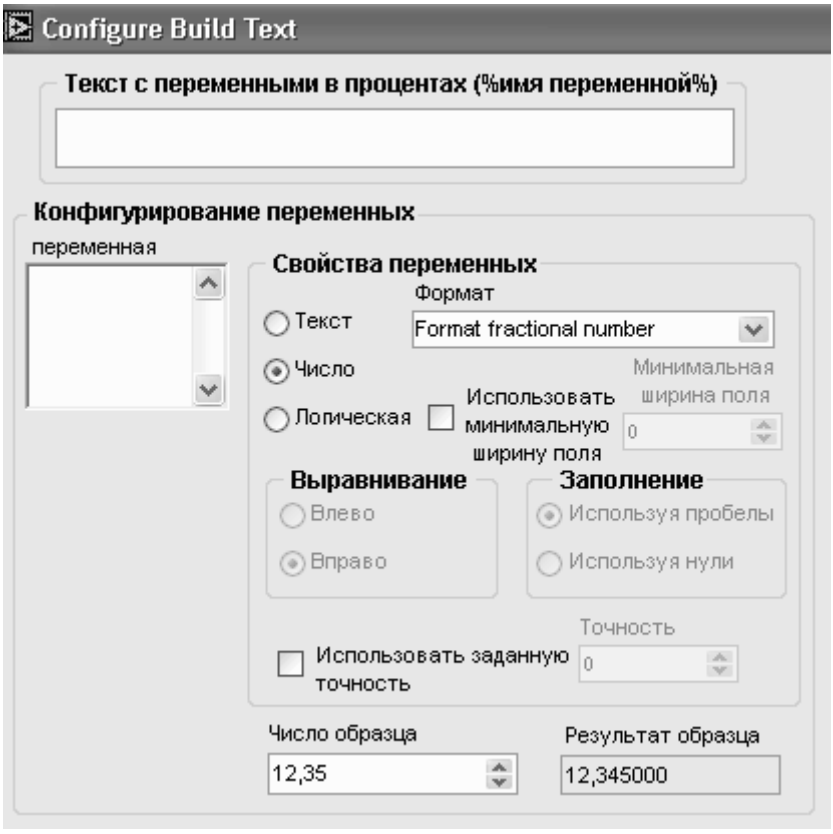
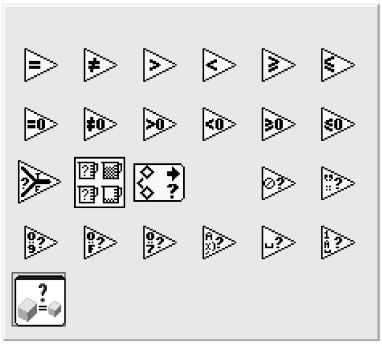


Рис. 2.16. Вид диалогового окна конфигурирования Экспресс-ВП ***Сформировать текст*** *(Build Text)*

Таблица строковых констант:



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

2.4. Функции сравнения

Функции сравнения (рис. 2.17) позволяют проверять различные соотношения между скалярными и векторными переменными и константа-

Рис. 2.17. Палитра функций сравнения

Equal?

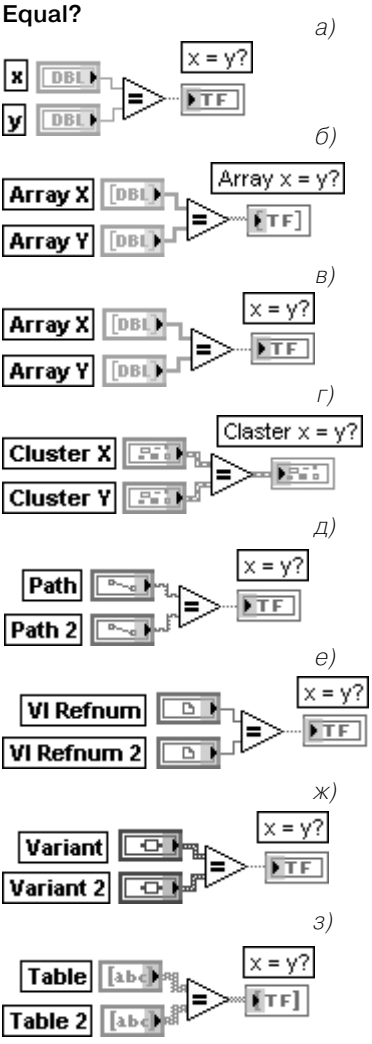


*Рис. 2.18. Варианты подключения функции **Равно?***

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.

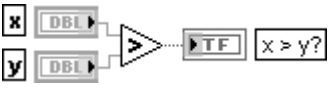
На рис. 2.18 последовательно показаны варианты использования функции **Равно?** для сравнения числовых скаляров (рис. 2.18а), числовых массивов в режиме **Сравнение элементов** (рис. 2.18б) и **Сравнение совокупности** (рис. 2.18в), кластеров в режиме **Сравнение элементов** (рис. 2.18г), путей (рис. 2.18д), ссылок (рис. 2.18е), данных типа Вариант (рис. 2.18ж) и таблиц в режиме **Сравнение элементов** (рис. 2.18з)

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

ми (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, проверять нахождение числа в заданном диапазоне.

Все вышеприведенные функции имеют те же режимы работы, что и рассмот-

Equal To 0?

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

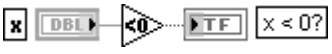
Not Equal To 0?

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

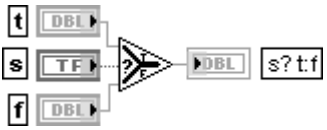
Less Or Equal To 0?

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

ренная более подробно функция **Равно?**.

Следующие шесть функций производят сравнение входной числовой скалярной или векторной переменной **x** с нулем. Результат сравнения логического типа

Select



Выбрать

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

Max & Min

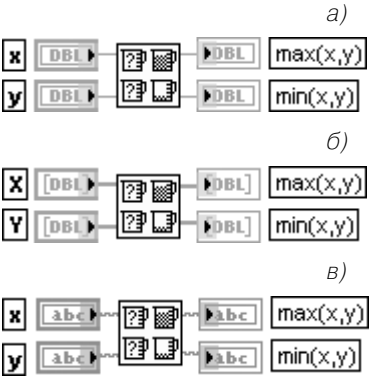


Рис. 2.19. Варианты подключения функции ***Максимум и минимум***

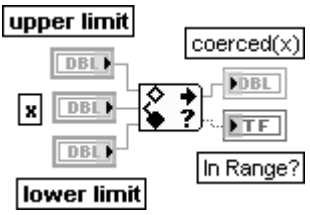
Максимум и минимум

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. 2.19 приведены варианты использования функции для сравнения скалярных числовых значений (рис. 2.19а), числовых массивов (рис. 2.19б) и строк (рис. 2.19в).

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



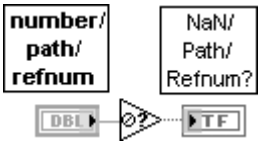
Нахождение в диапазоне и ограничение

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

имеет ту же структуру, что и входная величина.

Следующие три функции позволяют выбрать одну из двух величин, получить

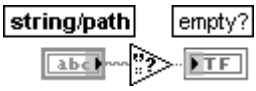
Not A Number/Path/Refnum?



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

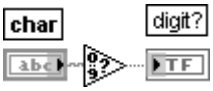
Empty String/Path?



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

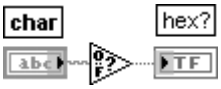
Decimal Digit?



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

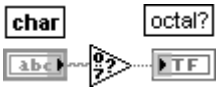
Hex Digit?



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?

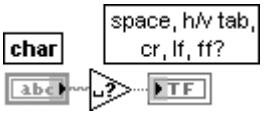
Функция возвращает значение ИСТИНА, если входной **символ** (char) представляет восьмеричную цифру, находящуюся в диапазоне от 0 до 7. Обработка данных на входе **символ** описана выше

Printable?

Печатный символ?

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

White Space?

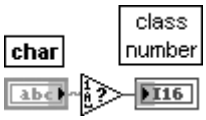


Непечатный символ?

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 неарифметические ASCII символы

значения большей и меньшей величины и определить нахождение величины в за-

Сравнение (Comparison)

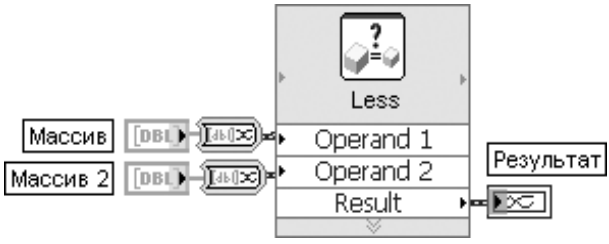


Рис. 2.20. Блок-диаграмма возможного подключения Экспресс-ВП

Экспресс-ВП **Сравнение** (Comparison) (рис. 2.20) сравнивает заданные входы с целью определения их равенства или неравенства. Назначение опций диалогового окна конфигурирования очевидно из их перевода (рис. 2.21).

В Экспресс-ВП **Сравнение** используется функциональность следующих ВП: And Array Elements, Not, Equal?, Greater?, Greater Or Equal?, In Range and Coerce, Less?, Less Or Equal?, Not Equal?

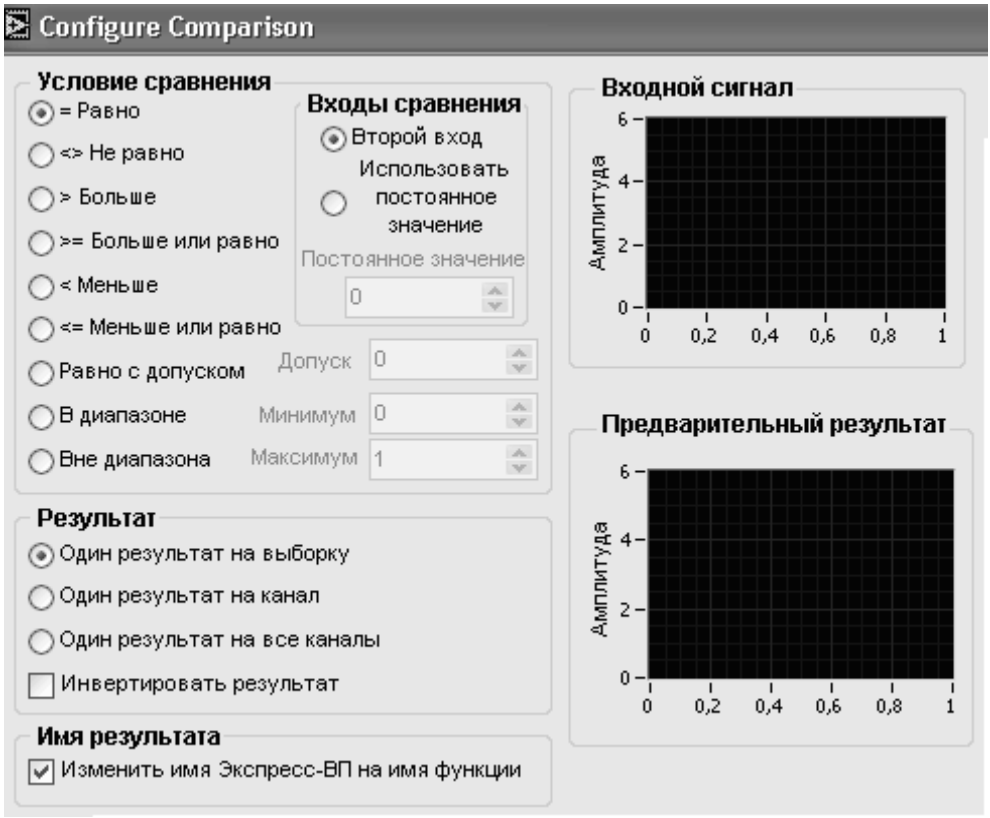


Рис. 2.21. Вид диалогового окна конфигурирования Экспресс-ВП ***Сравнение*** *(Comparison)*

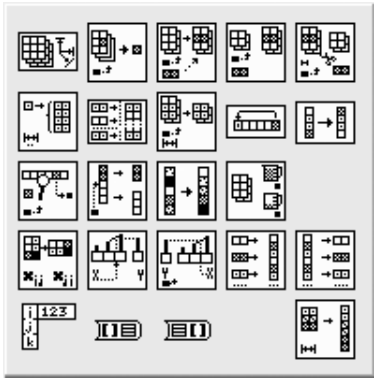
данном диапазоне.

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

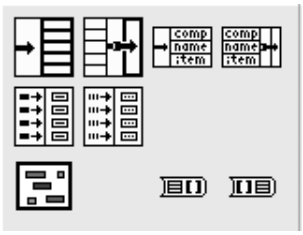
В состав палитры входит Экспресс-ВП **Сравнение** (Comparison).

2.5. Функции работы с массивами и кластерами

Функции работы с массивами (рис. 2.22а) 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.



а)

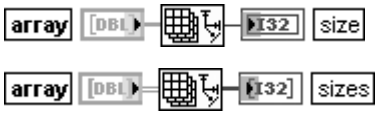


б)

Рис. 2.22. Палитры функций работы с массивами (а) и кластерами (б)

Часть функций для увеличения числа входов может быть растянута в вертикаль-

Array Size **Размер массива**

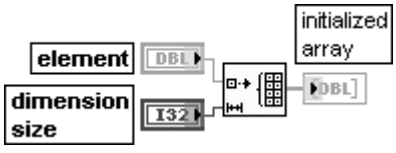


Функция возвращает число элементов массива каждой размерности. Функция является полиморфной, то есть она может определять размеры массива произвольной размерности. На рисунках показаны варианты определения размеров одномерного и двумерного массивов.

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

Initialize Array **Инициализировать массив**



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**

Index Array **Индексировать массив**

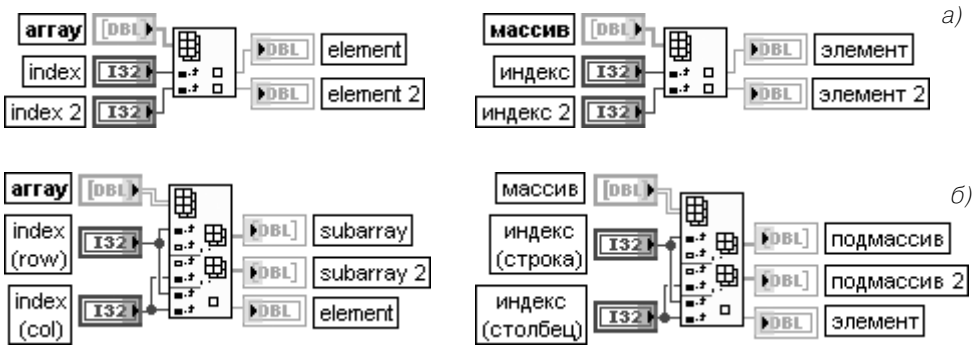


*Рис. 2.23. Варианты подключения функции **Индексировать массив***

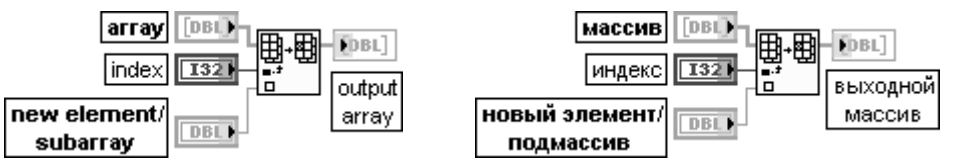
Функция возвращает **элемент** (element) или **подмассив** (sub-array) входного **массива** (array) начиная с **индекса** (index). При подключении входного массива функция автоматически перестраивается в соответствии с его размерностью, отображая входы **индекс** для каждой размерности (рис. 2.23б). 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.

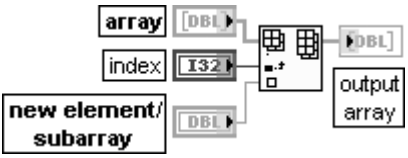
Выход **элемент** или **подмассив** имеет тот же тип, что и элементы входного массива. Наряду с извлечением элемента массива функция позволяет извлекать подмассив из массива, если один из входов индексирования был оставлен неподключенным. Так, например, оставляя неподключенным вход индексирования по столбцам (рис. 2.23б), 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 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**

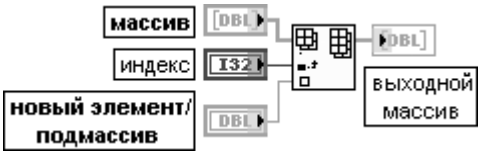


Функция заменяет элемент или подмассив входного **массива** начиная с точки, определенной на входе **индекс** (index). Функция также перестраивается в соответствии с размерностью подключаемого массива. Если входы индексирования по определенной размерности не подключены, то функция заменяет все элементы 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 массив или элемент, 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

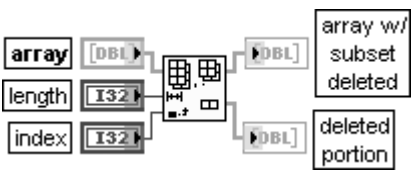


Вставить в массив

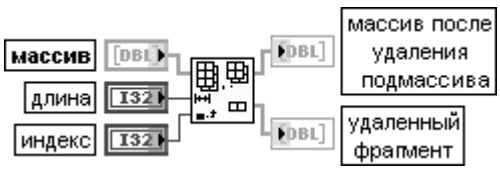


Функция вставляет элемент или подмассив во входной массив начиная с точки, определенной на входе **индекс** (index). Функция также перестраивается в соответствии с размерностью подключаемого массива. Если входы индексирования по определенной размерности не подключены, то функция добавляет новый элемент 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.

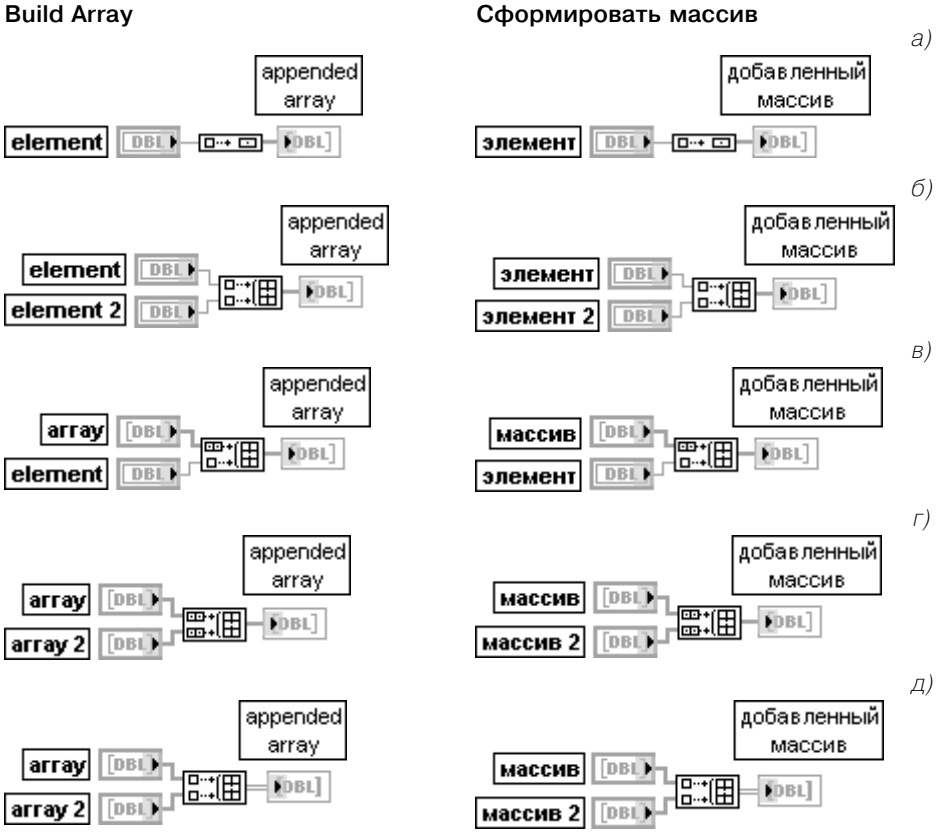


Рис. 2.24. Варианты подключения функции ***Сформировать массив***

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. 2.24а). Использование ее в таком виде позволяет преобразовать скалярную величину в одномерный массив, содержащий один элемент.

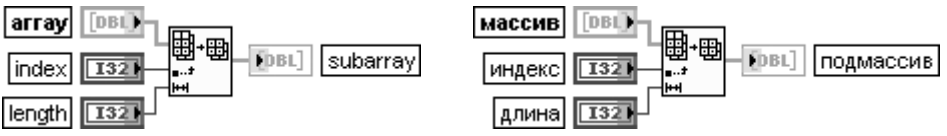
Количество входов можно увеличить с помощью вызова строки **добавить вход** (Add Input) контекстного меню или увеличивая размер функции в вертикальном направлении с помощью инструмента перемещения (рис. 2.24б, 2.24в, 2.24г, 2.24д).

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. 2.24г). Если же эта опция не выбрана, то функция формирует выходной массив с размерностью, на единицу большей размерности входного массива (рис. 2.24д). 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. 2.24в), то опция **объединить входы** выбирается автоматически и не может быть снята. Если все входы являются скалярными элементами (рис. 2.24б), 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 прямоугольники в иконке **Сформировать массив** изменяются с целью отражения различий двух типов входов. Входы, имеющие такую же размерность, что и выходы, отображаются прямоугольниками массива, в то время как входы, имеющие меньшую по сравнению с выходом размерность, отображаются отдельным прямоугольником

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.

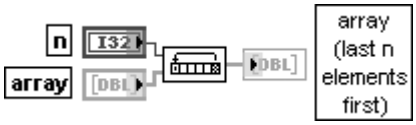
Вход **массив** может быть n-мерным массивом произвольного типа.

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**

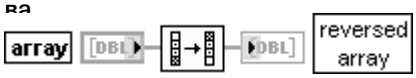
Rotate 1D Array
массив



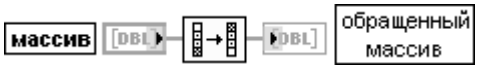
Циклически сместить одномерный

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 м**ассив (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
ва

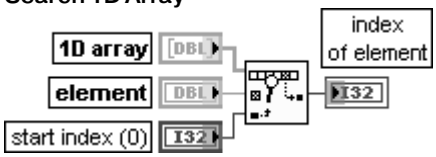


Обратить элементы одномерного масси-



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**

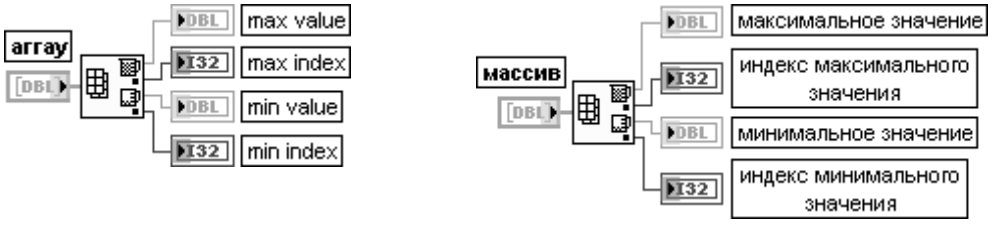
Функция делит **массив** (array) по **индексу** (index) и возвращает два подмассива.
Вход **массив** может быть одномерным массивом любого типа.
Вход **индекс** должен быть числовым. Если **индекс** меньше или равен 0, **первый подмассив** (first subarray) является пустым. Если **индекс** больше или 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, функция сравнивает вторые и последующие элементы

Array Max & Min **Максимум и минимум массива**

а)



б)

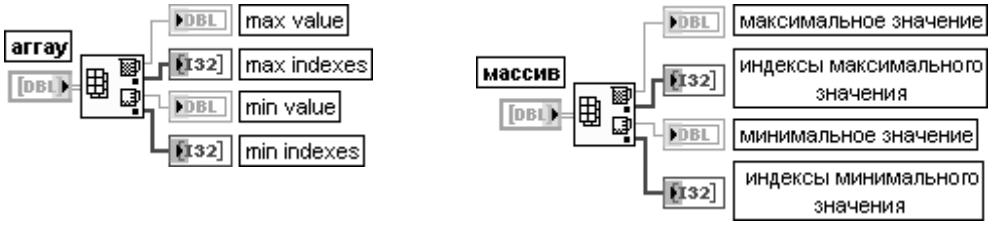
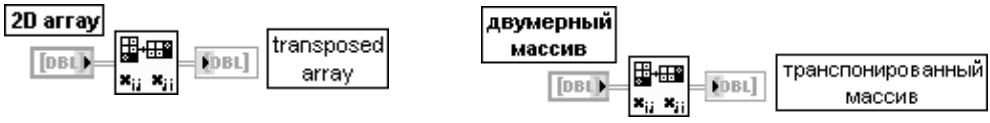


Рис. 2.25. Варианты подключения функции ***Максимум и минимум массива***

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. 2.25а). Если числовой **массив** является многомерным, то эти выходы представляют одномерный массив, который содержит индексы максимального и минимального значений (рис. 2.25б)

Transpose 2D Array **Транспонировать двумерный массив**



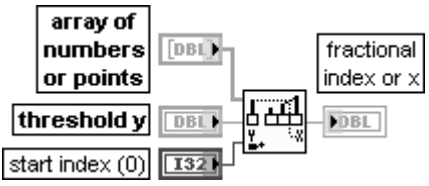
Функция переставляет элементы в **двумерном массиве** (2D array) так, что **двумерный массив**[i,j] становится **транспонированным массивом**[j,i]

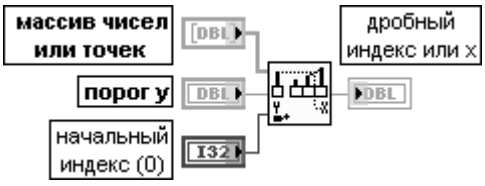
Interpolate 1D Array **Интерполировать одномерный массив**

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 индекса или x** как ссылку к элементам массива. При подключении массива наборов точек данных функция интерпретирует **дробный индекса 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 индекса 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 индекса 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 **дробный индекса или x**. Например, если массив содержит две точки, (3, 7) и (5, 9) и на входе **дробный индекса или x** установлено значение 3,5, то функция вернет значение 7,5.
Вход **дробный индекса или x** не интерполирует за границами массива или набора точек данных. Для корректной работы функции значение **дробный индекса 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 входной массив **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, 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) и 7,8 (индекс 2). Если **порог y** равно 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 индекс 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)

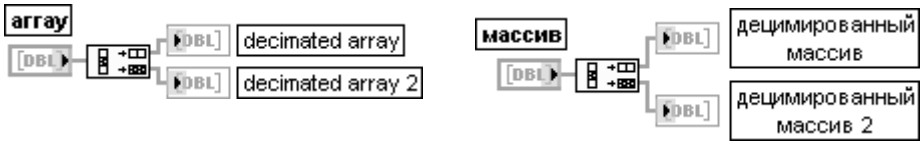
Interleave 1D Arrays **Чередовать одномерные массивы**

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 **Децимировать 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 **децимированный 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 **децимированный массив 2** представляет второй выходной массив и содержит элементы 1, n+1, 2n+1**...** и т. д.

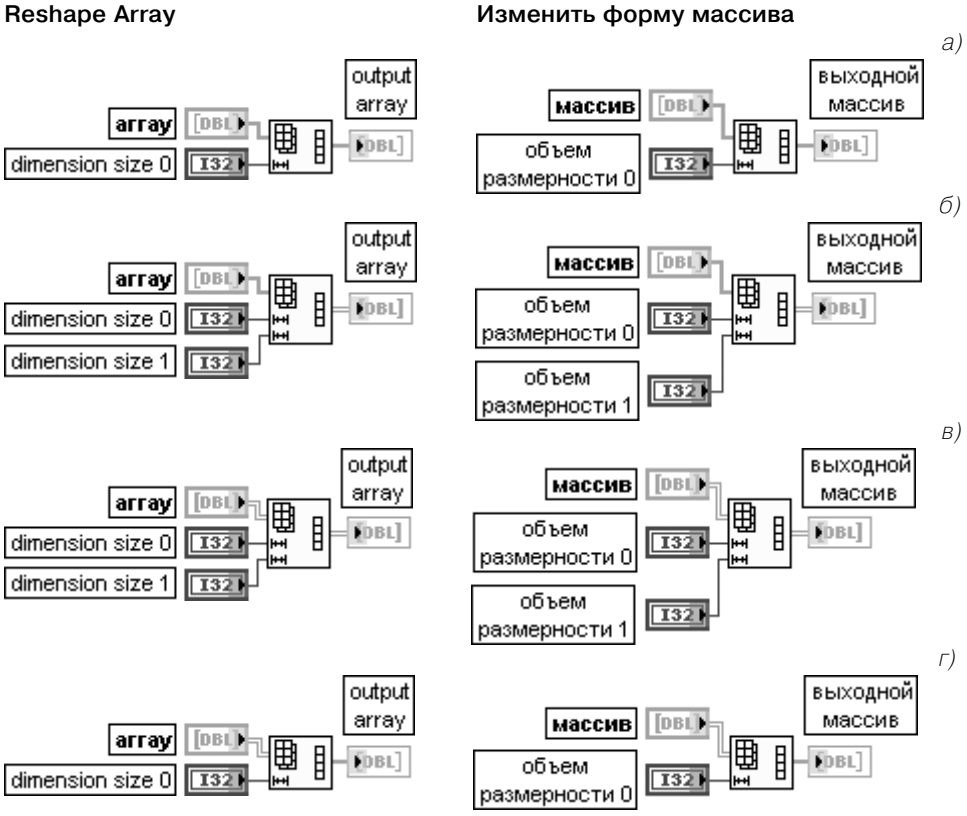


*Рис. 2.26. Варианты подключения функции **Изменить форму массива***

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. 2.26 показаны варианты применения функции для изменения размера одномерного массива (рис. 2.26а), преобразования одномерного массива в двумерный (рис. 2.26б), изменения размера двумерного массива (рис. 2.26в) и преобразования двумерного массива в одномерный (рис. 2.26г)

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

ном направлении с помощью инструмента перемещения.

Функции работы с кластерами (рис. 2.22б) позволяют собирать кластеры из отдельных элементов и разделять кластеры на элементы, а также формировать

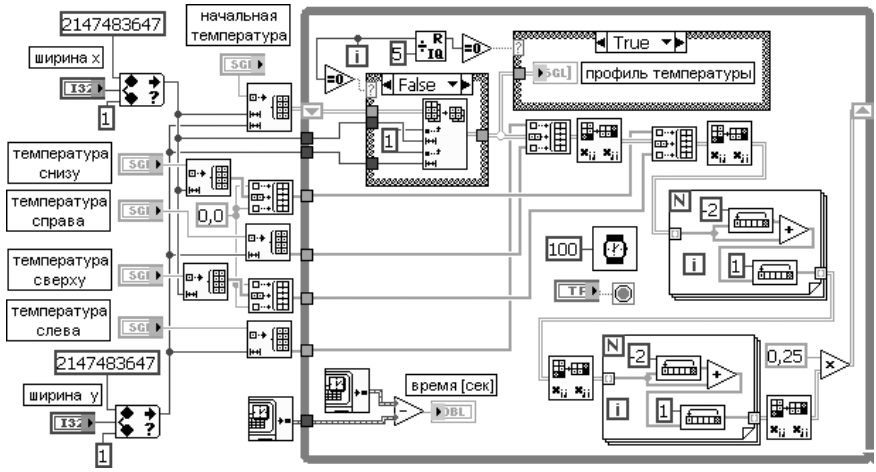
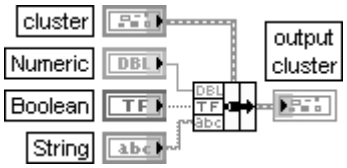


*Рис. 2.27. Блок-диаграмма ВП **Уравнение нагрева** (Heat Equation)*

массивы кластеров.

Bundle

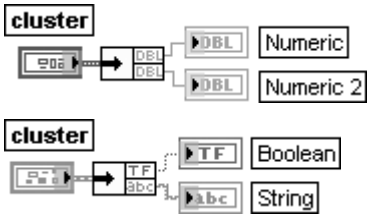


Сборка кластера

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

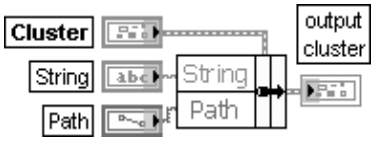
Unbundle



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

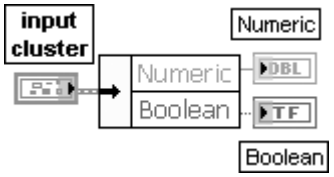
Bundle By Name



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

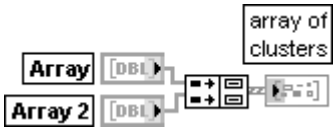
Unbundle By Name



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

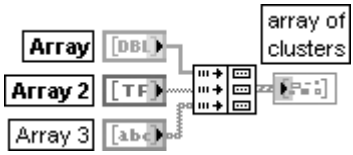
Build Cluster Array



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

Ниже в таблицах описаны функции работы с массивами.

В качестве примера эффективного использования функций работы с массивами на рис. 2.27 приведена блок-диаграмма ВП **Уравнение нагрева** (Heat Equation) из набора примеров NI Example Finder LabVIEW.

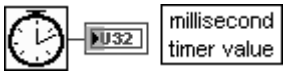
В следующей таблице рассмотрены функции работы с кластерами.

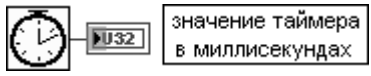
2.6. Функции времени и диалогов

Функции времени и диалогов (рис. 2.28) позволяют вводить задержки в работу ВП или измерять длительность выполнения отдельных операций, вводить текущее время и дату, а также выводить диалоговые окна с различным числом

Рис. 2.28. Палитра функций времени и диалогов

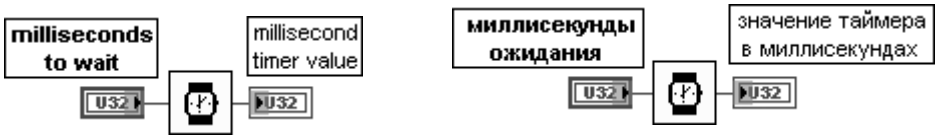
Tick Count (ms) — **Счетчик времени (мс)**





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)**

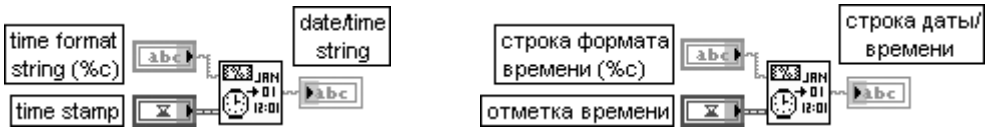


Функция ожидает заданное число миллисекунд и возвращает значение таймера в миллисекундах. Данная функция выполняет асинхронные системные вызовы, но сама работает синхронно. Следовательно, она не завершит выполнение до истечения заданного времени

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

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.

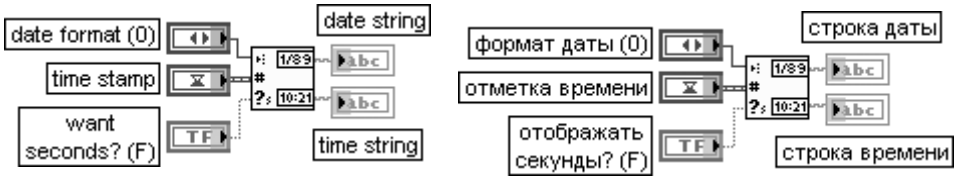
Вход **отметка времени** (time stamp) может быть значением времени или числом. В числовом представлении этот параметр представляет не зависящее от временного пояса число полных секунд, прошедших с 0:00 1 января 1904 года по Гринвичу. По умолчанию берется текущая дата и время.

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 **Форматировать строку даты/времени** (Format Date/Time String). В таблице приведены варианты формата даты при использовании системного формата:

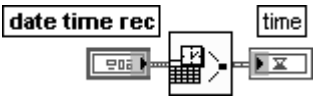
0	**Короткий** (short) – 20.01.2004
1	**Длинный** (long) – 20 января 2004 г.
2	**Сокращенный** (abbreviated) – 20 янв 2004 г.

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**

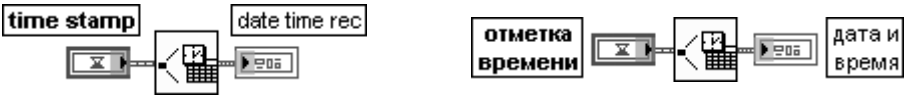
Функция возвращает на выходе **текущее время** (current time) значение текущего времени, равного системному времени LabVIEW или числу полных секунд, прошедших с 0:00 1 января 1904 года по Гринвичу

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**

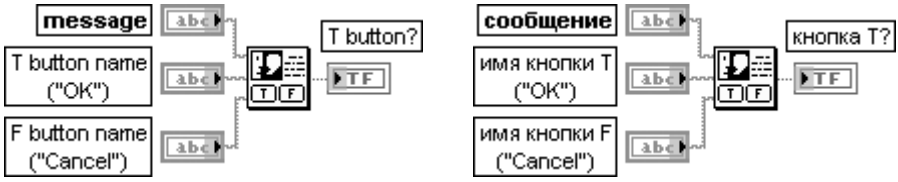


Функция преобразует значение времени или числовое значение в не зависящее от временного пояса число полных секунд, прошедших с 0:00 1 января 1904 года по Гринвичу, и возвращает кластер из девяти 32-битовых целых чисел со знаком, определяющих значение времени или числовое значение.
Компоненты кластера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

One Button Dialog **Диалоговое окно с одной кнопкой**

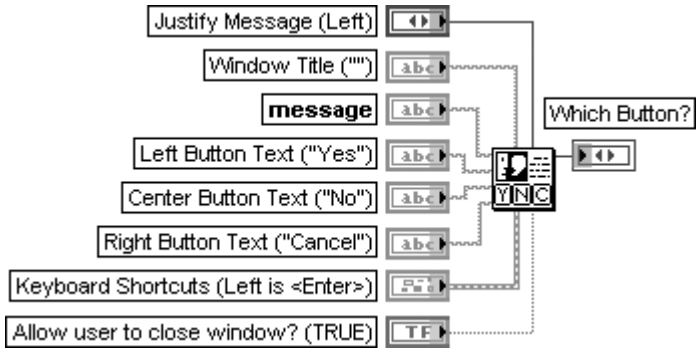
Функция отображает диалоговое окно, которое содержит сообщение и одну кнопку.
Вход **сообщение** (message) содержит текст, отображаемый в диалоговом окне.
Вход **имя кнопки** (button name) содержит имя, отображаемое на кнопке диалогового окна. По умолчанию отображается OK.
На выходе **истина** (true) выводится значение ИСТИНА при нажатии на кнопку

Two Button Dialog **Диалоговое окно с двумя кнопками**

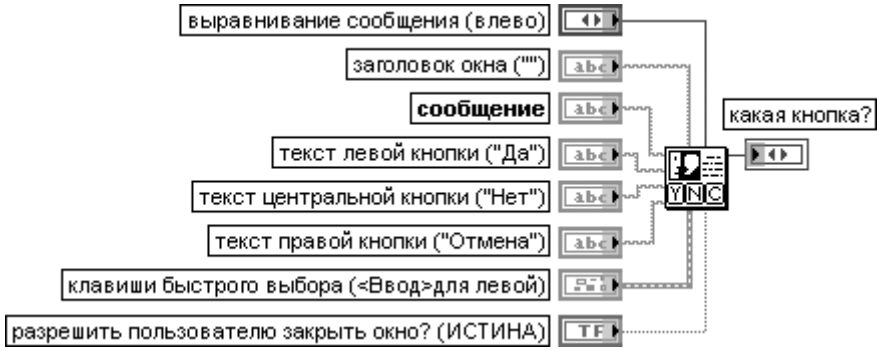


Функция отображает диалоговое окно, которое содержит диалоговое окно и две кнопки.
Вход **сообщение** (message) содержит текст, отображаемый в диалоговом окне.
Вход **имя кнопки T** (T button name) определяет имя, отображаемое на одной из кнопок диалогового окна. По умолчанию отображается OK.
Вход **имя 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



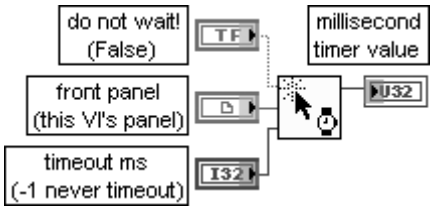
ВП отображает диалоговое окно, содержащее сообщение и три кнопки.
Вход **выравнивание сообщения** (Justify Message) устанавливает выравнивание отображаемого текста (таблица).

0	Left – выравнивает текст влево
1	Center – выравнивает текст по центру
2	Right – выравнивает текст вправо

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 клавишу, появляющуюся 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.

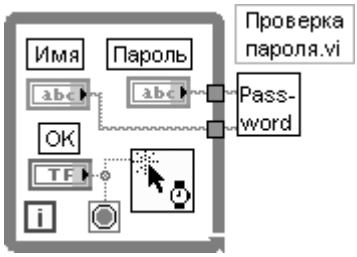
Ссылка к ВП или объекту должна относиться к локальной среде LabVIEW. Не допускается ссылка к ВП или объекту удаленной среды LabVIEW.

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 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.

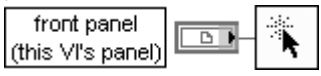


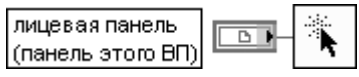
Блок-диаграмма следующего примера показывает, как функция **Ожидать активность лицевой панели** формирует паузу в работе ВП до момента ввода имени или пароля или нажатия кнопки OK.

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

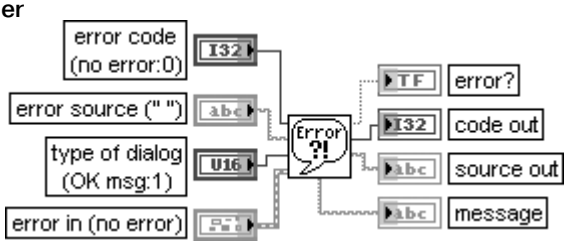
Генерировать активность лицевой пане-ли



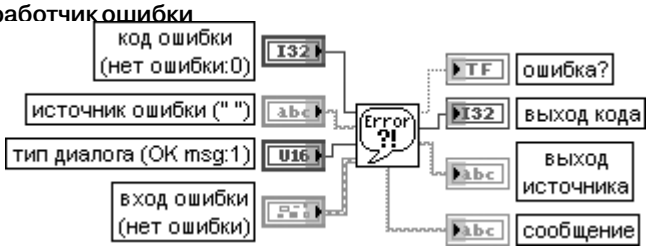


Функция программно генерирует активность лицевой панели, которая приводит к продолжению выполнения любого ВП, остановленного функцией **Ожидать активность лицевой панели** (Wait on Front Panel Activity). На лицевой панели при этом не происходит каких-либо изменений

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).

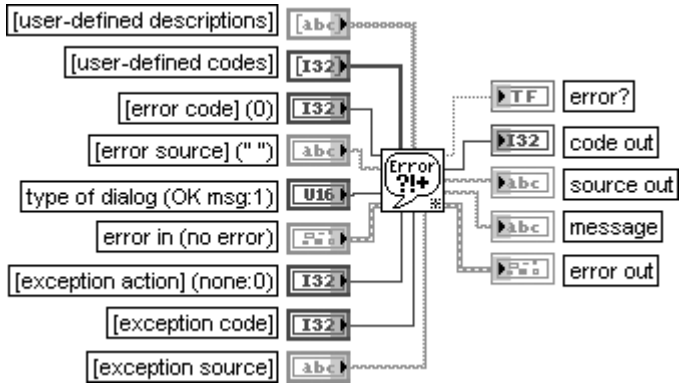
Вход **тип диалога** (type of dialog) определяет тип отображаемого диалогового окна, если таковое выводится. Независимо от этого значения ВП выводит информацию об ошибке и **сообщение** (message), описывающее ошибку. Варианты диалога приведены в таблице.

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 применять для продолжения или остановки ВП. При выборе пользователем кнопки **остановить** ВП вызывает

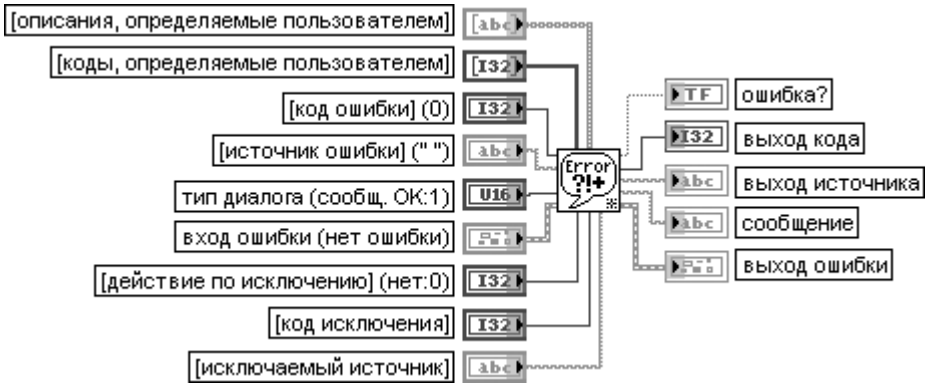
	одноименную функцию для остановки выполнения
3	**Сообщение OK с предупреждением** (OK message with warnings) – выводится диалоговое окно с каким-либо предупреждением и одной кнопкой OK. После подтверждения пользователем сообщения в диалоговом окне ВП возвращает управление основному ВП
4 **ем**	**Сообщение с кнопками «продолжить» или «остановить» с предупреждени-** (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 **код ошибки** (error code).
Выход источника (source out) показывает источник ошибки. Строка **выход источника** является более содержательной, чем строка **источник** на **входе ошибки** (error in).
Выход **сообщение** (message) выводит код произошедшей ошибки, источник ошибки и описание ошибки

General Error Handler

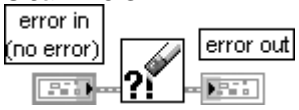


Общий обработчик ошибки

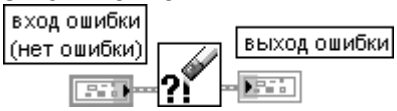


ВП показывает источник возникновения ошибки. Если ошибка возникла, то этот ВП возвращает описание ошибки и дополнительно отображает диалоговое окно. В связи со специфичностью ВП описание его функциональности ограничено переводом надписей входных и выходных терминалов

Clear Errors



Очистить ошибки

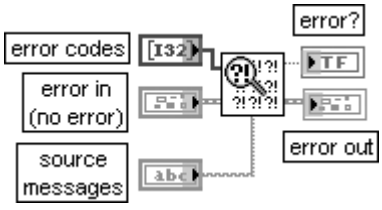


ВП сбрасывает (очищает) следующие элементы **выходного кластера ошибки** (error out): **статус** (status) ошибки устанавливается в состояние **нет ошибки** (no error), **код** (code) устанавливается в 0, и **источник** (source) задается пустой строкой. Данный ВП целесообразно использовать при необходимости игнорировать ошибки. Он фактически содержит разъединенные кластеры входной и выходной ошибок.

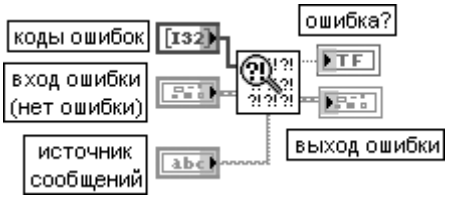
Вход кластера ошибки (error in) содержит аналогичные элементы, описывающие условия появления ошибок. Состав элементов и их назначение приведены в таблице.

TF	**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
I32	**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)
abc	**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

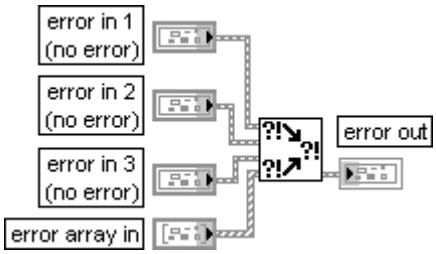


ВП проверяет **статус ошибки** (error status) одной или более функций или подприборов (subVIs), которые порождают на выходе числовой код ошибки.

Вход **коды ошибки** (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.

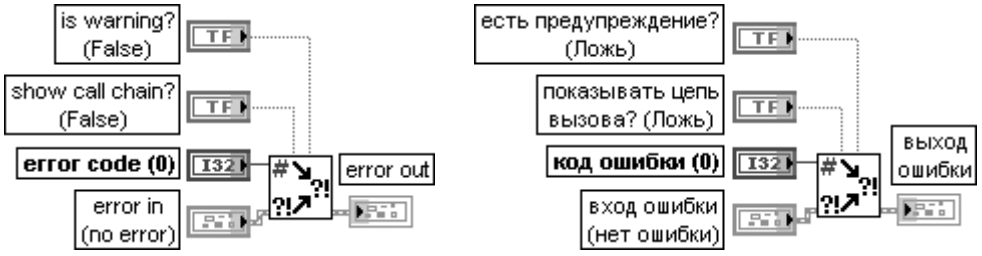
Вход **источник сообщений** (source messages) содержит источник сообщений, которые должны появляться в кластере **выход ошибки,** если ВП находит ошибку на входе **коды ошибок**. Использование этого входа носит дополнительный характер.
Выход **ошибка?** (error?) устанавливается в состояние ИСТИНА, если кластер **вход ошибки** или какой-либо код на входе **коды ошибки** 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 **выходном кластере 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

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

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 к 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)

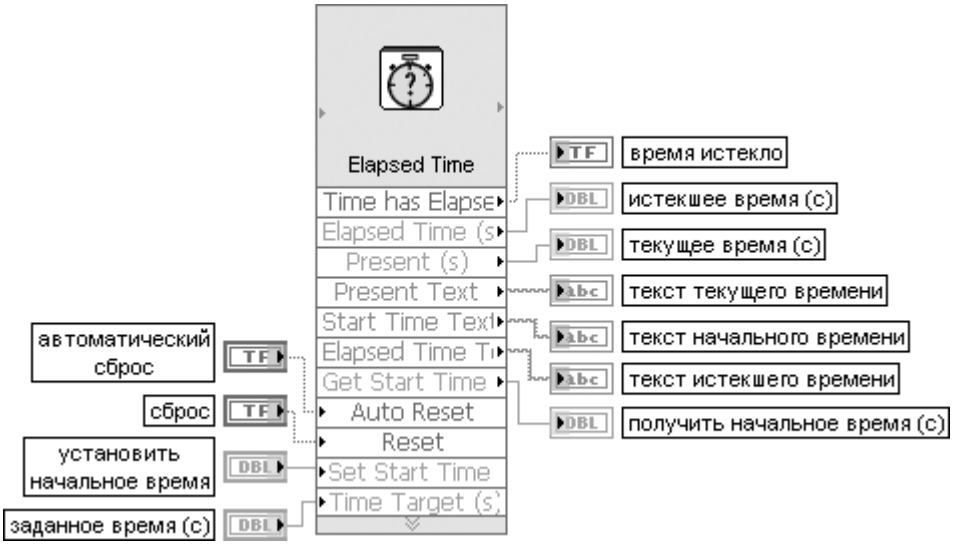


Рис. 2.29. Блок-диаграмма возможного подключения Экспресс-ВП

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. 2.29):
Автоматический сброс (Auto Reset) – устанавливает начальное время равным значению **текущего времени** (Present (s)), когда Экспресс-ВП достигает заданного времени (Time Target (s)).

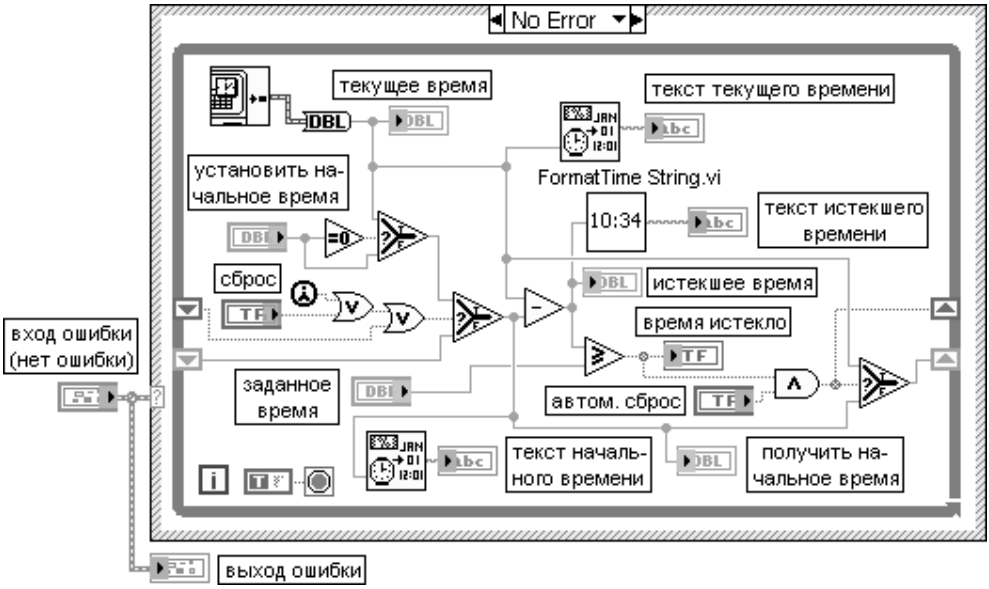


*Рис. 2.30. Блок-диаграмма Экспресс-ВП **Истекшее время** (Elapsed Time)*

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 з**аданного 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. 2.30)

Подсказка пользователю для ввода (Prompt User for Input)

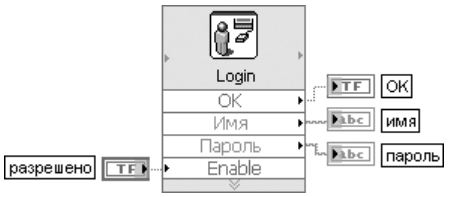


Рис. 2.31. Блок-диаграмма возможного подключения Экспресс-ВП

Рис. 2.32. Вид диалогового окна конфигурирования Экспресс-ВП ***Подсказка пользователю для ввода*** *(Prompt User for Input)*

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. 2.31).
Диалоговое окно данного Экспресс-ВП имеет следующие опции (рис. 2.32):
Отображаемое сообщение (Message to Display) – содержит текст, отображаемый в диалоговом окне.
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.

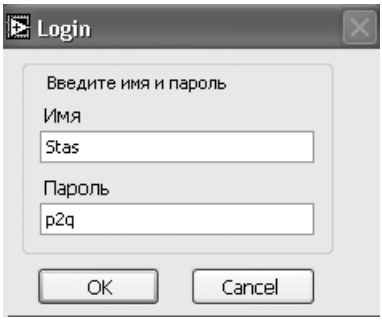


Рис. 2.33. Вид диалогового окна, сформированного Экспресс-ВП ***Подсказка пользователю для ввода*** *(Prompt User for Input)*

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).

На рис. 2.33 показан вид диалогового окна, выводимого Экспресс-ВП, сконфигурированным с помощью окна, приведенного на рис. 2.32.

целью экономии места не указываются, что, конечно, не отменяет необходимости их применения при построении реальных ВП.

В состав палитры функций времени и диалогов входят Экспресс-ВП **Временная задержка** (Time Delay), **Истекшее время** (Elapsed Time), **Подсказка пользователю для ввода** (Prompt User for Input) и **Отображение сообщения пользователю** (Display Message to User).

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**.

2.7. 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: **Ввод/вывод двоичных файлов** (Binary File VIs) (рис. 2.34б), **Дополнительные файловые функции** (Advanced File Functions) (рис. 2.34в), **Файловые константы** (File Constants) (рис. 2.34г) и **ВП файлов конфигурации** (Configuration File VIs) (рис. 2.34д). 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 трех форматов: текстовые, двоичные и файлы протокола (datalog file). Тип формата зависит от типа получаемых или формируемых данных и от приложения, в котором они бу-

дут 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

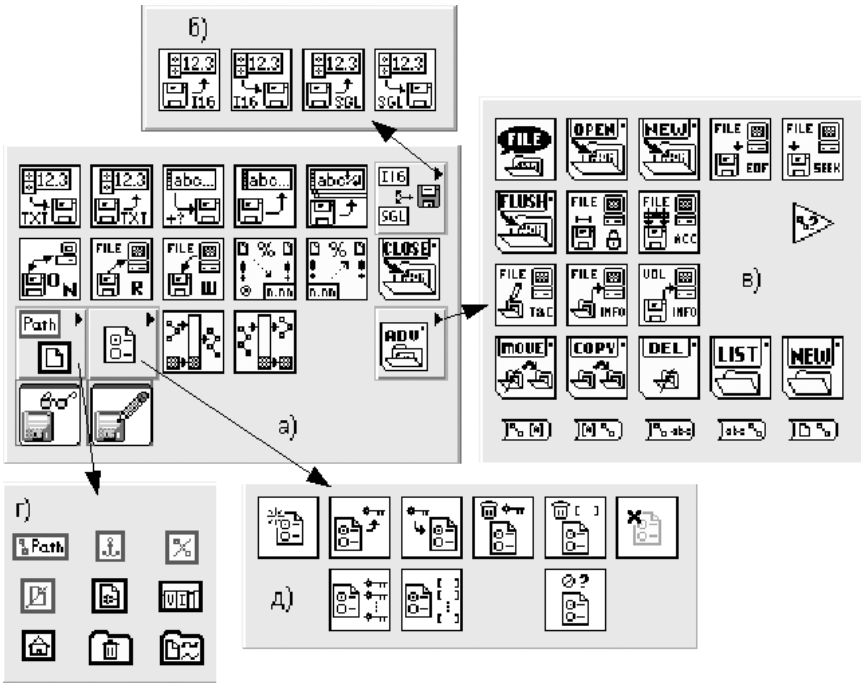


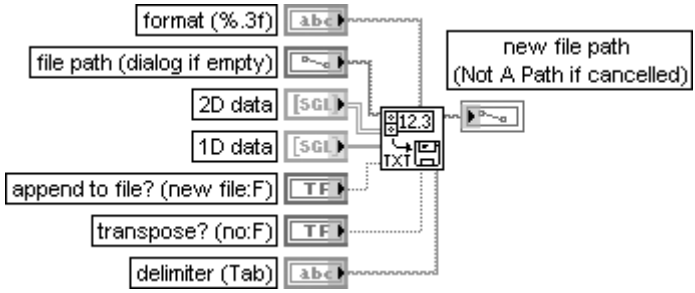
Рис. 2.34. Вид основной палитры (а) и дополнительных подпалитр (б–д) функций работы с файлами

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. Формат файлов протокола применяется при сохранении данных сложной структуры и последующем их использовании только в рамках LabVIEW.

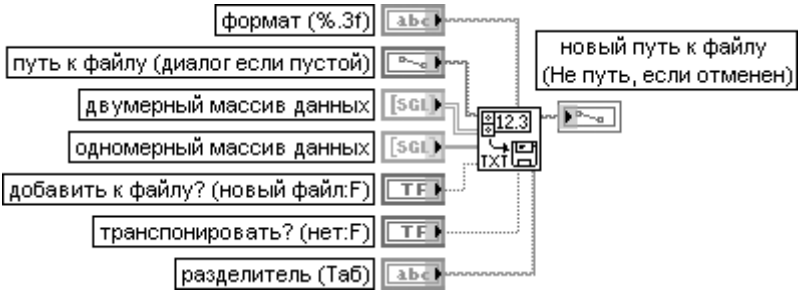
Среди функций ввода/вывода файлов можно выделить функции высокого уровня (High-Level), представленные в виде подприборов в верхней части основной палитры и подпалитре **Ввод/вывод двоичных файлов**, и функции низкого уровня (Low-Level), размещенные в оставшейся части основной палитры и в подпалитре **Дополнительные файловые функции**.

Функции высокого уровня используются для выполнения общих операций ввода/вывода текстовой или числовой информации. Выполнение таких операций включает, как правило, три этапа: открытие уже существующих файлов или создание новых файлов; запись в файл или чтение из файла; закрытие файла.

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.

Данный 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** (ID data) не подключен или на него ничего не поступает.

На вход **одномерные данные** (ID 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.

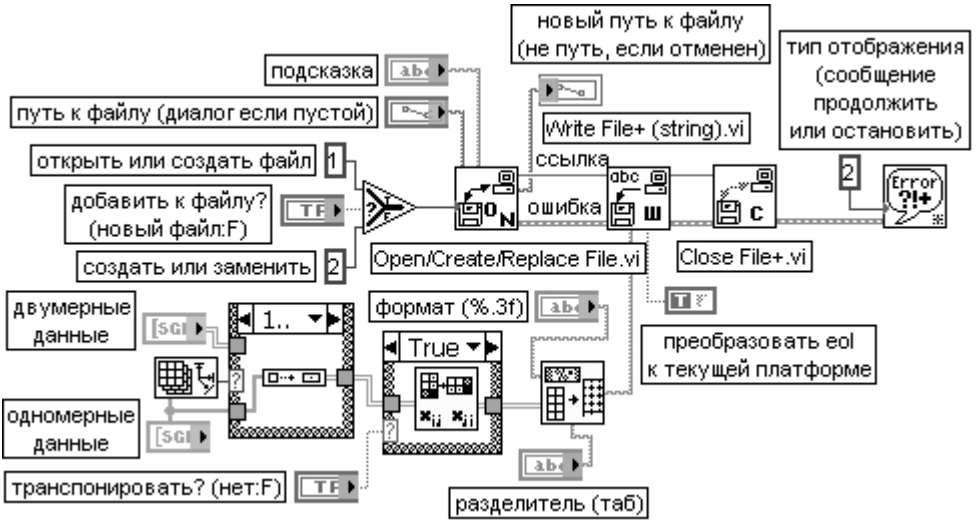
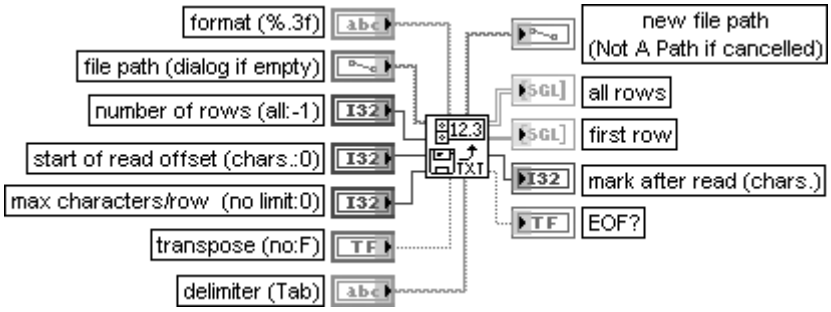
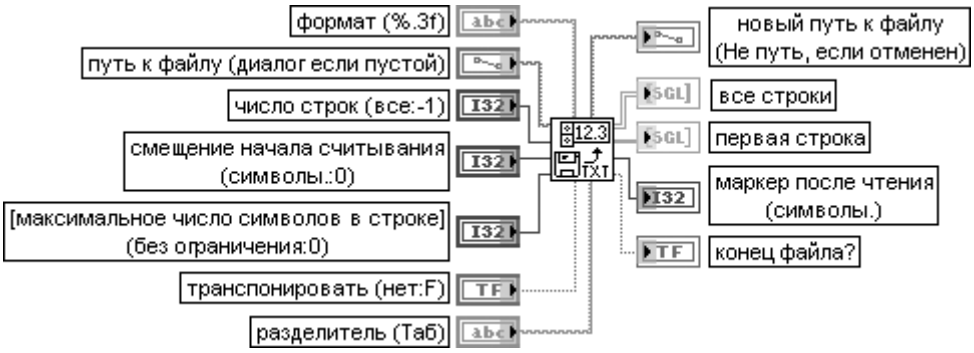


*Рис. 2.35. Блок-диаграмма ВП **Записать в файл табличного формата** (Write To Spreadsheet File)*

Read From Spreadsheet File



Считать из файла табличного формата



ВП считывает определенное число линий или строк из текстового файла, начиная с определенного начального символа, а затем преобразует данные в двумерный массив чисел с одинарной точностью. Этот ВП открывает файл перед выполнением операции чтения и закрывает его после завершения чтения. Данный 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», сопровождаемым 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.

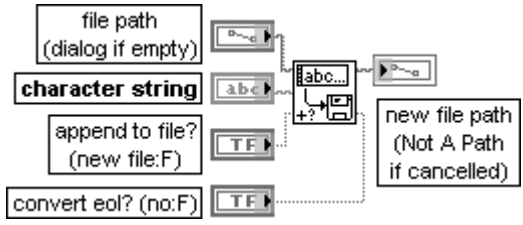
Выход **все строки** (all rows) отображает данные, считанные из файла, в форме двумерного массива чисел с одинарной точностью.

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

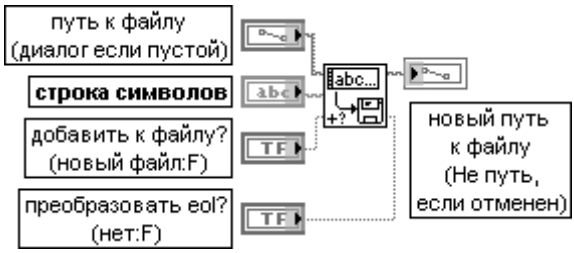
Выход **маркер после чтения** (mark after read) указывает на расположение маркера файла после чтения, на символ в файле, следующий за считанным.

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

Write Characters To File



Записать символы в файл



ВП записывает **строку символов** (character string) в новый файл потока байтов или добавляет строку к существующему файлу. ВП открывает или создает файл перед записью и закрывает его после завершения записи.

На входе **путь к файлу** (file path) задается путь к файлу. Если **путь** пустой (по умолчанию) или **<Не путь>** (<Not A Path>), то ВП отображает диалоговое окно, в котором можно указать файл. При отмене диалогового окна выводится сообщение об ошибке 43.

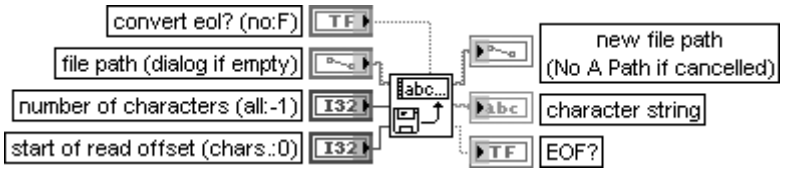
Вход **строка символов** (character string) представляет строку символов, содержащую записываемые в файл данные.

Вход **добавить к файлу?** (append to file?) управляет добавлением данных к существующему файлу при установке состояния ИСТИНА. При установке состояния ЛОЖЬ (по умолчанию) ВП заменяет данные в существующем файле. Если файл отсутствует, то ВП игнорирует значение на входе **добавить к файлу?** и создает новый файл.

Если вход **преобразовать eol?** (convert eol?) установлен в состояние ИСТИНА, то функция преобразует все встречающиеся маркеры конца строки в системные маркеры конца строки, которые в Windows включают символы возврата каретки и следующий за ним символ перевода строки. Если **преобразовать eol?** установлен в состояние ЛОЖЬ (по умолчанию), то функция не преобразует маркер конца строки при его записи. В LabVIEW маркером конца строки является символ перевода строки.

Выход **новый путь к файлу** (new file path) отображает путь к файлу, в который ВП произвел запись данных. Этот выход может использоваться для определения пути к файлу, который был указан пользователем в диалоговом окне. **Новый путь к файлу** будет иметь значение <**Не путь**> (<Not A Path>), если диалоговое окно отменено

Read Characters From File



Считать символы из файла

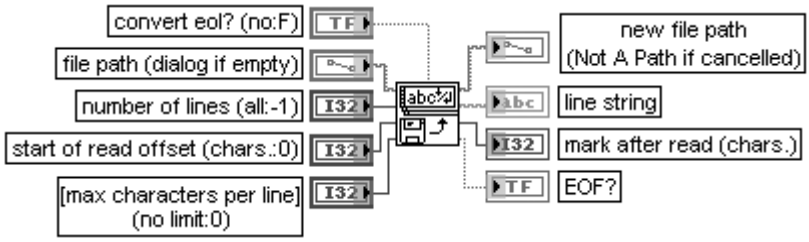
ВП читает определенное число символов файла, представляющего поток байтов, начиная с заданного смещения. ВП открывает или создает файл перед чтением и закрывает его после завершения чтения.

Входы **преобразовать eol?** (convert eol?) и **путь к файлу** (file path) идентичны одноименным входам рассмотренного выше ВП **Записать символы в файл** (Write Characters To File).

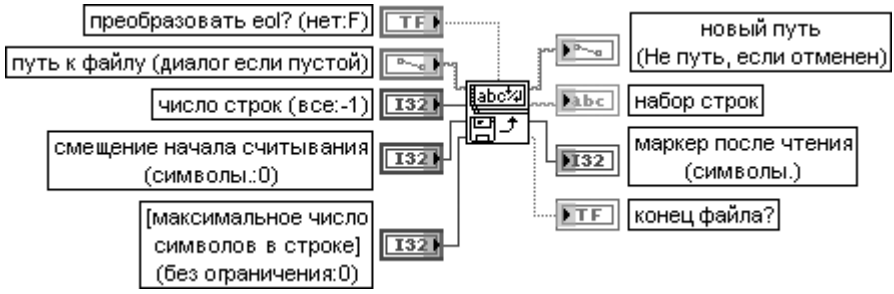
Вход **число символов** (number of characters) задает максимальное число символов, считываемых ВП. Если ВП сначала доходит до конца файла, то он будет считывать меньшее число символов. Если **число символов** меньше 0, то ВП читает весь файл. По умолчанию значение входа равно –1.

Вход **смещение начала считывания** (start of read offset) определяет позицию в файле, выражаемую в символах (или байтах), с которой ВП начинает чтение. В качестве единиц смещения выбраны байты, а не числа, по той причине, что файлы, представляющие поток байтов, могут содержать сегменты данных различного типа. Следовательно, для чтения массива из 100 чисел, которые расположены вслед за заголовком из 57 символов, необходимо установить на входе **смещение начала считывания** значение 57

Read Lines From File



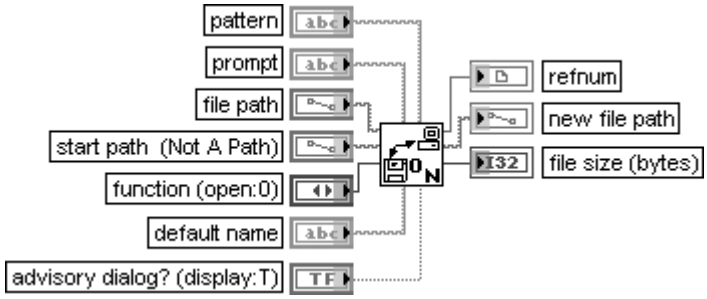
Считать строки из файла



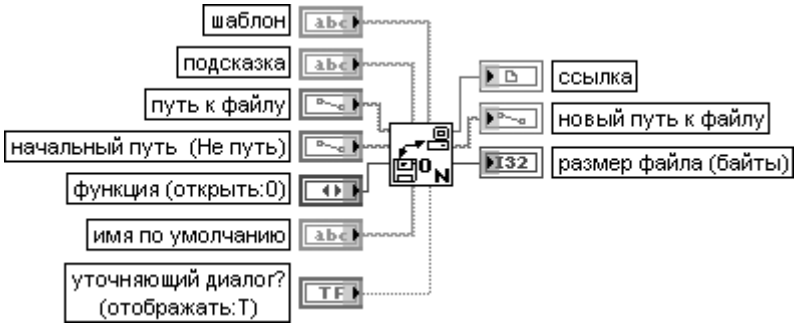
ВП читает определенное число строк файла, представляющего поток байтов, начиная с заданного смещения. ВП открывает или создает файл перед чтением и закрывает его после завершения чтения.

Назначение всех входов и выходов рассмотрено выше при анализе ВП **Считать символы из файла** (Read Characters From File)

Open/Create/Replace File



Открыть/Создать/Заменить файл



ВП открывает существующий файл, создает новый файл или заменяет существующий файл, программно или интерактивно используя файловое диалоговое окно. Пользователь может дополнительно определить **подсказку** диалога (prompt), имя файла по умолчанию, **начальный путь** (start path) или **шаблон имени файла** (pattern). Этот ВП используется, как видно из блок-диаграммы на рис. 2.35, совместно с ВП **Записать файл** (Write File) или **Считать файл** (Read File).

Вход **подсказка** (prompt) представляет сообщение, которое появляется в заголовке файлового диалогового окна. По умолчанию это пустая строка.

Вход **начальный путь** (start path) задает начальный путь, указывающий путь к каталогу или папке, первоначально отображаемым в файловом диалоговом окне.

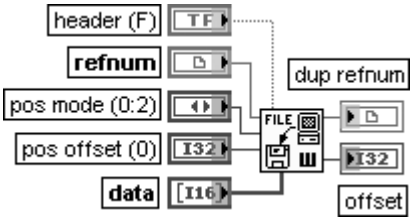
Вход **функция** (function) определяет выполняемую функцию и имеет следующие варианты:

0	**открыть** (open) – открыть существующий файл
1	**открыть или создать** (open or create) – открыть существующий или создать новый файл, если открываемый не существует
2	**создать или заменить** (create or replace) – создать новый файл или заменить существующий при наличии разрешения
3	**создать** (create) – создать новый файл
4	**открыть (только для чтения)** (open (read only)) – открыть существующий файл только для чтения

Вход **имя по умолчанию** (default name) определяет имя по умолчанию, появляющееся в строке **имя файла** файлового диалогового окна.

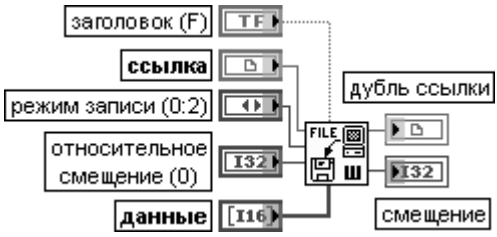
Следующие входы являются необязательными (optional).
Вход **шаблон** (pattern) задает шаблон имен файлов, который ограничивает перечень файлов, отображаемых в диалоговом окне. **Шаблон** не ограничивает отображаемые каталоги. Проверка соответствия шаблону в этой функции аналогична проверке шаблонов имен файлов в Windows и UNIX. 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 используются ***.html;_*.doc**, будут отображены только файлы, заканчивающиеся **.html**. Здесь символ _ показывает расположение пробела.
Вход **уточняющий диалог?** (advisory dialog?) определяет необходимость уточняющего диалога. Если на этом входе установлено состояние ИСТИНА (по умолчанию), то ВП отображает диалоговое окно, если на входе **функция** установлены значения 0 или 4 и файл не существует, или когда на входе **функция** установлены значения 2 или 3 и файл существует.
Выход **ссылка** (refnum) отображает ссылку на открытый файл. Если файл не может быть открыт, то на выход выводится значение **Не ссылка** (Not A Refnum).
Выход **новый путь к файлу** (new file path) представляет путь к открытому или созданному файлу.
Выход **размер файла** (file size) отображает размер файла в байтах. Значение выхода представляет также положение конца файла

Write File



Записать файл

а)



б)

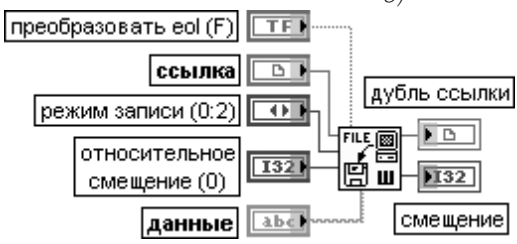


*Рис. 2.36. Варианты подключения функции **Записать файл** (Write File)*

Функция записывает данные в открытый файл, определенный с помощью **ссылки** (refnum). Запись начинается с позиции, определенной с помощью входов **режим записи** (pos mode) и **относительное смещение** (pos offset) для **файлов потока байтов** (byte stream files) (рис. 2.36а, б) и с конца файла для **файлов протокола** (datalog files) (рис. 2.36в). Входы **данные** (data), **заголовок** (header) и формат записываемого файла определяют количество записываемых данных.

Вход **ссылка** (refnum) определяет ссылку к файлу, в который производится запись. Если ссылка определяет **файл потока байтов**, то эта функция записывает данные в файл с позиции, определяемой входами **режим записи** и **относительное смещение**. Если данные на входе **данные** относятся к типам с переменной длиной, таким как строка или массив, то функция может записать **заголовок** файла (header), который определит размер данных. LabVIEW устанавливает файловый маркер в следующую за последним записанным байтом позицию.

Если ссылка определяет **файл протокола**, то функция записывает данные как записи в файл. Пользователь может только добавлять данные к файлам протокола, поэтому функция всегда начинает запись с конца этого файла. Следовательно, LabVIEW устанавливает маркер файла на запись, следующую за последней записью. Пользователь не должен подключать входы **преобразовать eol** (convert eol), **заголовок** (header), **режим записи** (pos mode) или **относительное смещение** (pos offset) для файла протокола (рис. 3.36в).

Вход **данные** содержит данные, записываемые в файл, и может быть произвольного типа. Для выполнения символьно-ориентированного ввода/вывода на входе **данные** необходимо установить строковый тип данных, не подключая при этом вход **заголовок**, поскольку по умолчанию он находится в состоянии ЛОЖЬ (рис. 3.36б). ВП записывает символы **данных** в виде последовательности без какой-либо заголовочной информации.

Если ссылка представляет ссылку к файлу протокола, то вход **данные** должен иметь такой тип данных, который бы соответствовал типу данных, определенному при открытии или создании файла, либо массиву данных такого типа. В первом случае эта функция записывает данные как простую запись в файл протокола. Если необходимо, функция приводит числовые данные к представлению, соответствующему установленному типу данных. В последнем случае эта функция записывает каждый элемент данных как отдельную запись в файл протокола в порядке возрастания элементов.

Вход **заголовок** (header) может быть подключен только когда данные, подаваемые на вход **данные**, относятся к типу данных с переменной длиной, то есть являются строками или массивами.

Если вход **заголовок** установлен в состояние ИСТИНА, то функция записывает данные на входе **данные** вместе с заголовком, определяющим их длину. Вход **заголовок** должен быть установлен в состояние ИСТИНА для записи динамических структур. Если **заголовок** установлен в состояние ЛОЖЬ (по умолчанию), то эта функция записывает данные без заголовка.

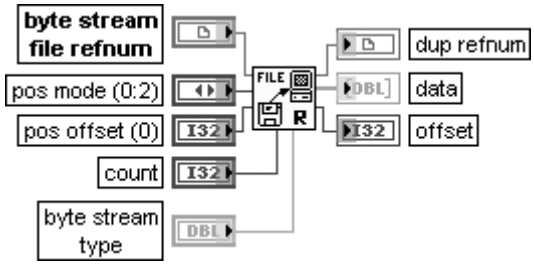
Вход **режим записи** вместе с входом **относительное смещение** определяют начало операции записи. Если вход **относительное смещение** подключен, то вход **режим записи** по умолчанию устанавливается в 0 и смещение определяется относительно начала файла. Если вход **относительное смещение** не подключен, то он устанавливается в 0, на входе **режим записи** по умолчанию устанавливается 2, и операция начинается с текущей позиции файлового маркера. Значения и описание режимов позиционирования приведены в таблице:

0	**начало** (start)– операция начинается с позиции, задаваемой входом **относительное смещение** относительно начала файла. В этом случае **относительное смещение** должно быть положительным
1	**конец** (end) – операция начинается с позиции, задаваемой входом **относительное смещение** относительно конца файла. В этом случае **относительное смещение** должно быть отрицательным
2	**текущее** (current) – операция начинается с текущего положения файлового маркера плюс **относительное смещение**

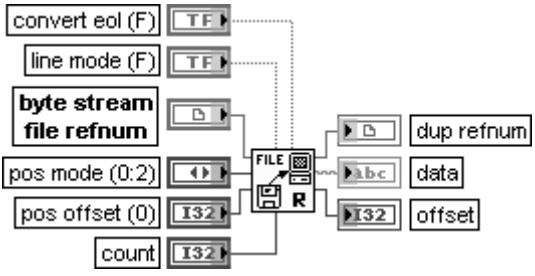
Вход **относительное смещение** определяет, на каком расстоянии от позиции, задаваемой параметром **режим записи**, необходимо начинать запись. По умолчанию значение на входе равно 0

Read File

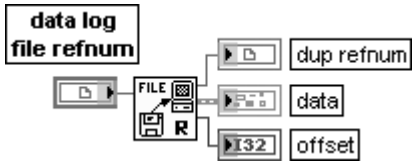
а)



б)

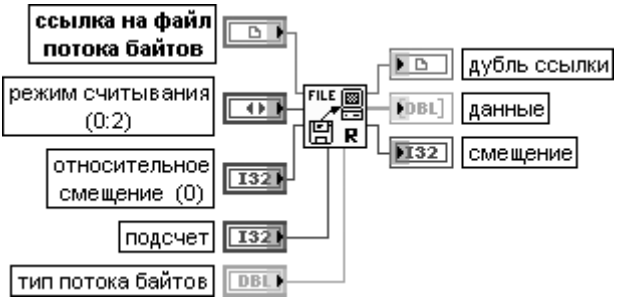


в)

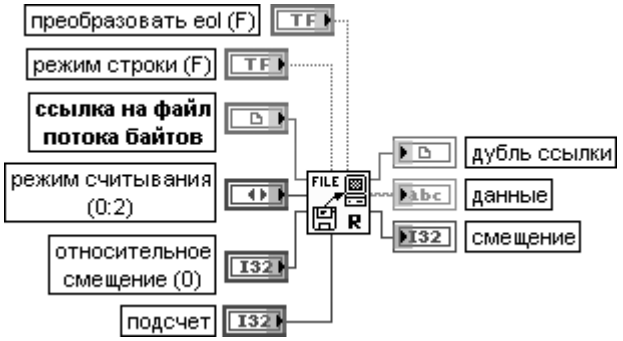


Считать файл

а)



б)



в)

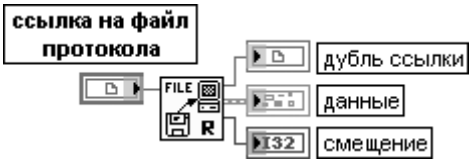


*Рис. 2.37. Варианты подключения функции **Считать файл** (Read File)*

Функция считывает данные из открытого файла, определяемого **ссылкой** (refnum), и возвращает эти данные на выходе **данные** (data). Чтение начинается с текущего положения маркера или положения в файле, определяемого параметрами на входах **режим считывания** (pos mode) и **относительное смещение** (pos offset). Способ чтения данных зависит от формата заданного файла.

Вход **ссылка** (refnum) определяет ссылку к файлу, из которого производится чтение. При необходимости чтения более одного элемента следует задать значение на входе **подсчет** (count) и использовать входы **режим считывания**, **относительное смещение** и выход **смещение** (offset) для отслеживания положения файлового маркера (рис. 2.37а, рис. 2.37б). Если **ссылка** относится к **файлу протокола** (datalog file) (рис. 2.37в), то функция считывает данные как записи из файла и возвращает одну или массив записей на выходе **данные**.

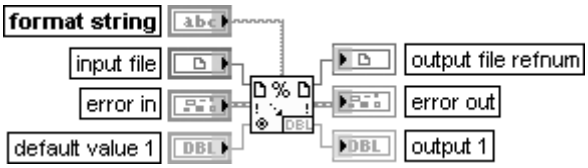
Вход **режим считывания** идентичен входу **режим записи** рассмотренной выше функции **Записать файл**.

Если входы **режим считывания** или **относительное смещение** не подключены, то функция читает данные, следующие за последними считанными данными.
Вход **подсчет** (count) задает число считываемых элементов данных. Элементы данных могут быть байтами, экземплярами потока байтов или записями файла протокола. Функция возвращает заданное на входе **подсчет** число элементов данных на выходе **данные** или, если был обнаружен конец файла, возвращает все полные считанные элементы данных и ошибку конца файла. По умолчанию функция возвращает простой элемент данных. Функция возвращает ошибку, если значение **подсчет** меньше 0.
Вход **режим строки** (line mode) определяет режим завершения чтения. Если на входе **режим строки** установлено значение ИСТИНА, то функция читает до обнаружения маркера конца строки или конца файла, или до считывания числа символов, заданных на входе **подсчет**, если он подключен и больше 0. Если на входе **режим строки** установлено значение ЛОЖЬ (по умолчанию), то функция читает число символов, заданное на входе **подсчет,** или, если вход **подсчет** не подключен, число читаемых символов равно 0.
Рассмотренный параметр применим только к текстовым файлам. Не следует подключать его при работе с двоичными файлами.
Выход **данные** (data) содержит данные, считанные из файла с определенным типом данных. Он может содержать строку, массив, кластер массивов, массив кластеров в зависимости от типа считываемых данных и вида установки входа **подсчет**.
Выход **смещение** (offset) отображает новое положение маркера файла относительно позиции, определенной параметром **режим считывания**. Смещение выражается в тех же единицах, что и относительное смещение, то есть в записях для файла протокола и в байтах для файла потока байтов.
Таким образом, в зависимости от подключения входов **ссылка**, **подсчет** и **тип потока байтов** рассматриваемая функция производит считывание по следующим вариантам.
1. Ссылка относится к **файлу протокола**, а:
- вход **подсчет** не подключен, то **данные** представляют одну запись;
- к входу **подсчет** подключен скаляр ***m***, то **данные** представляют одномерный массив из ***m*** записей;
- к входу **подсчет** подключен кластер из скаляров ***p***, ***q*** и ***r***, то **данные** представляют массив записей размером [***p*** x ***q*** x ***r***].

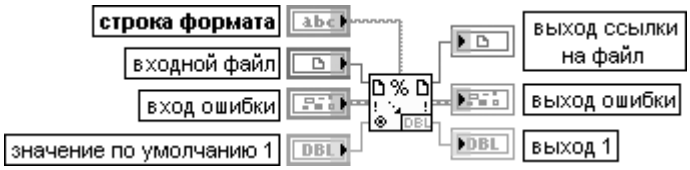
2. Ссылка относится к **файлу потока байтов** и **тип потока байтов** подключен, а:
- вход **подсчет** не подключен, то **данные** представляют один экземпляр потока байтов;
- к входу **подсчет** подключен скаляр ***m***, то **данные** представляют одномерный массив экземпляров потока байтов;
- к входу **подсчет** подключен кластер из скаляров ***p***, ***q*** и ***r***, то **данные** представляют массив записей потока байтов размером [***p*** x ***q*** x ***r***].

3. Ссылка относится к **файлу потока байтов** или текстовому файлу и **тип потока байтов** не подключен, а:
- вход **подсчет** не подключен, то **данные** представляют строку, содержащую 0 байтов;
- к входу **подсчет** подключен скаляр ***m***, то **данные** представляют строку, содержащую ***m*** байтов;
- к входу **подсчет** подключен кластер из скаляров ***p***, ***q*** и ***r***, то провод к этому входу будет разорван

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 сканирования. Эта функция может быть использована для чтения всего текста в файле. Конечно, данная функция не может быть использована для определения начальной точки сканирования. 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 это ссылка, то данная функция открывает файл, определенный этой ссылкой. По умолчанию предполагается 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.

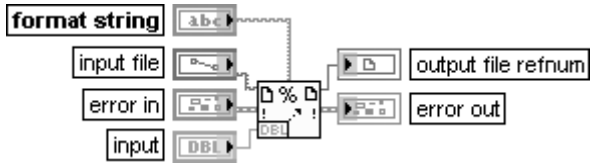
Выходы 1…n (output 1...n) определяют выходные параметры. Каждый выход может быть строкой, путем, типом перечисления или каким-либо числовым типом. С этой функцией не могут быть использованы массивы и кластеры.

В таблице приведены примеры использования функции **Просмотр файла** (Scan From File).

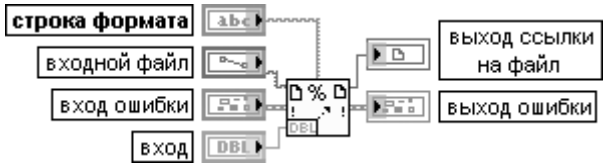
входная строка	строка формата	по умолчанию	выходы	оставшаяся строка
abc, xyz	%3s,	–	abc	00
12,3+56i 7200	%s%f%2d	–	xyz	
		0,00 + 0,00 i	12,30 + 56,00 i	
		–	72	
Q+1.27E-3 tail	Q%f t	–	1,27E-3	ail
0123456789	%3d%3d	–	12	6789
			345	
X:9,860 Z:3,450	X:%fY:%f	100(I32)	10	Z: 3,450
		100,00(DBL)	100,00	
set49,4,2	set%d	–	49	,4,2
color: red	color: %s	blue (enum {red, green, blue})	red	–

входная строка	строка формата	по умолчанию	выходы	оставшаяся строка
abcd012xyz3	%[a-z]%d%[a-z]%d	–	abcd 12 xyz 3	–
welcome to LabVIEW, John Smith	%[^,],%s	–	welcome to LabWIEW John	Smith

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, как преобразовывать входные аргументы. 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, перечисляемым типом или каким-либо числовым типом. С этой функцией не могут использоваться массивы и кластеры.

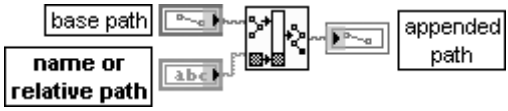
Выход ссылки на файл (output file refnum) представляет ссылку на файл, который ВП читает. Этот выход может быть подключен к другой файловой функции, зависящей от предполагаемых действий с файлом. По умолчанию это функция закрытия файла, если ссылка получена из файлового пути или диалогового окна. Если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

Увеличение числа входных параметров производится с помощью выбора функции **Добавить параметр** (Add Parameter) из контекстного меню или увеличением размера иконки функции по вертикали с помощью инструмента перемещения.

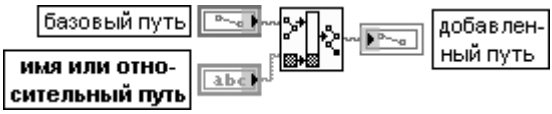
Пользователь может применять рассмотренную функцию для определения порядка, в котором данные окажутся в файле. Эту функцию нельзя использовать для добавления данных

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

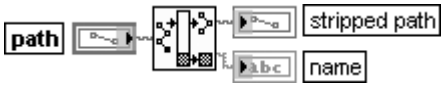


Сформировать путь

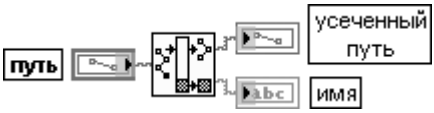


Функция формирует **добавленный путь** (appended path) путем добавления **имени** или **относительного пути** (name or relative path) к существующему **базовому пути** (base path)

Strip Path



Разделить путь



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

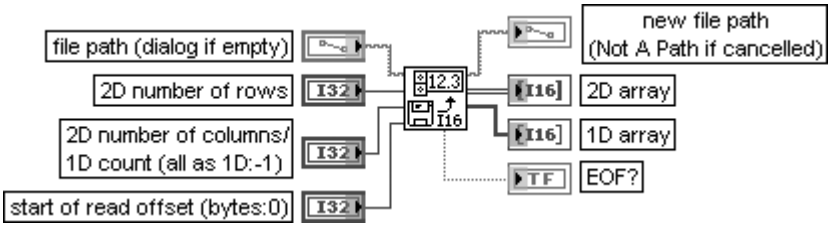
Close File

Закрыть файл

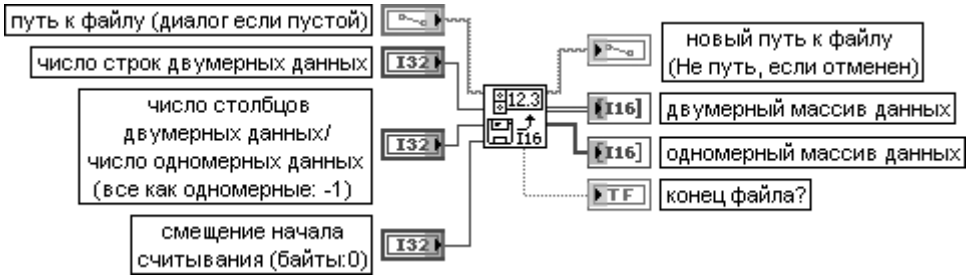
Функция закрывает открытый файл, определяемый **ссылкой** (refnum), и возвращает **путь** (path) к файлу, соответствующему ссылке. Ошибка ввода/вывода формируется только в этой функции, которая закрывает файл, несмотря на ошибки в предыдущих операциях. Это гарантирует, что файлы закрыты корректно

Подпалитра **ВП двоичных файлов (**Binary File VIs**)** содержит две пары идентичных ВП записи/чтения двоичных файлов.

Read From I16 File



Считать данные из файла в форме I16



ВП читает двумерные или одномерные данные в форме I16 из файла с потоком байтов. ВП открывает файл перед считыванием и закрывает его после завершения считывания. Предполагается, что данные в файл были записаны с помощью ВП **Записать данные в файл в форме I16** (Write To I16 File).

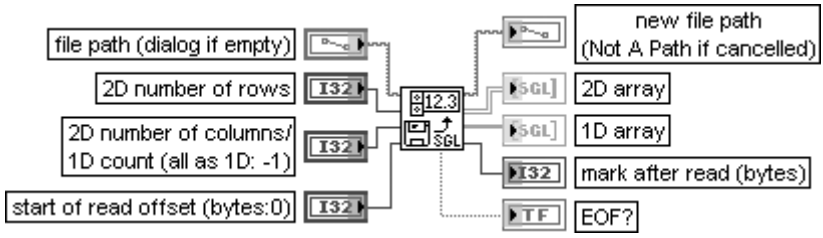
Вход **число строк двумерных данных** (2D number of rows) передает число строк считываемого двумерного массива. По умолчанию значение на этом входе равно 0.

Вход **число столбцов двумерных данных/число одномерных данных** (2D number of columns/1D count) передает число столбцов при считывании данных в виде двумерного массива или число элементов считываемого одномерного массива. По умолчанию значение на этом входе равно –1. Функции всех остальных входов и выходов были рассмотрены выше.

