Основы LabVIEW™ 1
Учебное пособие

Программное обеспечение курса версии 2009 г.

Издание – октябрь 2009

Шифр 325290A-01

Авторское право

Торговые марки

Патенты

Для получения информации о патентах, которыми защищены продукция или технологии National Instruments, запустите команду **Help»Patents** из главного меню Вашего программного обеспечения, откройте файл `patents.txt` на имеющемся у Вас компакт-диске или зайдите на сайт `ni.com/patents`.

Техническая поддержка по всему миру и информация о выпускаемой продукции

ni.com

Штаб-квартира корпорации National Instruments

11500 North Mopac Expressway Austin, Texas 78759-3504 USA Tel: 512 683 0100

Офисы по всему миру

Australia 1800 300 800, Austria 43 662 457990-0, Belgium 32 (0) 2 757 0020, Brazil 55 11 3262 3599, Canada 800 433 3488, China 86 21 5050 9800, Czech Republic 420 224 235 774, Denmark 45 45 76 26 00, Finland 358 (0) 9 725 72511, France 01 57 66 24 24, Germany 49 89 7413130, India 91 80 41190000, Israel 972 3 6393737, Italy 39 02 41309277, Japan 0120-527196, Korea 82 02 3451 3400, Lebanon 961 (0) 1 33 28 28, Malaysia 1800 887710, Mexico 01 800 010 0793, Netherlands 31 (0) 348 433 466, New Zealand 0800 553 322, Norway 47 (0) 66 90 76 60, Poland 48 22 328 90 10, Portugal 351 210 311 210, Russia 7 495 783 6851, Singapore 1800 226 5886, Slovenia 386 3 425 42 00, South Africa 27 0 11 805 8197, Spain 34 91 640 0085, Sweden 46 (0) 8 587 895 00, Switzerland 41 56 2005151, Taiwan 886 02 2377 2222, Thailand 662 278 6777, Turkey 90 212 279 3031, United Kingdom 44 (0) 1635 523545

За подробной информацией о поддержке обратитесь к приложению *Дополнительная информация и ресурсы*. Чтобы оставить свои комментарии о документации National Instruments, зайдите на сайт `ni.com/info` и введите код обратной связи `feedback`.

Содержание

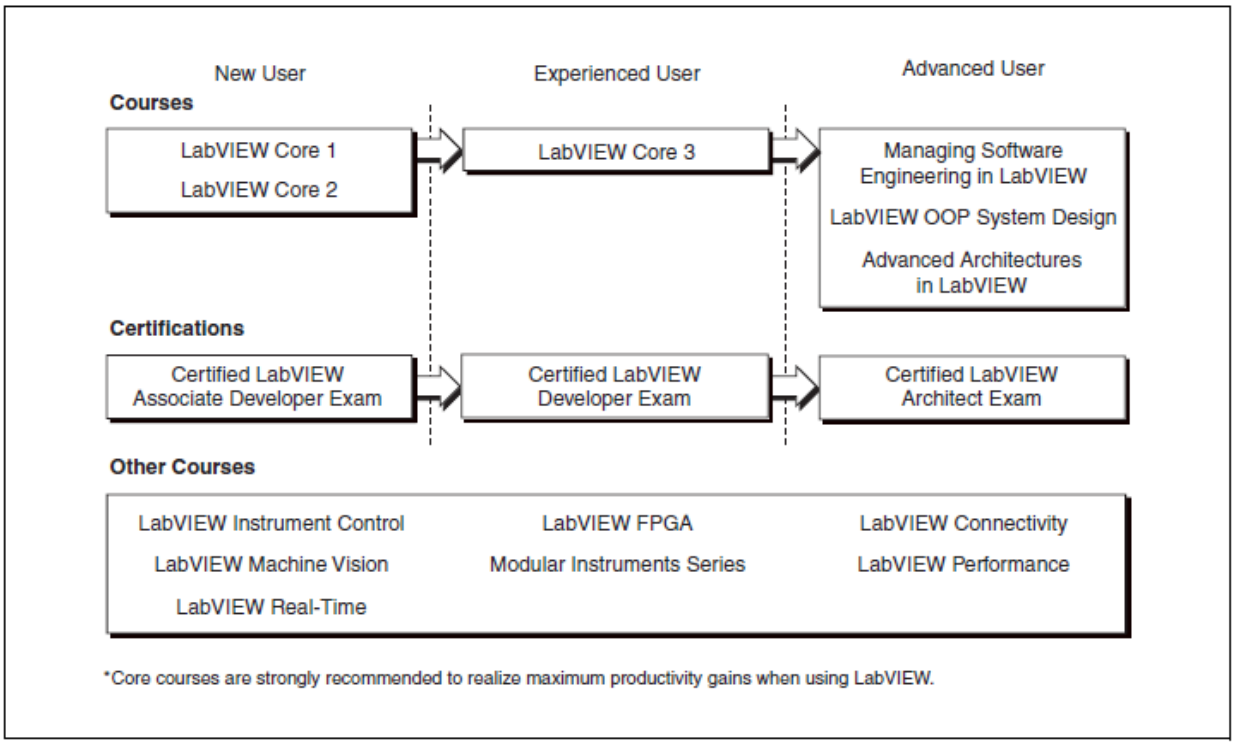
Для студентов

Благодарим вас за приобретение обучающего комплекта по курсу *LabVIEW Core 1*. Начать проектирование приложений вы сможете сразу после освоения курса. Настоящее руководство и прилагаемое программное обеспечение используются в трехдневном практическом курсе *LabVIEW Core 1*.

Вы можете приобрести полный комплект этого курса после соответствующего взноса за регистрацию, если вы зарегистрируетесь в течение 90 дней от даты приобретения. Расписание занятий "онлайн", программу курса, учебные центры и категории регистрации можно найти на сайте `ni.com/training`.

Примечание: Для получения обновлений и исправлений учебных материалов к лекциям и упражнениям зайдите на сайт `ni.com/training` и введите код информации `Core 1.`.

Курс *LabVIEW Core 1* является частью серии курсов, созданных для приобретения опыта работы в LabVIEW и подготовки к сдаче аттестационного экзамена. На следующем рисунке показана структура серии курсов. Подробную информацию об аттестации в NI можно найти по ссылке `ni.com/training`.