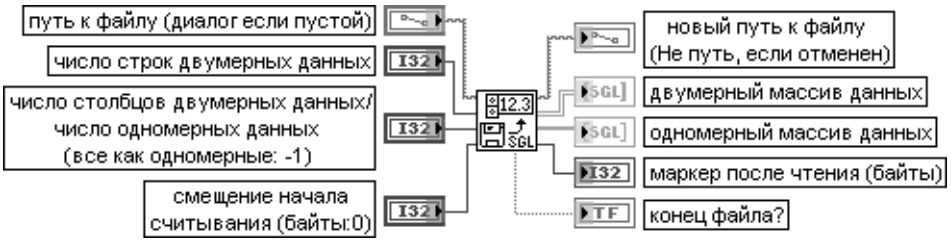
ВП передает на выход **двумерный** (2D array) или **одномерный** (1D array) массивы в соответствии с правилами, указанными в таблице:

Число строк двумерных данных	Число столбцов двумерных данных/число одномерных данных	Выход
Не имеет значения	N<0 (по умолчанию)	Весь файл в **одномерный массив** (1D array) (по умолчанию)
0 (по умолчанию)	N>0	N элементов в **одномерный массив**
<0	N>0	Весь файл в **двумерный массив**; ВП вычисляет число строк как целая часть результата деления **размер массива**/N
M>0	N>0	Данные размером M×N в **двумерный массив**
Другие комбинации		Пустые массивы

Read From SGL File

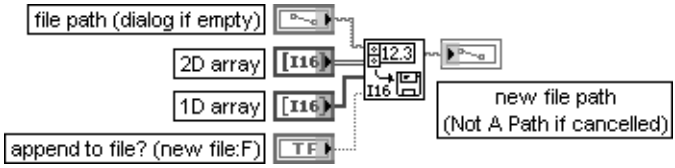


Считать данные из файла с одинарной точностью

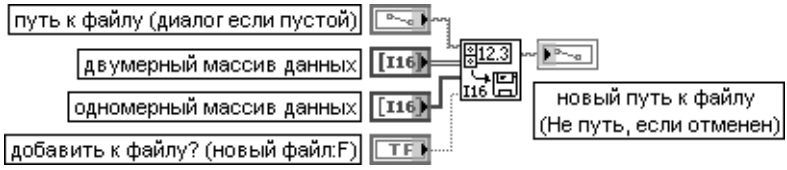


ВП читает двумерные или одномерные данные с одинарной точностью из файла с потоком байтов.
ВП открывает файл перед считыванием и закрывает его после завершения считывания. Предполагается, что данные в файл были записаны с помощью ВП **Записать данные в файл с одинарной точностью** (Write To SGL File).
Функции всех входов и выходов идентичны функциям одноименных входов и выходов ВП **Считать данные из файла в форме I16** (Read From I16 File), рассмотренного выше, за исключением того, что выходные данные в одномерных или двумерных массивах представлены в форме с одинарной точностью

Write To I16 File

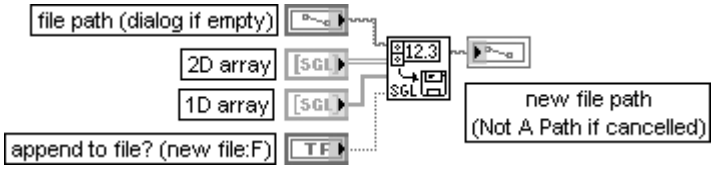


Записать данные в файл в форме I16

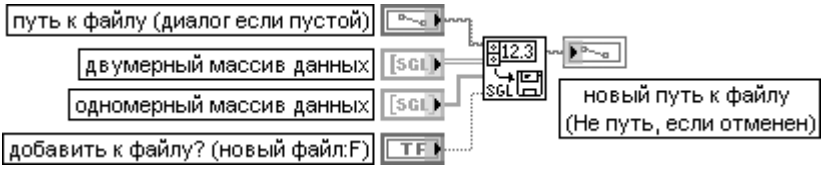


ВП записывает в новый файл потока байтов или добавляет к существующему файлу двумерные или одномерные данные в форме I16.
Функции всех входов и выходов данного ВП были рассмотрены выше при анализе ВП ввода/вывода файлов

Write To SGL File



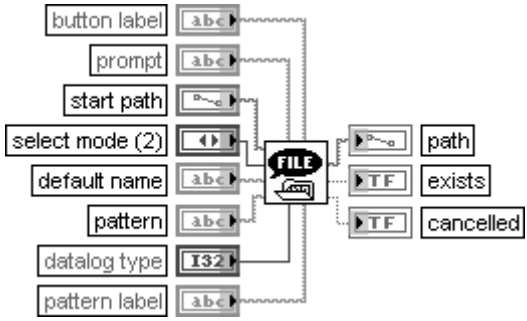
Записать данные в файл с одинарной точностью



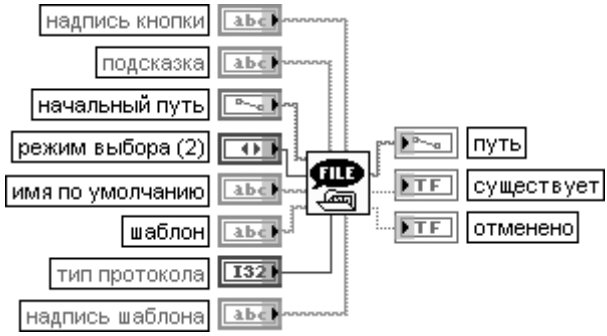
ВП записывает в новый файл потока байтов или добавляет к существующему файлу двумерные или одномерные данные с одинарной точностью.
Функции всех входов и выходов данного ВП были рассмотрены ранее

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

File Dialog



Файловый диалог



Функция 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. 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 **режим выбора** (select mode) устанавливает типы файлов или каталогов, которые пользователь может выбрать в файловом диалоговом окне. Предусмотрены следующие режимы выбора:

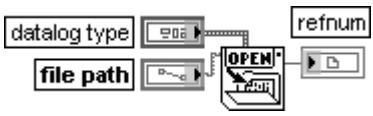
0	**существующий файл** (existing file) – пользователь может выбрать только существующий файл. Это значение позволяет пользователю выбрать открываемый файл. ВП может после этого открыть файл, используя функцию **Открыть файл** (Open File)
1	**новый файл** (new file) – пользователь может выбрать несуществующий файл. Это значение позволяет пользователю указать имя создаваемого файла. ВП может затем создать файл, используя функцию **Создать файл** (New File)
2	**новый или существующий файл** (new or existing file) (по умолчанию) – пользователь может выбрать существующий или несуществующий файл. Это значение позволяет пользователю выбрать имя создаваемого файла или указать файл, в который будут добавлены данные
3	**существующий каталог** (existing dir) – пользователь может выбрать только существующий каталог. Это значение позволяет пользователю выбрать имя каталога, который содержит файлы данных. ВП затем может получить доступ к этим файлам с помощью пути каталога
4	**новый каталог** (new dir) – пользователь может выбрать только каталог, который не существует. Это значение позволяет пользователю указать имя каталога, который ВП затем может сформировать, используя функцию **Создать каталог** (New Directory)
5	**новый или существующий каталог** (new or existing dir) – пользователь может выбрать существующий или несуществующий каталог. Данное значение позволяет пользователю выбрать имя каталога, в котором будут расположены файлы данных. ВП затем может создать каталог, если он не существует
6	**существующий файл** (используя файлы библиотеки) (existing file (use LLBs)) – пользователь может выбрать существующий файл, включая файлы в библиотеке
7	**новый файл** (используя файлы библиотеки) (new file (use LLBs)) – пользователь может выбрать файл, который не существует, включая файлы в библиотеке
8	**новый или существующий файл** (используя файлы библиотеки) (new or existing file (use LLBs)) – пользователь может выбрать существующий или несуществующий файл, включая файлы в библиотеке
9	**новый или существующий файл или каталог** (используя файлы библиотеки) (new or existing file or directory (use LLBs)) – пользователь может выбрать существующий или несуществующий файл или каталог, включая файлы в библиотеке

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.
Следующие входы являются **дополнительными** (optional).
Вход **надпись кнопки** (button label) представляет надпись, отображаемую на кнопке OK или Select Cur Dir файлового диалогового окна. Если **режим выбора** позволяет пользователю выбирать каталоги, этот вход можно использовать для определения надписи на кнопке Select Cur Dir. Если **режим выбора** не позволяет пользователю выбирать каталоги, то можно использовать этот вход для определения надписи на кнопке OK. Например, если на входе **режим**

выбора установлен 0 и пользователь должен выбрать существующий файл, к которому необходимо добавить данные, то на входе **надпись кнопки** можно задать строку **добавить** (Append).
Вход **подсказка** (prompt) представляет сообщение, которое появляется ниже списка файлов и каталогов в диалоговом окне. По умолчанию это пустая строка.
Вход **тип протокола** (datalog type) может быть любого типа и ограничивает файлы, отображаемые в диалоговом окне, файлами протокола, содержащими записи определенного типа.
Вход **надпись шаблона** (pattern label) определяет подпись, отображаемую в файловом диалоговом окне вслед за **шаблоном** (pattern).
Выход **путь** (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 **отменено** (cancelled) устанавливается в состояние ИСТИНА, если диалоговое окно отменено или при его использовании произошла ошибка

Open File **Открыть файл**

а)



б)

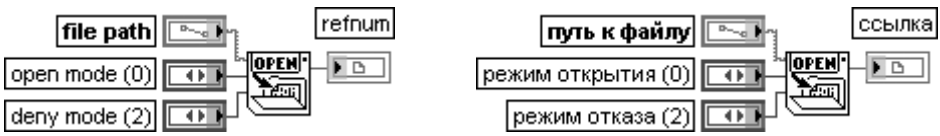


*Рис. 2.38. Варианты подключения функции **Открыть файл** (Open File)*

Функция открывает существующий файл для чтения или записи. Эта функция не может использоваться для создания или замены файла. Для закрытия ссылки к файлу необходимо использовать функцию **Закрыть файл** (Close File).
Вход **тип данных протокола** (datalog type) подключается только при открытии файлов протокола (рис. 2.38а) и может быть любого типа. При открытии файла протокола к этому входу необходимо подключить кластер, состав и порядок элементов которого соответствуют типу данных записей файла протокола. Подключение такого кластера позволяет LabVIEW считать, что заданный файл является файлом протокола с заданным типом записей. В этом случае **ссылка** (refnum) является ссылкой файла протокола. Если формат файла не будет соответствовать формату файла протокола, то функция вернет ошибку. По умолчанию предполагается, что заданный файл является файлом потока байтов с соответствующим изменением типа ссылки и подключением входов (рис. 2.38б).
Вход **режим открытия** (open mode) определяет, открывает ли эта функция файл для чтения и записи, только для чтения или только для записи. Предусмотрены следующие режимы:

0	**чтение/запись** (read/write) разрешает чтение и запись в файл (по умолчанию)
1	**только чтение** (read-only) разрешает только чтение из файла
2	**только запись** (write-only) разрешает только запись в файл Не может сокращать файл (удалять все данные)

3	**только запись** (write-only (truncate)) разрешает только запись в файл. Может сокращать файл (удалять все данные)

Вход **режим отказа** (deny mode) определяет права других пользователей для одновременной работы с файлом. Предусмотрены следующие режимы:

0	**запрет чтения/записи** (deny read/write) запрещает другим пользователям чтение и запись файла
1	**запрет только записи** (deny write-only) разрешает чтение, но запрещает запись в файл другим пользователям
2	**отсутствие запрета** (deny none) разрешает чтение и запись в файл другим пользователям (по умолчанию)

New File **Создать файл**

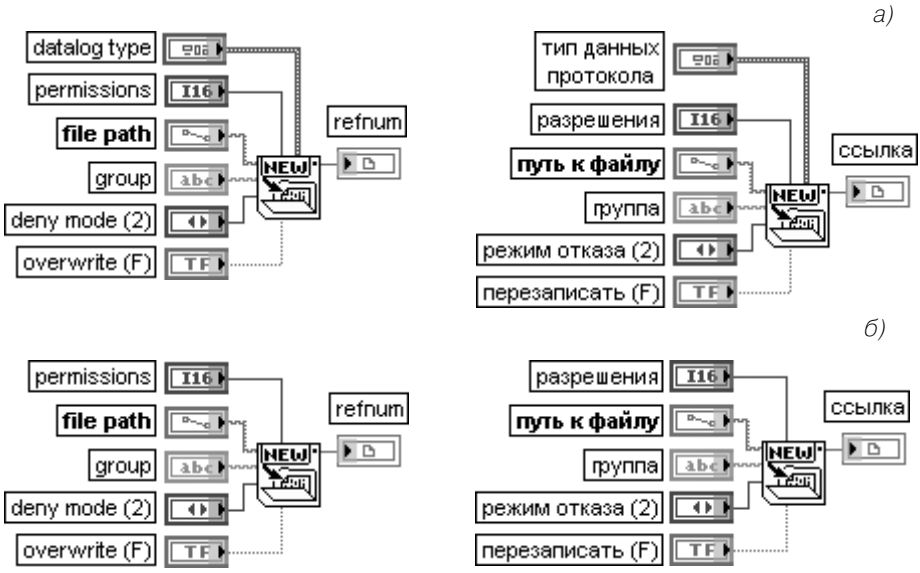


*Рис. 2.39. Варианты подключения функции **Создать файл** (New File)*

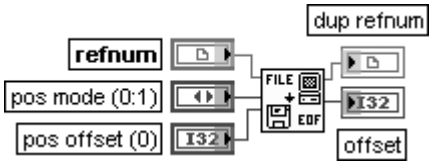
Функция создает новый файл и открывает его для чтения или записи. Новый файл доступен для записи независимо от установленного **разрешения** (permissions), но данная функция не может перезаписывать существующий файл, пока вход **перезаписать** (overwrite) находится в состоянии ИСТИНА и существующий файл не является доступным только для чтения. При попытке перезаписи файла без корректного разрешения данная функция не будет создавать или открывать файл и вернет ошибку.

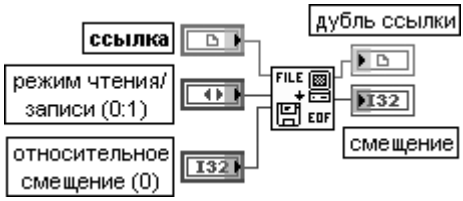
Подключение входа **тип данных протокола** (datalog type) идентично подключению одноименного входа функции **Открыть файл**, рассмотренному выше, и также приводит к формированию ссылки файла протокола (рис. 2.39а) или файла потока байтов (рис. 2.39б).

Вход **разрешения** (permissions) определяет файловые системные права доступа, назначаемые новому каталогу или файлу. По умолчанию эти права соответствуют системным, установленным также по умолчанию.

Вход **группа** (group) содержит групповые установки для файла или каталога, после того как данная функция будет выполнена. Если на вход **группа** подключена пустая строка, функция использует пользовательские установки по умолчанию.
Вход **перезаписать** (overwrite) определяет, заменяет ли функция файл или возвращает ошибку, если **путь к файлу** уже существует. Если на входе **перезаписать** установлено значение ЛОЖЬ (по умолчанию), то функция возвращает ошибку дублирования пути. При установке на входе **перезаписать** значения ИСТИНА функция заменяет файл, если он не является доступным только для чтения. Если файл доступен только для чтения, функция возвращает ошибку.
Функции **Создать файл** (New File) и **Создать каталог** (New Directory) позволяют управлять доступом к создаваемым файлам и каталогам. Эти функции включают параметр **разрешения** (permissions), представляющий 16-битовое целое число. LabVIEW использует 9 младших значащих бит для определения доступа к файлу или папке. Данный параметр может быть сформирован с помощью функции **Права доступа** (Access Rights) или с помощью константы либо элемента управления

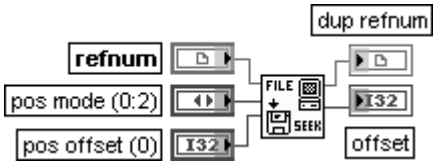
EOF **Конец файла**

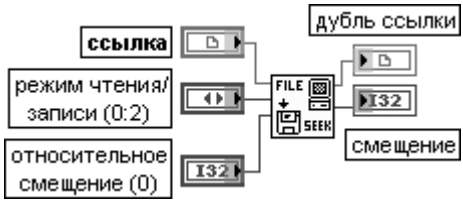




Функция устанавливает или возвращает **конец файла** (EOF (end-of-file)) для файла, заданного с помощью **ссылки** (refnum). Входы **режим чтения/записи** (pos mode) и **относительное смещение** (pos offset) определяют новое положение EOF. Эта функция всегда возвращает положение EOF относительно начала файла

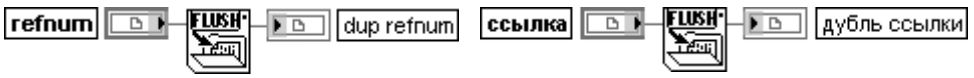
Seek **Поиск**





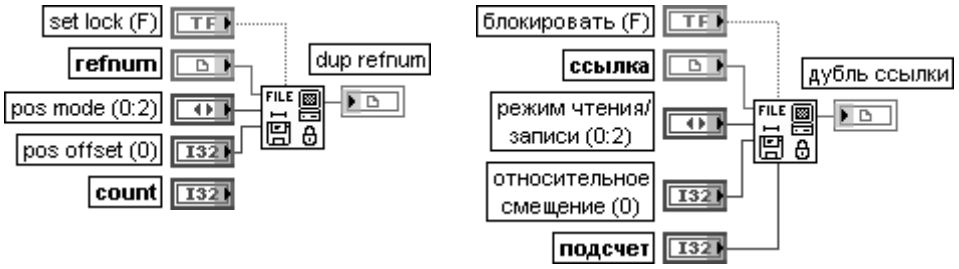
Функция перемещает текущий файловый маркер файла, который определен с помощью **ссылки** (refnum), в позицию, указанную на входе **относительное смещение** (pos offset), в соответствии с режимом, установленным на входе **режим чтения/записи** (pos mode). Эта функция перемещает файловый маркер только когда оба входа **режим чтения/записи** и **относительное смещение** операций чтения или записи не подключены. Если необходимо переместить файловый маркер при использовании функций **Считать файл** (Read File) и **Записать файл** (Write File), то следует установить **режим чтения/записи** и **относительное смещение** на входах этих функций вместо использования функции **Seek**

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

Lock Range **Блокировать диапазон**



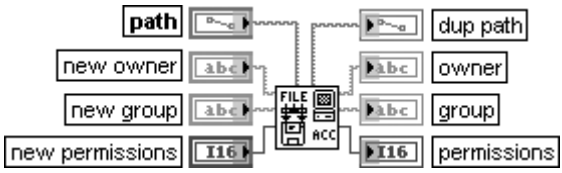
Функция блокирует или разблокирует область файла, определенного с помощью **ссылки** (refnum).

Если вход **блокировать** (set lock) установлен в состояние ИСТИНА, то данная функция блокирует определенную область данных заданного файла. Если две блокированные области перекрываются, то функция воспринимает их как одну заблокированную область.

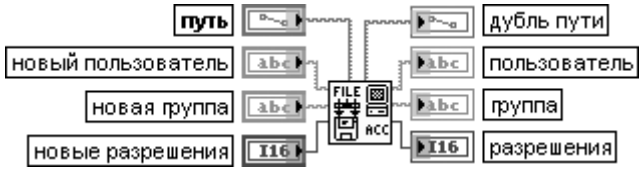
Если вход **блокировать** установлен в состояние ЛОЖЬ (по умолчанию), то функция разблокирует определенную область данных заданного файла. Если разблокирование части блокированной области оставляет часть данных на любом конце блокированными, то функция воспринимает эти два набора данных как отдельные блокированные области.

Вход **подсчет** (count) задает число байтов, которые данная функция блокирует или разблокирует. Диапазон блокирования или разблокирования может выходить за конец файла, чтобы гарантировать невозможность добавления данных в файл другими пользователями во время добавления данных самим пользователем

Access Rights

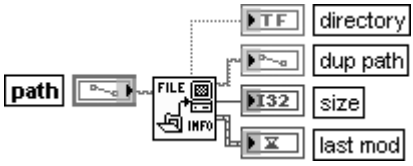


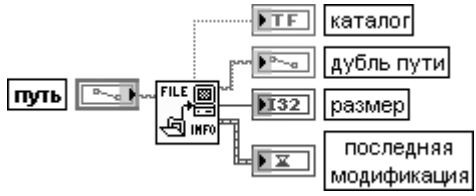
Права доступа



Функция устанавливает и возвращает пользователя, группу и разрешение доступа к файлу или каталогу, определенным с помощью **пути** (path).
Вход **новый пользователь** (new owner) определяет нового пользователя файла или каталога. Если на входе **новый пользователь** подключена пустая строка, функция не устанавливает нового пользователя.
Вход **новая группа** (new group) определяет новую группу для файла или каталога. Если на входе **новая группа** подключена пустая строка, функция не устанавливает новую группу.
Вход **новые разрешения** (new permissions) определяет новые разрешения для файла или каталога.
Выход **дубль пути** (dup path) возвращает **путь пользователя** (path owner), содержащий текущие установки пользователя для файла или каталога после выполнения данной функции.
Выход **группа** (group) содержит текущие установки группы для файла или каталога после выполнения данной функции.
Выход **разрешения** (permissions) содержит текущие установки разрешения для файла или каталога после выполнения данной функции

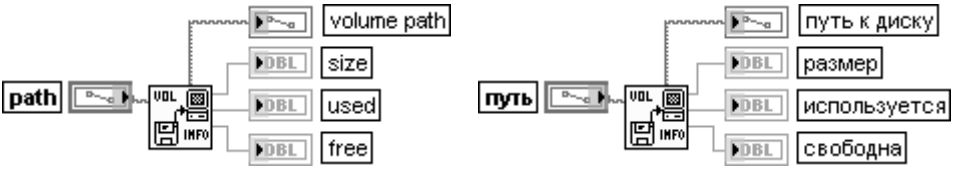
File/Directory Info **Информация о файле или каталоге**





Функция возвращает информацию о файле или каталоге, определенным с помощью **пути** (path), включая **размер** (size), дату их последней модификации и информацию о принадлежности объекта к каталогу. 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.
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

Volume Info **Информация о диске**



Функция возвращает информацию о диске, содержащем файл или каталог, которые заданы с помощью **пути** (path), включая общий объем памяти, предоставляемый диском, объем используемой памяти 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.
Выход **размер** (size) отображает объем памяти в байтах, предоставляемый определенным диском.
Выход **используется** (used) отображает объем используемой памяти в байтах на выбранном диске.
Выход **свободна** (free) отображает объем свободной памяти в байтах на выбранном диске

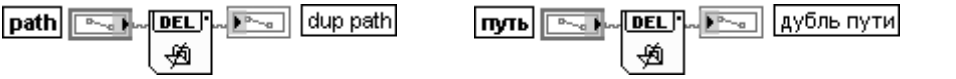
Move **Переместить**

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 файлы или каталоги

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

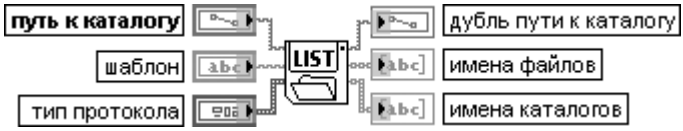
Delete **Удалить**



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

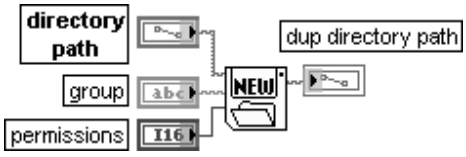
List Directory

Печать каталога

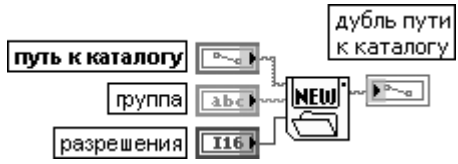


Функция возвращает два массива строк, в которых перечисляются имена всех файлов и каталогов, найденных по **пути к каталогу** (directory path). Имена обоих массивов фильтруются на основе **шаблона** (pattern), а массив **имен файлов** (file names) фильтруется на основе типа протокола, определенного на входе **тип протокола** (datalog type)

New Directory



Создать каталог



Функция программно создает каталог, определяемый на входе **путь к каталогу** (directory path). Если файл или каталог уже существуют в заданном месте, то данная функция возвращает ошибку вместо перезаписи существующего файла или каталога

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

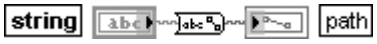
Array of Strings to Path **Массив строк в путь**

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**

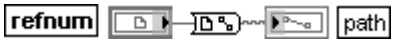
Функция преобразует **путь** (path) в **строку** (string), описывающую путь в стандартном формате платформы.
Вход **путь** может быть путем, массивом путей, кластером путей или массивом кластеров путей, которые необходимо преобразовать в строку. Если на вход **путь** подано <Not A Path>, функция устанавливает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



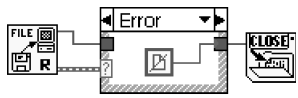
Функция возвращает **путь** (path), соответствующий определенной **ссылке** (refnum). Вход **ссылка** представляет ссылку открытого файла, путь к которому определяется данной функцией. Если **ссылка** не является достоверной ссылкой, то данная функция устанавливает на выходе **путь** значение <Not A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

Функции низкого уровня служат для выполнения отдельных разделов функций

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
	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)



высокого уровня и, помимо этого, таких файловых операций, как создание каталогов, перемещение, копирование или удаление файлов, вывод содержания каталогов, изменение файловых характеристик и манипулирование с путями.

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

Перечень и функции файловых констант приведены в следующей таблице

В состав палитры функций ввода/вывода файлов входят Экспресс-ВП **Записать файл результатов измерения LabVIEW** (Write LabVIEW Measurement

Записать файл результатов измерения LabVIEW (Write LabVIEW Measurement File)

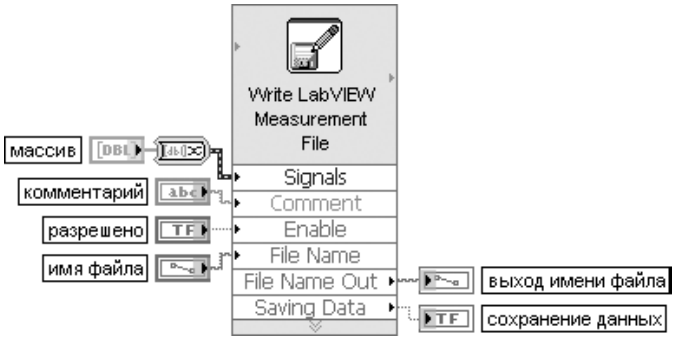


Рис. 2.40. Блок-диаграмма возможного подключения Экспресс-ВП

Экспресс-ВП записывает данные в файл результатов измерения LabVIEW с расширением .lvm.
Диалоговое окно данного Экспресс-ВП имеет следующие опции (рис. 2.41).

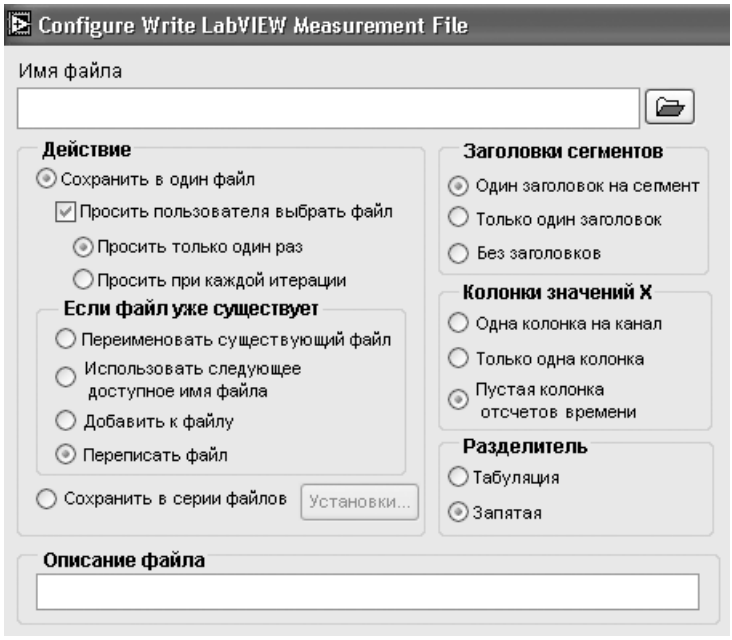


*Рис. 2.41. Вид диалогового окна конфигурирования Экспресс-ВП **Записать файл результатов измерения LabVIEW** (Write LabVIEW Measurement File)*

Имя файла (File name) отображает полный путь к файлу, в который записываются данные.
Действие (Action) содержит следующие опции:

- **Сохранить в один файл** (Save to one file) – сохраняет все данные в один файл.

- **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 отметки в окне **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) – отображает диалоговое окно **Конфигурировать установки сохранения в нескольких файлах** (Configure Multi-files Settings). Эта опция доступна только при установке отметки в окне **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 файле, в который LabVIEW записывает данные.
- **Без заголовков** (No headers) – не создает заголовка в файле, в который LabVIEW записывает данные.

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. 2.40):

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), **Форматирование для записи в файл** (Format Into File)

Считать из файла результатов измерения LabVIEW (Read LabVIEW Measurement File)

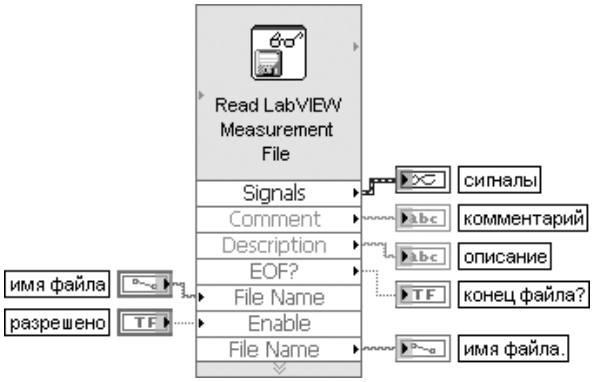


Рис. 2.42. Блок-диаграмма возможного подключения Экспресс-ВП

Экспресс-ВП считывает данные из файла результатов измерения LabVIEW. Диалоговое окно данного Экспресс-ВП имеет следующие опции (рис. 2.43).
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 с расширением .lvm.
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 – время канала** (First column is time channel) – определяет, что данные времени для каждого канала находятся в первом столбце файла данных.

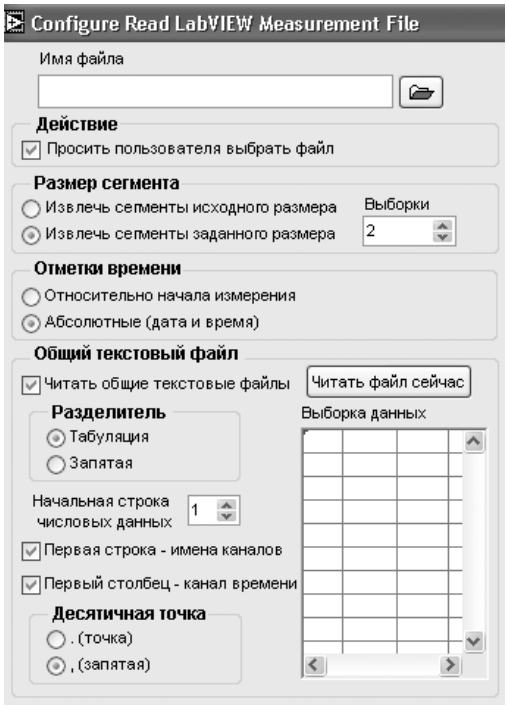


Рис. 2.43. Вид диалогового окна конфигурирования Экспресс-ВП ***Считать из файла результатов измерения LabVIEW*** *(Read LabVIEW Measurement File)*

• **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. 2.42):

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.

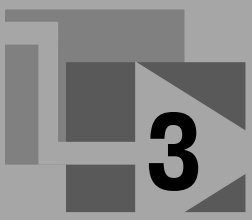
Комментарий (Comment) – возвращает комментарии, добавленные к каждому набору данных в файле с расширением .lvm.

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)

Функции генерации, ввода и обработки данных LabVIEW



3.1. Функции генерации и обработки сигналов

3.1.1. Функции генерации сигналов и шумов

ВП из палитры генерации сигналов и шумов (рис. 3.1) 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, служат для расчета шумов с различными законами амплитудного и спектрального распределения.

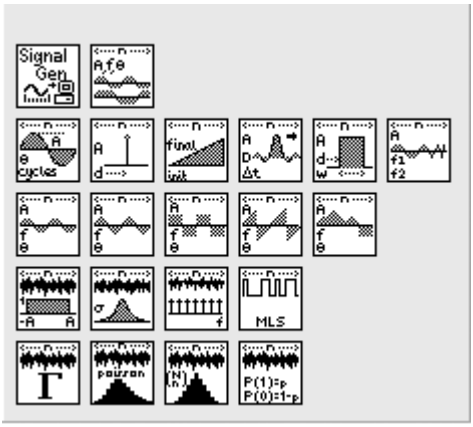
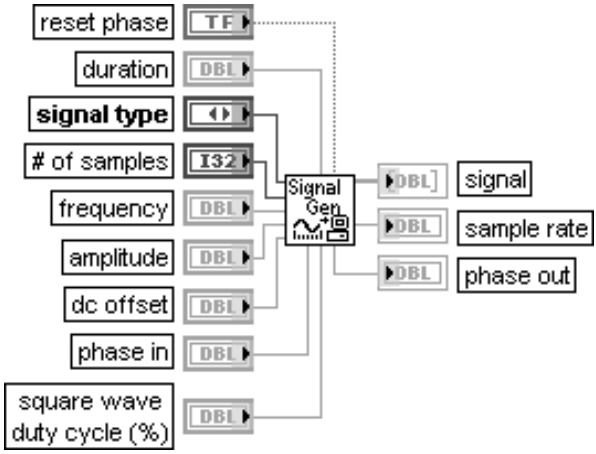
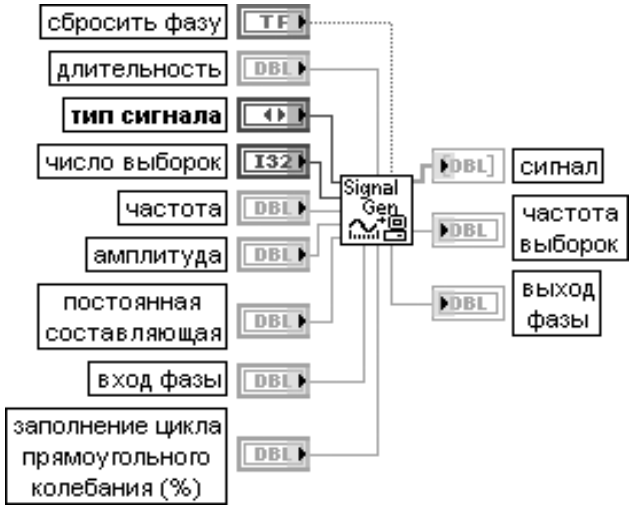


Рис. 3.1. Палитра функций генерации сигналов и шумов

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. По умолчанию на входе установлено состояние ИСТИНА. При этом начальная фаза сигнала устанавливается в соответствии со значением на входе **вход фазы** (phase in). Если на входе **сбросить фазу** установлено состояние ЛОЖЬ, то начальная фаза устанавливается равной значению фазы на **выходе фазы** (phase out) при последнем выполнении этого ВП.

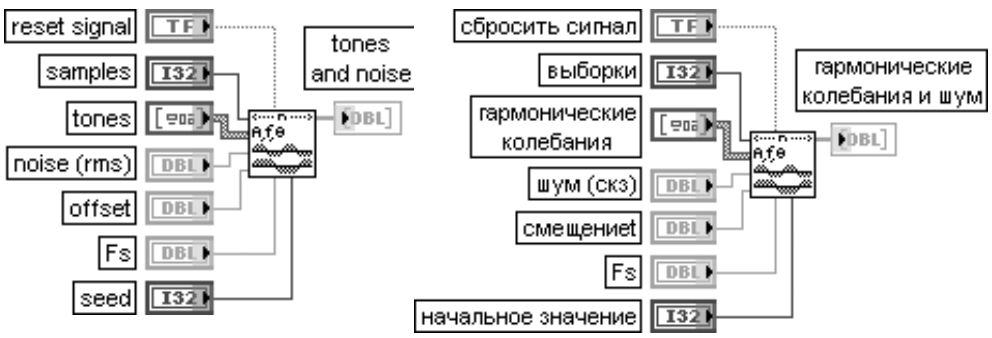
Вход **длительность** (duration) задает время в секундах, равное длительности генерируемого выходного сигнала. По умолчанию значение длительности равно 1,0.

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 % 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

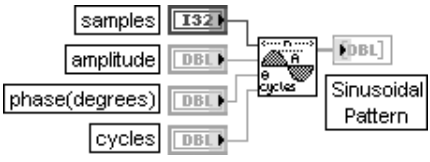
массива **гармонические колебания** (tones), значение начального числа при генерации шума равным значению входа **начало** (seed) и отметку времени равной нулю. По умолчанию состояние входа – ЛОЖЬ.
Вход **выборки** (samples) определяет число выборок выходного массива **гармонические колебания и шумы** (tones and noise). По умолчанию это число равно 1000.
Вход **гармонические колебания** (tones), являющийся кластером, содержит параметры каждого гармонического колебания (таблица):

DBL	**частота** (frequency) определяет частоту синусоидального колебания в герцах
DBL	**амплитуда** (amplitude) определяет амплитуду синусоидального колебания
DBL	**фаза** (phase) определяет начальную фазу синусоидального колебания. По умолчанию фаза равна 0

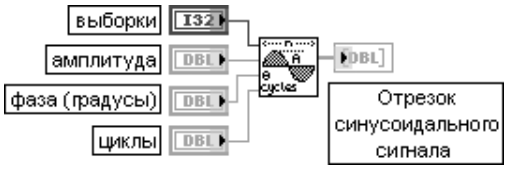
Вход **шум** (noise) определяет среднеквадратичное значение (с.к.з.) аддитивного гауссовского шума. По умолчанию значение **шума** равно 0,0.
Вход **смещение** (offset) задает уровень постоянной составляющей сигнала. По умолчанию уровень равен 0,0.
Вход **Fs** определяет частоту выборок в единицах число 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 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

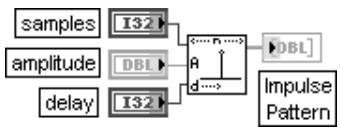
с плавающей запятой, то оно может быть нецелым. Более того, оно может быть и отрицательным, что является математически корректным и полезным для понимания отрицательных частот в преобразованиях Фурье и в спектральном анализе.

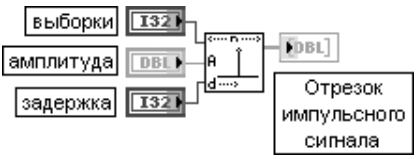
Значения массива **синусоидального сигнала** рассчитываются следующим образом:

$$y_i = a\sin\left(\frac{2\pi i k}{n} + \frac{\pi\varphi_0}{180}\right),$$

где k – **число периодов** (cycles), φ_0 – **начальная фаза в градусах** (phase)

Impulse Pattern **Отрезок импульсного сигнала**

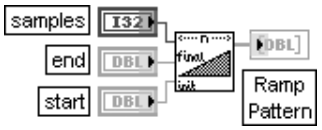


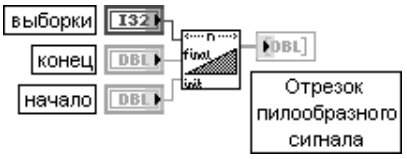


ВП генерирует массив, содержащий отрезок импульсного сигнала.

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 уровень **пилообразного сигнала.** По умолчанию значения этих параметров равны нулю.

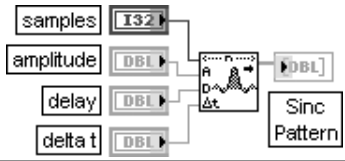
Значения y_i массива **пилообразного сигнала** рассчитываются следующим образом:

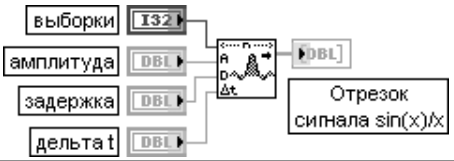
$$y = y_0 + i\,\Delta y,$$

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

$$i = d/\Delta t,$$

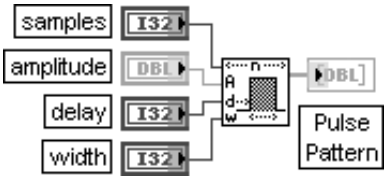
где d – **задержка** (delay), Δt – значение на входе delta t.

Вход Δt (delta t) определяет фактически частоту следования нулей сигнала **sin(x)/x**. По умолчанию значение равно 0,1, что соответствует расстоянию между нулями, равному 10 отсчетам, и ширине центрального лепестка сигнала равной 20 отсчетам.

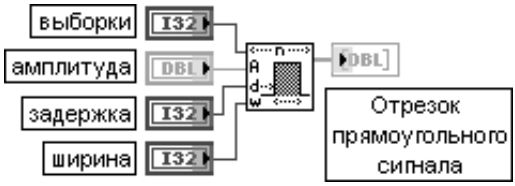
Значения y_i массива сигнала **sin(x)/x** рассчитываются следующим образом:

$$y_i = a\frac{\sin\pi(i\Delta t - d)}{\pi(i\Delta t - d)}$$

Pulse Pattern



Отрезок прямоугольного сигнала



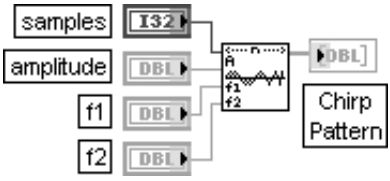
ВП генерирует массив, содержащий отрезок прямоугольного сигнала.

Значения массива **прямоугольного сигнала** рассчитываются следующим образом:

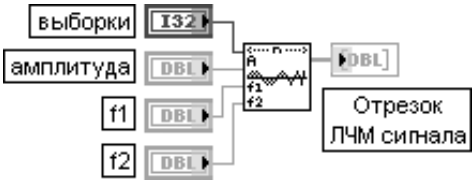
$$y_i = \begin{cases} a \quad если \quad d \le i < (d + w) \\ 0{,}0 \; иначе \end{cases},$$

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).

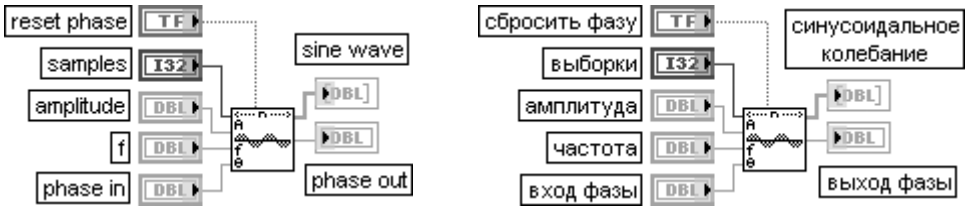
Входы f_1 и f_2 представляют начальную и конечную частоту **ЛЧМ сигнала**, заданные в нормализованных единицах 1/период, где период задан числом отсчетов.

Значения массива **ЛЧМ сигнала** рассчитываются следующим образом:

$$y_i = a\sin((\pi i(f_2 - f_1)/n + 2\pi f_1)\cdot i)$$

Для последующих ВП генерации детерминированных сигналов 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 массива **синусоидального колебания** рассчитываются следующим образом:

$$y_i = a\sin(phase[i]),$$

где $phase[i] = initial\ phase + f \cdot 360 \cdot i$

Triangle Wave **Треугольное колебание**

ВП генерирует массив, содержащий треугольное колебание. Значения массива **треугольного колебания** рассчитываются следующим образом:

$$y_i = a \cdot tri(phase[i]),$$

где

$$tri(phase[i]) = \begin{cases} \dfrac{p}{90} & 0 < p < 90 \\ 2 - \dfrac{p}{90} & 90 < p < 270 \\ \dfrac{p}{90} - 4 & 270 < p < 360 \end{cases},$$

$p = (phase[i]\ по\ модулю\ 360)$, $phase[i] = initial\ phase + f \cdot 360 \cdot i$

Square Wave **Прямоугольное колебание**

ВП генерирует массив, содержащий прямоугольное колебание.
Значения массива **прямоугольного колебания** рассчитываются следующим образом:

$$y_i = a \cdot square(phase[i]),$$

где

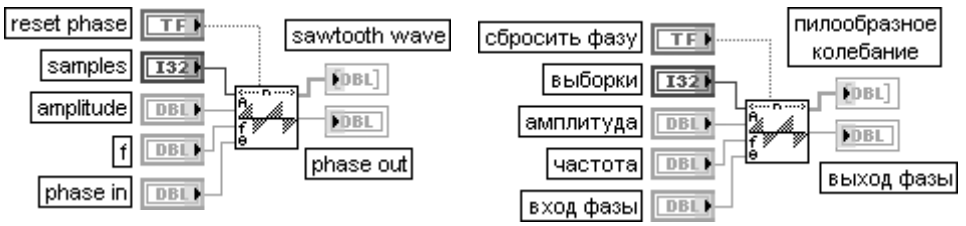
$$square(phase[i]) = \begin{cases} 1 & 0 \le p < (\frac{duty}{100} 360) \\ -1 & (\frac{duty}{100} 360) \le p < 360 \end{cases},$$

$p = (phase[i] по\ модулю 360)$, *duty*– длительность импульса.

Заполнение цикла (duty cycle) определяет время (в % от периода), в течение которого прямоугольный сигнал имеет высокий уровень. Значение по умолчанию для этого входа равно 50%.

$$phase[i] = initial\ phase + f \cdot 360 \cdot i$$

Sawtooth Wave **Пилообразное колебание**



ВП генерирует массив, содержащий пилообразное колебание.
Значения массива **пилообразного колебания** рассчитываются следующим образом:

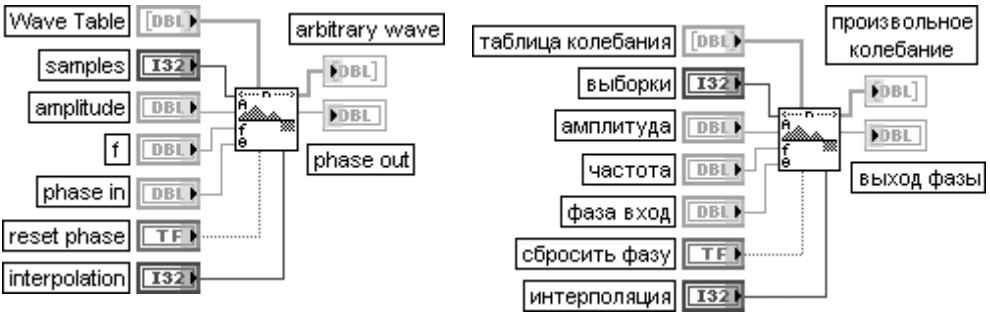
$$y_i = a \cdot sawtooth(phase[i]),$$

где

$$sawtooth(phase[i]) = \begin{cases} \frac{p}{180} & 0 \le p < 180 \\ \frac{p}{180} - 2 & 180 \le p < 360 \end{cases}$$

$p = (phase[i] по\ модулю 360)$, $phase[i] = initial\ phase + f \cdot 360 \cdot i$

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 массива **произвольного колебания** рассчитываются следующим образом:

$$y_i = a \cdot arb(phase[i]),$$

где $arb(phase[i]) = WT((phase[i] по\ модулю\ 360) \cdot m / 360)$,

m – число значений в **таблице колебания** (Wave Table),

$WT(x) = Wave\ Table[\mathrm{int}(x)]$ если вход **интерполяция = 0**.

$WT(x)$ = линейно интерполированным значениям $Wave\ Table[\mathrm{int}(x)]$ и

$Wave\ Table[(\mathrm{int}(x)+1) по\ модулю\ m]$ если вход **интерполяция = 1**.

$phase[i] = initial\ phase + f \cdot 360 \cdot i$

В последующих таблицах рассматриваются ВП генерации случайных сигналов с различными законами амплитудного и спектрального распределения.

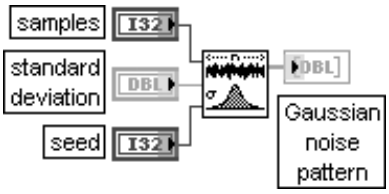
Uniform White Noise **Равномерный белый шум**

ВП генерирует псевдослучайный белый шум с равномерным законом амплитудного распределения, значения которого находятся в диапазоне ***[-a:a],*** где ***a*** представляет абсолютное значение **амплитуды**.

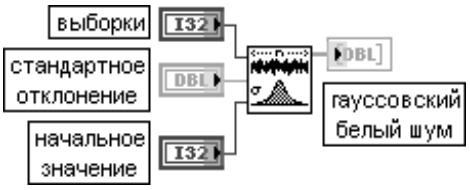
По умолчанию значение амплитуды равно 0,0.

Псевдослучайная последовательность повторяется приблизительно через 2^{90} выборок

Gaussian White Noise



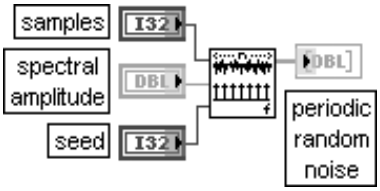
Гауссовский белый шум



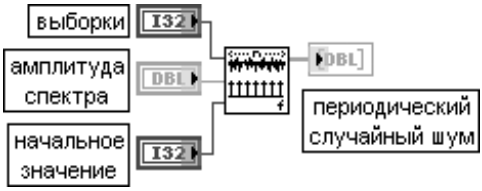
ВП генерирует псевдослучайную последовательность с гауссовским (нормальным) распределением с параметрами (**μ,sigma**) = (0,**s**), где **s** является абсолютным значением заданного **стандартного** (среднеквадратичного) **отклонения** (standard deviation). По умолчанию значение **стандартного отклонения** 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

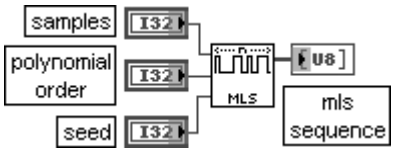


ВП генерирует массив, содержащий **периодический случайный шум** (PRN).
Вход **амплитуда спектра** (spectral amplitude) задает амплитуду частотных составляющих **периодического случайного шума**.
Выходной массив содержит все частоты, которые могут быть представлены целым числом периодов на установленном 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:

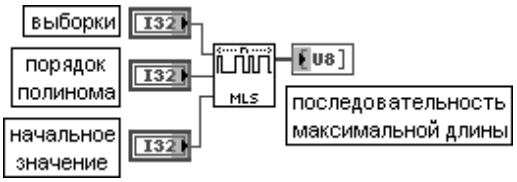
$$амплитуда\ спектра \cdot \left(\frac{выборки}{2}-1\right),\ если\ \textbf{выборки}\ являются\ четным\ числом;$$

$$амплитуда\ спектра \cdot \frac{выборки-1}{2},\ если\ \textbf{выборки}\ являются\ нечетным\ числом$$

Binary MLS

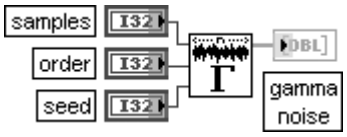


Двоичная последовательность максимальной длины

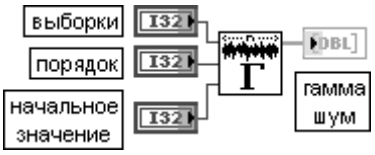


ВП генерирует двоичную **последовательность максимальной длины** (maximum length sequence – MLS), используя деление по модулю два простого полинома, имеющего порядок, заданный на входе **порядок полинома** (polynomial order). Значение по умолчанию на входе **порядок полинома** равно 31

Gamma Noise

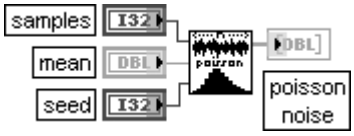


Гамма-шум

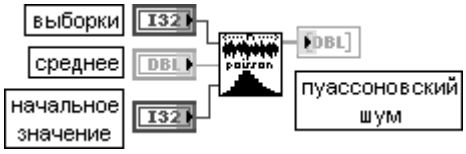


ВП генерирует псевдослучайный набор значений, которые представляют интервалы времени ожидания заданного числа событий пуассоновского процесса с единичным средним.
Вход **порядок** (order) определяет число событий. По умолчанию **порядок** равен 1

Poisson Noise

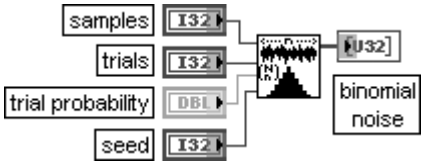


Пуассоновский шум

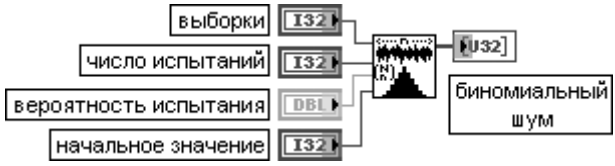


ВП генерирует псевдослучайную последовательность значений, которые представляют число событий ординарного Пуассоновского процесса, появляющихся на заданном интервале, определенном величиной на входе **среднее** (mean). По умолчанию значение **среднего** равно 1,0

Binomial Noise



Биномиальный шум

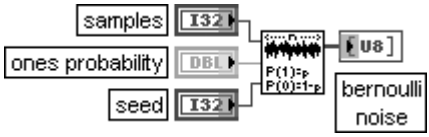


ВП генерирует псевдослучайную последовательность с биномиальным амплитудным распределением, значения которой представляют число реализаций событий, заданных вероятностью совершения событий и числом испытаний.

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.

Выход **шум Бернулли** содержит псевдослучайную последовательность с распределением Бернулли

3.1.2. Функции обработки сигналов во временной области

Функции обработки сигналов во временной области (рис. 3.2) включают функции свертки и корреляции сигналов, определения уровня постоянного и переменного напряжения в сигнале, дифференцирования и интегрирования, определения параметров пиков и импульсов.

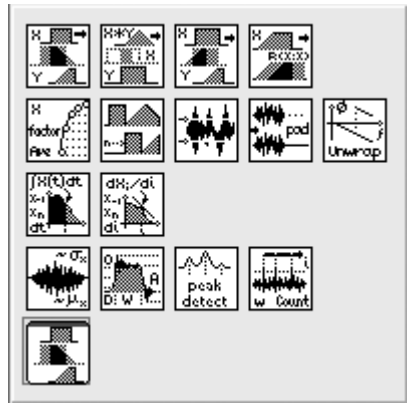
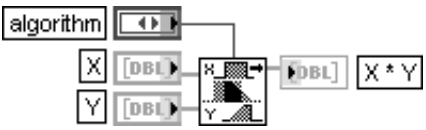


Рис. 3.2. Вид палитры функций обработки сигналов во временной области

Convolution **Свертка**



ВП выполняет свертку двух последовательностей ***X*** и ***Y***. Для непрерывных сигналов ***x(t)*** ***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 **прямая** (direct) ВП использует прямую форму линейной свертки.
При этом для дискретных последовательностей ***X*** и ***Y*** длиной ***n*** и ***m*** соответственно 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.
ВП **Свертка** (Convolution) размещен в палитрах **Временная область** (Time Domain) 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 входных последовательностей **X** * **Y** и **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** должна быть больше или равна длине последовательности **Y**.
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**

ВП вычисляет автокорреляционную функцию входной последовательности **X**.

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 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 $k \geq n$. 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. 3.3. Результат выполнения программы приведен на рис. 3.4.

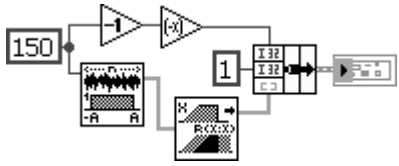


Рис. 3.3. Блок-диаграмма программы расчета автокорреляционной функции с центром в нулевой точке

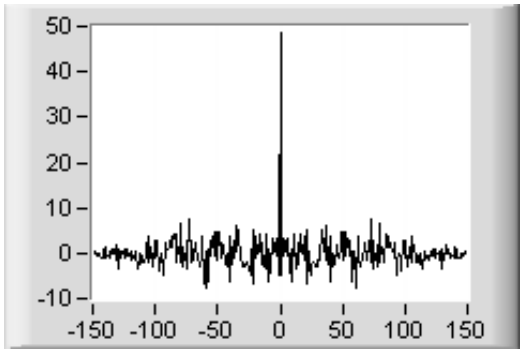
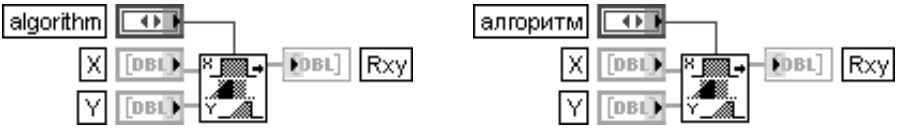


Рис. 3.4. График автокорреляционной функции с центром в нулевой точке

CrossCorrelation **Взаимная корреляционная**
функция



ВП вычисляет взаимную корреляционную функцию входных последовательностей **X** и **Y**.

Взаимная корреляционная функция $R_{xy}(\tau)$ сигналов $x(t)$ и $y(t)$ определяется выражением

$$R_{xy}(\tau) = \int_{-\infty}^{\infty} x(t)y(t+\tau)dt.$$

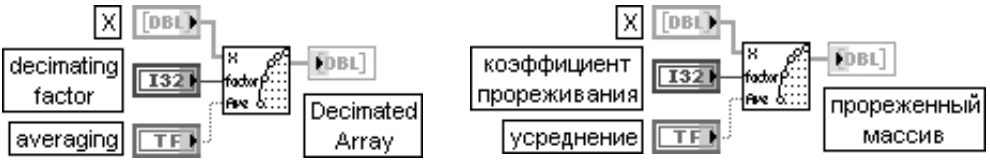
При дискретной реализации элементы выходной последовательности h_j вычисляются следующим образом:

$$h_j = \sum_{k=0}^{n-1} x_k y_{j+k},$$

где n и m – длина 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 $k \geq m$

Decimate **Прореживание**



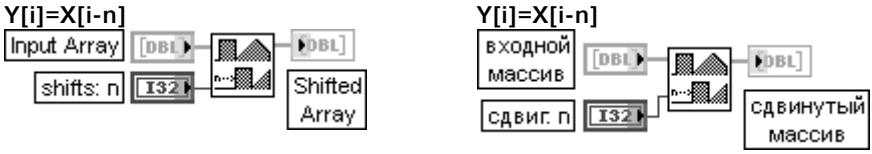
ВП формирует на выходе **прореженный массив** (Decimated Array) из входной последовательности **X** в зависимости от **коэффициента прореживания** (decimating factor) и состояния входа **усреднение** (averaging). Элементы выходного массива y_i рассчитываются следующим образом:

$$y_i = \begin{cases} x_{im} & \text{если усреднение} = \text{ЛОЖЬ} \\ \dfrac{1}{m}\displaystyle\sum_{k=0}^{m-1} x_{im+k} & \text{если усреднение} = \text{ИСТИНА} \end{cases},$$

где $i = \overline{0, size-1}$, $size = trunc(\dfrac{n}{m})$,

n – число элементов входной последовательности **X**, *m* – коэффициент прореживания, ***size*** **–** число элементов выходного массива. По умолчанию **коэффициент прорежива-**

ния равен 2, а состояние входа **усреднение –** ЛОЖЬ



ВП формирует на выходе последовательность в соответствии с выражением

$$y_i = \begin{cases} x_{i-shifts} & если\ \ 0 \le i - shifts < n \\ 0 & иначе \end{cases},$$

где $i = \overline{0, n-1}$; 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. 3.5.

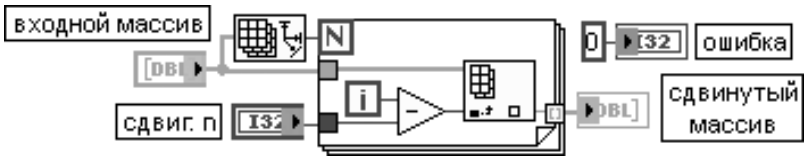
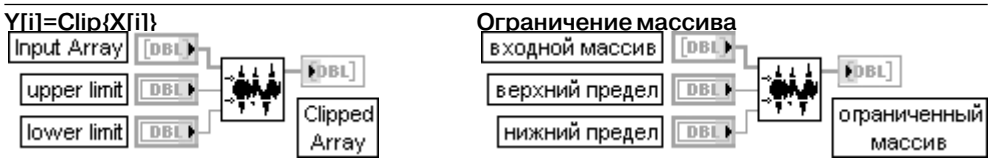


*Рис. 3.5. Блок-диаграмма ВП **Y[i] = X[i-n]***



ВП ограничивает элементы **входного массива** (Input Array) границами, заданным и **верхним пределом** (upper limit) и **нижним пределом** (lower limit). Таким образом, элементы выходного ограниченного массива будут связаны с элементами входного массива соотношением:

$$y_i = \begin{cases} a & x_i > a \\ x_i & b \le x_i \le a \\ b & x_i < b \end{cases},$$

где $i = \overline{0, n-1}$, a и b – значения, подаваемые на входы **верхний предел** и **нижний предел**.

По умолчанию верхний предел равен 1,0, а нижний – 0,0. Ниже на рис. 3.6 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 соответствую-

щего ВП приведена на рис. 3.7.

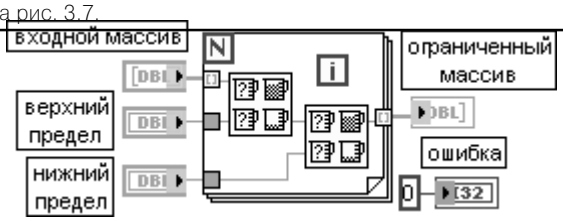


Рис. 3.6. Блок-диаграмма ВП ***Ограничение массива***

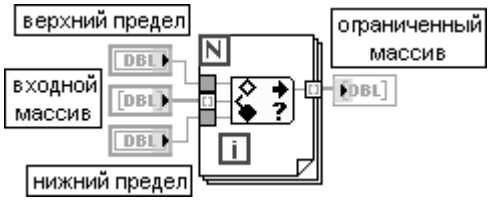


Рис. 3.7. Блок-диаграмма ВП на основе функции ***Нахождение в диапазоне и ограничение*** *(In Range and Coerce)*

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. 3.8.

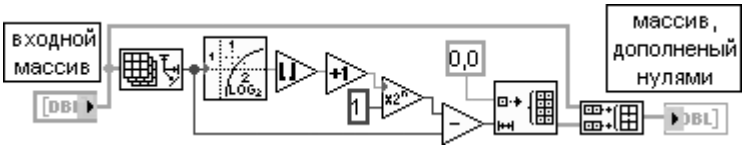


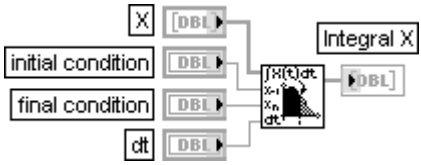
Рис. 3.8. Блок-диаграмма ВП ***Дополнение нулями*** *(Zero Padder)*

Unwrap Phase

Развертка фазы

ВП производит развертку массива значений фазы путем удаления разрывов, абсолютные значения которых превышают **2π**.

Integral x(t) **Интеграл x(t)**

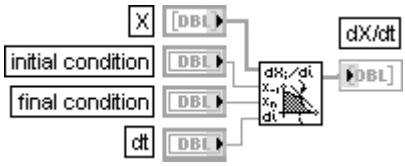


ВП выполняет численное интегрирование входной последовательности **X** в соответствии с формулой

$$y_i = \frac{1}{6}\sum_{j=0}^{i}(x_{j-1} + 4x_j + x_{j+1})dt,$$

где $i = \overline{0, n-1}$, n – длина входной последовательности.
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

Derivative x(t) **Производная x(t)**

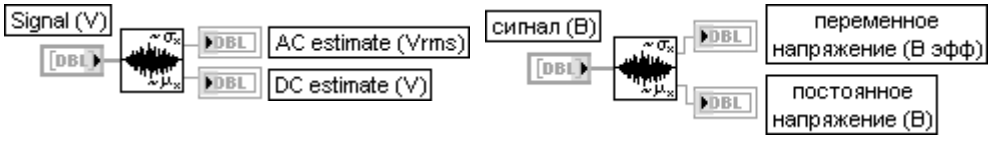


ВП выполняет численное дифференцирование входной последовательности **X** по формуле

$$y_i = \frac{1}{2dt}(x_{i+1} - x_{i-1}),$$

где $i = \overline{0, n-1}$, n – длина входной последовательности.
Значения, подаваемые на входы **начальное** (initial condition) и **конечное условия** (final condition), определяют величину x_{-1} и x_n соответственно. По умолчанию оба эти значения равны нулю.
На вход dt подается величина, определяющая шаг дифференцирования. По умолчанию dt =1,0

AC & DC Estimator **Оценка переменного и постоянного напряжения сигнала**



ВП рассчитывает оценки переменной и постоянной составляющих напряжения входного сигнала.

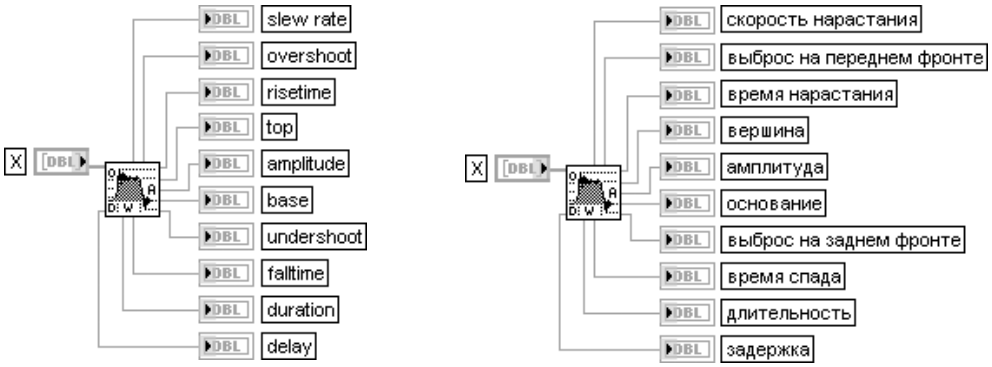
В общем случае для получения оценок переменной и постоянной составляющих должен производиться спектральный анализ сигнала и выделение постоянной составляющей, расположенной на нулевой частоте, и переменной составляющей, определяемой путем среднеквадратичного суммирования значений на всех остальных частотах.

Для устойчивой оценки значений входной сигнал должен содержать как минимум три цикла переменного напряжения.

В данном ВП (рис. 3.9) 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. 3.9. Блок-диаграмма ВП ***Оценка переменного и постоянного напряжения сигнала*** *(AC & DC Estimator)*

Pulse Parameters **Параметры импульса**



ВП анализирует входную последовательность **X**, задающую форму импульса, и определяет наилучший набор параметров данного импульса.

В состав определяемых параметров входят следующие:

скорость нарастания (slew rate) **–** отношение разности (90% амплитуды – 10% амплитуды) к времени нарастания;

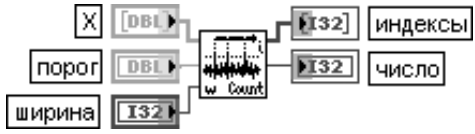
выброс на переднем фронте (overshoot) **–** разность между максимальным значением импульса и вершиной;

время нарастания (risetime) **–** время, за которое сигнал на переднем фронте нарастает от 10% до 90%;

вершина (top) – линия, наилучшим образом характеризующая величину импульса, когда он активен (включен);
амплитуда (amplitude) – разность между вершиной и основанием;
основание (base) – линия, наилучшим образом характеризующая импульс, когда он неактивен (выключен);
выброс на заднем фронте (undershoot) – разность между минимальным значением импульса и основанием;
время спада (falltime) **–** время, за которое сигнал на заднем фронте уменьшается от 90% до 10%;
длительность (duration) – разность между моментами достижения сигналом 50% уровня на переднем и заднем фронтах;
задержка (delay) – разность между моментами начала импульса и достижения 50% уровня на переднем фронте.
Входная последовательность **X** должна отвечать следующим условиям:
1) количество отсчетов **X** должно быть не менее трех;
2) импульс должен иметь явно выраженные передний фронт, вершину и задний фронт;
3) ожидаемое пиковое значение шума должно быть меньше 50% ожидаемой амплитуды импульса.
Если количество отсчетов **X** меньше трех, то ВП присваивает всем параметрам неопределенное значение (NaN) и возвращает ошибку.
Если **X** не содержит явно выраженные фрагменты (п. 2), то ВП анализирует данные и присваивает значения тем параметрам, которые он может идентифицировать, остальным параметрам присваивается значение NaN. При этом сообщение об ошибке не выдается.
Полярность импульса и знак амплитуды определяются по соотношению уровней вершины и основания, значения которых, в свою очередь, определяются путем анализа гистограммы амплитудного распределения отсчетов импульса. Поэтому если уровень шума превышает 50% амплитуды импульса (п. 3), ВП может присвоить некорректные значения параметрам импульса без сообщения об ошибке. Для уменьшения уровня шума может быть рекомендовано применение медианного фильтра

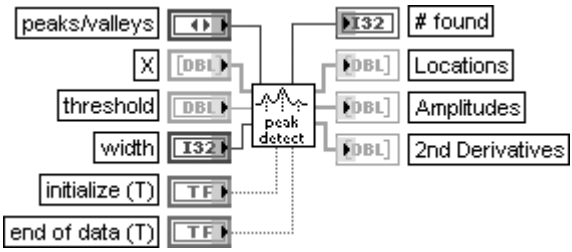
Threshold Peak Detector

Пороговый детектор пиков

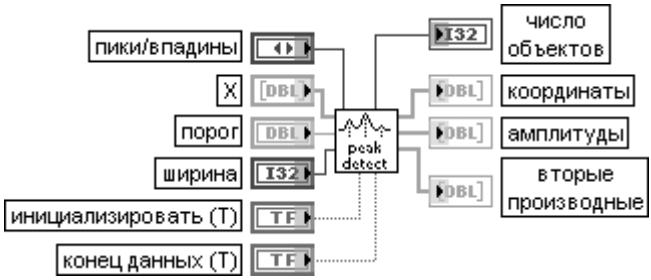


ВП анализирует входную последовательность **X** с целью определения **количества пиков** (count) и их **индексов** (Indices).
В качестве пика принимается участок последовательности, ширина которого, выраженная количеством элементов, не меньше значения, заданного на входе **ширина** (width), а уровень – 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 амплитуды 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 **координата** (Locations) и **амплитуда** (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)

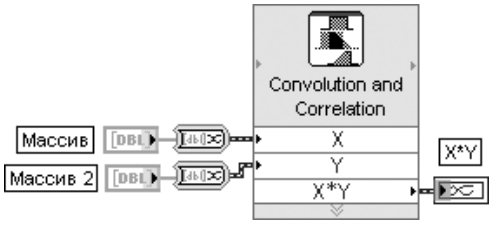


Рис. 3.10. Блок-диаграмма возможного подключения Экспресс-ВП

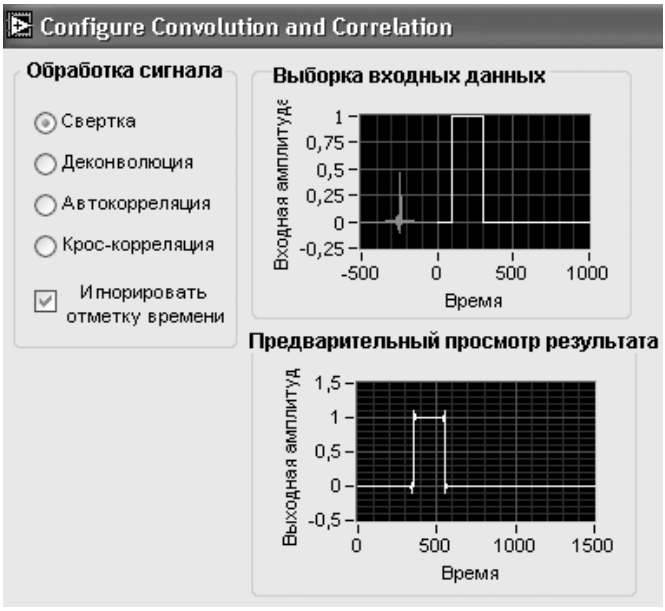


Рис. 3.11. Вид диалогового окна конфигурирования Экспресс-ВП ***Свертка и корреляция*** *(Convolution and Correlation)*

3.1.3. Функции обработки сигналов в частотной области

Палитра функций обработки сигналов в частотной области (рис. 3.12) содержит наборы функций, позволяющих выполнить прямое и обратное преобразования Фурье, Гильберта, Хартли и Уолша-Адамара, а также прямое и обратное вейвлет-преобразования. На базе функций преобразования Фурье разработан ряд высокоуровневых приборов для оценки взаимного спектра мощности, импульсной и частотной передаточной характеристик цепей, частотной функции когерентности и измеритель гармонических искажений сигнала, размещенные на данной палитре.

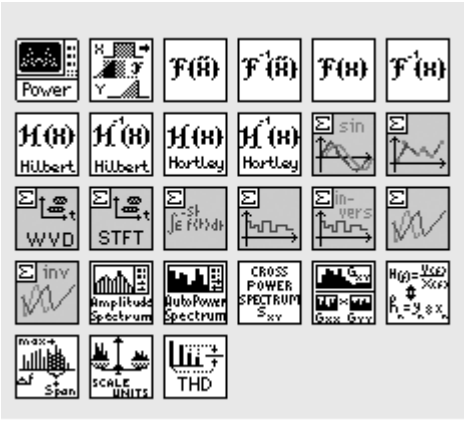


Рис. 3.12. Вид палитры функций обработки сигналов в частотной области

Методы анализа сигналов в частотной области являются широко распространенными, поскольку позволяют эффективно использовать свойства сигналов на основе хорошо разработанного математического аппарата преобразований Фурье [7–10].

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

$$x(t) \Leftrightarrow X(f).$$

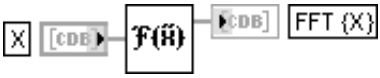
Для дискретных (по времени) сигналов вместо интегрального применяют **дискретное** преобразование Фурье (**ДПФ** или **DFT**) [10].

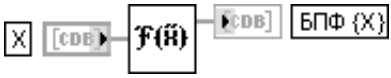
$$X_k = \sum_{i=0}^{n-1} x_i e^{-j2\pi ik/n}, \text{ для } k = \overline{0, n-1}; \tag{3.1}$$

$$x_i = \frac{1}{n}\sum_{i=0}^{n-1} X_k e^{j2\pi ik/n}, \text{ для } i = \overline{0, n-1}. \tag{3.2}$$

Вычисление ДПФ по формулам (3.1) и (3.2) 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.

Complex FFT **Комплексное преобразование Фурье**





ВП производит преобразование Фурье входной комплексной последовательности **X**. Выходная последовательность **FFT {X}** также является комплексной. Данный ВП сначала анализирует входные данные и, основываясь на этом анализе, выполняет преобразование Фурье данных.

Если длина входной последовательности равна $n = 2^m$, где m = 1, 2, 3, ...23, ВП выполняет расчет БПФ по алгоритму оптимизированного метода расщепления. Этот алгоритм использует последовательное разделение массива данных длиной ***n*** на ряд подмассивов и их обработку в соответствии с алгоритмом дискретного преобразования Фурье с минимальным числом арифметических операций. Наибольший объем данных, который может быть обработан с помощью ВП **Комплексного преобразования Фурье**, равен 2^{23} = 8,388,608 (8M).

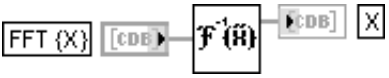
В случае, когда длина входной последовательности $n \neq 2^m$, выполняется ДПФ. При этом наибольший объем данных равен $2^{22} - 1$ = 4,194,303 (4M – 1).

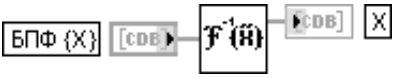
Элементы комплексного спектра Y_{n-i} в соответствии со свойствами ДПФ могут быть представлены как элементы, находящиеся на отрицательных частотах Y_{-i}. Размещение составляющих комплексного спектра на относительных частотах в этом случае приведено в таблице. При четном n максимальное значение относительной частоты (частоты Найквиста) равно $k = n/2$, при нечетном $n - k = (n-1)/2$. Переход к реальной частоте осуществляется путем умножения i на Δf, где $\Delta f = 1/T_d$, T_d – интервал дискретизации.

Элемент комплексного спектра	Относительная частота
Y_0	Постоянная составляющая
Y_1	1 гармоника (основная частота)
Y_2	2 гармоника
Y_3	3 гармоника

Элемент комплексного спектра	Относительная частота
…	…
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

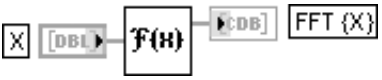
Inverse Complex FFT **Обратное комплексное БПФ**





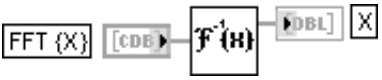
ВП вычисляет обратное преобразование Фурье комплексной входной последовательности **FFT {X}.** Так же как и при выполнении ВП **Комплексное преобразование Фурье** (Complex FFT), анализируется объем входной выборки и выбирается алгоритм обратного БПФ или обратного ДПФ в зависимости от того, равна или не равна длина последовательности степени числа 2

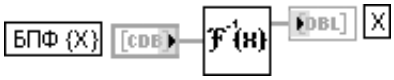
Real FFT **Действительное преобразование Фурье**



ВП вычисляет действительное преобразование Фурье входной последовательности **X.** Так же как и при выполнении ВП **Комплексное преобразование Фурье,** анализируется объем входной последовательности и выбирается алгоритм БПФ или ДПФ в зависимости от того, равна или не равна длина последовательности степени числа 2. Выходная последовательность – массив комплексных чисел

Inverse Real FFT **Обратное действительное преобразование Фурье**





ВП вычисляет обратное действительное преобразование Фурье комплексной входной последовательности **FFT {X}.** Так же как и при выполнении ВП **Комплексное преобразование Фурье,** анализируется объем входной выборки и выбирается алгоритм обратного БПФ или обратного ДПФ в зависимости от того, равна или не равна длина последовательности степени числа 2. Выходная последовательность – массив действительных чисел

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) гармоники
$S_{xx(k-1)} = S_{xx(n-(k-1))}$	Мощность (k – 1) гармоники
$S_{xxk} = S_{xx(n-k)}$	Мощность k гармоники на частоте Найквиста (n нечетно)
S_{xxk}	Мощность k гармоники на частоте Найквиста (n четно)

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.

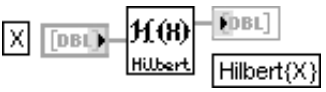
Размер N комплексной выходной последовательности S_{xy} равен

$$N = \begin{cases} \max(n,m), & \text{если } \max(n,m) = 2^k; \\ 2^i & \text{если } \max(n,m) \neq 2^k \end{cases}$$

для $k = \overline{1,23}$, где n, m – длина входной последовательности X и Y соответственно,

$i = trunc(\log_2(\max(n,m))) + 1, 1 \le i \le 23.$

Fast Hilbert Transform



Быстрое преобразование Гильберта

ВП вычисляет быстрое преобразование Гильберта входной последовательности .
Интегральное преобразование Гильберта непрерывной функции $x(t)$ в 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:

$$h(t) \Leftrightarrow H(f) = -j\,\mathrm{sgn}(f)X(f),$$

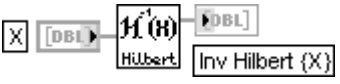
где $X(f)$ – преобразование Фурье $x(t)$, а $\mathrm{sgn}(f) = \begin{cases} 1 & f > 0 \\ 0 & f = 0 \\ -1 & f < 0 \end{cases}$.

ВП выполняет дискретную реализацию преобразования Гильберта с помощью процедур БПФ, основываясь на соответствующей взаимосвязи $h(t) \Leftrightarrow H(f)$. При этом реализуются следующие шаги:

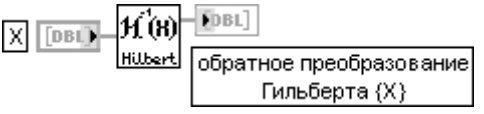
- выполняется преобразование Фурье входной последовательности X, $Y = F\{X\}$;
- постоянная составляющая приравнивается к нулю, $Y_0 = 0$;
- если длина входной последовательности четна, то приравнивается к нулю компонента на частоте Найквиста, $Y_{Nyq} = 0$;
- положительные гармоники в преобразовании Фурье умножаются на $-j$, а отрицательные – на j. Формируется новая последовательность $H_k = -j\,\mathrm{sgn}(k)Y_k$;
- 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:

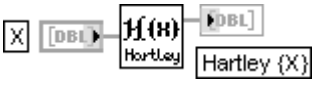
$$x(t) = H^{-1}\{h(t)\} = -H\{h(t)\}.$$

ВП выполняет дискретную реализацию обратного преобразования Гильберта с помощью прямого преобразования Гильберта, реализуя следующие шаги:

1) преобразование Гильберта входной последовательности ***X***: $Y = H\{X\}$;

2) получение 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 $cas(x) = \cos(x) + \sin(x)$.

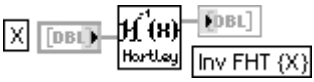
Для последовательности X_i дискретное преобразование Хартли $Y = Hartley\{X\}$ будет иметь вид

$$Y_k = \sum_{i=0}^{n-1} X_i\, cas\left(\frac{2\pi i k}{n}\right),$$

где $k = \overline{1, n-1}$, – число элементов последовательности X_i.

Преобразование Хартли отображает действительную последовательность значений во временной области в такую же последовательность в частотной области. В отличие 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 $cas(x) = \cos(x) + \sin(x)$.

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:

$$Y_k = \frac{1}{n}\sum_{i=0}^{n-1} X_i \, cas(\frac{2\pi i k}{n}),$$

где $k = \overline{1, n-1}$, – число элементов последовательности X_i.
Обратное преобразование Хартли отображает действительную последовательность значений в частотной области в такую же последовательность во временной области

Buneman Frequency Estimator **Оценка частоты колебания**

ВП производит оценку относительной частоты β гармонического колебания, представленного последовательностью X, с помощью анализа амплитудного спектра этой последовательности:

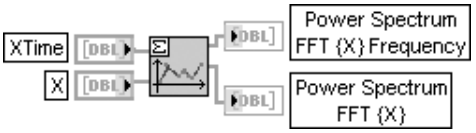
$$\beta = b + \frac{n}{\pi} a\tan\left(\frac{\sin\frac{\pi}{n}}{\cos\frac{\pi}{n} + \frac{|F_b(X)|}{|F_{b+1}(X)|}}\right),$$

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

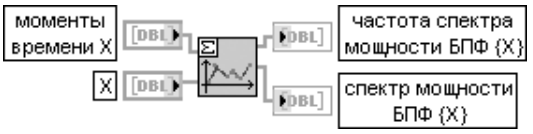
$$f = \beta df = \beta f_s / n,$$

где f_s – частота дискретизации

Unevenly Sampled Signal Spectrum



Спектр сигнала с неравномерными отсчетами



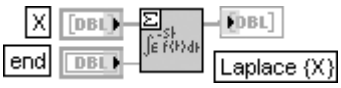
ВП осуществляет расчет спектра мощности последовательности **X** с неравномерным по времени распределением выборок, задаваемым последовательностью **моменты времени X** (XTime). Выходы **частота спектра мощности БПФ {X}** (Power Spectrum FFT {X} Frequency) и **спектр мощности БПФ {X}** (Power Spectrum FFT {X}) представляют, соответственно, массивы частот и значений спектра мощности, рассчитанные 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 $\tau = \frac{1}{2\sigma^2}\arctan\left(\frac{\sum_{k=0}^{n-1}\sin 2\omega t_k}{\sum_{k=0}^{n-1}\cos 2\omega t_k}\right)$, $\sigma^2 = \frac{1}{n-1}\sum_{k=0}^{n-1}(x_k - \bar{x})^2$, $\bar{x} = \frac{1}{n}\sum_{k=0}^{n-1}x_k$

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 $t \geq 0$.

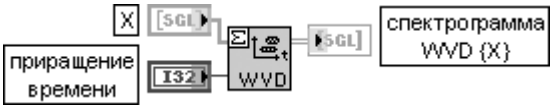
Дискретное преобразование Лапласа для равномерно расположенных отсчетов является аналогом непрерывного преобразования. Для повышения эффективности расчетов дискретного преобразования Лапласа используют дробное преобразование Фурье (Fractional FFT, FFFT), определяемое следующим образом:

$$FFFT\{X\}(t) = \int_0^\infty x(s)\exp(-i\alpha st)ds,$$

где α – произвольное комплексное число

WVD Spectrogram

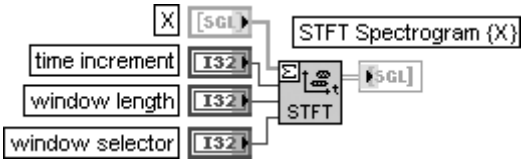
Спектрограмма WVD



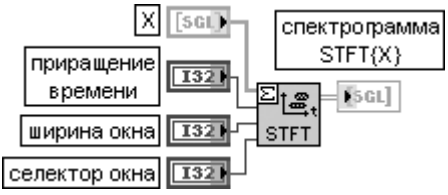
ВП производит расчет распределения энергии в пространственно-временной области, используя алгоритм **Wigner-Ville**.

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 рядами спектрограммы **WVD** будет равно **приращение времени** / f_s секунд

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 с**електор 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:

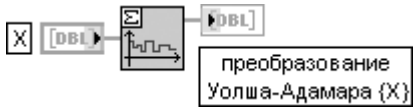
$$число\ строк = \frac{число\ элементов\ X}{приращение\ времени}.$$

Число градаций по частоте (число столбцов) рассчитывается следующим образом:

$$число\ столбцов = \frac{число\ элементов\ X}{2} + 1$$

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.

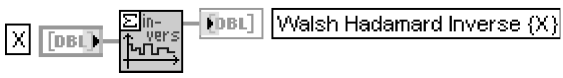
Преобразование Уолша-Адамара базируется на системе ортогональных функций, имеющих только два значения (–1 и +1). Например, для случая $n = 4$ преобразование Уолша-Адамара входной последовательности $X = \{x_0, x_1, x_2, x_3\}$ может быть представлено в матричной форме следующим образом:

$$WH\{X\} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix}.$$

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



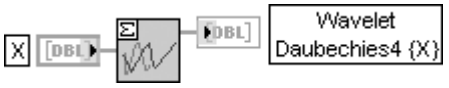
Обратное преобразование Уолша-Адамара

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

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 $Wavelet\ Daubechies4\{X\} = C \cdot X$, 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:

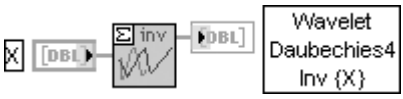
$$c_0^2 + c_1^2 + c_2^2 + c_3^2 = 1 \qquad c_2 c_0 + c_3 c_1 = 0$$

$$c_3 - c_2 + c_1 - c_0 = 0 \qquad 0c_3 - 1c_2 + 2c_1 - 3c_0 = 0,$$

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



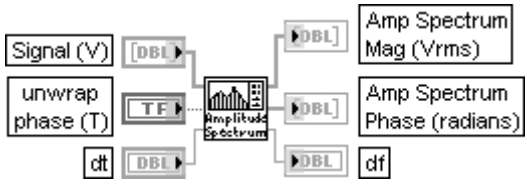
ВП выполняет обратное вейвлет-преобразование входной последовательности ***X***, базирующееся на функции Добеши4. Длина последовательности должна быть кратной степени 2.

Обратное вейвлет-преобразование Добеши4 определяется соотношением $Wavelet\ Daubechies4\ Inv\{X\} = C^{-1} \cdot X$, где матрица преобразования ***C*** равна

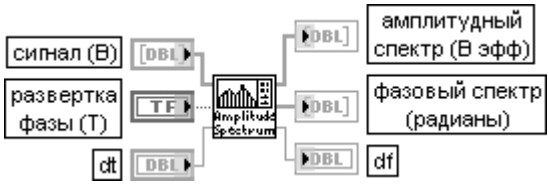
$$C = \begin{bmatrix}
c_0 & c_3 & & & & & & & c_2 & c_1 \\
c_1 & -c_2 & & & & & & & c_3 & -c_0 \\
c_2 & c_1 & c_0 & c_3 & & & & & & \\
c3 & -c0 & c_1 & -c_2 & & & & & & \\
 & & & \cdot & & & & & & \\
 & & & \cdot & & & & & & \\
 & & & & c_2 & c_1 & c_0 & c_3 & & \\
 & & & & c_3 & -c_0 & c_1 & -c_2 & & \\
 & & \cdot & \cdot & \cdot & \cdot & c_2 & c_1 & c_0 & c_3 \\
 & & \cdot & \cdot & \cdot & \cdot & c_3 & -c_0 & c_1 & -c_2
\end{bmatrix}$$

Коэффициенты c_0, c_1, c_2, c_3 имеют те же значения, что и в ВП **Вейвлет-преобразование Добеши4** (Wavelet Daubechies4)

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 производится развертка фазы выходного **фазового спектра** (Amp Spectrum Phase). По умолчанию на входе установлено состояние ИСТИНА. При установке 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:

$$амплитудный\ \ спектр = \frac{|FFT\{сигнал\}|}{N}.$$

Выход **фазовый спектр** отображает односторонний фазовый спектр, выраженный в радианах. Значение **фазового спектра** рассчитывается в соответствии с выражением

$$фазовый\ спектр = фаза(FFT\{сигнал\}).$$

Выход dt представляет частотный интервал [Гц], если интервал dt задан в секундах. Блок-диаграмма ВП **Амплитудный и фазовый спектр** приведена на рис. 3.13.

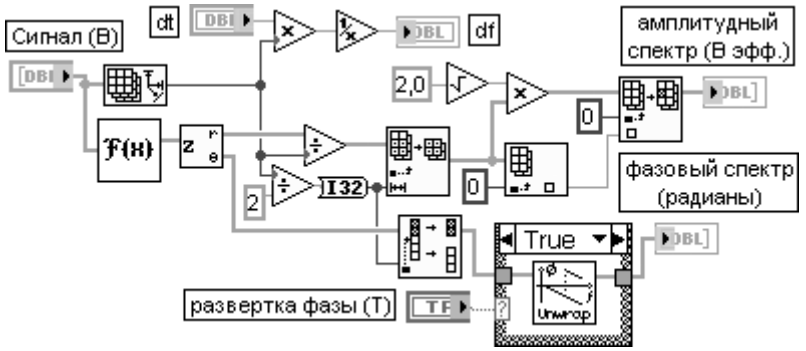
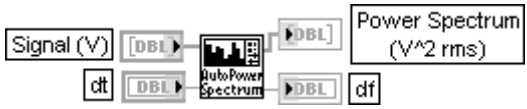
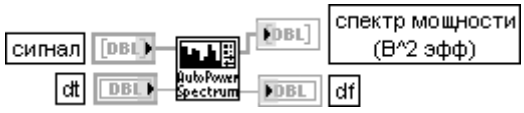


*Рис. 3.13. Блок-диаграмма ВП **Амплитудный и фазовый спектр** (Amplitude and Phase Spectrum)*

Auto Power Spectrum



Спектр мощности



ВП вычисляет 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.
В отличие от ВП **Спектр мощности** (Power Spectrum), входящего в его состав (рис. 3.14), диапазон частот ограничен положительными значениями, а величина спектральных составляющих увеличена в два раза.

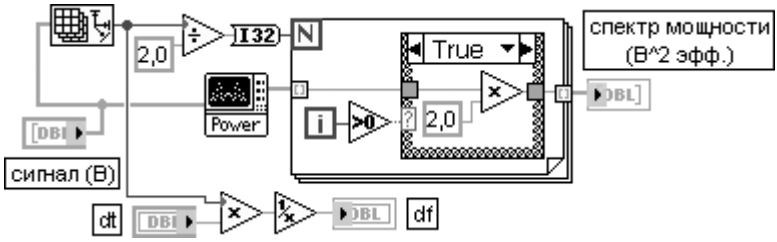
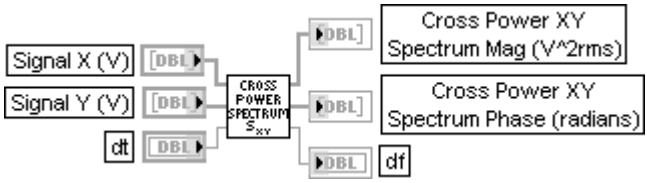
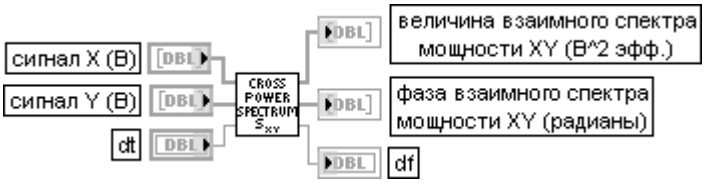


*Рис. 3.14. Блок-диаграмма ВП **Спектр мощности** (Auto Power Spectrum)*

Cross Power Spectrum



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 абсолютную 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 [рад].
Как видно из блок-диаграммы данного ВП (рис. 3.15), его основным элементом является рассмотренный выше ВП **Взаимный спектр мощности** (Cross Power).

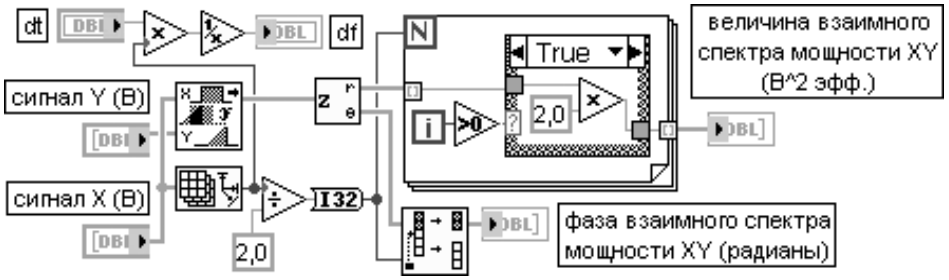
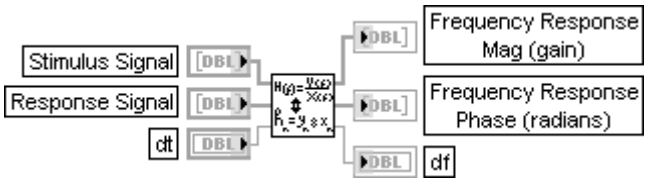
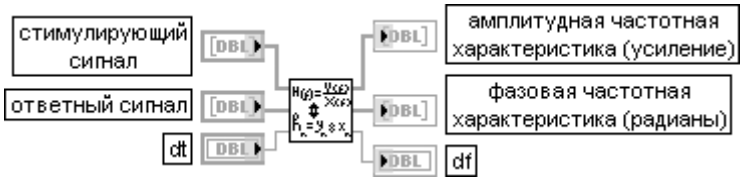


Рис. 3.15. Блок-диаграмма ВП ***Взаимный спектр мощности*** *(Cross Power Spectrum)*

Transfer Function



Передаточная функция



ВП производит расчет односторонней передаточной функции, также известной как частотная передаточная функция, на основе анализа заданных во временной области **стимулирующего сигнала** (Stimulus Signal) и **ответного сигнала** (Response Signal) на входе и выходе тестируемой электрической цепи. Блок-диаграмма ВП приведена на рис. 3.16.

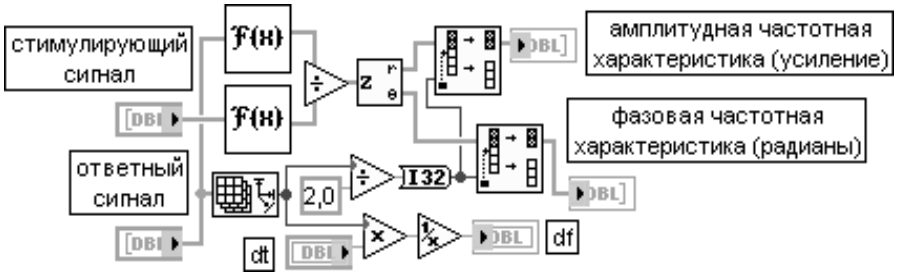
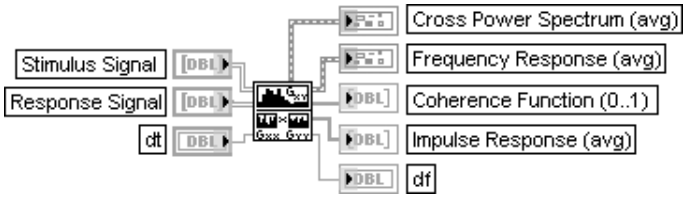
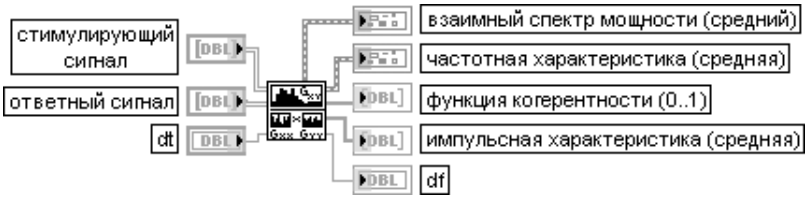


Рис. 3.16. Блок-диаграмма ВП ***Передаточная функция*** *(Transfer Function)*

Network Functions (avg)



Характеристики цепи (средние)



ВП рассчитывает усредненный односторонний **взаимный спектр мощности** (Cross Power Spectrum), усредненную одностороннюю **частотную характеристику** (передаточную функцию) (Frequency Response), одностороннюю **функцию когерентности** (Coherence Function) и **импульсную характеристику** (Impulse Response).

Вход **стимулирующий сигнал** (Stimulus Signal) используется для приема двумерного массива, содержащего сигнал во временной области, который выполняет функцию стимулирующего сигнала цепи. Каждый ряд в двумерном массиве **стимулирующего сигнала** представляет один кадр стимулирующего воздействия, который однозначно связан с одним рядом **ответного сигнала**, представляющего один кадр ответа цепи.

На вход **ответный сигнал** (Response Signal) подается двумерный массив, содержащий сигнал во временной области, который является ответным сигналом цепи.

Выход **взаимный спектр мощности** (Cross Power Spectrum) отображает усредненный односторонний взаимный спектр **стимулирующего** и **ответного сигналов** в виде кластера из массива **модулей** (Magnitude) и **фаз** (Phase). Перечисленные параметры были рассмотрены выше при анализе ВП **Взаимный спектр мощности** (Cross Power Spectrum).

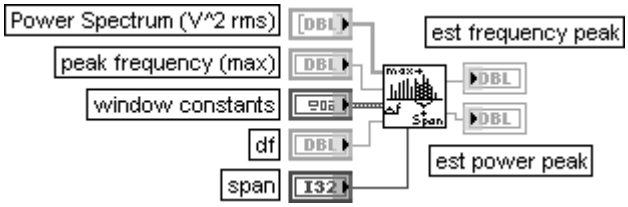
Выход **частотная характеристика** (Frequency Response) отображает усредненную одностороннюю частотную передаточную функцию, полученную в результате деления спектра **ответного сигнала** на спектр **стимулирующего сигнала**. Выход также представлен кластером из массива **модулей** (Magnitude) и **фаз** (Phase). Таким образом, массив модулей представляет амплитудно-частотную характеристику исследуемой цепи, а массив фаз – фазочастотную характеристику.

Выход **функция когерентности** (Coherence Function) представляет одностороннюю функцию когерентности спектра. **Функция когерентности** является безразмерной и изменяется от 0, соответствующего отсутствию когерентности, до 1, соответствующей полной когерентности. **Функция когерентности** показывает частотную компоненту **ответного сигнала**, обусловленную **стимулирующим сигналом,** и измеряет достоверность оценки частотной функции передачи цепи. ВП **Характеристики цепи (средние)** (Network Functions (avg)) рассчитывает функцию когерентности следующим образом:

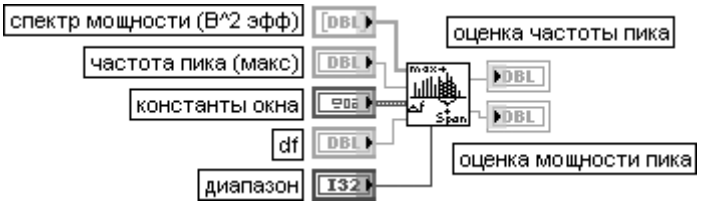
$$функция\ когерентности = \frac{\left|\overline{S_{xy}(f)}\right|^2}{\overline{S_{xx}(f)} \cdot \overline{S_{yy}(f)}},$$

где черта над параметрам означает усреднение по ряду реализаций.
Для правильного вычисления **функции когерентности** необходима подача более одного набора данных на входы **стимулирующего** и **ответного сигналов**. При подаче одного набора данных **функция когерентности** содержит единичное значение на всех частотах.
Выход **импульсная характеристика** (Impulse Response) представляет обратное действительное БПФ усредненной односторонней **передаточной функции**. Этот параметр является безразмерным

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 максимум 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:

$$частота\ пика = \frac{\sum спектр\ мощности(j)(j \cdot df)}{\sum спектр\ мощности(j)},$$

где

$$j = \overline{i - диапазон/2, i + диапазон/2,}$$

i – индекс пика, *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

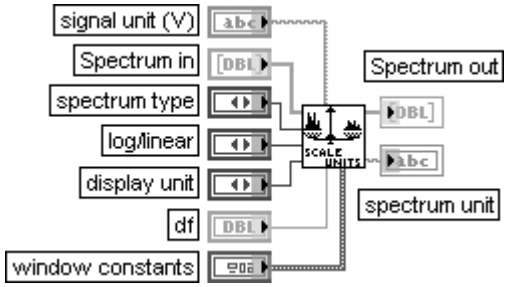
$$мощность\ пика = \frac{\sum спектр\ мощности(j)}{ENBW},$$

где $j = \overline{i - диапазон/2, i + диапазон/2}$,

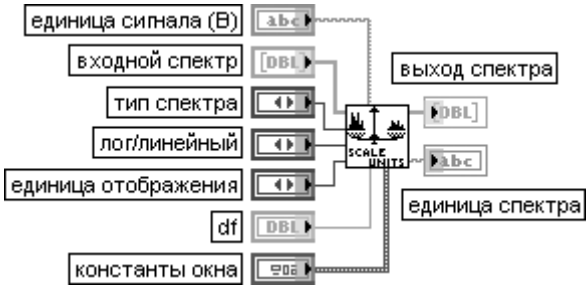
$ENBW$ – эквивалентная шумовая полоса окна (equivalent noise bandwidth of the window). $ENBW$ является одним из параметров **констант окна** (window constants**)**. По умолчанию значение $ENBW$ равно 1,0. Вторым параметром **констант окна** является coherent gain – величина, обратная масштабному коэффициенту окна.

Значения **констант окна** обычно формируются на выходе ВП **Масштабированное временное окно** (Scaled Time Domain Window)

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 следующей таблице:

0 **В эфф**. (вольты, эффективное значение)
Vrms (volts rms)

1 **В пик** (вольты, пиковое значение)
Vpk (volts peak)

2 **В² эфф**. (вольты², эффективное значение)
Vrms^2 volts (squared rms)

3 **В² пик** (вольты², пиковое значение)
Vpk^2 volts (squared peak)

4 **В эфф./$\sqrt{Гц}$** (вольты эфф/$\sqrt{Гц}$)
Vrms/sqrt(Hz) volts (rms per root Hz)

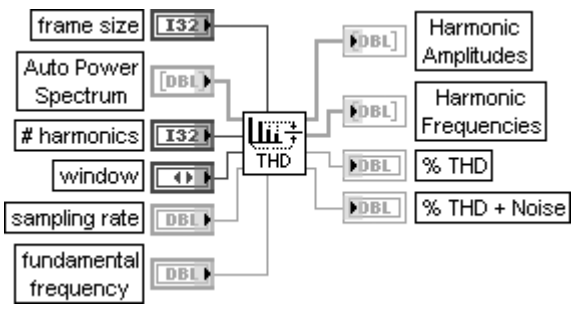
5 **В пик/$\sqrt{Гц}$** (вольты пиковые/$\sqrt{Гц}$)
Vpk/sqrt(Hz) volts (peak per root Hz)

6 **В² эфф/Гц**
Vrms^2/Hz volts (squared rms per Hz)

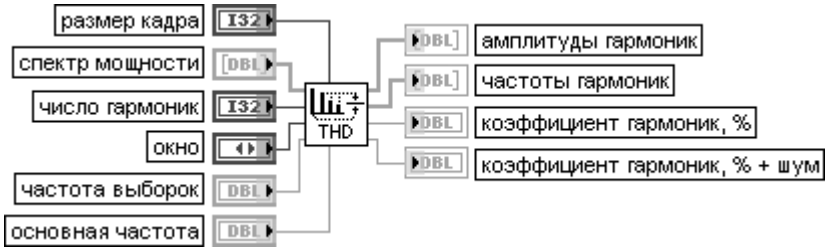
7 **В² пик/Гц**
Vpk^2/Hz volts (squared peak per Hz)

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**

Harmonic Analyzer



Анализатор гармоник



ВП определяет амплитуду и частоту основной и остальных гармонических составляющих, присутствующих во входном **спектре мощности** (A uto Power Spectrum), а также рассчитывает **коэффициент гармоник** (%THD) и **сумму коэффициента гармоник и шума** (%THD + Noise).

Вход **размер кадра** (frame size) задает размер массива данных, подаваемых на вход **спектр мощности** ВП. Если этот вход не подключен, то размер кадра устанавливается в два раза больше, чем размер массива на входе **спектр мощности**.

На вход **спектр мощности** (Auto Power Spectrum) подается односторонний спектр мощности сигнала, обработанного функцией окна.

Вход **число гармоник** (**#** harmonics) определяет число гармонических составляющих, по которым должен определяться коэффициент гармоник. Это число включает и основную частоту.

Вход **окно** (window) определяет тип окна, используемого в ВП **Масштабированное временное окно** (Scaled Time Domain Window).

Вход **частота выборок** (sampling rate) задает частоту дискретизации в герцах.

Вход **основная частота** (fundamental frequency) служит для подачи основной частоты, которую предполагается использовать для определения гармонических составляющих и коэффициента гармоник. При подаче на данный вход нуля (по умолчанию) в качестве основной частоты принимается частота наибольшего компонента, найденного в **спектре мощности**.

На выход **амплитуды гармоник** (Harmonic Amplitudes) подается массив амплитуд основной частоты и ее гармоник. Значения амплитуд положительны и имеют размерность [В эфф.], если задаваемые на входе **спектр мощности** значения имеют размерность [$В^2$ эфф.].

Выход **частоты гармоник** (Harmonic Frequencies) отображает значения основной частоты и ее гармоник. Эти значения имеют размерность [Гц], если **частота выборок** задана в герцах.

На выходе **коэффициент гармоник** (% THD) отображается коэффициент гармоник, расчет которого производится следующим образом:

$$\%THD = \frac{100\sqrt{A(f_2)^2 + A(f_3)^2 + ... + A(f_n)^2}}{A(f_1)},$$

где $A(f_1)$ – амплитуда основной компоненты, $A(f_N)$ – амплитуда N-ой компоненты, значение N определяется входом **число гармоник**.

Выход **коэффициент гармоник + шум** (% THD+ Noise) представляет суммарное значение коэффициента гармоник и шума на входе **спектр мощности**. Значение **% THD+ Noise** рассчитывается по формуле:

$$\%\,THD + Noise = \frac{100\sqrt{sum(APS)}}{A(f_1)},$$

где *sum*(*APS*) является суммой компонентов **спектра мощности**, за исключением мощности постоянной составляющей и мощности компоненты на основной частоте. Для корректного использования ВП **Анализатор гармоник** (Harmonic Analyzer) сигнал с выхода тестируемого нелинейного блока должен быть предварительно обработан с помощью ВП **Масштабированное временное окно** (Scaled Time Domain Window) и **Спектр мощности** (Auto Power Spectrum)

3.1.4. 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. 3.17а, 3.17б и 3.17в 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** находятся в подпалитре **Согласование осциллограмм** (Waveform Conditioning) палитры **Анализ** (Analyze).

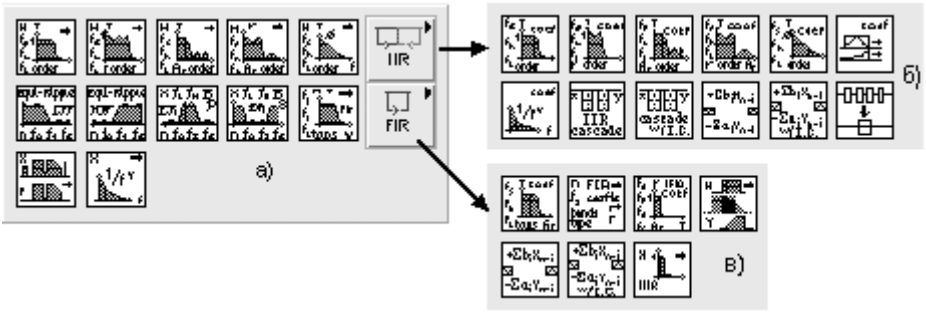


Рис. 3.17. Вид основной палитры (а) и дополнительных подпалитр (б, в) функций фильтров

Классификация цифровых фильтров приведена на рис. 3.18.

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 [9, 10]:

$$y_i=\sum_{j=0}^{N} b_j x_{i-j}-\sum_{k=1}^{M} a_k y_{i-k},$$

где N – порядок полинома прямой ветви, M – порядок полинома обратной ветви.

Рис. 3.18. Классификация цифровых фильтров

У линейных фильтров с конечной импульсной характеристикой (КИХ-фильтров) коэффициенты полинома обратной ветви равны 0 и текущее значение выхода зависит только от 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. 3.19) характеризуется гладкостью на всех частотах и монотонностью спада, начинающегося 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. 3.20) 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).

Рис. 3.19. Амплитудно-частотная характеристика фильтра Баттерворта

Рис. 3.20. Амплитудно-частотная характеристика фильтра Чебышева

Инверсные фильтры Чебышева (фильтры Чебышева второго типа) (рис. 3.21) 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.

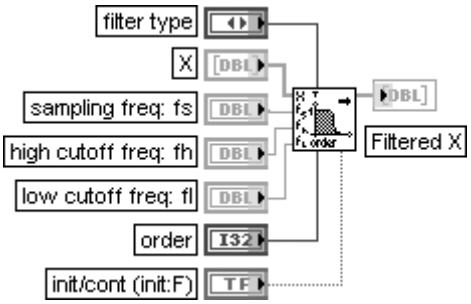
Рис. 3.21. Амплитудно-частотная характеристика инверсного фильтра Чебышева

Эллиптические фильтры минимизируют величину ошибки, распределяя ее по полосе пропускания и полосе заграждения. По сравнению с фильтрами Баттерворта и фильтрами Чебышева эллиптические фильтры обеспечивают самую высокую крутизну переходной области.

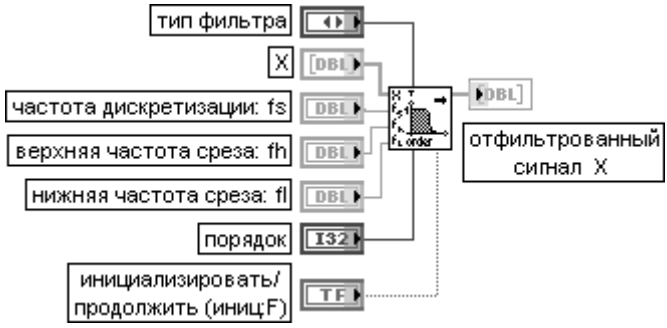
Фильтры Бесселя отличаются от рассмотренных выше фильтров большей линейностью фазочастотной характеристики в полосе пропускания.

ВП фильтров Баттерворта, Чебышева, Бесселя и эллиптических-фильтров имеют схожий состав входов и выходов, отличаясь небольшими деталями.

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. 3.22).

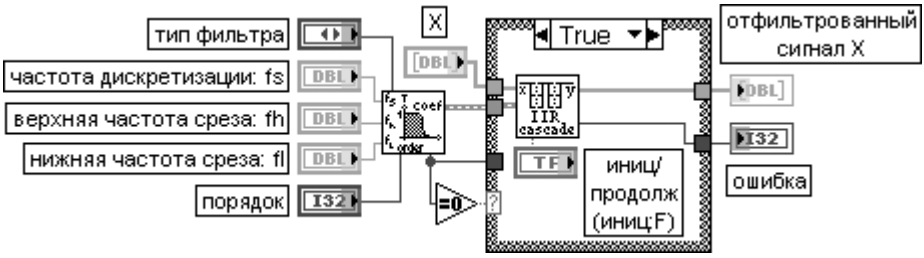


*Рис. 3.22. Блок-диаграмма ВП **Фильтр Баттерворта** (Butterworth Filter)*

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 должна соответствовать критерию Найквиста $0 < f_h < 0{,}5\, f_s$.
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. 3.17б), 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.

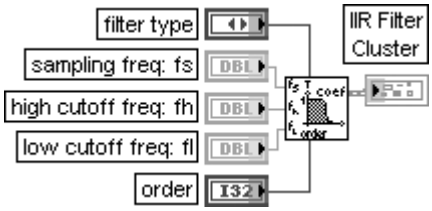
Выход **коэффициенты прямой передачи** (Forward Coefficients) отображает коэффициенты прямой передачи каскадного БИХ-фильтра.

Исходя из блок-диаграммы ВП, второй порядок устанавливается для фильтров типа 0 или 1, а четвертый порядок устанавливается 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**.

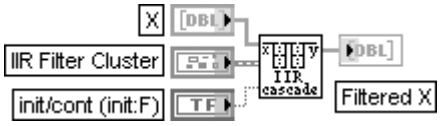
Набор и содержание входов и выходов ВП **Коэффициенты Баттерворта** (Butterworth Coefficients) и **Каскадный БИХ-фильтр** (IIR Cascade Filter) приведены в нижней части данной таблицы.

Butterworth Coefficients

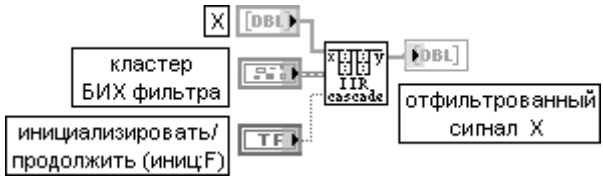


Коэффициенты Баттерворта

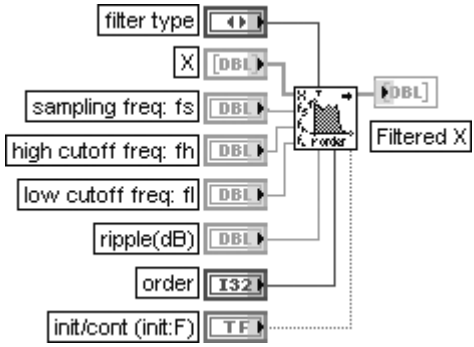
IIR Cascade Filter



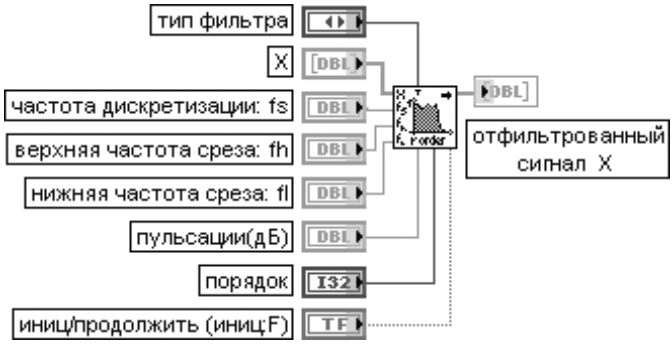
Каскадный БИХ-фильтр



Chebyshev Filter



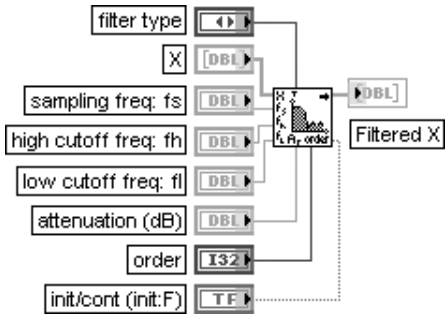
Фильтр Чебышева



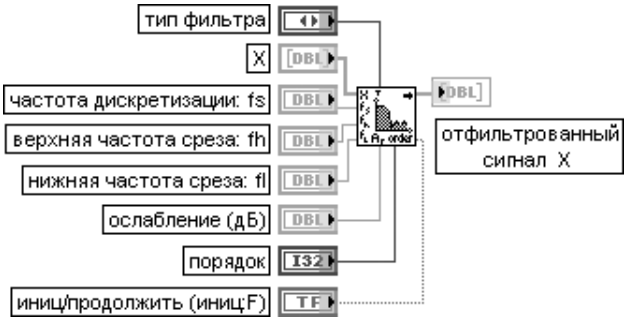
ВП выполняет функцию цифрового фильтра Чебышева путем вызова ВП **Коэффициенты Чебышева** (Chebyshev Coefficients) и ВП **БИХ каскадный фильтр** (IIR Cascade Filter).

Перечень входов ВП идентичен рассмотренному выше ВП **Фильтр Баттерворта** (Butterworth Filter), 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



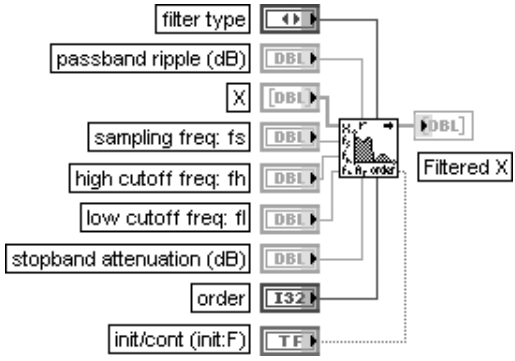
Инверсный фильтр Чебышева



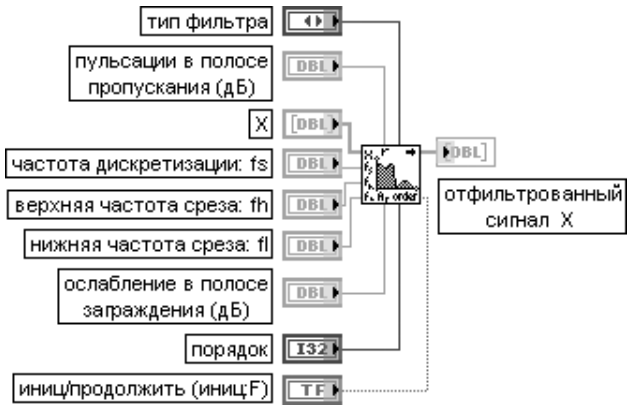
ВП выполняет функцию цифрового инверсного фильтра Чебышева путем вызова ВП **Коэффициенты инверсного фильтра Чебышева** (Inverse Chebyshev Coefficients) и ВП **БИХ каскадный фильтр** (IIR Cascade Filter).

Перечень входов ВП идентичен рассмотренному выше ВП **Фильтр Чебышева** (Chebyshev Filter), 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

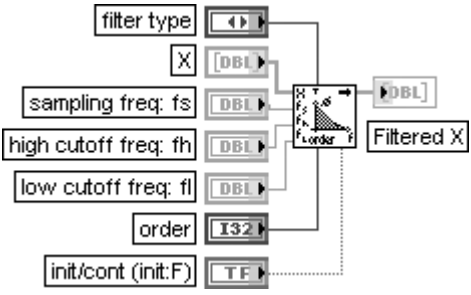


Эллиптический фильтр

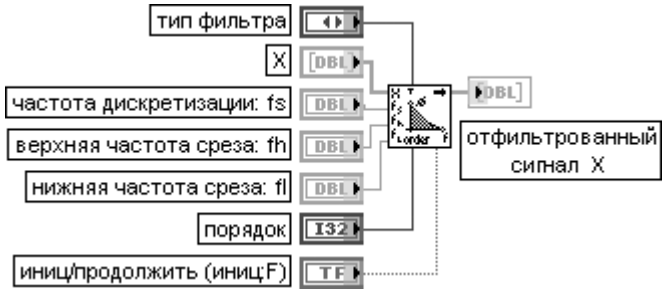


ВП выполняет функцию цифрового эллиптического фильтра путем вызова ВП **Коэффициенты эллиптического фильтра** (Elliptic Coefficients) и ВП **БИХ каскадный фильтр** (IIR Cascade Filter). Особенностью входов данного ВП является то, что с помощью входа **пульсации в полосе пропускания (дБ)** (passband ripple (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



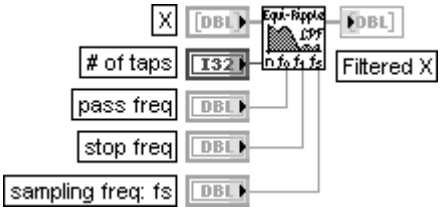
Фильтр Бесселя



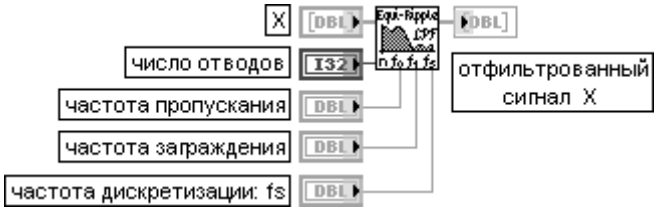
ВП выполняет функцию цифрового фильтра Бесселя путем вызова ВП **Коэффициенты Бесселя** (Bessel Coefficients) и ВП **БИХ каскадный фильтр** (IIR Cascade Filter).

Перечень входов ВП идентичен рассмотренному выше ВП **Фильтр Баттерворта** (Butterworth Filter)

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. 3.23).

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: $задержка = (m + 1)/2$.

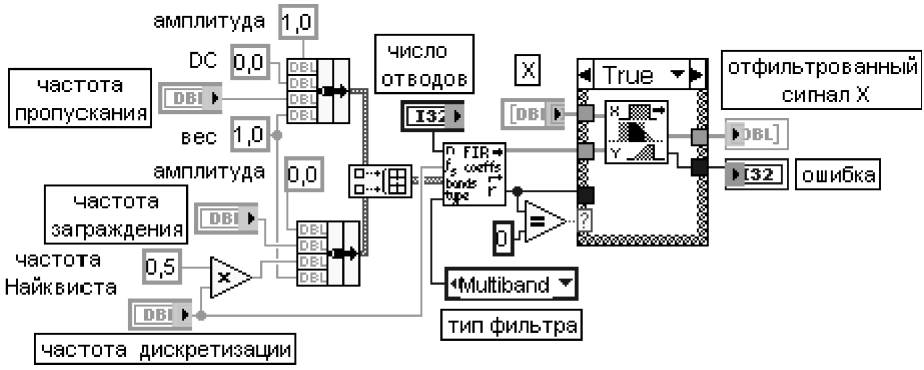
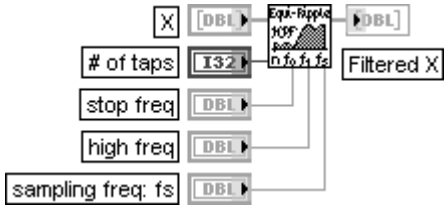


Рис. 3.23. Блок-диаграмма ВП
Фильтр нижних частот с равномерными пульсациями
(Equi-Ripple LowPass)

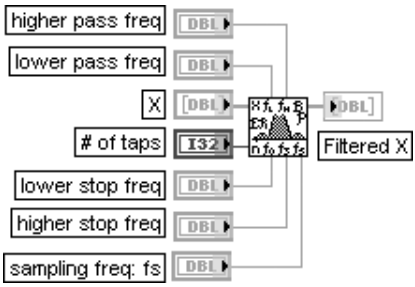
Equi-Ripple HighPass



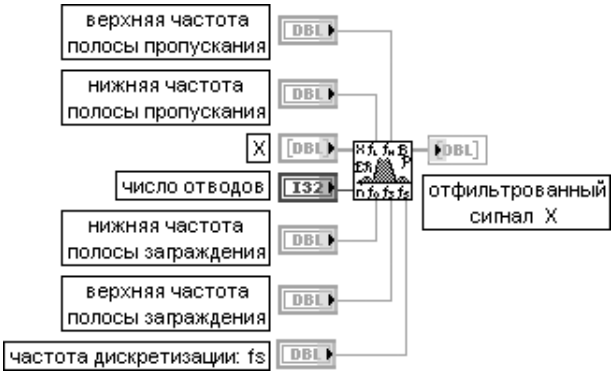
Фильтр верхних частот с равномерными пульсациями

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.

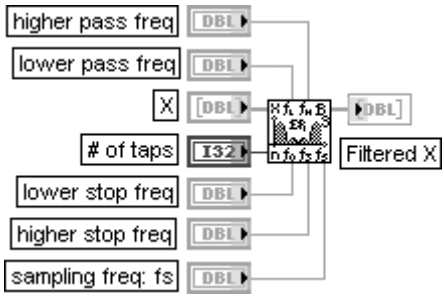
Отличие данного ВП от двух рассмотренных выше аналогичных ВП заключается в увеличении числа входов, определяющих характерные частоты фильтра.

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**.

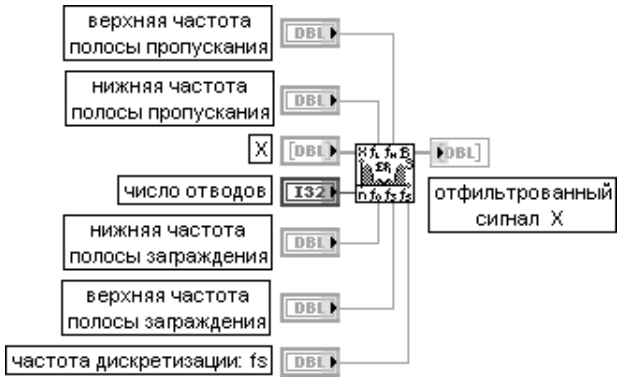
Между этими частотами для полосового фильтра должны выполняться следующие соотношения:

нижняя частота полосы заграждения < нижняя частота полосы пропускания < верхняя частота полосы пропускания < верхняя частота полосы заграждения < 0,5 · частота дискретизации: fs

Equi-Ripple BandStop

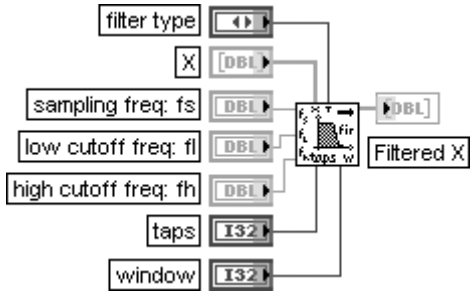


Режекторный фильтр с равномерными пульсациями

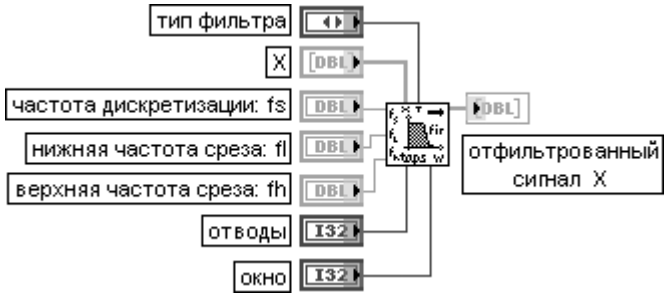


ВП реализует режекторный КИХ-фильтр с равномерными пульсациями АЧХ.
Для режекторного фильтра между характерными частотами должны выполняться следующие соотношения: **нижняя частота полосы пропускания < нижняя частота полосы заграждения < верхняя частота полосы заграждения < верхняя частота полосы пропускания < 0,5 · частота дискретизации: fs**

FIR Windowed Filter



Оконный КИХ-фильтр



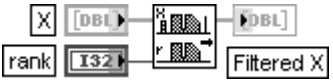
ВП фильтрует входную последовательность данных **X**, используя набор коэффициентов оконного КИХ-фильтра. Коэффициенты определяются значениями на входах **частота**

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).
Вход **отводы** (taps) определяет общее число КИХ-коэффициентов и должен быть больше 0. По умолчанию значение на входе 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



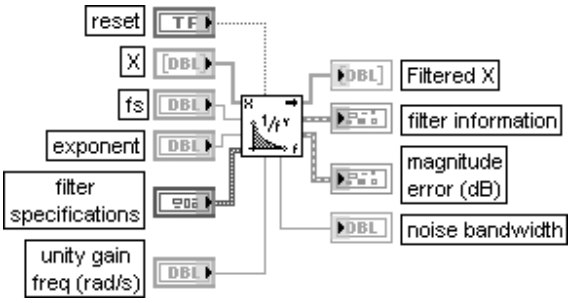
Медианный фильтр

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* $i = 0, n-1$, J_i *– фрагмент входной последовательности* ***X****, центрированный относительно i-го элемента* ***X****.* J_i *задается следующим выражением:*

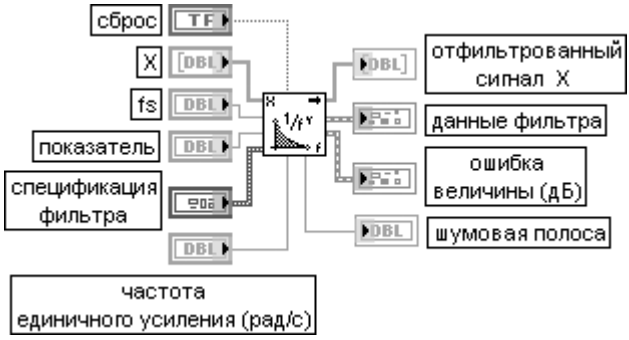
$$J_i = \{x_{i-r}, x_{i-r+1}, ..., x_{i-1}, x_i, x_{i+1}, ..., x_{i+r-1}, x_{i+r}\},$$

где *r* – ранг фильтра

Inverse f Filter



1/f фильтр



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 **спецификация фильтра** (filter specifications) определяет диапазон рабочих частот и порядок фильтра. В состав кластера **спецификация фильтра** входят следующие элементы:

DBL	**нижняя частота среза** (lower cutoff freq) определяет нижнюю граничную частоту диапазона рабочих частот фильтра. По умолчанию значение параметра равно 0,1
DBL	**верхняя частота среза** (higher cutoff freq) определяет верхнюю граничную частоту диапазона рабочих частот фильтра. По умолчанию значение параметра равно 100
I32	**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 **спецификацией 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. Кластер **данные фильтра** содержит следующие элементы:

	величина (magnitude) возвращает величину частотной передаточной характеристики разрабатываемого 1/f фильтра в децибелах

DBL	**частота** (frequency) возвращает частоты частотной передаточной характеристики разрабатываемого 1/f фильтра в герцах
DBL	**величина** (magnitude) возвращает величины частотной передаточной характеристики разрабатываемого 1/f фильтра в децибелах
	фаза (phase) возвращает фазу частотной ередаточной характеристики разрабатываемого 1/f фильтра в градусах
DBL	**частота** (frequency) возвращает частоты частотной передаточной характеристики разрабатываемого 1/f фильтра в герцах
DBL	**фаза** (phase) возвращает значения фаз частотной передаточной характеристики разрабатываемого 1/f фильтра в градусах

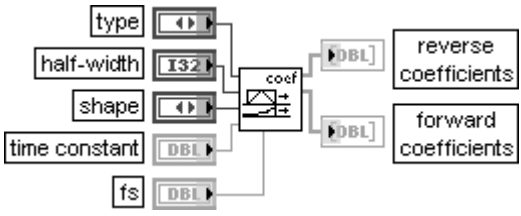
Выход **ошибка величины** (magnitude error) возвращает величину отклонения частотной передаточной характеристики реального 1/f фильтра от идеального

DBL	**частота** (frequency) возвращает набор частот, на которых определяется величина ошибки
DBL	**величина** (magnitude) возвращает величину ошибки в децибелах

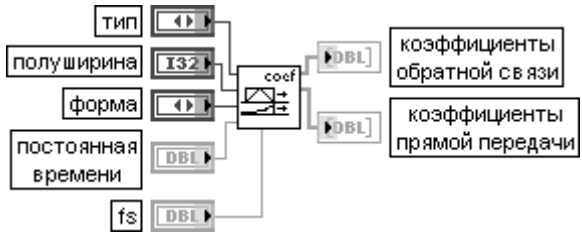
Выход **шумовая полоса** (noise bandwidth) возвращает ожидаемую шумовую полосу создаваемого 1/f фильтра

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, и не требуют отдельного рассмотрения. Аналогичное замечание относится и к ВП **Каскадный БИХ-фильтр** (IIR Cascade Filter), являющемуся 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.

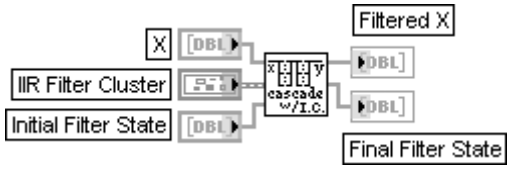
0	**скользящее усреднение** (moving average) (по умолчанию) – рассчитывает только коэффициенты прямой передачи (КИХ-коэффициенты)
1	**экспоненциальное усреднение** (exponential) – рассчитывает БИХ-коэффициенты первого порядка

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.

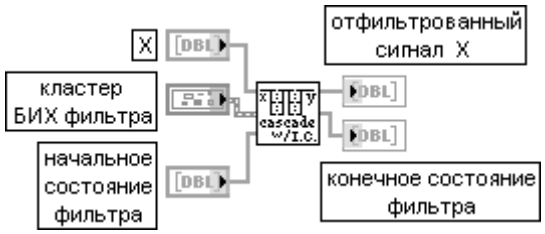
0	**прямоугольное** (rectangular) (по умолчанию) – все выборки в скользящем окне берутся с равными весами при расчете каждого сглаженного выходного отсчета
1	**треугольное** (triangular) – скользящее окно имеет треугольный закон распределения весовых коэффициентов, при котором максимальное значение находится 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

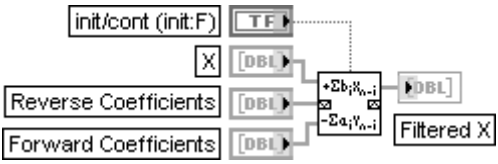


ВП производит фильтрацию входной последовательности **X,** используя каскадную форму БИХ-фильтра, определенную с помощью входа **Кластер БИХ-фильтра** (IIR Filter Cluster).

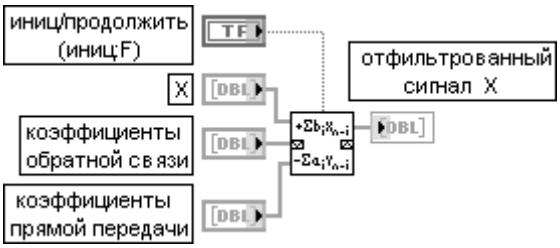
Вход **начальное состояние фильтра** (Initial Filter State) должен быть того же размера, что и массив **коэффициентов обратной связи** (Reverse Coefficients) в **кластере БИХ-фильтра**.

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



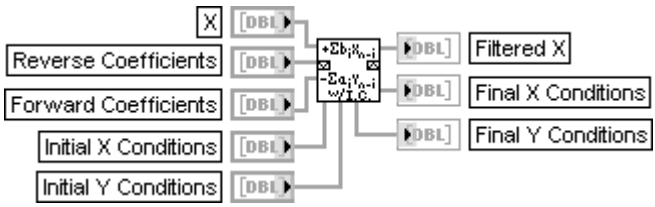
ВП производит фильтрацию входной последовательности **X**, используя прямую форму БИХ-фильтра, определенную с помощью **коэффициентов обратной связи** (reverse coefficients) и **коэффициентов прямой передачи** (forward coefficients).

Расчет элементов **отфильтрованного сигнала X** производится с помощью следующего выражения:

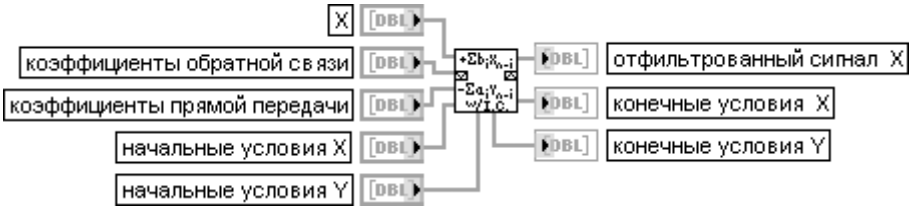
$$y_i = \frac{1}{a_0}\left(\sum_{j=0}^{N} b_j x_{i-j} - \sum_{k=1}^{M} a_k y_{i-k}\right),$$

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

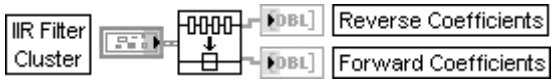


Отличие данного ВП от аналогичного рассмотренного выше ВП **БИХ-фильтр** заключается в том, что при фильтрации блоков непрерывных данных возможно инициализировать фильтр при $i < 0$ с помощью следующих выражений:

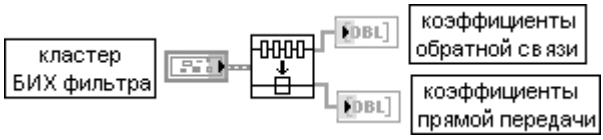
$$y_i = y_{ic}[M + i - 1];\ x_i = x_{ic}[N + i - 1],$$

где y_{ic} представляет массив **начальные условия** Y, а x_{ic} – массив **начальные условия** X

Cascade->Direct Coefficients



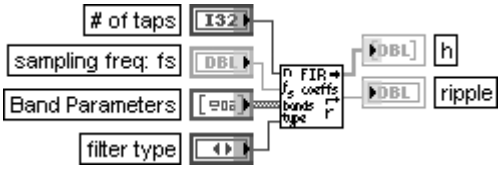
Коэффициенты из каскадной формы в прямую



ВП преобразует коэффициенты БИХ-фильтра из каскадной формы в прямую.

Из подпалитры **Дополнительная КИХ-фильтрация** (Advanced FIR Filtering) далее рассмотрен ВП **Паркс – Мак-Клеллан** (Parks-McClellan), включенный в состав ВП фильтров с равномерными пульсациями (Equi-Ripple Filter) (рис. 3.23) 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.

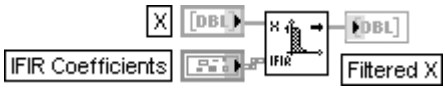
Вход **тип фильтра** (filter type) может иметь следующие значения:

0	1	2
Многополосный (Multiband) (по умолчанию)	**Дифференциатор** (Differentiator)	**Гильберт** (Hilbert)

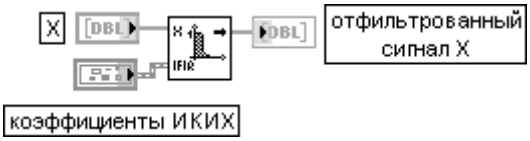
Выход **h** отображает массив коэффициентов КИХ-фильтра, которые ВП рассчитывает, используя алгоритм Паркса – Мак-Клеллана с техникой замены Ремеза.

Выход **пульсация** (ripple) отображает оптимальное отклонение, которое рассчитывается ВП и служит показателем отклонения от идеальной спецификации фильтра

FIR Narrowband Filter

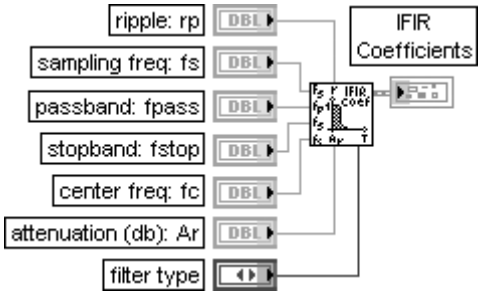


Узкополосный КИХ-фильтр

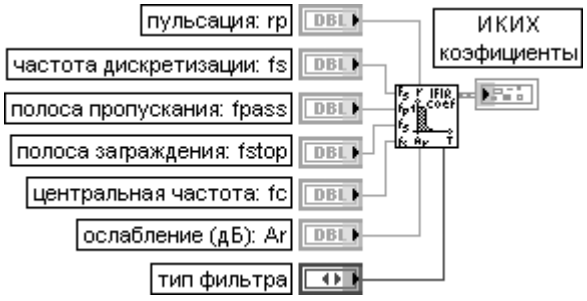


ВП производит фильтрацию входной последовательности **X** с помощью **интерполированного КИХ-фильтра** (interpolated FIR (IFIR) filter), заданного **коэффициентами ИКИХ** (IFIR Coefficients)

FIR Narrowband Coefficients



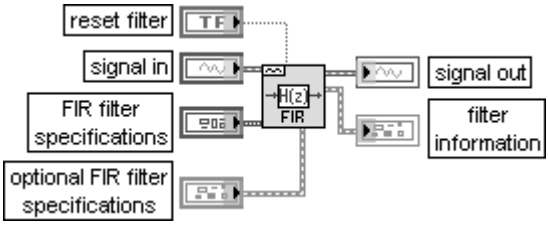
Коэффициенты узкополосного КИХ-фильтра



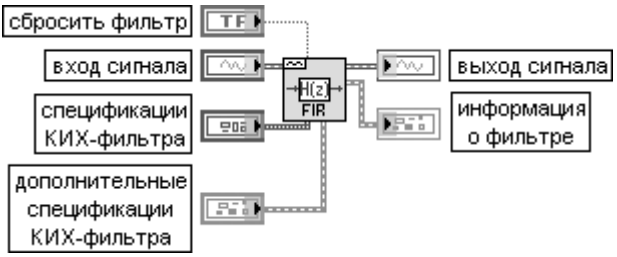
ВП формирует набор коэффициентов фильтра для реализации цифрового **интерполированного КИХ-фильтра** (interpolated FIR (IFIR) filter)

Ниже рассмотрены функции фильтров, находящиеся 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 к 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

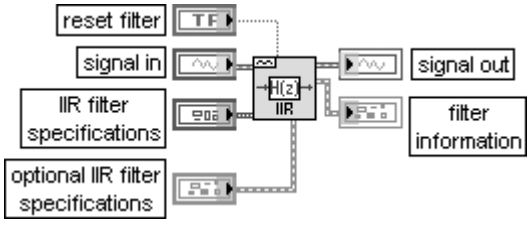
0	1	2	3
Фильтр нижних частот (Lowpass)	Фильтр верхних частот (Highpass)	Полосовой фильтр (Bandpass)	Режекторный фильтр (Bandstop)

I32	**Число отводов** (#Taps) задает число отводов КИХ-фильтра. По умолчанию равно 50
DBL	**Нижняя PB** (Lower PB) определяет нижнюю из двух частот полосы пропускания. По умолчанию равна 100 Гц
DBL	**Верхняя PB** (Upper PB) определяет верхнюю из двух частот полосы пропускания. По умолчанию равна 0
DBL	**Нижняя SB** (Lower SB) определяет нижнюю из двух частот полосы заграждения. По умолчанию равна 200 Гц
DBL	**Верхняя SB** (Upper SB) определяет верхнюю из двух частот полосы заграждения. По умолчанию равна 0

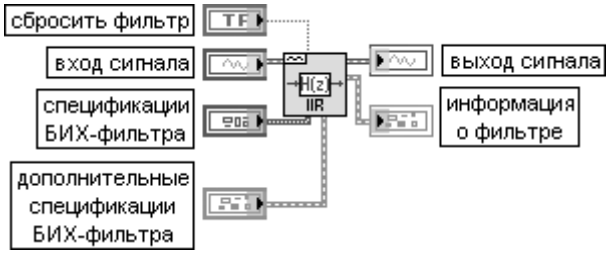
Вход **дополнительные спецификации КИХ-фильтра** (optional FIR filter specifications) представляет кластер дополнительных параметров, которые могут использоваться при задании КИХ-фильтра

DBL	**Усиление PB** (PB Gain) определяет усиление на частотах пропускания. Усиление может быть задано в линейных единицах или децибелах. По умолчанию равно –3 дБ
DBL	**Усиление SB** (SB Gain) определяет усиление на частотах заграждения. Усиление также может быть задано в линейных единицах или децибелах. По умолчанию равно –60 дБ
TF	**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 к входам **сигнал** и **спецификация БИХ-фильтра,** определяют используемую реализацию полиморфного ВП.
Вход **спецификации 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 **дополнительную спецификацию 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 спецификации 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)

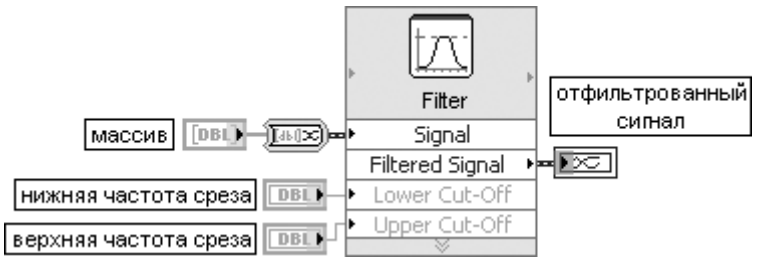


Рис. 3.24. Блок-диаграмма возможного подключения Экспресс-ВП

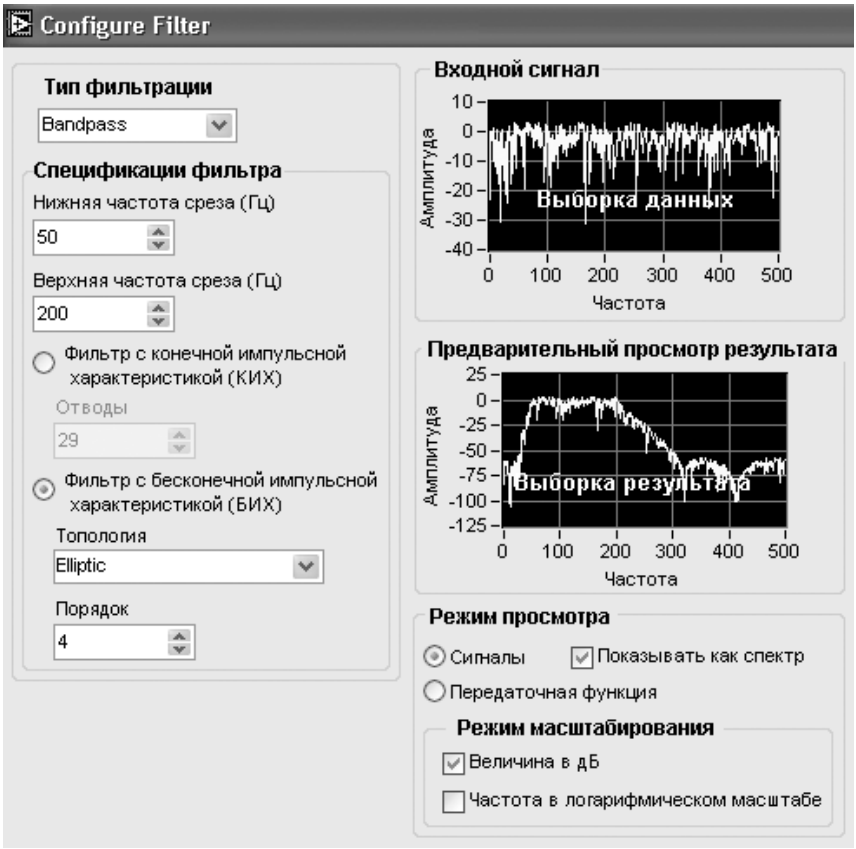


Рис. 3.25. Вид диалогового окна конфигурирования Экспресс-ВП ***Фильтр*** *(Filter)*

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. 3.24, 3.25) далее не приводятся

3.1.5. Функции обработки весовыми окнами

Функции обработки весовыми окнами (рис. 3.26) 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 [9]. Рассмотренные ниже окна как раз и отличаются различным соотношением степени подавления боковых лепестков и расширения центрального лепестка частотной характеристики.

В работе [8] приведен график зависимости приведенной ширины F_h основного лепестка АЧХ от уровня боковых лепестков h_L для различных весовых окон (рис. 3.27). Приведенная ширина основного лепестка равна произведению ширины основного лепестка и длительности функции окна.

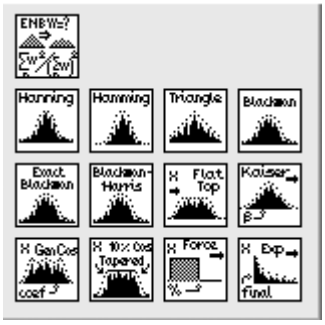


Рис. 3.26. Вид палитры функций обработки весовыми окнами

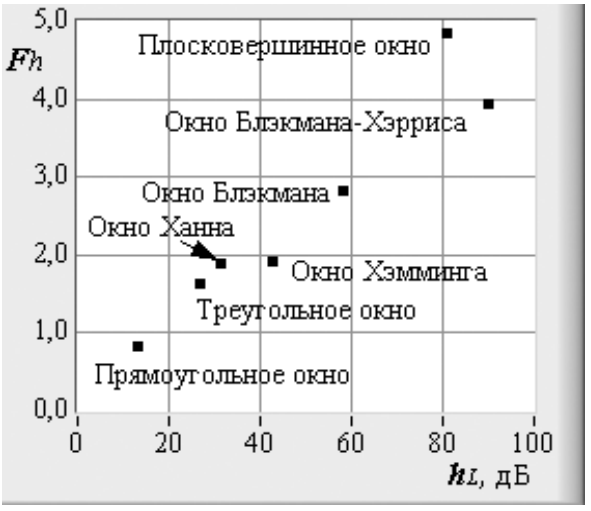


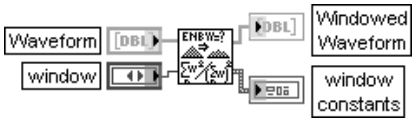
Рис. 3.27. Сравнительный анализ параметров весовых окон

В выражениях, приводимых при анализе рассмотренных ниже окон, 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 приведены в таблице.

0	**Прямоугольное** (Uniform)	3	**Блэкмана-Хэрриса** (Blackman-Harris)	6	**Плосковершинное** (Flat Top)
1	**Ханна** (Hanning)	4	**Точное Блэкмана** (Exact Blackman)	7	**Четырехзвенное Блэкмана-Хэрриса** (Four Term Blackman-Harris)
2	**Хэмминга** (Hamming)	5	**Блэкмана** (Blackman)	8	**Семизвенное Блэкмана-Хэрриса** (Seven Term Blackman-Harris)
				9	**Low Sidelobe**

Графики окон с номерами, соответствующими номерам в таблице, приведены на рис. 3.28 и 3.29.

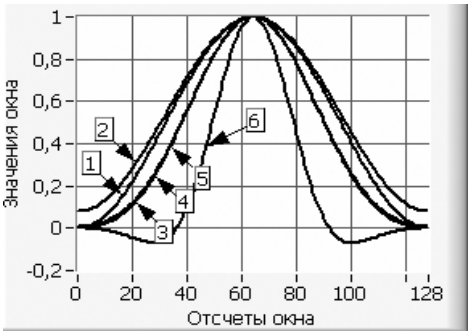


Рис. 3.28. Весовые окна Хана (1), Хэмминга (2), Блэкмана-Хэрриса (3), Точное Блэкмана (4), Блэкмана (5), Плосковершинное (6)

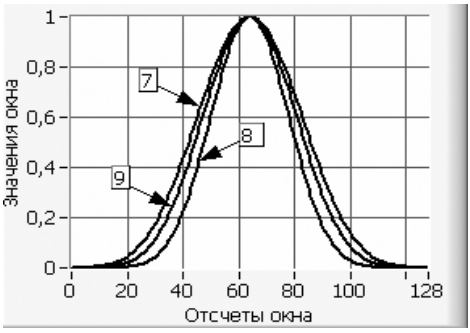


Рис. 3.29. Весовые окна Четырехзвенное Блэкмана-Хэрриса (7), Семизвенное Блэкмана-Хэрриса (8), Low Sidelobe (9)

Выход **оконная осциллограмма** (Windowed Waveform) отображает результаты обработки заданным окном **входной осциллограммы** (Waveform).
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;
coherent gain представляет величину, обратную масштабному фактору окна

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

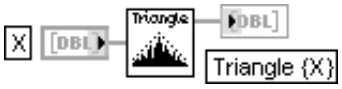
$$y_i = 0{,}5x_i[1-\cos(w)]$$

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

$$y_i = x_i[0{,}54 - 0{,}46\cos(w)]$$

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

$$y_i = \begin{cases} x_i \dfrac{2i}{n} & 0 \le i \le \dfrac{n}{2} \\ x_i \dfrac{2(n-i)}{n} & \dfrac{n}{2} < i \le n-1 \end{cases} \quad для\ \ четного\ \ n,$$

$$y_i = \begin{cases} x_i \dfrac{2i}{n-1} & 0 \le i \le \dfrac{n-1}{2} \\ x_i \dfrac{2(n-i)}{n-1} & \dfrac{n+1}{2} \le i \le n-1 \end{cases} \quad для\ \ нечетного\ \ n,$$

где $i = \overline{0,n-1}$, n – число элементов последовательности

Blackman Window **Окно Блэкмана**

ВП применяет окно Блэкмана для обработки входного сигнала **X**.
Если y представляет выходную последовательность, обработанную **окном Блэкмана** (Blackman {X}), то ВП получает ее значения с помощью выражения

$$y_i = x_i[0{,}42 - 0{,}5\cos(w) + 0{,}08\cos(2w)]$$

Exact Blackman Window **Точное окно Блэкмана**

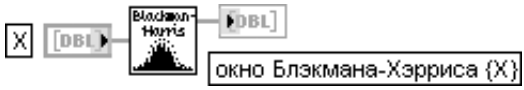
ВП применяет точное окно Блэкмана для обработки входного сигнала **X**.
Если y представляет выходную последовательность, обработанную **точным окном Блэкмана** (Exact Blackman {X}), то ВП получает ее значения с помощью выражения

$$y_i = x_i[a_0 - a_1\cos(w) + a_2\cos(2w)],$$

где $a_0 = 7938/18608$, $a_1 = 9240/18608$, $a_2 = 1430/18608$

Blackman-Harris Window

Окно Блэкмана-Хэрриса

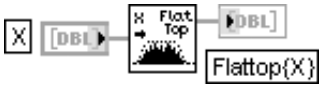


ВП применяет окно Блэкмана-Хэрриса для обработки входного сигнала **X**.
Если y представляет выходную последовательность, обработанную **окном Блэкмана-Хэрриса** (Blackman-Harris {X}), то ВП получает ее значения с помощью выражения

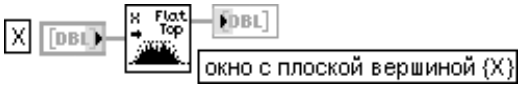
$$y_i = x_i[a_0 - a_1\cos(w) + a_2\cos(2w)],$$

где $a_0 = 0{,}4223223$, $a_1 = 0{,}49755$, $a_2 = 0{,}07922$

Flat Top Window



Плосковершинное окно



ВП применяет плосковершинное окно для обработки входного сигнала **X**.
Если y представляет выходную последовательность, обработанную **плосковершинным окном** (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 $a_0 = 0{,}21557895$, $a_1 = 0{,}41663158$, $a_2 = 0{,}277263158$, $a_3 = 0{,}083578947$, $a_2 = 0{,}006947368$

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. 3.30. приведены графики окон Кайзера-Бесселя при β = 1 ÷ 5

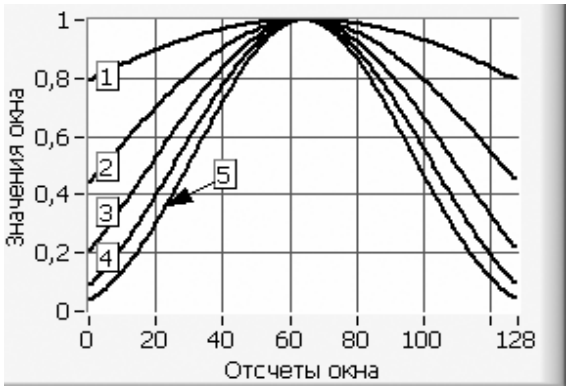


Рис. 3.30. Графики окон Кайзера-Бесселя при β = 1 ÷ 5

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

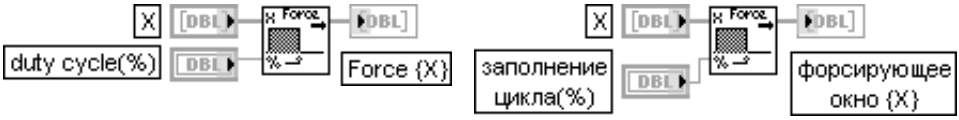
Косинусное 10% окно

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

$$y_i = \begin{cases} 0.5x_i(1-\cos w) & i = 0,1,2..m-1 \quad u \quad i = n-m, n-m+1,..n-1 \\ x_i & m \le i \le n-m-1 \end{cases},$$

где $m = round\left(\frac{n}{10}\right)$, n – число элементов последовательности X

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

$$y_i = \begin{cases} x_i & если\ 0 \le i \le d \\ 0 & иначе \end{cases},$$

где $d = 0.01n(duty\ cycle)$, $i = \overline{0,n-1}$, 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 $i = \overline{0,n-1}$

3.2. Математические функции

3.2.1. Функции численных методов и решения дифференциальных уравнений

Функции численных методов (рис. 3.31) позволяют выполнять численное интегрирование массивов входных данных, рассчитывать значения как самой функции, заданной формулой, так и ее интеграла и производной, а также такие параметры, как предел и длину кривой. В состав палитры входят также функции расчета частных производных, поиска экстремумов двумерной функции и поиска нулей и экстремумов одномерной функции. Набор функций численных методов решения дифференциальных уравнений включает ВП, реализующие методы Эйлера, Рунге-Кутта и Кэш-Капа, а также ВП решения линейного ОДУ n-го порядка и системы линейных ОДУ в численном и символьном виде.

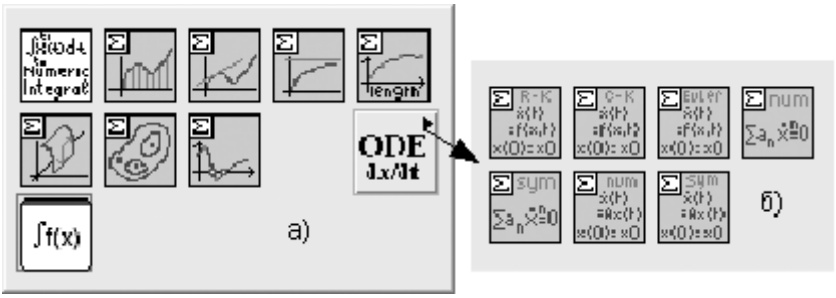


Рис. 3.31. Вид основной палитры (а) и дополнительной подпалитры (б) функций численных методов и решения дифференциальных уравнений

Numeric Integration **Численное интегрирование**

ВП выполняет численное интегрирование массива входных данных с помощью одного из четырех числовых методов интегрирования.

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.

Вход **метод интегрирования** (integration method) определяет метод, используемый для численного интегрирования (таблица).

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 $i = 0, 1, 2, 3...$ *Целая часть* $[(N-1)/k]$,

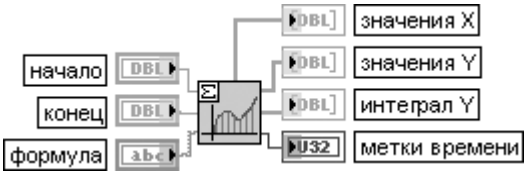
где N – число отсчетов данных, k – целое, зависящее от метода, и x – входной массив. Если число отсчетов при выборе определенного метода не позволяет сформировать целое число частичных сумм, то этот метод применяется для точек, вошедших в частичные суммы. Для оставшихся точек применяется метод более 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

Integration

Интегрирование



ВП рассчитывает значения функции и интеграла одномерной функции от **начала** (start) до **конца** (end). Функция задается с помощью **формулы** (formula). Число рассчитываемых значений зависит от сложности данной функции.

Вход **начало** (start) задает начальную точку интервала. По умолчанию его значение равно 0,0.

Вход **конец** (end) задает конечную точку интервала. По умолчанию его значение равно 1,0.

Вход **формула** (formula) представляет строку, описывающую исследуемую функцию. При записи формулы необходимо использовать только допустимые имена переменных.

Выход **значения X** (X Values) отображает массив точек, расположенных в интервале (**начало**, **конец**).

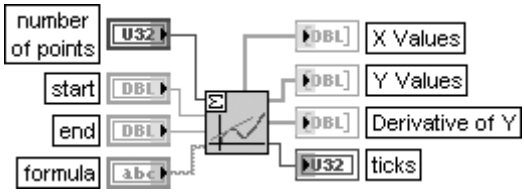
Выход **значения Y** (Y Values) отображает значения функции.

Выход **интеграл Y** (Integral of Y) представляет значения интеграла выражения, заданного с помощью формулы между **началом** и **концом** при всех **значениях X**.

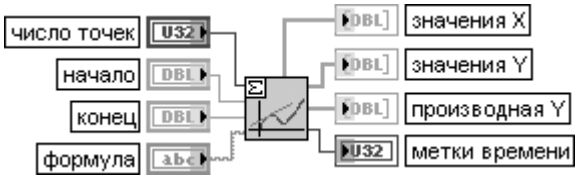
Выход **метки времени** (ticks) отображает время в миллисекундах, затраченное на анализ формулы и расчет массива **значений X** и массива значений **интеграла Y**.

Расчет интеграла $I = \int_{start}^{end} f(t)dt$ функции **f(t)** производится с помощью решения обыкновенного дифференциального уравнения $\frac{dI(s)}{ds} = f(s)$ при $I(start) = 0$. При решении дифференциального уравнения используется метод Рунге-Кутта

Differentiation



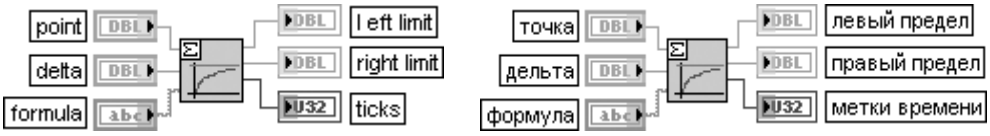
Дифференцирование



ВП рассчитывает значения функции и значения производной одномерной функции, заданной формулой, в равномерно расположенных точках интервала.

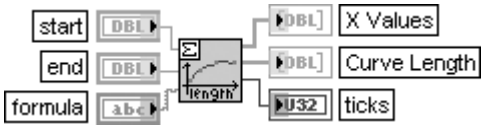
Вход **число точек** (number of points) задает число всех рассчитываемых точек. Независимая переменная разделяется на равновеликие интервалы. Значение по умолчанию равно 10.
Входы **начало**, **конец**, **формула** и выходы **метки времени**, **значения X**, **значения Y** идентичны одноименным входам и выходам рассмотренного выше ВП **Интегрирование** (Integration).
Выход **производная Y** (Derivative of Y) отображает значения производной функции в точках **значения X**.

Limit **Предел**

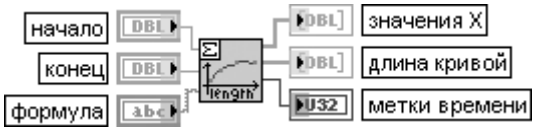


ВП определяет левый и правый пределы одномерной функции в заданной точке.
Вход **точка** (point) определяет точку, в которой рассчитываются пределы. По умолчанию значение равно 0,0.
Вход **дельта** (delta) задает расстояние между левой и правой границами окрестности **точки**. По умолчанию значение равно 1E –10.
Вход **формула** (formula) представляет строку, описывающую исследуемую функцию.
Выходы **левый предел** (left limit) и **правый предел** (right limit) определяют, соответственно, левый и правый пределы заданной функции в **точке**. Точность параметров превышает 8 десятичных цифр.
Выход **метки времени** (ticks) определяет время анализа формулы и расчета пределов. Обычно это время незначительно.
Алгоритм рассчитывает только два значения **f**(**точка** – **дельта**) и **f**(**точка** + **дельта**).
Очень малое значение **дельта** может привести к числовым погрешностям. Значение **дельта** необходимо выбирать равным 1E – 10 во всех случаях

Curve Length



Длина кривой

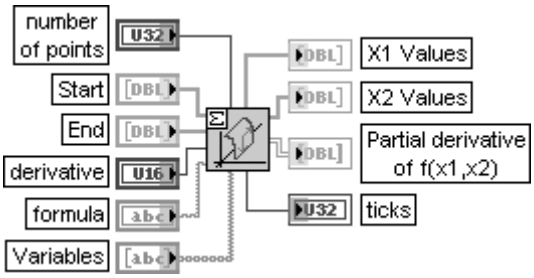


ВП рассчитывает длину кривой одномерной функции между точками **начало** (start) и **конец** (end).
Входы **начало**, **конец**, **формула** и выходы **значения X**, **метки времени** идентичны одноименным входам и выходам рассмотренного выше ВП **Интегрирование** (Integration).
Выход **длина кривой** (Curve Length) представляет массив значений длины кривой, заданной **формулой**, между точками **начало** и конец при всех **значениях X**.
ВП рассчитывает **длину кривой** заданной функции $f(t)$ между точками **начало** и **конец,** используя следующее выражение:

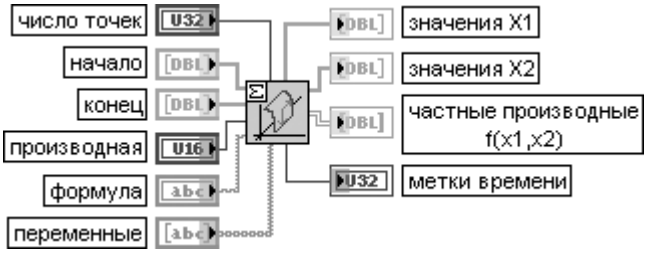
$$L = \int_{start}^{end} \sqrt{1 + \left(\frac{df(t)}{dt}\right)^2}\, dt.$$

Этот расчет осуществляется с помощью ВП **Интегрирование**, в основе работы которого лежит метод Рунге-Кутта

Partial Derivatives of f(x1,x2)

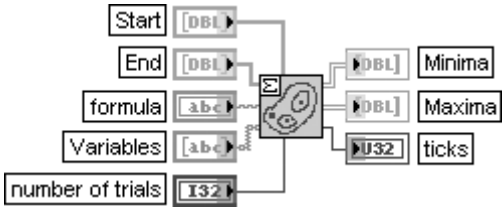


Частные производные f(x1,x2)

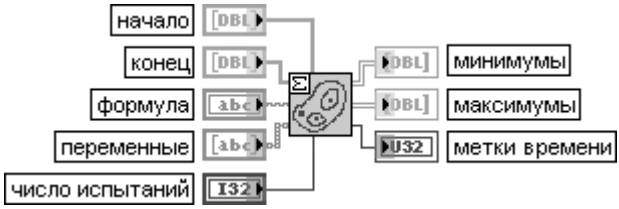


ВП рассчитывает частные производные функции двух независимых переменных.
Вход **число точек** (number of points) описывает число точек сетки для обеих переменных. По умолчанию число равно 25.
Вход **начало** (Start) определяет начальные точки обеих переменных и представляет массив из двух элементов. По умолчанию это массив (0,0).
Вход **конец** (End) определяет конечные точки обеих переменных и также представляет массив из двух элементов. По умолчанию это массив (1,1).
Вход **производная** (derivative) определяет вид рассчитываемой частной производной. Значение 0 представляет частную производную первой переменной. Значение 1 представляет частную производную второй переменной.
Вход **формула** (formula) представляет строку, описывающую функцию.
Вход **переменные** (Variables) является массивом из двух строк, представляющих две переменные в соответствии с **соглашением о именах** (naming conventions). По умолчанию используются переменные $(x1, x2)$.
Выходы **значения X1** (X1 Values) и **значения X2** (X2 Values) представляют результирующие одномерные массивы.
Выход **частная производная** $f(x1, x2)$ (Partial derivative of $f(x1, x2)$) отображает значения частных производных в фиксированных точках сетки в виде двумерного массива. При значении **производная**, равном 0, рассчитывается функция $f(x1, x2)/x1$. При значении **производная**, равном 1, рассчитывается функция $f(x1, x2)/x2$

Extrema of f(x1,x2)



Экстремумы f(x1,x2)



ВП ищет локальные экстремумы заданной функции двух переменных в заданном прямоугольнике. Абсолютное расстояние между двумя экстремумами должно быть больше или равно 1E – 6.

Входы **начало**, **конец**, **формула** и **переменные** идентичны одноименным входам рассмотренного выше ВП **Частные производные f(x1,x2)** (Partial Derivatives of f(x1,x2)).

Вход **число испытаний** (number of trials) задает число случайно выбранных двумерных начальных точек алгоритма.

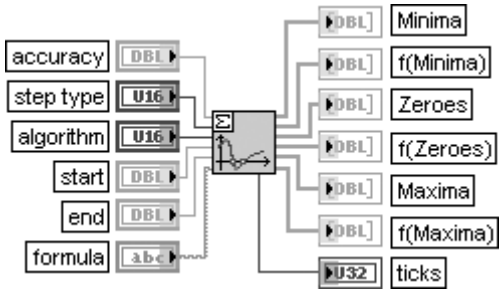
Выходы **минимумы** (Minima) и **максимумы** (Maxima) представляют двумерные массивы соответственно минимумов и максимумов заданной функции. Все локальные минимумы и максимумы определяются двумя координатами.

Несмотря на большое число испытаний, не существует гарантии, что могут быть найдены все, или по крайней мере один нуль, или локальный экстремум заданной функции.

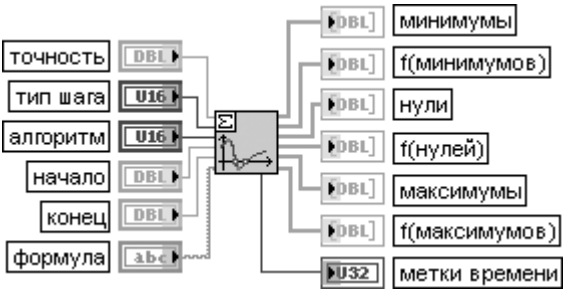
Хотя случайно выбранные начальные точки алгоритма поиска экстремумов принадлежат заданному прямоугольнику, точки экстремумов могут оказаться за пределами этого прямоугольника.

В этом случае полученные значения также отображаются

Zeroes and Extrema of f(x)

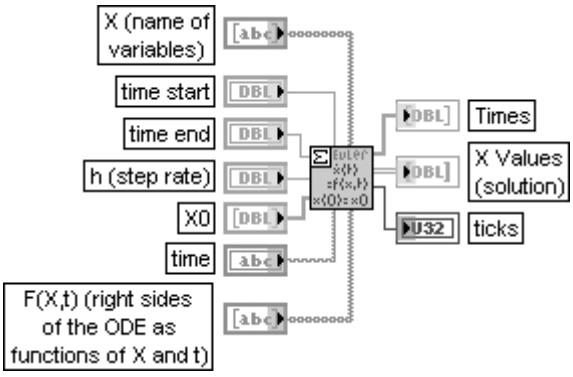


Нули и экстремумы f(x)

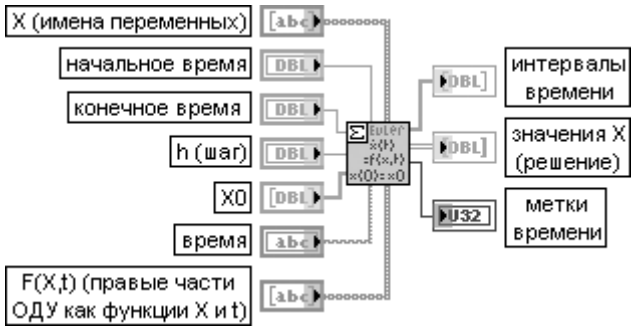


ВП определяет нули и экстремумы одномерной функции на заданном интервале.
Вход **точность** (accuracy) устанавливает точность нулей и экстремумов. По умолчанию значение равно 1E – 8.
Вход **тип шага** (step type) устанавливает вид размещения значений функции. При установке 0 на входе **тип шага** используются равномерно расположенные значения функции. При установке на данном входе 1 используется модифицированная функция с оптимальным размером шага. По умолчанию установлено значение 0. В общем случае выбор модифицированной функции ведет к более точному определению нулей и экстремумов.
Вход **алгоритм** (algorithm) определяет метод, используемый в ВП. При установке 0 на входе **алгоритм** выбирается метод Риддера. При установке 1 выбирается метод Ньютона-Рафсона. По умолчанию установлено значение 0.
Функции входов **начало**, **конец** и **формула** идентичны одноименным входам рассмотренных выше ВП.
Выходы **минимумы** (Minima) представляют значения минимумов выражения, заданного с помощью формулы, а **f**(**минимумов**) – значения функции в этих минимумах.
Выходы **нули** (Zeroes) отображают нули выражения, заданного с помощью формулы, а выходы **f**(**нулей**) – значения функции в этих точках. Как правило, эти значения близки к 0.
Выходы **максимумы** (Maxima) представляют значения максимумов выражения, заданного с помощью формулы, а **f**(**максимумов**) – значения функции в этих максимумах

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 его значение равно 0.

Вход **конечное время** (time end) определяет конечную точку временного интервала решения. По умолчанию его значение 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 **интервал 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:

$$\begin{cases} \dot{x}_1(t) = f_1(x_1(t),...x_n(t),t) \\ \quad\quad\quad\quad . \\ \quad\quad\quad\quad . \\ \dot{x}_n(t) = f_n(x_1(t),...x_n(t),t) \end{cases} \qquad \begin{cases} x_1(t_0) = x_{10} \\ \quad . \\ \quad . \\ x_n(t_0) = x_{n0} \end{cases}$$

Предполагается, что функции $f_1,...f_n$, начальные условия $x_{10},...x_{n0}$ и начальный момент времени заданы. Необходимо определить вид зависимостей $x_1(t),...x_n(t)$.

Используя обозначения $F = (f_1,...f_n)$, $X = (x_1(t),...x_n(t))$ и $X_0 = (x_{10},...x_{n0})$, систему уравнений можно записать в векторной форме

$$\begin{cases} \dot{X}(t) = F(X(t),t) \\ \quad X(t_0) = X_0 \end{cases}.$$

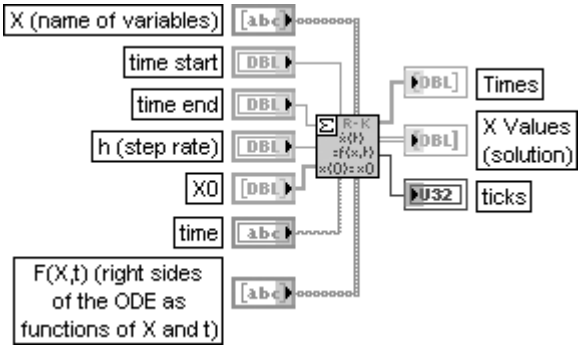
Метод Эйлера является наиболее простым методом решения обычных дифференциальных уравнений (ОДУ). Его суть заключается в использовании при пошаговом интегрировании зависимой переменной двух первых членов ряда Тейлора. Таким образом, итерационная процедура вычисления значений $X(t_i)$ при начальном моменте времени t_0 и достаточно малом шаге интегрирования h может быть записана в следующем виде:

$$X(t_0 + h) = X(t_0) + hF(X(t_0), t_0);$$

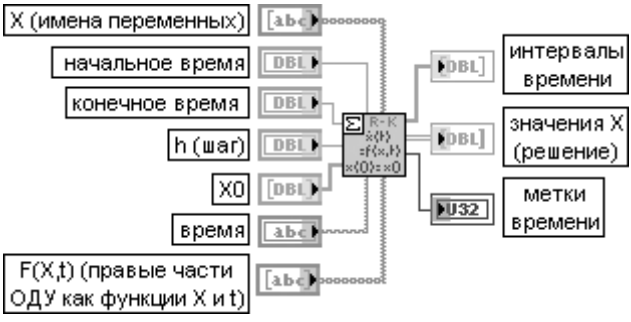
$$X(t_0 + 2h) = X(t_0 + h) + hF(X(t_0 + h), t_0 + h).$$

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-Кутта 4-го порядка



ВП решает обыкновенные дифференциальные уравнения (ОДУ) с начальными условиями, используя метод Рунге-Кутта. Метод Рунге-Кутта 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-Кутта имеет 4-ый порядок, и итерационные расчетные выражения выглядят следующим образом:

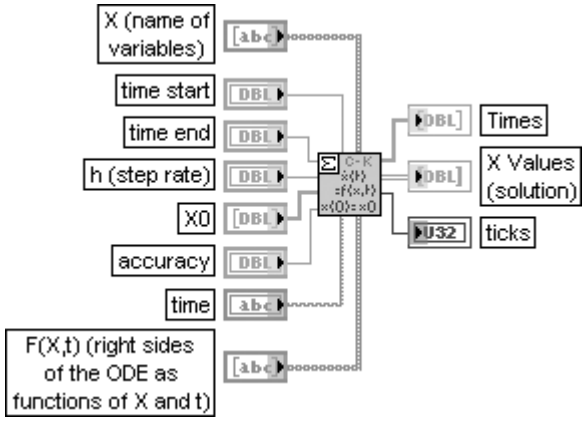
$$K_1 = hF(X(t_n), t_n) \qquad K_2 = hF(X(t_n) + \frac{K_1}{2}, t_n + \frac{h}{2})$$

$$K_3 = hF(X(t_n) + \frac{K_2}{2}, t_n + \frac{h}{2}) \qquad K_4 = hF(X(t_n) + K_3, t_n + h)$$

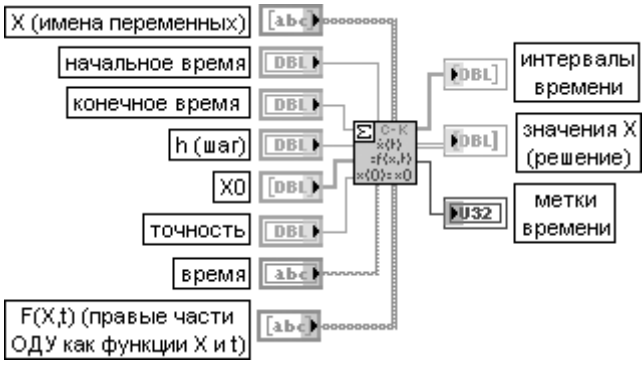
$$X(t_{n+1}) = X(t_n) + \frac{K_1}{6} + \frac{K_2}{3} + \frac{K_3}{3} + \frac{K_4}{6}$$

Этот процесс останавливается, когда **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-го 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-Кутта.

Итерационные расчетные выражения для метода выглядят следующим образом:

$$k_1 = hF(X(t_n), t_n);$$

$$k_2 = hF(X(t_n) + a_2 h, t_n + b_{21}k_1);$$

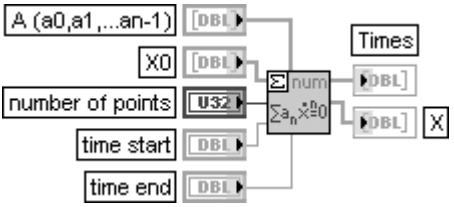
$$k_6 = hF(X(t_n) + a_5 h, t_n + b_{61} + ...b_{65}k_5);$$

и

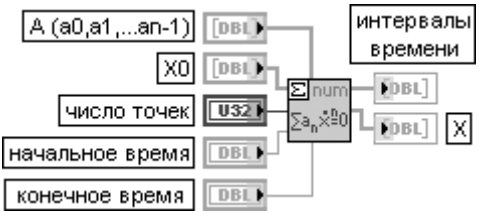
$$X(t_{n+1}) = X(t_n) + c_1 k_1 + ... + c_6 k_6;$$

$$\dot{X}(t_{n+1}) = X(t_n) + c_1^* k_1 + ... + c_6^* k_6$$

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], …, 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 точках на оси времени, заданных в массиве **интервалы времени**.

Решение линейного однородного дифференциального уравнения ***n***-го порядка

$$x^{(n)} + a_{n-1}x^{(n-1)} + ... + a_1 x^{(1)} + a_0 x = 0$$

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 $\lambda_1,...,\lambda_n$ общее решение дифференциального уравнения n-го порядка может быть записано в следующем виде:

$$x(t) = \beta_1 \exp(\lambda_1 t) + ... + \beta_n \exp(\lambda_n t).$$

Неизвестные коэффициенты $\beta_1,...\beta_n$ могут быть определены с помощью начальных условий

$$x(0) = \beta_1 + ... + \beta_n;\ \ x^{(1)}(0) = \beta_1\lambda_1 + ... + \beta_n\lambda_n;\ \ ....;\ \ x^{(n-1)}(0) = \beta_1\lambda_1^{n-1} + ... + \beta n\lambda_n^{n-1},$$

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:

$$x(t) = \beta_1 \exp(\lambda_1 t) + ... + \beta_n \exp(\lambda_n t)$$

с комплексными $\beta_1,...,\beta_n$ и $\lambda_1,...,\lambda_n$.

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, $x[10], ... x[n0]$. 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

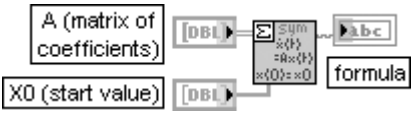
$$\frac{dX(t)}{dt} = AX(t); \quad X(0) = X_0.$$

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 $Y(t)=SX(t)$ приводит запись системы к следующему виду:

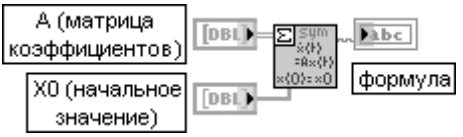
$$\frac{dY(t)}{dt}=SAS^{-1}Y(t); \quad Y(0)=SX_0.$$

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

$$(+1{,}62\cdot\exp(-12{,}46\cdot t)-1{,}28\cdot\exp(-6{,}30\cdot t)+0{,}63\cdot\exp(1{,}34\cdot t)+0{,}04\cdot\exp(5{,}42\cdot t))+(0)$$

$$(+0{,}84\cdot\exp(-12{,}46\cdot t)-0{,}29\cdot\exp(-6{,}30\cdot t)+1{,}51\cdot\exp(1{,}34\cdot t)-0{,}06\cdot\exp(5{,}42\cdot t))+(0)$$

$$(-0{,}73\cdot\exp(-12{,}46\cdot t)+0{,}01\cdot\exp(-6{,}30\cdot t)+3{,}69\cdot\exp(1{,}34\cdot t)+0{,}02\cdot\exp(5{,}42\cdot t))+(0)$$

$$(+0{,}87\cdot\exp(-12{,}46\cdot t)+2{,}67\cdot\exp(-6{,}30\cdot t)+0{,}45\cdot\exp(1{,}34\cdot t)+0{,}01\cdot\exp(5{,}42\cdot t))+(0)$$

Для получения решения необходимо ввести следующие значения:

$$A:[-7,-6,4,-1;-6,2,1,-2;4,1,0,2;-1,-2,2,-7],\ X0[1,2,3,4]$$

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)

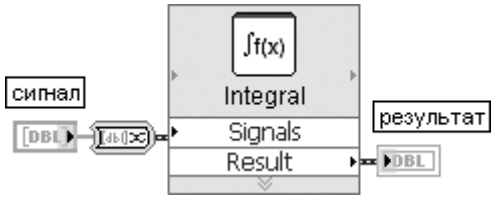


Рис. 3.32. Блок-диаграмма возможного подключения Экспресс-ВП

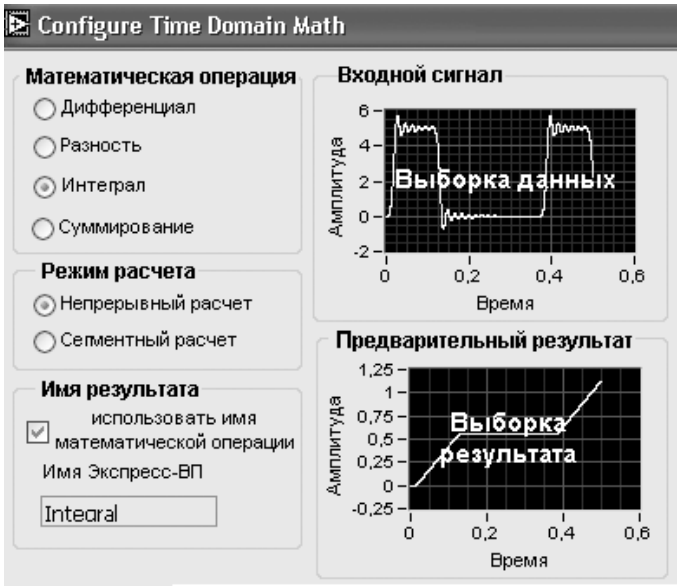


Рис. 3.33. Вид диалогового окна конфигурирования Экспресс-ВП ***Математическая обработка во временной области*** *(Time Domain Math)*

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))

3.2.2. Функции статистической обработки данных

Функции статистической обработки данных (рис. 3.34) позволяют рассчитать основные статистические параметры наборов данных: среднее, среднеквадратичес-

кое отклонение, дисперсию, средний квадрат ошибки, центральные моменты различного порядка, медиану и моду, а также рассчитать значения гистограммы набора данных. В одной из дополнительных подпалитр размещены функции расчета вероятностей (интегральных распределений) для различных законов распределения и функции расчета квантилей распределений по заданным значениям вероятности.

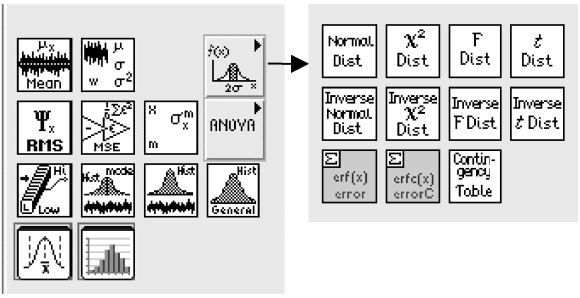
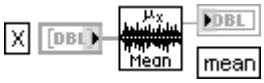


Рис. 3.34. Вид основной палитры и дополнительной подпалитры функций статистической обработки данных

Mean

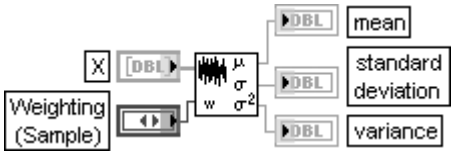


Среднее

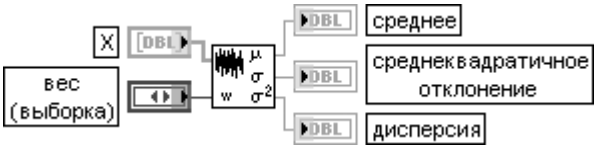
ВП производит расчет **среднего** (mean) входной последовательности **X** в соответствии с выражением

$$среднее = \frac{1}{n}\sum_{i=0}^{n-1} x_i,$$ где n – число элементов последовательности **X**

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 **вес** (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 **вес** по умолчанию – **выборка**.

Значение выхода **среднее** рассчитывается так же, как и в ВП **Среднее** (Mean).

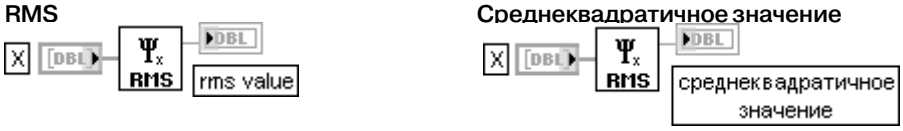
Значение выхода **стандартное отклонение** рассчитывается в соответствии с выражением

$$\sigma=\sqrt{\frac{\sum_{i=0}^{n-1}(x_i-\mu)^2}{w}},$$

где μ – среднее.

Значение выхода **дисперсия** определяется как σ^2

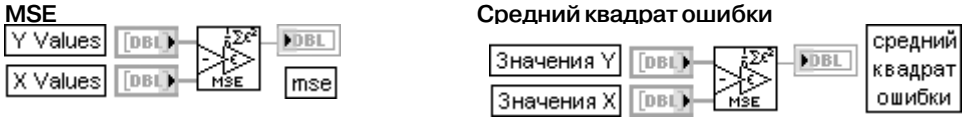
RMS **Среднеквадратичное значение**



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.

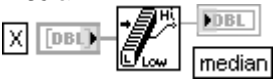
Вход **порядок** должен быть больше 0. Если вход **порядок** меньше или равен 0, то ВП выводит на выход **момент** значение NaN и возвращает ошибку. По умолчанию значение порядка равно 2.

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**

Median



Медиана

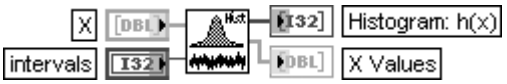
ВП находит **медиану** (median) входной последовательности **X** путем сортировки значений **X** и выбора среднего(их) элемента(ов) из отсортированного массива.

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** (Histogram: h(x)) 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

$$\Delta_i = \left[\chi_i - 0{,}5\Delta x : \chi_i + 0{,}5\Delta x\right)$$

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**

Mode

Мода

ВП находит оценку **моды** (mode) входной последовательности ***X***.
Вход **X** должен содержать по крайней мере одну выборку. Если входная последовательность имеет постоянное значение, 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.
Блок-схема ВП **Мода** (Mode) приведена на рис. 3.35. Значение моды, определенное с помощью гистограммы, является более устойчивым по сравнению с непосредственным определением моды для реальных зашумленных данных.

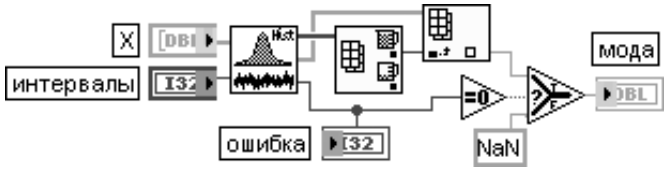
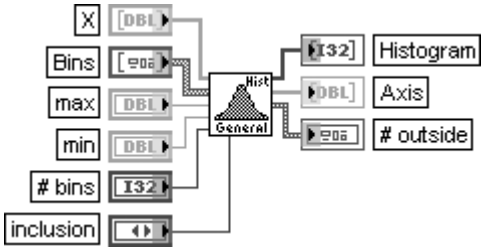
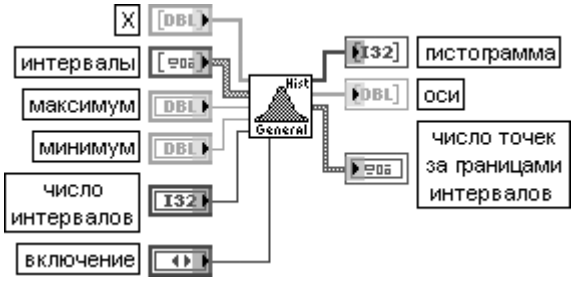


Рис. 3.35. Блок-диаграмма ВП ***Мода*** *(Mode)*

General Histogram



Общая гистограмма



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** (Bins) не заданы, то для определения равномерно распределенных интервалов используются входы **max**, **min**, **# bins** и **inclusion**.

Входы **max** и **min** определяют максимальное и минимальное значения, включаемые в гистограмму. Если **max** и **min** не определены, ВП использует максимальное и минимальное значения входной последовательности **X**.

Вход **число интервалов** (# bins) определяет число интервалов гистограммы. **Число интервалов** 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, 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***-й полосы.
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

$$Bins[0].upper \le Bins[1].lower < Bins[1].upper, \ldots - < Bins[k-1].lower < Bins[k-1].upper,$$

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

$$Bins[0].lower$$

В последующей таблице приведены функции расчета вероятностей (интегральных распределений) для различных законов распределения и функции расчета квантилей распределений по заданным значениям вероятности.

Normal Distribution | **Нормальное распределение**

ВП рассчитывает одностороннюю **вероятность** (probability) ***p*** нормально распределенной случайной переменной ***x***.
Вероятность рассчитывается исходя из выполнения условия

$$p = Prob\{X \le x\},$$

где ***X*** – переменная со стандартным нормальным распределением, ***p*** – вероятность, ***x*** – значение

Chi Square Distribution | **Хи-квадрат распределение**

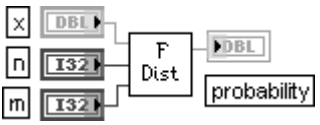
ВП рассчитывает одностороннюю **вероятность** (probability) ***p*** случайной переменной **x**, имеющей хи-квадрат распределение с заданным числом степеней свободы (**degrees of freedom**).

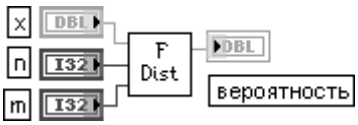
Число степеней свободы должно быть больше 0.
Вероятность рассчитывается исходя из выполнения условия

$$p = Prob\{X \le x\},$$

где ***X*** – переменная, имеющая распределение χ^2 с числом степеней свободы ***n***, ***p*** – вероятность, ***x*** – значение

F Distribution **F-распределение**



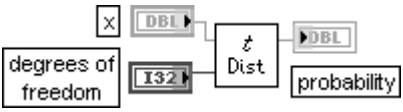


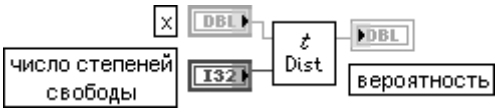
ВП рассчитывает одностороннюю **вероятность** (probability) ***p*** случайной переменной ***x***, имеющей ***F***-распределение с заданным числом **степеней свободы** (degrees of freedom**)** ***n*** и ***m***.
Число степеней свободы ***n*** и ***m*** должно быть больше 0.
Вероятность рассчитывается исходя из выполнения условия

$$p = Prob\{F_{m,n} \le x\},$$

где $F_{m,n}$ – переменная, имеющая F-распределение с числом степеней свободы ***n***,***m, p*** – вероятность, ***n –*** первое число степеней свободы, ***m –*** второе число степеней свободы, ***x*** – значение

T Distribution **t-распределение**



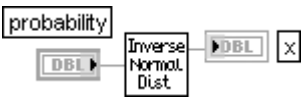


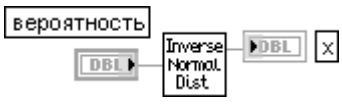
ВП рассчитывает одностороннюю **вероятность** (probability) **p** случайной переменной ***x***, имеющей ***t***-распределение с заданным числом **степеней свободы** (degrees of freedom).
Число степеней свободы ***n*** должно быть больше 0.
Вероятность рассчитывается исходя из выполнения условия

$$p = Prob\{T_n \le x\},$$

где T_n – переменная, имеющая t-распределение с числом степеней свободы ***n, p*** – вероятность, ***n –*** число степеней свободы, ***x*** – значение

Inv Normal Distribution **Обратное нормальное распределение**





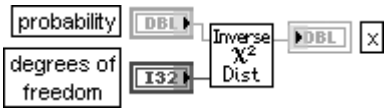
ВП рассчитывает значение ***x*** при заданном значении **вероятности** (probability) ***p*** нормально распределенной случайной величины ***X***.
Величина вероятности должна находиться в диапазоне от 0 до 1.
Выход ***x*** отображает значение квантиля нормально распределенной случайной величины.

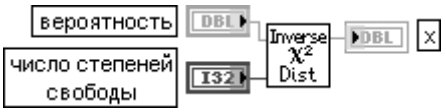
ВП рассчитывает значение **x** таким образом, чтобы выполнялось следующее соотношение

$$p = Prob\{X \le x\},$$

где X – переменная, имеющая стандартное нормальное распределение, ***p*** – вероятность, ***x*** – значение

Inv Chi Square Distribution **Обратное Хи-квадрат распределение**





ВП рассчитывает значение ***x*** при заданном значении **вероятности** (probability) ***p*** случайной величины ***X***, распределенной по закону χ^2 с заданным числом **степеней свободы** (degrees of freedom).

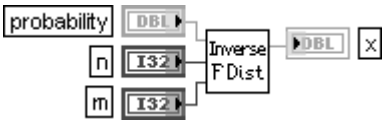
Выход **x** отображает значение квантиля случайной величины, распределенной по закону χ^2.

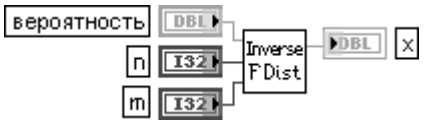
ВП рассчитывает значение **x** таким образом, чтобы выполнялось следующее соотношение

$$p = Prob\{X \le x\},$$

где ***X*** – переменная, имеющая χ^2 распределение, ***p*** – вероятность, ***x*** – значение

Inv F Distribution **Обратное F-распределение**





ВП рассчитывает значение **x** при заданном значении **вероятности** (probability) ***p*** случайной величины ***X***, имеющей ***F***-распределение с заданным числом **степеней свободы** (degrees of freedom) ***n*** и ***m***.