A. Об этом руководстве

Используйте это учебное пособие для того, чтобы, изучив принципы, технику и особенности программирования в LabVIEW, а также VI и функции LabVIEW, вы смогли создавать приложения для тестирования, измерения, сбора данных, управления измерительными приборами, регистрации данных, обработки результатов измерений и генерации отчетов. В настоящем руководстве предполагается, что вы знакомы с Windows и имеете опыт составления алгоритмов в форме диаграмм или блок-схем. Учебное пособие и руководство к выполнению упражнений разбиты на лекции (занятия), как описано ниже.

В учебном пособии каждая лекция состоит из:

- Введения, в котором приводятся цель лекции и что подлежит изучению
- Содержание тем (разделов) лекции
- Заключительный контрольный опрос, который позволяет проверить и закрепить важные принципы и навыки, изученные и приобретенные на лекции

В руководстве к выполнению упражнений для каждого занятия включены:

- Набор упражнений для закрепления темы лекции
- В некоторые занятия включены необязательные усложненные разделы упражнений или набор дополнительных упражнений, которые можно выполнять, если позволяет время

В отдельных упражнениях используется одно из следующих устройств National Instruments:

- Встраиваемый многофункциональный модуль сбора данных (DAQ), соединенный со вспомогательным сигнальным блоком, в состав которого входят датчик температуры, функциональный генератор и светодиодные индикаторы
- Модуль интерфейса GPIB, соединенный с симулятором измерительных приборов (NI Instrument Simulator)

Вы можете выполнять упражнения, даже если у вас нет упомянутого оборудования. Альтернативные инструкции позволяют выполнять упражнения без технических средств. Те задания, для выполнения которых обязательно использование оборудования, обозначены иконкой показанной слева. Вы можете заменить указанное в упражнениях оборудование на другое аналогичное. Например, вместо симулятора NI Instrument Simulator вы можете использовать какой-нибудь измерительный прибор с интерфейсом GPIB, а в качестве источника сигналов другое DAQ устройство National Instrument, например, функциональный генератор.

B. С чего нужно начинать

Необходимые условия для обучения

Обязательным условием для курсов являются минимальные знания всеми студентами основных вопросов теории и принципов, относящихся к содержанию курса *LabVIEW Core 1*. Для получения максимальной эффективности обучения подготовьте все условия до начала занятий.

Доступ к каждому материалу по упомянутым вопросам открыт по адресу `ni.com/info` после ввода кода информации, соответствующего выбранной теме:

- LabVIEW Core 1 - The Software Development Method (Методы проектирования программного обеспечения) (info code: SoftDev)
- Introduction to Data Acquisition (Введение в сбор данных) (info code: DAQ)
- GPIB Instrument Control Tutorial (Учебник по управлению измерительными приборами) (info code: GPIB)
- Serial Communication Overview (Обзор средств обмена данными через последовательный порт) (info code: Serial)

Используемые компоненты

Прежде, чем начать работу с настоящим руководством, убедитесь в наличии следующих компонентов:

- Windows 2000 или выше, инсталлированная на вашем компьютере. Курс оптимизирован под Windows XP.
- Многофункциональное DAQ устройство, сконфигурированное в Measurement & Automation Explorer (MAX) как `device 1`
- Вспомогательный сигнальный блок, проводники и кабели
- Модуль интерфейса GPIB
- NI Instrument Simulator и блок питания
- Полная (Full) или профессиональная (Professional) система проектирования LabVIEW 2009 или выше
- DAQmx 8.9.5 или выше
- NI-488.2 2.7.1 или выше
- NI-CAN 2.6.3 или выше (для студентов, изучающих CAN)
- Кабель к последовательному порту
- Кабель для порта GPIB
- Компакт-диск с курсом LabVIEW Core 1, инсталлированный в следующие папки:

Имя папки	Описание
Exercises	Папка для VI, создаваемых в процессе обучения, и для VI, создаваемых при выполнении конкретных упражнений. Кроме того, в этой папке содержатся subVI, необходимые для некоторых упражнений, и архивный файл (nidevsim.zip), содержащий драйвер LabVIEW для симулятора NI Instrument Simulator
Solutions	Папка с решениями всех заданий курса

C. Инсталляция программного обеспечения для курса

Выполните следующие действия для инсталляции программного обеспечения к курсу:

1. Вставьте компакт-диск с материалами курса в компьютер
2. Инсталлируйте файлы Exercises и Solutions по указанным адресам

Примечание: Папки, имена которых заключены в угловые скобки, например, <Exercises>, должны находиться в корневом каталоге.

D. Цель курса

Этот курс научит вас:

- Понимать назначение лицевых панелей, блок-диаграмм, иконок и панелей подключения
- Использовать программные структуры и типы данных LabVIEW
- Применять различные способы редактирования и отладки
- Создавать и сохранять VI так, чтобы их можно было использовать в качестве subVI
- Отображать и регистрировать данные
- Создавать приложения на основе встраиваемых DAQ-устройств
- Создавать приложения с использованием последовательного порта и приборов с интерфейсом GPIB

В этом курсе *не* изучаются:

- Все встроенные VI, функции или объекты; Обратитесь к *LabVIEW Help* для получения дополнительной информации о свойствах LabVIEW, не представленных в настоящем курсе
- Теория аналого-цифрового преобразования
- Принцип действия последовательного порта
- Принцип действия шины GPIB
- Разработка драйверов измерительных приборов
- Проектирование законченных приложений всеми студентами аудитории; обратитетсь к поисковику примеров NI Example Finder, выбрав в меню **Help>>Find Examples**, чтобы найти примеры VI, которые можно использовать и включать в создаваемые вами VI

E. Условные обозначения

В настоящем учебном пособии используются следующие условные обозначения:

»	Символ » служит, чтобы показать путь выбора цели во вложенных меню и диалоговых окнах. Например, последовательность **File»Page Setup»Options** означает, что следует открыть меню **File**, выбрать там пункт **Page Setup** и затем выбрать команду **Options** в появившемся диалоговом окне.
	Пиктограмма подсказки, совета.
	Пиктограмма примечания с важной информацией.
	Пиктограмма предупреждения, содержащая рекомендации, как избежать травм, потерь данных или выхода из строя системы..
	Пиктограмма сообщает, что для выполнения упражнения нужен встраиваемый модуль интерфейса GPIB или DAQ-устройство.
Bold	Полужирным шрифтом выделены пункты меню или диалоговых окон, которые нужно выбрать, а также обозначения параметров, элементов управления и кнопок на лицевой панели, диалоговых окон и их фрагментов, меню и палитр.
italic	Курсивом выделены имена переменных, важные фрагменты текста, перекрестные ссылки, а также пояснения к ключевым понятиям. Курсивом также выделено место в тексте, которое нужно заменить словом или значением.
`monospace`	Шрифтом одинаковой ширины записывается текст или отдельные символы, которые следует вводить с клавиатуры, фрагменты текстов программ, примеры программ, а также примеры синтаксиса. Этот шрифт используется также для идентификаторов дисковых накопителей, путей, папок, программ, подпрограмм, имен устройств, функций, операций, переменных, имен и расширений файлов.
`monospace bold`	Выделенным шрифтом одинаковой ширины отличаются сообщения и ответы, которые компьютер автоматически выводит на экран. Этот шрифт используется также в строках кода, отличающихся от других примеров.
Platform	Этот шрифт применяется для указания платформы и означает, что последующий текст относится только к этой платформе.

1

1. Настройка оборудования

LabVIEW – это среда графического программирования, используемая инженерами и научными сотрудниками для разработки современных систем измерений, испытаний и управления. LabVIEW позволяет интегрировать программное обеспечение с широким диапазоном аппаратных средств. В этом курсе вы будете работать с оборудованием DAQ, GPIB и оборудованием, подключаемым через последовательный порт, конфигурируя технические средства с помощью Measurement & Automation Explorer.

План занятия

A. Технические средства DAQ
B. Программное обеспечение DAQ
C. Управление измерительными приборами
D. GPIB
E. Коммуникации через последовательный порт
F. ПО для управления измерительными приборами
G. Курсовой проект

A. Технические средства DAQ

Системы сбора данных (DAQ) используют DAQ-устройства для ввода предварительно преобразованных аналоговых сигналов в компьютер с целью последующей обработки и регистрации данных программным способом. DAQ-устройства могут быть выбраны разных типов – использующие шину PCI, PCI Express, PXI, компьютерный порт USB или IEEE 1394. В настоящем разделе рассматриваются вопросы применения и конфигурирования технических средств в системах сбора данных.

Типовая DAQ-система состоит из устройств трех основных типов – коннекторного блока, кабеля и DAQ-устройства, изображенных на рис. 1-1.

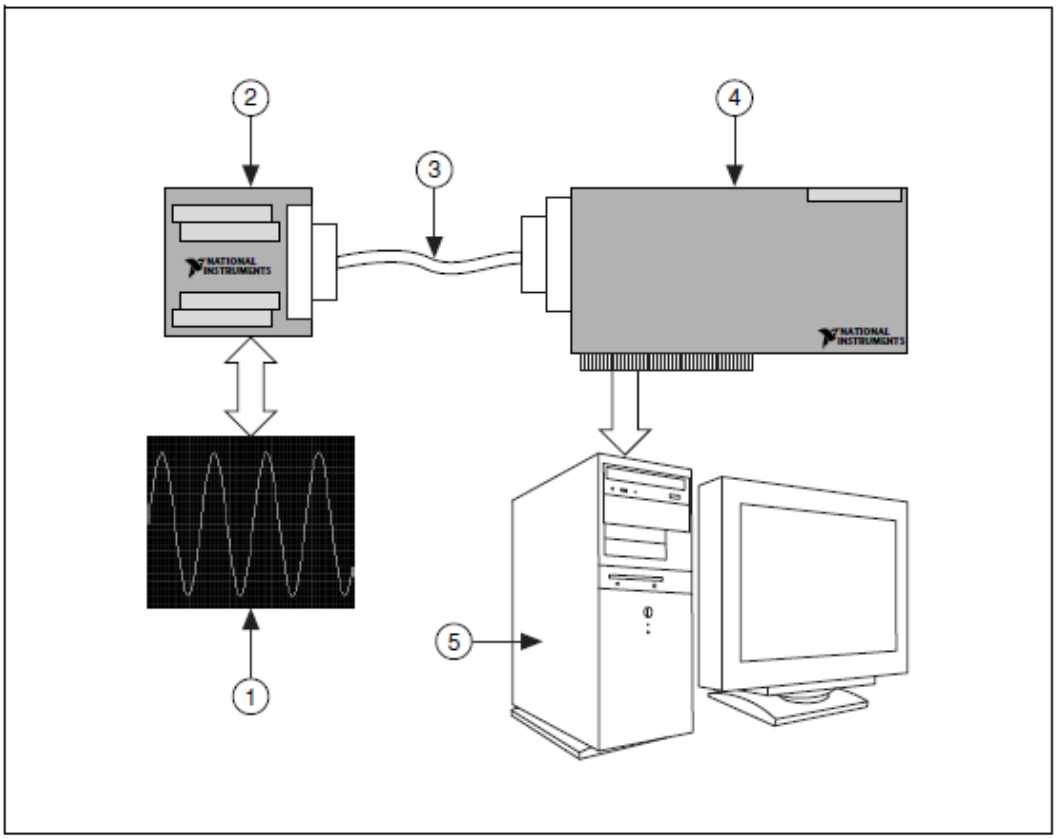


Рисунок 1.1. Типовая DAQ-система

После преобразования физической величины в измеряемый сигнал, возможно, с дополнительным аналоговым преобразованием, вам надо принять этот сигнал. Для этого необходимы коннекторный блок, кабель, DAQ-устройство и компьютер. Подобное сочетание устройств превращает компьютер в систему измерений и автоматизации.

Коннекторный блок и кабель

Коннекторный блок служит для подключения сигналов. Он состоит из винтовых или пружинных клемм для подключения сигналов и разъема для подключения кабеля связи коннекторного блока с DAQ устройством. В коннекторном блоке может быть 100, 68 или 50 клемм. Выбор типа коннекторного блока зависит от двух факторов – назначения устройства и количества измеряемых сигналов. В коннекторном блоке с 68-ю клеммами больше клемм для заземления, чем в блоке с 50-ю клеммами. Большее количество клемм заземления предотвращает наложение проводников в клеммах заземления, которое может привести к взаимному влиянию и искажению сигналов.