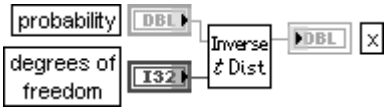
Выход **x** отображает значение квантиля случайной величины, имеющей ***F***-распределение.

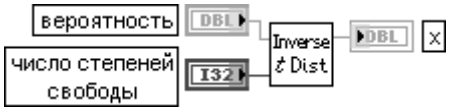
ВП рассчитывает значение **x** таким образом, чтобы выполнялось следующее соотношение

$$p = Prob\{F_{n,m} \le x\},$$

где $F_{n,m}$ – переменная, имеющая F-распределение с числом степеней свободы ***n***, ***m***, ***p*** – вероятность, ***n*** – первое число степеней свободы, ***m*** – второе число степеней свободы, ***x*** – значение

Inv T Distribution **Обратное t-распределение**





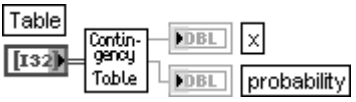
ВП рассчитывает значение ***x*** при заданном значении **вероятности p** случайной величины X, имеющей ***t***-распределение с заданным числом **степеней свободы** (degrees of freedom) ***n***.

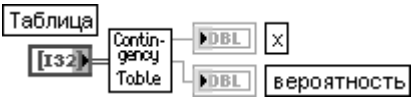
Вероятность рассчитывается исходя из выполнения условия

$$p = Prob\{T \le x\},$$

где T – переменная, имеющая ***t***-распределение с числом степеней свободы ***n***, ***p*** – вероятность, ***n*** – число степеней свободы, ***x*** – значение

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 **probability**, меньше установленного порога, 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 считает их по двум схемам категорирования. ВП проверяет гипотезу о том, что схемы категорирования являются независимыми.

Пусть $y_{p,q}$ – число событий в $(p, q)^{-й}$ ячейке таблицы вероятностей для $p = \overline{0, s-1}$ и $q = \overline{0, k-1}$, где ***s*** и ***k*** – число строк и столбцов в таблице вероятностей **Table**.

Пусть

$$y_p = \sum_{q=0}^{k-1} y_{p,q},\ y_q = \sum_{p=0}^{s-1} y_{p,q},\ y = \sum_{p=0}^{s-1}\sum_{q=0}^{k-1} y_{p,q},\ e_{p,q} = \frac{y_p, y_q}{y},\ x = \sum_{p=0}^{s-1}\sum_{q=0}^{k-1} \frac{\left[y_{p,q} - e_{p,q}\right]^2}{e_{p,q}}$$

ВП использует **x** для расчета **вероятности** (probability) **p** исходя из выполнения следующего соотношения

$$p = Prob\{X \ge x\},$$

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 функций статистической обработки данных входят Экспресс-ВП **Статистика** (Statistics) и **Гистограмма** (Histogram), рассмотренные ниже.

Статистика (Statistics)

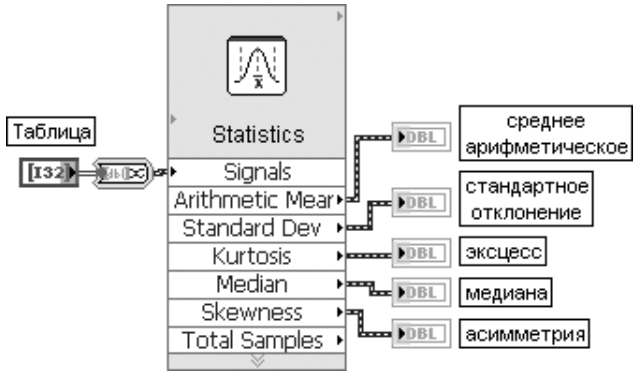


Рис. 3.36. Блок-диаграмма возможного подключения Экспресс-ВП

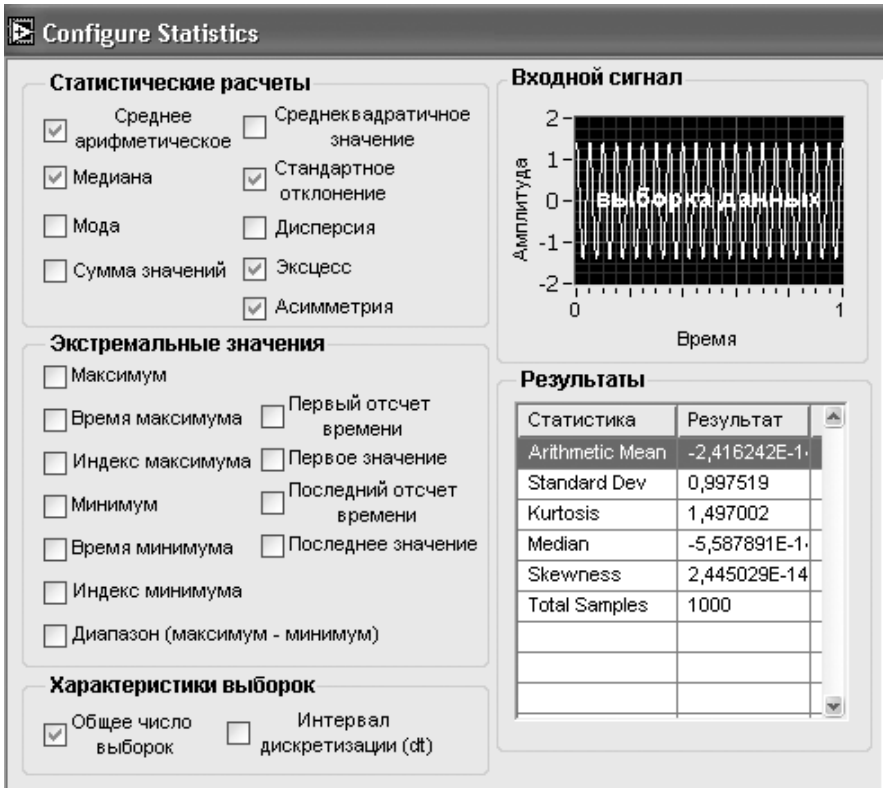


*Рис. 3.37. Вид диалогового окна конфигурирования Экспресс-ВП **Статистика** (Statistics)*

Экспресс-ВП **Статистика** (Statistics) возвращает выбранные параметры первого сигнала осциллограммы. Экспресс-ВП использует функциональность следующих ВП: **Получить компоненты осциллограммы** (Get Waveform Components), **Получить время окончания осциллограммы** (Get Final Time Value), **Число выборок осциллограммы** (Number of Waveform Samples), **Длительность осциллограммы** (Waveform Duration), **Минимум и максимум осциллограммы** (Waveform Min Max), **Детектор пика осциллограммы** (Waveform Peak Detection), **Логическая функция И для элементов массива** (And Array Elements).

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. 3.36, 3.37) далее не приводятся

Гистограмма (Histogram)

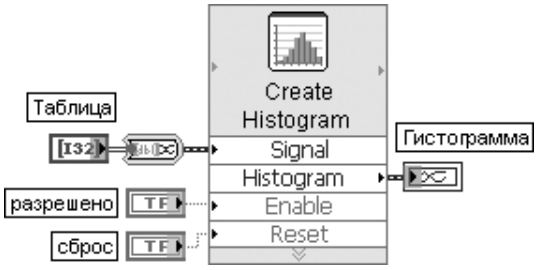


Рис. 3.38. Блок-диаграмма возможного подключения Экспресс-ВП

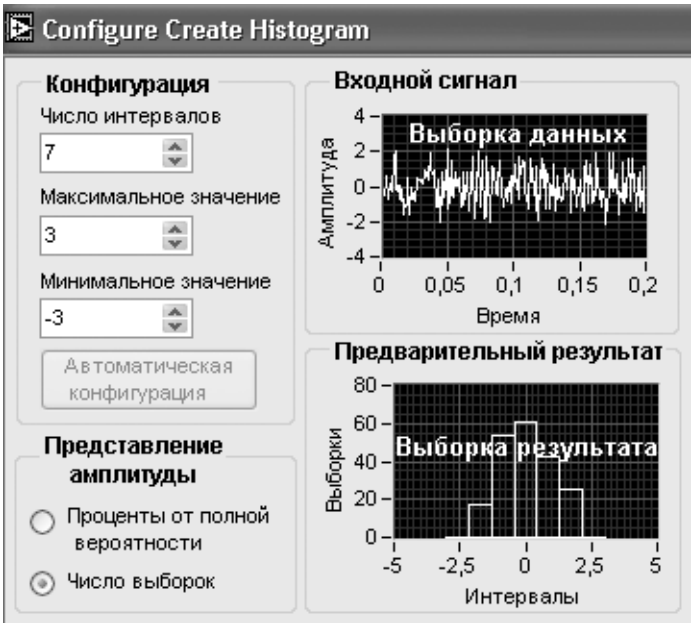


Рис. 3.39. Вид диалогового окна конфигурирования Экспресс-ВП ***Гистограмма*** *(Histogram)*

Экспресс-ВП **Гистограмма** (Histogram) рассчитывает гистограмму входного сигнала. Данный Экспресс-ВП использует функциональность одноименного ВП, рассмотренного выше

3.2.3. Функции сглаживания данных

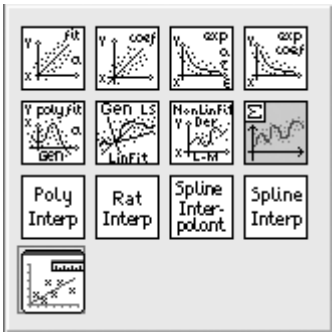
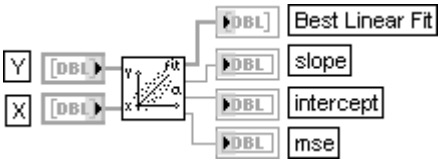


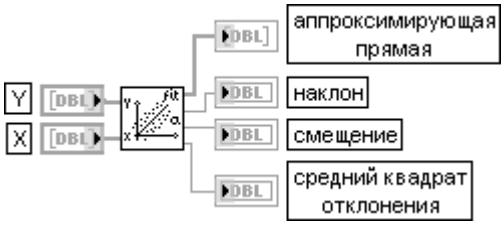
Рис. 3.40. Вид палитры функций сглаживания данных

Функции сглаживания данных (рис. 3.40) позволяют выполнять как аппроксимацию, так и интерполяцию данных. Функции аппроксимации включают линейную, экспоненциальную, полиномиальную и нелинейную аппроксимацию данных, а функции интерполяции – полиномиальную интерполяцию, интерполяцию рациональными полиномами и сплайн-интерполяцию данных. В состав палитры функций входит Экспресс-ВП **Сглаживание кривой** (Curve Fitting), позволяющий реализовать основные методы сглаживания данных.

Ниже в таблице приведены пояснения к функциям сглаживания данных.

Linear Fit **Линейная аппроксимация**





ВП производит аппроксимацию набора входных данных **Y** и **X** линейной функцией по методу наименьших квадратов, выводит значения прямой на выход **аппроксимирующая прямая** (Best Linear Fit) и значения ее коэффициентов на выходы **наклон** (slope) и **смещение** (intercept).

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 **средний квадрат ошибки** (mse) устанавливает значение NaN, а на выходе **ошибка** (error) возвращает ошибку.

Общий вид линейной аппроксимирующей функции задается следующим выражением

$$F = mx + b,$$

где F – аппроксимирующая прямая, m – наклон, b – смещение.

Значение **среднего квадрата ошибки** (mse) рассчитывается по формуле

$$mse = \frac{1}{n}\sum_{i=0}^{n-1}(f_i - y_i)^2,$$

где f_i – значение выходной последовательности **аппроксимирующая прямая**,

y_i – значение входной последовательности Y, $i = \overline{0, n-1}$.

Блок-диаграмма ВП **Линейная аппроксимация** (Linear Fit) приведена на рис. 3.41. Как видно, основным элементом этого ВП является ВП **Коэффициенты линейной аппроксимации** (Linear Fit Coefficients)**,** перечень входов и выходов которого приведен в нижней части таблицы.

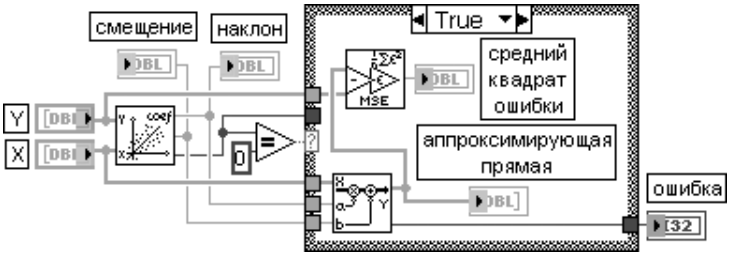
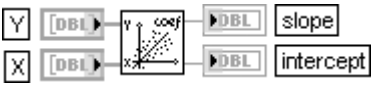
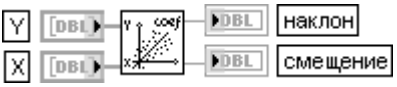


*Рис. 3.41. Блок-диаграмма ВП **Линейная аппроксимация** (Linear Fit)*

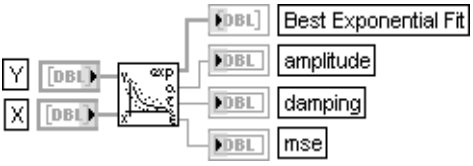
Linear Fit Coefficients



Коэффициенты линейной аппроксимации



Exponential Fit



Экспоненциальная аппроксимация

ВП производит аппроксимацию набора входных данных **Y** и **X** экспоненциальной функцией по методу наименьших квадратов, выводит значения аппроксимирующей функции на выход **аппроксимирующая экспонента** (Best Exponential Fit) и ее коэффициенты на выходы **амплитуда** (amplitude) и **затухание** (damping).

Значения **Y** должны быть одного знака, и их число должно превышать 2.

В общем виде экспоненциальная аппроксимирующая функция задается следующим выражением

$$F = ae^{\tau X},$$

где F – выходная последовательность **аппроксимирующая экспонента**, a – **амплитуда**, τ – **затухание**.

Экспоненциальная аппроксимация предполагает следующий вид модели данных:

$$y = ae^{\tau x},$$

где x – независимая переменная, y – зависимая переменная, a – амплитуда, τ – затухание.

Задачей аппроксимации является определение τ и a для заданных x и y.

Данная задача решается с помощью следующих шагов:
1) берется натуральный логарифм от обеих частей $y = ae^{\tau x}$, $\ln(y) = \ln(a) + \tau x$;
2) считая, что в полученном выражении зависимая переменная $\tilde{y} = \ln(y)$ и независимая переменная x связаны линейно $\tilde{y} = \tau x + \tilde{a}$, где $\tilde{a} = \ln(a)$, выполняется расчет коэффициентов линейной регрессии τ и $\tilde{a}$;
3) производится расчет коэффициента a с помощью выражения $a = e^{\tilde{a}}$.
Расчет **среднего квадрата ошибки** (mse) производится так же, как и в ВП **Линейная аппроксимация**. Блок-диаграмма ВП приведена на рис. 3.42. Она включает ВП **Коэффициенты экспоненциальной аппроксимации** (Exponential Fit Coefficients), перечень входов и выходов которого приведен в нижней части таблицы.

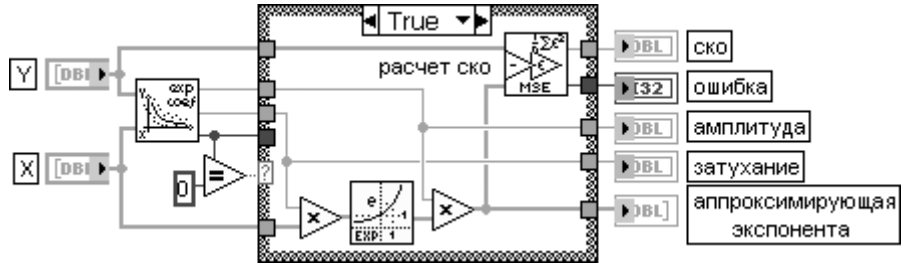
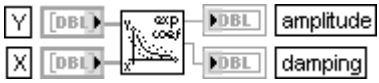


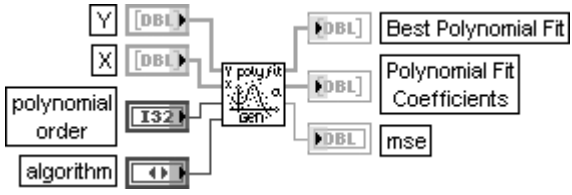
*Рис. 3.42. Блок-диаграмма ВП **Экспоненциальная аппроксимация** (Exponential Fit)*

Exponential Fit Coefficients

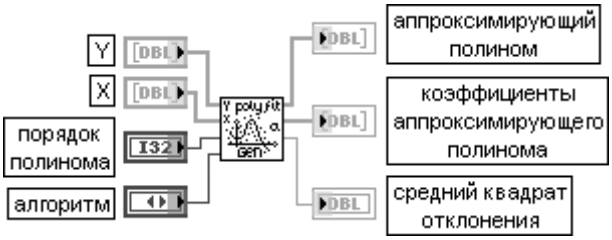


Коэффициенты экспоненциальной аппроксимации

General Polynomial Fit



Общая полиномиальная аппроксимация



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.

Порядок полинома (polynomial order) должен быть больше или равен нулю. По умолчанию порядок полинома равен 2. Еще одно ограничение связано с числом точек 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)	3	Householder
1	Givens	4	LU decomposition
2	Givens2	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**.

Для выполнения своей функции ВП **Общая полиномиальная аппроксимация** использует ВП **Общая линейная аппроксимация методом наименьших квадратов** (General LS Linear Fit) как подпрограмму (рис. 3.43), передавая ей матрицу . Матрица H формируется следующим образом:

$h_{ij} = f_j(x_i) = x_i^j$, где $j = \overline{0,m};\ i = \overline{0,n-1}$.

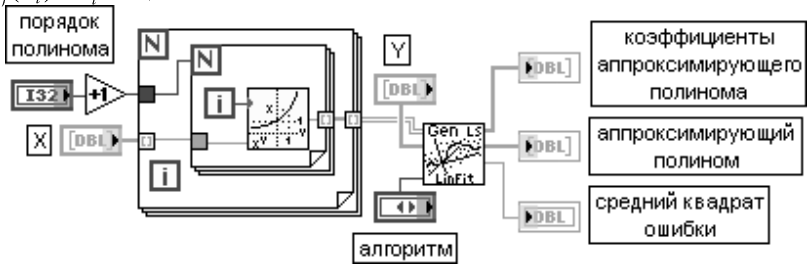
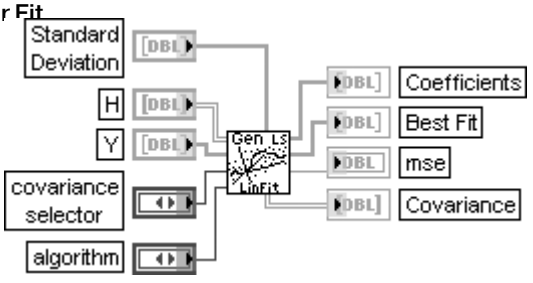
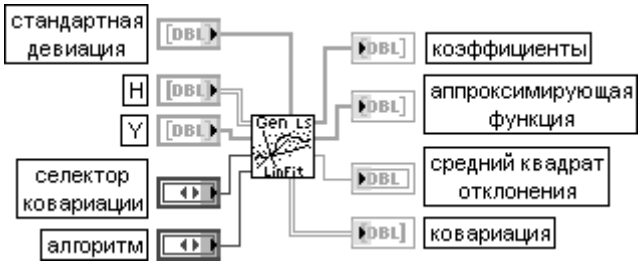


*Рис. 3.43. Блок-диаграмма ВП **Общая полиномиальная аппроксимация***

General LS Linear Fit



Общая линейная аппроксимация методом наименьших квадратов



ВП находит значения ***k***-мерной линейной кривой и набор ***k***-мерных коэффициентов линейной кривой, которые описывают ***k***-мерную линейную кривую, наилучшим образом аппроксимирующую набор входных данных с помощью метода наименьших квадратов. Под линейной кривой понимается кривая, образованная суммой функций с линейными коэффициентами.

Вход **стандартное отклонение** (Standard Deviation) представляет массив стандартных отклонений для входных данных . Если стандартные отклонения одинаковы или неизвестны, то этот вход не подключается и используется значение по умолчанию 1,0.

Вход **H** служит для передачи матрицы, 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 рядов 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) представляет 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.

В разделе **Помощь** (Help) LabVIEW применение ВП **Общая линейная аппроксимация методом наименьших квадратов** (General LS Linear Fit) поясняется с помощью ВП **Общее сглаживание методом наименьших квадратов** (General LS Fitting), размещенного в разделе Help ⇒ Find Examples ⇒ Analyzing and Processing Signals ⇒ Curve Fitting and Interpolation.

Блок-диаграмма этого ВП приведена на рис. 3.44, а его лицевая панель – на рис. 3.45.

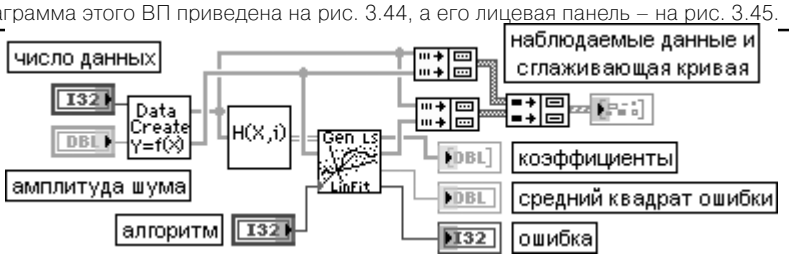


Рис. 3.44. Блок-диаграмма ВП
Общее сглаживание методом наименьших квадратов *(General LS Fitting)*

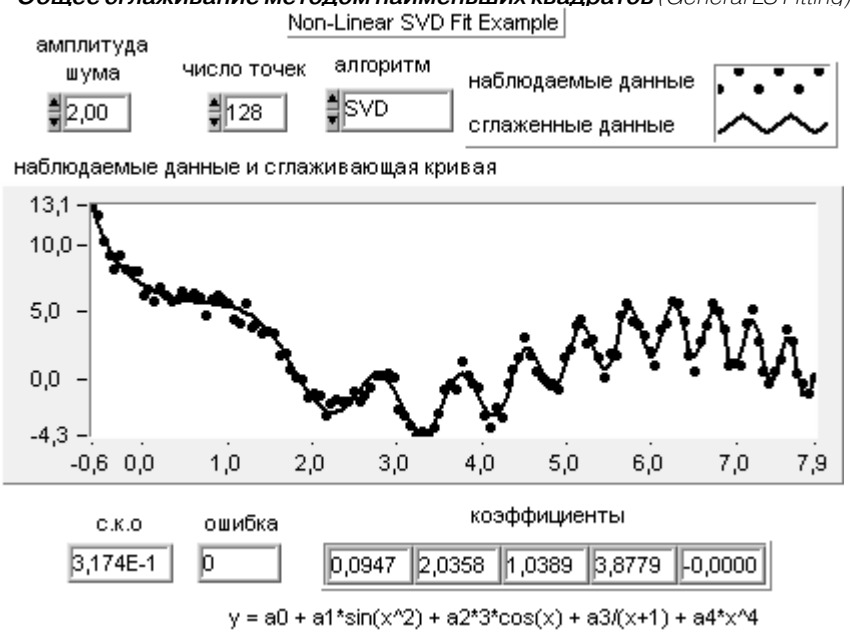


Рис. 3.45. Лицевая панель ВП
Общее сглаживание методом наименьших квадратов *(General LS Fitting)*

Целью ВП является нахождение с помощью метода наименьших квадратов коэффициентов a, которые позволяют наилучшим образом представить набор исходных данных (x, y). Зависимость между x и y задается в виде

$$y = f(a,x) = \sum_{i=0}^{m-1} a_i f_i(x) = a_0 f_0(x) + a_1 f_1(x) + ... + a_{m-1} f_{m-1}(x),$$

где $a = \{a_0, a_1, a_2, ... a_{m-1}\}$, m – общее число функций.
Пусть данные формируются с помощью выражения

$$y = 2h_0(x) + 3h_1(x) + 4h_2(x) + noise,$$

где $h_0(x)=\sin(x^2)$, $h_1(x)=\cos(x)$, $h_2(x)=\dfrac{1}{x+1}$, $noise$ – набор случайных чисел.

Блок-диаграмма ВП, выполняющего расчет данных, приведена на рис. 3.46 (подприбор **Генери-**

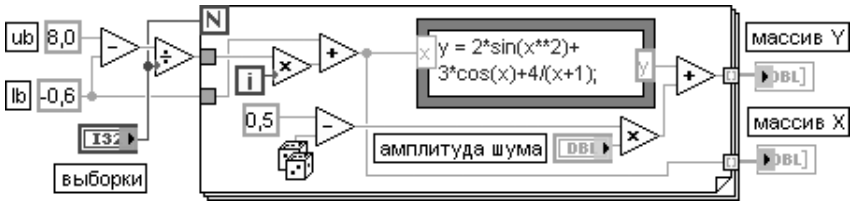


*Рис. 3.46. Блок-диаграмма ВП **Генерировать данные** (Generate Data)*

ровать данные (Generate Data) на блок-диаграмме **Общее сглаживание методом наименьших квадратов** (General LS Fitting)).

Пусть в то же время функция, используемая для аппроксимации данных, имеет вид

$$y=a_0f_0(x)+a_1f_1(x)+a_2f_2(x)+a_3f_3(x)+a_4f_4(x),$$

где $f_0(x)=1{,}0$, $f_1(x)=\sin(x^2)$, $f_2(x)=3\cos(x)$, $f_3(x)=\dfrac{1}{x+1}$, $f_4(x)=x^4$.

Для получения коэффициентов a необходимо на вход ВП **Общее сглаживание методом наименьших квадратов** (General LS Fitting) подать совокупность данных **X** и **Y**, а также базовую функцию **H(X,i)**, являющуюся двумерным массивом. Блок-диаграмма подпрограммы LinBasFns.H(x,i), выполняющей формирование матрицы **H(X,i)**, приведена на рис. 3.47.

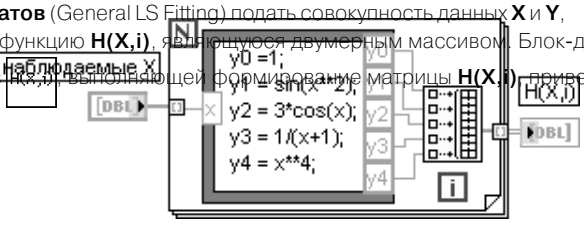
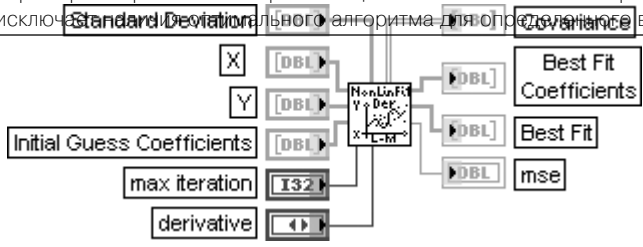


Рис. 3.47. Блок-диаграмма подпрограммы LinBasFns

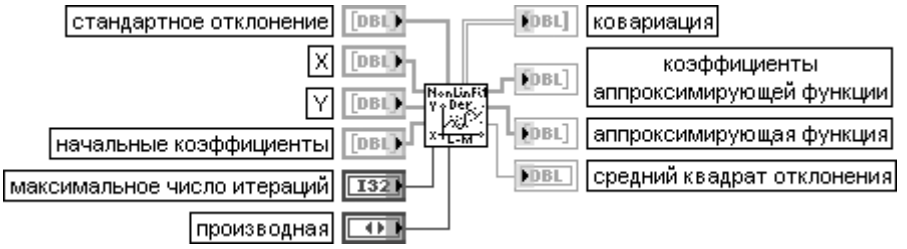
Из показаний индикаторов на лицевой панели ВП **Общее сглаживание методом наименьших квадратов** (General LS Fitting) можно сделать вывод, что в результате выполнения аппроксимации вместо истинного набора коэффициентов $a=\{0{,}0\ \ 2{,}0\ \ 1{,}0\ \ 4{,}0\ \ 0{,}0\}$ возвращается их достаточно близкая оценка.

В приведенном примере погрешность аппроксимации слабо зависит от выбора алгоритма. Однако это не исключает возможности использования оптимального алгоритма для определенного вида данных

Nonlinear Lev-Mar Fit



Нелинейная аппроксимация Левенберга-Марквардта



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.

Вход **стандартное отклонение** (Standard Deviation) представляет массив стандартных отклонений $\sigma[i]$ для входных данных $(x[i], y[i])$. Если стандартные отклонения одинаковы или неизвестны, то этот вход не подключается и используется значение по умолчанию 1,0.

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 **максимальное число итераций** (max iteration) определяет максимальное допустимое число итераций процедуры поиска коэффициентов. 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**.

Вход **производная** (derivative) определяет метод, используемый для расчета Якобиана. При этом возможен выбор двух вариантов: **численный расчет** (Numerical calculation) (0) и **расчет по формуле** (Formula calculation) (1). При выборе **расчета по формуле** пользователь должен определить вид нелинейной модельной функции $f = f(X, A)$ и ее частных производных в структуре узел **Формула** (Formula Node) на блок-диаграмме ВП Target Fnc & Deriv NonLin  , являющегося подприбором (subVI) ВП **Нелинейная аппроксимация Левенберга-Марквардта** (Nonlinear Lev-Mar Fit). Для доступа к подприбору Target Fnc & Deriv NonLin необходимо выбрать в меню Browse ⇒ This VI's SubVIs ⇒ Target Fnc & Deriv NonLin.vi.

Расчет по формуле является более эффективным методом, поскольку не требует численной аппроксимации Якобиана.

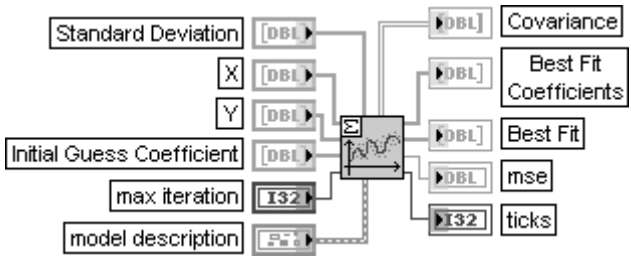
Выход **ковариация** (Covariance) отображает матрицу коэффициентов ковариации **C**. c_{jk} является коэффициентом ковариации между a_j и a_k. c_{jj} является дисперсией a_j.

Выход **коэффициенты аппроксимирующей функции** (Best Fit Coefficients) представляет набор коэффициентов, позволяющих минимизировать сумму квадратов отклонений χ^2. Значение χ^2 определяется следующим образом:

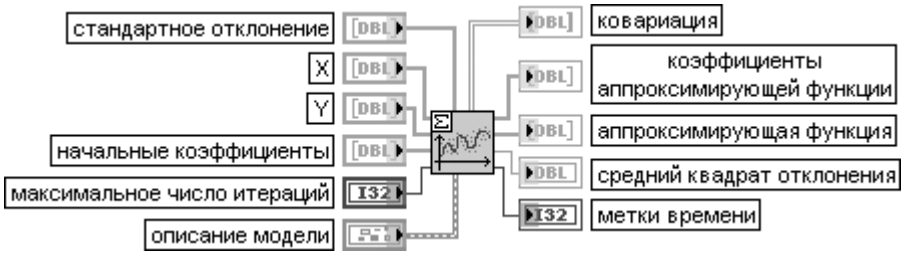
$$\chi^2 = \sum_{i=0}^{n-1} \left(\frac{y_i - f(x_i, a_1 \ldots a_m)}{\sigma_i} \right)^2.$$

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)

Levenberg Marquardt



Левенберг-Марквардт



ВП отличается от описанного выше ВП **Нелинейная аппроксимация Левенберга-Марквардта** (Nonlinear Lev-Mar Fit) наличием входа **описание модели** (model description) и выхода метки времени (ticks).

Вход **описание модели** представляет кластер, содержащий аппроксимирующее выражение. В состав кластера входят следующие элементы:

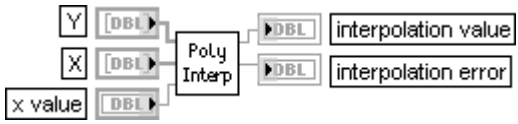
модель (model) – является строковым описанием модельного выражения;

параметры (Parameters) **–** являются массивом строк, описывающим неизвестные параметры;

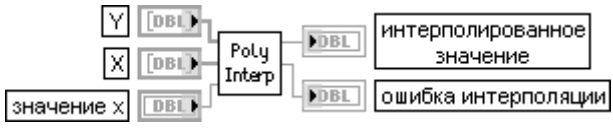
x – представляет строку, описывающую независимую переменную.

Выход **метки времени** (ticks) отображает время (в миллисекундах), затраченное на выполнение ВП

Polynomial Interpolation

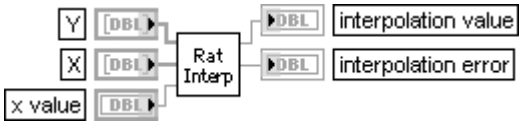


Полиномиальная интерполяция

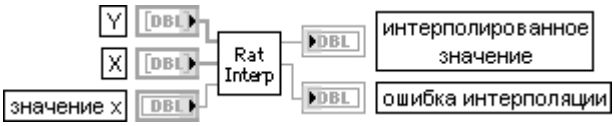


ВП интерполирует или экстраполирует функцию ***f***, заданную набором ***n*** точек ***(x[i]y[i])***, в точке ***x***, заданной **значением x** (x value).
Y и **X** представляют массивы значений зависимой и независимой переменных. Если число элементов **Y** и **X** отличается, то ВП устанавливает на 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

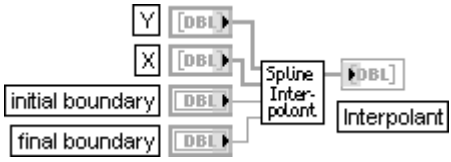


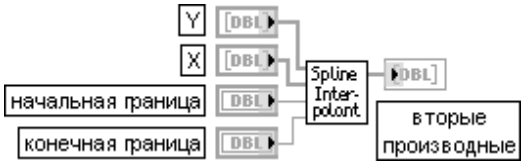
Интерполяция рациональной функцией



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 $\frac{P(x_i)}{Q(x_i)} = \frac{p_0 + p_1 x_i + ... + p_m x_i^m}{q_0 + q_1 x_i + ... + q_m x_i^m}$, 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 $p_i(x)$, 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:

$$\frac{x_i - x_{i-1}}{6} g''(x_{i-1}) + \frac{x_{i+1} - x_{i-1}}{3} g''(x_i) + \frac{x_{i+1} - x_i}{6} g''(x_{i+1}) = \frac{y_{i+1} - y_i}{x_{i+1} - x_i} - \frac{y_i - y_{i-1}}{x_i - x_{i-1}},$$

$i = \overline{0, n-2}$.

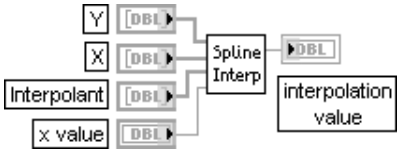
Таким образом, имеет место система из $n-2$ линейных уравнений с n неизвестными $g''(x_i)$, $i = \overline{0, n-1}$.
ВП рассчитывает недостающие значения $g''(x_0)$, $g''(x_{n-1})$, используя **начальную границу** и **конечную границу** по формуле

$$g'(x) = \frac{y_{i+1} - y_i}{x_{i+1} - x_i} + \frac{3A^2 - 1}{6}(x_{i+1} - x_i) g''(x_i) + \frac{3B^2 - 1}{6}(x_{i+1} - x_i) g''(x_{i+1}),$$

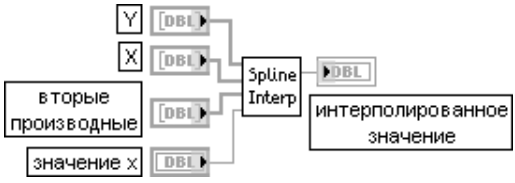
где $A = \frac{x_{i+1} - x}{x_{i+1} - x_i}$, $B = 1 - A = \frac{x - x_i}{x_{i+1} - x_i}$.

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 $A = \frac{x_{i+1} - x}{x_{i+1} - x_i}$, $B = 1 - A$,

$$C = \frac{1}{6}(A^3 - A)(x_{i+1} - x_i)^2, \qquad D = \frac{1}{6}(B^3 - B)(x_{i+1} - x_i)^2$$

В состав палитры функций сглаживания данных входит Экспресс-ВП **Сглаживание кривой** (Curve Fitting), рассмотренный ниже.

Сглаживание кривой (Curve Fitting)

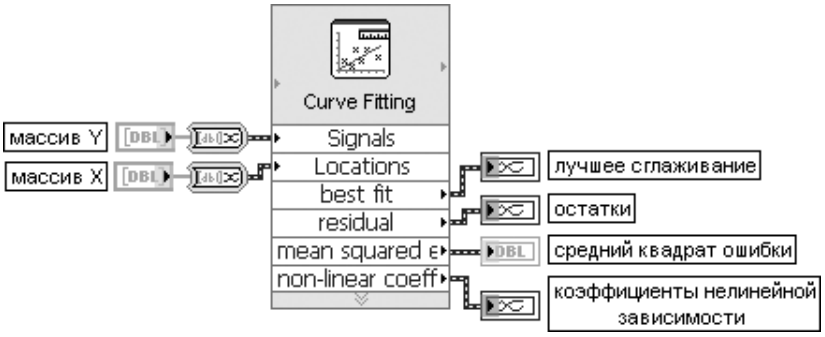


Рис. 3.48. Блок-диаграмма возможного подключения Экспресс-ВП

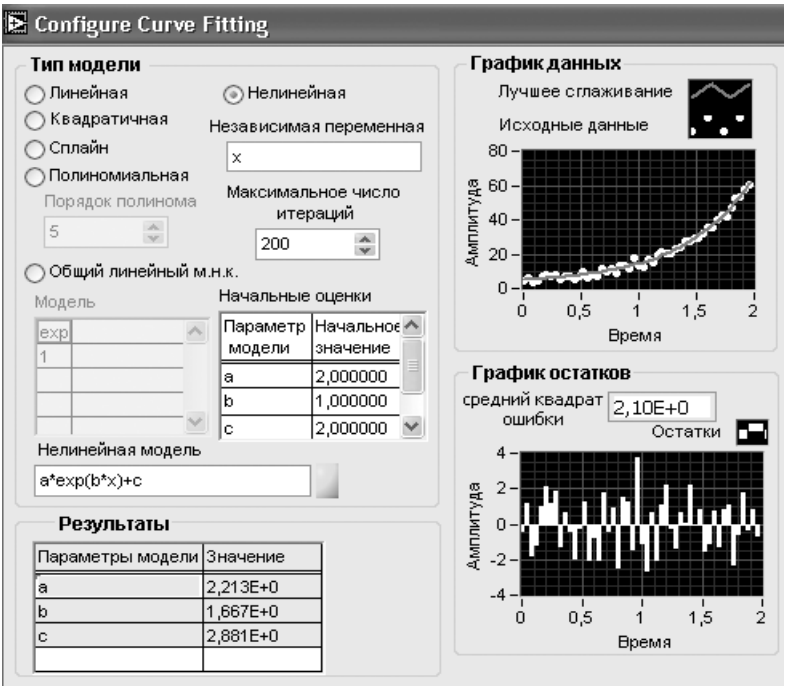


Рис. 3.49. Вид диалогового окна конфигурирования Экспресс-ВП ***Сглаживание кривой*** *(Curve Fitting)*

Данный Экспресс-ВП использует функциональность следующих ВП и функций: **Линейная аппроксимация** (Linear Fit), **Нелинейная аппроксимация Левенберга-Марквардта** (Nonlinear Lev-Mar Fit), **Полиномиальная интерполяция** (Polynomial Interpolation), **Сплайн-интерполяция** (Spline Interpolation)

3.2.4. Функции линейной алгебры и обработки массивов

Функции линейной алгебры (рис. 3.50) 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. **Дополнительные функции линейной алгебры** (Advanced Linear Algebra) (рис. 3.50б) позволяют выполнять различного рода разложения матриц, находить такие параметры матриц, как след, ранг, норму и число обусловленности. **Комплексные функции линейной алгебры** (Complex Linear Algebra) (рис. 3.50в) в справочнике не рассматриваются вследствие их идентичности функциям обработки действительных матриц.

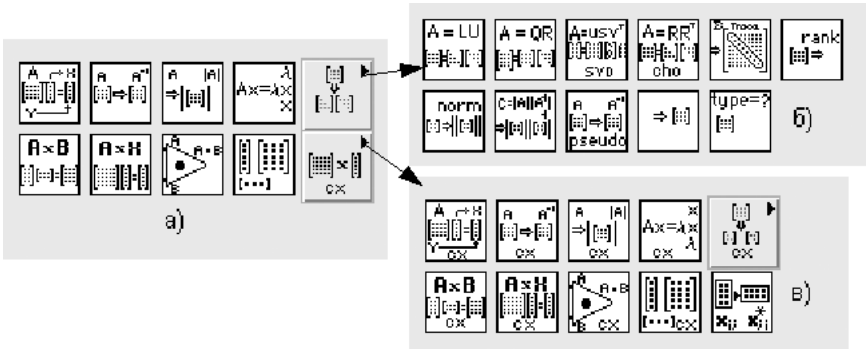
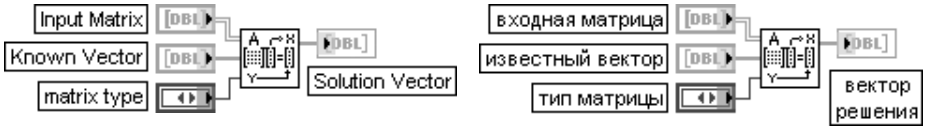


Рис. 3.50. Вид основной палитры (а) и дополнительных подпалитр (б и в) функций линейной алгебры

Solve Linear Equations — **Решение линейных уравнений**



ВП 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.

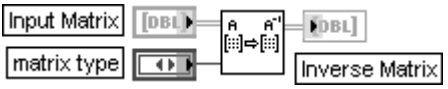
0	1	2	3
Общего вида (по умолчанию) (General (default))	**Положительно определенная** (Positive definite)	**Нижняя треугольная** (Lower triangular)	**Верхняя треугольная** (Upper triangular)

Выход **вектор решения** (Solution Vector) представляет вектор решения ***X*** системы ***AX = Y.***
Пусть ***A*** будет входной матрицей размером ***m*** × ***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 недоопределенной. 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

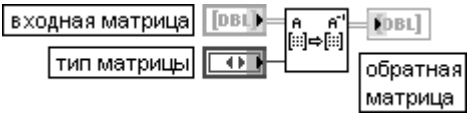
$$AX = LZ = Y \text{ и } Z = UX$$

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*** ≠ ***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. ВП **Решение линейных уравнений** (Solve Linear Equations) обнаруживает вырожденные матрицы и возвращает ошибку, в связи с чем пользователю нет необходимости проверять правильность системы перед использованием данного ВП.
Численная реализация обратимости матриц вследствие рекурсивной природы очень чувствительна к ошибкам округления, вносимых сопроцессором операций с плавающей запятой. Хотя вычисления используют максимально возможную точность, ВП не гарантирует решения систем

Inverse Matrix

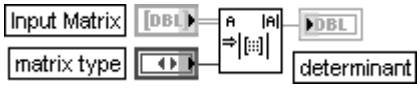


Обратная матрица



ВП 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 рассмотренном выше ВП **Решение 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 линейной системы, заданной уравнением ***AB = I***, где ***A*** – входная матрица, ***B*** – обратная матрица, ***I*** – единичная матрица

Determinant **Определитель**

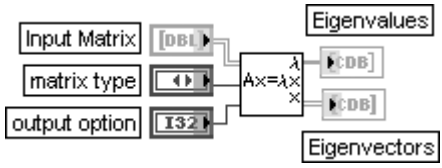


ВП рассчитывает **определитель** (determinant) действительной квадратной **входной матрицы** (Input Matrix)**.**
Вход **тип матрицы** (matrix type) имеет те же значения, что и в рассмотренном выше ВП **Решение линейных уравнений** (Solve Linear Equations).
Пусть ***A*** является квадратной матрицей, 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***

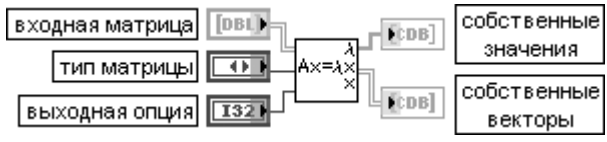
$$|A| = \prod_{i=0}^{n-1} u_{ii},$$

где $|A|$ – определитель матрицы ***A***, ***n*** – размер матрицы ***A***

EigenValues and Vectors



Собственные значения и векторы



ВП 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 симметрическая (**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 вычисляемых ВП компонентов. 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 несимметрическая.

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 несимметрична.

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. Симметричная действительная матрица всегда имеет действительные собственные значения и собственные векторы

A × B **Матричное умножение A × B**

ВП выполняет матричное умножение двух входных матриц.

A является первой входной матрицей. Число колонок матрицы **A** должно быть равно числу строк матрицы **B** и больше нуля: k > 0. При нарушении этих условий ВП выводит на выход **A × B** пустой массив и возвращает ошибку. **B** является второй матрицей.

Выход **A × B** представляет матрицу, содержащую результат матричного умножения **A × 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 уравнение:

$$c_{ij} = \sum_{l=0}^{k-1} a_{il} b_{lj} \quad \text{для} \quad \begin{cases} i = \overline{0, n-1} \\ j = \overline{0, m-1} \end{cases};$$

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***

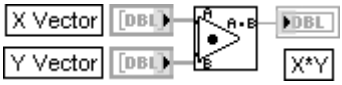
A x Vector **Умножение матрицы на вектор**

ВП выполняет умножение входной матрицы и входного вектора.
Если ***A*** является матрицей ***n*** × ***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**

Dot Product **Точечное произведение**

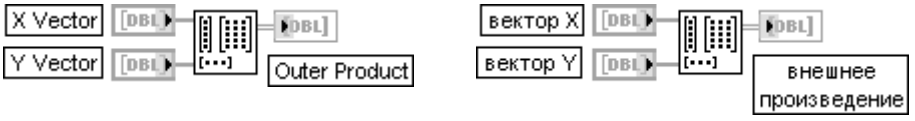


ВП рассчитывает точечное произведение вектора **X** (**X Vector)** и вектора **Y** (**Y Vector),** используя следующее выражение

$$XY = \sum_{i=0}^{n-1} x_i y_i,$$

где ***n*** – число данных

Outer Product **Внешнее произведение**



ВП рассчитывает внешнее произведение **вектора X** (X Vector) и **вектора Y** (Y Vector), используя следующее выражение:

$$a_{ij} = x_i y_j, \qquad \text{где} \quad \begin{cases} i = \overline{0, n-1} \\ j = \overline{0, m-1}, \end{cases}$$

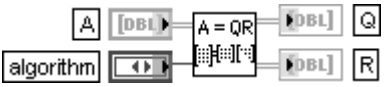
где ***A*** представляет двумерную выходную последовательность **внешнее произведение** (Outer Product), ***n*** – число элементов входной последовательности **X** и ***m*** – число элементов входной последовательности **Y**

В следующей таблице рассмотрены **Дополнительные функции линейной алгебры (**Advanced Linear Algebra).

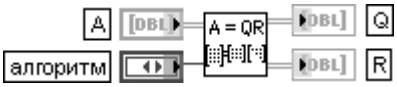
LU Factorization **LU разложение**

ВП выполняет **LU** разложение действительной квадратной матрицы **A**.
Выход **L** представляет нижнюю треугольную матрицу.
Выход **U** представляет верхнюю треугольную матрицу.
Выход **P** отображает матрицу перестановок.
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 Factorization



QR разложение

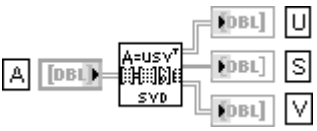


ВП выполняет **QR** разложение действительной матрицы **A**.
A является действительной матрицей размером ***m*** × ***n***, где ***m*** – число строк и ***n*** – число столбцов матрицы **A**. Матрица **A** может быть прямоугольной или квадратной.
Вход **алгоритм** (algorithm) определяет вид алгоритма, применяемого при выполнении **QR** разложения (таблица).

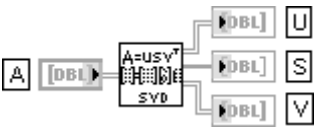
0	1	2
Хаусхолдер (Householder) (по умолчанию)	**Гивенс** (Givens)	**Быстрый Гивенс** (Fast givens)

Q является ортогональной матрицей размером ***m*** × ***m***.
R является верхней треугольной матрицей размером ***m*** × ***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 Factorization



SVD разложение



ВП выполняет декомпозицию по сингулярным значениям (**SVD**) заданной ***m*** × ***n*** действительной матрицы ***A*** с ***m*** > ***n***, где ***m*** определяет число строк, а ***n*** – число столбцов матрицы ***A***. Если ***m*** < ***n***, то перед использованием данного ВП такую матрицу необходимо транспонировать.
Выход ***U*** отображает матрицу размером ***m*** × ***n***, содержащую ***n*** ортогональных столбцов.
Выход ***S*** представляет массив, который содержит ***n*** сингулярных значений матрицы ***A*** в порядке уменьшения.
Выход ***V*** представляет ортогональную матрицу размером ***n*** × ***n***.

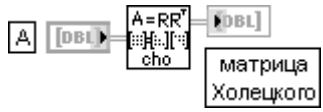
SVD разложение создает три матрицы, для которых выполняется следующее условие:

$A = US_0V^T,$

где ***U*** и $\boldsymbol{V}^{\boldsymbol{T}}$ являются ортогональными матрицами, а $\boldsymbol{S_0}$ является диагональной матрицей размером $\boldsymbol{n} \times \boldsymbol{n}$ с элементами матрицы ***S*** на диагонали в порядке уменьшения

Cholesky Factorization

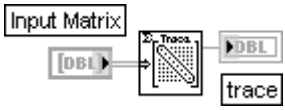
Разложение Холецкого



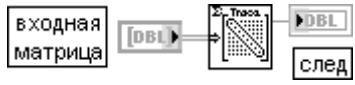
ВП выполняет разложение Холецкого для действительной положительно определенной матрицы ***A***. Выход матрицы **Холецкого** (Cholesky) содержит полученную в результате разложения верхнюю треугольную матрицу ***R***.

Если действительная квадратная матрица ***A*** является положительно определенной, то она может быть разделена на множители в соответствии с выражением $\boldsymbol{A = R^TR}$, где ***R*** является верхней треугольной матрицей, а $\boldsymbol{R^T}$ – результатом транспонирования матрицы ***R***

Trace



След

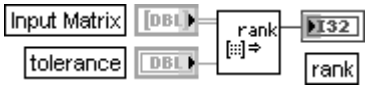


ВП находит **след** (trace) **входной матрицы** (Input Matrix).

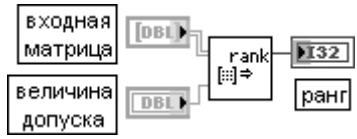
Входная матрица должна быть квадратной, и ее размер должен быть больше 0. Если входная матрица пустая или неквадратная, то ВП устанавливает на выходе **след** значение NaN и возвращает ошибку.

След (trace) представляет сумму элементов на главной диагонали входной матрицы

Matrix Rank



Ранг матрицы



ВП находит **ранг** (rank) действительной прямоугольной **входной матрицы** (Input Matrix)**.**

Вход **величина допуска** (tolerance) определяет такой уровень, при котором число сингулярных значений, превысивших этот уровень, 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

$$\textit{величина допуска} = max(m,n) \cdot ||A|| \cdot \varepsilon,$$

где ***A*** представляет входную матрицу, ***m*** и ***n*** представляют, соответственно, число строк и столбцов матрицы **A**, $||\boldsymbol{A}||$ 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 строк или колонок входной матрицы

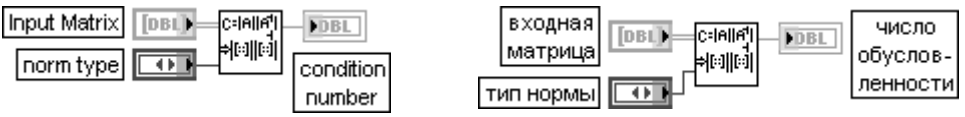
Matrix Norm **Норма матрицы**

ВП 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 элементов столбца входной матрицы ***A***
2	***F***-норма – $\|\|A\|\|_f$ равна $\sqrt{\sum diag(A^T A)}$, где $diag(A^T A)$ является наибольшей суммой колонки входной матрицы ***A***
3	***inf***-норма – $\|\|A\|\|_\infty$ равна наибольшей сумме элементов строки входной матрицы ***A***

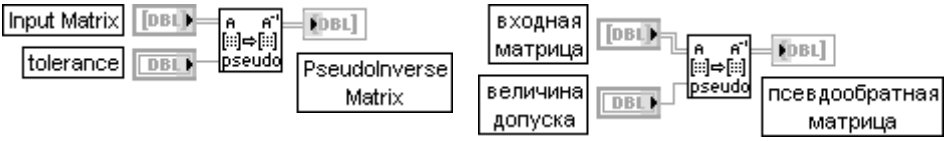
Выход **норма** представляет скаляр, который дает некоторую меру величины элементов входной матрицы, зависящую от выбранного типа нормы

Matrix Condition Number **Число обусловленности матрицы**



ВП 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 ВП **Норма матрицы** (Matrix Norm).
Число обусловленности рассчитывается с помощью выражения $c = ||A||_p||A^{-1}||_p$, 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.

PseudoInverse Matrix **Псевдообратная матрица**



ВП 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 ВП **Ранг** (Rank).
ВП рассчитывает **псевдообратную матрицу A^+**, используя алгоритм **SVD** и любое сингулярное значение, меньшее **величины допуска** (tolerance).
Матрица **A^+** размером ***m*** × ***n*** называется псевдообратной матрице A, если A^+ удовлетворяет следующим четырем условиям Мура-Пенроуза:

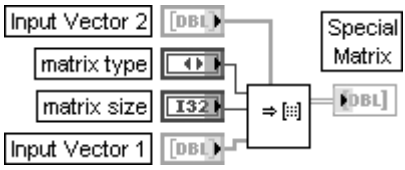
1) $A A^+ A = A$;

2) $A^+ A A^+ = A^+$;

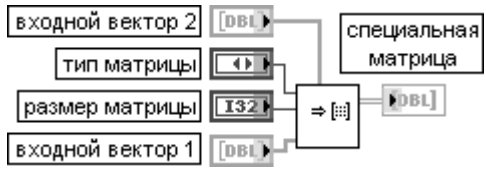
3) $A A^+$ является симметрической матрицей;

4) $A^+ A$ является симметрической матрицей

Create Special Matrix



Создать специальную матрицу



ВП 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 – входной вектор 1** (Input Vector1), ***nx*** – размер **X**, **Y** – **входной вектор 2** (Input Vector2), ***ny*** – размер **Y** и **B** – выход **специальная матрица**.

0	**Единичная** (Identity) – порождает единичную матрицу размером ***n*** × ***n***
1	**Диагональная** (Diagonal) – порождает диагональную матрицу размером ***nx*** × ***nx***, диагональные элементы которой являются элементами **X**
2	**Матрица Теплица** (Toepliz) – порождает матрицу Теплица размером ***nx*** × ***nx***, у которой ***X*** является первым столбцом, а ***Y*** – первой строкой. При различии первых элементов ***X*** и ***Y*** в матрице используется первый элемент ***X***
3	**Вандермонда** (Vandermonde) – порождает матрицу Вандермонда размером ***nx*** × ***nx***, у которой столбцы являются степенями ***X***. Элементы матрицы Вандермонда рассчитываются следующим образом: $b_{ij} = x_i^{nx-j-1},$ где $i, j = 0, nx-1$
4	**Парная** (Companion) – порождает парную матрицу размером ***nx*** **– 1** × ***nx*** **– 1**. Если вектор ***X*** является вектором полиномиальных коэффициентов, то коэффициент при самой высокой степени является первым элементом ***X***, а свободный член – последним элементом.

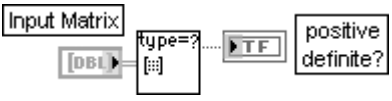
При этом соответствующая парная матрица формируется следующим образом: первая строка

$$b_{0,j-1} = -\frac{x_j}{x_0} \quad j = \overline{1, nx-1}.$$

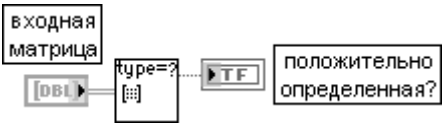
Оставшаяся часть ***B***, начиная со второй строки, является единичной матрицей. Собственные значения парной матрицы содержат корни соответствующего полинома

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

Test Positive Definite



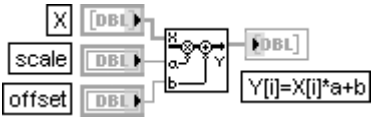
Тест положительной определенности



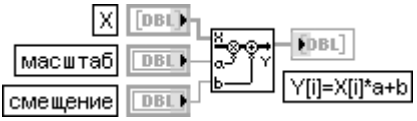
ВП проверяет **входную матрицу** (Input Matrix) на положительную определенность. Входная матрица должна быть квадратной действительной матрицей.
Выход **положительно определенная?** (positive definite?) содержит результат тестирования. Если входная матрица является положительно определенной, то на выход **положительно определенная?** передается значение ИСТИНА, в противном случае передается значение ЛОЖЬ

Функции обработки массивов позволяют выполнять линейное и полиномиальное преобразование одномерных и двумерных массивов, масштабировать и нормализовать одномерные и двумерные массивы, преобразовывать массивы декартовых координат в массивы полярных координат и выполнять обратное преобразование.

1D Linear Evaluation



Линейное преобразование одномерного массива

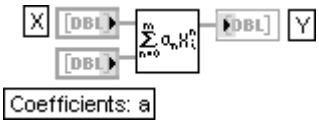


ВП выполняет линейное преобразование входного одномерного массива **X** в выходной массив **Y** в соответствии с выражением

$$Y[i] = X[i]a + b,$$

где a – мультипликативный коэффициент (масштаб), b – аддитивный коэффициент (смещение)

1D Polynomial Evaluation **Полиномиальное преобразование одномерного массива**

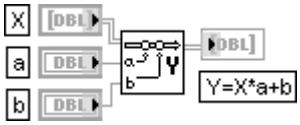


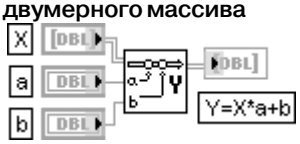
ВП выполняет полиномиальное преобразование входного одномерного массива **X** в выходной массив **Y** с использованием **коэффициентов** (Coefficients: a):

$$Y = \sum_{n=0}^{m} a_n X^n,$$

где m – порядок полинома, a – массив коэффициентов

2D Linear Evaluation **Линейное преобразование двумерного массива**



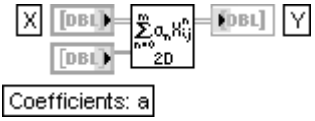


ВП выполняет линейное преобразование входного двумерного массива **X** в выходной массив **Y** в соответствии с выражением

$$Y[i, j] = X[i, j]a + b,$$

где a – мультипликативный коэффициент (масштаб), b – аддитивный коэффициент (смещение)

2D Polynomial Evaluation **Полиномиальное преобразование двумерного массива**



ВП выполняет полиномиальное преобразование входного двумерного массива ***X*** в выходной массив ***Y*** с использованием **коэффициентов** (Coefficients: a).

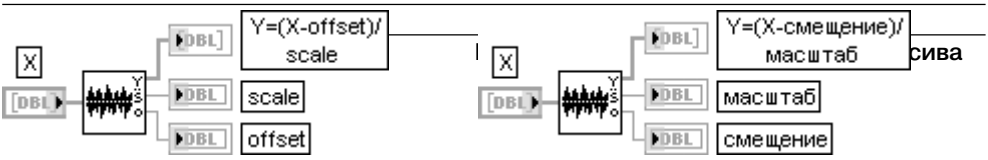
$$Y = \sum_{n=0}^{m} a_n X^n,$$

где m – порядок полинома, a – массив коэффициентов.
При этом каждый элемент выходной последовательности ***Y*** рассчитывается следующим образом:

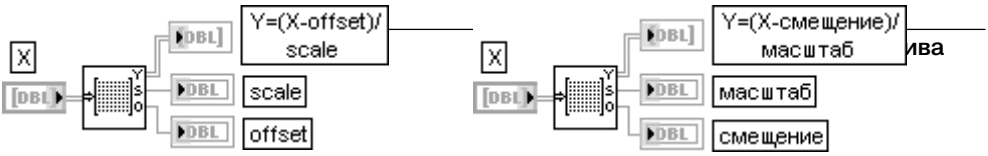
$$Y[i,j]=\sum_{n=0}^{m} a[n]X^{n}[i,j]$$

ВП определяет максимальное абсолютное значение входного одномерного массива ***X*** и использует его затем для масштабирования данного массива

ВП определяет максимальное абсолютное значение входного двумерного массива ***X*** и использует его затем для масштабирования данного массива

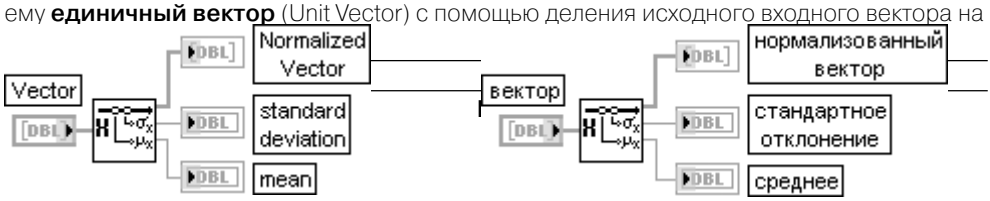


ВП определяет **масштаб** (scale) и **смещение** (offset) входного одномерного массива ***X*** и использует эти параметры для масштабирования данного массива



ВП определяет **масштаб** (scale) и **смещение** (offset) входного двумерного массива ***X*** и использует эти параметры для масштабирования данного массива

ВП находит **норму** (norm) **входного вектора** (Input Vector) и получает соответствующий ему **единичный вектор** (Unit Vector) с помощью деления исходного входного вектора на

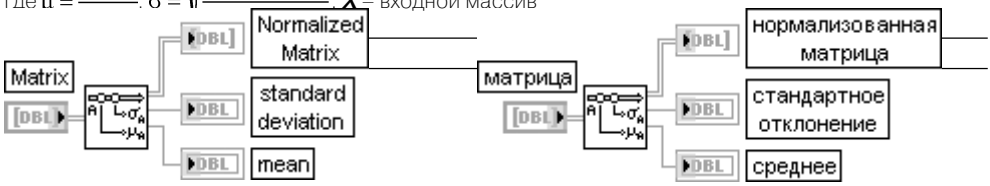


ВП нормализует входной **вектор** (Vector), используя статистические параметры (**μ,*s***), где **μ** представляет **среднее** (mean), ***s* – стандартное отклонение** (standard deviation) вектора с целью получения **нормализованного вектора** (Normalized Vector), у которого эти параметры равны (0,1).

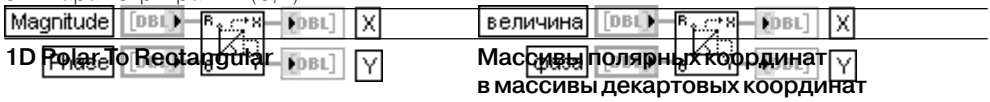
ВП рассчитывает нормализованный вектор ***Y***, используя следующие выражения:

$$Y = \frac{X - \mu}{\sigma},$$

где $\mu = \frac{\sum_{i=0}^{n-1} x_i}{}$. $\sigma = \sqrt{\frac{\sum_{i=0}^{n-1} (x_i - \mu)^2}{}}$. ***X*** – входной массив

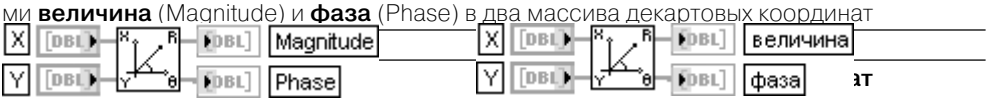


ВП нормализует входную **матрицу** (Matrix), используя статистические параметры (**μ,*s***), где **μ** представляет **среднее** (mean), ***s*** – **стандартное отклонение** (standard deviation) матрицы с целью получения **нормализованной матрицы** (Normalized Matrix), у которой эти параметры равны (0,1)



1D Polar To Rectangular **Массивы полярных координат в массивы декартовых координат**

ВП преобразует два массива полярных координат, представленных входными массивами **величина** (Magnitude) и **фаза** (Phase) в два массива декартовых координат



в массивы полярных координат

3.2.5. Функции оптимизации и поиска нулей

ВП преобразует два массива декартовых координат в два массива полярных координат

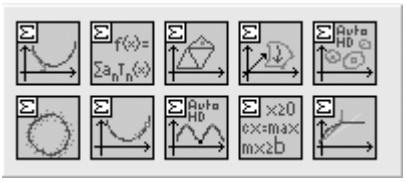
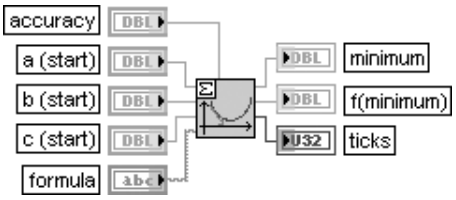


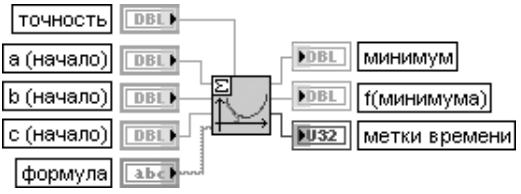
Рис. 3.51. Вид палитры функций оптимизации

Функции оптимизации (рис. 3.51) 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.

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 **формулой** (formula).

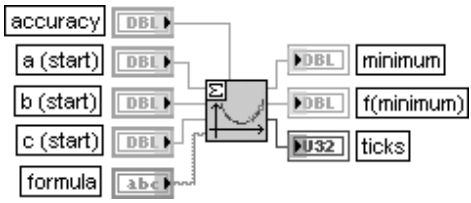
Вход **точность** (accuracy) устанавливает точность определения минимума выражения, заданного на входе **формула**. Метод завершает работу, если результаты двух последовательных итераций отличаются на величину меньшую, чем **точность**.

Входы **a**, **b** и **c** представляют соответственно левую, среднюю и правую точки отрезка локализации минимума. По умолчанию значения этих входов равны 0,0.

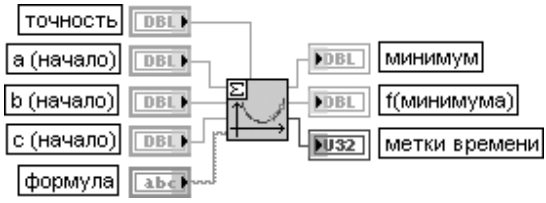
Вход **формула** (formula) представляет строку, описывающую исследуемую функцию.

Выход **минимум** (minimum) представляет найденный локальный минимум функции, заданной ф**ормулой**.
Выход **f(минимума)** (f(minimum)) представляет значение функции в определенном локальном минимуме.
Выход **метки времени** (ticks) определяет время (в миллисекундах) выполнения расчета

Golden Section 1D



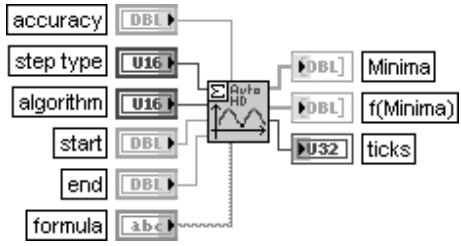
Метод золотого сечения



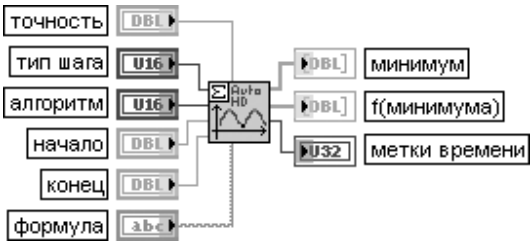
ВП определяет локальный минимум заданной одномерной функции с помощью метода золотого сечения.
Все входы и выходы этого ВП имеют то же назначение, что и одноименные входы и выходы рассмотренного выше ВП **Поиск минимума одномерной функции с помощью производных** (Brent with Derivatives 1D).
Ограничивающий 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 в большинстве случаев ведет к быстрому нахождению минимума. Новая граничная точка рассчитывается с помощью следующего выражения:

$$\left|\frac{x-b}{c-a}\right| = \sqrt{5} - 2$$

Find All Minima 1D

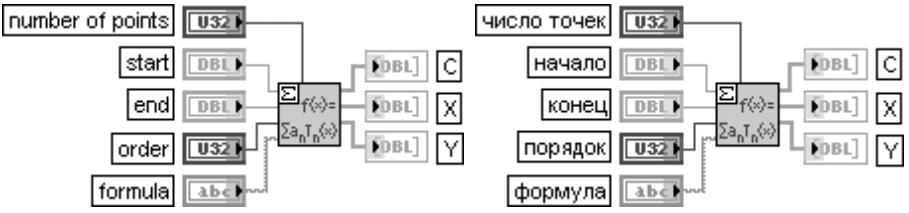


Найти все минимумы одномерной функции



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 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:

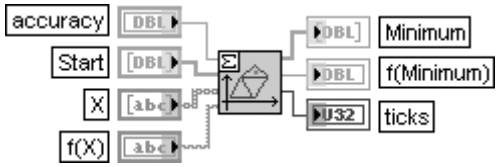
$$f(x) = c_0 T_0 + ... + c_m T_m.$$

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 могут быть рассчитаны с помощью следующего выражения:

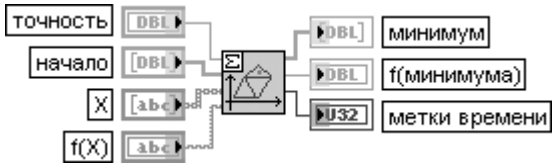
$$c_j = \sum_{k=1}^{n} f(x_k) T_j(x_k),$$

где $x_k = \cos(\frac{\pi k}{n}),\ k = \overline{1,n}$

Downhill Simplex nD



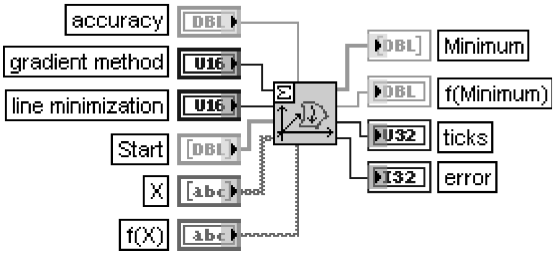
Симплекс-метод скорейшего спуска поиска минимума функции n переменных



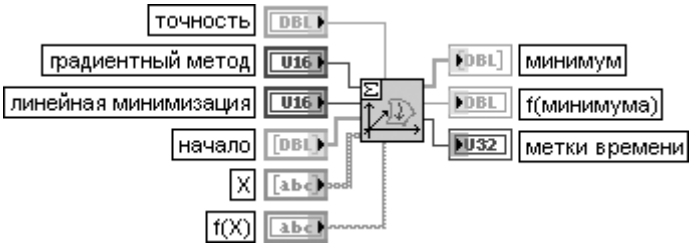
ВП ищет локальный минимум функции **n** независимых переменных с помощью симплекс-метода скорейшего спуска.
Вход **точность** (accuracy) задает точность определения минимума. Метод останавливается, если результаты двух последовательных итераций отличаются на величину меньшую, чем **точность**. По умолчанию значение равно 1,00E – 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 и 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 очень маленького симплекса

Conjugate Gradient nD

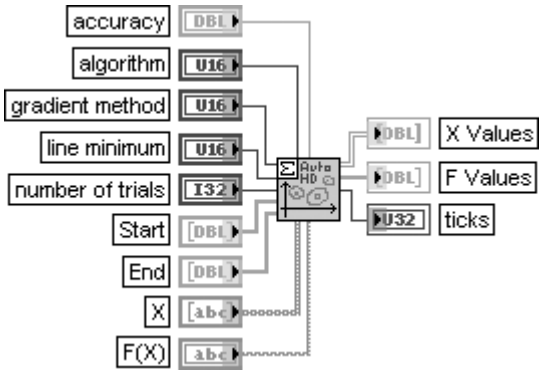


Метод сопряженных градиентов поиска минимума функции n переменных

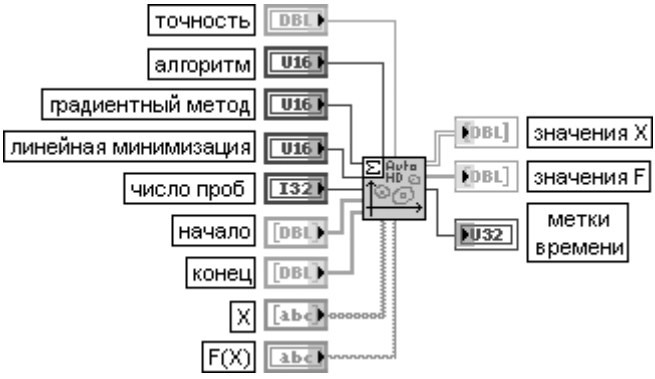


ВП определяет минимум функции **n** независимых переменных с помощью метода сопряженных градиентов.
Входы **точность**, **начало**, **X**, **f(X)** и выходы **минимум**, **f(минимума)**, **метки времени** имеют то же назначение, что и одноименные входы и выходы рассмотренного выше ВП **Симплекс-метод 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** (without derivatives) (значение 0) и **с использованием производных** (with derivatives) (значение 1). По умолчанию значение входа равно 0

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**, **минимизация линии**, **начало**, **X**, **f(X)** и выходов **минимум**, **f(минимума)**, **метки времени** идентично назначению одноименные входов и выходов, 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).

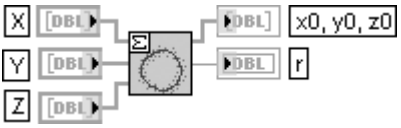
Вход **число проб** (number of trials) задает число случайно выбранных начальных точек процесса оптимизации. Эти точки принадлежат интервалу (**начало**, **конец**). По умолчанию число равно 5.

Входы **начало** (Start) и **конец** (End) определяют координаты начальной и конечной точек в **n**-мерном пространстве.

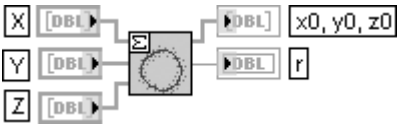
Выход **значения X** (X Values) представляет матрицу, описывающую все найденные локальные минимумы.

Выход **значения F** (F Values) представляет значения функции в точках минимумов **значения X**

Fitting on a Sphere



Аппроксимация сферой



ВП определяет наилучшую сферическую аппроксимацию области точек в трехмерном пространстве.

Входы ***X***, ***Y***, и ***Z*** задают соответственно ***x***, ***y*** и ***z*** координаты точек в области.

Выход x_0, y_0, z_0 отображает рассчитанные координаты центра области.

Выход ***r*** представляет рассчитанное значение радиуса области.

В основе алгоритма аппроксимации лежит минимизация суммы

$$\sum_{i=1}^{n}((x_i - x_0)^2 + (y_i - y_0)^2 + (z_i - z_0)^2 - r^2)^2$$

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:

$$cx = max$$

с ограничениями $x \geq \mathbf{0}$ и $mx \geq b$.

Для проблемы оптимизации $cx = max$ необходимо использовать следующие определения:

$$X = (x_1, ..., x_n); \quad C = (c_1, ..., c_n); \quad B = (b_1, ..., b_k); \quad M - матрица\ k \times n.$$

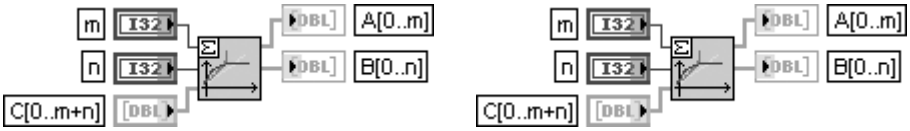
Для решения проблемы оптимизации необходимо решить, существует ли оптимальный вектор ***X***. Если он существует, его можно определить.

Решение проблемы линейного программирования представляет двухэтапную задачу.

1. Исходную проблему необходимо преобразовать в проблему с ограничением в нормальной форме, по существу без неравенств в формулировках.
2. Решить проблему с ограничением в нормальной форме.

Формулировка с ограничением в нормальной форме представляется специфичной. Однако существует ряд способов изменения условий. Так, в частности, условие $dx \leq e$ эквивалентно условию $-dx \geq e$, а $dx = e$ эквивалентно комбинации $dx \geq e$ и $-dx \geq -e$

Pade Approximation **Аппроксимация Паде**



ВП определяет коэффициенты рационального полинома, наиболее подходящего для заданного набора первых производных.
Входы **m** и **n** определяют степени полиномов числителя и знаменателя.
Вход **C[0..m+n]** представляет массив, описывающий первые производные заданной функции.
Выход **A[0..m]** отображает коэффициенты полинома числителя.
Выход **B[0..n]** отображает коэффициенты полинома знаменателя.
Пусть f – заданная функция с известными производными и значением
$f_0, f_0^{'},..., f_0^{(m+n)}$.
Существует единственный рациональный полином с $m \geq n$, такой что

$$R(x) = \frac{a_0 + a_1 x + ... + a_m x^m}{1 + b_1 x + ... + b_n x^n} \; c \; R(0) = f(0), R'(0) = f'(0), R^{(m+n)}(0) = f^{(m+n)}(0).$$

Рациональный полином может быть определен путем решения специального линейного уравнения

Функции поиска нулей (рис. 3.52) позволяют находить все нули (корни) одномерной функции, заданной формулой, с помощью методов Ньютона-Рафсона или Риддера, находить комплексные нули комплексного полинома и решение системы нелинейных уравнений.

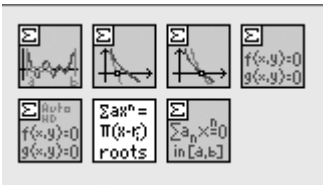


Рис. 3.52. Вид палитры функций поиска нулей

Find All Zeroes of f(x) **Найти все нули функции f(x)**

ВП определяет все нули одномерной функции на заданном интервале.
Набор обозначений входов этого ВП идентичен набору обозначений входов рассмотренного выше ВП **Найти все минимумы одномерной функции** (Find All Minima 1D). Единственное отличие связано с тем, что вход выбора **алгоритма** (algorithm) содержит в качестве вариантов алгоритм **Риддера** (Ridders) (по умолчанию) и алгоритм **Ньютона-Рафсона** (Newton Raphson).

На выходе **нули** (Zeroes) отображаются значения нулей функции, заданной на входе **формула**, а на выходе **f(нулей)** – значения функции в точках нулей

Newton Raphson Zero Finder **Поиск нуля методом Ньютона-Рафсона**

ВП определяет нуль одномерной функции, ограниченной двумя точками, с помощью производной этой функции.

ВП использует метод, который объединяет метод средней точки и метод Ньютона. По первому методу положение нуля определяется как среднее значение абсцисс двух граничных точек

$$x_0 = (x_1 + x_2)/2,$$

а по второму – с помощью выражения

$$x_0 = x_1 - f(x_1)/f'(x_1),$$

где значения x_1 и x_2 должны выбираться из условия

$$f(x_1)f(x_2) < 0.$$

Вход ***h*** представляет величину приращения для расчета производной выражения, заданного **формулой** (formula). По умолчанию значение этого параметра равно 1E – 8

Ridders Zero Finder **Поиск нуля методом Риддера**

ВП определяет нуль одномерной функции на заданном интервале. Функция должна быть непрерывной и иметь различные знаки на концах интервала.

Суть метода Риддера заключается в выборе для заданной функции $f(x)$ таких граничных точек a и b, что $f(a)f(b) < 0$, расчете средней точки c интервала $[a, b]$ и уточнении ее положения с помощью выражения

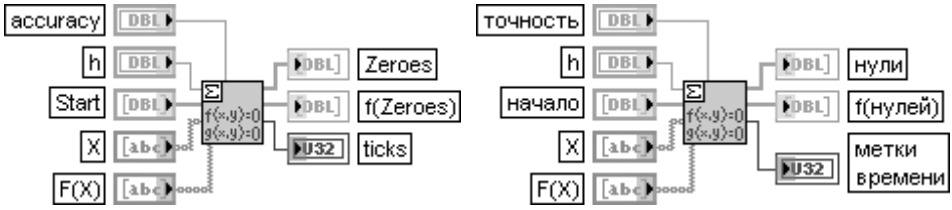
$$c_{new} = c + (c - a)\frac{sign(f(a) - f(b))f(c)}{\sqrt{f^2(c) - f(a)f(b)}}.$$

Набор из трех точек **начало**, c_{new} и **конец** является основой для новой итерации, при этом выбор точек для итерации зависит от того, какое из условий выполняется —

$f(начало)f(c_{new}) < 0$ или $f(c_{new})f(конец) < 0$.

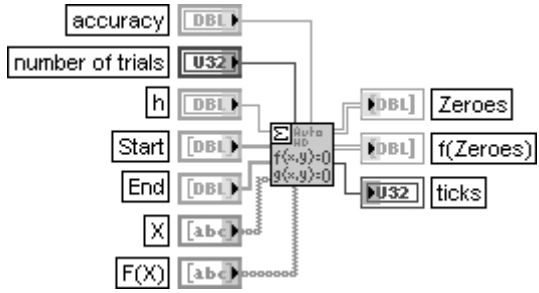
Алгоритм поиска останавливается при выполнении условия $|a - b| < точность$

Nonlinear System Single Solution **Единственное решение системы нелинейных уравнений**

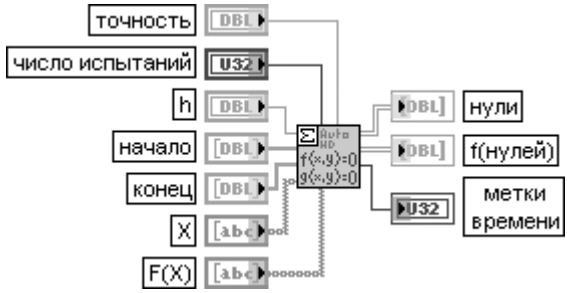


ВП определяет единственное решение системы нелинейных уравнений с **n** переменными и начальной точкой, заданной **n** координатами.
Вход **начало** (Start) определяет начальную точку по **n** координатам.
Вход **X** содержит массив строк, представляющих переменные **x**.
Вход **F(X)** содержит массив строк, определяющих функцию **n** переменных.
Выход **нули** (Zeroes) содержит найденные нули функции **F(X)**

Nonlinear System Solver



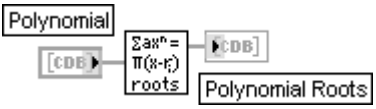
Решение системы нелинейных уравнений

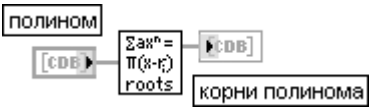


ВП определяет набор решений системы нелинейных уравнений по **n** переменным, каждое из которых ищется в окрестности случайно выбранной точки с **n** координатами.
Вход **число испытаний** (number of trials) определяет число случайно выбранных начальных точек. Алгоритм начинает выполняться с этих точек и ищет нули в их окрестности. По умолчанию число точек равно 5.

Входы **начало** (Start) и **конец** (End) содержат массивы, описывающие, соответственно, левую и правую границы ***n***-мерного интервала. Случайно выбранные начальные точки алгоритма поиска нулей задаются внутри ***n***-мерного многоугольника, ограниченного точками **начала** и **конца**.
В основе работы данного ВП лежит выполнение рассмотренного выше ВП **Единственное решение системы нелинейных уравнений** (Nonlinear System Single Solution).
Алгоритм, используемый для нахождения решения системы нелинейных уравнений, является статистическим по природе. В связи с этим для повышения вероятности нахождения всех решений необходимо увеличивать **число испытаний** (number of trials)

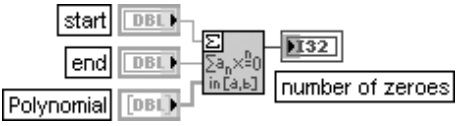
Complex Polynomial Roots **Комплексные корни полинома**

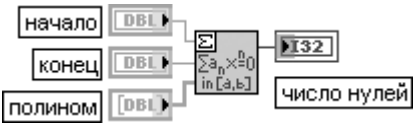




ВП находит комплексные корни комплексного полинома.
Вход **полином** (Polynomial) содержит массив комплексных коэффициентов полинома, размещенных в порядке увеличения степени соответствующей переменной.
Выход **корни полинома** (Polynomial Roots) представляет массив комплексных корней комплексного полинома

Polynomial Real Zero Counter **Расчет числа нулей действительного полинома**





ВП рассчитывает число нулей заданного действительного полинома на действительном интервале без определения значений этих нулей.
Входы **начало** (start) и **конец** (end) определяют левую и правую границы интервала существования полинома. По умолчанию значения этих входов равны 0,0.
Вход **полином** (Polynomial) содержит массив, определяющий коэффициенты полинома в порядке возрастания степени.
Выход **число нулей (number of zeroes)** отображает точное число нулей полинома с коэффициентами, заданными на входе **полином** (Polynomial), на интервале (**начало**, **конец**)

3.3. Дополнительные функции LabVIEW

К числу дополнительных функций LabVIEW можно отнести функции, размещенные в палитрах **Расширенные** (Advanced) и **Графики и звук** (Graphics & Sound). В состав палитры **Расширенные** (рис. 3.53а) входят подпалитры **Манипуляция данными** (Data Manipulation) (рис. 3.53б), **Синхронизация** (Synchronization) (рис. 3.53в), **Порт ввода/вывода** (Port I/O), **ВП доступа к реестру Windows** (Windows

Registry Access VIs) (рис. 3.53г) и **ВП управления входными устройствами** (Input Device Control VIs) (рис. 3.53д), а также функции **Вызвать библиотечную функцию** (Call Library Function) и **Узел кодового интерфейса** (Code Interface Node). Методика разработки динамически связываемых библиотек в LabVIEW рассмотрена далее в разделе 3.3.1, функции манипуляции данными – в разделе 3.3.2, функции синхронизации – в разделе 3.3.3, функции доступа к реестру Windows – в разделе 3.3.4.

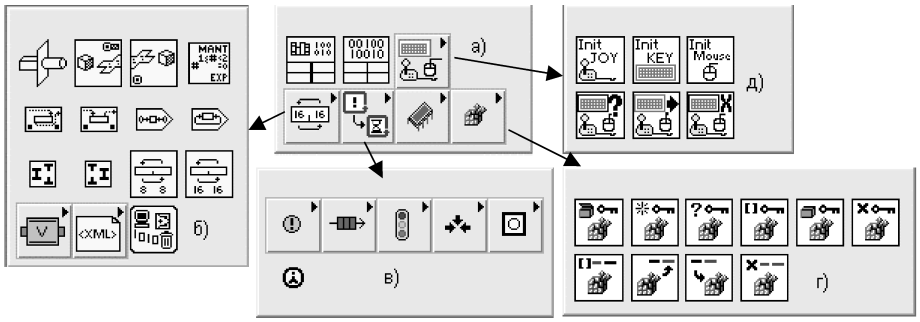


Рис. 3.53. Вид основной палитры ***Расширенные*** *(Advanced) (а) и ее подпалитр (б–е)*

На рис. 3.54 показан вид палитры **Графики и звук** (Graphics & Sound) (рис. 3.54а) и ее подпалитр: **ВП свойств трехмерных графиков** (3D Graph Properties VIs) (рис. 3.54б), **ВП изображений графиков** (Picture Plots VIs) (рис. 3.54в), **Звуковые ВП** (Sound VIs) (рис. 3.54г), **ВП функций изображений** (Picture Functions VIs) (рис. 3.54д), **ВП графических форматов** (Graphics Formats VIs) (рис. 3.54е).

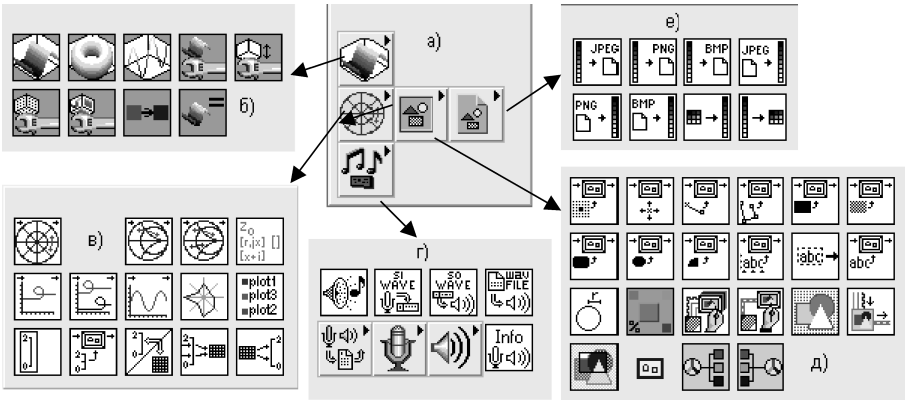


Рис. 3.54. Вид палитры ***Графики и звук*** *(Grafics & Sound) (а) и ее подпалитр (б–е)*

Функции из раздела **ВП графических форматов** (Graphics Formats VIs) и часть функций из раздела **ВП функций изображений** (Picture Functions VIs) рассмотре-

ны далее в разделе 3.3.5, а функции подпалитры **Звуковые ВП** (Sound VIs) – в разделе 3.3.6.

3.3.1. Разработка динамически связываемых библиотек в LabVIEW

Одним из достоинств пакета LabVIEW является возможность вызова процедур, написанных на других языках, с помощью подключения внешних динамически связываемых библиотек (Dynamic Link Library, DLL). Для создания таких библиотек можно использовать любой компилятор процедурного языка, позволяющий создавать динамически связываемые библиотеки для Windows. Необходимость применения DLL возникает обычно 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 функция **Вызвать библиотечную функцию** (Call Library Function)  , находящаяся в палитре **Расширенные** (Advanced). Первоначально функция не имеет параметров и возвращает тип **void**. Настройка атрибутов функции производится с помощью одноименного диалогового окна (рис. 3.55), вызываемого двойным щелчком ЛКМ на иконке функции или выбором пункта **Конфигурировать** (Configure) из контекстного меню функции.

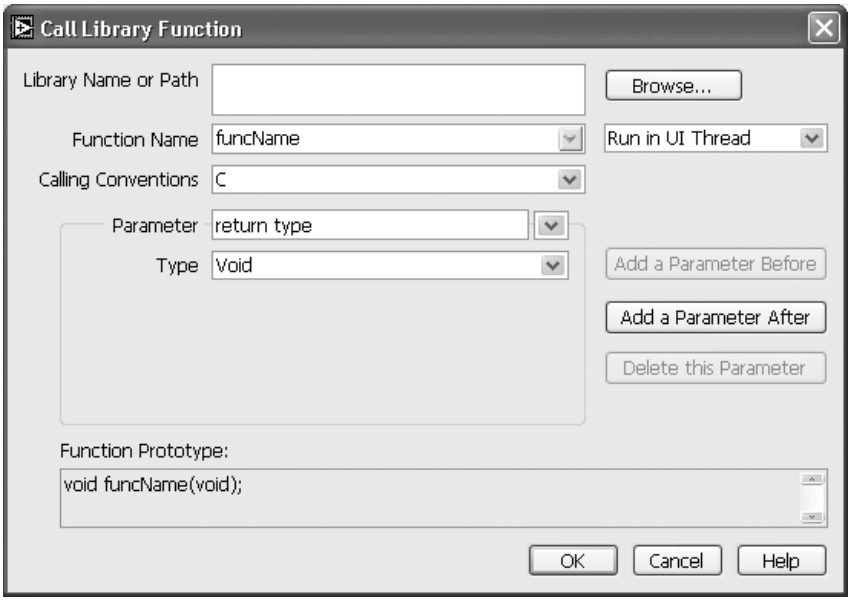


Рис. 3.55. Вид диалогового окна функции ***Вызвать библиотечную функцию***

Поле **имя или путь библиотеки** (Library Name or Path) служит для указания полного пути к двоичному файлу откомпилированной внешней библиотеки.

В поле **имя функции** (Function Name) указывается имя вызываемой из библиотеки функции. Имя должно быть соблюдено с точностью до регистра букв, так как система Windows и компиляторы языка **C** различают строчные и прописные буквы. Справа от этого поля находится выпадающее меню, содержащее следующее строки:

- **Выполняться в потоке пользовательского интерфейса** (Run in UI Thread);
- **Выполняться с возможностью повторного вызова** (Reentrant).

Это меню определяет способность функции работать в многопоточной среде. Так, например, если функция может выполняться только в одном потоке одновременно, 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.

Поле **соглашения вызова** (Calling Conventions) описывает стандарт вызова функции. LabVIEW поддерживает два стандарта:

- **Stdcall** – стандарт, принятый для системных вызовов и динамически связываемых библиотек Windows. Также этот стандарт принят как основной для языка Pascal;
- **C** – стандарт вызовов языка **C**.

Ниже и правее находятся три кнопки манипулирования параметрами: **добавить параметр до отображаемого** (Add a Parameter Before), **добавить параметр после отображаемого** (Add a Parameter After), **удалить этот параметр** (Delete this Parameter). Эти кнопки позволяют устанавливать число и тип аргументов вызываемой функции. Следует быть внимательным при настройке данных параметров. LabVIEW не может определить, соответствуют ли указанные число и тип аргументов реальным параметрам. Ошибка, допущенная при описании прототипа вызова функции, может приводить к краху пакета LabVIEW и выгрузке его из памяти.

Сразу после создания у функции присутствует только один аргумент **возвращаемый тип** (return type), который описывает возвращаемое функцией значение и может принадлежать 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), **цифровая таблица** (Digital Table), ActiveX и **адаптивный к типу** (Adapt to Type).

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.

Типу **числовой** (Numeric) в LabVIEW принадлежат 8 различных подтипов, шесть из которых целочисленные и два – 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
Восьмиразрядное целое со знаком	Signed 8-bit Integer	signed char
16-разрядное целое со знаком	Signed 16-bit Integer	signed short int
32-разрядное целое со знаком	Signed 32-bit Integer	signed long int
Восьмиразрядное 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
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 может принимать два значения:

- **Значение** (Value) – в стек вызовов помещается значение параметра, поэтому даже если внешняя функция изменит значение своего аргумента, то значение аргумента, хранящееся в памяти пакета LabVIEW, не изменится и значение на выходных терминалах функции **Вызвать библиотечную функцию** будет равно соответствующим входным терминалам/аргументам;
- **Указатель (ссылка) на значение** (Pointer to Value) – в стек вызовов помещается указатель на значение параметра. Это позволяет изменить значение параметра, хранящееся в памяти LabVIEW, и вернуть 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.

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. 3.56 показана блок-диаграмма ВП чтения данных по шине USB с помощью интерфейсной микросхемы FT232B [13], в котором для передачи массива используется формат **указатель на данные массива**.

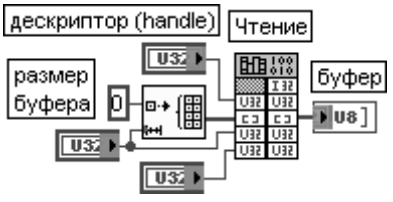


Рис. 3.56. Блок-диаграмма ВП чтения данных по шине USB

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 четыре 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 подобного 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 четырехбайтного значения.

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. 3.56).

```
#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;
}
```

3.3.2. Функции манипуляции данными

Функции манипуляции данными (рис. 3.57) 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, перевода данных в строку XML и обратно, записи и считывания файлов XML. Потребность в таких преобразованиях данных может возникать, например, при работе с сетевым протоколом TCP/IP или при обмене данными с внешними устройствами.