Коннекторный блок может быть экранированным или неэкранированным. Экранированный блок лучше защищает от помех. Некоторые коннекторные блоки обеспечивают дополнительные возможности, например, компенсацию температуры холодного спая, необходимую при измерении температуры с помощью термопар.

По кабелю сигналы передаются от коннекторного блока к DAQ устройству. Кабели поставляются в конфигурациях на 100, 68 и 50 контактов. Выбор конфигурации кабеля определяется используемым типом коннекторного блока и DAQ-устройства. Кабели, как и коннекторные блоки могут быть экранированными и неэкранированными.

Дополнительную информацию о специальных типах коннекторных блоков и кабелей можно найти в разделе DAQ каталога National Instrument или по ссылке `ni.com/products`.

Аксессуары для сигналов

На рисунке 1-2 показан используемый в курсе коннекторный блок – вспомогательный сигнальный блок (DAQ Signal Accessory) для систем сбора данных.

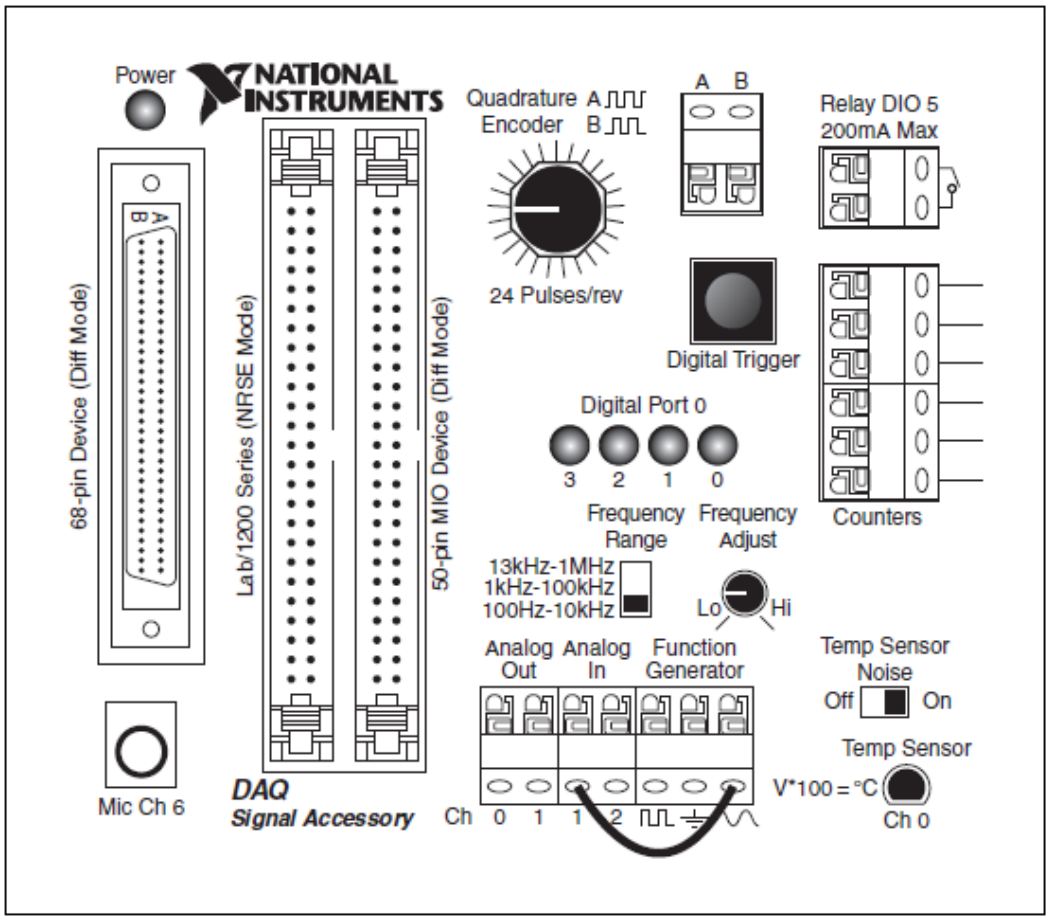


Рисунок 1.2. Вспомогательный блок для сбора системы данных

Вспомогательный сигнальный блок – это коннекторный блок, спроектированный специально для целей обучения. Он снабжен тремя различными разъемами для кабелей, подходящих к различным DAQ-устройствам, и пружинными клеммами для подключения сигналов. Вы можете воспользоваться тремя аналоговыми входными каналами, один из которых подключен к датчику температуры, а два других – к аналоговым выходным каналам.

В состав вспомогательного блока входит функциональный генератор с переключателем для выбора диапазона частот и регулятором частоты. С выхода генератора можно получить сигнал синусоидальной или прямоугольной формы. Клемма заземления находится между клеммами выходов синусоидального и прямоугольного сигналов.

Кнопка цифрового запуска формирует ТТЛ импульс для запуска операций аналогового ввода или вывода. При нажатии на кнопку запуска уровень сигнала изменяется с +5 В на 0 В и возвращается обратно при отпускании кнопки. 4 светодиода подключены к первым четырем цифровым линиям

DAQ-устройства. На светодиодах сигналы отображаются в инверсной логике – светодиод отключен при высоком уровне на линии и наоборот – включен – при низком.

Во вспомогательном блоке находится также квадратурный энкодер, который формирует две последовательности импульсов при включении кнопки энкодера. На клеммы выведены входные и выходные сигналы двух счетчиков DAQ-устройства. Кроме того, во вспомогательном блоке имеются реле, вход для термопары и разъем для микрофона.

Применение DAQ-устройства

Большинство DAQ устройств имеют 4 стандартных узла: аналогового ввода, аналогового вывода, цифрового ввода-вывода и блок счетчиков.

Вы можете передавать результаты измерений сигналов DAQ-устройством в компьютер через различные шины. Например, вы можете использовать DAQ-устройство, встраиваемое в шину PCI или PCI Express компьютера, DAQ-устройство, подключенное к сокету PCMCIA лаптопа, или DAQ-устройство, подключенное к порту USB компьютера. Портативные, универсальные и надежные измерительные системы могут быть созданы также на основе стандартов PXI/Compact PCI.

Если у вас нет DAQ-устройства, его можно симулировать в Measurement & Automation Explorer (MAX), чтобы протестировать ваше программное обеспечение. Вы научитесь симулировать устройства в разделе *Симуляция DAQ-устройств* этой лекции.

Дополнительную информацию о специальных типах DAQ-устройств можно найти в разделе DAQ каталога National Instrument или по ссылке `ni.com/products`.

Аналоговый ввод

Аналоговый ввод – это процесс измерения аналогового сигнала и передачи результатов измерения в компьютер для обработки, визуализации и запоминания. Аналоговым сигналом называется сигнал, изменяющийся непрерывно. Аналоговый ввод, как правило, предполагает измерение напряжения или тока. Для аналогового ввода могут применяться различные типы устройств – многофункциональные (MIO) DAQ-устройства, высокоскоростные дигитайзеры, цифровые мультиметры (DMM) и устройства сбора динамически изменяющихся сигналов (Dynamic Signal Acquisition).

Ввод аналогового сигнала в компьютер осуществляется путем его аналого-цифрового преобразования, при котором аналоговый электрический сигнал преобразуется в данные цифровой формы, которые могут обрабатываться компьютером. Аналого-цифровое преобразователи (ADC) – это схемные компоненты, преобразующие уровни напряжений в набор нулей и единиц.

ADC воспринимают аналоговый сигнал по положительному или отрицательному фронту импульсов выборки (Sample Clock). По каждому

импульсу выборки ADC фиксирует уровень аналогового сигнала, а затем измеряет его, преобразуя в цифровое значение. Импульсами выборки определяется частота получения отсчетов входного сигнала. Поскольку входной неизвестный сигнал является непрерывным сигналом реального физического окружения, его можно рассматривать, как сигнал, заданный с бесконечной точностью, ADC аппроксимирует его уровнями конечной точности. Уровни, аппроксимирующие сигнал, могут быть затем преобразованы в последовательности чисел. Подобная операция не требуется для некоторых методов аналого-цифрового преобразования, т.к. она выполняется непосредственно в ADC.

Аналоговый вывод

Аналоговый вывод – это процесс генерации аналогового электрического сигнала, выполняемый путем цифроаналогового (D/A) преобразования данных, получаемых в компьютере. Доступные типы выводимых аналоговых сигналов – напряжение и ток. Для вывода напряжения или тока в компьютер должно быть установлено устройство, способное формировать эти типы сигналов.

Цифроаналоговое преобразование – операция, обратная аналого-цифровому преобразованию. При цифроаналоговом преобразовании данные, формируемые компьютером, могут быть получены с помощью операции аналогового ввода или сгенерированы программным способом. Цифроаналоговый преобразователь (DAC) принимает каждое значение этих данных и, трансформируя их в уровень напряжения, выдает на выходной контакт устройства. Генерируемый DAC сигнал может быть отправлен в другие устройства или схемы.

Каждое новое значение на выходе DAC появляется по импульсу обновления (Update Clock), назначение которого аналогично назначению импульса выборки для ADC. По каждому импульсу обновления DAC преобразует входное число (код) в уровень аналогового напряжения, которое передается на выходной контакт устройства. При высокой частоте импульсов обновления DAC генерирует сигнал, похожий на непрерывный и гладкий.

Цифровой ввод/вывод

Цифровые сигналы – это электрические сигналы, которые передают цифровые данные по проводникам. Обычно эти сигналы представляются двумя состояниями – включено/выключено, высокий/низкий или 1 и 0. При отправке цифрового сигнала передатчик формирует уровень напряжения на проводнике, а приемник по этому уровню определяет значение отправленного сигнала. Диапазон уровней напряжения для каждого из значений цифрового сигнала стандартизирован. Цифровые сигналы имеют множество применений; простейшее из них – управление цифровыми устройствами с конечным числом состояний, например, переключателями или светодиодами, или определение состояний подобных устройств. Цифровые сигналы могут также передавать данные; их можно использовать для программирования устройств или обмена информацией между устройствами. Кроме того, цифровые сигналы можно использовать в

качестве сигналов тактирования или запуска для управления или синхронизации других измерений.

Цифровые линии DAQ-устройства могут использоваться для ввода цифровых данных, основанного на программной синхронизации. В некоторых устройствах эти линии можно сконфигурировать индивидуально на ввод или вывод цифровых значений. Каждая линия соответствует каналу в задаче.

Вы можете использовать цифровой порт(ы) DAQ-устройства для ввода цифровых данных от совокупности цифровых линий. Подобный ввод синхронизируется программно. Порты можно сконфигурировать индивидуально на ввод или вывод цифровых значений. Каждый порт соответствует каналу в задаче.

Счетчики

Счетчик – это цифровое устройство для таймирования. Обычно счетчики применяют для счета событий, измерения частоты, периода, положения, а также для генерации импульсов

Когда вы конфигурируете счетчик на простой счет событий, он инкрементируется при поступлении активного фронта от источника событий. При этом счетчик начинает счет по команде начала счета. Счетчик характеризуется разрешающей способностью (Counter Resolution), фиксированным числом, которое определяет, сколько событий может быть подсчитано. Например, 24-разрядный счетчик может считать до:

$$2^{(\text{Counter Resolution})} - 1 = 2^{24} - 1 = 16{,}777{,}215$$

Это число является пределом счета и, когда счетчик досчитает до 16,777,215, следующим активным фронтом счетчик переполняется и счет начинается с 0.

B. Программное обеспечение DAQ

Устройства сбора данных National Instrument снабжаются драйверами, которые обеспечивают связь между устройством и прикладной программой. Возможен выбор между двумя различными классами драйверов: NI-DAQmx и Traditional NI-DAQ. Вы можете использовать LabVIEW с каждым из этих классов драйверов.

Кроме того, вы можете использовать MAX для конфигурирования устройств сбора данных. В этом разделе вы познакомитесь с семействами драйверов и научитесь применять MAX для конфигурирования устройств сбора данных.

Применение NI-DAQ

NI-DAQ 7.x состоит из двух семейств драйверов – Traditional NI-DAQ (Legacy – наследуемый) и NI-DAQmx, каждый со своим собственным программным интерфейсом приложений (API), конфигурированием

технических и программных средств. NI-DAQ 8.0 и более новые версии поставляются только с драйверами NI-DAQmx, заменяющими семейство Traditional NI-DAQ (Legacy).