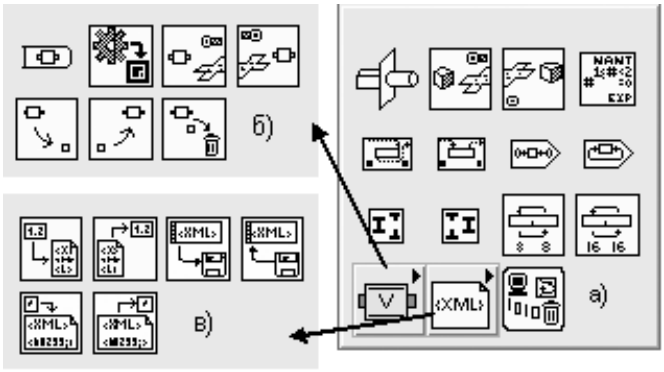


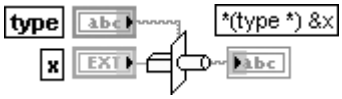
Рис. 3.57. Вид основной палитры (а) и подпалитр (б–в) функций манипуляции данными

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 о 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.

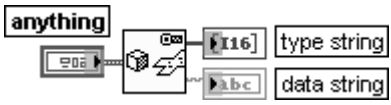
Ниже в таблицах приведены описания функций манипуляции данными основной палитры.

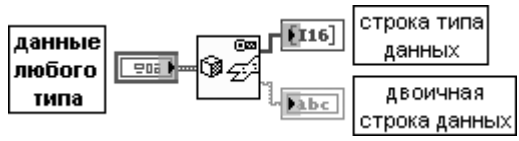
Type Cast **Приведение типа**



Функция преобразует данные на входе **x** к типу данных, заданных 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. Соединительные выводы отображают тип данных этой полиморфной функции.
Вход **тип данных** (type) представляет тип данных, к которому производится преобразование. Он является только «шаблоном», служащим для определения типа, любые реальные данные на этом входе игнорируются [2].
Вход **x** используется для передачи данных, преобразуемых к заданному **типу** (type).
Вход **x** может быть любого типа.
Выход ***(тип*)&x** (*(type *) &x) отображает значение, поданное на вход **x** и преобразованное к заданному **типу данных**.
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.
Для преобразования данных логического типа, полученных с помощью версии LabVIEW 4. или ранее, необходимо в контекстном меню этой функции выбрать строку Convert 4.x Data для преобразования данных к формату, который удобочитаем для версии LabVIEW 5.0
и более поздних. В режиме Convert 4.x Data эта функция интерпретирует **x** как данные, хранящиеся в формате LabVIEW 4.x, и отображает иконку для этой функции
с помещенной на ней красной надписью. LabVIEW 4.x и более ранние версии хранили данные логического типа в двух байтах, за исключением массива, для которого LabVIEW хранит каждый логический элемент в виде отдельных битов. LabVIEW 5.0 и более поздние версии хранят логические значения в виде байта, в том числе и в случае представления в виде массива. National Instruments рекомендует доработку приложений, которые используют режим Convert 4.x Data как долговременное решение

Flatten To String **Перевести в строку**

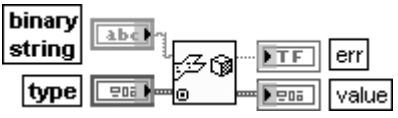




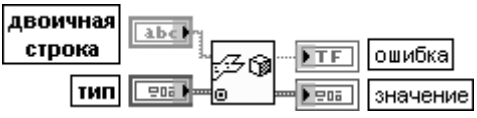
Функция преобразует **данные любого типа** (anything) в **приведенные данные** (flattened) и возвращает **двоичную строку данных** (data string), содержащую двоичное представление данных, и **строку типа данных** (type string), содержащую дескриптор типа, который описывает тип **данных любого типа**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

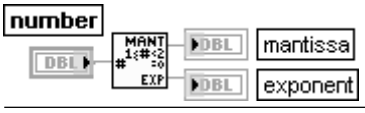


Восстановить из строки

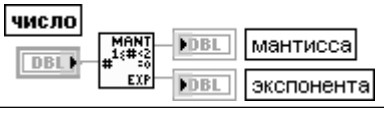


Функция преобразует **двоичную строку** (binary string) с типом, указанным на входе **тип данных** (type), в соответствующее значение на выходе **значение** (value). Вход **двоичная строка** должен содержать данные **приведенного типа** (flatten), которые могут быть преобразованы к типу, указанному на входе **тип данных**.
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 данных** 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 данных**

Mantissa & Exponent



Мантисса и экспонента



Функция возвращает мантиссу и экспоненту входного **числа** (number) так, что **число** = **мантисса** * 2^**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 **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 **Сдвиг вправо с переносом**

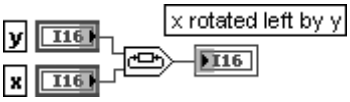
Функция сдвигает каждый бит входного **числа** (value) вправо (от более значимого к менее значимому биту), вставляет **перенос** (carry) в старший бит и возвращает младший бит.
Вход **перенос** передает новый старший бит числа.
Вход **число** должен быть целым. Он не может быть массивом или кластером.
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

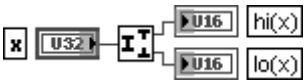
нули в младшие биты. Если **y** меньше 0, функция сдвигает **x** вправо на **y** бит (от более значимого к менее значимому биту) и вставляет нули в старшие биты.
Вход **x** может быть любого целого типа. Если **x** является 8-, 16-, или 32-битовым целым и **y** больше 8, 16 или 32 или меньше −8, −16 или −32 соответственно, то выходное значение будет нулевым.
Выход **x << y** отображает результат сдвига и имеет такое же числовое представление, что и **x**

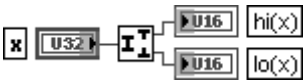
Rotate **Циклический сдвиг**



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, **y** ± 32 соответственно даст такое же выходное значение, что и сдвиг на **y**. Например, если **x** является восьмибитовым 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**

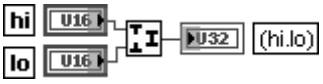
Split Number **Разделить число**

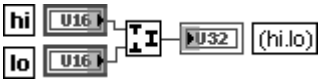




Функция разделяет число на составляющие его байты или слова.
Вход **x** может быть 16- или 32-битовым целым числом, массивом или кластером чисел с таким представлением.
Выходы **hi**(**x**) и **lo**(**x**) отображают целые числа, занимающие половину ширины **x**. **hi**(**x**) и **lo**(**x**) могут быть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

В следующих таблицах приведены описания функций из подпалитры **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

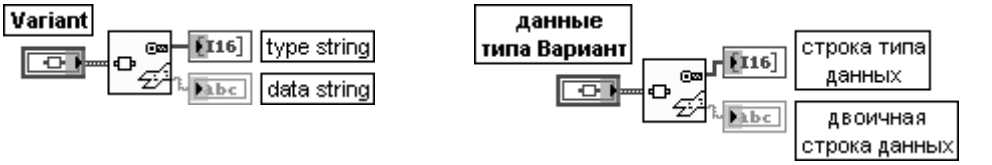
To Variant **Преобразование к типу Вариант**

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**

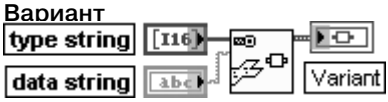
Функция преобразует данные типа **Вариант** в данные типа LabVIEW так, что LabVIEW может отображать или обрабатывать данные. Эту функцию можно использовать для преобразования данных типа **Вариант** в данные ActiveX.
Вход **тип** (type) задает тип данных LabVIEW, к которому необходимо преобразовать данные 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 данных**.
Выход **данные** (data) отображает данные типа **Вариант**, измененные в соответствии с типом, установленным на входе **тип данных**. Если **Вариант** не может быть преобразован к заданному типу данных, выход **данные** возвращает значение с типом данных по умолчанию

Variant To Flattened String **Данные типа Вариант в приведенную строку**

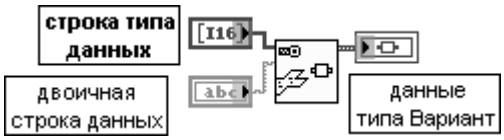


Функция преобразует данные типа **Вариант** в **приведенные данные** (flattened) и возвращает **двоичную строку данных** (data string), содержащую двоичное представление данных, и **строку типа данных** (type string), содержащую дескриптор типа, который описывает тип данных **Вариант**.
Выходы **двоичная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

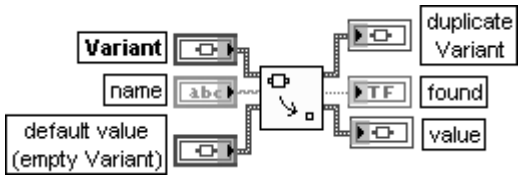


Приведенную строку в данные типа

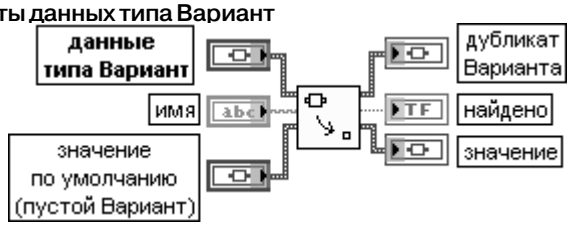


Функция преобразует приведенную строку в данные типа **Вариант**. Для формирования **двоичной строки данных** (data string) и **строки типа данных** (type string) необходимо использовать функцию **Перевести в строку** (Flatten to String).
Входы **двоичная строка данных** (data string) и **строка типа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

Get Variant Attribute



Получить атрибуты данных типа Вариант

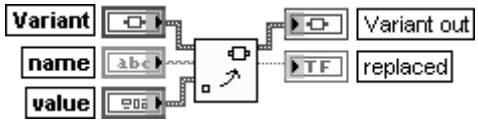


Функция извлекает **имена** (names) и **значения** (values) всех атрибутов или **значение** (value) простого атрибута в зависимости от вида параметра на входе **имя** (name). Соединительная панель отображает типы данных по умолчанию для этой полиморфной функции.
Вход **Вариант** представляет данные типа **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 **дубль Варианта** (duplicate Variant) повторяет данные типа **Вариант**, поданные на

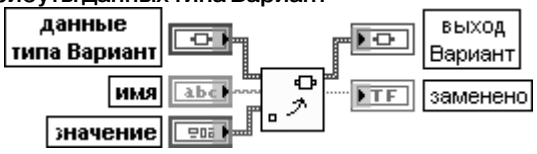
одноименный вход.

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 заданные 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



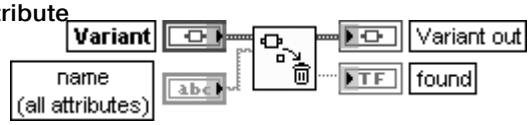
Установить атрибуты данных типа Вариант



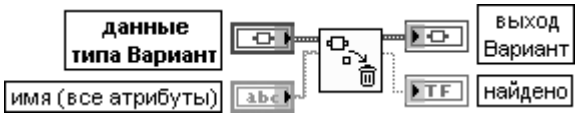
Функция изменяет или создает атрибут и значение для данных типа **Вариант**.
Вход **Вариант** передает данные типа **Вариант**, для которых необходимо создать атрибут и значение или заменить значение.
Вход **имя** (name) представляет имя атрибута, который редактируется или создается. Если имя совпадает 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



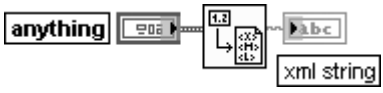
Функция удаляет атрибут(ы) и значение(я) данных типа **Вариант**, подаваемых на вход Variant.

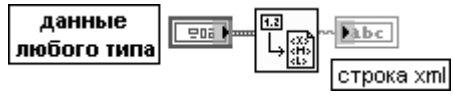
Вход **имя** (name) содержит имя атрибута, который необходимо удалить. По умолчанию предполагается удаление всех атрибутов и связанных с ними значений данных типа **Вариант**. Если **имя** совпадает с атрибутом, то эта функция удаляет атрибут и его значение.

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**

Flatten To XML **Перевести в XML**



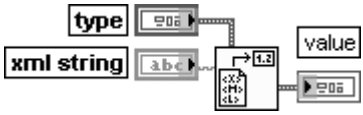


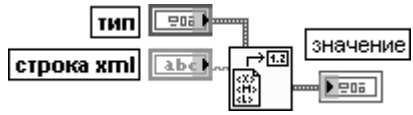
Функция преобразует **данные любого типа** (anything) в строку XML в соответствии со схемой преобразования LabVIEW XML. Если **данные любого типа** содержат символы <, > или &, то функция преобразует их в символы <, > или & соответственно. Для преобразования других символов, таких как «, необходимо использовать 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**

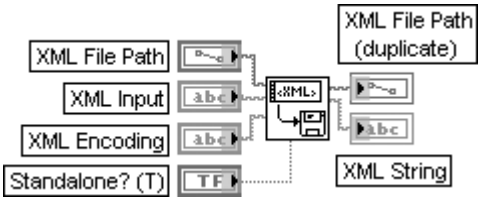




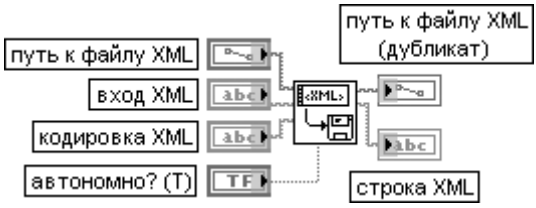
Функция преобразует строку XML в тип данных LabVIEW. Если вход **строка xml** (xml string) содержит символы <, > или &, то функция преобразует их в символы <, >

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

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 ли 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 тэги XML верхнего уровня, найденные между концами тэгов </Version> и </LVData> в массиве строк (при вводе массива строк) или в простой строке (при вводе строки). Этот выход может быть подключен к 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 < > or & в <, > или & соответственно. Данный ВП должен использоваться для преобразования других символов, таких как "

Request Deallocation **Запрос освобождения**

Функция освобождает неиспользуемую память после работы ВП, в состав которого она включена. При вызове подпрограммы из ВП верхнего уровня LabVIEW выделяет пространство памяти, в котором эта подпрограмма работает. Когда подпрограмма завершает работу, LabVIEW не освобождает пространство данных до завершения работы ВП верхнего уровня или до остановки всего приложения, что может привести к переполнению памяти и ухудшению характеристик. Размещение функции **Запрос освобождения** (Request Deallocation) в подпрограмме позволяет освободить память сразу после завершения ее работы. Для этого необходимо установить **флаг** (flag) в состояние ИСТИНА

3.3.3. Функции и ВП синхронизации

Функции и ВП синхронизации (рис. 3.58а) используются для синхронизации параллельно выполняющихся задач и передачи данных между такими задачами. В состав функций и ВП входят функции **Операции уведомителя** (Notifier Operations) (рис. 3.56б), **Операции очереди** (Queue Operations) (рис. 3.58в), ВП **Семафор** (Semaphore VIs) (рис. 3.58г), ВП **Встреча (**Rendezvous VIs) (рис. 3.58д) и **Функции случаев** (Occurrences Functions) (рис. 3.58е).

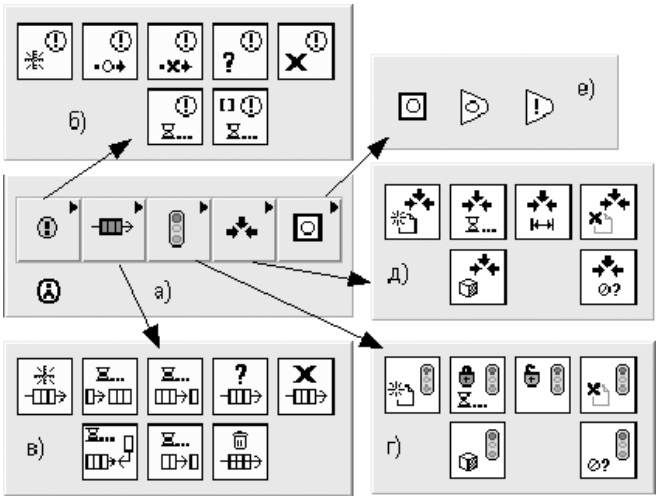


Рис. 3.58. Вид палитры (а) и подпалитр (б–е) функций и ВП синхронизации

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.

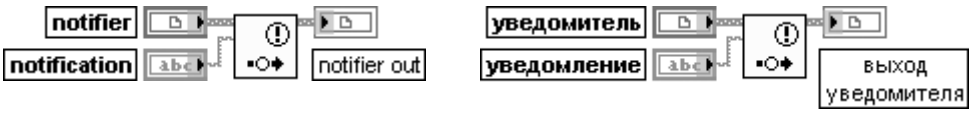
Вход **тип элемента данных** (element data type) представляет тип данных, которые должен содержать уведомитель. К этому входу могут быть подключены данные любого типа.

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

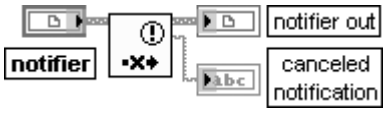
Send Notification **Отправить уведомление**



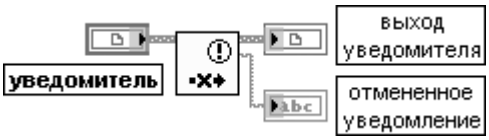
Функция отправляет сообщение всем функциям, ожидающим уведомителя. Все функции **Ожидать уведомления** (Wait on Notification) и **Ожидать уведомления от множества** (Wait on Notification from Multiple), ожидавшие уведомителя, выходят из ожидания и продолжают выполнение.

Вход **уведомление** (notification) содержит отправляемое сообщение. Его тип данных изменяется в соответствии с подтипом **уведомителя** (notifier)

Cancel Notification



Отменить уведомление

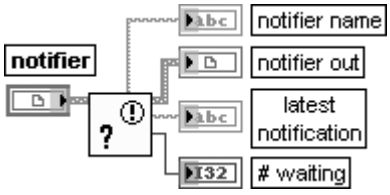


Функция стирает какие-либо сообщения, содержащиеся в уведомителе, и возвращает сообщение об отмене уведомления.

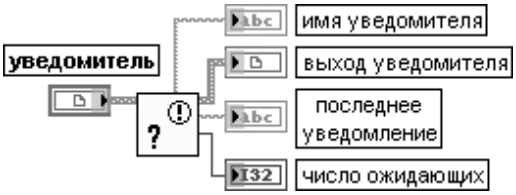
Выход **отмененное уведомление** (cancelled notification) содержит последнее уведомление, отправленное уведомителю. Тип данных выхода изменяется в соответствии с подтипом **уведомителя** (notifier).

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



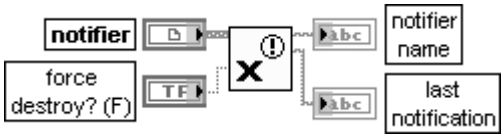
Функция возвращает информацию о текущем состоянии уведомителя, таком как последнее неотмененное уведомление, отправленное уведомителю.

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 к входу функции **Получить уведомителя**.

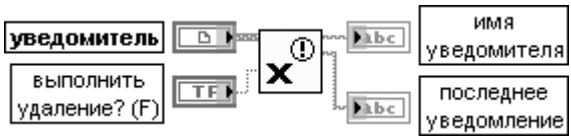
Этот тип данных изменяется в соответствии с подтипом уведомителя.

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



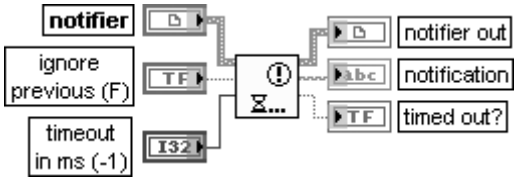
Закрыть уведомителя



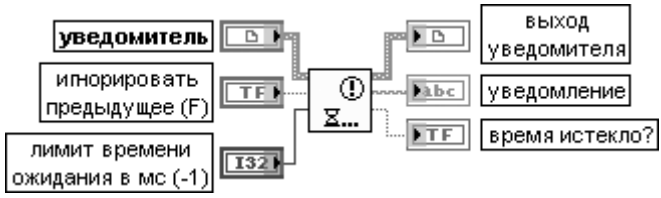
Функция закрывает ссылку к уведомителю.

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 ее многократного вызова 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.

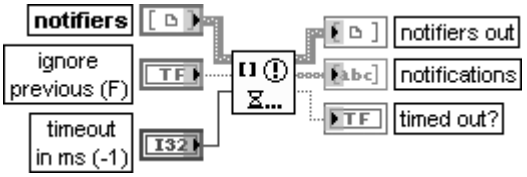
Вход **игнорировать предыдущее** (ignore previous) определяет возможность игнорировать сообщения, отправленные уведомителю перед вызовом этой функции. Если при установке на входе состояния ИСТИНА уведомитель содержал сообщение перед вызовом этой функции, то функция ожидает принятия уведомителем другого 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.

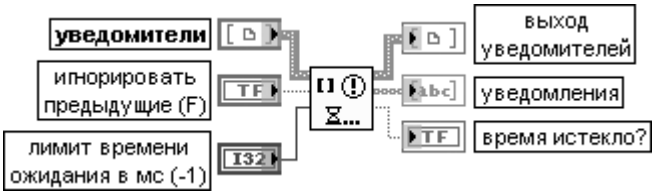
Если на входе **игнорировать предыдущее** (ignore previous) установлено состояние ЛОЖЬ, то каждый экземпляр функции **Ожидать уведомления** (Wait on Notification) ожидает, пока сообщение

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 функция **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 данной функции или функции **Ожидать уведомления 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 уведомителям.
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 уведомителями. 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. 3.59 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).

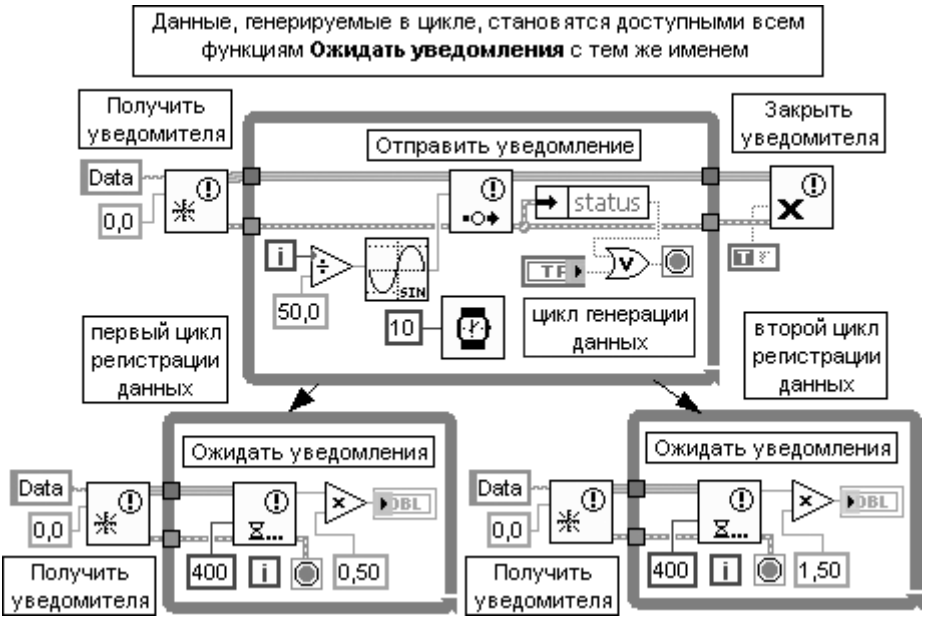
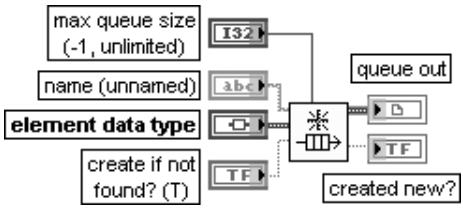
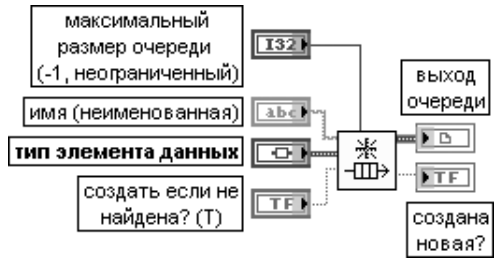


Рис. 3.59. Блок-диаграмма ВП ***Уведомитель–демультиплексор*** *(Notifier Demultiplexer)*

В следующих таблицах приведены 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.

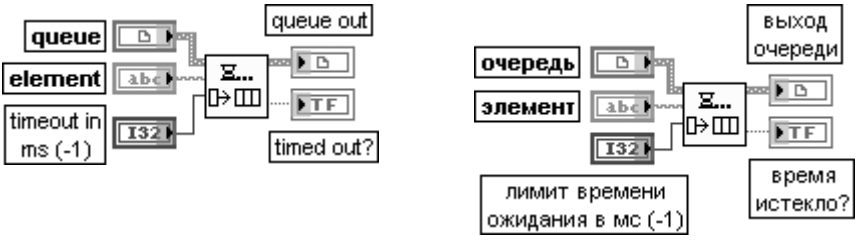
Obtain Queue	**Получить очередь**
	

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 может содержаться в очереди. По умолчанию это число равно –1, что соответствует неограниченному числу элементов очереди.

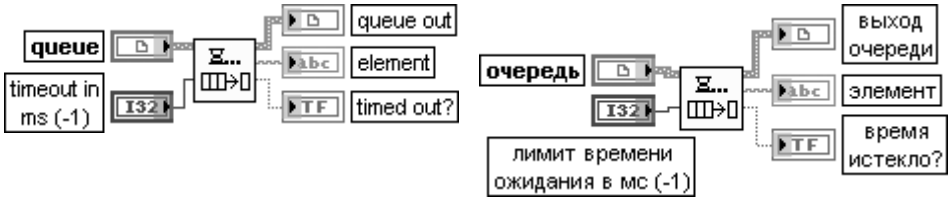
Если очередь достигает **максимального размера очереди**, то функции **Поставить элемент в очередь** (Enqueue Element) или **Поставить элемент в очередь на противоположном конце** (Enqueue Element at Opposite End) ожидают, пока функции **Убрать элемент из очереди** (Dequeue Element) или **Сбросить очередь** (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**



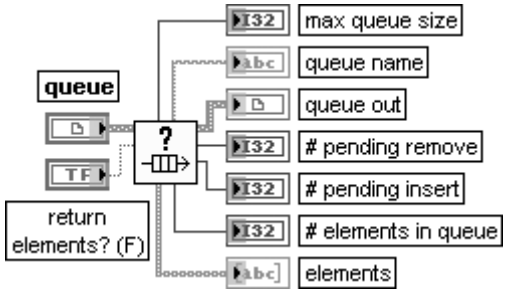
Функция добавляет элемент в конец очереди.
Если очередь заполнена, то функция ожидает в течение времени, заданного на входе **лимит времени ожидания в мс** перед выходом по времени ожидания. Если место в очереди во время ожидания становится доступным, то функция вставляет элемент и выход **время истекло?** (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**

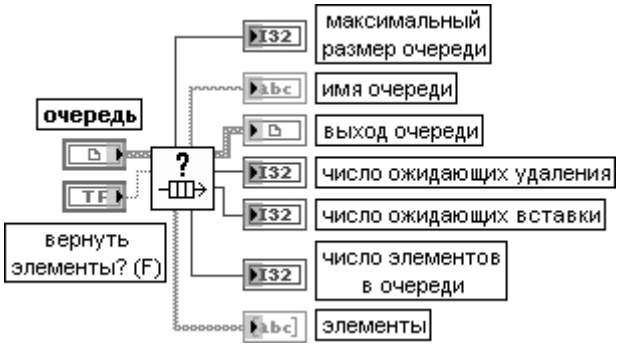


Функция возвращает элемент из начала очереди без его удаления из очереди.
Для возврата элемента из начала очереди и удаления из очереди необходимо использовать функцию **Убрать элемент из очереди** (Dequeue Element). Если очередь пустая, то функция ожидает в течение интервала времени, заданного на входе **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



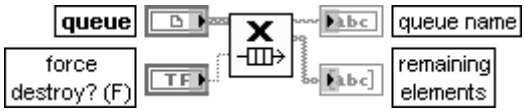
Получить статус очереди



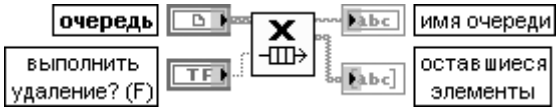
Функция возвращает информацию о текущем состоянии очереди, такую, например, как число элементов, находящихся в очереди. Эту функцию можно использовать также для проверки того, что **очередь** является достоверной ссылкой очереди. Если **очередь** не является достоверной ссылкой очереди, то функция возвращает 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

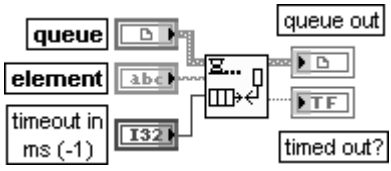


Закрыть очередь

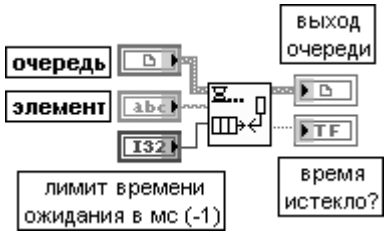


Функция закрывает ссылку к очереди.
Вход **выполнить удаление?** (force destroy?) показывает возможность удаления очереди. При установке значения ЛОЖЬ (по умолчанию) для удаления очереди необходимо вызвать данную функцию столько раз, сколько было получено ссылок к очереди, или остановить все ВП, используя ссылку к очереди. При установке значения ИСТИНА функция удаляет очередь без многократного вызова функции или остановки всех ВП, использующих ссылку к очереди

Enqueue Element At Opposite End



Поставить элемент в очередь на противоположном конце

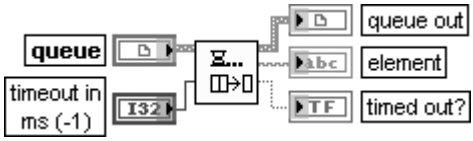


Функция добавляет элемент к началу очереди. Эта функция аналогична функции **Поставить элемент в очередь** (Enqueue Element).
Вход **лимит времени ожидания в мс** (timeout in ms) показывает, сколько миллисекунд функция ожидает доступного места в очереди, если очередь заполнена. По умолчанию значение входа

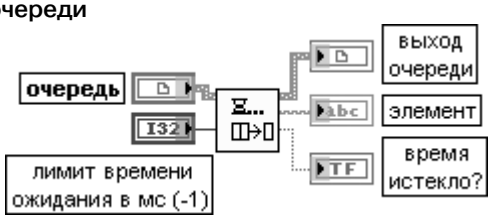
равно –1, что соответствует отсутствию ожидания.

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

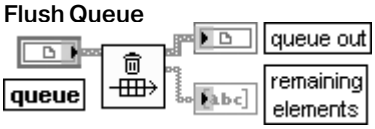


Убрать элемент из очереди

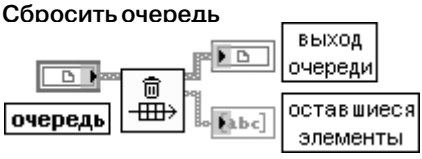


Функция удаляет элемент из начала очереди и возвращает его на выходе **элемент**. Роль входа **лимит времени ожидания в мс** (timeout in ms) и выходов **элемент** (element) и **время истекло?** (timed out?) идентична роли одноименного входа и выходов функции **Просмотреть элемент очереди** (Preview Queue 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. 3.60 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)

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.

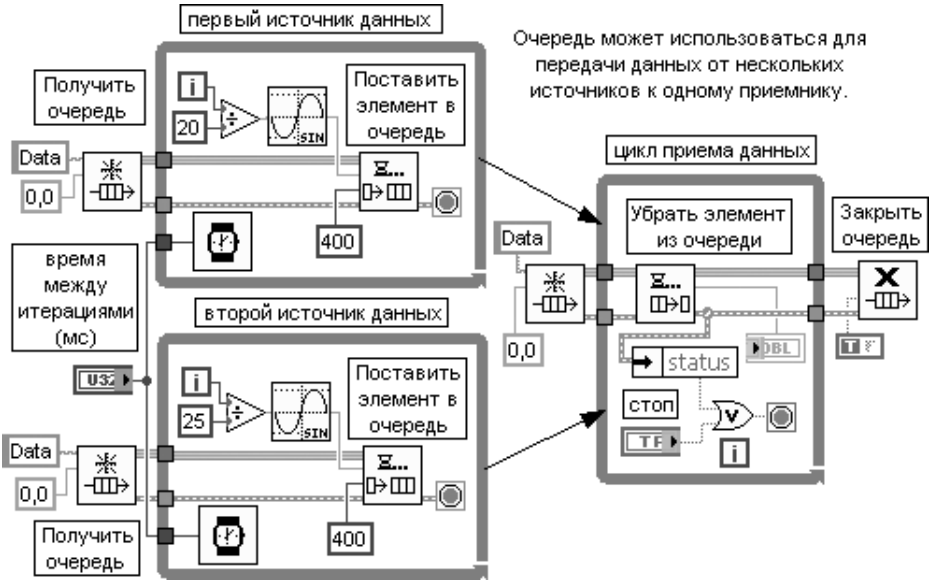
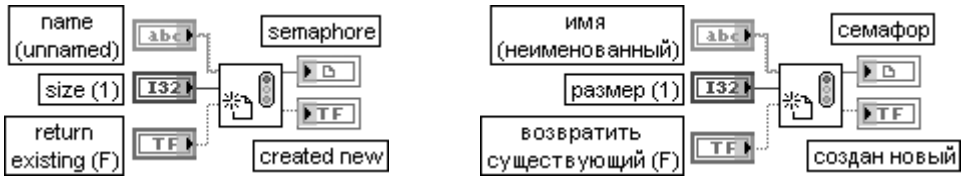


Рис. 3.60. Блок-диаграмма ВП ***Очередь-мультиплексор*** *(Queue Multiplexer)*

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, чтобы создать неименованный семафор. 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

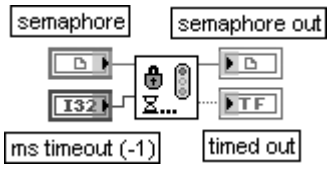
семафор в любом ВП, даже не находящемся в иерархии ВП **Приложение**, но семафор прекратит существование, когда иерархия ВП **Приложение** закроется.

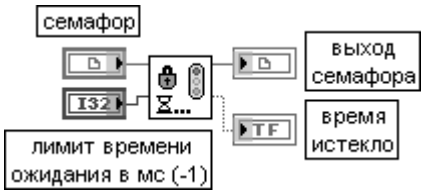
Вход **размер** (size) определяет максимальное число задач, которые могут одновременно получить семафор. Если именованный семафор уже существует, то подключение значения к этому входу не изменяет размер семафора. По умолчанию размер равен 1.

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**



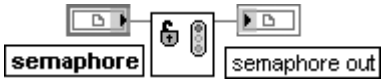


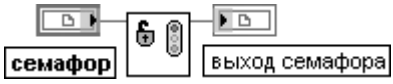
Функция обеспечивает получение доступа к семафору.

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

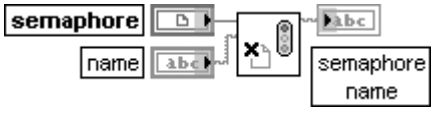
Release Semaphore **Закрыть семафор**

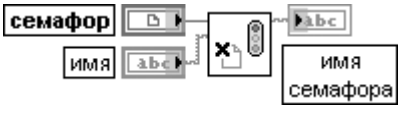




Функция закрывает доступ к семафору. Если ВП **Получить семафор** (Acquire Semaphore) ожидает семафор, который закрывается данным ВП, то он прекращает ожидание и продолжает выполнение

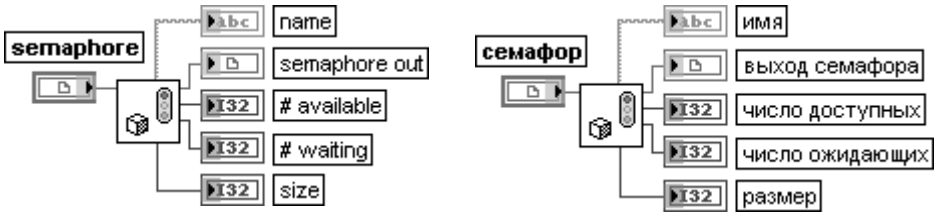
Destroy Semaphore **Ликвидировать семафор**





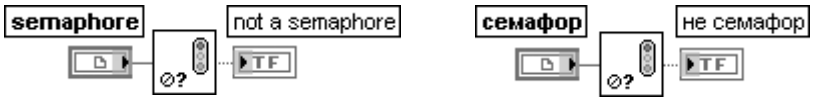
Функция ликвидирует определенный семафор. Все ВП **Получить семафор** (Acquire Semaphore), которые ожидают этот семафор, немедленно выходят из ожидания и возвращают ошибку

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. 3.61 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. 3.61а). При этом в каждом подприборе (рис. 3.61б) 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**.

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.

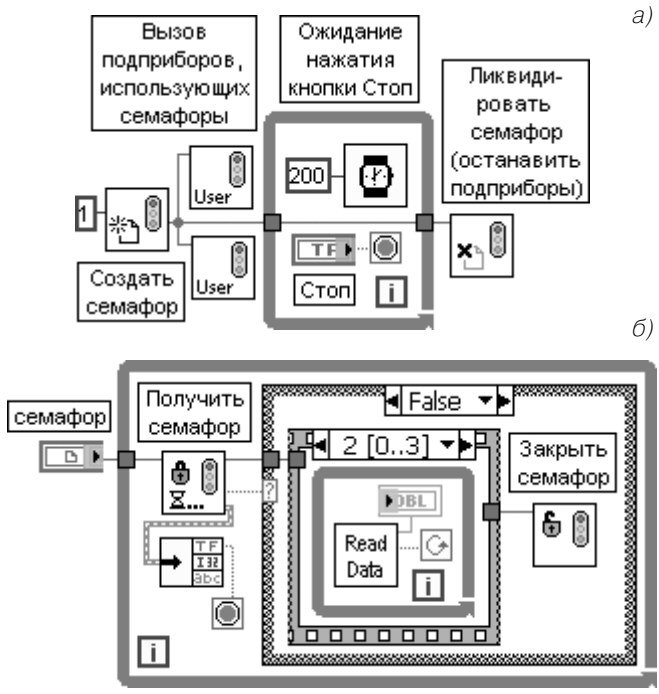


*Рис. 3.61. Блок-диаграммы ВП **Семафор с подприборами** (Semaphore with SubVIs) (а) и подприбора (б)*

Create Rendezvous **Создать встречу**

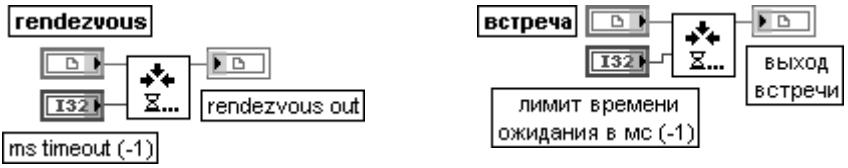
Функция просматривает существующую встречу или создает новую встречу и возвращает ссылку. Эту ссылку можно использовать для вызова других ВП из подпалитры **Встреча** (Rendezvous).

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 к входу **размер** не изменяет размер встречи. Для изменения размера именованной встречи необходимо использовать ВП **Изменить размер встречи** (Resize Rendezvous).

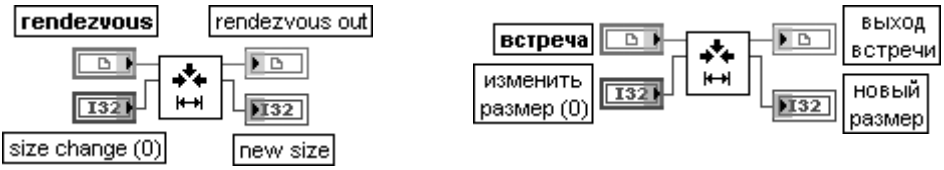
Вход **вернуть существующую** (return existing) определяет возможность создания новой встречи, если встреча с таким именем не существует. По умолчанию значение входа равно ЛОЖЬ, что позволяет создавать встречу, если она не существует

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**

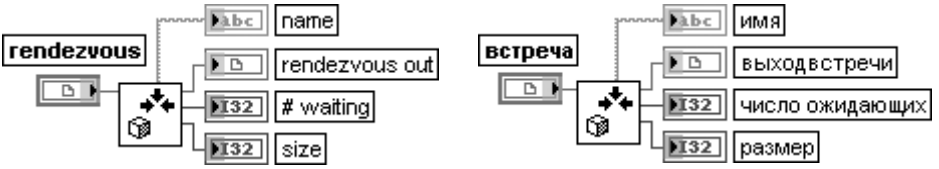


Функция изменяет размер **встречи** (rendezvous) с помощью входа **изменить размер** (size change) и возвращает **новый размер** (new size).
Вход **изменить размер** (size change) определяет максимальное количество задач, которые должны поступить на встречу. Отрицательный размер уменьшает число задач. 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

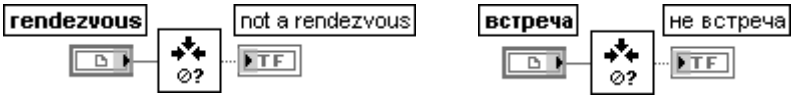
Get Rendezvous Status **Получить статус встречи**



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

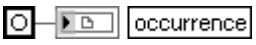
Not A Rendezvous **Не встреча**



Функция возвращает значение ИСТИНА, если **встреча** (rendezvous) не является достоверной ссылкой к встрече

Функции случаев (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 **Генерировать 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 **Генерировать случай** (Generate Occurrence) подключается к какому-либо набору функций **Ожидать случай** и **Установить случай**.
В отличие от других **ВП синхронизации** (Synchronization VIs) каждая функция **Генерировать случай** на блок-диаграмме представляет простой, уникальный **случай**. В этом отношении функция **Генерировать случай** аналогична константе. При выполнении ВП вызов функции **Генерировать случай** формирует одно и то же значение. Например, если функция **Генерировать случай** устанавливается в структуре цикла, то она выводит одинаковое значение при каждой итерации. Если функция **Генерировать 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** (Wait on Occurrence) и **Установить случай** (Set Occurrence).

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).

3.3.4. ВП доступа к реестру Windows

Реестр системы Windows представляет специальную базу данных, в которой находится почти вся информация, необходимая для загрузки и конфигурирова-

ния системы и настройки ее под конкретного пользователя [14]. 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 имеется ряд ключей, называемых **корневыми ключами**.

С помощью ВП доступа к реестру Windows (рис. 3.62) можно создавать, открывать, запрашивать, перечислять, закрывать и удалять ключи реестра Windows. Также можно перечислять, считывать, записывать и удалять значения 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.

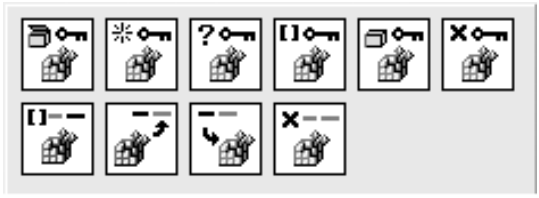
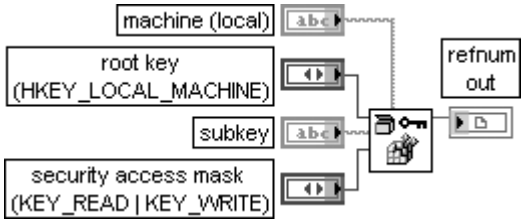


Рис. 3.62. Вид палитры ВП доступа к реестру Windows

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:

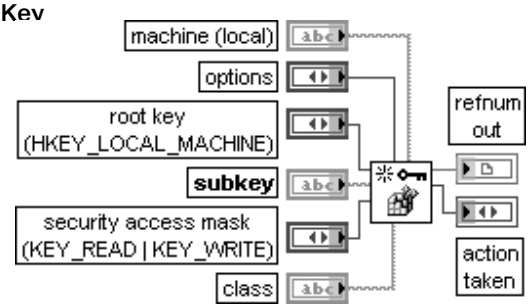
Ключ	Описание
HKEY_CLASSES_ROOT	Ссылка на подключ HKEY_LOCAL_MACHINE\SOFTWARE\CLASSES
HKEY_CURRENT_USER	Ссылка на текущий профиль пользователя
HKEY_LOCAL_MACHINE	Свойства аппаратного и программного обеспечения
HKEY_USERS	Информация о пользователях
HKEY_CURRENT_CONFIG	Ссылка на текущий профиль аппаратного 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:

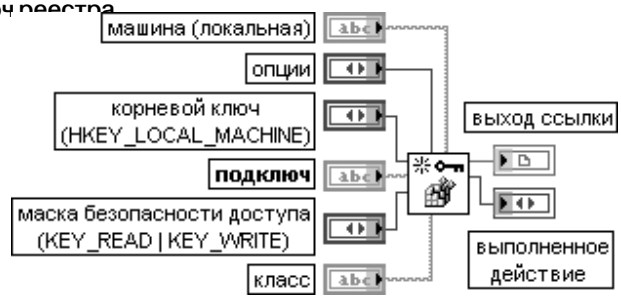
KEY_QUERY_VALUE	Запросить значение ключа
KEY_SET_VALUE	Установить значение ключа
KEY_CREATE_SUB_KEY	Создать подключи ключа
KEY_ENUMERATE_SUB_KEYS	Перечислить подключи ключа
KEY_NOTIFY	Оповестить ключ
KEY_CREATE_LINK	Создать связь ключа
KEY_READ	Считать ключ
KEY_WRITE	Записать ключ
KEY_ALL_ACCESS	Общий доступ к ключу

Выход ссылки (refnum out) содержит ссылку к открытому ключу

Create Registry Kev



Создать ключ реестра



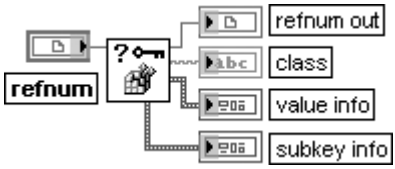
ВП создает ключ в реестре Windows или открывает его, если он уже существует.
Вход **опции** (options) определяет специальные опции для ключа.

REG_OPTION_NON_VOLATILE	Ключ с присвоенным значением
REG_OPTION_VOLATILE	Ключ без присвоенного значения
REG_OPTION_BACKUP_RESTORE	Дублирование ключа для восстановления

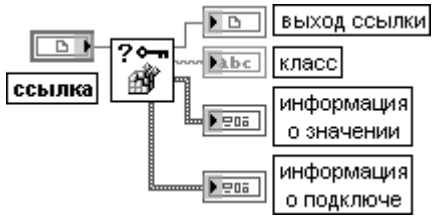
Вход **класс** (class) определяет класс (тип объекта) ключа.
Выход **выполненное действие** (action taken) показывает действие, выполненное ВП.

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



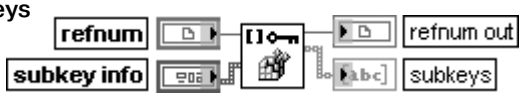
ВП извлекает информацию о ключах.
Выход **информация о значении** (value info) содержит набор параметров, описывающих набор значений и данных.

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.

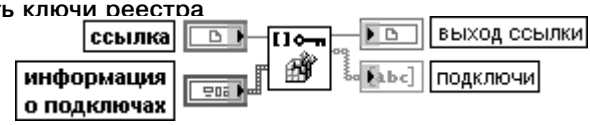
Выход **информация о подключе** (subkey info) содержит набор параметров, описывающих подключи.

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



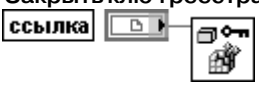
Перечислить ключи реестра



ВП отображает подключи определенного ключа или подключа. Этот ВП необходимо использовать вместе с ВП **Запросить информацию о ключе реестра** (Query Registry Key Info)

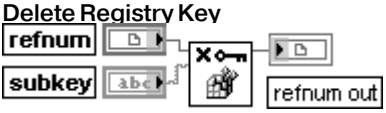
Close Registry Key

Закрыть ключ реестра



ВП закрывает ключ в реестр Windows

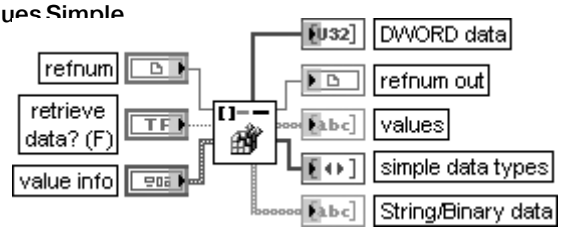
Delete Registry Key



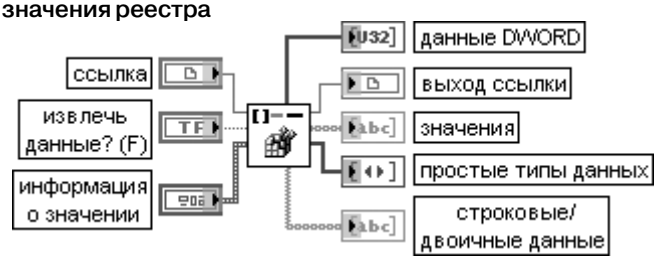
Удалить ключ реестра

ВП удаляет определенный ключ или подключ

Enum Registry Values Simple



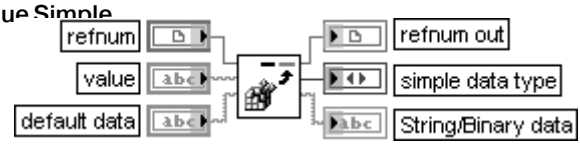
Перечислить значения реестра



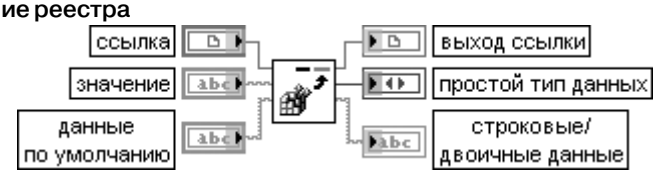
ВП отображает значения определенного ключа или подключа.
Если на входе **извлечь данные?** (retrieve data?) установлено состояние ИСТИНА, то ВП возвращает **32-разрядные целые числа** (DWORD data) и **строковые/двоичные данные** (String/Binary data)

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

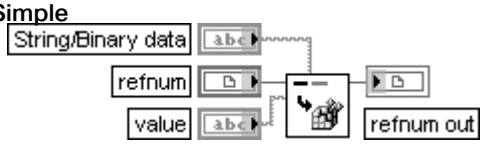


Читать значение реестра

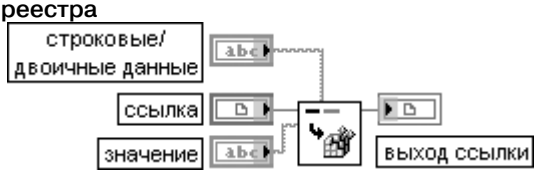


ВП читает данные и упрощенный тип данных из значения реестра.
Если ключ или значение не существует или возникает неисправимая ошибка, ВП возвращает данные по умолчанию. Этот полиморфный ВП может использоваться для записи строковых или числовых данных. Тип данных, подключенных к входу **данные по умолчанию** (default data), определяет реализацию используемого полиморфного ВП

Write Registry Value Simple

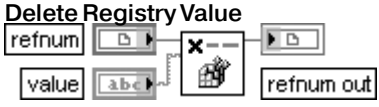


Записать значение реестра



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 к входу **строковые/двоичные данные** (String/Binary data), определяет реализацию используемого полиморфного ВП

Delete Registry Value



Удалить значение реестра

ВП удаляет значение реестра

На рис. 3.63 в качестве примера приведена блок-диаграмма ВП управления параметрами Web-камеры с помощью ВП доступа к реестру.

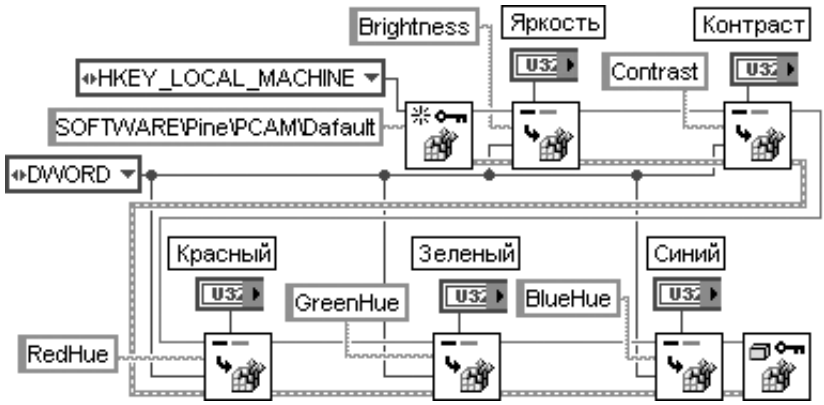
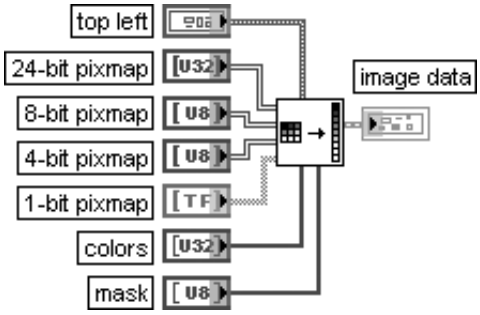


Рис. 3.63. Блок-диаграмма ВП управления параметрами Web-камеры

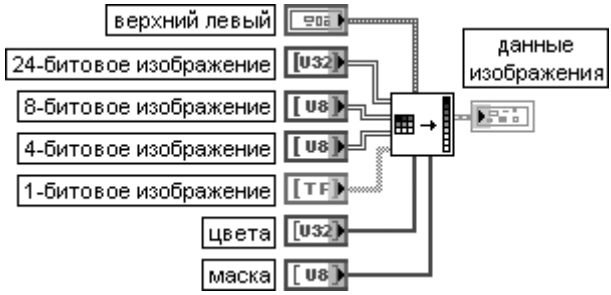
3.3.5. Функции преобразования и отображения графических файлов

Функции преобразования и отображения графических файлов позволяют записывать и читать файлы в формате 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).

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), **восьмибитовое изображение** (8-bit pixmap), **четырехбитовое изображение** (4-bit pixmap) и **однобитовое 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 **восьмибитового изображения** и **четырехбитового изображения** ВП использует их данные как индексы массива цветов. При преобразовании **однобитового 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 **восьмибитового изображения** массив может иметь 256 элементов. При подключении **четырехбитового изображения** массив может иметь 16 элементов. В случае подключения **однобитового 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 параметры:

тип изображения (image type) – зарезервирован для последующих приложений;

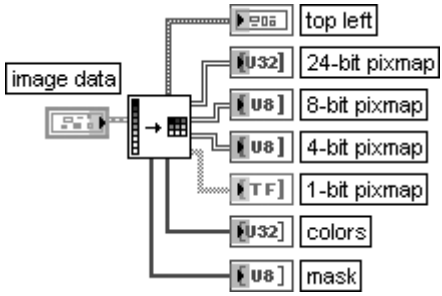
глубина изображения (image depth) – задает глубину цвета изображения, которая определяется числом битов, используемых для описания цвета каждого элемента изображения. Допустимые значения включают 1, 4, 8, и 24 бита на элемент. **Глубина изображения** влияет на то, как LabVIEW интерпретирует значения массивов **изображение** (image) и **цвета** (colors);

изображение (image) – представляет массив байтов, которые описывают цвет каждого элемента растрового изображения.

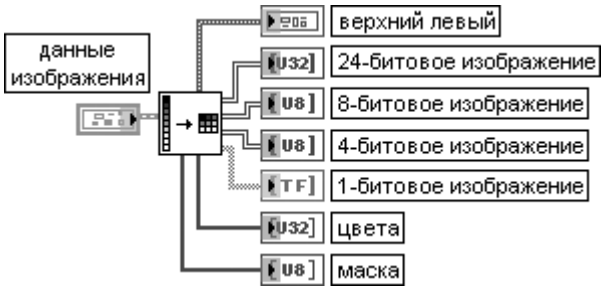
Если **глубина изображения** равна 24, то цвет каждого элемента описывается с помощью трех байтов. Первый байт каждого элемента описывает величину красного цвета, второй байт – зеленого, третий байт – синего.

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, 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 **изображения** соответствуют нулевому элементу в массиве **цвета**. 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 первых восьми элементов, второй байт – вторых восьми 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

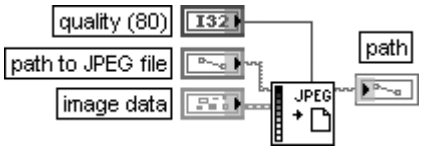


Восстановить изображение из кластера



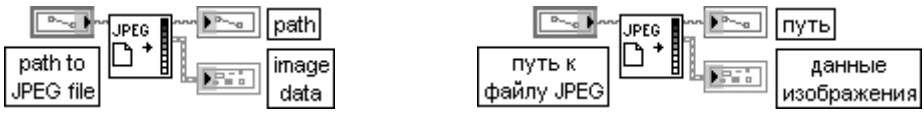
ВП преобразует кластер **данных изображения** в двумерный массив.
Данный ВП целесообразно использовать на выходе одного из ВП подпалитры **Графические форматы** (Graphics Formats), выполняющих чтение графических файлов, для преобразования данных изображения в двумерное представление.
Если на вход ВП 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**



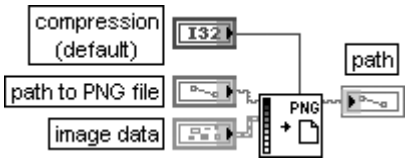
ВП производит запись **данных изображения** (image data) в файл в формате JPEG. При записи изображения в файл необходимо использовать ВП **Перевести массив данных в изображение** (Flatten Pixmap) для преобразования данных в кластер **данных изображения** перед использованием этого ВП. При записи рисунка 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 JPEG File)

Write PNG File **Записать в файл PNG**

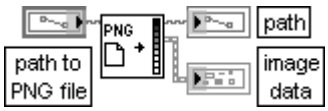


ВП производит запись файла в формате PNG.
Вход **сжатие** (compression) определяет требуемый уровень сжатия файла PNG. Допустимые значения изменяются в диапазоне от –1 до 9, обеспечивая баланс между сжатием файла и скоростью. Примеры значений **сжатия** приведены в таблице.

–1	Хорошее сжатие и скорость (по умолчанию)
0	Без сжатия
1	Наилучшая скорость со сжатием
9	Наилучшее сжатие

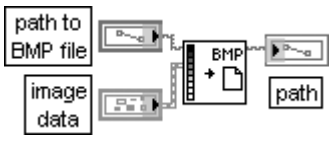
Вход **путь к файлу PNG** (path to PNG file) определяет путь и имя файла PNG, в который производится запись. Если путь не задан, LabVIEW отображает окно файлового диалога, с помощью которого пользователь может указать путь к файлу.
Вход **данные изображения** (image data) описывает изображение, которое должно быть записано в файл PNG

Read PNG File **Читать файл PNG**



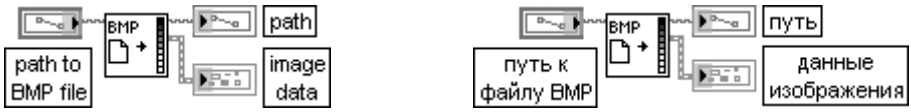
ВП читает файл PNG и формирует данные изображения, необходимые для отображения файла на индикаторе рисунка.
Функции входа **путь к файлу PNG** (path to PNG file) и выходов **путь** и **данные изображения** (image data) идентичны функциям одноименного входа и одноименных выходов рассмотренного выше ВП **Читать файл JPEG** (Read JPEG File)

Write BMP File **Записать в файл BMP**



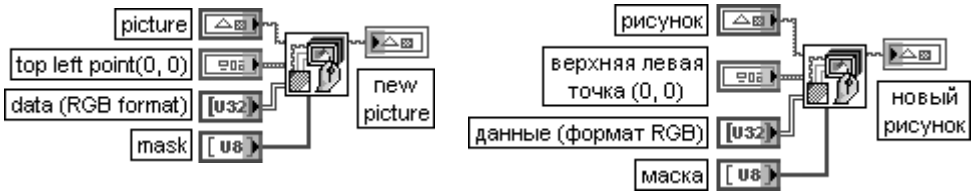
ВП производит запись файла в формате BMP.
Функции входов **путь к файлу BMP** (path to BMP file) и **данные изображения** (image data), а также выхода **путь** идентичны функциям одноименных входов и выхода рассмотренного выше ВП **Записать в файл JPEG** (Write JPEG File)

Read BMP File **Читать файл BMP**



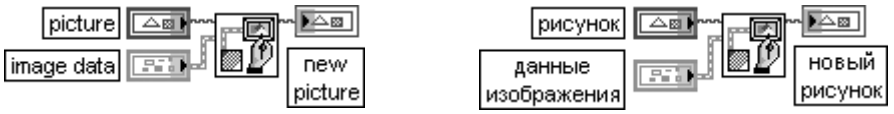
ВП читает файл BMP и формирует данные изображения, необходимые для отображения файла на индикаторе рисунка.
Функции входа **путь к файлу BMP** (path to BMP file) и выходов **путь** и **данные изображения** (image data) идентичны функциям одноименного входа и одноименных выходов рассмотренного выше ВП **Читать файл JPEG** (Read JPEG File)

Draw Unflattened Pixmap **Рисовать восстановленное изображение**



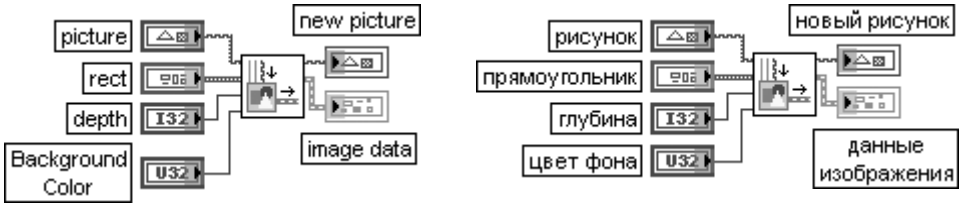
ВП преобразует изображение в рисунок, что позволяет использовать другие ВП из подпалитры **Функции рисунка** (Picture Functions) для добавления инструкций рисования элементов на изображении.
Данный полиморфный ВП позволяет преобразовывать однобитовые, четырехбитовые, восьмибитовые или полноцветные изображения. Тип данных, подключенных к входу **данные,** определяет тип выбираемого ВП из набора ВП, обеспечивающего его полиморфность. При необходимости преобразования четырехбитовых или восьмибитовых 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 растрового изображения. Цвет каждого пиксела описывается с помощью трех байтов. Первый байт каждого пиксела описывает величину красного цвета, второй байт – зеленого, 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**



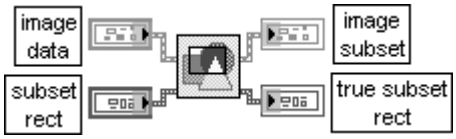
ВП выводит одно-, четырех-, или восьмибитовое изображение или 24-битовое RGB изображение на рисунок. Этот ВП берет одномерный массив байтов, предполагая, что пользователь выполнил все операции упаковки и заполнения

Picture to Pixmap **Рисунок в изображение**

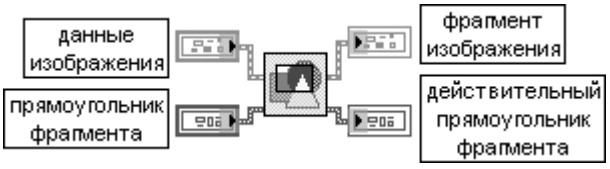


ВП преобразует рисунок в кластер данных изображения, который может быть далее сохранен в файле с помощью ВП из подпалитры **Графические форматы** (Graphics Formats).
Вход **цвет фона** (Background Color) устанавливает фоновый цвет изображения

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, 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.

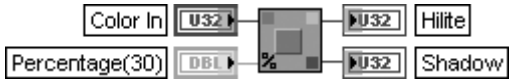
На вход **цвет** (Color) подается преобразуемый цвет. Выход **разделенный цвет** (resolved color) отображает разделяемый цвет в RGB формате.

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 возвращается 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 процентов.

Выход **подсветка** (Hilite) отображает новый цвет, отличающийся большей насыщенностью

Выход **тень** (Shadow) отображает новый цвет, отличающийся меньшей насыщенностью

На рис. 3.64 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.

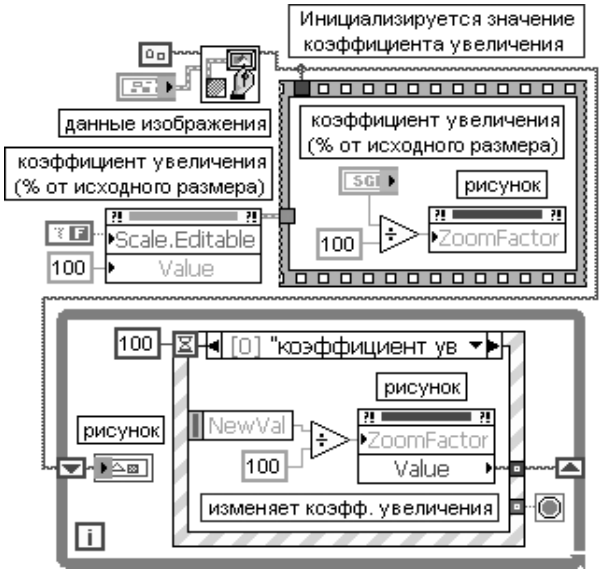
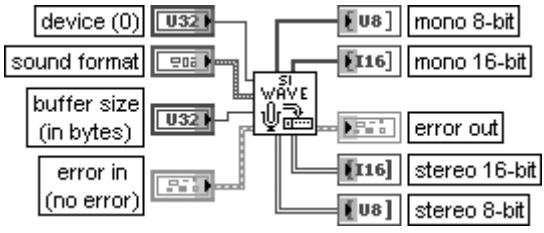


Рис. 3.64. Блок-диаграмма ВП ***Управление рисунком – Атрибут увеличения*** *(Picture Control – Zoom Attribute)*

3.3.6. Функции записи и воспроизведения звуковых сигналов

Функции записи и воспроизведения звуковых сигналов (рис. 3.65) 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, воспроизводить звуковой файл с расширением .wav на выходе звуковой карты. В совокупности функций записи и воспроизведения звуковых сигналов можно выделить функции высокого уровня, размещенные в основной палитре **Звук** (Sound) (рис. 3.65а) и подпалитре **Звуковой файл** (Sound File) (рис. 3.65б), и функции низкого уровня, размещенные в остальных подпалитрах (рис. 3.65в, рис. 3.65г).

Snd Read Waveform



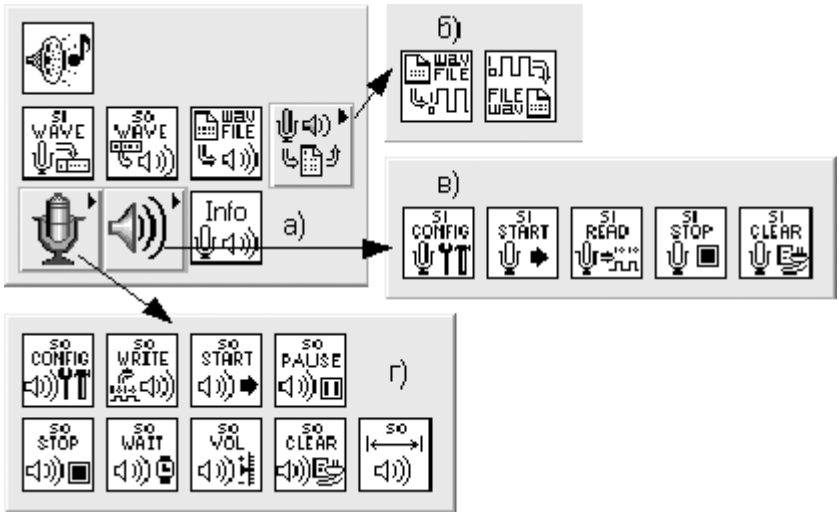
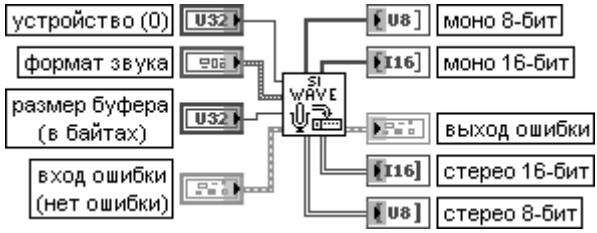


Рис. 3.65. Вид основной палитры (а) и дополнительных подпалитр (б – г) функций записи и воспроизведения звуковых сигналов

Считать звуковую осциллограмму



ВП считывает данные с входа звуковой карты. Размер данных определяется на входе **размер буфера** (buffer size).

Вход **устройство** (device) определяет входное устройство, к которому разрешается доступ для выполнений операций со звуком в операционных системах Windows 2000/NT/XP/Me/9*x*. В общем случае большинство пользователей могут выбрать значение по умолчанию, равное 0.

Вход **формат звука** (sound format) определяет такие установки звуковых операций, как режим моно или стерео, частоту дискретизации и размер отсчета – 8 или 16 бит.

качество звука (sound quality) устанавливает звуковые операции как моно или стерео

частота (rate) устанавливает частоту дискретизации входных звуковых операций или корректирует частоту для выходных операций. Предусмотрена возможность выбора частоты равной 8000, 11025, 22050 или 44100 Гц

бит на выборку (bits per sample) устанавливает звуковые операции для восьмибитовых или 16-битовых выборок

Вход **размер буфера** (buffer size) задает размер внутреннего буфера, который LabVIEW использует для передачи данных от устройства. По умолчанию размер буфера равен 8192 байта.
В состав блок-диаграммы ВП **Считать звуковую осциллограмму** (Snd Read Waveform) (рис. 3.66) входят ВП **Конфигурировать входное звуковое устройство** (SI Config), **Начать накопление данных входного звукового устройства** (SI Start), **Считать данные входного звукового устройства** (SI Read) и **Очистить входное звуковое устройство** (SI Clear).

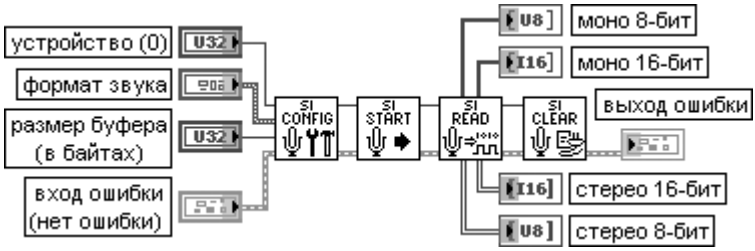
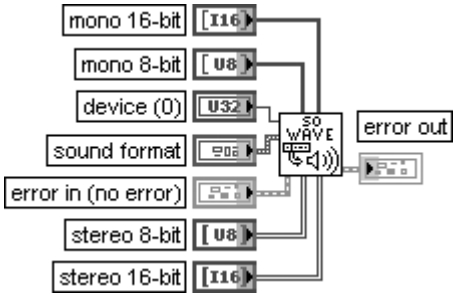
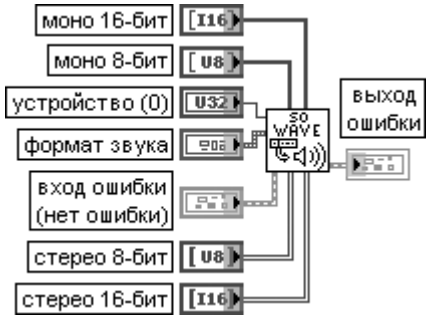


*Рис. 3.66. Блок-диаграмма ВП **Считать звуковую осциллограмму** (Snd Read Waveform)*

Snd Write Waveform



Записать звуковую осциллограмму



ВП выводит массив данных на выход звуковой карты и ожидает завершения воспроизведения данных.
Функциональное назначение входов ВП соответствует назначению входов или одноименных выходов рассмотренного выше ВП **Считать звуковую осциллограмму**. Блок-диаграмма ВП

Записать звуковую осциллограмму приведена на рис. 3.67 и включает ВП **Конфигурировать выходное звуковое устройство** (SO Config), **Записать данные в выходное звуковое устройство** (SO Write), **Начать вывод данных выходного звукового устройства** (SO Start), **Ожидать завершение вывода данных** (SO Wait) и **Очистить выходное звуковое устройство** (SO Clear).

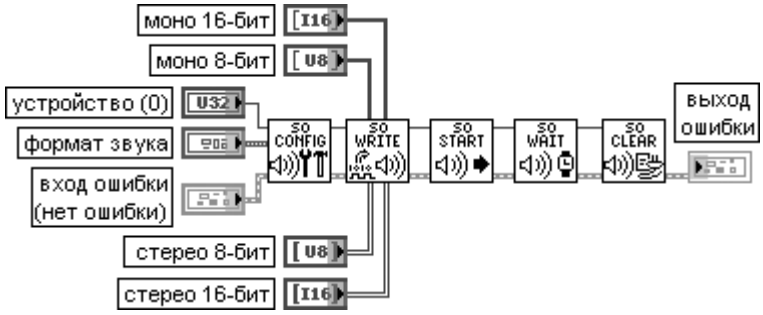
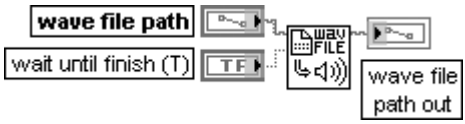
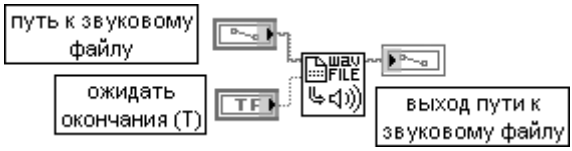


*Рис. 3.67. Блок-диаграмма ВП **Записать звуковую осциллограмму** (Snd Write Waveform)*

Snd Play Wave File

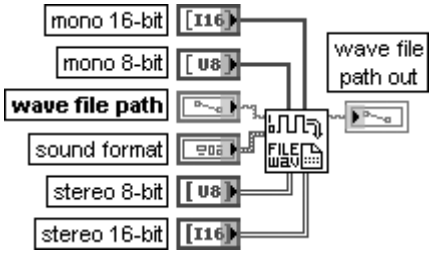


Воспроизвести звуковой файл

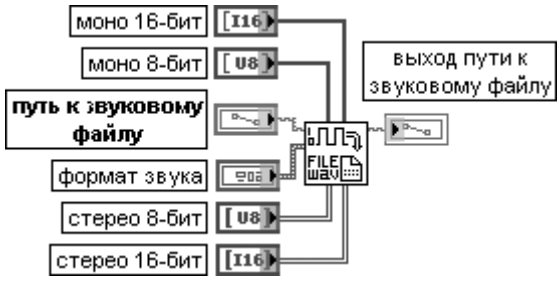


ВП воспроизводит звуковой файл с диска. Если вход **ожидать окончания** (wait until finish) установлен в состояние ИСТИНА, то ВП ожидает завершения воспроизведения. В противном случае выполнение ВП продолжается

Snd Write Wave File

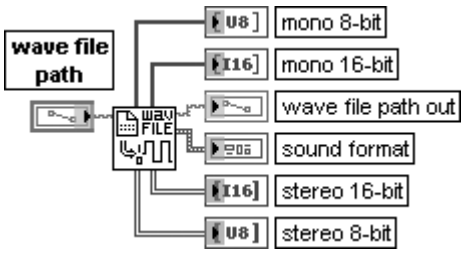


Записать звуковой файл

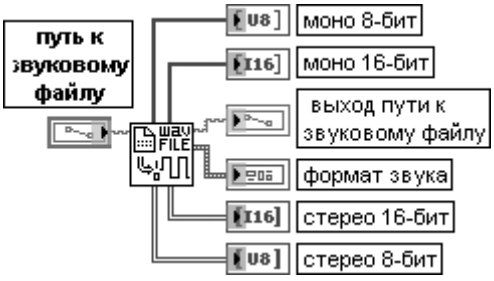


ВП сохраняет данные сигнала и информацию о формате звука в звуковом файле. Файл имеет расширение .wav и может быть считан с помощью ВП **Считать звуковой файл** (Snd Read Wave File) или с помощью других приложений

Snd Read Wave File

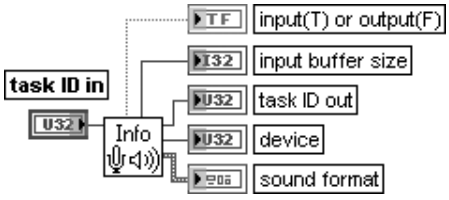


Считать звуковой файл

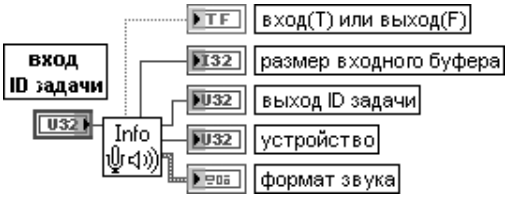


ВП извлекает звуковой файл с расширением .wav, путь к которому задан на входе **путь к звуковому файлу** (wave file path). Возвращаемая информация включает как данные сигнала, так и данные формата звука, необходимые для конфигурирования звуковой карты с целью воспроизведения сигнала

Get Sound Info



Получить информацию о звуковой операции



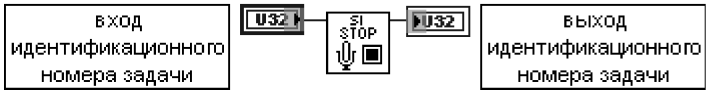
ВП возвращает информацию, связанную со звуковой операцией, заданной **идентификационным номером задачи**.

Вход **идентификационный номер задачи** (task ID in) определяет операцию со звуком, которую необходимо выполнить с помощью сконфигурированного устройства. Идентификационный номер задачи формируется на выходе ВП **Конфигурировать входное звуковое устройство** или **Конфигурировать выходное звуковое устройство**.

Выход **вход или выход** (input or output) отображает принадлежность операции, определяемой идентификационным номером задачи, к типу входных или выходных операций

SI Stop

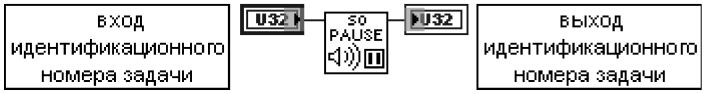
Остановить входное звуковое устройство



ВП выполняет остановку накопления данных во входном звуковом устройстве

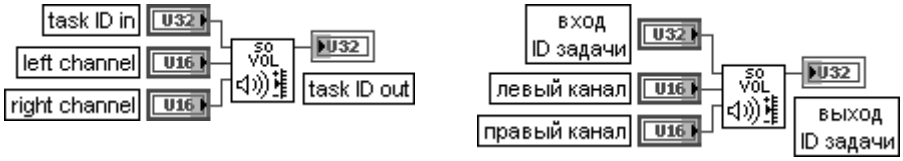
SO Pause

Установить паузу в работе выходного звукового устройства



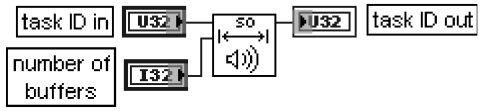
ВП устанавливает паузу в работе выходного звукового устройства, заданного **идентификационным номером задачи** (task ID in). Для повторного начала вывода необходимо использовать ВП **Начать вывод данных выходного звукового устройства** (SO Start)

SO Volume **Громкость выходного звукового устройства**

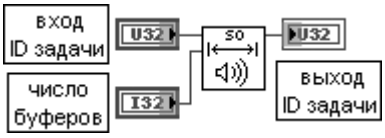


ВП устанавливает громкость на выходе звукового устройства. Громкость задается с помощью 16-битовых целых чисел без знака в **левом канале** (left channel) и **правом канале** (right channel), где 0 соответствует минимальной, а 65 535 – максимальной громкости

SO Set Num Buffers



Установить число выходных буферов



ВП устанавливает число выходных буферов, связанных с **входным идентификационным номером задачи** (task ID in)

Ниже на рис. 3.68 и 3.69 в качестве примеров использования функций записи и воспроизведения звуковых сигналов приведены блок-диаграммы ВП **Непрерывный вывод звука** (Continuous Sound Output) и **Спектр мощности звуковой карты** (Sound Card AutoPower Spectrum) из библиотеки примеров NI Example Finger. В первом ВП установка числа буферов больше нуля позволяет устранить паузы при циклическом воспроизведении звуковых колебаний. Вместе с тем установка числа буферов больше единицы приводит к большой задержке между генерацией и воспроизведением таких колебаний.

*Рис. 3.68. Блок-диаграмма ВП **Непрерывный вывод звука** (Continuous Sound Output)*

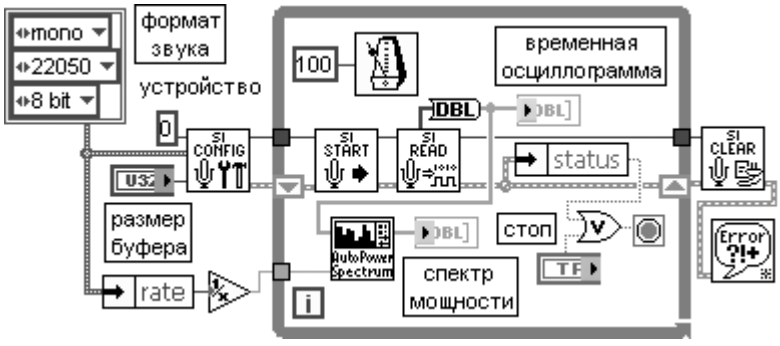
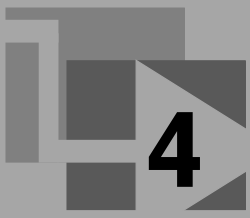


*Рис. 3.69. Блок-диаграмма ВП **Спектр мощности звуковой карты** (Sound Card AutoPower Spectrum)*

Функции интерфейса ВП и приложений в LabVIEW



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. Функции из палитры **Коммуникация** (Communication) открывают доступ к свойствам и методам самой среды LabVIEW и отдельных ВП другим приложениям Windows с помощью технологии ActiveX. С помощью этой же технологии ВП могут получить доступ к свойствам и методам других приложений Windows, таким как Excel и Word. В первом случае LabVIEW и ВП выполняют функции **сервера**, во втором – **клиента**. LabVIEW может использоваться и как клиент .NET для доступа к объектам, свойствам и методам, связанным с серверами .NET. Наряду с перечисленными технологиями для обмена данными между локальными и удаленными приложениями и ВП могут использоваться такие протоколы низкого уровня, как DataSocket, SMTP Email, TCP/IP и UDP. При этом в качестве сервера может выступать как специальная программа-сервер, так и ВП, настроенный соответствующим образом. В зависимости от конкретной технологии взаимодействия ВП и приложений можно указать следующие варианты реализации серверов и клиентов.

1. Использование ВП в качестве сервера (рис. 4.1). Для обращения к серверу может использоваться технология ActiveX, вызов ВП по ссылке с помощью функций палитры **Управление приложением**, а также протокол TCP/IP. Для использования в качестве сервера ВП должен быть настроен 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. Функции сервера ВП из палитры **Управление приложением** рассмотрены далее в разделе 4.1, а функции ActiveX – в разделе 4.2.1.

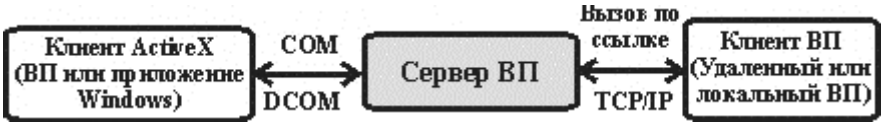


Рис. 4.1. Схема взаимодействия клиентов и сервера ВП

2. Использование технологии и сервера DataSocket (рис. 4.2). Технология и функции DataSocket рассмотрены в разделе 4.2.3.

Рис. 4.2. Схема взаимодействия клиентов и сервера DataSocket

3. Использование технологии и сервера .NET (рис. 4.3). Технология и функции .NET рассмотрены в разделе 4.2.2.

Рис. 4.3. Схема взаимодействия клиентов и сервера .NET

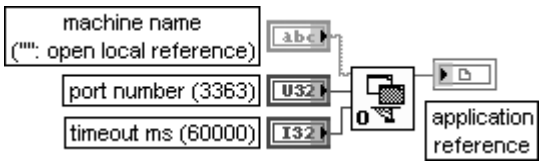
4. Использование встроенного Web-сервера или Интернет-сервера Internet Toolkit's server.

4.1. Функции управления приложением

Функции управления приложением (рис. 4.4) позволяют программно с помощью сервера ВП управлять свойствами приложения (LabVIEW), виртуального прибора, элементов управления и индикации. Под управлением понимается динамическое изменение свойств (атрибутов) ВП или объектов лицевой панели, динамический вызов и загрузка ВП в память с последующим запуском и закрытием.

Описание функций управления приложением приведено в следующих таблицах.

Open Application Reference



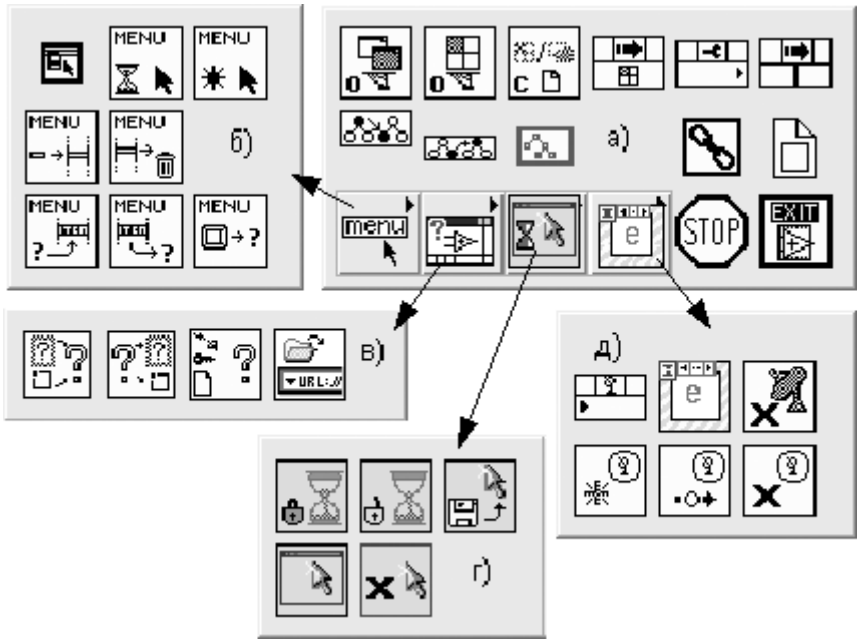


Рис. 4.4. Вид основной палитры (а) и подпалитр (б–д) функций управления приложением

Открыть ссылку на приложение

Функция возвращает ссылку на приложение сервера ВП, работающее на заданном компьютере.

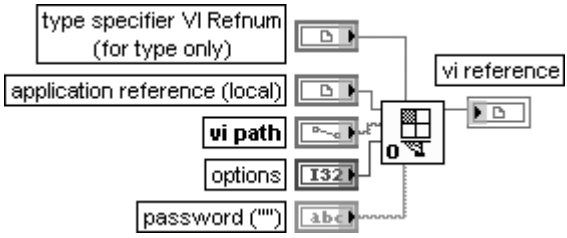
Вход **имя компьютера** (machine name) задает адрес компьютера, работающего в среде LabVIEW, с которой необходимо установить соединение. Если на входе **имя компьютера** (machine name) будет подключена пустая строка, то функция возвратит ссылку на локальную среду LabVIEW, в которой она выполняется. Если 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 выполнение 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.

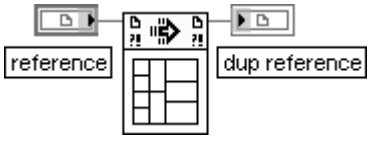
Вход **опции** (options) представляет набор битов, которые определяют способ обработки ссылки на ВП. Предусмотрены следующие комбинации битов на входе **опции**:

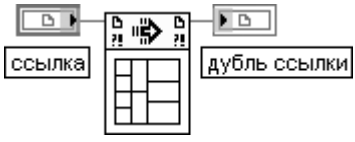
0x01	Запись модификаций. Звездочка (*) появляется в названии ВП для индикации того, что изменения были сделаны с помощью сервера ВП. ВП должен находиться в режиме редактирования, чтобы разрешить LabVIEW записать модификации
0x02	Открытие шаблонов для редактирования. Эта опция действует только на шаблонные 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 таким же образом, как и к терминалам подприбора, могут быть подключены элементы управления и индикаторы.

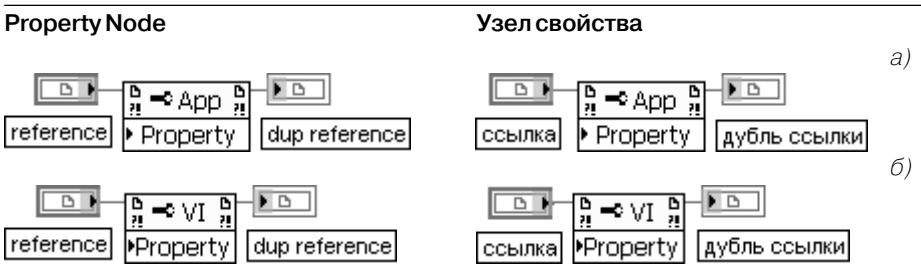


*Рис. 4.5. Варианты конфигурирования **Узла свойства***

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. 4.5а показан вид узла при выборе класса **Приложение** (Application), а на рис. 4.5б – 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.
Перечень свойств класса **Приложение** приведен в приложении 4, а аналогичный перечень для класса **ВП** – в приложении 5. В приложении 8 приведены характеристики 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 выбрать строку **Игнорировать ошибки внутри узла** (Ignore Errors Inside Node) контекстного меню

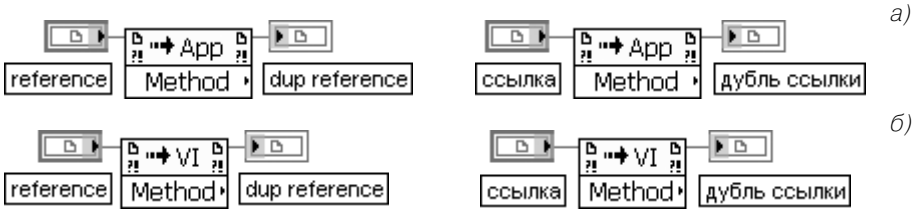
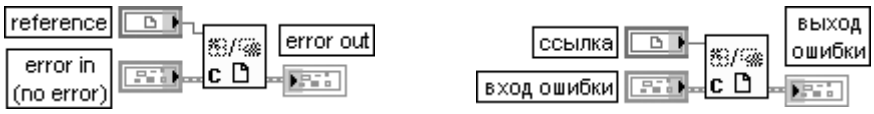


*Рис. 4.6. Варианты конфигурирования **Узла вызова***

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. 4.6 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**.
Перечень методов класса **Приложение** приведен в приложении 6, а класса **ВП** – в приложении 7. В приложении 8 приведены характеристики методов.
Узел позволяет получать (читать) и/или устанавливать (записывать) параметры методов. Параметры с белым фоном являются необходимыми входами, а параметры с серым фоном являются 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**

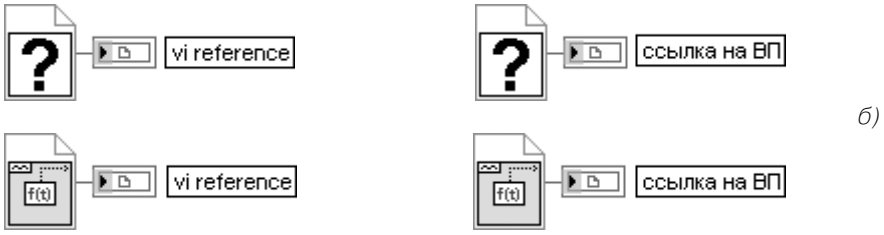


*Рис. 4.7. Варианты конфигурирования функции **Статическая ссылка на ВП***

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. 4.7 показан вид функции непосредственно после вызова (а) и после установки ссылки с помощью строки Browse for Path контекстного меню (б)

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 к 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. 4.8 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** производится динамическая загрузка и запуск ВП **Создать набор данных** (Create Data Set), который формирует на выходе два одномерных массива данных. После выполнения ВП с помощью функции **Закрыть ссылку** он выгружается из памяти.

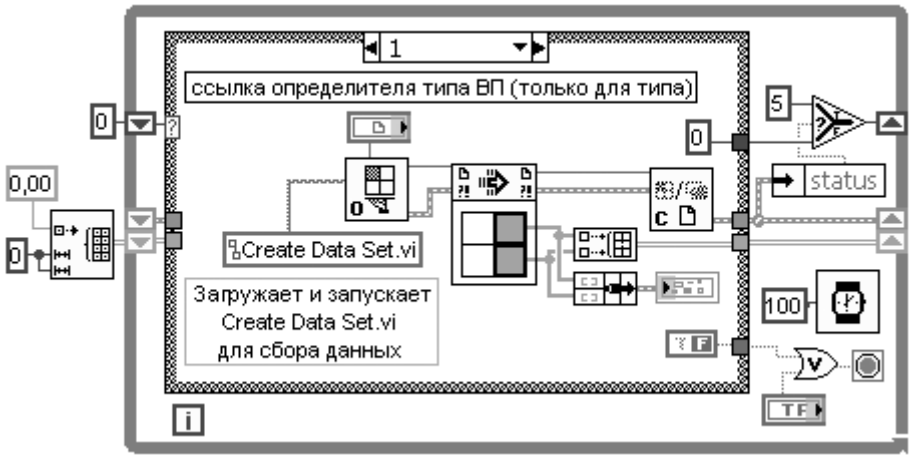
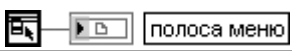


*Рис. 4.8. Блок-диаграмма ВП **Пример динамической загрузки***

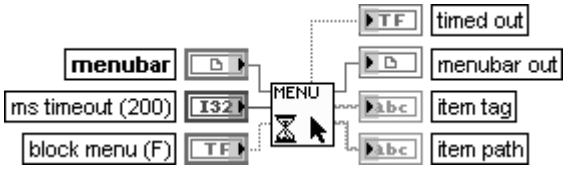
Использование функций подпалитры **Меню** (рис. 4.4б) позволяет модифицировать меню приложений LabVIEW. Функции, расположенные в верхнем ряду этой подпалитры, позволяют обрабатывать обращения к меню.

Current VI's Menubar **Полоса меню текущего ВП**

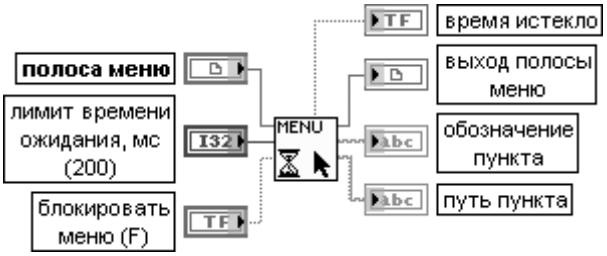


Функция возвращает ссылку на **полосу меню** (menubar) текущего ВП. Ссылка на **полосу меню** необходима для использования других функций подпалитры **Меню**

Get Menu Selection



Получить выбор меню



Функция возвращает обозначение последнего выбранного пункта меню после ожидания в течение 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 он равен 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.

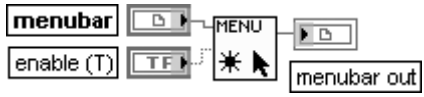
Отображение значения ИСТИНА на выходе **время истекло** (timed out) означает, что выбор меню не был произведен пользователем за время, заданное на входе **лимит времени ожидания**.

Выход полосы меню возвращает неизменное значение одноименного входа.

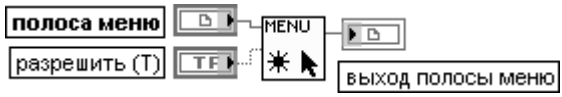
Выход **обозначение пункта** (item tag) содержит выбранный пункт меню. Для обработки каждого пункта меню следует соединить этот выход с терминалом выбора структуры **Вариант**. При создании структуры **Вариант** для обработки каждого пункта меню необходимо ввести обозначения пунктов приложения в ярлыке 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



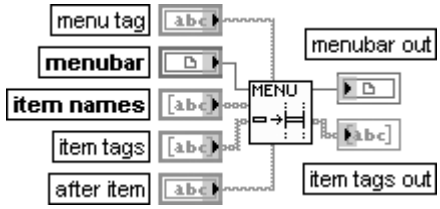
Разрешить отслеживание меню



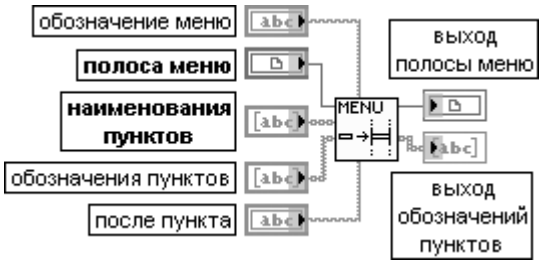
Функция разрешает или запрещает отслеживание выбора меню. Если функция **Получить выбор меню** (Get Menu Selection) используется для блокирования меню, то данная функция должна использоваться для разрешения меню.

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).