- Traditional NI-DAQ (Legacy) – это обновленная версия NI-DAQ 6.9x, более ранней версии NI-DAQ. Traditional NI-DAQ (Legacy) состоит из таких же VI и функции и работает так же, как и NI-DAQ 6.9x. Вы можете использовать Traditional NI-DAQ (Legacy) на том же компьютере, что и драйверы NI-DAQmx, которые не могут работать вместе с драйверами NI-DAQ 6.9x. Однако вы не можете использовать Traditional NI-DAQ (Legacy) под операционной системой Windows Vista.
- NI-DAQmx – последняя версия драйверов NI-DAQ с новыми VI, функциями и средствами проектирования для управления измерительными устройствами. Преимущества NI-DAQmx по сравнению с предыдущими версиями NI-DAQ: помощник DAQ Assistant для конфигурирования каналов и задач измерений, увеличенная производительность, включая одноточечный аналоговый ввод-вывод и многопоточность; а также более простой API для создания DAQ-приложений, использующий иные VI и функции, чем в более ранних версиях NI-DAQ.

Примечание:
(Windows) LabVIEW поддерживает NI-DAQmx и DAQ Assistant.
(Mac OS) LabVIEW поддерживает NI-DAQmx Base, но не поддерживает DAQ Assistant.
(Linux) LabVIEW поддерживает NI-DAQmx, но не поддерживает DAQ Assistant.

Traditional NI-DAQ (Legacy) и NI-DAQmx поддерживают различные наборы устройств. Список поддерживаемых устройств находится на сайте National Instruments в разделе Data Acquisition (DAQ) Hardware.

Конфигурирование оборудования сбора данных

Прежде чем использовать устройство сбора данных, нужно его сконфигурировать, чтобы убедиться, что программное обеспечение может связываться с устройством. В этой аудитории устройства уже сконфигурированы.

Windows

Менеджер конфигурирования Windows отслеживает все технические средства, установленные в компьютер, в том числе и DAQ-устройства National Instruments. Если у вас устройство Plug & Play (PnP), как, например, устройство MIO E серии, менеджер конфигурирования Windows автоматически обнаруживает и конфигурирует устройство. Если ваше устройство не PnP или устаревшей модификации, вы должны его сконфигурировать вручную, используя функцию **Add New Hardware** на панели управления Windows.

Вы можете проверить конфигурацию Windows, открыв менеджер устройств. В разделе **Data Acqusition Devices** перечислены все DAQ-устройства,

установленные в компьютер. Для вывода диалогового окна с закладками для настройки, щелкните дважды по DAQ-устройству. На закладке **General** отображается общая информация, относящаяся к устройству. На закладке **Driver** приведена версия драйвера и место установки устройства. На закладке **Resources** указываются ресурсы, выделенные системой для устройства, такие, как уровни прерывания, DMA и базовый адрес программно конфигурируемого устройства.

Measurement & Automation Explorer

В MAXe задаются все конфигурационные параметры устройств и каналов. После установки DAQ-устройства в компьютер вы должны запустить эту утилиту конфигурирования. MAX считывает записи менеджера устройств в системном реестре Windows и присваивает каждому DAQ-устройству логический номер, который используется в LabVIEW в качестве ссылки на устройство. Открывается MAX двойным щелчком по иконке на рабочем столе или из меню LabVIEW выбором пункта **Tools»Measurement & Automation Explorer.** Ниже показано стартовое окно MAX. В MAXe конфигурируется также оборудование SCXI и SCC.

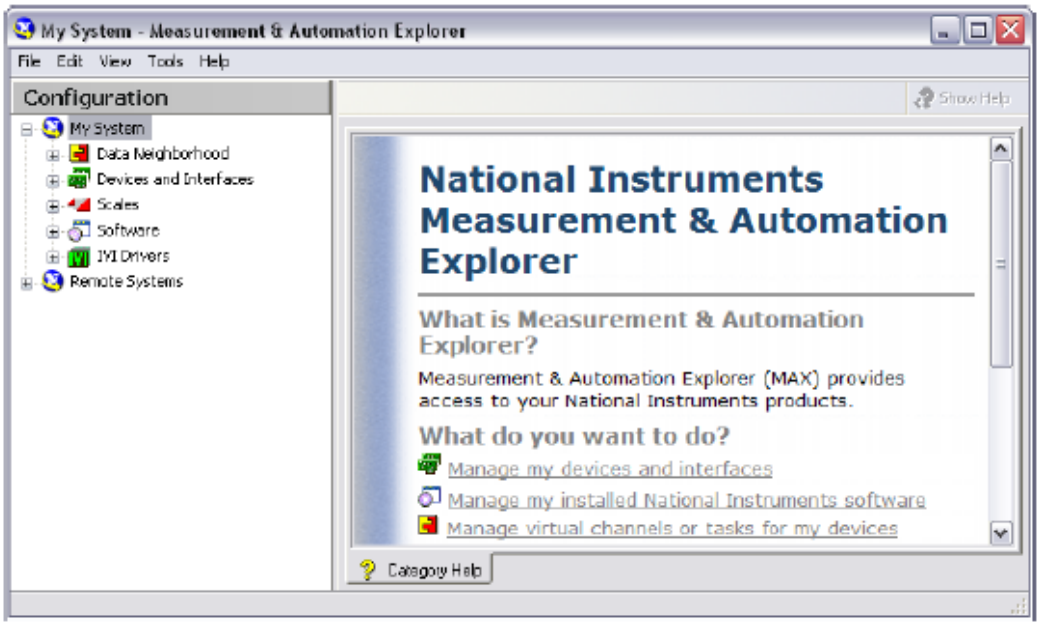


Рисунок 1.3. Стартовое окно MAX

Параметры устройства, которые вы можете установить с помощью утилиты конфигурирования, зависят от типа устройства. MAX сохраняет логический номер устройства и конфигурационные параметры в реестре Windows.

Свойство Plug & Play Windows автоматически обнаруживает и конфигурирует DAQ-устройства, не содержащие переключателей, такие, как, например, NI PCI-6024E, когда вы устанавливаете его в компьютер.

Шкалы

Вы можете сконфигурировать пользовательскую шкалу для вашей задачи измерений. Это очень полезно при работе с сенсорами, т.к. позволяет

вводить в ваше приложение отмасштабированные значения, не работая напрямую с необработанными данными. Например, в нашем курсе используется датчик температуры, который представляет значения температуры уровнями напряжения. Уравнение преобразования напряжения в температуру: `Voltage x 100 = Celsius`. После создания шкалы вы можете использовать в приложении уже не уровни напряжения, а значения температуры.

Симуляция DAQ-устройства

Вы можете создавать симуляторы NI-DAQmx устройств, поддерживаемых NI-DAQmx 7.4 и выше. С помощью симуляторов NI-DAQmx устройств можно опробывать изделия NI в составе ваших приложений, не имея их физически. Когда позднее вы получите оборудование, то сможете импортировать конфигурацию симулированных NI-DAQmx устройств на реальные устройства, используя мастер MAX Portable Configuration Wizard. С помощью симуляторов NI-DAQmx устройств можно также экспортировать конфигурацию реального устройства в систему, в которую реальные устройства не установлены. Затем, используя симулированные NI-DAQmx устройства, можно работать с приложением в программной системе, а вернувшись в оригинальную систему – можно просто импортировать приложение.

C. Управление измерительными приборами

При использовании персонального компьютера для автоматизации тестовых систем, вы не ограничены в выборе типов управляемых измерительных приборов – можно применять приборы одинаковых и разных категорий. Наиболее распространенные категории приборов с интерфейсами – это приборы с GPIB или с последовательным портом и модульные измерительные приборы. Кроме того, применяются такие типы приборов, как устройства захвата изображений, управления движением, приборы подключаемые через порты USB, Ethernet, параллельный порт, NI-CAN и другие.

При использовании персонального компьютера для управления измерительными приборами нужно знать свойства приборов, например, используемый протокол для обмена информацией. Информация о свойствах прибора содержится в документации на прибор.

D. GPIB

Стандарт ANSI/IEEE Standard 488.1-1987, известный также, как General Purpose Interface Bus (GPIB) описывает стандартный интерфейс для связи измерительных приборов и контроллеров разных изготовителей. Приборы с GPIB предоставляют инженерам по тестированию и производству широчайший выбор поставщиков и приборов, как общего назначения, так и специализированных для вертикального рынка тестовых приложений. Приборы с GPIB часто применяют, как автономные настольные приборы, управление измерениями в которых осуществляется вручную.

Автоматизировать подобные измерения можно, используя персональный компьютер для управления приборами через GPIB.

IEEE 488.1содержит информацию об электрических, механических и функциональных характеристиках. Стандарт ANSI/IEEE Standard 488.2-1992 расширяет IEEE 488.1 определением коммуникационного протокола, общего набора кодов данных и форматов, а также основного набора команд устройства. GPIB – это цифровой 8-битовый параллельный коммуникационный интерфейс со скоростью передачи данных 1 Мбайт/с и выше, использующий алгоритм рукопожатия по 3-м проводам. Шина поддерживает один системный контроллер, обычно – компьютер, и до 14-ти дополнительных приборов.

Протокол GPIB классифицирует устройства на Контроллеры, Источники сообщений и Приемники сообщений, определяя, какое из устройств осуществляет активное управление шиной. Каждому устройству присваивается уникальный первичный адрес GPIB в диапазоне от 0 до 30. Контроллер определяет соединения узлов, отвечает устройствам на запросы обслуживания, отправляет GPIB команды и передает/принимает управление шиной. Контроллер выдает Передатчику команду на передачу и размещение данных на шине. Одновременно можно адресоваться только к одному Передатчику. Контроллер обращается к Приемнику для перехода в режим чтения и считывания данных с шины GPIB. Одновременно данные с шины могут считывать несколько устройств.

Завершение передачи данных

Завершение информирует приемники, что все данные переданы. Завершить передачу данных по GPIB можно тремя способами:

- В шину GPIB входит аппаратная линия End Of Identify (EOI), которая служит для уведомления о последнем байте данных. Этот способ предпочтителен.
- Размещение специального символа End of String (EOS) в конце строки данных. Некоторые приборы используют такой способ вместо или в дополнение уведомления по линии EOI.
- Приемник подсчитывает передаваемые байты при рукопожатии и останавливает чтение, когда количество байт станет равным пределу. Этот способ часто является методом завершения по умолчанию, т.к. передача прекращается при логическом ИЛИ EOI, EOS (если используется) в сочетании со счетом байт. Из предосторожности количество байт в приемнике часто задается больше, чем ожидаемое число байт, при этом данные не теряются.

Скорость передачи данных

GPIB был разработан для обеспечения высокой скорости передачи данных, и чтобы это реализовать, необходимо ограничивать количество устройств на шине и физическое расстояние между устройствами.

Более высокую скорость передачи данных предоставляют устройства и контроллеры стандарта HS488, который является расширением GPIB и поддерживается большинством контроллеров NI.

Примечание: Дополнительную информацию о GPIB можно найти на сайте поддержки GPIB National Instrument по ссылке `ni.com/support/gpibsupp`.

E. Обмен данными через последовательный порт

При обмене информацией через последовательный порт данные передаются между компьютером и периферийными устройствами, такими, как программируемый измерительный прибор или другой компьютер. Последовательный обмен информацией использует передатчик для побитной посылки данных приемнику по одной линии. Такой способ применяют при низких скоростях обмена данными или при передаче данных на большие расстояния. Большинство компьютеров имеют один или несколько последовательных портов, так что пользователю не требуется никакого дополнительного оборудования, кроме кабеля для подключения прибора к компьютеру или для соединения двух компьютеров между собой.

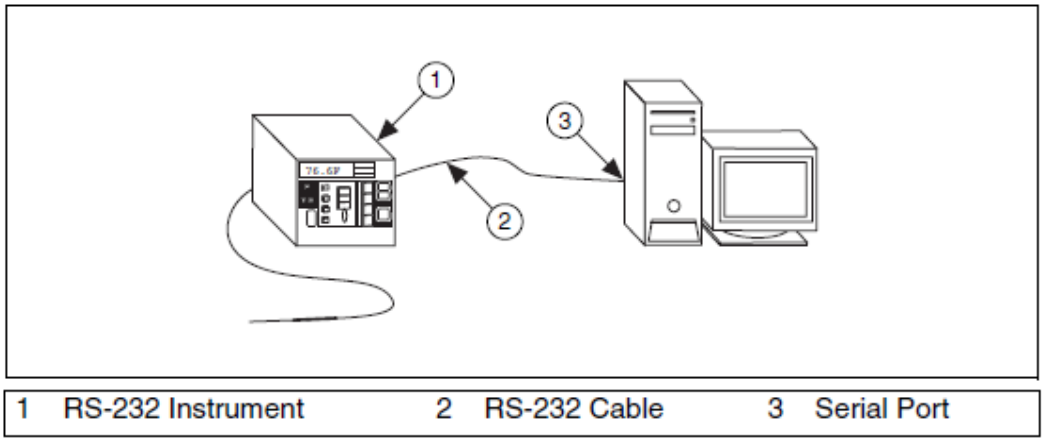


Рисунок 1.4. Пример подключения прибора с последовательным портом

Для обмена данными через последовательный порт нужно задать четыре параметра: скорость передачи данных, количество бит данных, используемых для кодирования символа, способ возможного контроля по биту четности и количество стоп-бит. Каждый передаваемый символ упаковывается во фрейм символа, начинающийся с одного старт-бита.