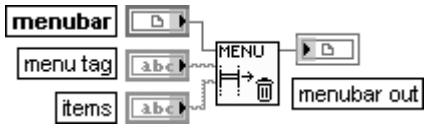
Вход **наименования пунктов** (item names) идентифицирует пункты, которые необходимо вставить в меню. **Наименования пунктов** является строкой, которая появляется в меню. Можно подключить входы **наименования пунктов** или **обозначения пунктов**,

в этом случае как имена, так и обозначения имеют одни и те же значения. Если вставляется только один пункт, то необходимо подключить строку к входу **наименования пунктов**.

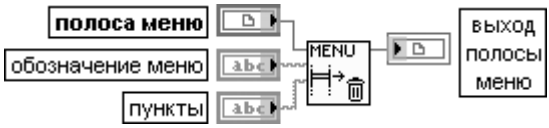
Вход **обозначения пунктов** (item tags) идентифицирует пункты, которые необходимо вставить в меню. **Обозначения пунктов** представляют строку, которая возвращается при выборе пункта меню. Можно подключить входы **наименования пунктов** или **обозначения пунктов**, в этом случае как имена, так и обозначения имеют одни и те же значения. Если вставляется только один пункт, то необходимо подключить строку к входу **обозначения пунктов**.

Для вставки пунктов меню приложения необходимо использовать **обозначения пунктов приложения** (application item tags)

Delete Menu Items

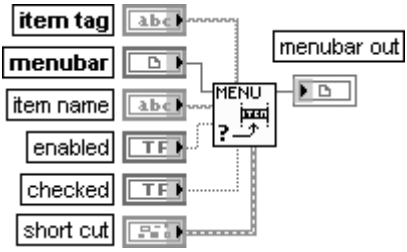


Удалить пункты меню



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).

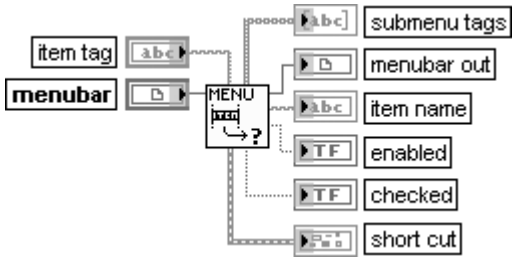
Если на вход **разрешено** (enabled) подается значение ЛОЖЬ, то пункту меню придается серый цвет, свидетельствующий о его отключении.

Если на входе **отмечено** (checked) установлено значение ИСТИНА, то пункт меню имеет отметку, расположенную перед ним.

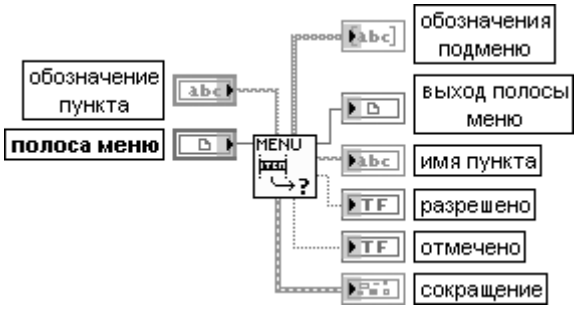
На входе **сокращение** (short cut) задается клавишный эквивалент выбора пункта меню с помощью мыши. **Сокращение** обычно является комбинацией клавиши меню с одной из других клавиш. В Windows клавишей меню является клавиша <Ctrl>. Дополнительно может использоваться клавиша <Shift>. В состав кластера **сокращение** входят следующие элементы.

DBL	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)
DBL	Строковый элемент управления **«горячая» клавиша** (shortcut key) описывает клавишу, связанную с **сокращением**

Get Menu Item Info



Получить информацию о пункте меню

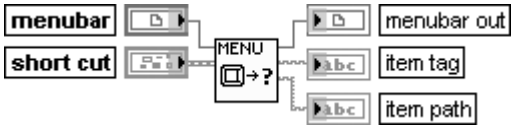


Функция возвращает атрибуты пункта меню или полосы меню.

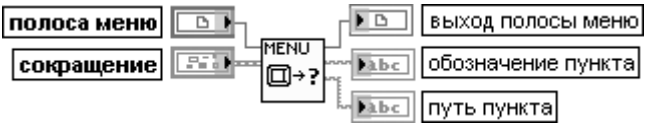
Выход **обозначения подменю** (submenu tags) содержит обозначения пунктов подменю, если пункт имеет подменю.

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. 4.9 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.

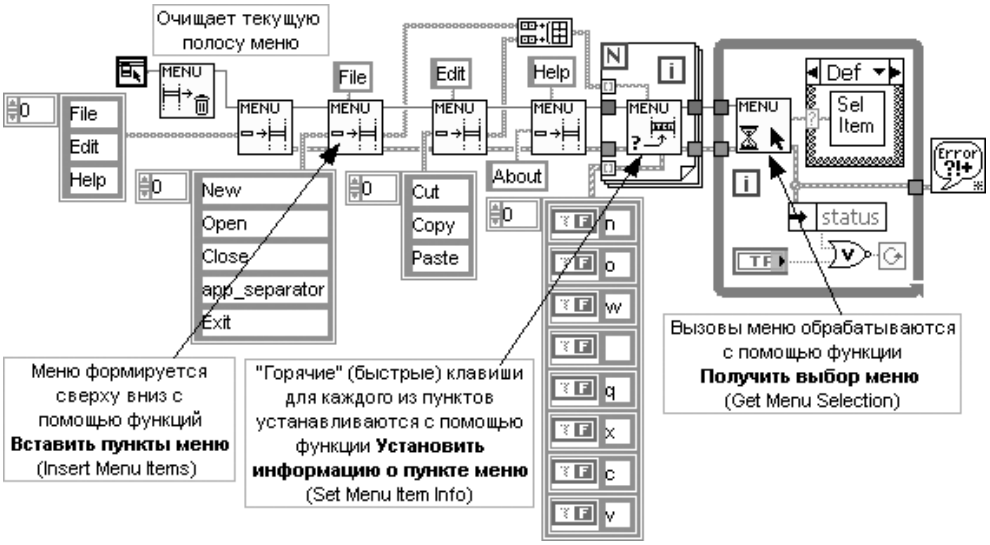


Рис. 4.9. Блок-диаграмма модернизированного ВП ***Демонстрация динамической вставки***

Использование ВП из подпалитры **Курсор** (рис. 4.4г) 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 **Сбросить занятость** (Unset Busy). Использование ВП **Установить занятость** аналогично использованию ВП **Установить курсор** (Set Cursor) при подключении 1 к 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.

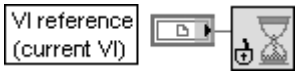
Вход **ссылка на ВП** (VI reference) определяет ссылку на ВП, для которого производится изменение курсора. По умолчанию это ссылка на ВП, который содержит данный ВП как подприбор. Для получения ссылки на другой ВП можно использовать функцию **Открыть ссылку на ВП** (Open VI Reference).

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 **Сбросить занятость** (Unset Busy).

При блокировании мыши на лицевой панели остается возможность нажатия кнопки **Прервать выполнение** (Abort Execution) в инструментальной панели

Unset Busy **Сбросить занятость**



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.

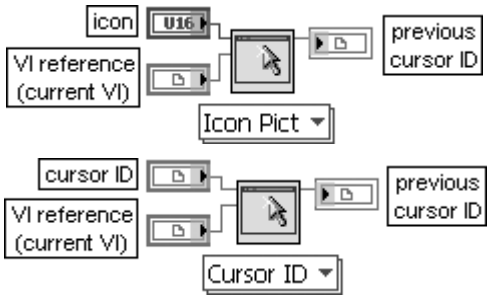
Вход **файл курсора** (cursor file) передает путь к файлу, который содержит курсор. Если файл не существует или файл не является файлом курсора, то LabVIEW возвращает код ошибки 7. В Windows файл должен иметь расширение .ani или .cur.

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».

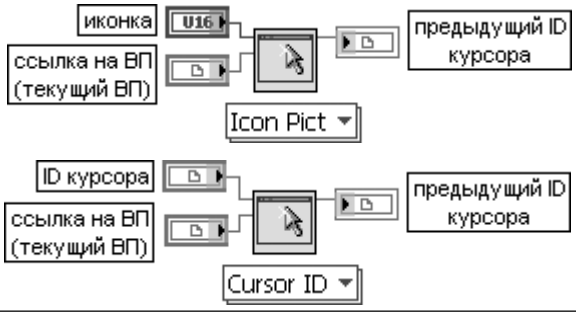
Выход **ID курсора** (cursor ID) представляет ссылку на курсор, содержащийся в **файле курсора**. Если ссылка на курсор уже существует, то ВП возвращает существующую ссылку. При необходимости изменения вида курсора на лицевой панели следует подключить эту ссылку к ВП **Установить курсор** или **Установить занятость**.

Для закрытия ссылки на курсор после его установки необходимо использовать ВП **Уничтожить 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 к 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. 4.10.

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**

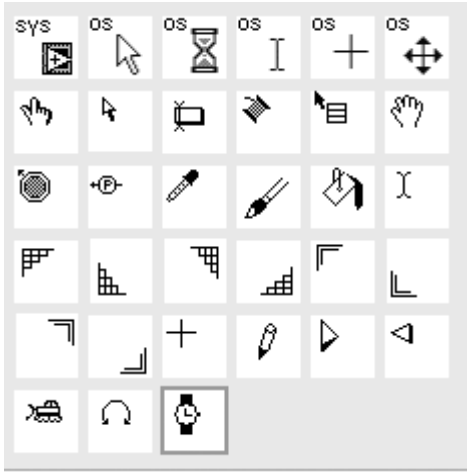


Рис. 4.10. Вид палитры иконок курсора

Destroy Cursor **Уничтожить 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. 4.11 приведена блок-диаграмма модернизированного ВП **Изменить иконку курсора** (Change Cursor Icon) из библиотеки CursorUtilities.llb набора примеров NI Example Finder.

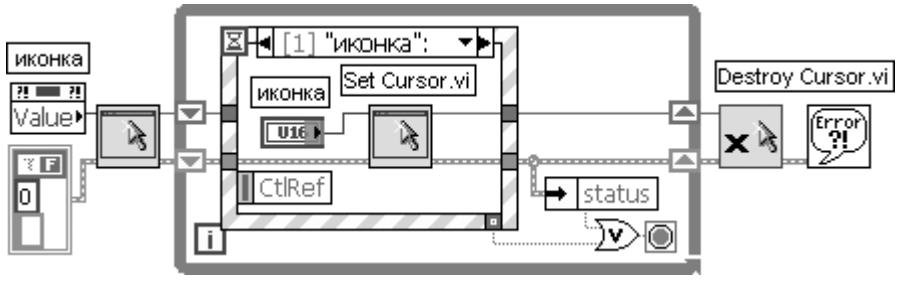
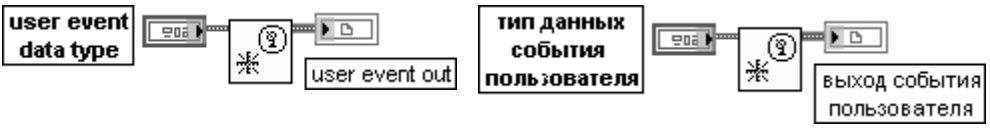


*Рис. 4.11. Блок-диаграмма модернизированного ВП **Изменить иконку курсора***

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.

Использование функций и узлов подпалитры **Событие** (рис. 4.4д) позволяет динамически регистрировать события и создавать события пользователя.

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 типа кластера как имя события пользователя. Имена и типы данных элементов кластера определяют данные события, переносимые событием пользователя. При подключении к входу отдельного элемента данных LabVIEW использует имя типа элемента как имя события пользователя и также 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 события** (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

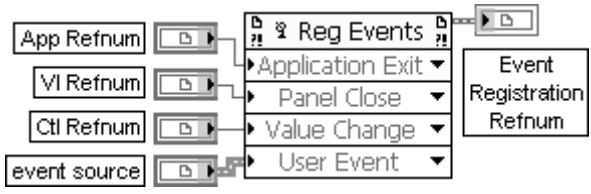
Unregister For Events **Отменить регистрацию событий**

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

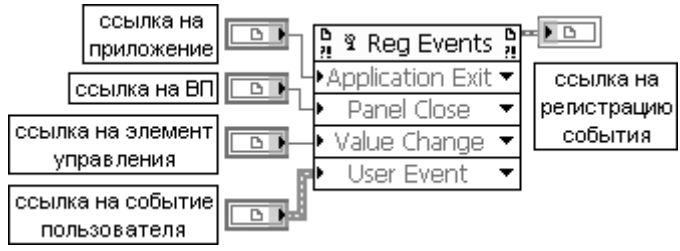
Destroy User Event **Уничтожить событие пользователя**

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

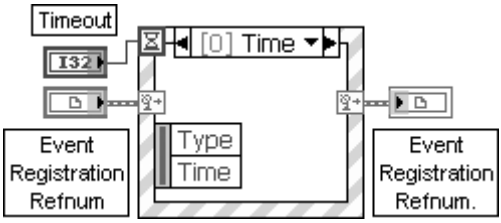


Регистрация событий

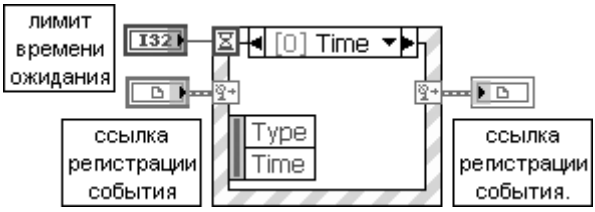


Функция динамически регистрирует события. События, которые можно регистрировать, зависят от типа ссылки, которая подключена к каждому входу **источник события** (event source). Выход **ссылка на регистрацию события** (event reg refnum out) следует подключать к структуре **Событие** или к другом узлу **Регистрация событий**.
Вход **ссылка регистрации 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 подключен 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 общая версия 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 лимита 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. 4.12 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.

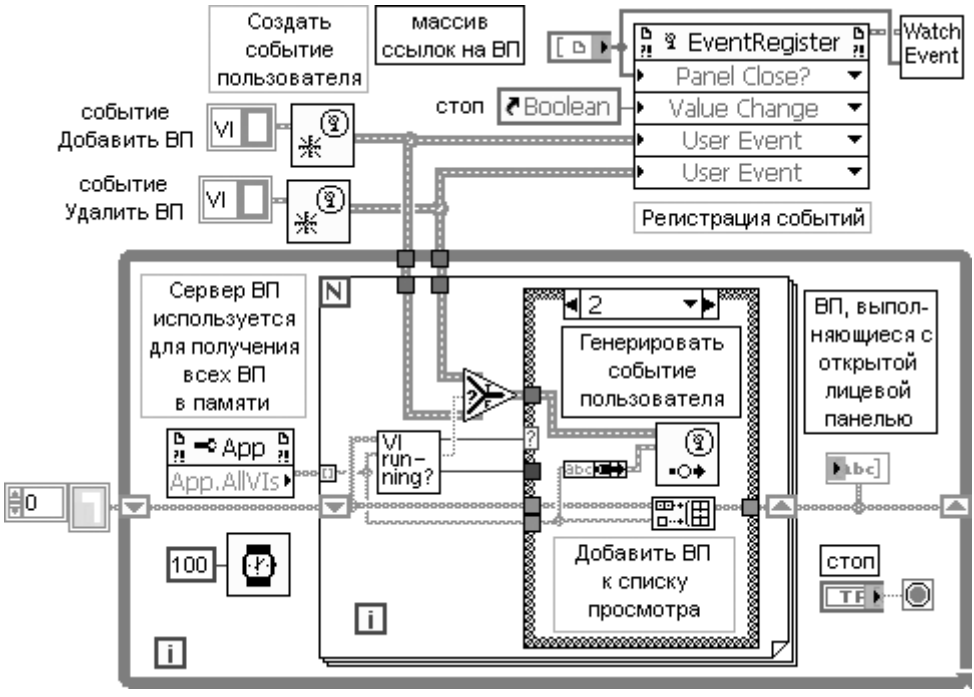


Рис. 4.12. Блок-диаграмма модернизированного ВП ***Динамическая проверка ВП***

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 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. 4.13) проверяется на совпадение со списком ВП, выполняющихся на компьютере с открытой лицевой панелью (рис. 4.12). В зависимости от результата сравнения имя ВП может добавляться к списку ВП, удаляться из него, или же список ВП может оставаться неизменным. На рис. 4.12 показан вариант добавления имени к списку ВП.

Подприбор **Выполняющиеся с открытой панелью** (рис. 4.13) 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).

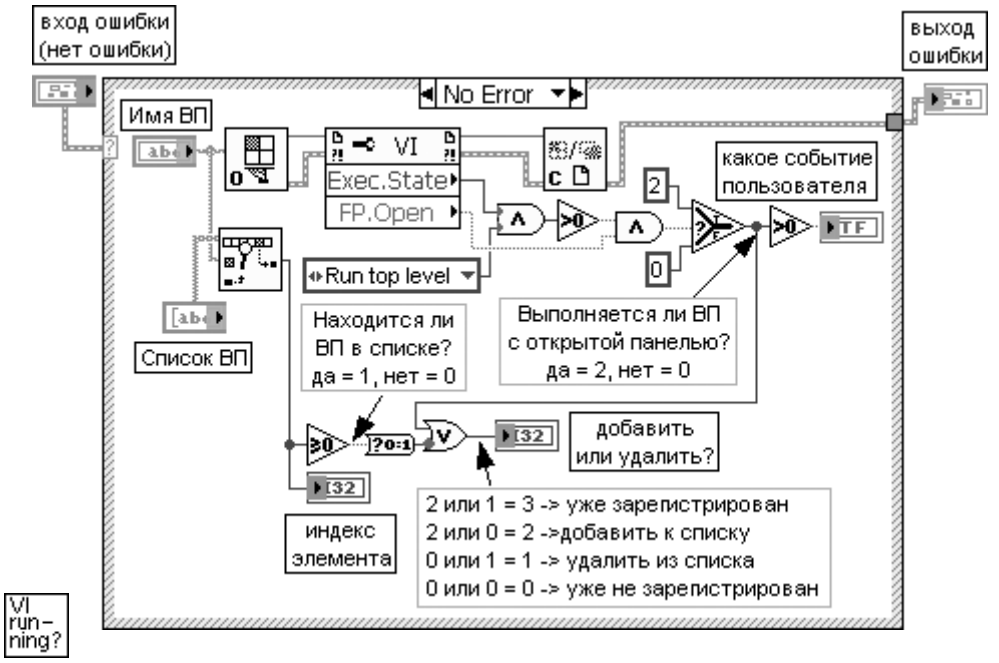


Рис. 4.13. Иконка и блок-диаграмма модернизированного подприбора ***Выполняющиеся с открытой панелью***

Одновременно с добавлением имени ВП к списку имен на блок-диаграмме ВП **Динамическая проверка ВП** (рис. 4.12) производится генерация события функ-

цией **Генерировать событие пользователя**. Генерируемое событие пользователя передается структуре **Событие**, находящейся в подприборе **Ожидать событий** (Wait for Events) (рис. 4.14) 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**.

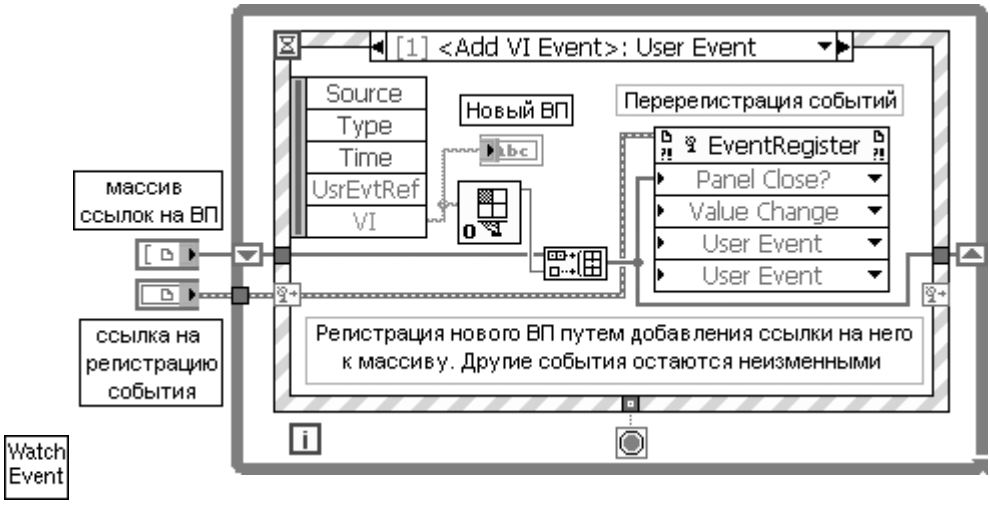
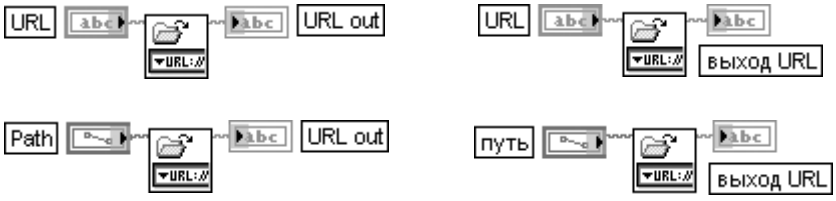


Рис. 4.14. Иконка и блок-диаграмма модернизированного подприбора ***Ожидать событий***

Из подпалитры **Помощь** (Help) (рис. 4.4в) ниже рассмотрена только функция **Открыть URL в браузере по умолчанию** (Open URL in Default Browser).

Open URL in Default Browser **Открыть URL в браузере по умолчанию**



Функция отображает URL или файл HTML в Web-браузере по умолчанию. Тип данных, подключенных к входу **URL**, определяет используемый экземпляр этого полиморфного ВП. 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

4.2. Функции коммуникации среды LabVIEW

Функции коммуникации среды LabVIEW расположены в разделе Communication палитры функций (рис. 4.15а) и включают функции обмена данными между приложениями с использованием технологии .NET (рис. 4.15б) и ActiveX (рис. 4.15в), а также функции сетевого обмена данными по протоколам DataSocket (рис. 4.15г), SMPT E-mail (рис. 4.15д), TCP (рис. 4.15е), UDP (рис. 4.15ж), IrDA (рис. 4.15е) и System Exec. Первой ниже рассмотрена группа функций технологии обмена данными ActiveX.

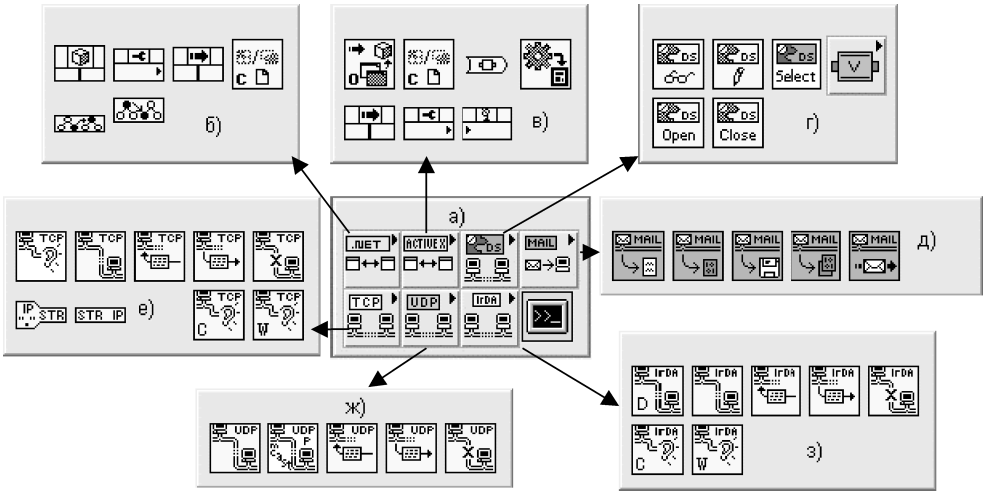


Рис. 4.15. Вид основной палитры (а) и подпалитр (б–з) функций коммуникаций

4.2.1. 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 часть программного обеспечения предоставляет свои сервисы другой. В COM любая часть программного обеспечения реализует свои сервисы как один или несколько **объектов COM** (рис. 4.16). 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.

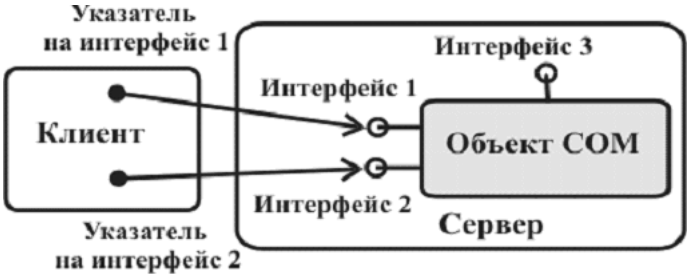


Рис. 4.16. Схема взаимодействия клиента и сервера в технологии обмена ActiveX

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 с технологией ActiveX можно ознакомиться с помощью книг [15, 16].

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.

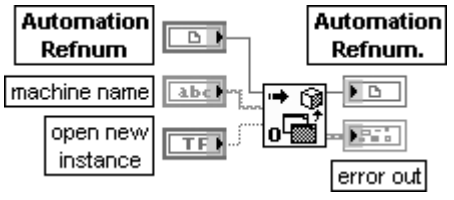
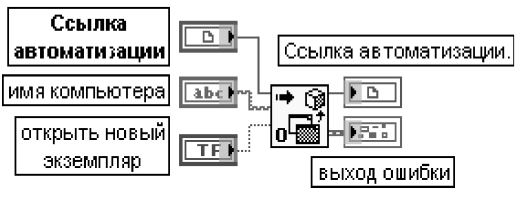
Объекты ActiveX могут быть видимыми для пользователя, такими, например, как кнопки, окна, картины, документы и диалоговые окна, или невидимыми, такими как реестровые объекты приложения. В LabVIEW отображаемые объекты ActiveX, с помощью которых формируется пользовательский интерфейс, устанавливаются на 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.

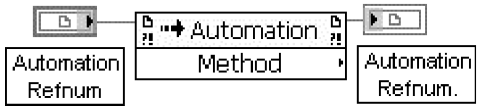
Еще одним элементом ActiveX являются **события** (events), которые включают такие действия над объектами, как щелчок мыши, нажатие клавиши или работа за пределами памяти. Совершение этих действий с объектом вызывает посылку объектом соответствующего извещения и данных в контейнер ActiveX.

Перечень функций и узлов технологии ActiveX рассмотрен в следующей таблице.

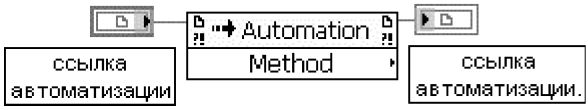
Automation Open	Открыть автоматизацию
	

Функция возвращает ссылку автоматизации, которая указывает на определенный объект ActiveX.
Вход **ссылка автоматизации** (A utomation 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 типов** (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

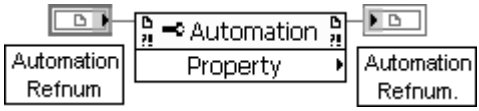


Узел вызова

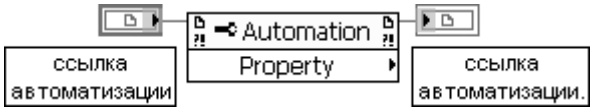


Узел вызывает метод или действие по ссылке. Большинство методов имеют связанные с ними параметры.
Вход **ссылка** (reference) является ссылкой, связанной с объектом, который вызывает метод или выполняет действие. Ссылка может быть получена от функции **Открыть автоматизацию** (Automation Open) или от элементов управления **ссылка** (Refnum) лицевой панели. **Узел вызова** (Invoke Node) автоматически адаптируется к классу объекта, задаваемого по ссылке.

Property Node

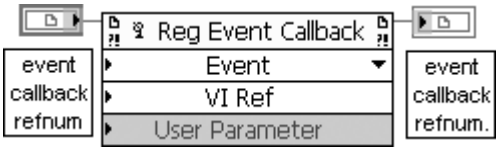


Узел свойства

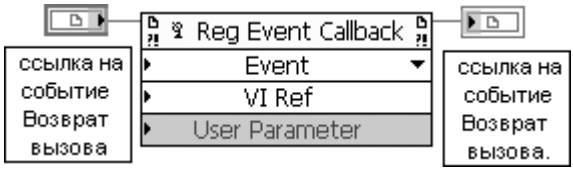


Узел получает (читает) и/или устанавливает (записывает) свойства по **ссылке** (reference). **Узел свойства** (Property Node) автоматически адаптируется к классу объекта, задаваемого по ссылке.
Выбор конкретного свойства при установленной ссылке производится с помощью строки **свойства** (Properties) 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 рисунке 4.17 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), т.е. 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 к входу VI Ref функции **Регистрация события Возврат вызова**.

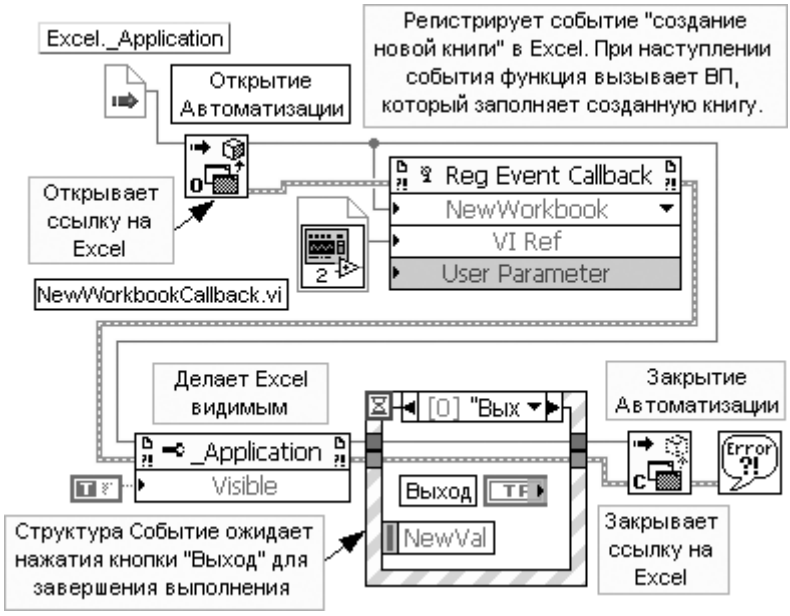
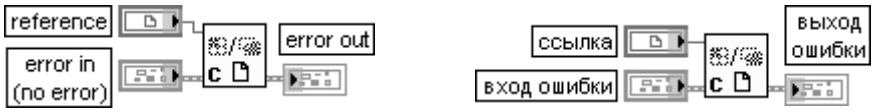


*Рис. 4.17. Блок-диаграмма ВП **Событие Возврат вызова ActiveX для Excel***

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 приложением можно привести размещенный в этом же разделе ВП **Слайд-шоу** (Slideshow), блок-диаграмма которого приведена на рис. 4.18. 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. 4.19 приведен еще один простой пример использования технологии ActiveX для программного вызова сервера DataSocket.

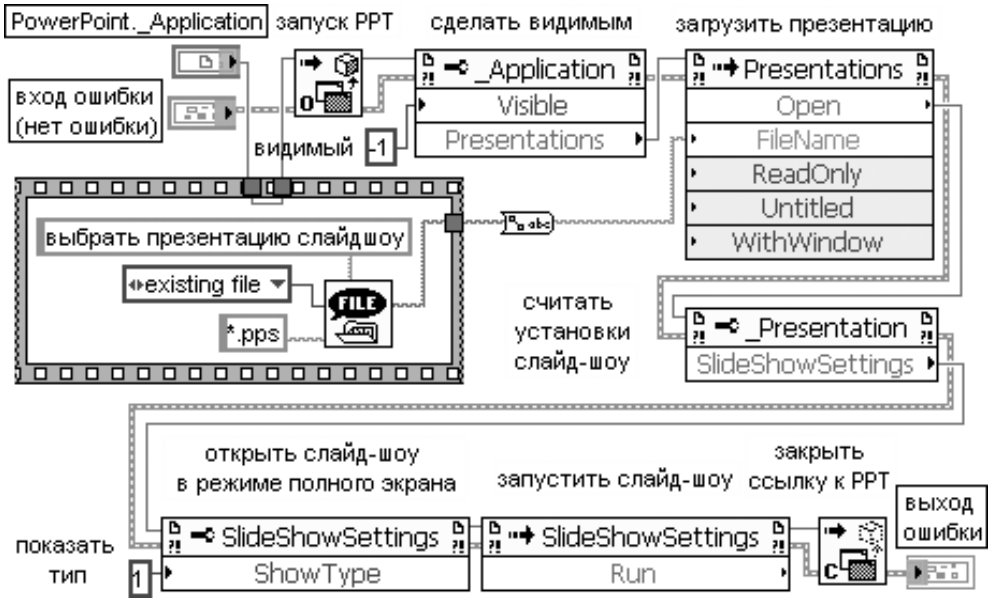


*Рис. 4.18. Блок-диаграмма ВП **Слайд-шоу** (Slideshow)*

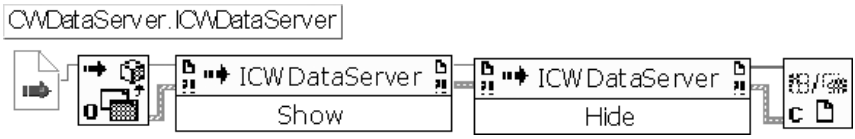


Рис. 4.19. Блок-диаграмма ВП программного вызова сервера DataSocket

4.2.2. Технология и функции .NET

Microsoft .NET является дальнейшим развитием технологии ActiveX. Так же как и ActiveX, .NET используется в LabVIEW для обеспечения доступа к другим при-

ложениям 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)[17].

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.

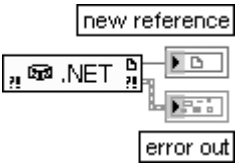
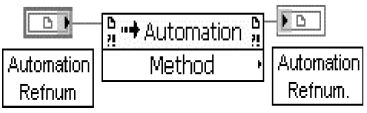
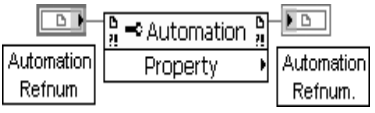
Библиотеки классов (Class Libraries) представляют набор классов, которые обеспечивают стандартную функциональность, такую как ввод и вывод, обработка строк, управление безопасностью, сетевое взаимодействие, управление потоками, управление текстом, средства конструирования пользовательского интерфейса и т. д. Эти классы содержат те же функции, которые использует система Win32/COM. В .NET framework пользователь может использовать 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 об имени сборки, информацию о версии, 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

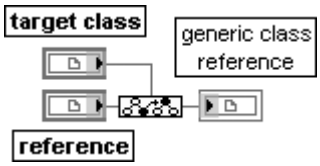
сборки могут иметь внешние ресурсы, такие как файлы изображений, звуковые файлы и т. п., или иметь один файл для кода ядра и другие для библиотек помощников.

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.

В следующей таблице рассмотрены функции подпалитры .NET.

Constructor Node 	**Узел конструктора** Узел формирует реализацию объекта .NET. Этот узел определяет конструктора, которые должен создать объект .NET. При помещении узла на блок-диаграмму LabVIEW отображает диалоговое окно **Выбрать конструктор .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 передавать ся 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.

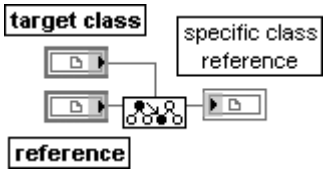
Вход **целевой класс** (target class) определяет класс, до которого повышается **ссылка** (reference). К этому входу можно подключить константу описателя класса или ссылку .NET.

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 элемента 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 передней 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. 4.20 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.

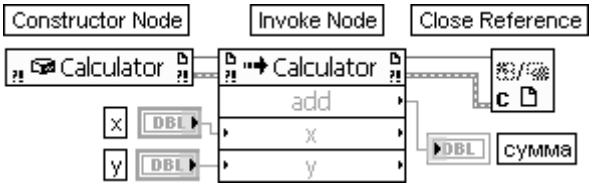


Рис. 4.20. Блок-диаграмма упрощенной версии ВП ***Калькулятор*** *(Calculator)*

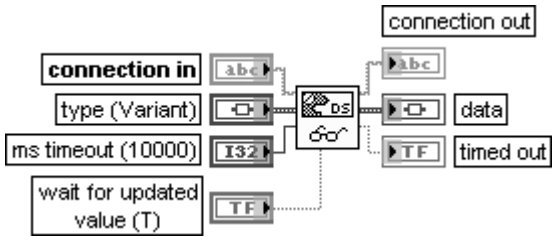
4.2.3. Технология передачи данных и функции DataSocket

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 источника. В первом случае они являются **источниками** (publishers) DataSocket, во втором – **приемниками** (subscribers) DataSocket.

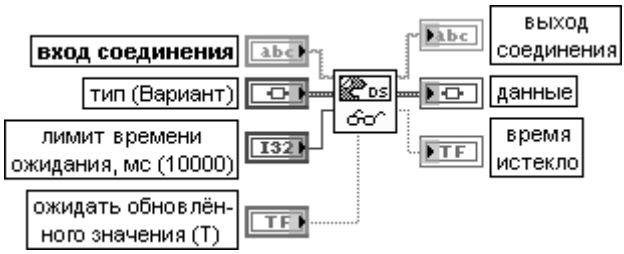
Передача данных с помощью технологии DataSocket может осуществляться как с помощью функций DataSocket, так и непосредственно между элементами лицевых панелей локальных или удаленных ВП. В последнем случае необходимо создать соединение DataSocket с помощью диалогового окна **Соединение DataSocket** (DataSocket Connection), вызываемого с помощью строки **Операции с данными** ⇒ **Соединение DataSocket** (Data Operations ⇒ DataSocket Connection) контекстного меню элемента. В окне указывается тип соединения – **Передать данные** (Publish) или **Принять данные** (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, применяющиеся для программной передачи данных с помощью соединения Data Socket, рассмотрены в следующей 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.

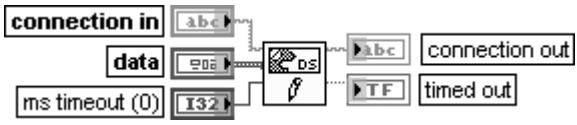
Вход **лимит времени ожидания, мс** (ms timeout) определяет необходимое время ожидания нового значения, становящегося доступным в буфере соединения. Функция игнорирует этот вход и не ожидает если на входе **ожидать обновленного значения** (wait for updated value) установлено состояние ЛОЖЬ и начальное значение поступило. По умолчанию значение входа равно 10,000 мс (10 секунд).

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.

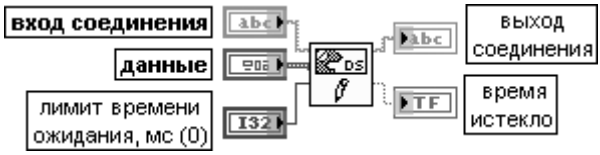
Выход соединения (connection out) представляет источник данных, который определяет соединение DataSocke**t**.

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.

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.

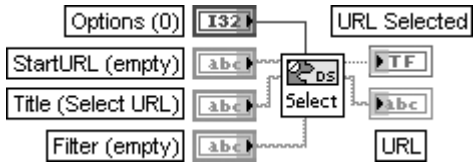
Вход **данные** (data) служит для передачи данных, записываемых в соединение DataSocket. Вход **данные** может быть любого типа.

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, отражая простое текстовое окно, или содержать протокол, например file:, отражая 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 правильного 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. 4.21. В ее основе лежит вызов функции ActiveX CWDataSocket. При этом после открытия ссылки к функции последовательно вызываются метод SelectURL и свойство URL.

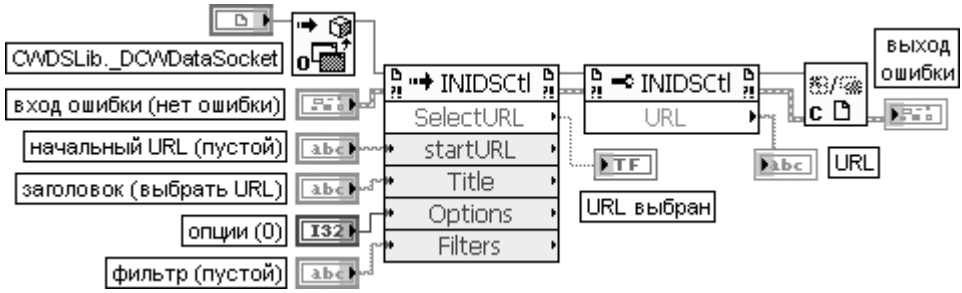
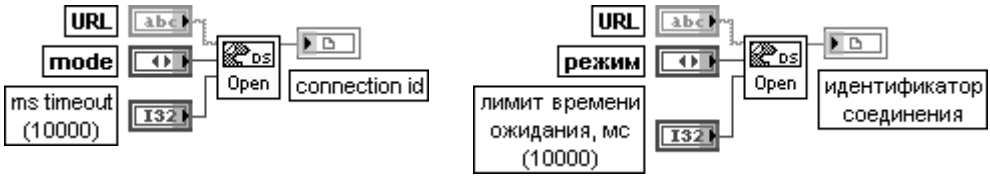


*Рис. 4.21. Блок-диаграмма ВП **Выбрать URL DataSocket***

DataSocket Open **Открыть соединение 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 **режим** (mode) определяет тип соединения DataSocket. Варианты режимов 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)

Буферизирование применяется только при использовании функции **Читать из соединения DataSocket** для приема данных, передаваемых сервером. Буферизирование не применяется при использовании соединения DataSocket для приема данных от элементов передней 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 неопределенное время. При установке нулевого значения LabVIEW не пытается осуществить соединение DataSocket и возвращает ошибку 56.
Выход **id соединения** (connection id) однозначно определяет идентификационный номер соединения DataSocket

DataSocket Close **Закрыть соединение DataSocket**

Функция закрывает соединение DataSocket, определенное с помощью ссылки **id соединения** (connection id).
Назначение входа **лимит времени ожидания, мс** (ms timeout (0)) идентично назначению одноименного входа описанной выше функции **Запись в соединение DataSocket**

На рис. 4.22 и 4.23 приведены блок-диаграммы ВП **Устройство записи данных трехмерного графика в соединение DataSocket** (DS 3D Graph Writer) и **Устройство считывания данных трехмерного графика из соединения DataSocket** (DS 3D Graph Reader) из набора примеров LabVIEW.

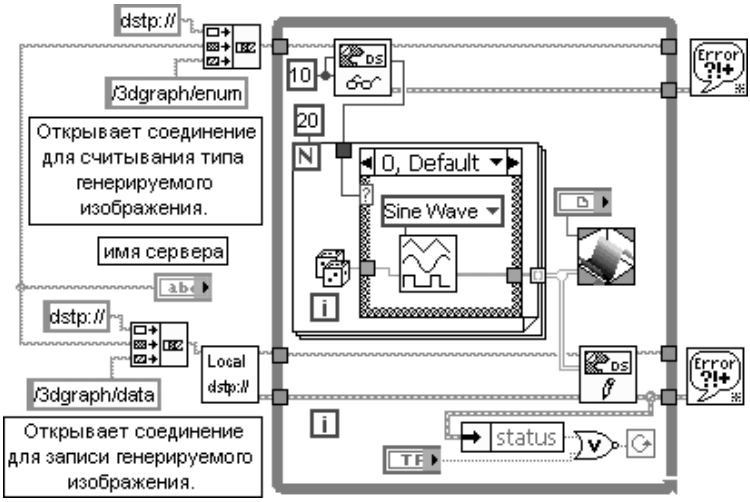


*Рис. 4.22. Блок-диаграмма **ВП Устройство записи данных трехмерного графика в соединение DataSocket** (DS 3D Graph Writer)*

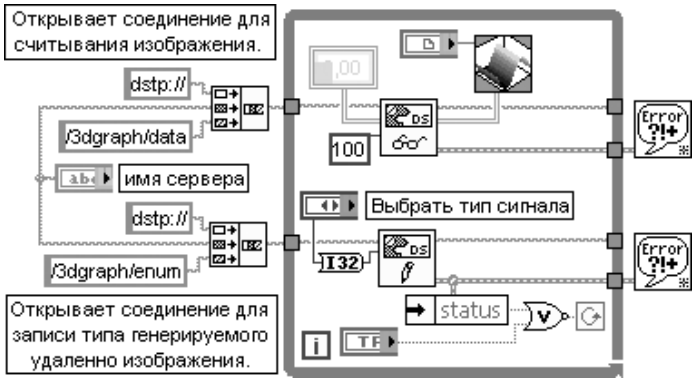
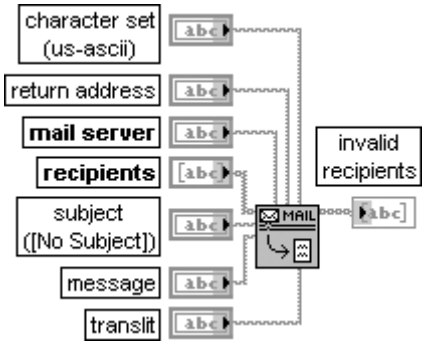


Рис. 4.23. Блок-диаграмма ***ВП Устройство считывания данных трехмерного графика из соединения DataSocket*** *(DS 3D Graph Reader)*

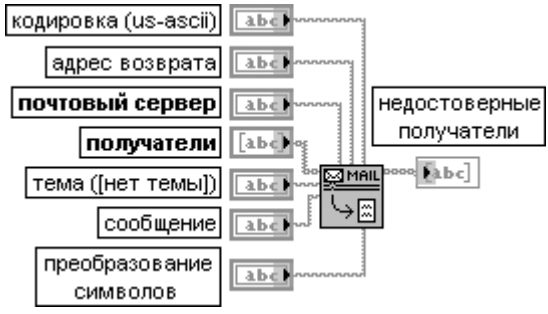
4.2.4. Функции электронной почты

Функции электронной почты рассмотренны в следующей таблице.

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.

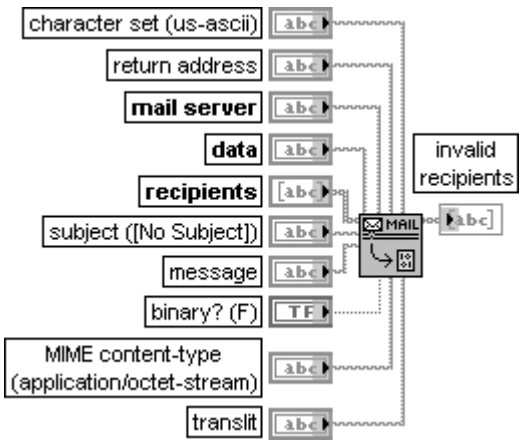
Вход **тема** (subject) задает строку темы сообщения. По умолчанию в этой строке тема не задана [No Subject].

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



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**.

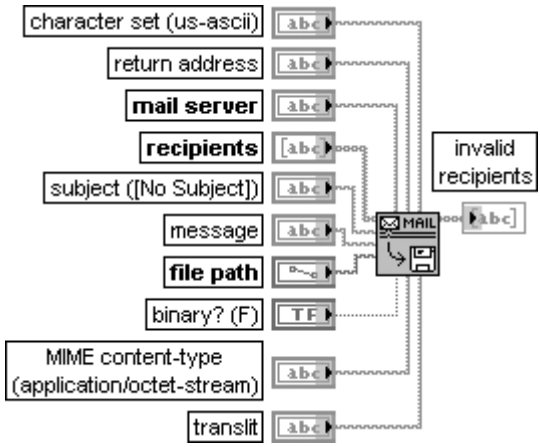
Отправить данные по электронной почте

Вход **двоичный?** (binary?) определяет тип вложения – текстовое, описываемое параметром **набор символов** (character set), или же двоичное, описываемое параметром **тип содержимого MIME**. По умолчанию данный вход установлен в состояние ЛОЖЬ, которое определяет текстовое вложение.

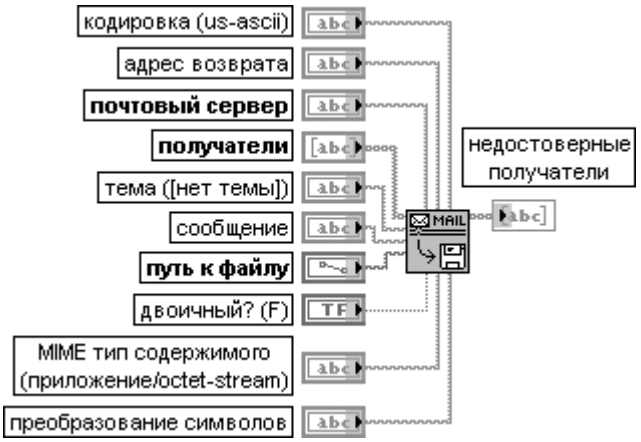
Вход **тип содержимого MIME** (MIME content-type) определяет тип содержимого двоичного вложения к сообщению. Могут быть определены ряд типов, поддерживаемых стандартом RFC 2045, в частности следующие наиболее используемые типы:

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

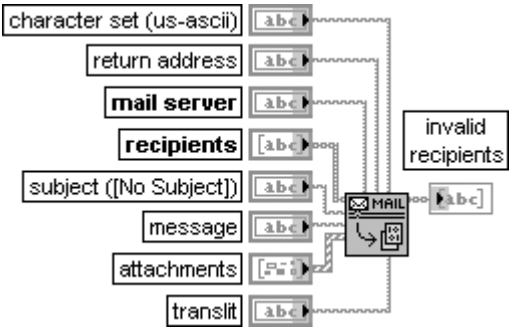


Отправить файл по электронной почте

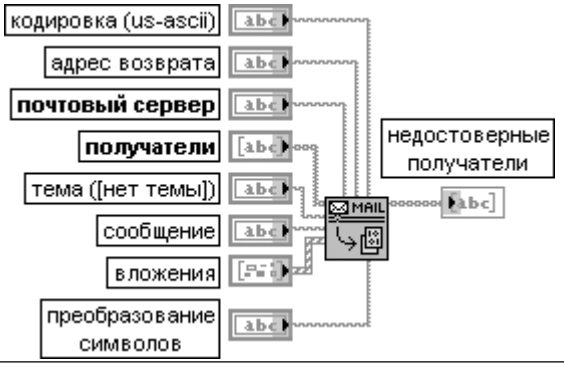


ВП отправляет электронное письмо с вложенным файлом по списку получателей. Вход **путь к файлу** (file path) определяет путь к вложенному файлу. Вложенной может быть строка или файл на локальном компьютере

SMTP Email Send Multiple Attachments

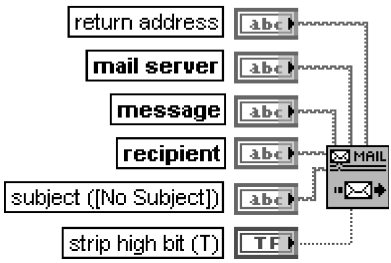


Отправить набор вложений по электронной почте



ВП отправляет электронное письмо с набором вложенных файлов и данных по списку получателей

SMTP Email Send Message (Small)



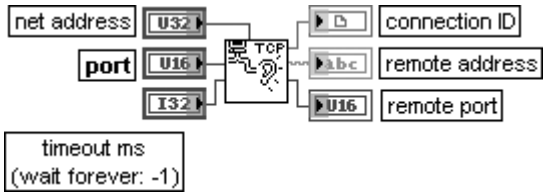
Отправить небольшое сообщение по электронной почте

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

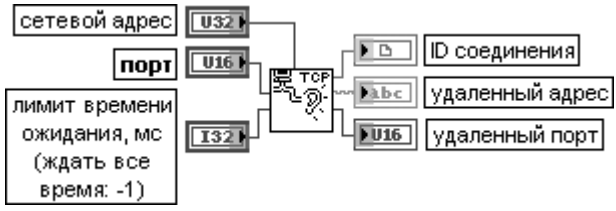
4.2.5. Функции протоколов TCP/IP и UDP

Функции протоколов TCP/IP и UDP рассмотренны в следующей таблице.

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.

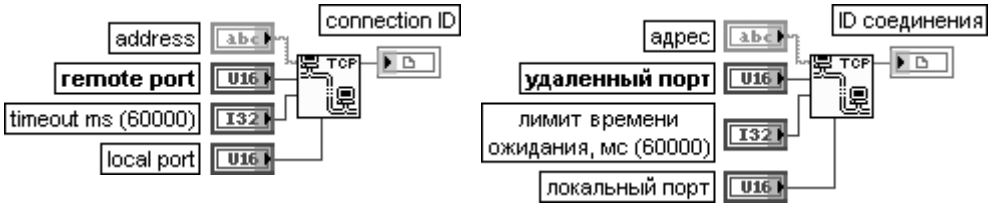
Вход **лимит времени ожидания, мс** (timeout ms) определяет время в миллисекундах, в течение которого должно быть установлено соединение. Если этого не произошло, то ВП завершается и возвращает ошибку. По умолчанию значение входа равно –1, что определяет неограниченное ожидание.

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.

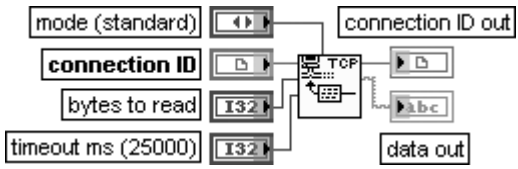
Вход **удаленный порт** (remote port) определяет порт для заданного **адреса** (address), с которым осуществляется соединение.

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.

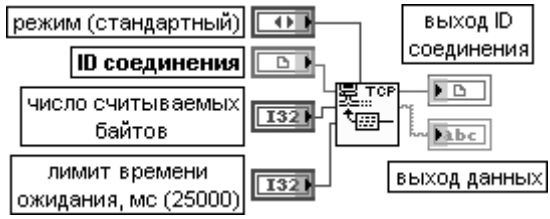
Вход **локальный порт** (local port) задает локальный порт соединения. Некоторые серверы разрешают соединения клиентам, которые используют номера портов из определенного диапазона, который зависит от сервера. Если значение на этом входе равно 0, то операционная система выбирает неиспользуемый порт.

Функция выхода **ID-соединения** (connection ID) была рассмотрена в предыдущем разделе.

TCP Read



Читать из TCP

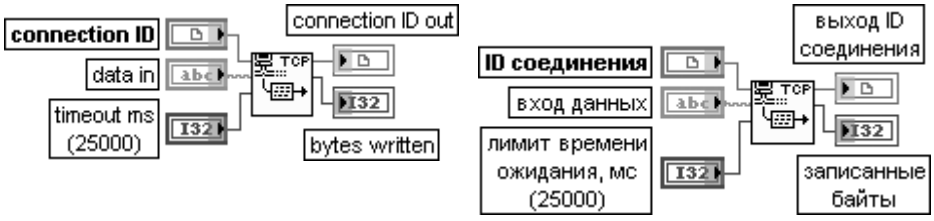


Функция читает заданное число байтов из сетевого соединения по протоколу TCP, возвращая результат на **выходе данных** (data out).
Вход **режим** (mode) устанавливает следующие режимы операции чтения.

0	**стандартный** (standard) (по умолчанию) – ожидается поступление всех запрошенных байтов или истечение времени, заданного на входе **время 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. 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. 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 миллисе-кундах, которое вход **режим** использует как максимальное время перед сообщением об ошибке по истечении 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**



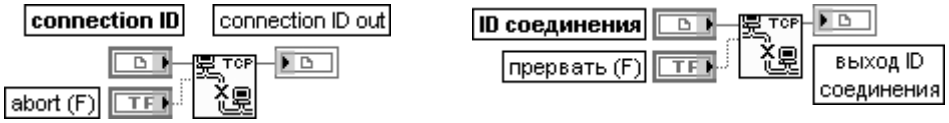
Функция записывает данные в сетевое TCP-соединение.

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 режим е множественного выхода.

Для переключения между режимами простого и множественного выхода необходимо выбрать из контекстного меню узла строку **множественный выход** (Multiple output).

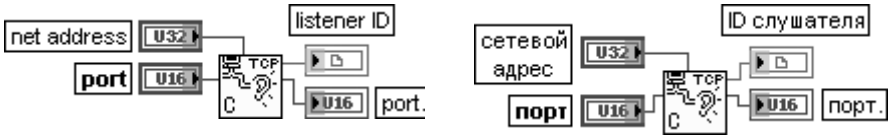
Вход **имя** (name) представляет преобразуемую строку. Если строка пустая, то **сетевой адрес** содержит IP-адрес текущей машины.

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**

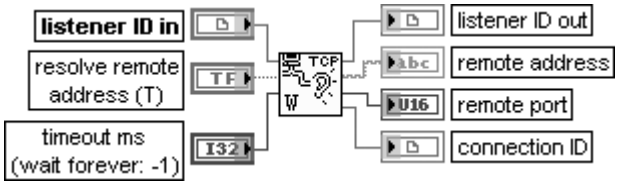
Функция преобразует сетевой адрес в строку.
Вход **сетевой адрес** (net address) представляет сетевой IP-адрес в виде набора целых чисел без знака, имеющих формат точечной записи.
Вход **точечная запись** (dot notation) определяет представление **имени** (name) в формате точечной записи. По умолчанию на входе установлено значение ЛОЖЬ.
Выход **имя** (name) представляет строковый эквивалент **сетевого адреса**

TCP Create Listener **Создать слушателя TCP**

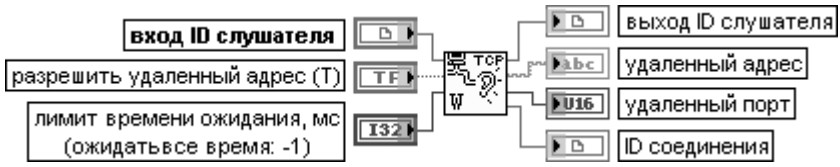


Функция создает слушателя для сетевого TCP-соединения. Подключение 0 к входу порта позволяет операционной системе динамически выбирать доступный порт для установки TCP-соединения.
Назначение входов и выходов функции идентично одноименным входам и выходам рассмотренной выше функции **TCP Listen**

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. 4.24 и 4.25 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.

Протокол UDP отличается от TCP тем, что не производится проверки

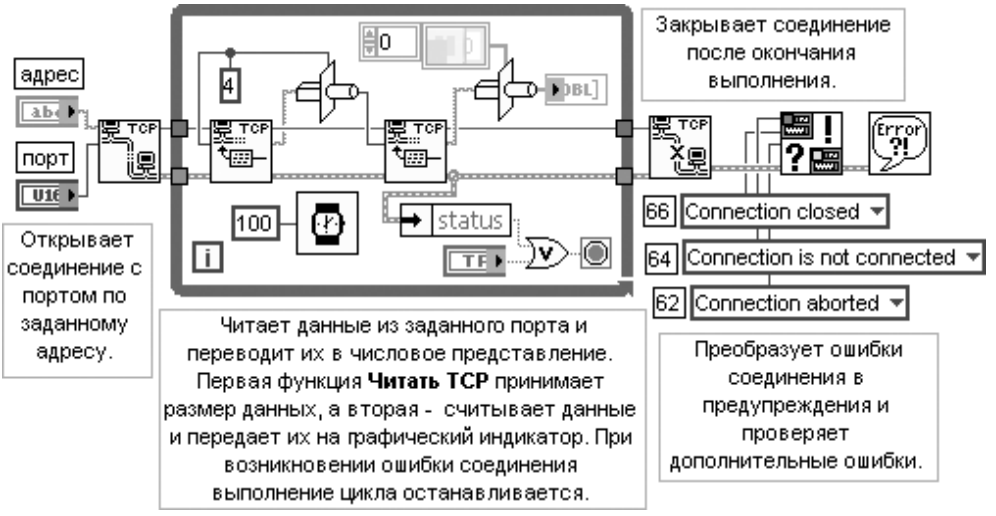


*Рис. 4.24. Блок-диаграмма ВП **Простой клиент данных** (Simple Data Client)*

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.

На рис. 4.26 и 4.27 приведены блок-диаграммы ВП **Отправитель UDP** (UDP

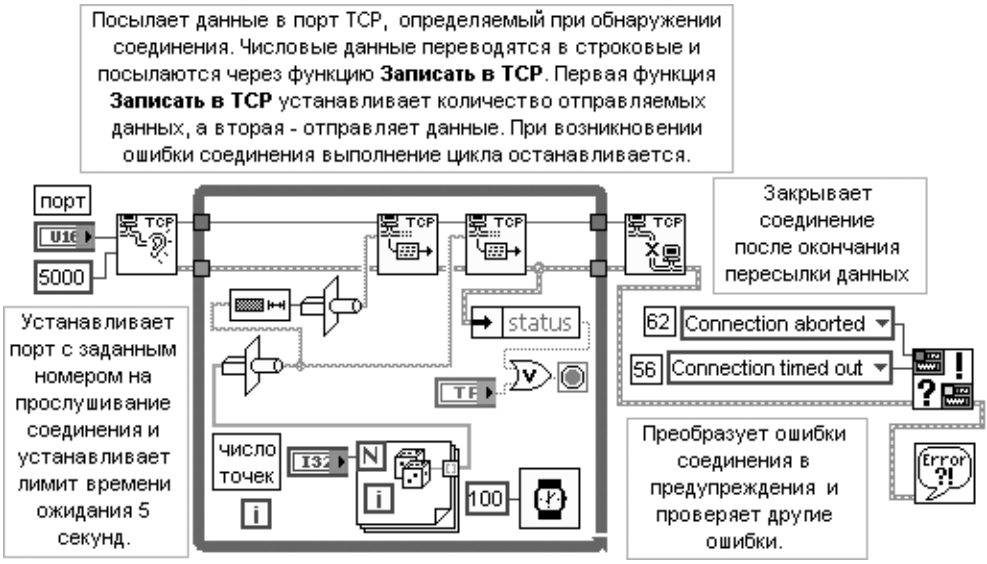
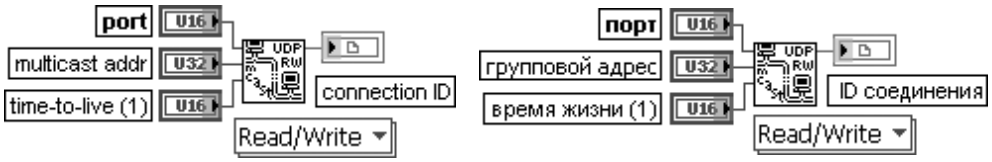


Рис. 4.25. Блок-диаграмма ВП **Простой сервер данных** (Simple Data Server)

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**

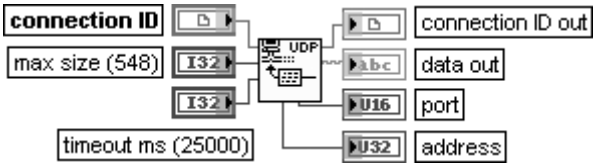


ВП открывает групповое UDP-соединение по заданному **порту** (port). Данный полиморфный ВП позволяет производить чтение, запись или чтение и запись данных по протоколу UDP. Выбор функции производится вручную с помощью меню.

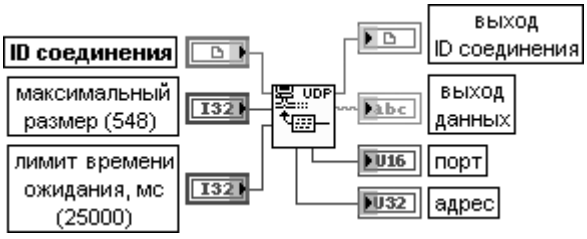
Вход **порт** (port) определяет локальный порт, с которым необходимо создать UDP- соединение.
Вход **групповой адрес** (multicast addr) определяет IP-адрес группы, с которой необходимо установить соединение. Если адрес на данном входе не установлен, то соединение с группой не производится и возвращаемое 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. Следующая 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

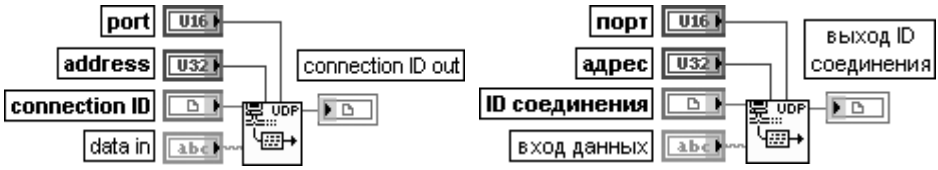


Читать из UDP



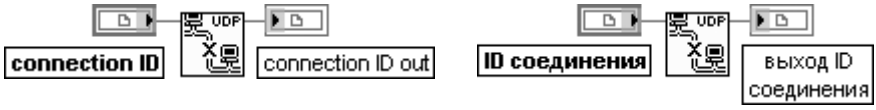
Функция читает дейтаграмму из UDP-соединения, сохраняя результат на **выходе данных** (data out). Функция возвращает данные при приеме какого-либо числа байтов и ожидает полное **время ожидания** (timeout ms), если прием байтов отсутствует.
Вход **максимальный размер** (max size) определяет максимальное 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. В сети LocalTalk объем данных ограничен 1458 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

Sender) и **Получатель UDP** (UDP Receiver) из набора примеров NI Example Finder, использующих функции UDP. Первым должен запускаться ВП **Получатель UDP**. Для обеспечения функционирования плат ввода/вывода данных и стандартных

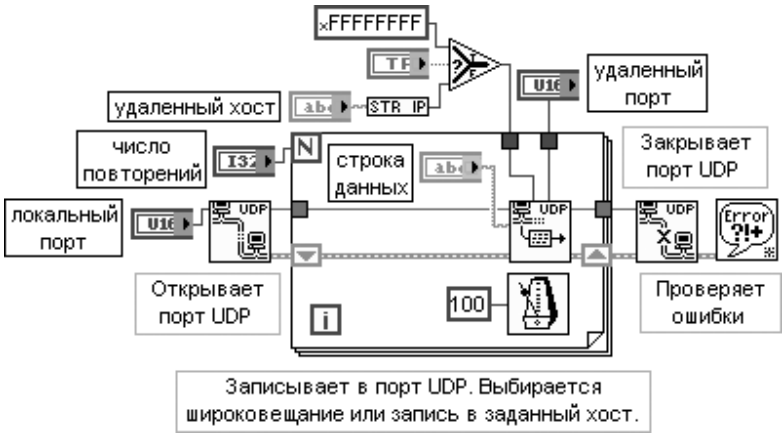


*Рис. 4.26. Блок-диаграмма ВП **Отправитель UDP** (UDP Sender)*

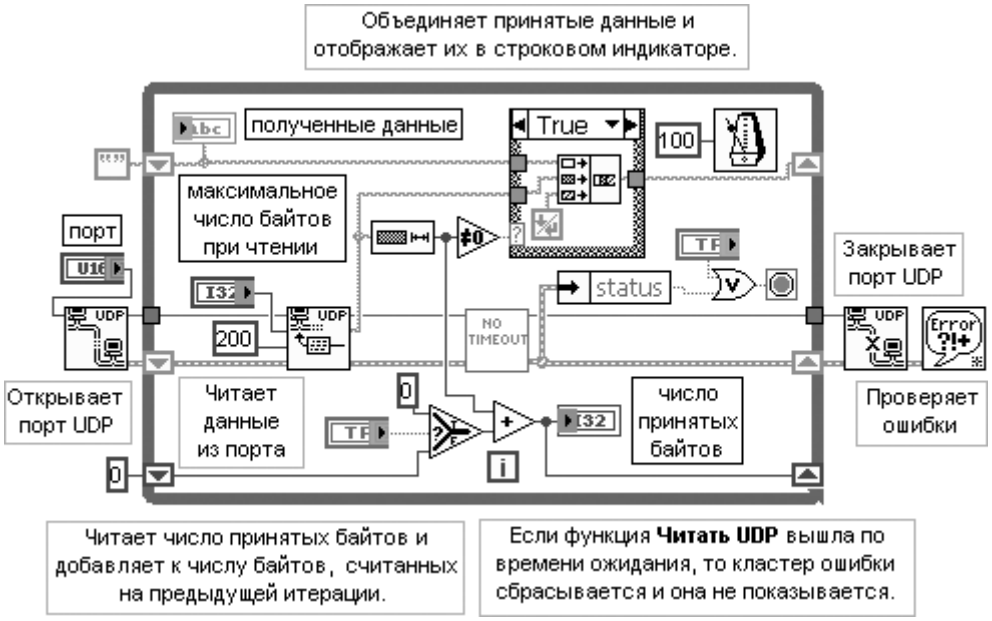
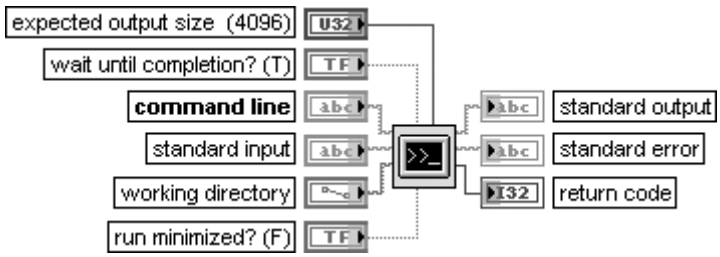
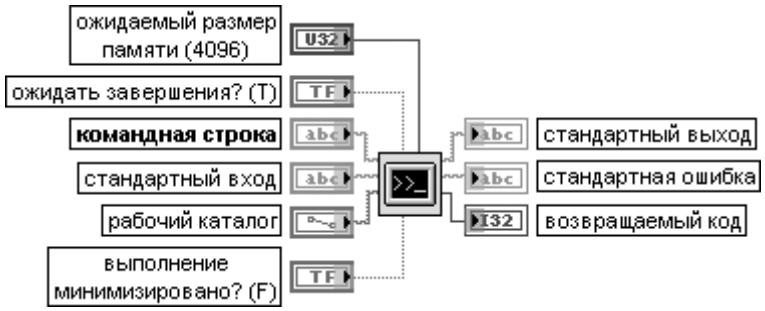


*Рис. 4.27. Блок-диаграмма ВП **Получатель UDP** (UDP Receiver)*

System Exec

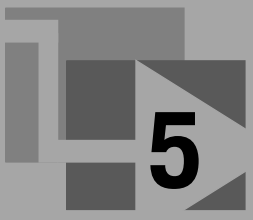


Системная командная строка



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

Функции плат и стандартных интерфейсов ввода/вывода данных



интерфейсов в состав палитры функций LabVIEW включены подпалитры **Измерения NI** (NI Measurements), **Связь с прибором** (Instrument I/O) и **Осциллограмма**. Перечисленные подпалитры имеют достаточно сложную иерархическую структуру.

Так, в частности, в состав подпалитры **Измерения NI** входят подпалитры **Сбор данных** (Data Acquisition), **Сбор данных DAQmx** (DAQmx – Data Acquisition), **Переключатель NI** (NI SWTCH), **Зрение** (Vision) и **Движение** (Motion). В данном справочном пособии более подробно в разделе 5.2. рассмотрены только функции из подпалитры **Сбор данных DAQmx**, поскольку функции из подпалитры **Сбор данных** были рассмотрены в книгах [1, 2].

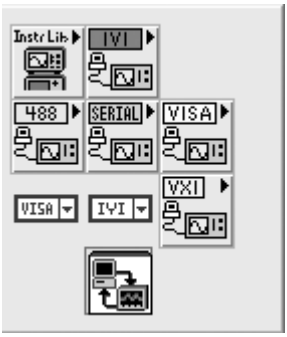


Рис. 5.1. Вид палитры Связь с прибором

В свою очередь, в состав подпалитры **Связь с прибором** (рис. 5.1) входят подпалитры **Драйверы приборов** (Instrument Drivers), **Драйверы класса IVI** (IVI Class Drivers), **Канал общего пользования (КОП)** (General Purpose Interface Bus (GPIB)), **Последовательный канал** (Serial), **Архитектура программного обеспечения виртуальных приборов** (Virtual Instrument Software Architecture (VISA)), **Расширение шины VME для использования в приборах** (VMEbus extension for Instrumentation (VXI)) и Экспресс-ВП **Помощник по связи с прибором** (Instrument I/O Assistant). В справочном пособии функции КОП рассмотрены в разделе 5.3, а функции последовательного канала – в разделе 5.4.

Начальным этапом работы с встроенными платами и стандартными интерфейсами ввода/вывода данных после установки плат и драйверов является 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. 5.2), которое позволяет выбирать и устанавливать различные режимы функционирования плат и стандартных интерфейсов.

В левом окне **Конфигурация** (Configuration) показываются разделы кон-

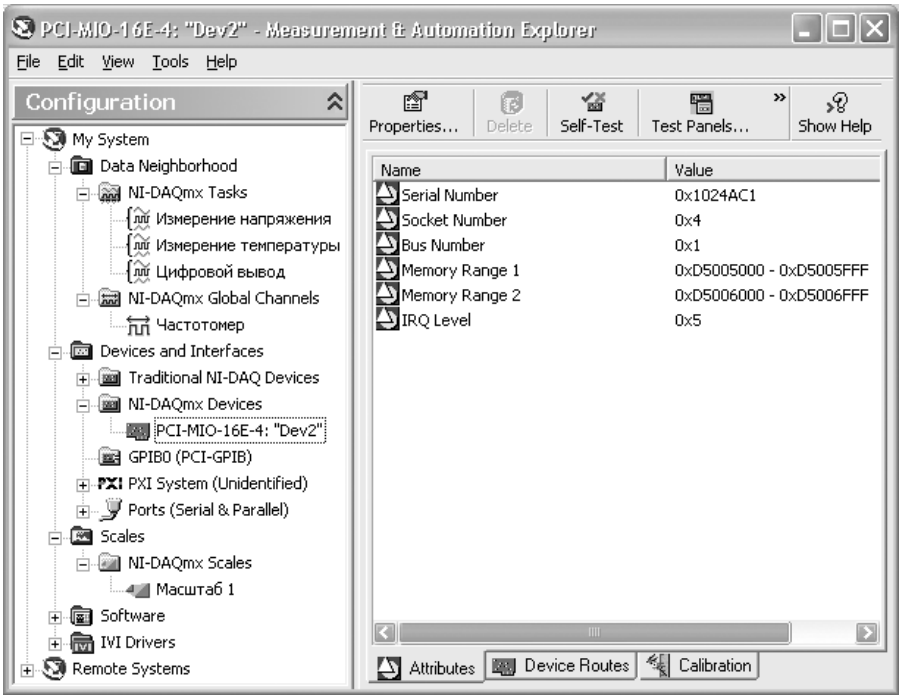


Рис. 5.2. Вид диалогового окна программы MAX

фигурации 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).

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. 5.3), 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.

Раздел **Устройства и интерфейсы** окна **Конфигурация** (рис. 5.2) позволяет

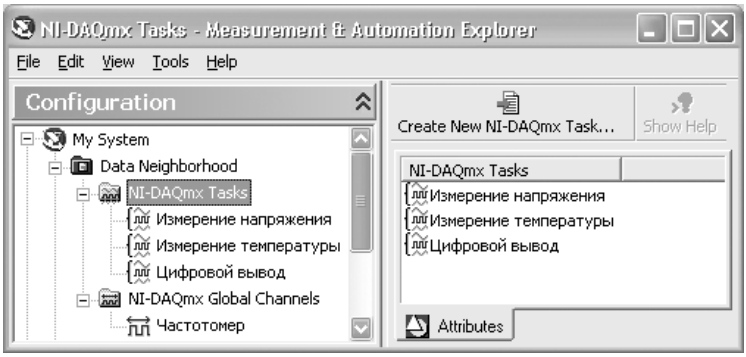


*Рис. 5.3. Вид диалогового окна MAX при выборе пункта **Задачи NI-DAQmx***

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.

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. Более подробно функции работы с аналоговыми и цифровыми осциллограммами рассмотрены далее в разделе 5.1.

5.1. Функции формирования и обработки осциллограмм

Вид основной палитры функций **Осциллограмма** (Waveform) показан на рис. 5.4а. На последующих рисунках показаны подпалитры функций: **Аналоговая осциллограмма** (Analog Waveform) (рис. 5.4б), **Цифровая осциллограмма** (Digital Waveform) (рис. 5.4в), **Ввод/вывод осциллограмм в/из файл(а)** (Waveform File I/O) (рис. 5.4г), **Измерения параметров осциллограмм** (Waveform Measurements) (рис. 5.4д) и **Генерация осциллограмм** (Waveform Generation) (рис. 5.4е).

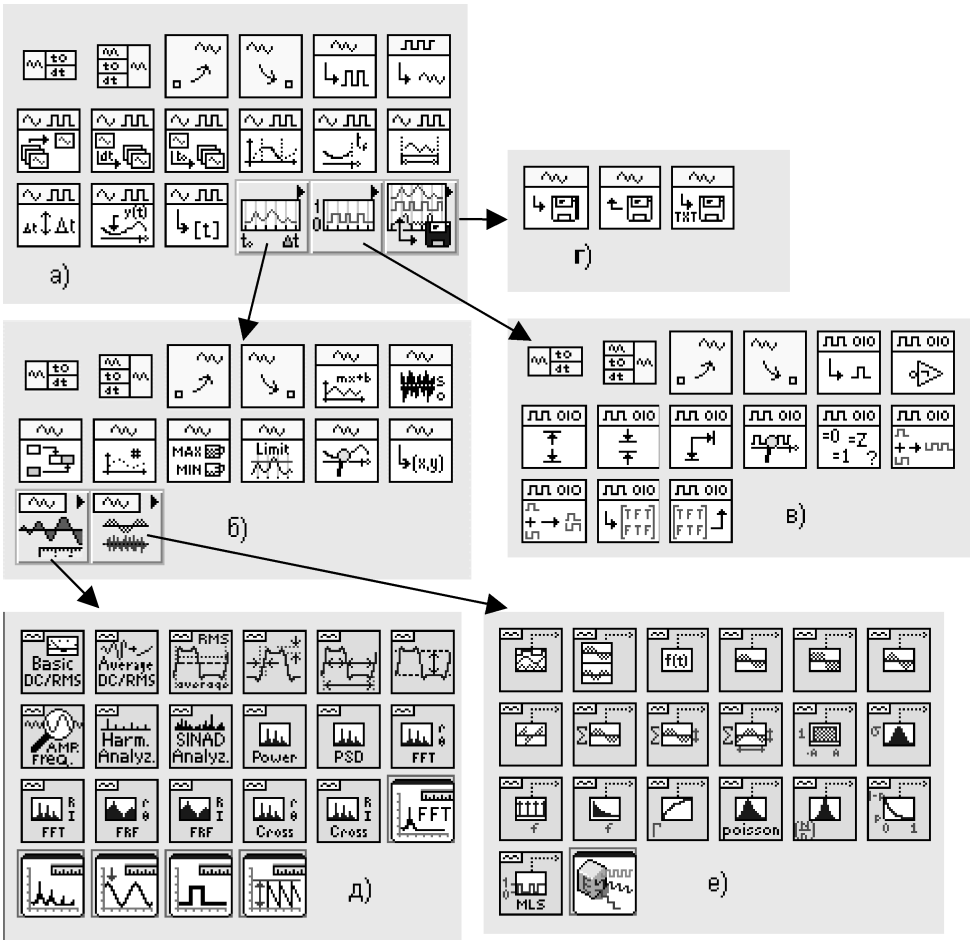
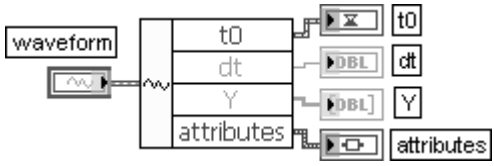


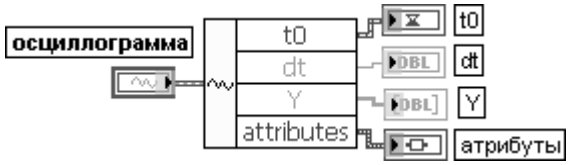
Рис. 5.4. Вид палитры (а) и дополнительных подпалитр (б–е) функций работы с осцилограммами

5.1.1. Базовые функции обработки аналоговых и цифровых осциллограмм

Get Waveform Components



Получить компоненты осциллограммы



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.

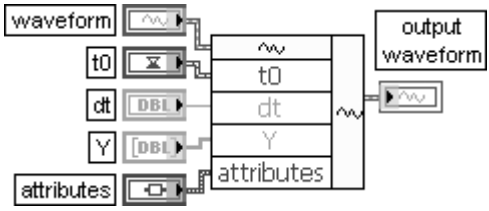
Выход **t0** представляет время (в соответствии с системными часами) получения первой точки массива **Y**.

Выход **dt** представляет интервал времени между точками массива **Y**.

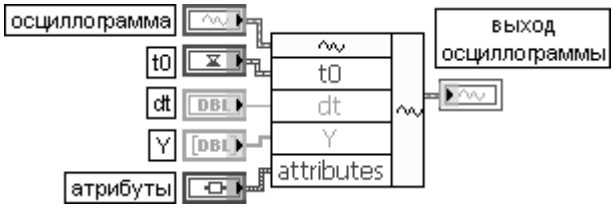
Выход **Y** представляет значения данных осциллограммы.

Выход **атрибуты** (attributes) позволяет передавать и выводить дополнительные данные

Build Waveform

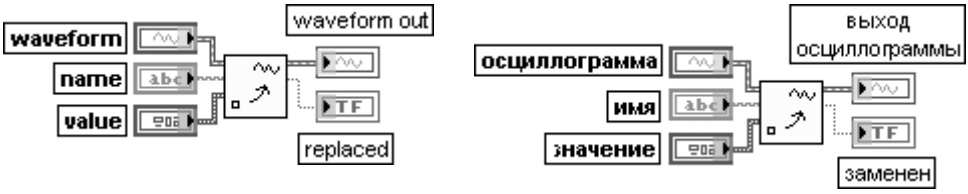


Создать осциллограмму



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

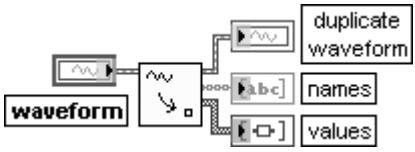
Set Waveform Attribute **Установить атрибут осциллограммы**



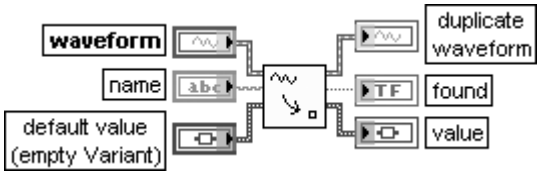
Функция добавляет или заменяет атрибут осциллограммы. В качестве **значения** (value) атрибута могут использоваться данные произвольного типа. Если атрибут в **имени** (name) уже существует, функция заменяет его значение новым значением и устанавливает выход **заменен** (replaced) в состояние ИСТИНА. Если атрибут в имени не существует, функция создает новый атрибут

Get Waveform Attribute

а)

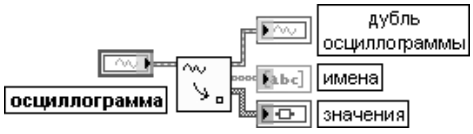


б)



Получить атрибуты осциллограммы

а)



б)

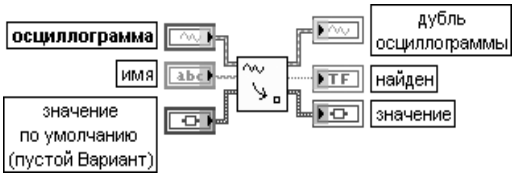


Рис. 5.5. Варианты подключения функции ***Получить атрибуты осциллограммы***

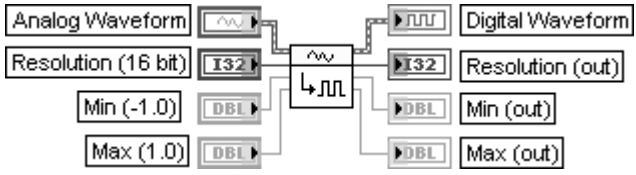
Функция извлекает **имена** (names) и **значения** (values) всех атрибутов или **значение** (value) единственного атрибута в зависимости от способа подключения входа **имя** (name). Атрибуты могут быть именами каналов. Соединения отображают типы данных по умолчанию для этой полиморфной функции.

Эта функция имеет два режима поведения 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. 5.5а). Если вход имя (name) подключен, то выход **имена** (names) изменяется на логический выход **найден** (found) и выходы **значения** (values) изменяются на **значение** (value) (рис. 5.5б), функция ищет только заданный атрибут. Если функция не находит заданного атрибута 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**.

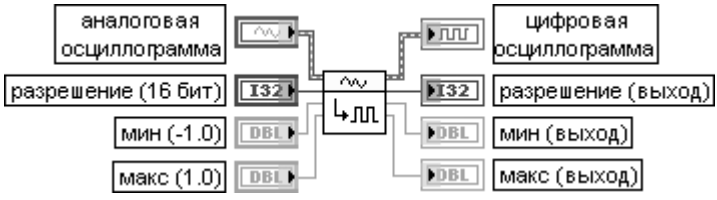
Следующая таблица включает список атрибутов осциллограммы, устанавливаемых NI-DAQ.

Имя	Атрибут	Тип данных	Допустимое значение	Описание
Номер устройства (Hardware Device Number)	NI_Device Number	строка	NI_DeviceNumber устанавливается с помощью MAX	NI_DeviceNumber является номером устройства, формирующего осциллограмму
Имя канала	NI_Channel Name	строка	NI_ChannelName устанавливается с помощью MAX	NI_ChannelName является именем канала, расположенного в устройстве, формирующем осциллограмму
Единица данных	NI_Unit Description	строка	Вольты и аналогичные единицы, являющиеся приемлемыми значениями для NI_UnitDescription	NI_UnitDescription является единицами измерения осциллограммы

Analog to Digital Waveform

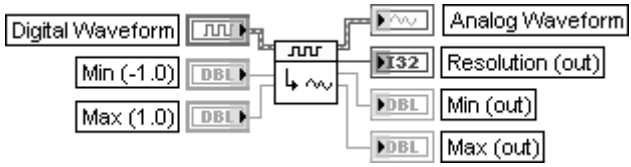


Аналоговую осциллограмму в цифровую

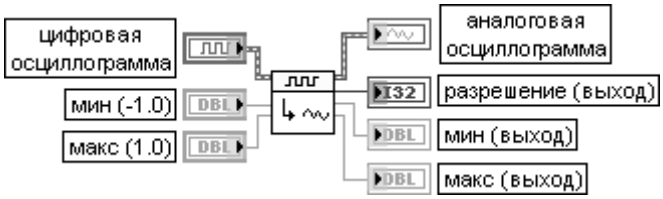


ВП преобразует аналоговую осциллограмму в цифровую.
Вход **разрешение** (Resolution) определяет число битов, представленных в цифровой осциллограмме.
Входы **мин** (Min) и **макс** (Max) определяют минимальное и максимальное значения, которые могут быть представлены в цифровой осциллограмме

Digital to Analog Waveform

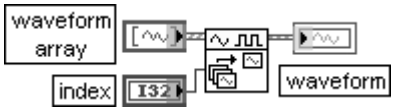


Цифровую осциллограмму в аналоговую

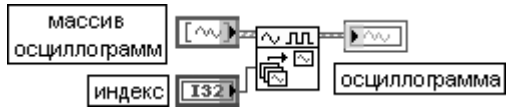


ВП преобразует цифровую осциллограмму в аналоговую.
Входы **мин** (Min) и **макс** (Max) определяют минимальное и максимальное значения, которые могут быть представлены в цифровой осциллограмме.
Выход **разрешение** (Resolution) возвращает число битов, представленных в цифровой осциллограмме

Index Waveform Array



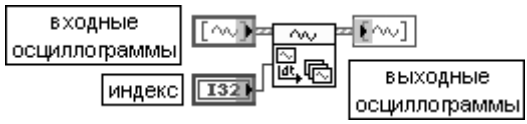
Индексировать массив осциллограмм



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 к входам **массив осциллограмм** (waveform array) и **индекс** (index), а также от типа данных атрибута **Y** аналоговой осциллограммы.
Блок-диаграмма ВП приведена на рисунке справа и не отличается высокой сложностью

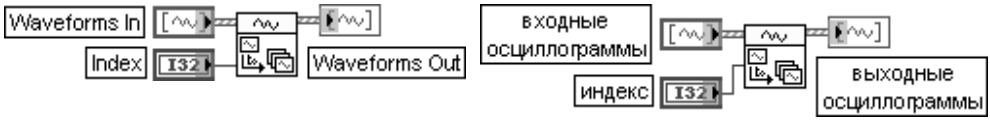
Copy Waveform dt

Копировать dt осциллограммы



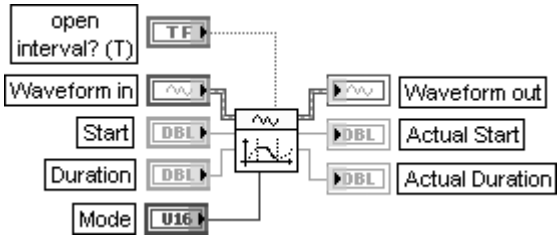
ВП заменяет все значения **dt** в массиве **входных осциллограмм** (Waveforms In) на значение **dt** осциллограммы, расположенной в этом массиве в позиции, заданной **индексом** (Index). Все типы осциллограмм, с которыми работает данная полиморфная функция, были перечислены выше

Align Waveform Timestamps **Выровнять начальное время осциллограмм**

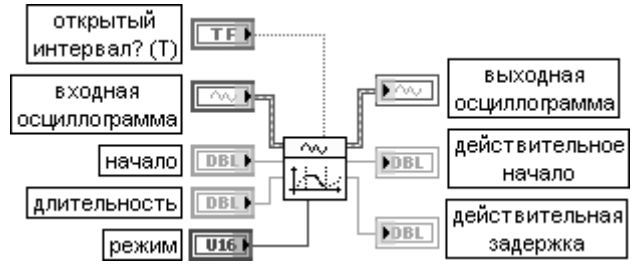


ВП заменяет все значения начального времени **t0** в массиве **входных осциллограмм** (Waveforms In) на значение **t0** осциллограммы, расположенной в этом массиве в позиции, заданной **индексом** (Index). Все типы осциллограмм, с которыми работает данная полиморфная функция, были перечислены выше

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 на 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.

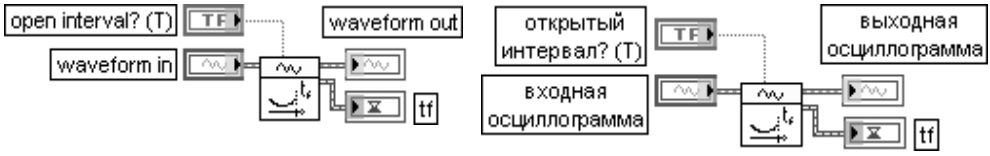
Вход **длительность** (Duration) определяет в зависимости от состояния входа **режим** (Mode) длительность извлекаемого набора данных или число элементов данных.

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.

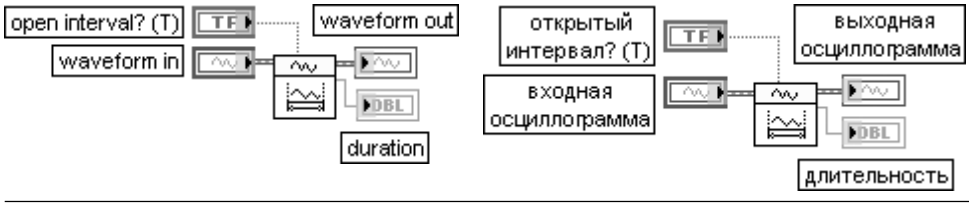
Значения на выходах **действительное начало** и **действительная длительность** зависят от величины параметра **dt** осциллограммы при установке входа **режим** в состояние **относительное время**

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) · **dt**. Блок-диаграмма ВП приведена на рис. 5.6. Число выборок осциллограммы определяется с помощью ВП **Число выборок осциллограммы**, рассмотренного ниже.

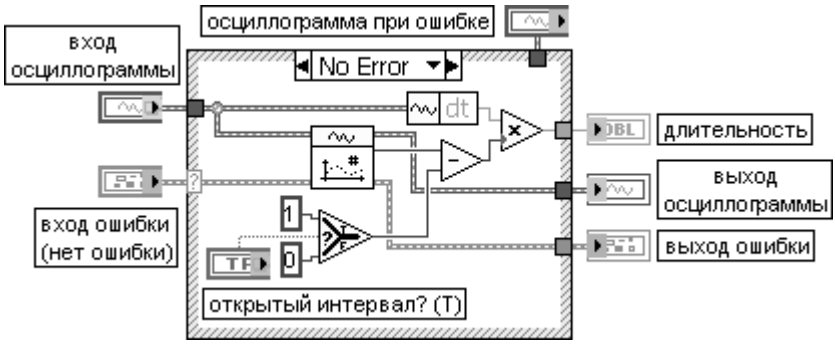
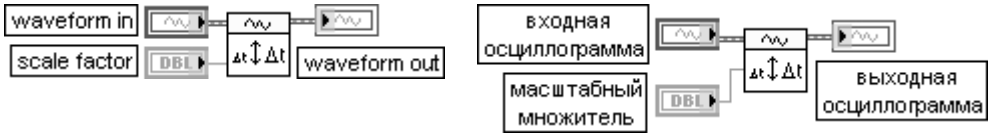


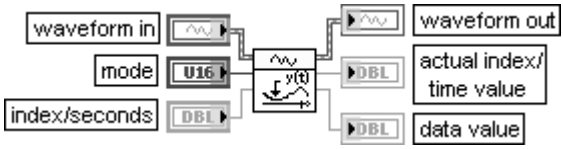
*Рис. 5.6. Блок-диаграмма ВП **Длительность осциллограммы***

Scale Delta t **Изменить интервал дискретизации**

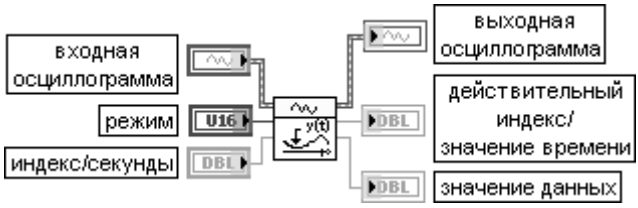


ВП умножает атрибут **dt** осциллограммы на заданный **масштабный множитель** (scale factor). С помощью такой операции изменяется частота выборок осциллограммы

Get Y Value



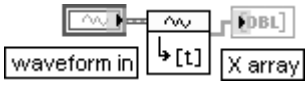
Получить значение Y



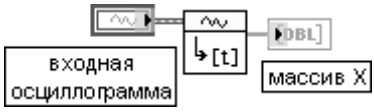
ВП возвращает значение Y **входной осциллограммы** или набора цифровых данных.

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 содержащихся в нем числа 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



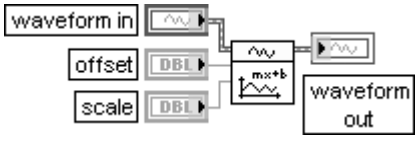
Получить массив отметок времени осциллограммы



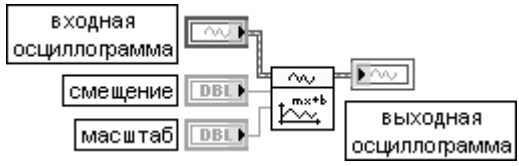
ВП создает массив отметок времени для каждого значения данных осциллограммы

В следующих таблицах приведены функции из подпалитры **Аналоговая осциллограмма** (Analog Waveform).

Waveform Scale and Offset



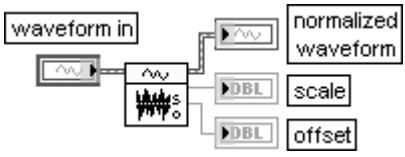
Масштабирование и смещение осциллограммы



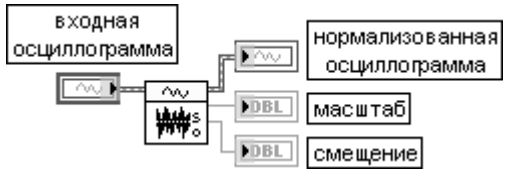
ВП масштабирует данные **входной осциллограммы,** используя выражение

выходная осциллограмма = (масштаб · входная осциллограмма + смещение)

Normalize Waveform



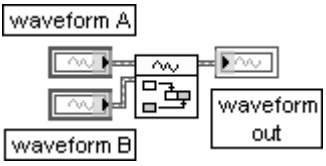
Нормализовать осциллограмму



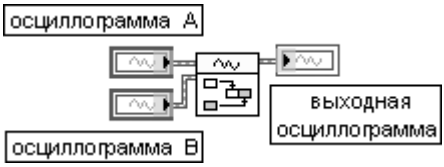
ВП определяет **масштаб** (scale) и **смещение** (offset), необходимые для преобразования данных **входной осциллограммы** к диапазону изменения от –1,0 до 1,0.

Осциллограмма, приведенная к такому диапазону, передается на выход **нормализованная осциллограмма**

Append Waveforms

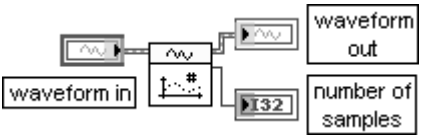


Добавить осциллограмму

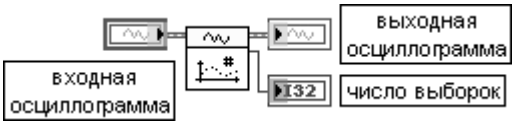


ВП добавляет осциллограмму **B** к концу осциллограммы **A**. Если частоты выборок не совпадают, то кластер ошибки возвращает ошибку. Время запуска осциллограммы **B** игнорируется. Тип данных атрибута **Y** осциллограммы **A** и осциллограммы **B** определяет используемую полиморфную реализацию ВП

Number of Waveform Samples

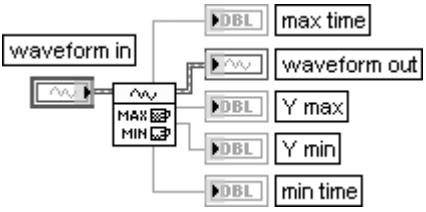


Число выборок осциллограммы

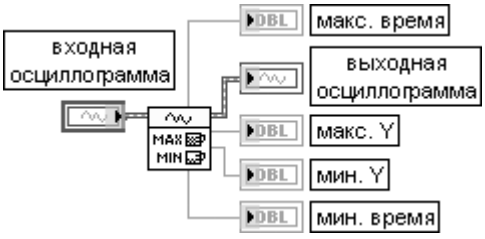


ВП возвращает число элементов осциллограммы

Waveform Min Max

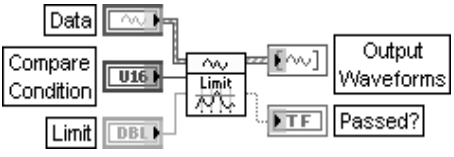


Минимум и максимум осциллограммы

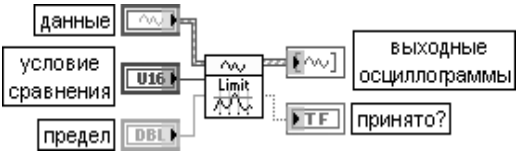


ВП определяет максимальное и минимальное значения осциллограммы и соответствующие значения времени

Waveform Scalar Limit Comparison



Сравнение осциллограммы со скалярным пределом

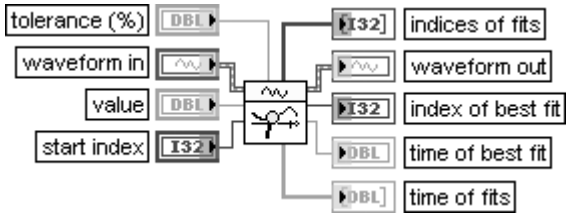


ВП сравнивает значения осциллограммы со скалярным пределом.
Вход **условие сравнения** (Compare Condition) отображает вариант сравнения максимального или минимального значений осциллограммы с числом, заданным на входе **предел** (Limit). Предусмотрены следующие варианты условия сравнения:

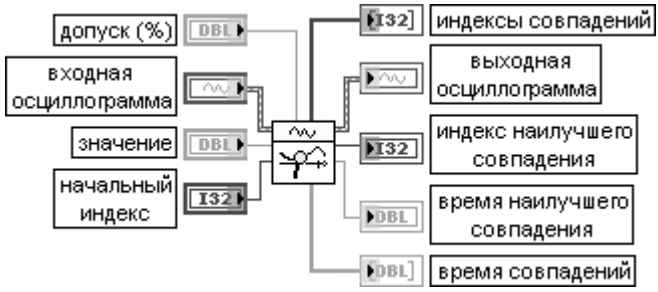
0	< – значение осциллограммы меньше **предела**
1	<= – значение осциллограммы меньше или равно **пределу**
2	> – значение осциллограммы больше **предела**
3	>= (по умолчанию) – значение осциллограммы больше или равно **пределу**

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



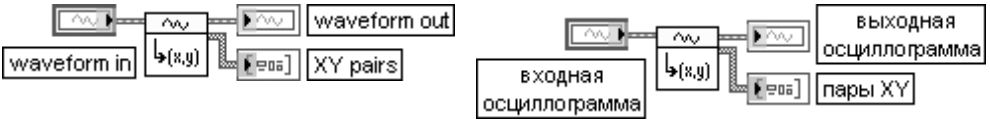
Поиск в осциллограмме



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)

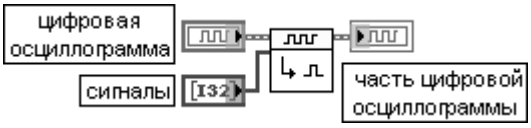
Waveform to XY Pairs **Осциллограмму в пары XY**



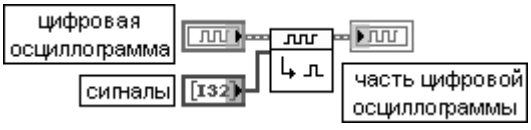
ВП создает массив значений данных и соответствующих им отметок времени.
Выход **пары XY** (XY pairs) содержит значения осциллограммы и соответствующие значения отметок времени

В следующих таблицах рассмотрены функции подпалитры **Цифровая осциллограмма** (Digital Waveform) и подпалитры **Ввод/вывод осциллограмм в/из файл(а)** (Waveform File I/O).

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. 5.7 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.

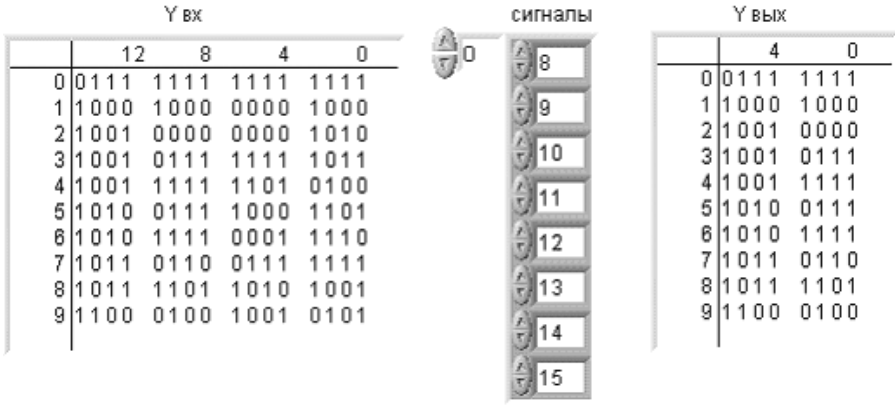
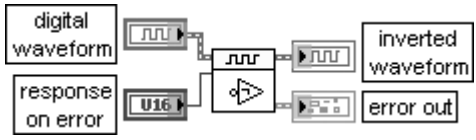
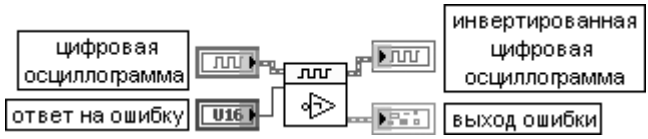


*Рис. 5.7. Значения данных цифровых осциллограмм на входе (Y вх) и выходе (Y вых) ВП **Часть цифровой осциллограммы** при заданных значениях входа **сигналы***

Invert Digital



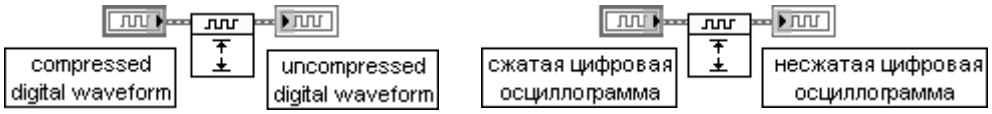
Инвертировать цифровую осциллограмму



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 в случае, если значение на цифровом входе не является 0 или 1.

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**

Uncompress Digital **Извлечь цифровую осциллограмму**



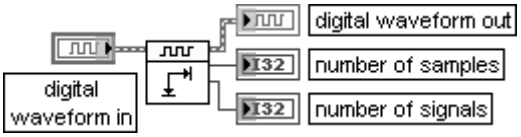
Функция извлекает цифровой сигнал на цифровом входе и возвращает результат на цифровом выходе

Compress Digital **Сжать цифровую осциллограмму**

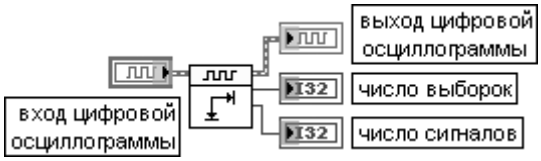
Функция сжимает цифровой сигнал на цифровом входе и возвращает результат на цифровом выходе.

Вход **выполнить сжатие?** (force compression?) определяет необходимость выполнения сжатия в случае, если данные оказываются сжатыми

Digital Size



Размер осциллограммы



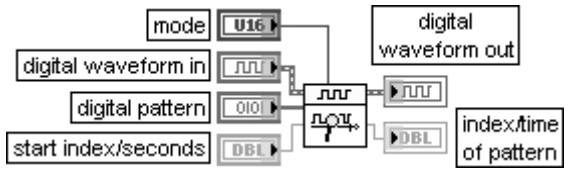
ВП возвращает число выборок и сигналов, содержащихся во входной цифровой осциллограмме.

Выход **число выборок** (number of samples) возвращает число выборок элементов данных, содержащихся в цифровой входной осциллограмме.

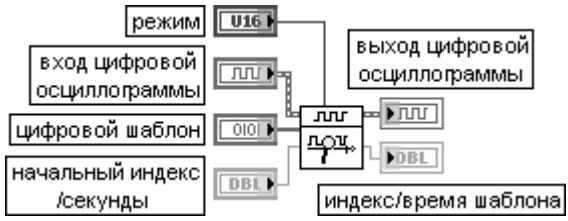
Выход **число сигналов** (number of signals) возвращает число сигнальных элементов, содержащихся в цифровой входной осциллограмме.

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

Search for Digital Pattern



Искать по цифровому шаблону



ВП осуществляет поиск информации во входной цифровой осциллограмме по цифровому шаблону.

Вход **режим** (mode) определяет метод поиска.

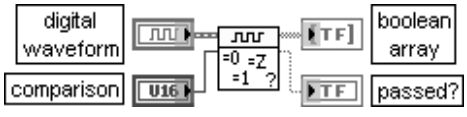
0	**Индексный** (Index) (по умолчанию) – поиск начинается с заданного элемента из набора данных **входной цифровой осциллограммы** (digital waveform in)
1	**Относительное время** (Relative Time) – поиск начинается с заданного времени

Вход **цифровой шаблон** (digital pattern) определяет цифровой шаблон, по которому осуществляется поиск.

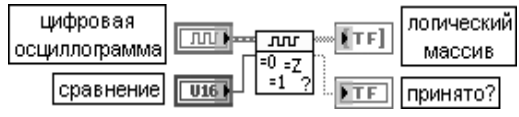
Вход **начальный индекс/секунды** (start index/seconds) определяет точку **входной цифровой осциллограммы**, с которой начинается поиск. Тип точки – индекс или время определяется входом **режим**. По умолчанию на этом входе установлено значение 0, что определяет поиск с начала **входной цифровой осциллограммы**. 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**.

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

Digital Comparison



Цифровое сравнение



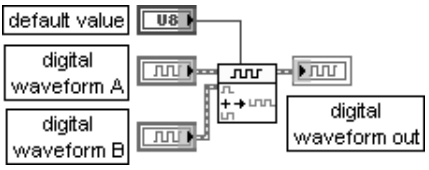
ВП сравнивает цифровые данные, содержащиеся в цифровой осциллограмме, со значением на входе **сравнение** (comparison) и возвращает состояние ИСТИНА, если это значение было найдено.

Вход **сравнение** (comparison) определяет значение, с которым сравниваются цифровые данные, содержащиеся в цифровой осциллограмме. Набор значений приведен в таблице.

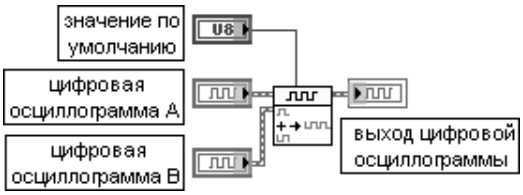
0	**Равно 0** (Equal 0) (по умолчанию)	6	**Равно T** (Equal T)
1	**Равно 1** (Equal 1)	7	**Равно V** (Equal V)
2	**Равно Z** (Equal Z)	8	**Равно 0 или L** (Equal 0 or L)
3	**Равно L** (Equal L)	9	**Равно 0 или H** (Equal 1 or H)
4	**Равно H** (Equal H)	10	**Не равно 0 или 1** (Not Equal 0 or 1)
5	**Равно X** (Equal X)	11	**Не равно 0, 1, L или H** (Not Equal 0, 1, L, or H)

Выход **принято?** (passed?) возвращает значение ИСТИНА, если хотя бы одно значение цифровых данных, содержащихся в цифровой осциллограмме, соответствует запросу на сравнение, установленному на входе **сравнение**

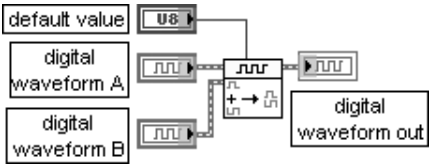
Append Digital Signals



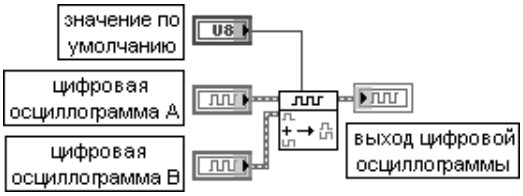
Добавить цифровые сигналы



Append Digital Samples



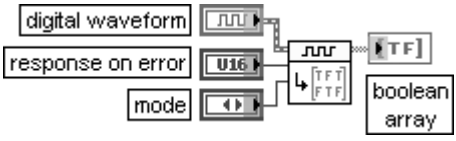
Добавить цифровые выборки



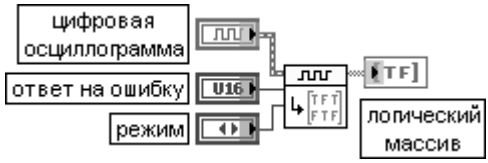
ВП добавляет **цифровую осциллограмму B** (digital waveform B) к концу **цифровой осциллограммы A** (digital waveform A). Если частоты дискретизации осциллограмм не совпадают, то на **выходе ошибки** (error out) возвращается предупреждение. Время запуска (trigger time) цифровой осциллограммы B игнорируется.

Этот полиморфный ВП может использоваться для добавления цифровой осциллограммы к другой цифровой осциллограмме или набора цифровых данных к другому набору цифровых данных

Digital to Boolean Array

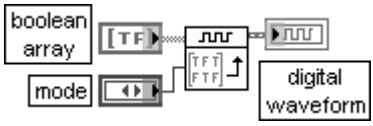


Цифровую осциллограмму в логический массив

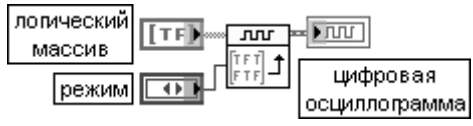


ВП преобразует цифровые данные на цифровом входе в двумерный логический массив

Boolean Array to Digital

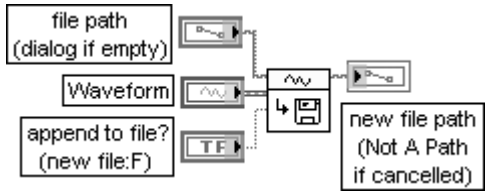


Логический массив в цифровую осциллограмму

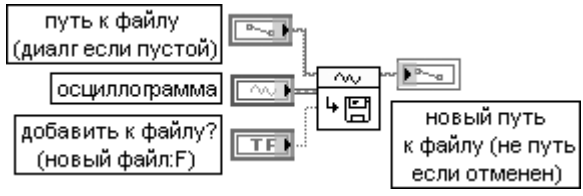


ВП преобразует двумерный массив в цифровую осциллограмму

Write Waveforms to File



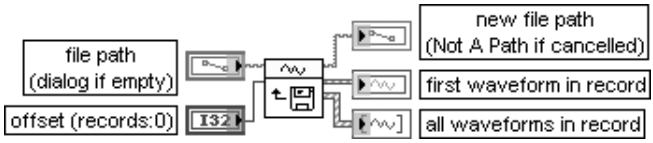
Записать осциллограмму в файл



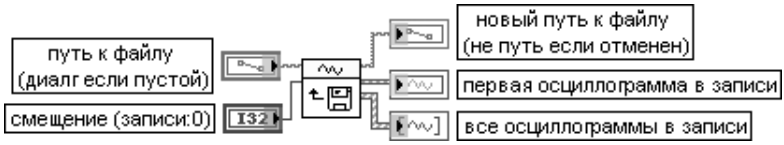
ВП создает новый файл или добавляет к существующему файлу, записывает заданное число записей в файл и закрывает файл, проверяя наличие ошибки. Каждая запись представляет массив осциллограмм. С помощью этого полиморфного ВП можно записывать в файл одномерный массив осциллограмм, двумерный массив осциллограмм или единичную осциллограмму. Тип данных, подключенных к 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

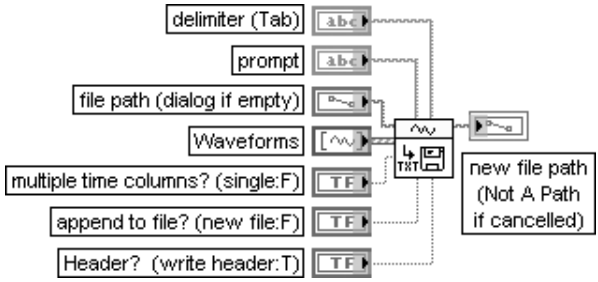


ВП открывает файл, созданный с помощью ВП **Записать осциллограмму в файл** (Write Waveforms to File), и читает одну запись файла. Каждая запись может содержать одну или более осциллограмм. ВП возвращает на отдельных выходах **все осциллограммы в записи** (all waveforms in record) и **первую осциллограмму 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 осциллограммы 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 к 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.

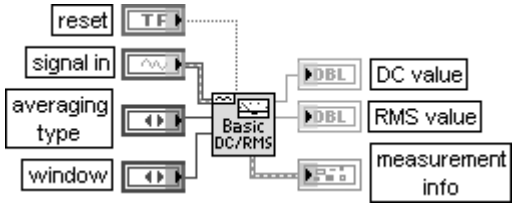
Вход **множество колонок времени?** (multiple time columns?) при установке в состояние ИСТИНА вызывает запись отдельных колонок для каждого отдельного канала, записываемого в файл. При установке в состояние ЛОЖЬ (по умолчанию) единственная колонка времени представляет один и тот же временной интервал 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.

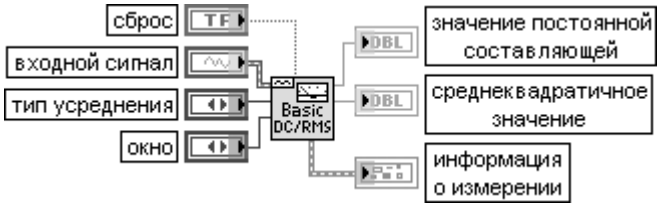
Вход **заголовок** (Header?), будучи установлен в состояние ИСТИНА (по умолчанию), вызывает транспонирование данных перед их преобразованием в строку

5.1.2. Функции измерения параметров осциллограмм

Basic Averaged DC-RMS



Основное 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 к 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. Варианты выбора окна приведены в таблице.

0	1	2
Прямоугольное (без окна) (Rectangular)	Ханна (Hanning)	(Low side lobe)

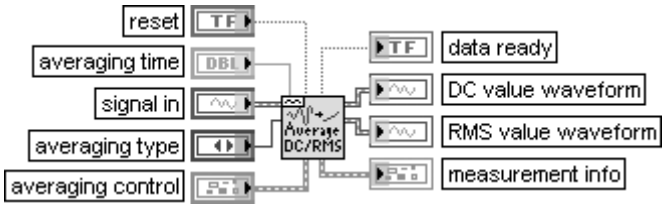
Выход **значение постоянной составляющей** (DC value) отображает измеренное значение **постоянной составляющей** в вольтах, если входной сигнал был задан в вольтах.

Выход **среднеквадратичное значение** (RMS value) отображает измеренное значение **среднеквадратичного значения** в вольтах, если входной сигнал был задан в вольтах.

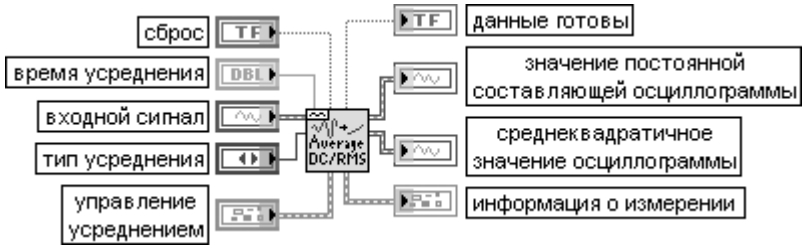
Выход **информация о измерении** (measurement info) возвращает информацию о измерении:

DBL	**Неопределенность** (uncertainty) зарезервировано для будущих приложений
TF	**Предупреждение** (Warning) устанавливается в состояние ИСТИНА, если во время обработки формируется предупреждение
abc	**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;

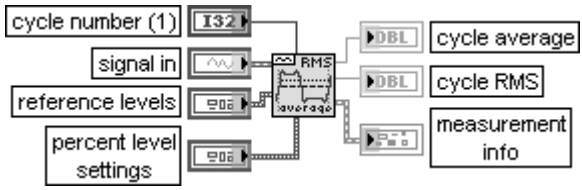
выходная функция (output function) устанавливает тип выполняемых измерений. При необходимости измерения только постоянной составляющей или только среднеквадратичного значения установка соответствующей **выходной функции** увеличивает скорость вычислений. Предусмотрены следующие варианты данного входа:

0	1	2
Только постоянная составляющая (DC only)	**Только среднеквадратичное значение** (RMS only)	**Постоянная составляющая и среднеквадратичное значение** (DC and RMS)

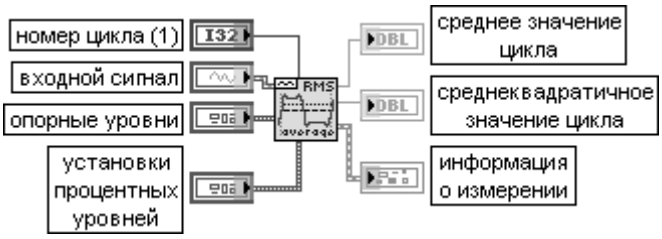
постоянная времени экспоненты (exp. time constant) определяет постоянную времени измерения постоянной составляющей и среднеквадратичного значения. Установка значения –1,00 на данном входе соответствует выбору постоянной времени на уровне половины длительности входного блока данных;

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:

DBL	**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
DBL	**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**
DBL	**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
U16	**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.

U32	**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
U32		**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:

$$среднее = \frac{1}{число\ точек} \sum_{i \subset цикл} осциллограмма[i],$$

где i отображает выборки осциллограммы, попадающие в один период, заданный **номером цикла** (cycle number), **число точек** определяется с помощью следующего выражения:

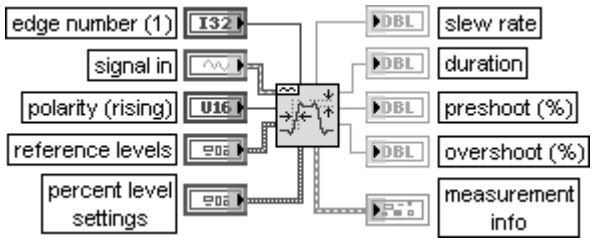
$$число\ точек = целая\ часть(период/dt + 0{,}5),$$

где dt является интервалом дискретизации.

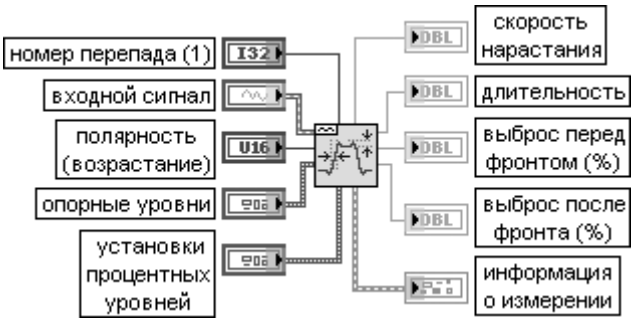
Выход **среднеквадратичное значение цикла** (cycle RMS) рассчитывается с помощью следующего выражения:

$$RMS_{цикл} = \sqrt{\frac{1}{число\ точек} \sum_{i \subset цикл} осциллограмма[i]^2}$$

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:

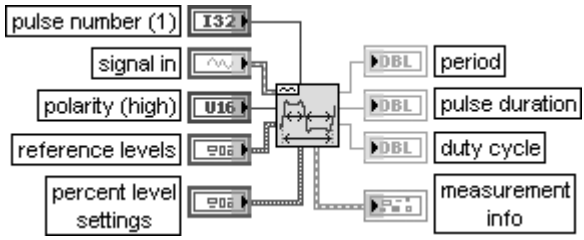
$$\textit{выброс перед фронтом} = 100\frac{(\textit{локальный максимум} - \textit{верхний уровень состояния})}{\textit{амплитуда}}$$

Выход **выброс после фронта** (**%**) (overshoot) содержит оценку локального максимума (минимума), следующего за возрастающим (спадающим) перепадом, выраженную

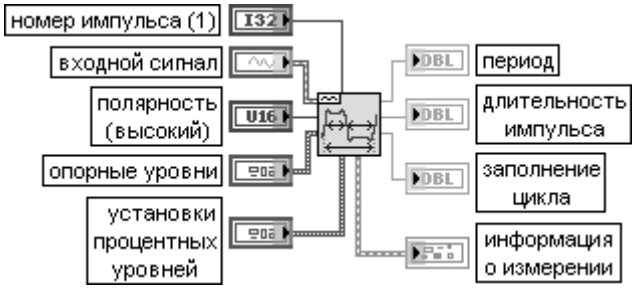
в процентах от амплитуды сигнала, определенной гистограммным методом. Если вход **полярность** определяет спадающий перепад, то **выброс после фронта** рассчитывается по следующему выражению:

$$выброс\ после\ фронта = 100\frac{(нижний\ уровень\ состояния - локальный\ минимум)}{амплитуда}$$

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.

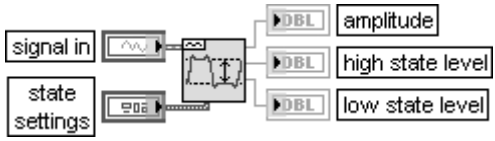
Выход **период** (period) возвращает время между соседними пересечениями **среднего опорного уровня** (mid ref level) в одном направлении в секундах. Интервал измерения включает импульс, определенный с помощью **номера импульса** (pulse number).

Выход **длительность импульса** (pulse duration) возвращает интервал времени в секундах между первыми двумя пересечениями **среднего опорного уровня** импульсом, заданным на входе **номер импульса**.

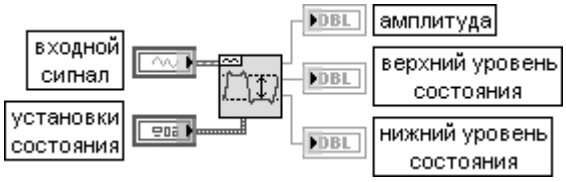
Выход **заполнение цикла** (duty cycle) рассчитывается с помощью следующего выражения:

$$заполнение\ цикла = 100\frac{(длительность\ импульса)}{период}$$

Amplitude and Levels



Амплитуда и уровни

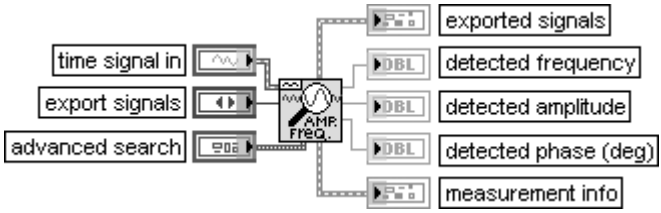


ВП возвращает **амплитуду** (amplitude), **верхний уровень состояния** (high state level) и **нижний уровень состояния** (low state level) осциллограммы или массива осциллограмм.

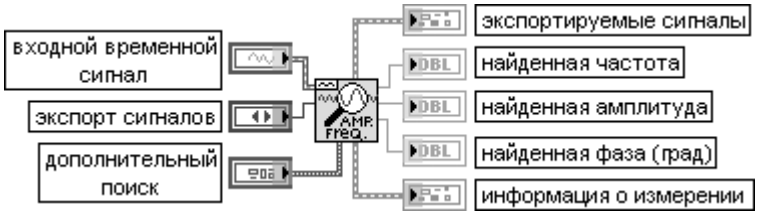
Выход **амплитуда** (amplitude) определяется как разность между **верхним и нижним уровнями состояния**.

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 о 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**.

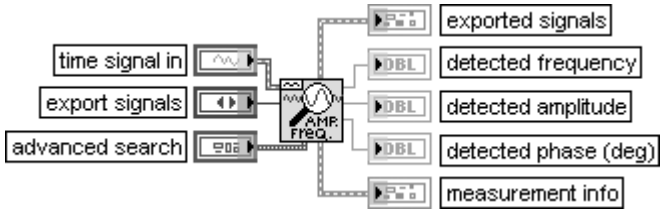
f0 представляет начальную частоту спектра, выраженную в герцах

df представляет разрешение по частоте, выраженное в герцах

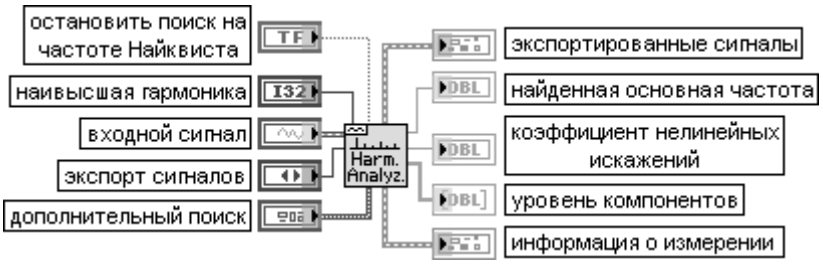
спектр (Ханн) дБ (dB Spectrum (Hann)) представляет спектр входного сигнала, выраженный в децибелах, после обработки сигнала окном Хана

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 возвращается основная частота, все значения амплитуд гармоник и коэффициент нелинейных искажений.

Вход **остановить поиск 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 равных половине 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.

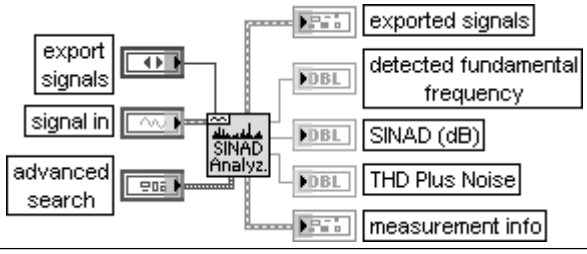
Вход **наивысшая гармоника** (highest harmonic) устанавливает наивысшую гармонику, включая основной тон, используемую для гармонического анализа. Например, для анализа по трем гармоникам на этом входе должно быть установлено значение 3.

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.

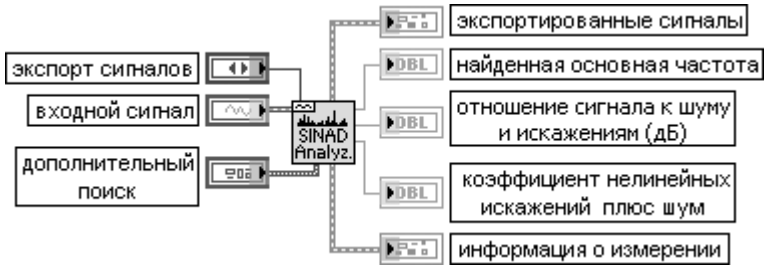
Выход **коэффициент нелинейных искажений** (**КНИ**) (THD) содержит измеренное значение коэффициента нелинейных искажений в диапазоне до **наивысшей гармоники**. **КНИ** определяется как отношение среднеквадратичного значения суммы гармоник к амплитуде основного тона. Для вычисления **КНИ** в процентах 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

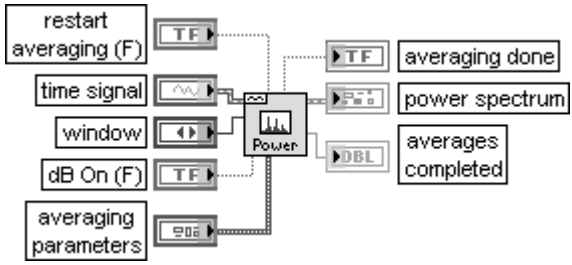


ВП принимает входной сигнал и выполняет полный **анализ отношения сигнала к шуму и искажениям** (Signal in Noise and Distortion (SINAD) analysis), включая измерение тона основной частоты. ВП возвращает основную частоту и отношение сигнала к шуму и искажениям в децибелах.

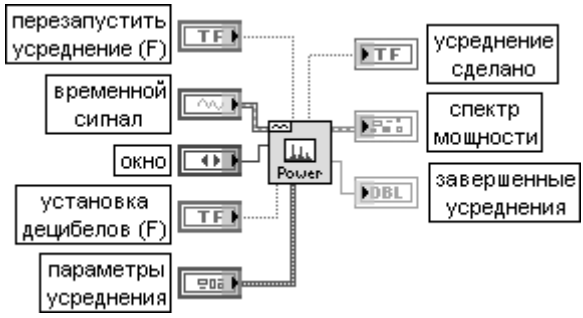
Выход **отношение сигнала к шуму и искажениям (дБ)** (SINAD (dB)) содержит измеренное значение отношения среднеквадратичного значения **входного сигнала** к разности среднеквадратичных значений **входного сигнала** и основного тона.

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 **окно** (window) задает одно из временных окон, приведенных в таблице.

0	**Прямоугольное** (Uniform)	3	**Блэкмана-Хэрриса** (Blackman-Harris)	6	**Плосковершинное** (Flat Top)
1	**Ханна** (Hanning)	4	**Точное Блэкмана** (Exact Blackman)	7	**Четырехзвенное Блэкмана-Хэрриса** (Four Term Blackman-Harris)
2	**Хэмминга** (Hamming)	5	**Блэкмана** (Blackman)	8	**Семизвенное Блэкмана-Хэрриса** (Seven Term Blackman-Harris)
				9	Low Sidelobe

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 усреднения**.

0	**линейный** (Linear)
1	**экспоненциальный** (Exponential) (по умолчанию)

число усреднений (number of averages) определяет число усреднений, которое используется при **среднеквадратичном** и **векторном усреднении**. 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:

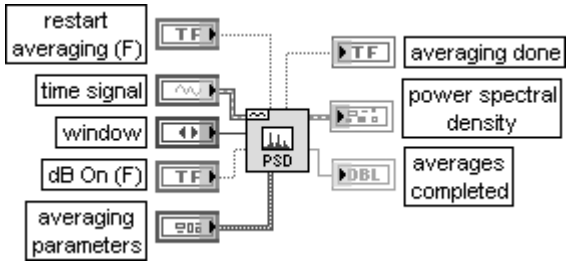
f0 является начальной частотой спектра, выраженной в герцах

df является частотным разрешением спектра, выраженным в герцах

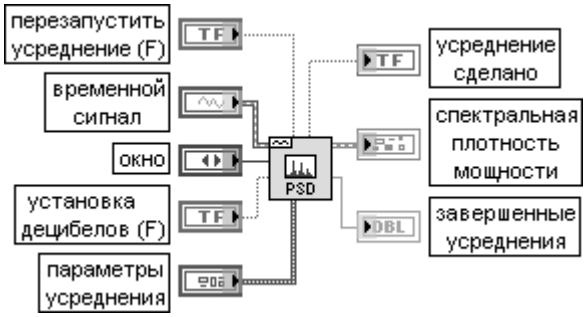
величина (magnitude) является величиной усредненного спектра мощности

Выход **завершенные усреднения** (averages completed) возвращает число выполненных к текущему времени усреднений

FFT Power Spectral Density

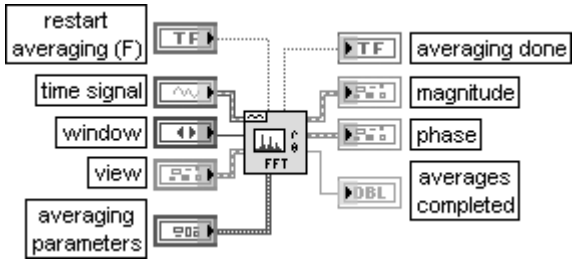


Спектральная плотность мощности БПФ

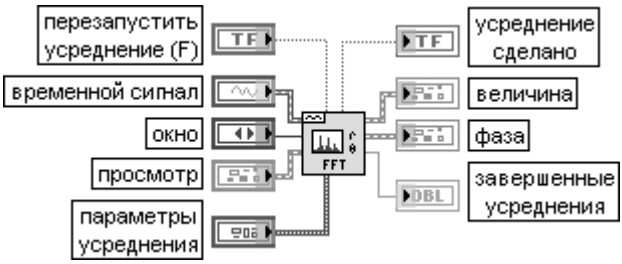


ВП рассчитывает среднюю спектральную плотность мощности **временного сигнала** (time signal) с помощью быстрого преобразования Фурье (БПФ)

FFT Spectrum (Mag-Phase)



Спектр БПФ (величина-фаза)



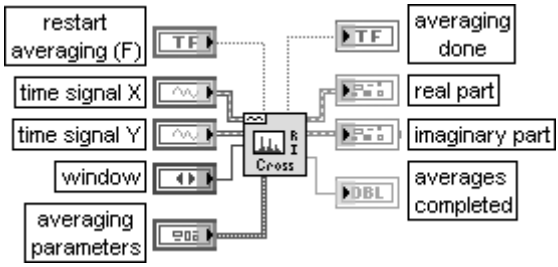
ВП рассчитывает усредненный БПФ спектр **временного сигнала** (time signal). Результаты БПФ возвращаются в виде амплитуды и фазы.
Вход **просмотр** (view) определяет параметры представления результатов, возвращаемых данным ВП.

установка децибелов (dB On) определяет отображение результатов в децибелах. По умолчанию установлено состояние ЛОЖЬ

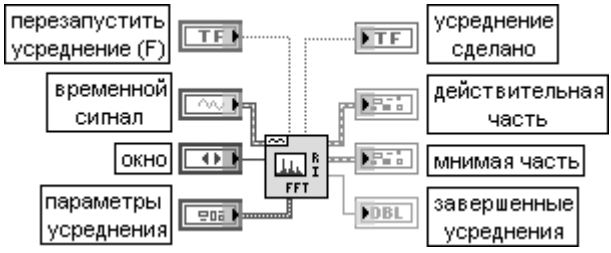
развертка фазы (unwrap phase) определяет развертку фазы. Развертка устраняет разрывы фазы при превышении значения 2π. По умолчанию установлено состояние ЛОЖЬ, означающее, что развертка фазы не производится

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)

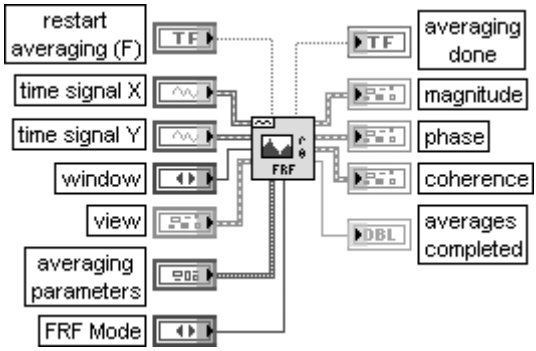


Спектр БПФ (действительная – мнимая часть)

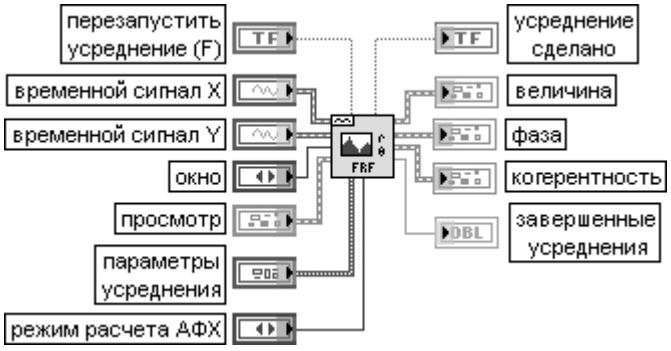


ВП рассчитывает усредненный БПФ спектр **временного сигнала** (time signal). Результаты БПФ возвращаются в виде действительной и мнимой частей

Frequency Response Function (Mag-Phase)



Функция частотного коэффициента передачи (величина – фаза)



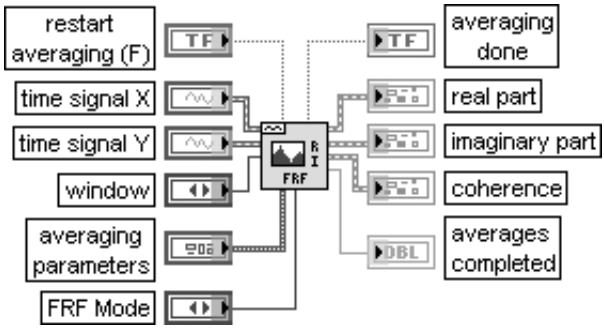
ВП рассчитывает частотный коэффициент передачи и когерентность на основе анализа входных сигналов. Результаты возвращаются в виде **величины** (magnitude), **фазы** (phase) и **когерентности** (coherence).

Как правило, **временной сигнал X** (time signal X) является стимулом, а **временной сигнал Y –** реакцией системы. Каждая временная осциллограмма соответствует единичному блоку БПФ.

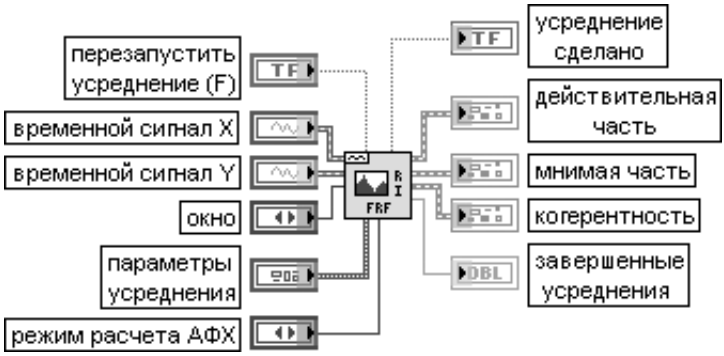
Вход **режим расчета амплитудно-фазовой частотной характеристики (АФХ)** (FRF mode) определяет способ расчета АФХ:

0	1	2
H1 (по умолчанию)	H2	H3
$A\Phi X_1(f) = \frac{\lvert X \cdot (f) Y(f) \rvert}{X^2(f)}$	$A\Phi X_2(f) = \frac{Y^2(f)}{\lvert X \cdot (f) Y(f) \rvert}$	$A\Phi X(f) = (A\Phi X_1(f) + A\Phi X_2(f))/2$

Frequency Response Function (Real-Im)

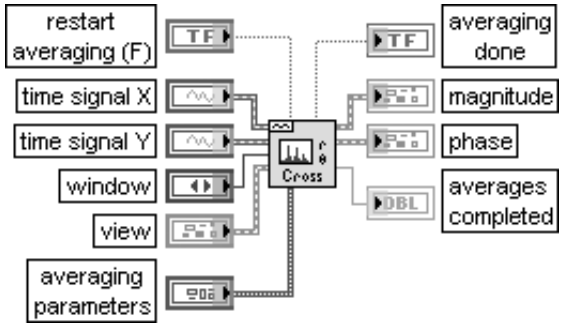


Функция частотного коэффициента передачи (действительная – мнимая часть)

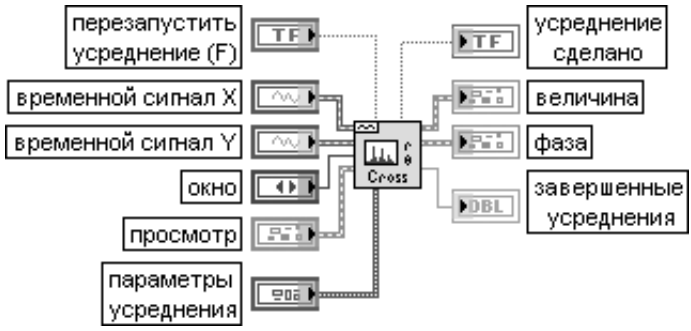


ВП рассчитывает частотный коэффициент передачи и когерентность на основе анализа входных сигналов. Результаты возвращаются в виде **действительной части** (real part), **мнимой части** (imaginary part) и **когерентности** (coherence)

Cross Spectrum (Mag-Phase)

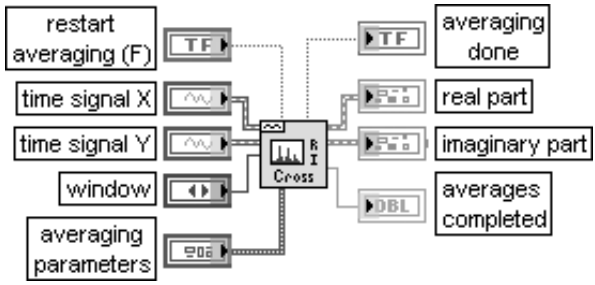


Кросс-спектр (величина – фаза)

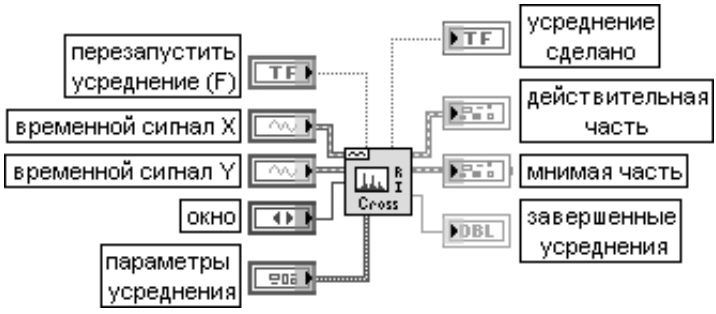


ВП рассчитывает усредненный кросс спектр мощности входных сигналов. Результаты возвращаются в виде **величины** и **фазы**

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)

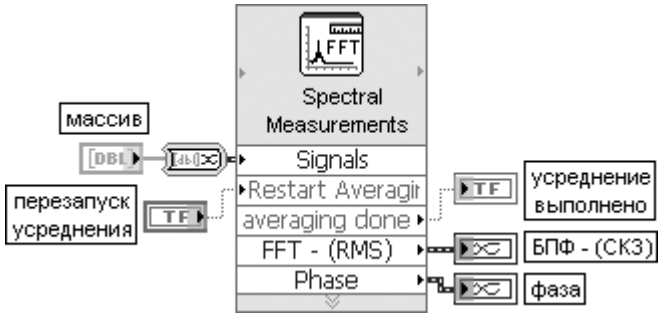


Рис. 5.8. Блок-диаграмма возможного подключения Экспресс-ВП

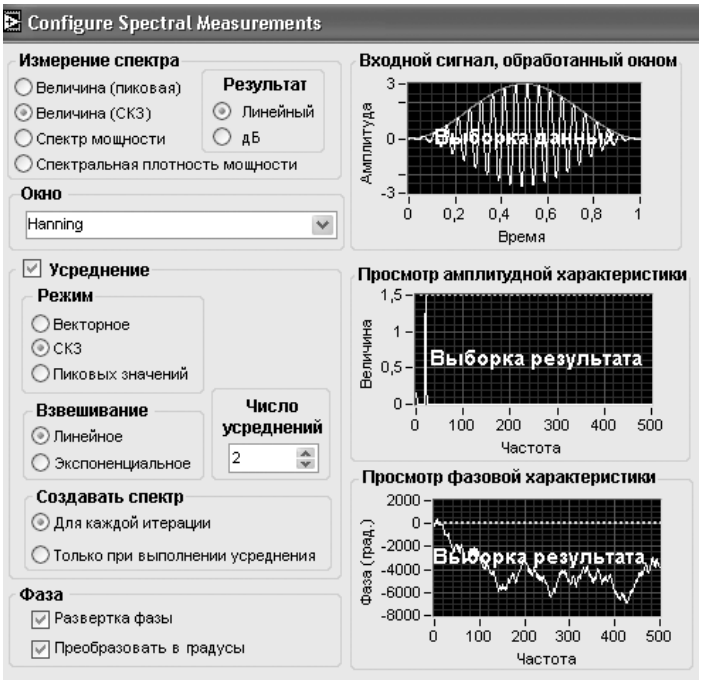


Рис. 5.9. Вид диалогового окна конфигурирования Экспресс-ВП ***Измерения спектра*** *(Spectral Measurements)*

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)

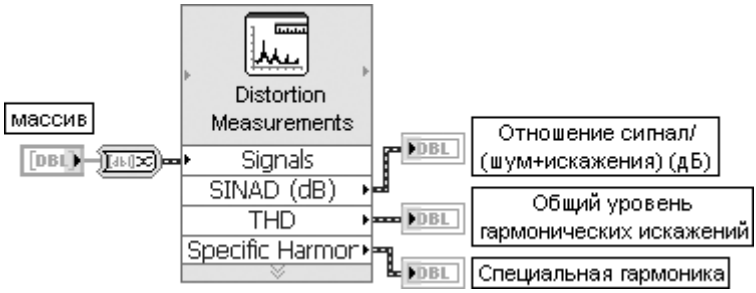


Рис. 5.10. Блок-диаграмма возможного подключения Экспресс-ВП

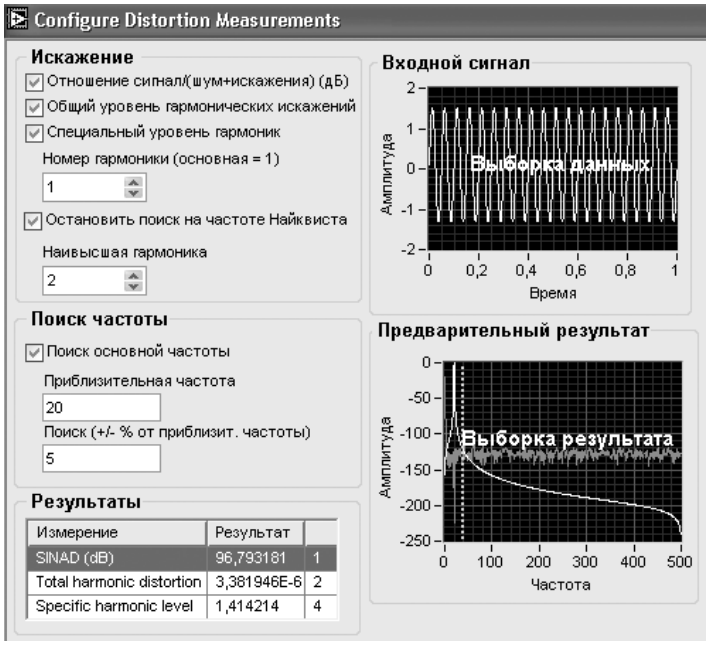


Рис. 5.11. Вид диалогового окна конфигурирования Экспресс-ВП ***Измерения искажений*** *(Distortion Measurements)*

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)

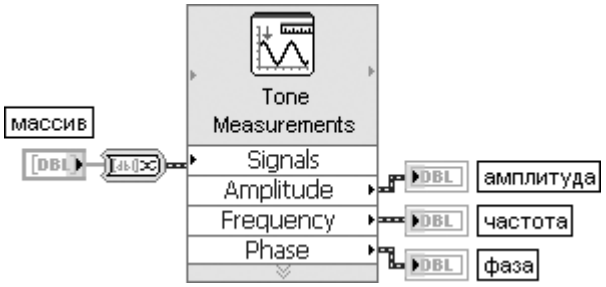


Рис. 5.12. Блок-диаграмма возможного подключения Экспресс-ВП

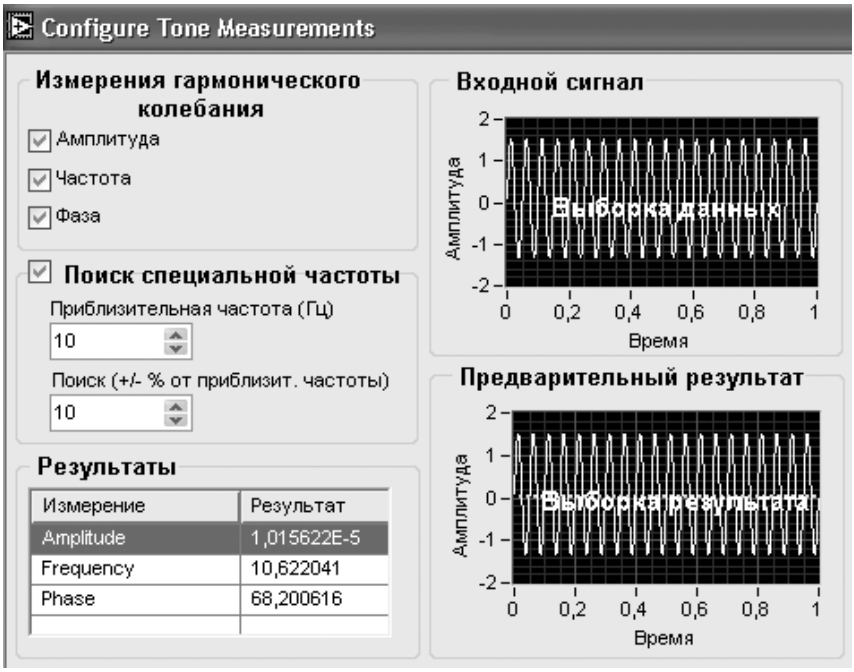


Рис. 5.13. Вид диалогового окна конфигурирования Экспресс-ВП ***Измерения гармонического колебания*** *(Tone Measurements)*

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)

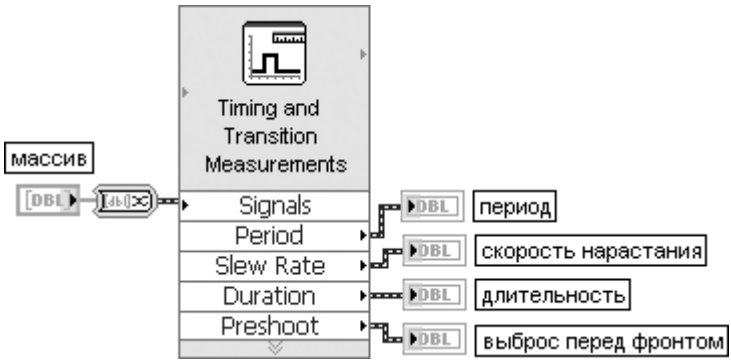


Рис. 5.14. Блок-диаграмма возможного подключения Экспресс-ВП

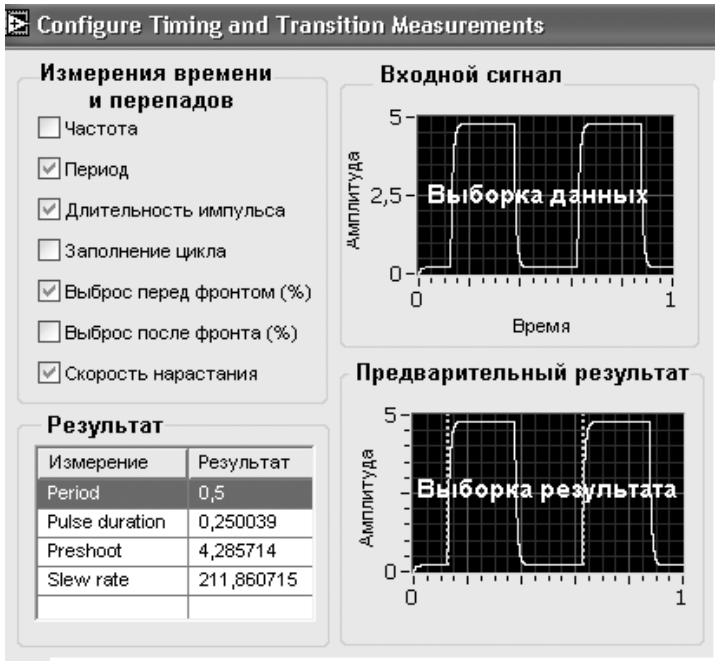


Рис. 5.15. Вид диалогового окна конфигурирования Экспресс-ВП
Измерения временных и переходных параметров
(Timing and Transition Measurements)

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)

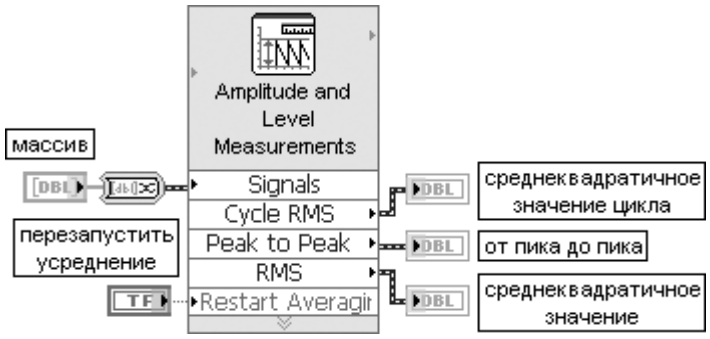


Рис. 5.16. Блок-диаграмма возможного подключения Экспресс-ВП

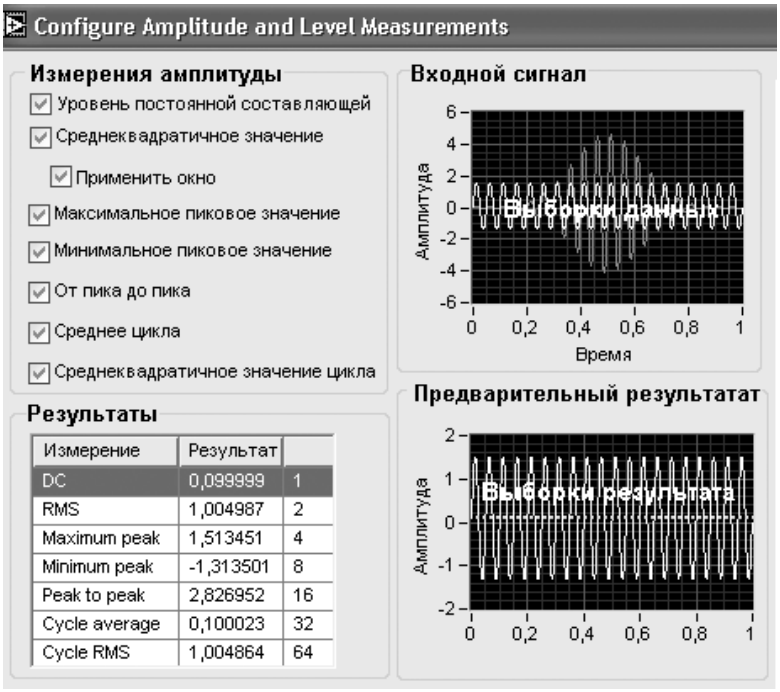


Рис. 5.17. Вид диалогового окна конфигурирования Экспресс-ВП ***Измерения амплитуды и уровня*** *(Amplitude and Level Measurements)*

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).

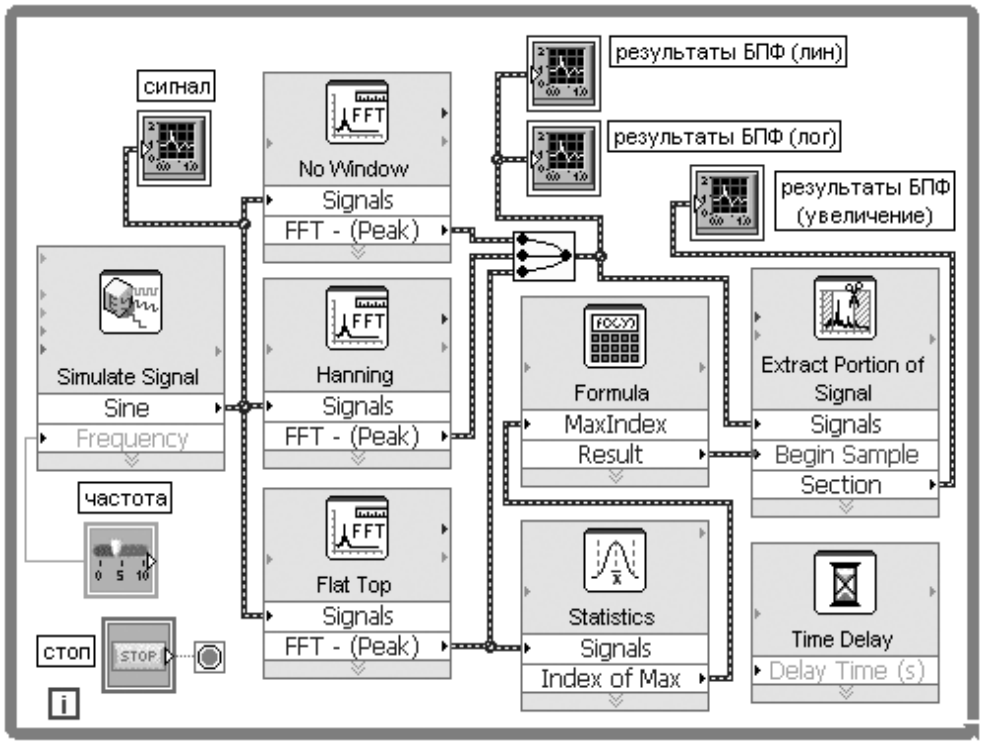
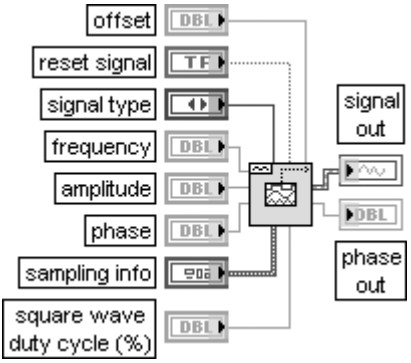


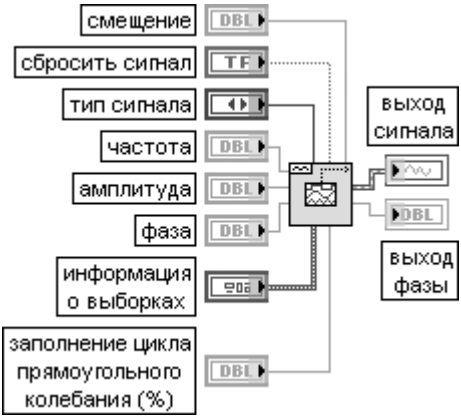
*Рис. 5.18. Блок-диаграмма ВП **Спектральные измерения** (Spectrum Measurements)*

5.1.3. 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:

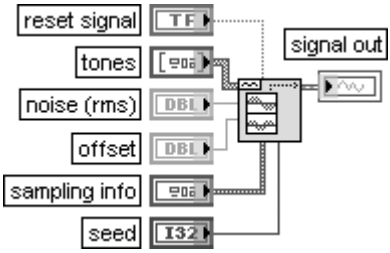
0	1	2	3
синусоидальная (Sine Wave) (по умолчанию)	треугольная (Triangle Wave)	прямоугольная (Square Wave)	пилообразная (Sawtooth Wave)

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 **сбросить сигнал** установлено значение ЛОЖЬ.
Вход **информация о выборках** (sampling info) содержит следующую информацию о выборках:

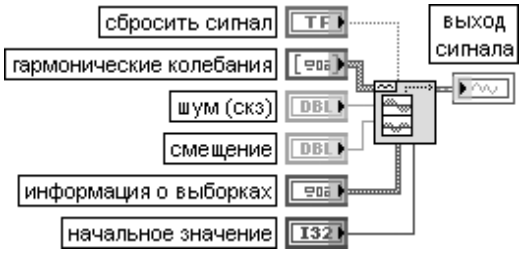
DBL	**Fs** является частотой выборок, выраженной числом выборок в секунду. По умолчанию значение равно 1000.
DBL	**#s** является числом выборок в осциллограмме. По умолчанию значение равно 1000

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.

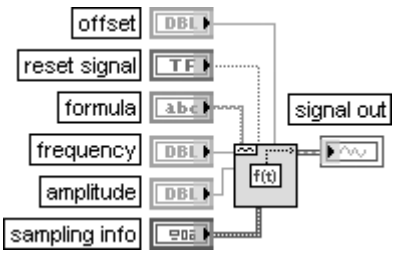
Вход **сбросить сигнал** (reset signal) в состоянии ИСТИНА устанавливает фазу каждого гармонического колебания равной значению соответствующего элемента управления **фаза**, находящегося в массиве кластеров **гармонические колебания** (tones), начальное значение – значению на соответствующем входе **начальное 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:

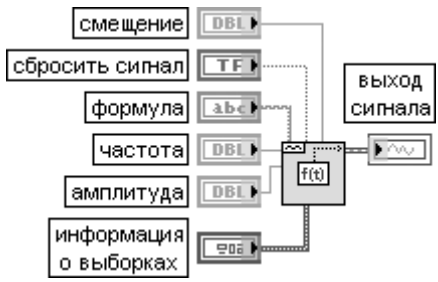
DBL	**частота** (frequency) определяет частоту синусоидального колебания в герцах
DBL	**амплитуда** (amplitude) определяет амплитуду синусоидального колебания
DBL	**фаза** (phase) определяет начальную фазу синусоидального колебания в градусах. По умолчанию значение равно 0

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).

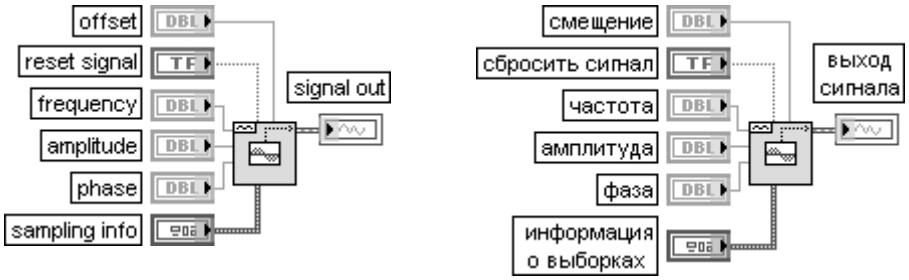
В следующей таблице приведены допустимые имена переменных.

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 3.1. В связи с этим пояснения к ним ограничены переводом наименований входов и выходов.

Sine Waveform

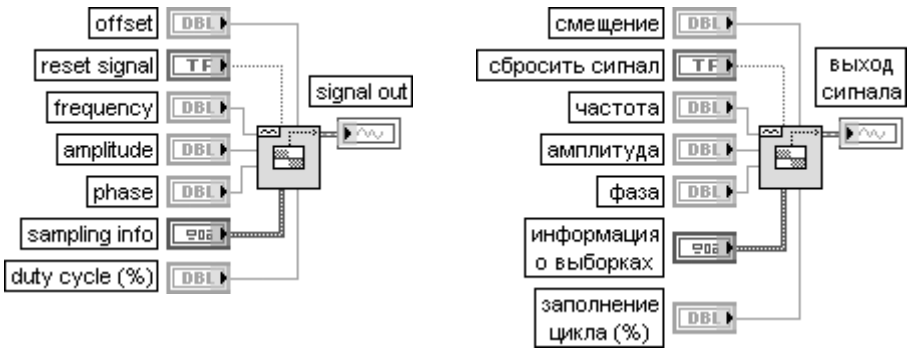
Осциллограмма с синусоидальным колебанием



ВП генерирует осциллограмму, содержащую синусоидальное колебание. Основным элементом данного ВП является ВП **Синусоидальное колебание** (Sine Wave)

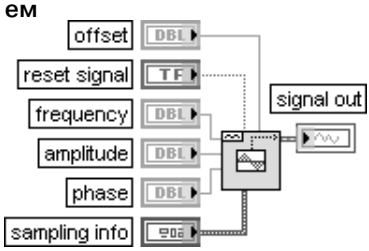
Square Waveform

Осциллограмма с прямоугольным колебанием

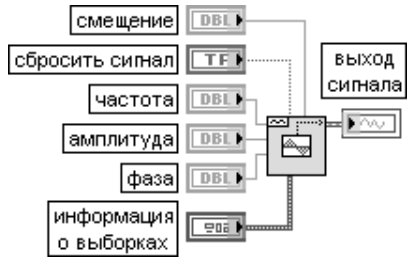


ВП генерирует осциллограмму, содержащую прямоугольное колебание. Основным элементом данного ВП является ВП **Прямоугольное колебание** (Square Wave)

Triangle Waveform

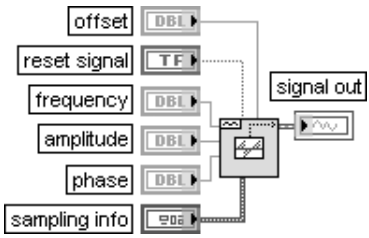


Осциллограмма с треугольным колебанием

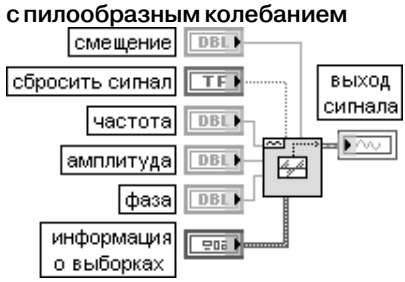


ВП генерирует осциллограмму, содержащую треугольное колебание. Основным элементом данного ВП является ВП **Треугольное колебание** (Triangle Wave)

Sawtooth Waveform



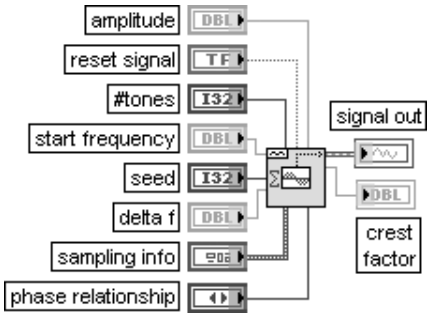
Осциллограмма с пилообразным колебанием



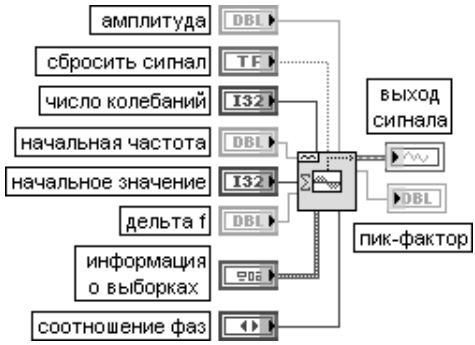
ВП генерирует осциллограмму, содержащую пилообразное колебание. Основным элементом данного ВП является ВП **Пилообразное колебание** (Sawtooth Wave)

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

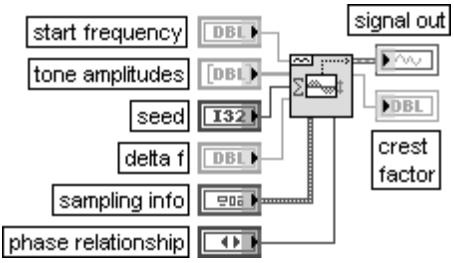


ВП генерирует осциллограмму, представляющую сумму заданного числа синусоидальных колебаний с целым числом периодов и случайной начальной фазой.
Вход **амплитуда** (amplitude) определяет значение, к которому нормируется сумма всех гармонических колебаний, и, соответственно, наибольшее абсолютное значение, 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.

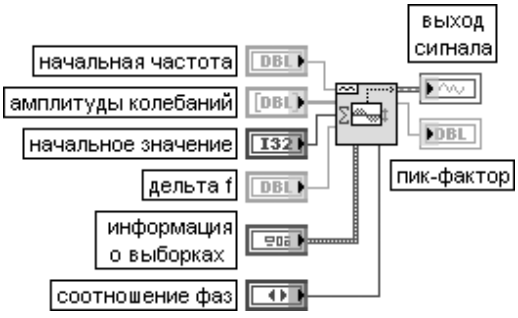
0	**случайное** (Random) – каждое значение фазы выбирается случайно в диапазоне от 0 до 360 градусов
1	**линейное** (Linear) – дает лучшее отношение пикового и среднеквадратичного значений, но может вызвать появление в сигнале периодических компонентов с периодом, равным длительности осциллограммы

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



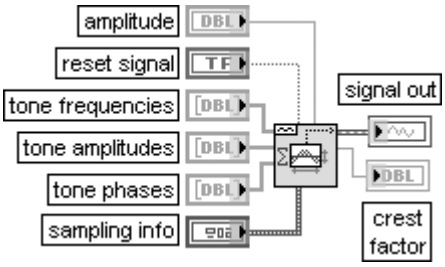
Основной многотональный генератор с заданными амплитудами



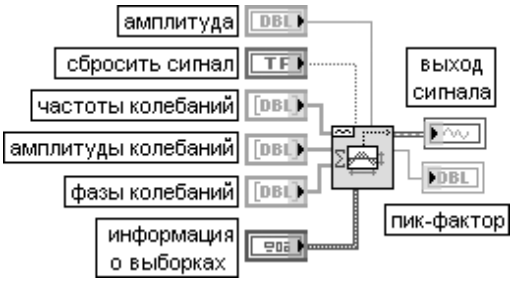
Данный ВП выполняет ту же функцию, что и рассмотренный выше ВП **Основной многотональный генератор** (Basic Multitone), отличаясь от него наличием входа **амплитуды колебаний** (tone amplitudes).

Вход **амплитуды колебаний** (tone amplitudes) представляет массив, в котором каждый элемент определяет амплитуду колебания, а размер массива – число генерируемых колебаний

Multitone Generator

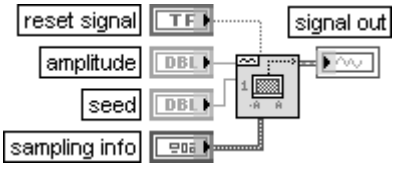


Многотональный генератор

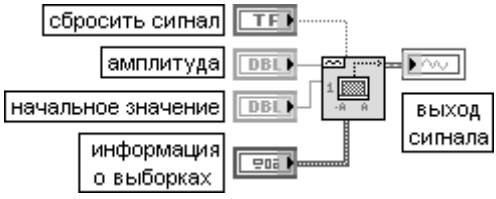


ВП генерирует осциллограмму, представляющую сумму синусоидальных колебаний с заданной частотой, амплитудой и фазой

Uniform White Noise Waveform

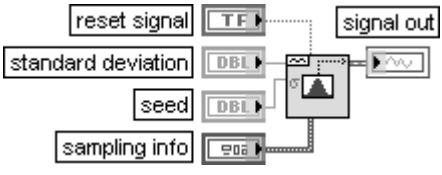


Осциллограмма с равномерным белым шумом



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

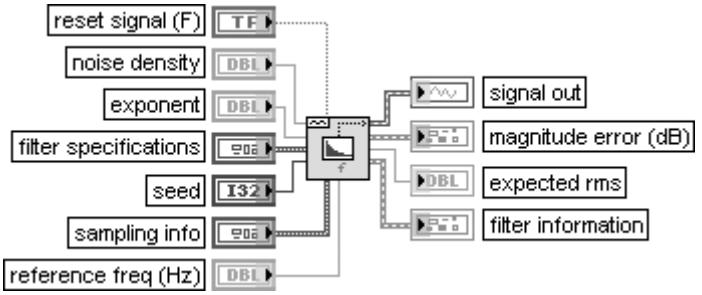
ВП генерирует псевдослучайный гауссовский белый шум, имеющий статистические параметры (0,**s**), где **s** является абсолютным значением заданного стандартного отклонения

Periodic Random Noise Waveform

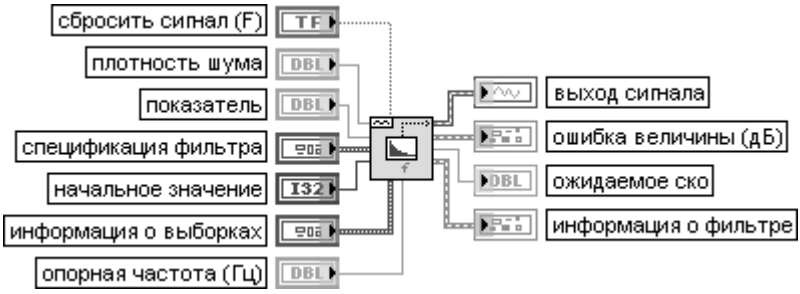
Осциллограмма с периодическим случайным шумом

ВП генерирует осциллограмму, содержащую периодический случайный шум

Inverse f Noise Waveform



Осциллограмма шума 1/f

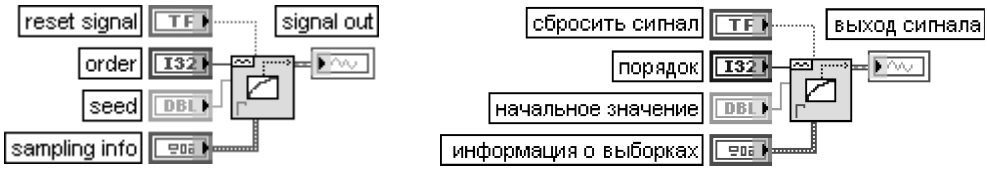


ВП генерирует осциллограмму шума, у которого спектральная плотность мощности обратно пропорциональна частоте в заданном спектральном диапазоне.
Генерация осуществляется путем пропускания белого гауссовского шума через цифровой фильтр, у которого квадрат частотной характеристики изменяется по закону 1/(частота^**показатель**). 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**.

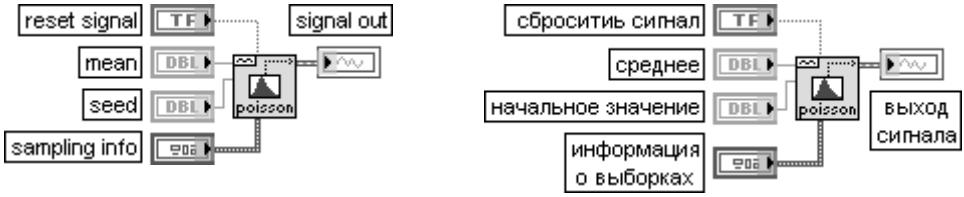
Выход **ожидаемое ско** (expected rms) возвращает ожидаемое среднеквадратичное значение генерируемой шумовой осциллограммы

Gamma Noise Waveform **Осциллограмма Гамма шума**



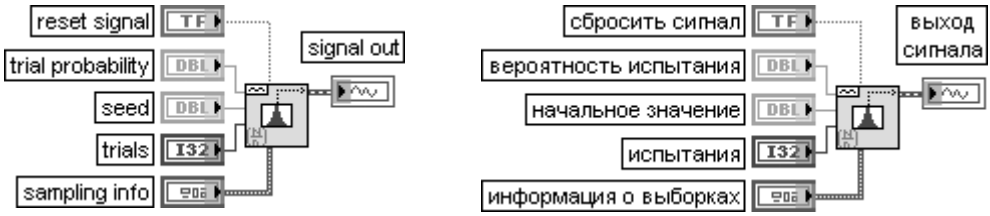
ВП генерирует псевдослучайный набор значений, которые представляют интервалы времени ожидания заданного числа событий пуассоновского процесса с единичным средним. Вход **порядок** (order) определяет число событий. По умолчанию **порядок** равен 1

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**

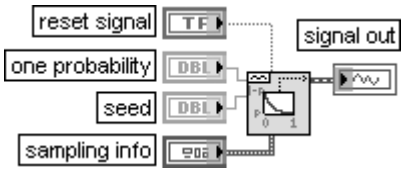


ВП генерирует псевдослучайную последовательность с биномиальным амплитудным распределением, значения которой представляют число реализаций событий, заданных вероятностью совершения событий и числом испытаний.

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



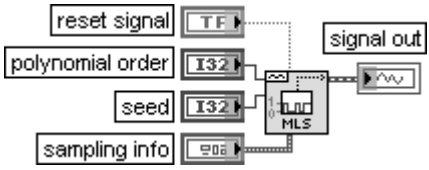
Осциллограмма шума Бернулли

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 два 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)

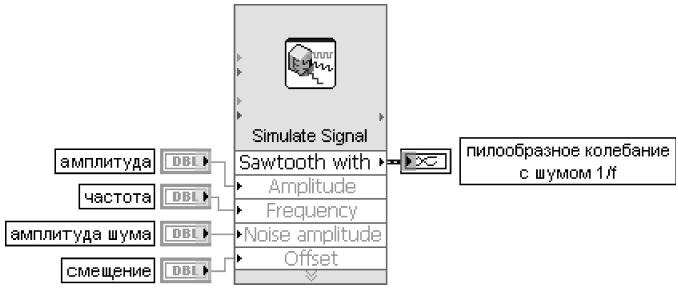


Рис. 5.19. Блок-диаграмма возможного подключения Экспресс-ВП

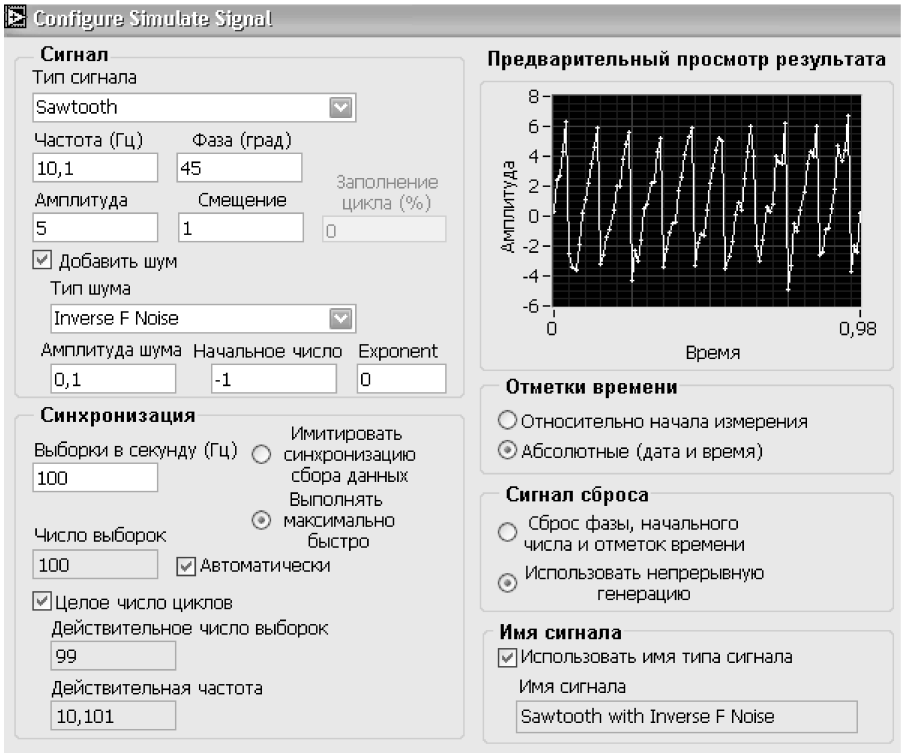


Рис. 5.20. Вид диалогового окна конфигурирования Экспресс-ВП ***Имитировать сигнал*** *(Simulate Signal)*

Экспресс-ВП имитирует синусоидальное, прямоугольное, треугольное, пилообразное колебания 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 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)

5.2. Функции сбора данных DAQmx

ВП и функции сбора данных NI-DAQmx (рис. 5.21) относятся к следующему поколению драйверов NI-DAQ.

Во введении к главе 5 было отмечено, что основным элементом NI-DAQmx является задача. Также было отмечено, что задача может быть создана как в прог-

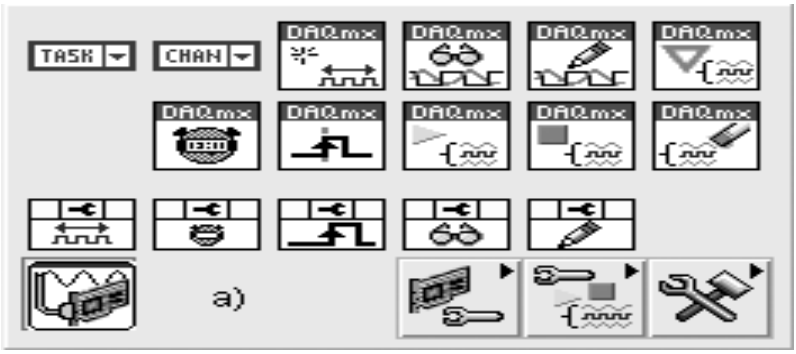


Рис. 5.21. Вид основной (а) и одной из дополнительных подпалитр (б) функций сбора данных DAQmx

рамме MAX, так и в LabVIEW. В LabVIEW задача может быть создана несколькими способами:

- путем размещения элемента управления **Имя задачи DAQmx** (DAQmx Task Name) на лицевой панели или константы с таким же именем (рис. 5.21а) 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;
- 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;
- путем выбора пункта **Создать задачу DAQmx** (Create DAQmx Task) в меню **Конфигурировать** (Configure) при запуске LabVIEW;
- путем выбора шаблона **Сбор данных с помощью NI-DAQmx** (Data Acquisition with NI-DAQmx) в меню **Новый ⇒ Новый DAQ** (New ⇒ New DAQ) при запуске LabVIEW.

При создании задачи как в MAX, так и в LabVIEW открывается диалоговое окно **Создать новую** (Create New). На рис. 5.23 показан вид диалогового окна при его открытии из MAX. С помощью набора кнопок в правой части окна производится выбор типа измерения или генерации сигнала. Набор основных типов задач включает **Аналоговый ввод** (Analog Input), **Аналоговый вывод** (Analog Output), **Вход счетчика** (Counter Input), **Выход счетчика** (Counter Output) и **Цифровой ввод/вывод** (Digital I/O). Иерархия типов задач измерения и генерации сигналов приведена на рис. 5.22.

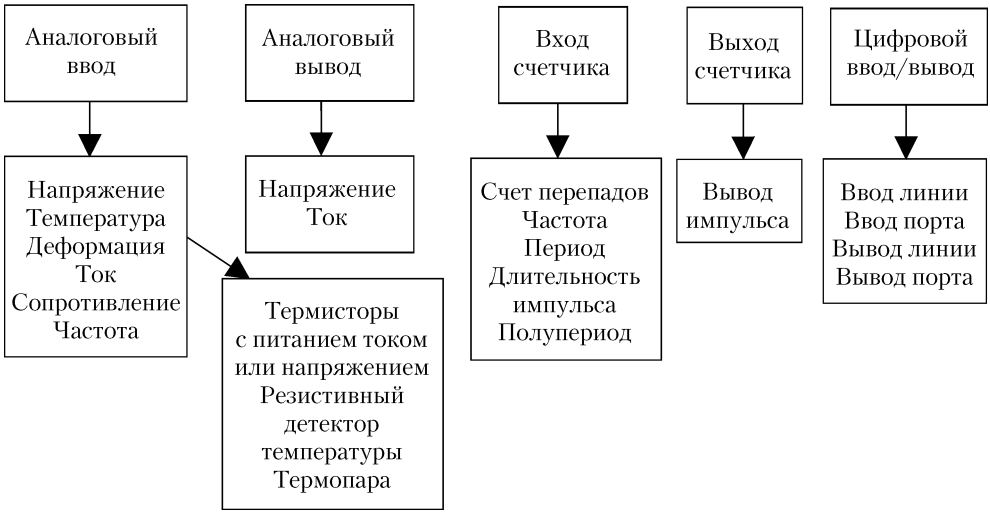


Рис. 5.22. Иерархия типов задач измерения и генерации сигнала

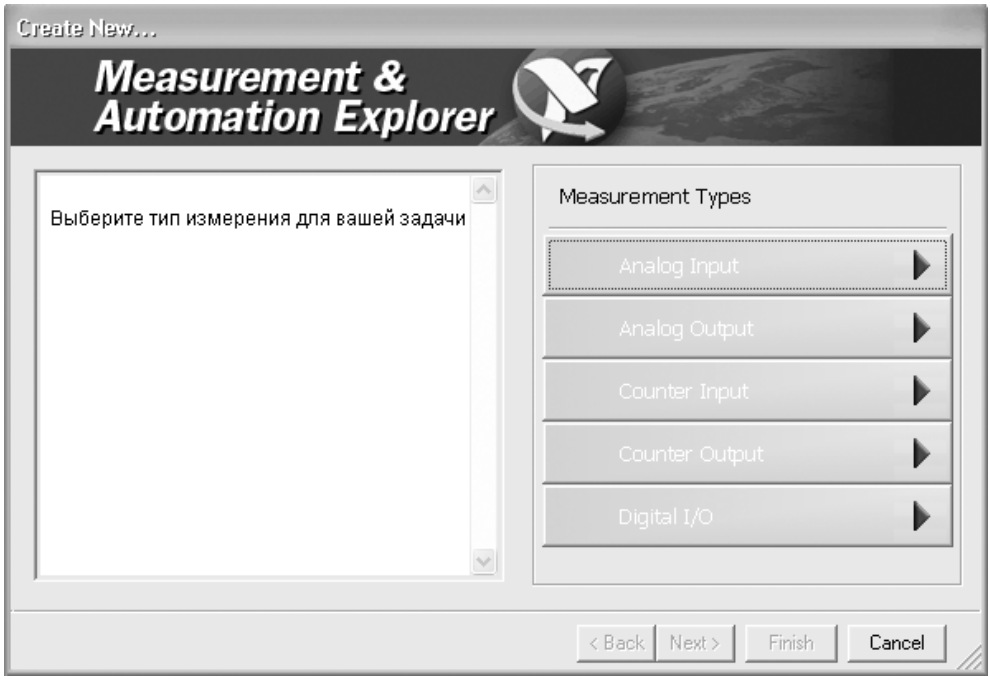


Рис. 5.23. Вид диалогового окна создания новой задачи

После выбора задачи производится выбор физического канала (каналов) из списка установленных (рис. 5.24).

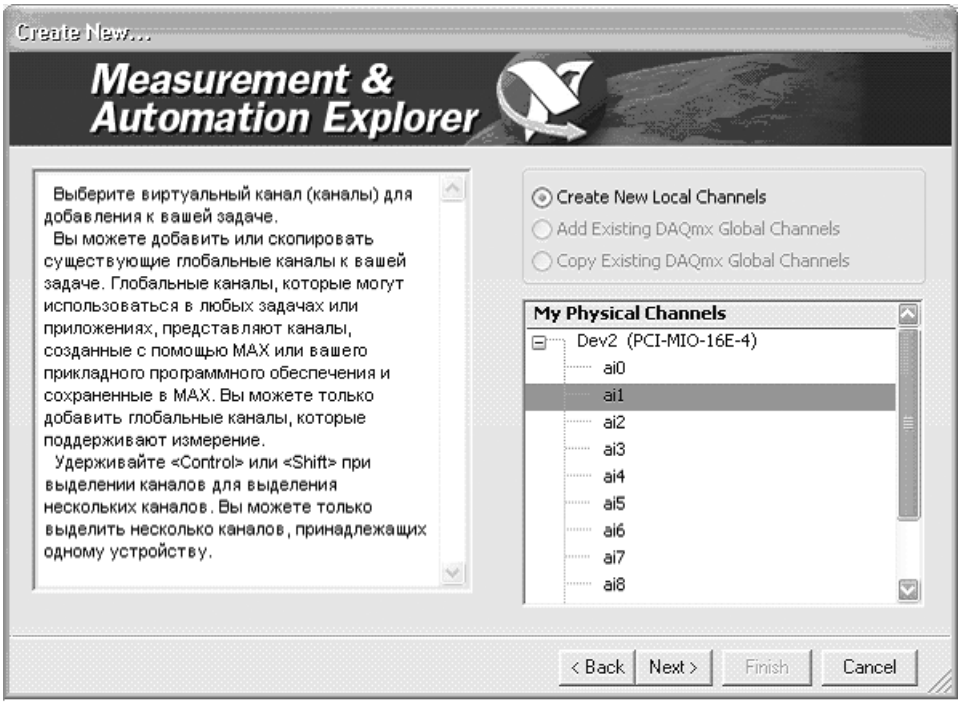


Рис. 5.24. Вид окна выбора физического канала

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. 5.25 показан вид данного раздела для задачи измерения температуры с помощью **резистивного детектора температуры** (Resistance temperature detector (RTD)).
2. Раздел **Синхронизация задачи** (Task Timing). На рис. 5.26 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. 5.27 показан возможный вид раздела с параметрами 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).

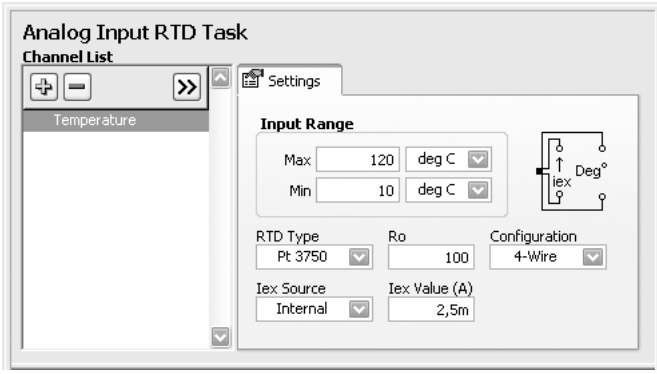


*Рис. 5.25. Вид раздела **Установки** диалогового окна настройки параметров физического канала*

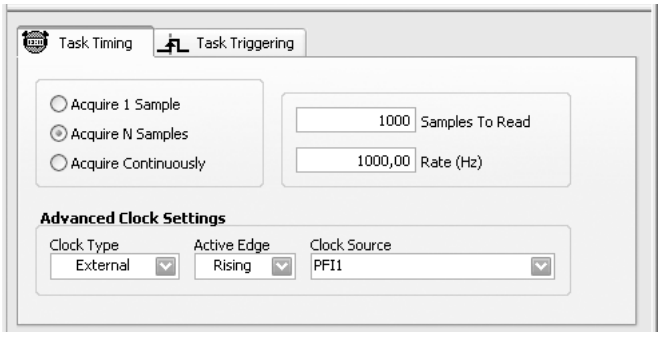


*Рис. 5.26. Вид раздела **Синхронизация задачи***

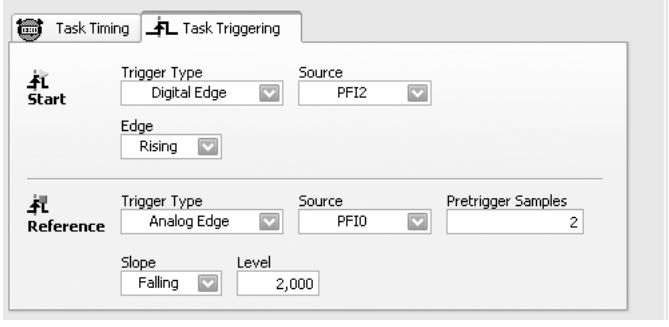


*Рис. 5.27. Вид раздела **Запуск задачи***

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. 5.23–5.27), только вместо кнопки **Сохранить задачу** после завершения ее конфигурирования 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. 5.28.

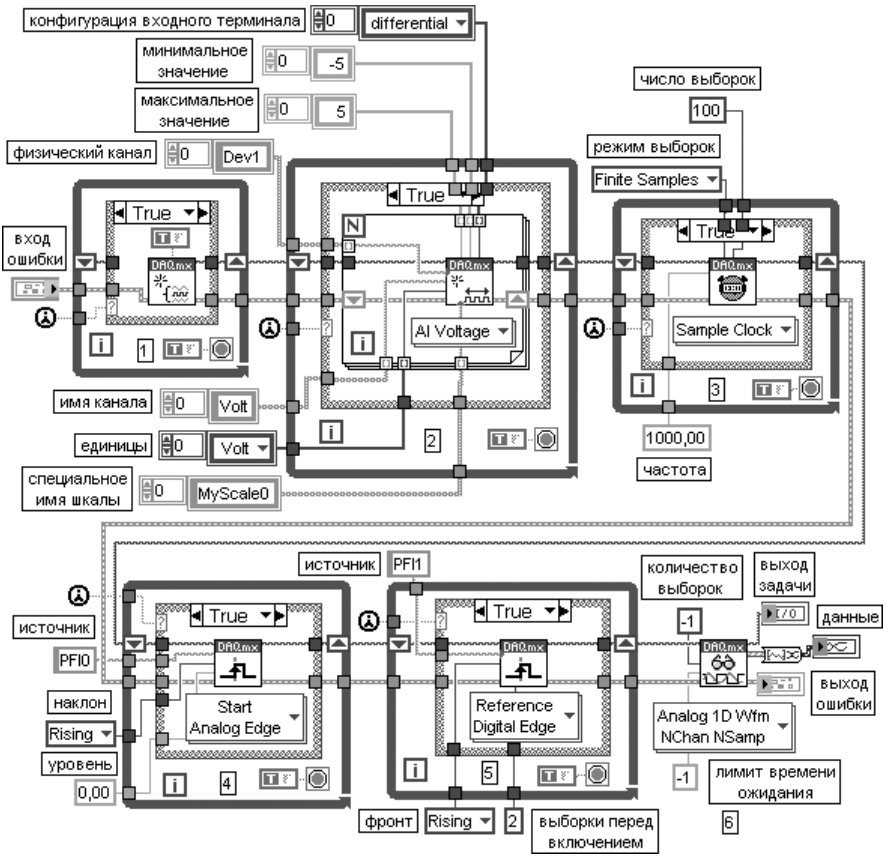


Рис. 5.28. Блок-диаграмма ВП измерения переменного напряжения

Из блок-диаграммы видно, что каждая из функций создания и конфигурирования задачи, заключенная в структуру **Вариант** и 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. 5.29):

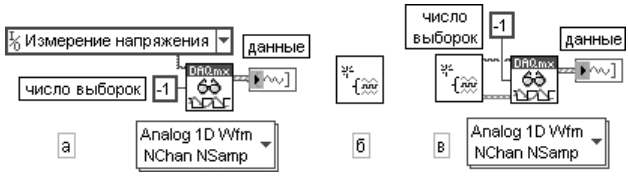


Рис. 5.29. Варианты кода ВП измерения напряжения

На рис. 5.30 приведена блок-диаграмма подприбора, изображенного на рис. 5.29 б.

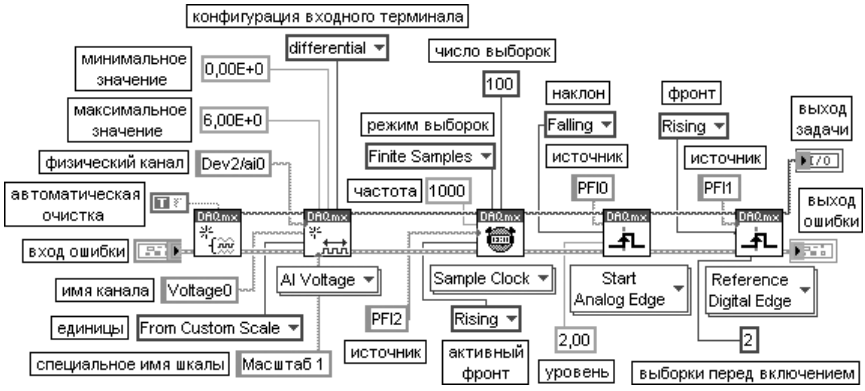


Рис. 5.30. Блок-диаграмма подприбора, изображенного на рис. 5.29б

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.

Аналогичную структуру в виде цепочки ВП из палитры DAQmx имеют ВП из набора примеров NI Example Finder LabVIEW. На рис. 5.31 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).

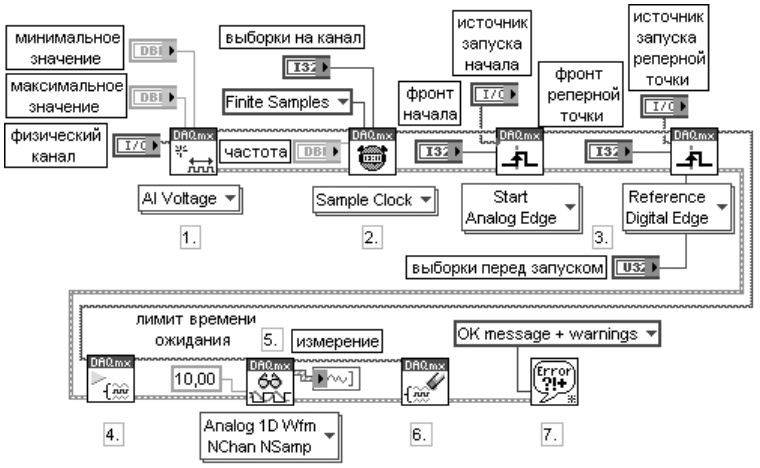


Рис. 5.31. Блок-диаграмма модернизированного ВП Acq&Graph Voltage-Int Clk-Dig Start&Ref

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;

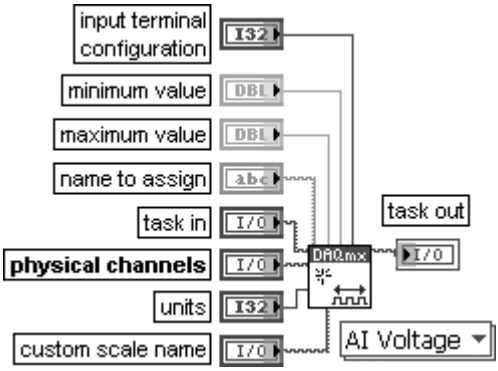
5) используется ВП **Читать DAQmx** для измерения совокупности выборок из N каналов платы сбора данных. Лимит времени ожидания устанавливается так, что в случае его превышения будет возвращаться ошибка;

6) вызывается ВП **Очистить задачу**;

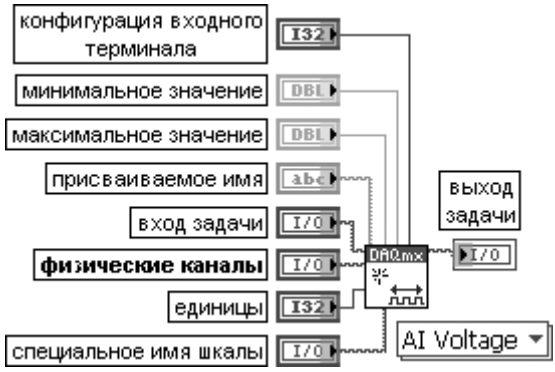
7) для отображения возможной ошибки используется диалоговое окно.

Блок-диаграммы, приведенные на рис. 5.28–5.31, показывают программный способ создания задачи с помощью ВП **Создать виртуальный канал DAQmx** (DAQmx Create Virtual Channel) или ВП **Создать задачу DAQmx** (DAQmx Create Task). В последующих таблицах более подробно рассмотрены функции из основной палитры **Сбор данных DAQmx** (DAQmx – Data Acquisition) и подпалитры **Дополнительные опции задачи DAQmx** (DAQmx Advanced Task Options) (рис. 5.21).

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. 5.32. Как видно из блок-диаграммы, для выполнения конфигурации канала используется узел свойства **Канал DAQmx** (DAQmx Channel). Перечень свойств изменяется в зависимости от вида измерения. Свойства **Канал DAQmx** являются подклассом свойств **DAQmx**.

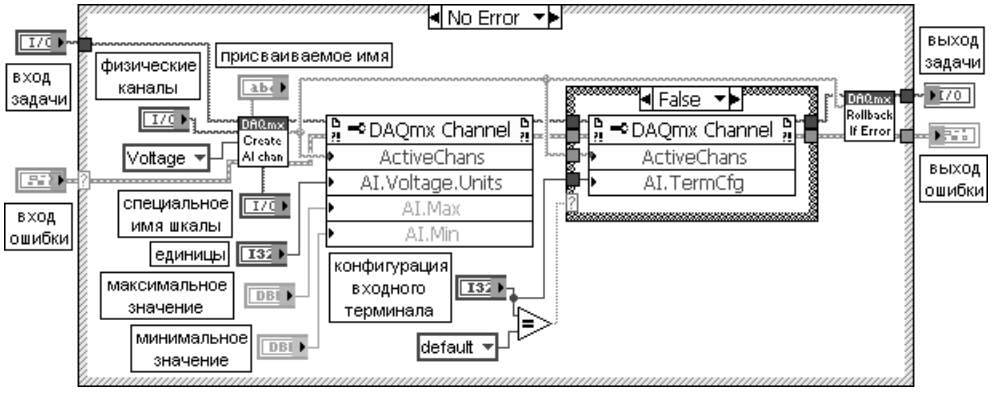
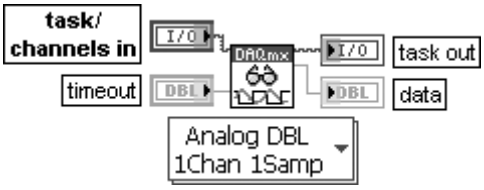
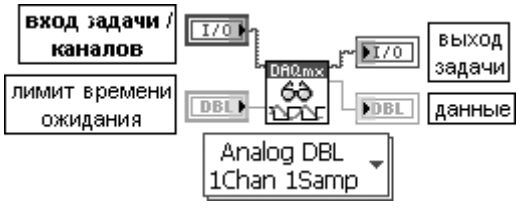


*Рис. 5.32. Блок-диаграмма ВП **Создать виртуальный канал DAQmx***

DAQmx Read



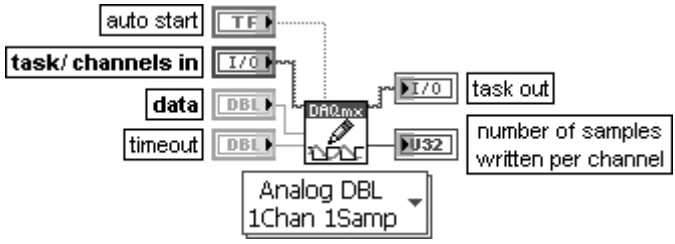
Читать DAQmx



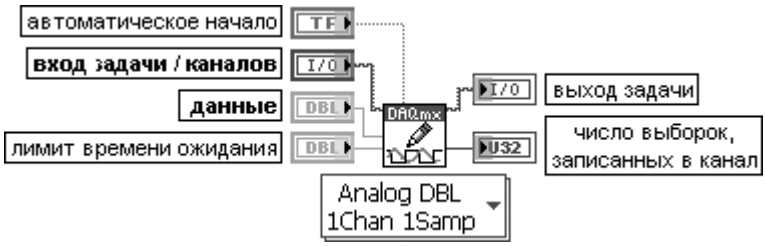
ВП считывает выборки из задачи или каналов, определенных пользователем. Реализация этого полиморфного ВП предусматривает выбор формата возвращаемых выборок,

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



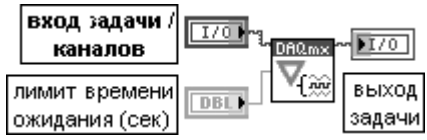
Записать в DAQmx



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 задание не выполнено

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

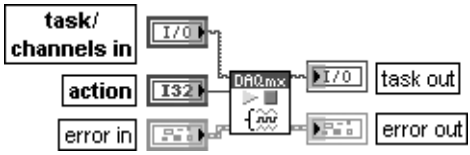
В следующих таблицах рассмотрены функции из подпалитры **Дополнительные опции задачи DAQmx** (DAQmx Advanced Task Options).

DAQmx Create Task **Создать задачу DAQmx**

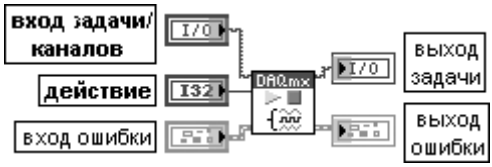
ВП создает **задачу** (task) и добавляет **виртуальные каналы** (virtual channels) к этой задаче, если они определены на входе **глобальные виртуальные каналы** (global virtual channels). Если определена **задача для копирования** (task to copy), то этот ВП добавляет любые виртуальные каналы к копии 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



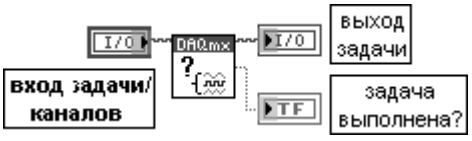
ВП изменяет состояние задачи в соответствии с **действием** (action), заданным пользователем.
Вход **действие** (action) определяет, как необходимо изменить состояние задачи.

Прервать (abort) (6)	Прерывает выполнение задачи
Совершить (commit) (3)	Максимально возможно программирует аппаратуру в соответствии с конфигурацией задачи
Резервировать (reserve) (4)	Резервирует ресурсы аппаратуры, необходимые для задачи. Никакие другие задачи не могут резервировать эти ресурсы
Отменить 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 **отменить резервирование** (unreserved) или **прервать** (abort)

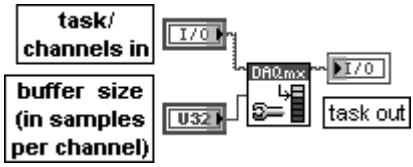
DAQmx Is Task Done

Задача DAQmx выполнена?



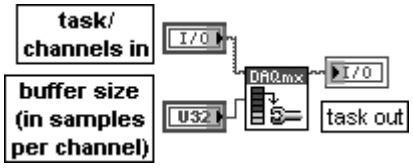
ВП запрашивает статус задачи и возвращает информацию о завершении ее выполнения. Этот ВП используется для обеспечения гарантии того, что заданная операция выполнена перед остановкой задачи

DAQmx Configure Input Buffer **Конфигурировать входной буфер DAQmx**



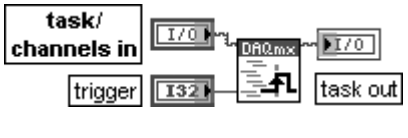
ВП аннулирует автоматическое назначение входного буфера, выполняемое NI-DAQmx. Вход **размер буфера (в выборках на канал)** (buffer size (in samples per channel)) определяет число выборок, которые буфер может содержать для каждого канала в задаче. При нулевом значении буфер не создается. Такое значение позволяет выполнить операцию с аппаратной синхронизацией без использования буфера

DAQmx Configure Output Buffer **Конфигурировать выходной буфер DAQmx**



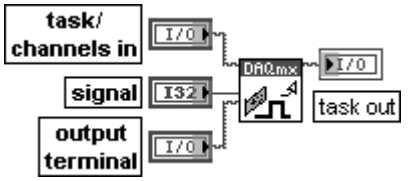
ВП аннулирует автоматическое назначение выходного буфера, выполняемое NI-DAQmx

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.

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 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. 5.33–5.35 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. 5.33 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).

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 бу-**

фера (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.

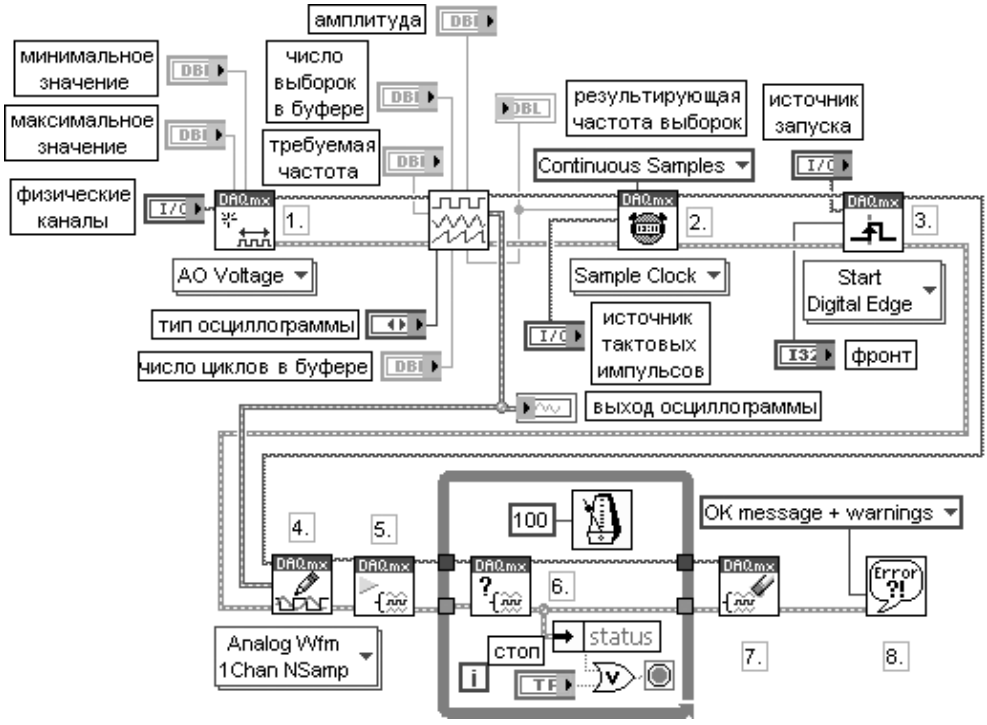


Рис. 5.33. Блок-диаграмма модернизированного ВП Cont Gen Voltage Wfm-Ext Clk-Dig Start

На рис. 5.34 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.

В процессе выполнения данного ВП реализуются следующие шаги.

1. Создается канал **Вход счетчика** для измерения частоты. Параметр **фронт** используется для выбора 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.

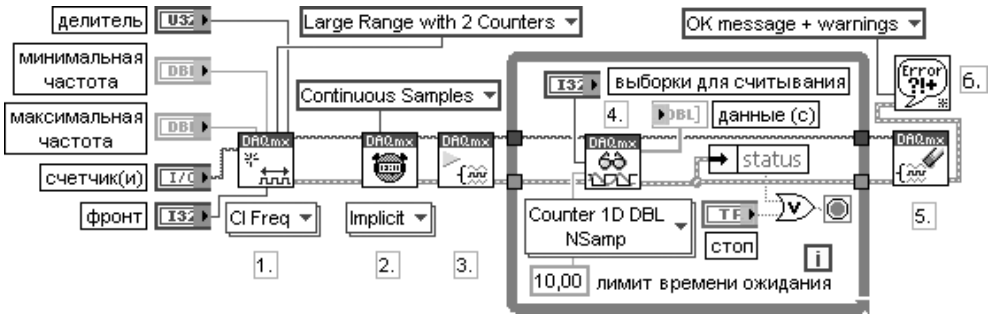


Рис. 5.34. Блок-диаграмма ВП Meas Dig Freq-Buffered-Cont-Large Range 2

На рис. 5.35 приведена блок-диаграмма ВП **Считать цифровой канал** (Read Dig Chan) из библиотеки **Считать значения** (Read Values) набора примеров NI Example Finder.

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 единич-

ную выборку цифровых данных по запросу, поэтому нет необходимости в тайм-ауте.

4. Для просмотра считываемых данных в виде единственного числа используется функция **Логический массив в число**.
5. Вызывается ВП **Очистить задачу**.
6. Для отображения возможной ошибки используется диалоговое окно.

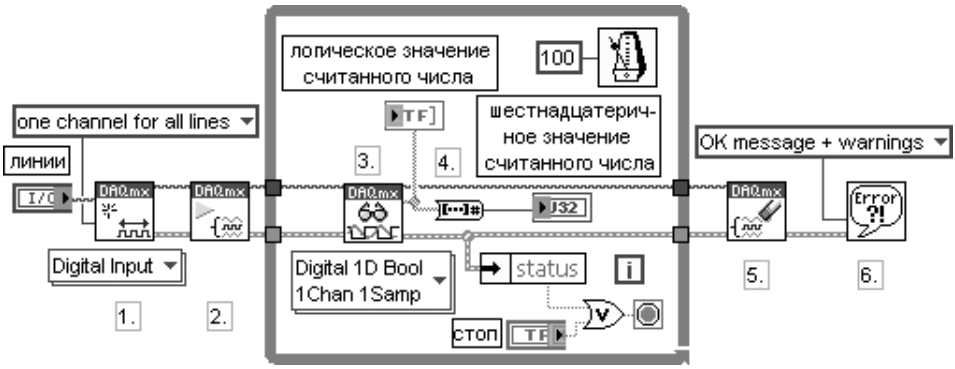


*Рис. 5.35. Блок-диаграмма ВП **Считать цифровой канал** (Read Dig Chan)*

5.3. 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 2.3.

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. 5.36.

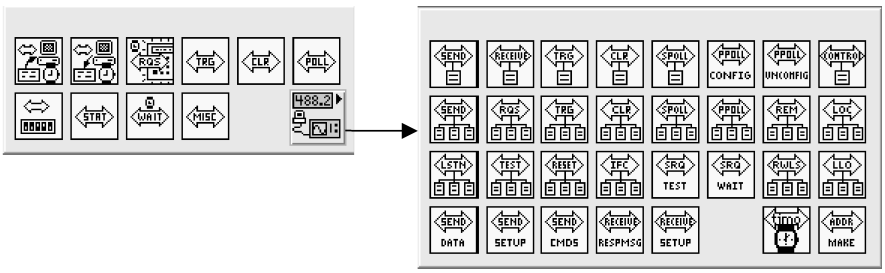
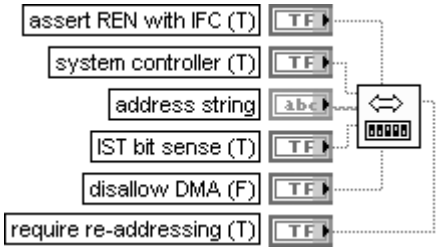


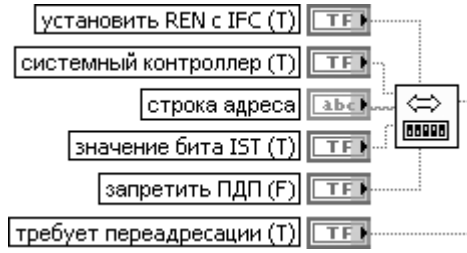
Рис. 5.36. Вид палитры функций GPIB и подпалитры функций 488.2

Рассмотрение функций интерфейса канала общего пользования ограничено функциями из палитры GPIB.

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.

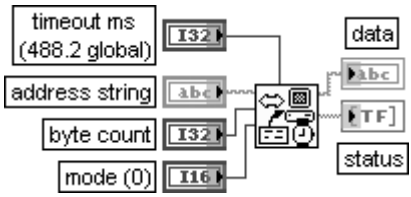
Если LabVIEW использует несколько контроллеров GPIB, то для указания используемого функцией контроллера перед строкой адреса указывается префикс в виде **ID:адрес** (или **ID:** если нет необходимости указания адреса). Если ID контроллера отсутствует, то функция по умолчанию относится к контроллеру (или 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



Чтение 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 адрес, 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 следующей 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.

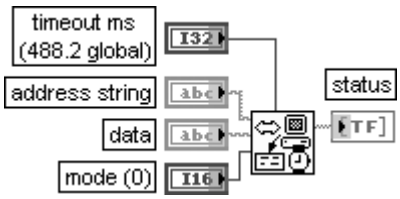
0	Отсутствие символа EOS. Режим завершения по концу строки отключен
1	Символом EOS является CR (Возврат каретки). Чтение завершается по сигналу линии EOI, по счетчику байтов или по символу CR
2	Символом EOS является LF (Перевод строки). Чтение завершается по сигналу линии EOI, по счетчику байтов или 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



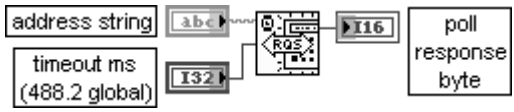
Запись GPIB

Функция записывает **данные** (data) в GPIB-устройство, определяемое **строкой адреса** (address string).
Вход **режим** (mode) определяет вариант завершения функции **Запись GPIB**.

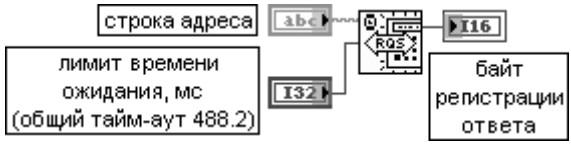
0	Послать EOI с последним символом строки
1	Добавить CR к строке и послать EOI с CR
2	Добавить LF к строке и послать EOI с LF
3	Добавить CR LF к строке и послать EOI с LF
4	Добавить CR к строке, но не посылать EOI
5	Добавить LF к строке, но не посылать EOI
6	Добавить CR LF к строке, но не посылать EOI
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

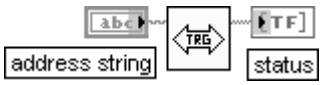


Ожидать GPIB RQS

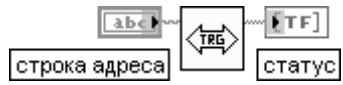


Функция ожидает, пока устройство, указанное в **строке адреса** (address string), установит сигнал RQS.
Выход **байт регистрации ответа** (poll response byte). Если линия SRQI установлена, то функция опрашивает устройство по заданному адресу для обнаружения запроса на обслуживание. Когда заданное устройство 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



Запустить GPIB

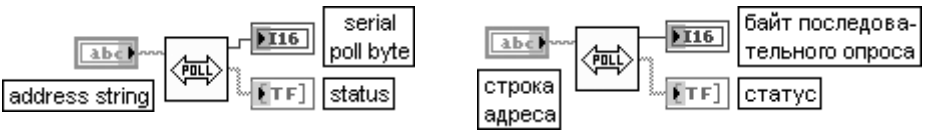


Функция посылает команду **Запуск прибора** (GET – Group Execute Trigger) устройству, указанному в **строке адреса** (address string)

GPIB Clear **Очистить 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**

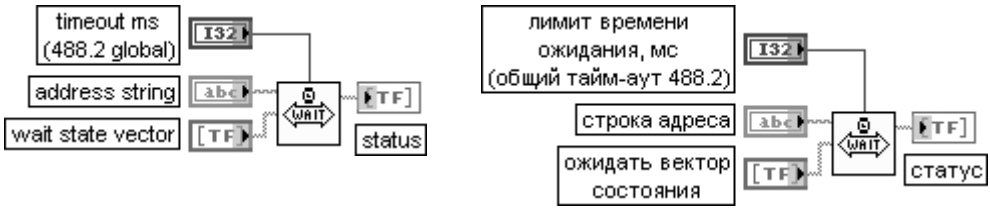
Функция показывает текущий статус GPIB-контроллера, заданного в строке адреса. Следующая таблица показывает числовое значение и символический статус каждого бита выходного логического массива **статус** (status). Таблица также включает описание каждого бита.

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)
12	4096	SRQI	**Обнаружен SRQ пока CIC** (SRQ Detected while CIC)

13	8192	END	**Обнаружен EOI или EOS** (Detected EOI or EOS Detected)
14	16 384	TIMO	**Истечение времени ожидания** (Timeout)
15	– 32 768	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 **Статус GPIB**, рассмотренной выше. Отличие связано с отсутствием значений с символическим обозначением CMPL и ERR

GPIB Misc **Набор функций GPIB**

Функция выполняет операции GPIB, отображаемые в **строке команды** (command string). Эта функция низкого уровня используется, когда другие функции высокого уровня неприменимы.

Следующая таблица показывает функции устройства или контроллера, которые могут быть определены в **строке команды** (command string) (*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

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 **идентифицировать** (Identify (IDY)) во время параллельного опроса. Устройство сравнивает начение **индивидуального бита статуса** (individual status bit (ist)) и переводит отображаемую линию DIO в состояние ИСТИНА или ЛОЖЬ

Функции контроллера	Описание
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
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 рис. 5.37 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. 5.38). 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.

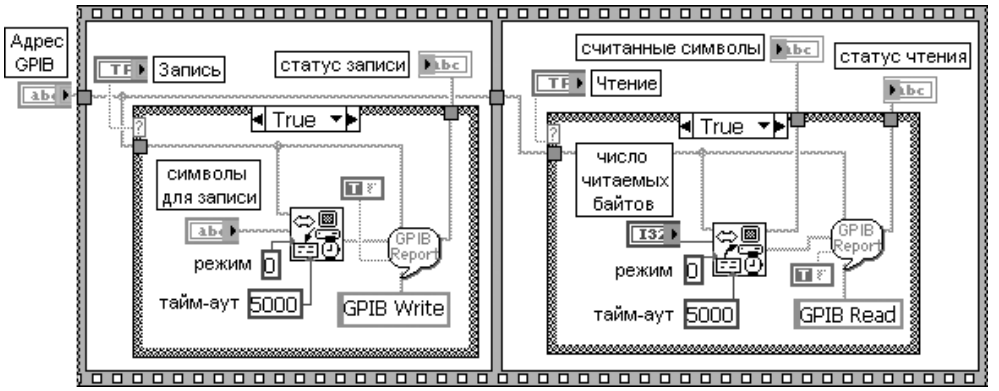


Рис. 5.37. Блок-диаграмма ВП LabVIEW <–> GPIB

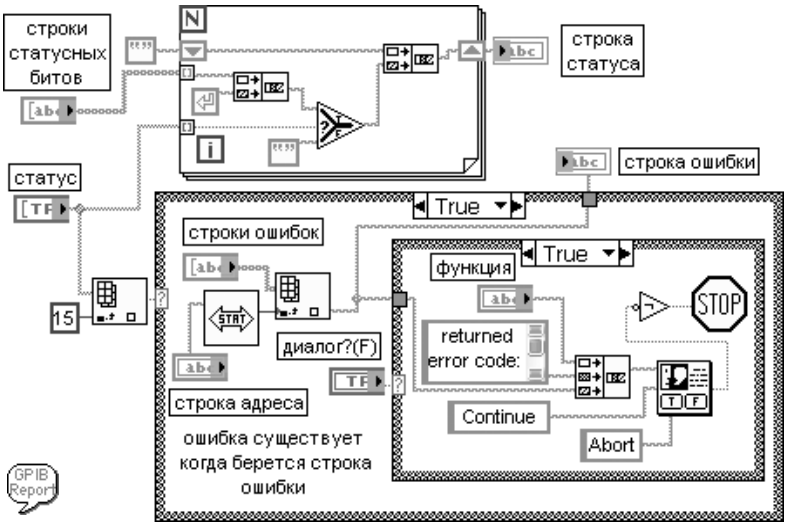


*Рис. 5.38. Иконка и блок-диаграмма ВП **Сообщение от ошибке GPIB** (GPIB Error Report)*

5.4. 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. 5.39а. Данные функции являются специализированными функциями VISA для последовательных портов. В состав палитры входят функции конфигурирования 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. 5.39б).

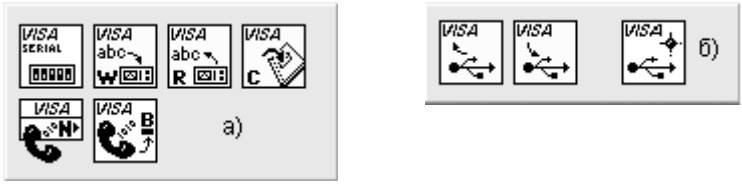
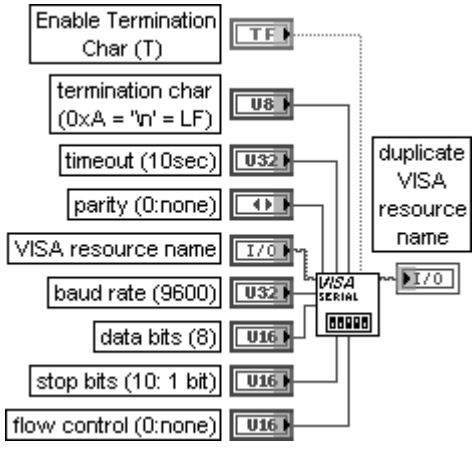
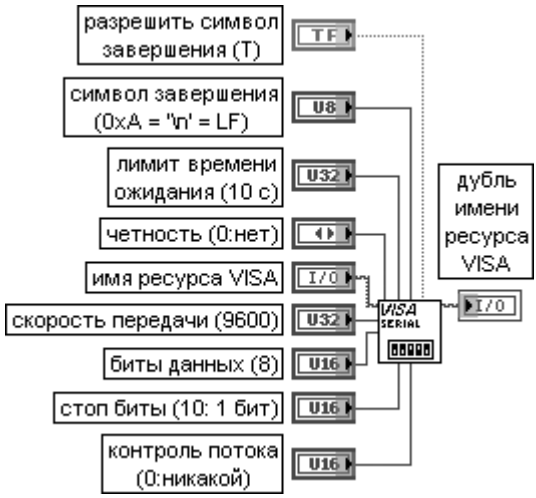


Рис. 5.39. Вид палитры функций последовательной коммуникации (а) и функций управления передачей по каналу USB (б)

VISA Configure Serial Port



Конфигурирование последовательного порта 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 бит. По умолчанию число битов равно 8.

Вход **четность** (parity) определяет способ проверки четности, используемый для каждого передаваемого или принимаемого кадра.

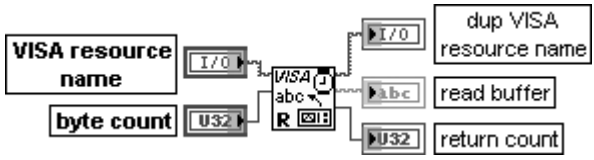
0	1	2	3	4
Нет проверки четности (**по умолчанию**) (no parity)	**Проверка на нечетность** (odd parity)	**Проверка на четность** (even parity)	mark parity	space parity

Вход **стоп-биты** (stop bits) определяет число стоповых битов, используемых для индикации конца кадра

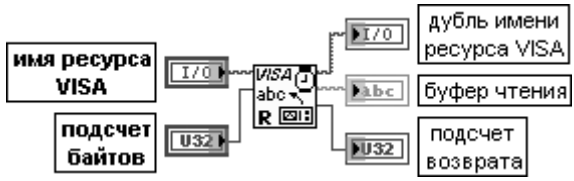
10	15	20
1 стоп-бит	1.5 стоп-бита	2 стоп-бита

Вход **контроль потока** (flow control) устанавливает тип контроля, используемого механизмом передачи

VISA Read

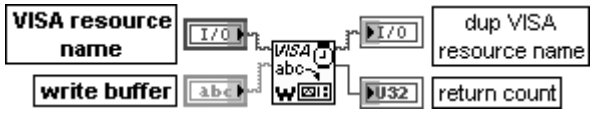


Чтение VISA

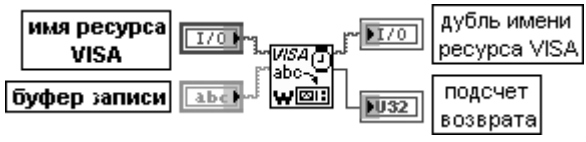


Функция считывает заданное число байтов из устройства или интерфейса, определенного с помощью **имени ресурса VISA** (VISA resource name), и возвращает данные в **буфере чтения** (read buffer). Синхронность или асинхронность считывания данных зависит от платформы. Для установки синхронного чтения необходимо в контекстном меню функции выбрать опцию **выполнять ввод/вывод синхронно** (Do I/O Synchronously). Операция возвращается только после завершения передачи

VISA Write



Запись VISA

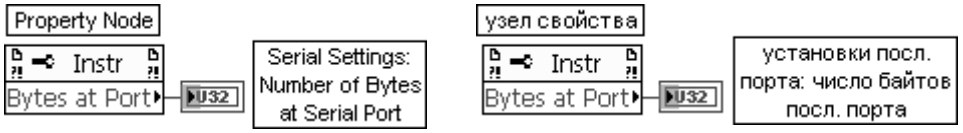


Функция записывает данные из **буфера записи** (write buffer) в устройство или интерфейс, определенные с помощью **имени ресурса VISA** (VISA resource name).
Выход **подсчет возврата** (return count) содержит действительное число записанных байтов

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.

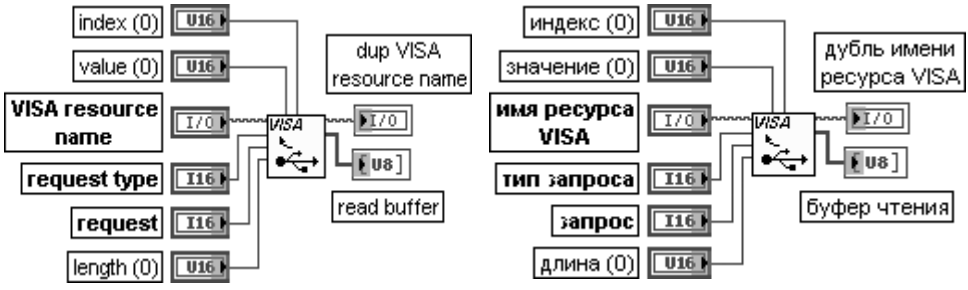
Свойство **число байтов последовательного порта** (Number of Bytes at Serial Port) определяет число байтов, доступных в текущее время в последовательном порту, используемом в данной сессии. Функция **Байты последовательного порта VISA** является **Узлом свойств** (Property Node) класса VISA:I/O Session:Instr.

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**

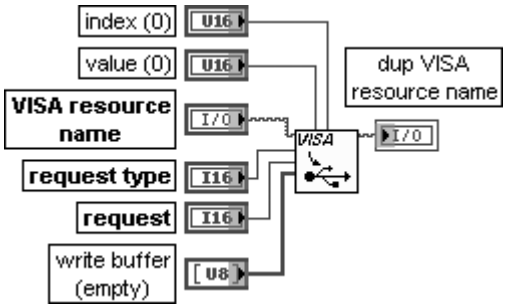
Функция посылает паузу длительностью не менее 250 мс в заданный выходной порт

VISA USB Control In **Управление входной точкой VISA USB**

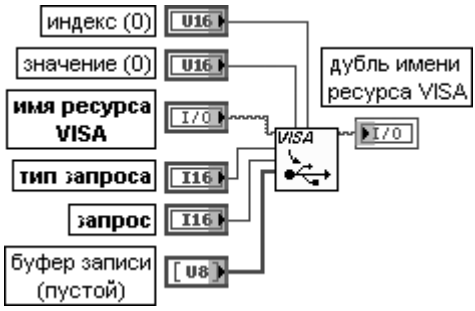


Функция выполняет управление передачей по каналу USB от устройства USB. Значения данных, загружаемые на стадии установки управления передачей, берутся как параметры и включают bmRequestType, bRequest, wValue, wIndex и wLength. Дополнительный буфер данных также считывается, если для этой передачи требуется 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. 5.40 приведена блок-диаграмма модернизированного ВП LabVIEW ↔ Serial из набора примеров NI Example Finder LabVIEW.

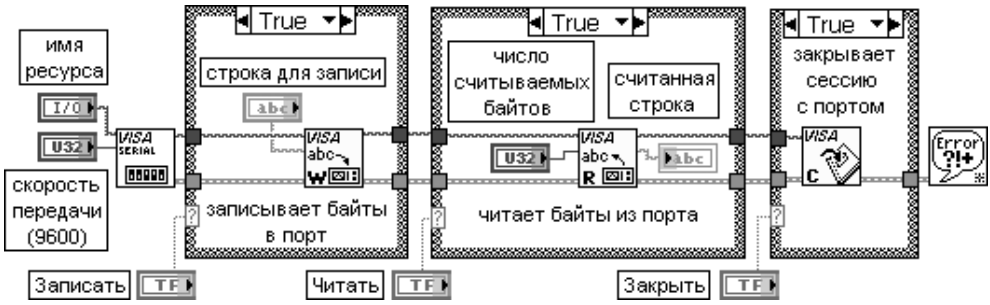


Рис. 5.40. Блок-диаграмма модернизированного ВП LabVIEW ↔ Serial

Приложение 1
Новые возможности LabVIEW 7.0 и 7.1

- **Экспресс-ВП** (Express VIs) – для решения общих измерительных задач используются Экспресс-ВП, которые представляют узлы с минимальным числом соединений, поскольку пользователь может сконфигурировать их с помощью диалоговых окон.
- **Динамический тип данных** (Dynamic Data Type) – большая часть Экспресс-ВП принимает или возвращает данные динамического типа. Данные динамического типа можно подключать к любому индикатору или входу, которые принимают числовые и логические данные, а также осциллограммы.
- **Улучшения диалогового окна LabVIEW** (LabVIEW Dialog Box Enhancements) – общие возможности и утилиты LabVIEW перечислены в кратких меню диалогового окна LabVIEW, выводимого при запуске. Так, в частности, кнопка **Новые** (New) вызывает одноименное диалоговое окно, в котором выводится список встроенных **шаблонов ВП** (VI templates). Кнопка **Конфигурировать** (Configure) открывает выпадающее меню, которое позволяет запускать **Проводник измерения и автоматизации** (Measurement & Automation Explorer (MAX)) или конфигурировать новые каналы или задачи с помощью строк **Создать задачу DAQmx** (Create DAQmx Task) или **Создать канал DAQmx** (Create DAQmx Channel) с помощью **Помощника DAQ** (DAQ Assistant). Строка **Опции LabVIEW** (LabVIEW Options) позволяет вызвать одноименное диалоговое окно для настройки опций LabVIEW. При этом в выводимом окне, которое является первым в списке окон, приведен перечень новых и измененных опций LabVIEW 7.0 (New and Changed in 7.0).
- **Использование шаблонов ВП для создания новых ВП** (Using Template VIs to Create New VIs) – в состав шаблонов ВП входят шаблоны ВП сбора данных (DAQ) и приборного ввода/вывода (Instrument I/O), а также набор шаблонов из раздела **Конструирование образов** (Design Patterns), в которых заложен ряд моделей построения ВП. Лицевую панель и блок-диаграмму шаблонов можно оперативно просмотреть, выбрав опцию Large dialog в диалоговом окне New.

- **Улучшения палитр элементов управления и функций** (Controls and Functions Palette Enhancements) – для изменения формата или вида палитр используется кнопка **Опции** (Options) из панели инструментов этих палитр. Для создания или редактирования индивидуальных палитр необходимо выбрать меню **Инструменты ⇒ Расширенные ⇒ Редактировать вид палитры** (Tools ⇒ Advanced ⇒ Edit Palette Views).
- **Диалоговые окна свойств** (Property Dialog Boxes) – диалоговые окна свойств используются для установки вида и поведения элементов управления или индикаторов. Вызов диалогового окна свойств осуществляется с помощью строки **Свойства** (Properties) контекстного меню элемента или его терминала.
- **Автоматическая обработка ошибки** (Automatic Error Handling) – по умолчанию LabVIEW автоматически обрабатывает любые ошибки, которые происходят при выполнении ВП, путем приостановки выполнения, выделения подприбора или функции, в которых произошли ошибка и отображение диалогового окна ошибки. Пользователь может отключить автоматическую обработку ошибки.
- **Иконки для терминалов элементов лицевой панели** (Icons for Front Panel Terminals) – элементы управления или индикаторы лицевой панели могут отображаться на блок-диаграмме как иконки (по умолчанию) или как терминалы данных определенного типа. Отключение опции отображения иконки производится путем вызова диалогового окна **Опции** (Options) с помощью меню **Инструменты ⇒ Опции** (Tools ⇒ Options), выбора строки **Блок-диаграмма** (Block Diagram) из верхнего выпадающего меню и удаления отметки из раздела **Устанавливать терминалы элементов лицевой панели как иконки** (Place front panel terminals as icons).
- **Автоматическая маршрутизация проводов** (Automatic Wire Routing) – LabVIEW автоматически находит маршрут для провода после его проведения. Вызов данной функции для существующего провода производится с помощью строки **Упорядочить провод** (Clean Up Wire) контекстного меню провода. Отключение автоматической маршрутизации проводов осуществляется путем удаления отметки из раздела **Разрешить автоматическую маршрутизацию проводов** (Enable automatic wire routing), находящегося в диалоговом окне **Блок-диаграмма**, путь к которому приведен в предыдущем разделе.
- **Автоматическое изменение размеров структур** (Resizing Structures Automatically) – при помещении или перемещении объекта в структуре возле ее края структура изменяет размеры с целью увеличения пространства для этого объекта. При ручном изменении размеров структуры они не могут быть сделаны меньше, чем это позволяют элементы, находящиеся внутри их.
- **Структура Открытая последовательность** (Flat Sequence Structure) – сразу отображает все ее кадры и выполняет их слева направо до последнего кадра. Структуру Открытая последовательность рекомендуется использо-

вать для лучшего документирования блок-диаграммы и предотвращения использования терминалов локальной переменной.

- **Искатель примеров NI** (NI Example Finder) – для просмотра или поиска установленных примеров ВП или примеров ВП из Web необходимо запустить Искатель примеров NI с помощью меню **Помощь ⇒ Найти примеры** (Help ⇒ Find Examples).
- **Каталог данных LabVIEW** (LabVIEW Data Directory) – при установке LabVIEW создается подкаталог данных LabVIEW в каталоге файлов операционной системы по умолчанию. Подкаталог данных LabVIEW помогает организовать и расположить файлы данных, генерируемые ВП, в частности файлы с расширением .lvm. Пользователь сам может определить каталог данных по умолчанию.
- **Помощник DAQ** (DAQ Assistant) – помощник DAQ представляет графический интерфейс для конфигурирования NI-DAQmx измерительных задач, каналов и масштабов для использования в LabVIEW 7.0 и следующих версиях. Помощника DAQ можно запустить из стартового диалогового окна LabVIEW, из элементов управления **Задача DAQmx** (DAQmx Task) и **Общий канал ввода/вывода DAQmx** (DAQmx Global Channel I/O), а также из Экспресс-ВП **Помощник DAQ**.
- **Помощник приборного ввода/вывода** (Instrument I/O Assistant) – применяется для связи с приборами, которые используют последовательный порт, Ethernet или канал общего пользования (КОП) (General Purpose Interface Bus (GPIB)) и графический анализ отклика. Помощник приборного ввода/вывода организовывает связь в виде упорядоченных шагов. Он запускается при помещении Экспресс-ВП **Помощник приборного ввода/вывода** на блок-диаграмму или двойным щелчком на иконке этого Экспресс-ВП.
- **Элементы управления вводом/выводом** (I/O Controls) – палитра ввода / вывода включает элементы управления вводом/выводом устройств управления двигателями, устройств FieldPoint и NI-DAQmx. С помощью элементов управления, размещенных в палитре DAQmx Name Controls, можно получить доступ к ресурсам заданий, общих каналов, физических каналов, терминалов, масштабов и переключателей, которые были сконфигурированы с помощью MAX и Помощника DAQ.
- **Функции .NET** (.NET Functions) – функции из палитры .NET позволяют создавать объекты .NET и устанавливать свойства и методы этих объектов.
- **Улучшения среды LabVIEW** (LabVIEW Environment Enhancements) – в среде LabVIEW сделаны такие улучшения, как сокращенные меню, иконки на терминалах элементов лицевой панели и автоматический выбор инструментов. С помощью выбора меню **Инструменты ⇒ Опции** (Tools ⇒ Options) и строки **Новое и измененное в LabVIEW 7.0** (New and Changed in 7.0) из выпадающего меню окна **Опции** можно настроить эти и ряд других улучшений среды LabVIEW.
- **Выравнивание объектов с помощью сетки** (Aligning Objects Using the Alignment Grid) – с помощью выравнивающей сетки можно выровнять

положение объектов при их помещении на лицевую панель или на блок-диаграмму. Опции сетки можно установить для всех новых ВП с помощью раздела **Выравнивающая сетка** (Alignment Grid) диалогового окна **Опции** или для текущего ВП с помощью раздела **Опции редактирования** (Edit Options) диалогового окна **Свойства ВП** (VI Properties).

- **Редактирование и настройка ярлыков единиц измерения** (Editing and Customizing Unit Labels) – для редактирования ярлыка единицы измерения используется диалоговое окно **Создать строку единицы измерения** (Build Unit String), вызываемое с помощью одноименной строки контекстного меню ярлыка единицы.
- **Улучшения печати и генерации отчета** (Printing and Report Generation Enhancements) – ВП **Легкая печать панели или документации** (Easy Print VI Panel or Documentation) позволяет вывести на печать лицевую панель или документацию ВП, или сохранить лицевую панель или документацию ВП в отчете. Для настройки документации ВП, выводимой на печать или сохраняемой в отчете, можно использовать ВП из подпалитры **Документация ВП** (VI Documentation). Для вывода списка доступных принтеров, в том числе принтера по умолчанию, служит ВП **Запрос доступных принтеров** (Query Available Printers).
- **Использование элемента управления Дерево** (Using Tree Controls) – элемент управления Дерево позволяет расположить выбираемые пункты в виде иерархического списка.
- **Использование элемента управления Подпанель** (Using Subpanel Controls) – элемент управления Подпанель позволяет отображать на лицевой панели текущего ВП лицевую панель другого ВП.
- **Улучшения элемента управления Кольцо** (Ring Control Enhancements) – добавление пунктов в список элемента управления Кольцо производится в диалоговом окне **Свойства элемента управления Кольцо ⇒ Редактировать пункты** (Ring Properties ⇒ Edit Items), вызываемого с помощью строки **Редактировать пункты** (Edit Items) контекстного меню элемента управления. При конфигурировании списка пунктов элемента управления можно задать специфическое числовое значение для каждого пункта.
- **Использование элементов управления Комбинированное окно** (Using Combo Box Controls) – Комбинированное окно в качестве элемента управления служит для создания списка строк, из которых должна быть выбрана одна. При конфигурировании списка строк можно задать индивидуальное значение для каждой строки. По умолчанию комбинированное окно позволяет пользователю ввести строковые значения, отличающиеся от значений в списке строк элемента управления.
- **Динамическая регистрация событий и обработка событий пользователя** (Registering Events Dynamically and Handling User Events) – для динамической регистрации событий и обработки событий пользователя необходимо отредактировать варианты структуры **событие**.

- **Элемент управления Отметка времени и тип данных** (Time Stamp Control and Data Type) – позволяет использовать, наблюдать и хранить абсолютное время с высокой точностью. Тип данных Отметки времени позволяет аккуратно хранить 15 цифр точности до и после запятой при определении времени в секундах.
- **Вставленные и специальные пробники** (Supplied and Custom Probes) – устанавливаются с помощью контекстного меню проводника данных и позволяют наблюдать данные, проходящие по этим проводникам в процессе выполнения ВП в наиболее удобном виде.
- **Улучшения элементов управления на основе рисунка** (Picture Control Enhancements) – ВП **Создать маску** (Create Mask) позволяет наложить маску на изображение. С помощью ВП **Рисунок в изображение** (Picture to Pixmap) можно преобразовать данные рисунка в кластер **данных изображения** (image data), с тем чтобы выполнить определенные задачи, например сохранить данные изображения в файле. ВП **Получить фрагмент изображения** (Get Image Subset) позволяет получить фрагмент изображения вместо всего изображения. С помощью ВП **Рисовать восстановленное изображение** (Draw Unflattened Pixmap) можно вывести на рисунок данные растрового изображения.
- **Изменение вида курсора в ВП** (Changing the Cursor Appearance in a VI) – с помощью ВП из подпалитры **Курсор** (Cursor) пользователь может программно изменять вид курсора на лицевой панели ВП.
- **Типы данных цифровая осциллограмма и цифровой тип данных** (Digital Waveform Data Type and Digital Data Type) – для представления цифровых осциллограмм и цифровых данных используются соответствующие типы данных.
- **Узлы обратной связи** (Feedback Nodes) – LabVIEW автоматически вставляет узел обратной связи в цикл с фиксированным числом итераций или в цикл по условию при соединении выхода подприбора, функции или группы подприборов и функций с входом тех же самых подприборов, функций или их групп. Как и сдвигающий регистр, узел обратной связи хранит данные, когда цикл выполняет итерацию, посылает это значение в следующую итерацию цикла и передает любые типы данных. Стрелка узла обратной связи показывает направление передачи данных по проводу.
- **Отправка данных по электронной почте из ВП** (Emailing Data from VIs) – ВП из подпалитры **Электронная почта** (SMTP E-mail) позволяют отправлять электронные письма с вложенными файлами или данными, используя протокол SMTP.
- **Буферизация данных DataSocket** (Buffering DataSocket Data) – буферизация данных DataSocket на стороне клиента должна применяться для обеспечения считывания всех данных, опубликованных сервером DataSocket.
- **Программное открытие и закрытие соединений DataSocket** (Programmatically Opening and Closing DataSocket Connections) – для программного откры-

тия и закрытия соединения DataSocket необходимо использовать функции **Открыть DataSocket** (DataSocket Open) и **Закрыть DataSocket** (DataSocket Close). Для конфигурирования констант буферизации DataSocket, проверки статуса соединения DataSocket и чтения DataSocket URL необходимо использовать такие свойства DataSocket, как **Максимальное число байтов буфера** (Buffer Maximum Bytes), **Максимальное число пакетов буфера** (Buffer Maximum Packets), **Использование буфера (байты)** (Buffer Utilization (Bytes)), **Использование буфера (пакеты)** (Buffer Utilization (Packets)), **Статус соединения** (Connection Status) и URL.

- **Обработка событий ActiveX** (Handling ActiveX Events) – ВП **Событие ActiveX** отсутствует в палитре функций. Для использования в приложении событий ActiveX необходимо использовать узел **Регистрация события Возврат вызова** (Register Event Callback).
- **Новые методы и свойства сервера ВП** (New VI Server Properties and Methods) – LabVIEW 7.0 содержит ряд новых методов и свойств сервера ВП.
- **ВП управления устройствами ввода** (Input Device Control VIs) – ВП управления устройствами ввода позволяют получить информацию о джойстике, клавиатуре и мыши, подключенных к компьютеру.

Новые возможности LabVIEW 7.1

1. Использование элементов управления в виде радиокнопок.
2. Использование палитр ВП **Полиномы** и **Рациональные полиномы**.
3. Использование полиморфных ВП в палитре **Линейная алгебра** вместо ВП в палитрах **Дополнительные функции линейной алгебры** (Advanced Linear Algebra), **Комплексные функции линейной алгебры** (Complex Linear Algebra) и **Дополнительные комплексные функции линейной алгебры** (Advanced Complex Linear Algebra).
4. Использование новых ВП в палитре **Линейная алгебра**.
5. Использование ВП **Изменение интервала выборки** (Resample) в палитре функций обработки сигналов во временной области.
6. Использование полиморфных ВП **Быстрое преобразование Фурье** (FFT) и **Обратное быстрое преобразование Фурье** (Inverse FFT) в палитре функций обработки сигналов в частотной области вместо ВП **Комплексное преобразование Фурье** (Complex FFT), **Обратное комплексное БПФ** (Inverse Complex FFT), **Действительное преобразование Фурье** (Real FFT), **Обратное действительное преобразование Фурье** (Inverse Real FFT).
7. Использование ВП **Сохранение** (Storage VIs) для записи и чтения осциллограмм и свойств осциллограмм из файлов типа NI Test Data Exchange Format с расширением (.tdm). Файлы с расширением (.tdm) используются для обмена данными между такими приложениями NI, как LabVIEW и DIAdem. ВП **Сохранение** объединяют осциллограммы и свойства осциллограмм и образуют каналы. Набор каналов организуется в группу кана-

лов. Файл включает набор групп каналов. Наряду с числовыми значениями ВП **Сохранение** поддерживают массивы строк и массивы отметок времени. Номер ссылки представляет файлы, группы каналов и каналы на блок-диаграмме.

8. Использование Экспресс-ВП **Добавить сигналы** (Append Signals) для объединения сигналов.
9. Использование ВП и функций Bluetooth для связи с устройствами, которые используют данный протокол обмена.
10. Сохранение графиков или разверток осциллограмм, таблиц и цифровых элементов управления как изображений с помощью строки **Операции с данными ⇒ Экспортировать упрощенное изображение** (Data Operations ⇒ Export Simplified Image) контекстного меню элемента.
11. Использование окна **Навигация** для обзора активной лицевой панели в режиме редактирования или активной блок-диаграммы. Окно **Навигация** используется для обзора больших лицевых панелей или блок-диаграмм. Невидимая часть панели или блок-диаграммы в окне **Навигация** окрашены серым цветом.
12. Отображение мест расположения буферов с помощью строки главного меню **Инструменты ⇒ Дополнительные ⇒ Показать места расположения буферов** (Tools ⇒ Advanced ⇒ Show Buffer Allocations). Установка отметки перед типом данных в диалоговом окне **Показать места расположения буферов** и нажатие клавиши **Обновить** (Refresh) позволяет увидеть на блок-диаграмме места создания буферов данных, которые выделяются черными квадратами.
13. Использование папки **Предпочтения** (Favorites) и папки **Самые последние** (Most Recent) в таблице **Просмотр** (Browse) системы **Поиск примеров NI** (NI Example Finder) для организации и легкого доступа к примерам ВП, используемым наиболее часто.
14. **Цикл заданной длительности** (Timed Loop), выполняющий итерацию цикла в течение заданного интервала времени.
15. Справочные темы определенных ВП и функций в разделе **Помощь LabVIEW** (Help LabVIEW) содержат кнопки **Открыть пример** (Open example) и **Просмотреть связанные примеры** (Browse Related examples).

Приложение 2
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): **объявление переменной** (variable-declaration),
присваивание (assignment),
составной оператор (compound-statement),
условный оператор (conditional-statement),
итеративный оператор (iterative-statement),
оператор переключения (switch-statement),
оператор управления (control-statement).

объявление переменной (variable-declaration):
идентификатор определителя типа (type-specifier identifier),
идентификатор определителя типа (type-specifier identifier) **список индексов массива** (array-index-list),
идентификатор определителя типа (type-spec identifier) **= присваивание** (assignment).
список индексов массива (array-index-list):
[целая константа (integer-constant)**]**,
[целая константа (integer-constant)**] список индексов массива** (array-index-list).

определитель типа (type-specifier):
тип с плавающей точкой (floating-point- type),
целый тип (integer-type).
тип с плавающей точкой (floating-point-type): **float, float32, float64**.
целый тип (integer-type): **int, int8, int16, int32, uInt8, uInt16, uInt32**.

присваивание (assignment):
выражение (expression),
оператор присваивания (left-hand-side assignment-operator) **присваивание** (assignment).
выражение (expression):
выражение (expression) **двоичный оператор** (binary-operator) **выражение** (expression),
унарный оператор (unary-operator) **выражение** (expression),
выражение (expression) **унарный оператор** (unary-operator),
выражение (expression) **? выражение** (expression) **: выражение** (expression),
(выражение (expression)**),**
идентификатор (identifier),
константа (constant),
имя функции (function-name) **(список аргументов** (argument-list)**).**
left-hand-side:
идентификатор (identifier),
идентификатор (identifier) **описание массива** (array-subscription).
описание массива (array-subscription):
[присваивание (assignment)**]**
[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).
составной оператор (compound-statement):
{ список операторов (statement-list)**}**.
условный оператор (conditional-statement):
оператор if (if-statement),
оператор if-else (if-else-statement).
оператор if (if-statement):
if (присваивание) оператор (**if (**assignment**)** statement),
оператор if-else (if-else-statement):
if оператор else оператор (if-statement **else** statement).
итеративный оператор (iterative-statement):
оператор do (do-statement),
оператор for (for-statement),
оператор while (while-statement).
оператор do (do-statement):
do оператор while (присваивание) (**do** statement **while** (assignment)).

оператор while (while-statement):
while (присваивание) оператор (**while (** assignment **)** statement).
оператор for (for-statement):
for ([**присваивание**]; [**присваивание**]; [**присваивание**]) оператор** (**for (**[assignment]; [assignment]; [assignment]**)** statement).
оператор управления (control-statement):
прервать (break),
продолжить (continue).
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 3
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 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

«Горячая» клавиша	Действие
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
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. Otherwise, enables automatic tool selection

«Горячая» клавиша	Действие
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
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

«Горячая» клавиша	Действие
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
Свойства класса Приложение (Application Properties)

4

Раздел **Приложение** (Application)

- **Все ВП в памяти** (All VIs In Memory) – возвращает список всех ВП, находящихся в памяти. При использовании свойства для возврата списка удаленных ВП возвращается ошибка.
- **Аргументы командной строки** (Command Line Arguments) – возвращает массив определенных пользователем аргументов командной строки, передаваемых при запуске LabVIEW.
- **Каталог данных по умолчанию** (Default Data Directory) – путь к каталогу данных LabVIEW.
- **Путь к каталогу** (Directory Path) – абсолютный путь к каталогу, в котором находится приложение.
- **Экспортированные ВП в памяти** (Exported VIs In Memory) – возвращает список экспортированных ВП в памяти.
- **Тип** (Kind) – определяет тип пакета LabVIEW, установленного на компьютере.
- **Язык** (Language) – показывает язык среды LabVIEW в виде двухбуквенной строки в соответствии с ISO 639.
- **Имя** (Name) – имя исполняемого файла приложения.
- **Хост реального времени подключен** (Real-Time Host Connected) – если платформа не является платформой реального времени, то это свойство всегда имеет значение ИСТИНА. Если платформа является платформой реального времени и хост подключен, то это свойство имеет значение ИСТИНА. Когда это свойство имеет значение ЛОЖЬ, хост не подключен, что требует, чтобы **приложение модуля реального времени** (RT Module application) было способно выполнять все операции независимо от хоста. Это свойство необходимо использовать для определения, когда он находится в режиме сохранения для отображения диалогового окна, которое требует взаимодействия с пользователем.
- **Показать полосы подсказки лицевой панели** (Show FP Tip Strips)
  Запись свойства: установка этого свойства в состояние ИСТИНА разрешает отображение полос подсказки элемента управления лицевой панели. Установка свойства в состояние ЛОЖЬ предотвращает отображение.
  Чтение свойства: определяет возможность отображения полос подсказки при перемещении курсора над объектом лицевой панели.

- **Центральный процессор** (Target CPU) – показывает тип центрального процессора приложения.
- **Операционная система** (Target Operating System) – показывает тип операционной системы, в которой установлено приложение.
- **Имя пользователя** (User Name) – показывает имя пользователя, введенное при открытии приложения.
- **Номер версии** (Version Number) – показывает номер версии приложения

Раздел **Отображение** (Display)

- **Все мониторы** (All Monitors) – возвращает информацию о всех мониторах компьютера, включая разрядность и размер экрана.
- **Первичная рабочая область** (Primary Workspace) – дает ограничивающий прямоугольник первичного монитора компьютера.

Раздел **Операционная система** (Operating System)

- **Имя** (Name) – имя операционной системы, в которой приложение действительно выполняется.
- **Номер версии** (Version Number) – номер версии действительной операционной системы

Раздел **Печать** (Printing)

- **Доступные принтеры** (Available Printers) – возвращает массив имен доступных принтеров в алфавитном порядке.
- **Цветной/Полутоновый?** (Color/Grayscale?) – при установке в со стояние ИСТИНА LabVIEW посылает принтеру выход **цветной/полутоновый**. При установке состояния ЛОЖЬ LabVIEW посылает принтеру выход **монохромный**.
- **Выбрать все элементы управления?** (Custom All Controls?) – определяет печать всех элементов управления или только элементов, подключенных к соединительной панели, при использовании обычного формата для одного из методов печати документации ВП. Далее в свойствах печати используется обычный формат.
- **Выбрать соединитель?** (Custom Connector?) – определяет печать соединительной панели и иконки.
- **Выбрать описания элементов управления?** (Custom Control Descriptions?) – определяет печать описаний элементов управления лицевой панели.
- **Выбрать типы элементов управления?** (Custom Control Types?) – определяет печать информации о типах данных элементов управления.
- **Выбрать элементы управления?** (Custom Controls?) – определяет печать информацию об элементах управления лицевой панели.
- **Выбрать описание ВП?** (Custom Description?) – определяет печать описаний ВП.
- **Выбрать невидимую диаграмму?** (Custom Diagram Hidden?) – определяет печать скрытых кадров в структурах Вариант или Стековая последовательность, которые могут находиться на блок-диаграмме.
- **Выбрать повторяющуюся диаграмму?** (Custom Diagram Repeat?) – определяет печать видимых кадров в последовательности с невидимыми кадрами. Если печатаются видимые кадры последовательности, то эти кадры будут печататься дважды.
- **Выбрать диаграмму?** (Custom Diagram?) – определяет печать блок-диаграммы.
- **Выбрать конфигурацию Экспресс-ВП?** (Custom Express VI Configuration?) – определяет печать информации о конфигурации любых Экспресс-ВП блок-диаграммы.
- **Выбрать иерархию?** (Custom Hierarchy?) – определяет печать иерархии ВП в памяти.

- **Выбрать историю?** (Custom History?) – определяет печать информации об истории исправлений ВП.
- **Выбрать границы панели?** (Custom Panel Border?) —определяет печать лицевой панели с границами.
- **Выбрать панель?** (Custom Panel?) – определяет печать лицевой панели.
- **Выбрать подприбор?** (Custom SubVIs?) – определяет печать списка подприборов, включающего иконку, имя и путь.
- **Принтер по умолчанию** (Default Printer) – устанавливает или принимает имя принтера по умолчанию в LabVIEW.
- **Максимальное число символов в строке файла** (File Wrap Text Length) – действует на методы **Печатать ВП в формате HTML** (Print VI to HTML), **Печатать ВП в формате RTF** (Print VI to RTF) и **Печатать ВП в текстовом формате** (Print VI to Text). Установка этого свойства в 0 приводит к печати всех символов в одну строку.
- **Качество JPEG** (JPEG Quality) – значение от 0 до 100 процентов определяет уровень качества изображения в формате JPEG при печати ВП интерактивно или с помощью метода **Печатать ВП в формате HTML** (Print VI to HTML). Шкала балансирует качество изображения и размер файла. Значение в диапазоне 75–95 создает сжатый файл с высоким качеством изображения, а значение ниже 50 создает меньший по размеру файл с низким качеством изображения. По умолчанию значение равно 80.
- **Метод** (Method) – устанавливает или принимает метод печати LabVIEW. Допустимые значения включают 0 (Стандартный), 1 (PostScript) и 2 (Bitmap).
- **Уровень сжатия PNG** (Printing: PNG Compression Level) – число между 0 и 9, которое определяет уровень сжатия файла PNG для графиков ВП, выводимых на печать интерактивно или с помощью метода **Печатать ВП в формате HTML** (Print VI to HTML). Качество графика не зависит от уровня сжатия, но размер файла графика и скорость сжатия зависят от этого параметра. Допустимые значения изменяются в диапазоне от –1 до 9 и балансируют сжатие файла и скорость. 9 определяет отсутствие сжатия, а 0 – высокую компрессию. По умолчанию значение равно 5

Раздел **Удаленная панель** (Remote Panel)

- **Соединения с клиентами** (Connections To Clients) – возвращает массив кластеров, содержащих информацию о текущих соединениях клиентов, подключенных к компьютеру.
- **Соединения с серверами** (Connections To Servers) – возвращает массив кластеров, содержащих информацию о текущих соединениях компьютера с серверами

Раздел **Сервер** (Server)

- **Регистрация протокола разрешена** (Logging Enabled) – определяет возможность интерфейса TCP к серверу ВП для записи выполняемых операций в файл протокола.
- **Путь к файлу протокола** (Logging File Path) – путь к файлу протокола, созданному во время TCP/IP-соединения.
- **Порт** (Port) – устанавливает или принимает порт сервера ВП LabVIEW. LabVIEW возвращает ошибку при попытке установления доступа к этому свойству для удаленного приложения.
- **Слушатель TCP активен** (TCP Listener Active) – определяет возможность сервера ВП устанавливать соединения по интерфейсу TCP.
- **Список доступа TCP/IP** (TCP/IP Access List) – показывает список адресов TCP/IP удаленных клиентов, которые могут иметь доступ к серверу ВП.
- **Список доступных ВП** (VI Access List) – показывает список ВП, доступных удаленным клиентам на сервере ВП

Раздел **Web сервер** (Web Server)

- **Порт HTTP** (HTTP Port) – определяет порт, с помощью которого встроенный Web-сервер прослушивает запрос HTTP.
- **Регистрация протокола разрешена** (Logging Enabled) – определяет возможность встроенного Web-сервера записывать в файл протокола информацию о запросах HTTP.
- **Путь к файлу протокола** (Logging File Path) – определяет путь, по которому встроенный Web-сервер размещает файл протокола.
- **Время ожидания чтения** (Read Timeout) – определяет длительность ожидания встроенным Web-сервером выполнения операций чтения при чтении запросов HTTP от клиентов HTTP.
- **Путь к корневому каталогу** (Root Directory Path) – определяет корневой каталог, из которого встроенный Web-сервер загружает файлы при запросе документов.
- **Сервер активен** (Server Active) – определяет состояние активности встроенного Web-сервера.
- **Список доступа TCP/IP** (TCP/IP Access List) – устанавливает или принимает адреса TCP/IP, которые имеют доступ к Web-серверу. Это свойство также устанавливается или принимается, если каждый адрес имеет доступ для удаленного просмотра ВП, удаленного просмотра и управления ВП или не имеет доступа к Web-серверу.
- **Список доступных ВП** (VI Access List) – устанавливает или принимает ВП, которые являются видимыми через Web-сервер. Это свойство также устанавливается или принимается, когда LabVIEW получает доступ к каждому ВП.

Приложение 5
Свойства класса ВП

Перечисленные ниже свойства позволяют программно управлять свойствами ВП, которые могут быть установлены пользователем в режиме редактирования с помощью выбора пункта меню **Файл ⇒ Свойства ВП** (File ⇒ VI Properties). Если вход **ссылка** (reference) **Узла свойства** (Property Node) не подключен , то LabVIEW получает или устанавливает свойство текущего ВП.

Раздел **Автоматическая регистрация** (A uto Logging)

- **Путь к файлу протокола** (Log File Path) – путь к файлу протокола (datalog file), в который записаны данные лицевой панели и отметка времени.
- **Протокол после окончания** (Log at Finish) – указывает на запись данных лицевой панели в файл протокола после выполнения ВП.
- **Печать после окончания** (Print at Finish) – указывает на печать лицевой панели после выполнения ВП

- **Автоматическая обработка ошибок** (Automatic Error Handling) – при установке состояния ИСТИНА разрешает автоматическую обработку ошибок.
- **Имена вызываемых ВП** (Callees' Names) – возвращает список всех или вызываемых ВП.
- **Имена вызывающих ВП** (Callers' Names) – возвращает список всех загруженных ВП, которые вызывают ВП с данной ссылкой.
- **Соединительная панель** (Connector Pane) – это свойство конфигурирует ВП, который определяется ссылкой, поступающей на узел свойства, устанавливая его соединительную панель такой же, как и у ВП, ссылка от которого передается на вход узла свойства. Два ВП должны иметь одинаковое число параметров с идентичными типами данных и именами. Если такое условие не соблюдается, то узел свойства возвращает ошибку.
- **Режим редактирования при открытии** (Edit Mode On Open) – при установке состояния ИСТИНА ВП открывается в режиме редактирования. При установке состояния ЛОЖЬ ВП открывается в режиме выполнения, в котором полоса названия, полоса меню и полоса инструментов не появляются.

Раздел **Выполнение** (Execution)

- **Разрешить отладку** (Allow Debugging) – показывает разрешение отладки ВП.
- **Закрыть после вызова** (Close After Call) – показывает режим закрытия лицевой панели после выполнения ВП.
- **Является реентерабельным** (Is Reentrant) – показывает реентерабельность ВП.
- **Предпочтительная система выполнения** (Preferred Exec System) – показывает систему выполнения, в которой ВП запускается.
- **Приоритет** (Priority) – показывает приоритет ВП при его выполнении параллельно с другими задачами.
- **Запустить когда открыт** (Run When Opened) – показывает, что ВП запускается при открытии.
- **Показать лицевую панель при вызове** (Show Front Panel On Call) – показывает возможность отображения лицевой панели при вызове ВП.
- **Показать лицевую панель после загрузки** (Show Front Panel On Load) – индицирует возможность показа лицевой панели после загрузки ВП.
- **Состояние** (State) – показывает состояние выполнения ВП.
- **Приостановить при вызове** (Suspend On Call) – показывает, что выполнение ВП приостанавливается при его вызове как подприбора

Расширить (Expand When Dropped As SubVI**)** – расширяет для показа терминалов при Expands to show terminals when dropped as a subVI.

Лицевая панель (Front Panel**)** – ссылка к лицевой панели ВП.

Раздел **Окно лицевой панели** (Front Panel Window)

- **Разрешить показ контекстного меню во время выполнения** (Allow Runtime PopUp) – индицирует возможность показа контекстного меню для объекта лицевой панели во время выполнения ВП.
- **Автоматическое центрирование** (Auto Center) – определяет возможность автоматического центрирования окна лицевой панели на экране компьютера при каждом запуске ВП. При установке состояния ЛОЖЬ лицевая панель появляется в месте своего последнего позиционирования пользователем
- **Поведение** (Behavior) – устанавливает поведение окна лицевой панели. Допустимые значения включают 0 (ошибочное), 1 (по умолчанию), 2 (плавающее), 3 (плавающее/автоматически скрытое) и 4 (модальное).
- **Закрываемое** (Closeable) – показывает, что заблокированы кнопка закрытия в полосе названия и пункт **Закрыть** (Close) в меню **Файл** (File).
- **Обычный заголовок** (Custom Title) – показывает, имеет ли лицевая панель присущий ей заголовок. Запись значения ЛОЖЬ удаляет заголовок.
- **Выделить кнопку возврата** (Highlight Return Button) – показывает возможность выделения логического элемента управления, который имеет Indicates whether to highlight Boolean controls that have a shortcut key of <Enter>.
- **На передний план** (Is Frontmost) – показывает, что лицевая панель выводится на передний план.
- **Сохранить пропорции окна** (Keep Window Proportions) – показывает, что окно лицевой панели изменяет свой размер пропорционально изменению разрешения монитора. Окно изменяет размер таким образом, что оно занимает такую же относительную площадь экрана, что и при исходном разрешении.

- **Минимизируемо** (Minimizeable) – показывает, что пользователь может минимизировать окно лицевой панели во время выполнения ВП.
- **Открыто** (Open) National Instruments рекомендует использовать метод **Открыть лицевую панель** (Open FP) вместо этого свойства.
- **Начало** (Origin) – два элемента кластера являются вертикальной и горизонтальной координатами левого верхнего угла лицевой панели. Эти координаты можно использовать для прокрутки лицевой панели. Они отсчитываются относительно локальных координат окна, поэтому числа относятся к координатам внутри открытого окна. При перемещении полос прокрутки лицевой панели координаты, возвращаемые этим свойством, изменяются.
- **Границы панели** (Panel Bounds) – четыре элемента кластера являются верхней, левой, нижней и правой границами лицевой панели, не включающей полосы прокрутки, полосу названия, полосу меню и полосу инструментов. Они отсчитываются относительно глобальных координат экрана, поэтому числа относятся к координатам внутри экрана монитора компьютера, а не открытого окна.
- **Изменяемое по размерам** (Resizeable) – показывает, что пользователь может изменять размеры окна лицевой панели во время выполнения ВП.
- **Показать полосу меню** (Show Menu Bar) – показывает возможность отображения полосы меню во время выполнения ВП.
- **Показать полосы прокрутки** (Show Scroll Bars) – показывает возможность отображения полос прокрутки во время выполнения ВП.
- **Размер на весь экран** (Size To Screen) – показывает возможность автоматического увеличения размеров лицевой панели до размеров всего экрана.
- **Состояние** (State) – текущее состояние окна лицевой панели. Допустимые значения включают 0 (ошибочное), 1 (стандартное), 2 (закрытое), 3 (скрытое), 4 (минимизированное), and 5 (максимизированное).
- **Название** (Title) – строка, которая появляется в полосе названия. Эта строка не совпадает с именем файла ВП.
- **Полоса названия видима** (Title Bar Visible) – показывает возможность отображения полосы названия лицевой панели во время выполнения ВП.
- **Границы окна** (Window Bounds) – четыре элемента кластера представляют верхнюю, левую, нижнюю и правую границы окна лицевой панели, которое включает внутреннюю область, полосы прокрутки, полосу названия, полосу меню и полосу инструментов. Их положение определяется относительно глобальных координат экрана монитора, а не открытого окна

Раздел **Помощь** (Help)

- **Путь к документу** (Document Path) – путь к файлу HTML (.htm or .html) или откомпилированному файлу помощи (.chm or .hlp), с которым связан ВП.
- **Этикетка документа** (Document Tag) – индексное ключевое слово или имя файла HTML для темы в откомпилированном файле помощи, с которым связан ВП

Раздел **История** (History)

- **Всегда добавлять комментарии при сохранении** (Always Add Comments At Save) – показывает возможность добавлять комментарии к истории исправлений ВП каждый раз, когда ВП сохраняется.
- **Весь текст** (Entire Text) – возвращает весь текст, который был добавлен к истории исправлений ВП.
- **Подсказка для комментариев при закрытии** (Prompt for Comments At Close) – показывает возможность подсказки для комментариев к истории исправлений ВП при его закрытии.

- **Подсказка для комментариев при сохранении** (Prompt for Comments At Save) – показывает возможность подсказки для комментариев к истории исправлений ВП при его сохранении.
- **Записать комментарии приложения** (Record Application Comments) – показывает возможность добавления комментариев к истории исправлений ВП при возникновении таких событий, как преобразование в новую версию LabVIEW, изменение подприбора и изменение имени или пути ВП.
- **Номер исправления** (Revision Number) – текущий номер исправления ВП.
- **Использовать по умолчанию** (Use Defaults) – показывает возможность использования глобальной истории по умолчанию или использовать значения, введенные в других свойствах истории

Раздел **Метрика** (Metrics)

- **Размер кода** (Code Size) – показывает объем памяти, используемый для кода ВП, в байтах.
- **Размер блок-диаграммы** (Size of Block Diagram) – показывает объем блок-диаграммы в байтах.
- **Размер лицевой панели** (Size of Front Panel) – показывает объем лицевой панели в байтах.
- **Общий объем данных** (Total Data Size) – показывает объем памяти, выделенный для данных, в байтах

Раздел **Модификации** (Modifications)

- **Режимы установки битов блок-диаграммы** (Block Diagram Mods Bitset) – показывает возможность делать изменения в блок-диаграмме после сохранения или открытия ВП в зависимости от того, какая операция была последней. Изменения делаются только при ненулевом значении.
- **Режимы установки битов лицевой панели** (Front Panel Mods Bitset) – показывает возможность делать изменения в лицевой панели после сохранения или открытия ВП в зависимости от того, какая операция была последней. Изменения делаются только при ненулевом значении.
- **Установки битов модификации** (VI Modifications Bitset) ВП – показывает возможность делать изменения в установках ВП после его сохранения. Изменения делаются только при ненулевом значении

Раздел **Печать** (Printing)

- **Масштабировать блок-диаграмму?** (Block Diagram Scaling?) – при установке значения ИСТИНА LabVIEW масштабирует блок-диаграмму по размеру печатаемой страницы.
- **Масштабировать лицевую панель?** (Front Panel Scaling?) – при установке значения ИСТИНА LabVIEW масштабирует лицевую панель по размеру печатаемой страницы.
- **Содержание заголовка: Печатать дату?** (Header Content: Date Printed?) – при установке значения ИСТИНА LabVIEW включает печать даты в заголовки ВП. Для установки печати заголовков необходимо использовать свойство **Печатать заголовки страницы?** (Printing: Page Headers?).
- **Содержание заголовка: Изменять дату?** (Header Content: Modify Date?) – при установке значения ИСТИНА LabVIEW включает последнюю измененную дату в заголовки ВП.
- **Содержание заголовка: Номер страницы?** (Header Content: Page Number?) – при установке значения ИСТИНА LabVIEW включает номер страницы в заголовки ВП.

- **Содержание заголовок а: Иконка ВП?** (Header Content: VI Icon?) – при установке значения ИСТИНА LabVIEW включает иконку ВП в его заголовки.
- **Содержание заголовка: Имя ВП? (**Header Content: VI Name?) – при установке значения ИСТИНА LabVIEW включает имя ВП в его заголовки.
- **Содержание заголовка: Путь ВП?** (Header Content: VI Path?) – при установке значения ИСТИНА LabVIEW включает путь к в его заголовки.
- **Отступы** (Margins) – получает или устанавливает отступы страницы в дюймах или сантиметрах для использования при печати ВП.
- **Заголовки страницы** (Page Headers?) – при установке значения ИСТИНА LabVIEW печатает заголовки ВП.
- **Ориентация страницы** (Page Orientation) – получает или устанавливает ориентацию страницы при печати ВП. Возможные значения включают 0 (Вертикальная), 1 (Горизонтальная), 2 (Повернутая вертикальная) и 3 (Повернутая горизонтальная).

- **Путь к меню ВП в режиме выполнения (**Run-Time Menu Path**)** – при чтении это свойство возвращает путь к меню ВП в режиме выполнения. При записи это свойство обновляет путь к меню ВП в режиме выполнения. Если ВП выполняется при записи этого свойства, то оно обновляет меню данными нового пути.

Раздел **Полоса инструментов** (Tool Bar)

- **Показать кнопку прерывания** (Show Abort Button) – показывает возможность отображения кнопки **прерывание** (Abort Execution) на полосе инструментов при выполнении ВП.
- **Показать кнопку непрерывного запуска** (Show Free Run Button) – показывает возможность отображения кнопки **непрерывный запуск** (Run Continuously) на полосе инструментов при выполнении ВП.
- **Показать кнопку запуска** (Show Run Button) – показывает возможность отображения кнопки **запуск** (Run) на полосе инструментов при выполнении ВП.
- **Видимая** (Visible) – показывает возможность отображения полосы инструментов при выполнении ВП

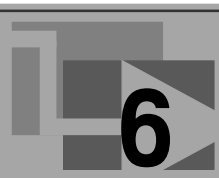
Раздел **Полоса инструментов** (Tool Bar)

- **Показать кнопку прерывания** (Show Abort Button) – показывает возможность отображения кнопки **прерывание** (Abort Execution) на полосе инструментов при выполнении ВП.
- **Показать кнопку непрерывного запуска** (Show Free Run Button) – показывает возможность отображения кнопки **непрерывный запуск** (Run Continuously) на полосе инструментов при выполнении ВП.
- **Показать кнопку запуска** (Show Run Button) – показывает возможность отображения кнопки **запуск** (Run) на полосе инструментов при выполнении ВП.
- **Видимая** (Visible) – показывает возможность отображения полосы инструментов при выполнении ВП.

- **Описание ВП (**VI Description**)** – описание ВП, которое появляется в окне **Контекстная помощь (**Context Help**)** при перемещении курсора через иконку ВП и которое создается в документации ВП.

- **Имя ВП** (VI Name) – имя файла ВП. Это свойство может быть записано только если ВП не был сохранен на диске.
- **Путь ВП** (VI Path) – путь к файлу ВП.
- **Тип ВП** (VI Type) – показывает тип ВП. Возможные варианты включают Стандартный ВП, Управляющий ВП, Глобальный ВП, Полиморфный ВП или Конфигурационный ВП.

Приложение Методы класса Приложение



- **Установить на передний план** (Bring To Front) – в операционной системе Windows и Mac OS устанавливает окно приложения на передний план.
- **Просмотреть DataSocket** (Browse DataSocket) – запускает диалоговое окно браузера DataSocket.
- **Проверить соединение** (Check Connection) – проверяет наличие соединения сервера ВП.
- **Информация о соединении** (Connection Info) – получает или устанавливает частоту, с которой LabVIEW проверяет наличия соединения сервера ВП.
- **Отсоединить от ведомого устройства** (Disconnect From Slave) – разъединяет модуль RT LabVIEW от RT Engine.
- **Получить версию ВП** (Get VI Version) – получает версию LabVIEW, с помощью которой производилось последнее сохранение ВП.
- **Массовая компиляция** (Mass Compile) – загружает и компилирует все ВП в каталоге и во всех его подкаталогах.
- **Завершение управления клиентом удаленной лицевой панелью** (Remote Panel Client Release Control) – возвращает серверу управление удаленной лицевой панелью. Сервер предоставляет управление следующему клиенту в очереди или восстанавливает управление, если очередь запросов пуста.
- **Запрос клиентом управления удаленной лицевой панелью** (Remote Panel Client Request Control) – запрашивает управление лицевой панелью ВП на компьютере сервера.
- **Закрытие соединения с сервером удаленной лицевой панели** (Remote Panel Close Connection To Server) – закрывает соединение с ВП на компьютере сервера.
- **Открытие соединения с сервером удаленной лицевой панели** (Remote Panel Open Connection To Server) – открывает соединение с сервером и отображает лицевую панель ВП на компьютере сервера. Также может быть запрошено управление лицевой панелью.
- **Разрешить символический путь** (Resolve Symbolic Path) – если входной путь представляет символический путь LabVIEW, такой, какой возвраща-

ется свойством ВП **Путь к документу** (Document Path), то этот метод преобразует его в абсолютный путь. Например, если к узлу подключен путь <helpdir>: \glang.chm, то этот метод вернет C:\Program Files\National Instruments\LabVIEW\help\glang.chm.

Приложение 7
Методы класса ВП

- **Прервать ВП** (Abort VI) – прерывает выполнение ВП.
- **Очистить историю** (Clear History) – очищает историю исправлений ВП с заданной ссылкой.
- **Закрыть лицевую панель** (Close FP) – закрывает окно лицевой панели.
- **Экспортировать строки ВП** (Export VI Strings) – экспортирует следующие строки, характеризующие ВП и элементы лицевой панели в теговый текстовый файл: имя и описание ВП, ярлыки заголовков объекта, свободные ярлыки объекта, данные по умолчанию (строки, таблицы, пути и массивы с типом данных по умолчанию), частные данные (имена окон-списков, заголовки строк и столбцов таблиц, имена графических индикаторов, имена курсоров графиков и заголовки страниц табличного элемента управления) и данные полиморфных ВП (имена реализаций полиморфного ВП и селектор краткого меню).
- **Получить значения всех элементов управления** (Get All Control Values) – получает значения всех элементов управления или всех индикаторов ВП в виде приведенных данных.
- **Получить значения всех элементов управления [Вариант]** (Get All Control Values [Variant]) – Получает значения всех элементов управления или индикаторов ВП в виде данных типа Вариант. Этот метод возвращает массив кластеров, которые содержат имена элементов управления или индикаторов и их значения в виде данных типа Вариант.
- **Получить значение элемента управления** (Get Control Value) – получает значение элемента управления или индикатора с заданным именем в виде приведенных данных.
- **Получить значение элемента управления [Вариант]** (Get Control Value [Variant]) – получает значение элемента управления или индикатора с заданным именем в виде данных типа Вариант.
- **Получить режим масштабирования лицевой панели** (Get Front Panel Scaling Mode) – получает режим масштабирования лицевой панели. Выходной параметр **режим масштабирования** (Scaling Mode) показывает сле-

дующие варианты режима масштабирования: 0 – без масштабирования; 1 – масштабировать все элементы по размеру панели; 2 – масштабировать единичный объект по размеру панели. Если выбран режим **масштабировать единичный объект по размеру панели** (scale single object with pane), то параметр **масштабируемый объект** (Scalable Object) возвращает ссылку на элемент управления/индикатор лицевой панели, который установлен для масштабирования. В других случаях параметр возвращает значение **Не ссылка** (Not a Reference).

- **Получить состояние защиты** (Get Lock State) – возвращает состояние защиты ВП и показывает наличие пароля в кэше пароля.
- **Получить изображение лицевой панели** (Get Panel Image) – возвращает изображение лицевой панели.
- **Получить иконку ВП как данные изображения** (Get VI Icon as Image Data) – возвращает иконку ВП как кластер данных изображения, который позволяет отобразить ее с помощью рисунка (picture), используя ВП **Рисовать приведенное изображение** (Draw Flattened Pixmap), или сохранить изображение в файле, используя ВП из палитры **Форматы графиков** (Graphics Formats).
- **Импортировать строки ВП** (Import VI Strings) – импортирует строки, перечисленные выше при рассмотрении метода **Экспортировать строки ВП**.
- **Блокировать управление удаленной панелью** (Lock Remote Panel Control) – Возвращает серверу управление лицевой панелью и ставит в очередь любые запросы от клиента к лицевой панели. LabVIEW игнорирует этот метод если лицевая панель уже блокирована. Этот метод аналогичен пунктам **восстановить управление** (Regain Control) и **блокировать управление** (Lock Control) контекстного меню.
- **Сделать текущие значения значениями по умолчанию** (Make Current Values Default) – изменяет значения по умолчанию всех элементов управления на текущие значения. Этот метод доступен только в режиме редактирования.
- **Открыть лицевую панель** (Open FP) – Открывает окно лицевой панели.
- **Вывести на принтер лицевую панель** (Print Panel To Printer) – выводит на текущий принтер только лицевую панель.
- **Сохранить информацию о ВП в файле HTML** (Print VI To HTML) – сохраняет информацию о ВП в файле HTML и сохраняет графики во внешних файлах. Для отображения HTML файла в текущем Web-браузере можно использовать ВП **Открыть URL в браузере по умолчанию** (Open URL in Default Browser).
- **Вывести на принтер информацию о ВП** (Print VI To Printer) – выводит информацию о ВП на принтер.
- **Сохранить информацию о ВП в файле RTF** (Print VI To RTF) – сохраняет информацию о ВП в файле RTF.
- **Сохранить информацию о ВП в текстовом файле** (Print VI To Text) – сохраняет информацию о ВП в текстовом файле.

- **Изменить все значения на значения по умолчанию** (Reinitialize All To Default) – изменяет текущие значения всех элементов управления лицевой панели на значения по умолчанию.
- **Соединения клиента с удаленной панелью** (Remote Panel Client Connections) – возвращает массив кластеров, содержащий информацию о соединениях клиентов, наблюдающих или управляющих ВП.
- **Закрыть соединение удаленной лицевой панели** с **клиентом** (Remote Panel Close Connection To Client) – закрывает соединение клиента с удаленной лицевой панелью.
- **Вернуть ВП** (Revert VI) – игнорирует изменения и перезагружает ВП с диска.
- **Выполнить ВП** (Run VI) – запускает выполнение ВП подобно кнопке **выполнить** (Run). Это существенно отличается от вызова ВП тем, что текущие значения всех элементов управления используются для выполнения (никакие параметры не передаются). Этот метод также игнорирует свойство ВП **Показать лицевую панель при вызове** (Show FP on Call) и **закрыть после вызова** (Close After Call).
- **Сохранить для предыдущих версий** (Save For Previous) – сохраняет копию ВП, которая читается LabVIEW версии 6.1 и более поздними.
- **Сохранить ВП** (Save Instrument) – сохраняет ВП.
- **Сохранить меню периода выполнения** – сохраняет меню периода выполнения в файл, заданный с помощью пути. Этот метод выполняется только при работе ВП. Он сохраняет только пункты меню с правильными тэгами.
- **Сохранить иконку ВП в файле** (Save VI Icon to File) – сохраняет изображение иконки ВП в файле.
- **Установить значение элемента управления** (Set Control Value) – устанавливает значение именованного элемента управления или индикатора с помощью приведенных данных.
- **Установить значение элемента управления [Вариант]** (Set Control Value [Variant]) – устанавливает значение именованного элемента управления или индикатора с помощью данных типа Вариант. Для этого метода могут использоваться данные любого типа без применения функции **Преобразование к типу Вариант** (To Variant).
- **Установить режим масштабирования лицевой панели** (Set Front Panel Scaling Mode) – устанавливает режим масштабирования лицевой панели. Режимы масштабирования были приведены выше при рассмотрении метода **Получить режим масштабирования лицевой панели**.
- **Установить состояние защиты** (Set Lock State) – устанавливает состояние защиты ВП. Если на вход **интерактивный** (interactive) подано состояние ЛОЖЬ (по умолчанию), то можно использовать вход **пароль** (password) для разблокирования защищенного паролем ВП или установки пароля на незащищенный ВП. Если на вход **интерактивный** (interactive) подано состояние ИСТИНА, то LabVIEW игнорирует вход пароля и отображает диалоговое окно, которое позволяет изменить состояние защиты.

- **Установить иконку ВП из файла** (Set VI Icon from File) – устанавливает изображение иконки ВП из файла.
- **Установить иконку ВП из данных изображения** (Set VI Icon from Image Data) – устанавливает изображение иконки ВП из данных изображения.
- **Разблокировать управление удаленной панелью** (Unlock Remote Panel Control) – если лицевая панель заблокирована, то этот метод предоставляет управление следующему клиенту в очереди. Если очередь клиентов пуста, то метод разблокирует лицевую панель. LabVIEW игнорирует данный метод, если лицевая панель уже разблокирована. Это метод аналогичен пункту **Разблокировать управление** (Unlock Control) контекстного меню.

Приложение 8
Характеристики свойства и метода

Следующий список определяет характеристики для каждого свойства и метода.

- **Возможность установки при выполнении ВП** (Settable when the VI is running) – определяет возможность записи значения этого свойства или метода при выполнении ВП. Если такая возможность не предусмотрена, то LabVIEW возвращает ошибку 1000 или 1073. Эта характеристика не применяется для свойств и методов класса Приложение.
- **Необходимость получения удостоверения перед использованием** (Need to authenticate before use) – определяет необходимость получить удостоверение перед тем, как использовать свойство или метод. Например, пользователю может быть необходимо разблокировать ВП, защищенный паролем, перед использованием метода, установленного в состояние ИСТИНА.
- **Требует загрузки блок-диаграммы** (Requires the block diagram to be loaded) – определяет необходимость загрузки блок-диаграммы в память. Эта характеристика не применяется для свойств и методов класса Приложение.
- **Доступны только в локальной среде LabVIEW** (Available on local LabVIEW only) – определяет возможность использования свойства или метода только в локальной среде LabVIEW. При установке значения ИСТИНА ВП, выполняющийся удаленно, не сможет вызвать другой ВП, у которого установлены данное свойство или метод. Только локальные ссылки на это свойство или метод допустимы.
- **Требует загрузки лицевой панели** (Requires the front panel to be loaded) – определяет необходимость загрузки лицевой панели в память. Эта характеристика не применяется для свойств и методов класса Приложение.
- **Должен ожидать пока пользовательский интерфейс неактивен** (Must wait until user interface is idle) – определяет, что можно использовать это свойство или метод, пока пользователь выбирает пункт в диалоговом окне, следит за элементом управления и т. д.
- **Доступно с управляющими ВП** (Available with control VIs) – определяет возможность использования этого свойства и метода с ВП, которые явля-

ются управляющими. Эта характеристика применима только к свойствам и методам класса ВП.

- **Доступно с глобальными ВП** (Available with global VIs) – определяет возможность использования этого свойства и метода с ВП, которые являются глобальными. Эта характеристика применима только к свойствам и методам класса ВП.
- **Доступно с определениями строгого типа** (Available with strict type definitions) – определяет возможность использования этого свойства и метода с определениями строгого типа. Эта характеристика применима лишь к свойствам и методам класса ВП.
- **Доступно с полиморфными ВП** (Available with polymorphic VIs) – определяет возможность использования этого свойства и метода с полиморфными ВП. Эта характеристика применима только к свойствам и методам класса ВП.
- **Доступно во встроенном приложении** (Available in Run-Time Engine) – определяет возможность использования этого свойства и метода во встроенном приложении. Некоторые свойства и методы допустимы лишь при редактировании.
- **Разрешения** (Permissions) – определяет отношение этого свойства к только читаемому (Read Only), только записываемому (Write Only), читаемому/записываемому (Read/Write) или записываемому только в режиме редактирования (Write In Edit Only).

Литература

1. Жарков Ф. П., Каратаев В. В, Никифоров В. Ф., Панов В. С. Использование виртуальных инструментов LabVIEW / Под ред. К. С. Демирчяна и В. Г. Миронова. – М.: Радио и связь, 1999.
2. Тревис Дж. LabVIEW для всех / Джефри Тревис / Пер. с англ. Клушина Н. А. – М.: ДМК Пресс; Приборкомплект, 2004.
3. Пейч Л. И., Точилин Д. А., Поллак Б. П. LabVIEW для новичков и специалистов. – М.: Горячая линия – Телеком, 2004.
4. LabVIEW Help.
5. LabVIEW User Manual.
6. LabVIEW Measurements Manual.
7. Цифровая обработка сигналов / А. Б. Сергиенко. – СПб.: Питер, 2003.
8. Гутников В. С. Фильтрация измерительных сигналов. – Л.: Энергоатомиздат, 1990.
9. Отнес Р., Эноксон Л. Прикладной анализ временных рядов. Основные методы. – М.: Мир, 1982.
10. Рабинер Л., Гоулд Б. Теория и применение цифровой обработки сигналов / Пер. с англ. Под. ред. Ю. И. Александрова. – М.: Мир, 1978.
11. Гилл Ф., Мюррей У., Райт М. Практическая оптимизация / Пер. с англ. – М.: Мир, 1985.
12. Васильев Ф. П. Численные методы решения экстремальных задач: Учебное пособие для вузов. – М.: Наука, 1988.
13. Суранов А. Я., Белых С. В. Микропроцессорный регистратор одномерных изображений на базе фотодиодного приемника. – ПТЭ. – 2003. – № 6. С. 140—142.
14. Таненбаум Э. Современные операционные системы. 2-е изд. – СПб.: Питер, 2002.
15. Дэви Чеппел. Технологии ActiveX и OLE / Пер. с англ. – М.: Издательский отдел «Русская редакция», 1997.
16. Дайл Роджерсон. Основы COM / Пер. с англ. – М.: Издательский отдел «Русская редакция», 1997.
17. Дэвид С. Плат. Знакомство с Microsoft. NET / Пер. с англ. – М.: Издательско-торговый дом «Русская редакция», 2001.

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 7
Справочник по функциям

Главный редактор *Мовчан Д. А.*
dm@dmkpress.ru
Корректор *Синяева Г. И.*
Верстка *Шарапов В. Ю.*
Дизайн обложки *Мовчан А. Г.*

Подписано в печать 10.01.2005. Формат 70×100 $^{1}/_{16}$.
Гарнитура «Петербург». Печать офсетная.
Усл. печ. л. 32. Тираж 3000 экз.
№

Электронный адрес издательства: www.dmkpress.ru