Скорость передачи данных определяет, как быстро последовательные данные передаются из/в прибор.

Биты данных передаются в инверсной логике и младшим значащим битом вперед. Для интерпретации битов данных, содержащихся во фрейме символа, нужно читать биты справа налево и принимать за "1" низкий уровень напряжения, а за "0" – высокий.

Необязательный бит контроля четности следует за битами данных во фрейме символа также в инверсной логике и используется для контроля ошибок. Предварительно задается режим контроля – по четности или нечетности. Если выбран режим контроля по нечетности, контрольный бит устанавливается так, чтобы суммарное число единиц бит данных и контрольного бита было нечетным.

Заключительная часть фрейма символа состоит из 1, 1.5 или 2 стоп-бит, также представленных в отрицательной логике. Если далее передача символов не производится, на линии удерживается низкий уровень (MARK). Передача следующего символа (фрейма) начинается со старт-бита (SPACE) представляемого положительным уровнем напряжения.

Ниже на рисунке показан типичный фрейм символа с кодом буквы m.

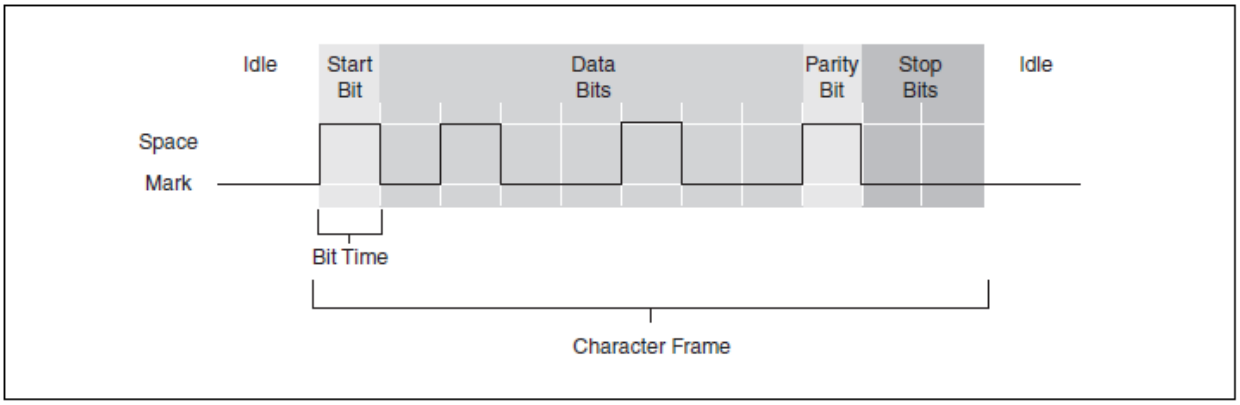


Рисунок 1.5. Фрейм сиувола для буквы m

В RS-232 используются только два уровня напряжения, называемые MARK и SPACE. В такой двухуровневой схеме кодирования скорость передачи данных совпадает с максимальным количеством бит информации, включая биты контроля, которые передается за секунду.

MARK представляется отрицательным уровнем напряжения, а SPACE – положительным. На предыдущем рисунке показано, как выглядит идеализированный сигнал на экране осциллографа. Ниже приведена таблица истинности для RS-232:

Signal > +3 V = 0

Signal < +- V = 1

Обычно уровни сигнала принимают значения + 12 V и – 12 V. Зона нечувствительности между +3 V и -3 V предназначена для отсечки помех в линии.

Старт-бит сигнализирует о начале каждого фрейма символа. Он передается переключением из отрицательного (MARK) уровня в положительный (SPACE). Длительность старт-бита обратно пропорциональна скорости передачи данных. Если прибор передает данные со скоростью 9600 бод, длительность старт-бита и каждого последующего бита приблизительно равна 0,104 мс. Полностью фрейм символа из 11 бит передается за 1146 мс.

Интерпретация переданных бит данных дает 1101101 (в двоичном коде) или 6D (в шестнадцатеричном). Преобразовав по таблице символов ASCII, получим букву m.

В приведенном примере передачи используется контроль по нечетности числа бит. Поскольку количество единиц в битах данных равно 5, и т.к. это нечетное число, бит контроля четности установлен в 0.

Скорость передачи данных

Вы можете вычислить максимальную скорость передачи данных в количестве передаваемых символов в секунду, разделив скорость передачи (baud rate) на количество бит во фрейме символа.

В предыдущем примере фрейм символа состоит из 11 бит. При скорости передачи (baud rate) в 9600 бод получим 9600/11 = 872 символа в секунду. Обращаем внимание, что полученный результат характеризует максимальную скорость передачи символов. По различным причинам оборудование может не обеспечивать вычисленное значение скорости.

Стандарты обмена данными через последовательный порт

Ниже приведены примеры наиболее распространенных рекомендуемых стандартов, используемых для обмена информацией через последовательный порт.

- RS-232 (ANSI/EIA-232 Standard) применяется во многих задачах, например, для подключения мыши, принтера или модема, а также во многих промышленных приборах. Усовершенствованные модификации кабелей и драйверов передатчиков часто позволяют в конкретных приложениях улучшить характеристики RS-232 по длине линий связи и скорости передачи. Возможные схемы подключения RS-232 ограничены только схемой "точка-точка" между последовательными портами компьютера и устройствами.
- RS-422 (ANSI/EIA-422 Standard) – использует дифференциальный электрический сигнал в отличие от несимметричного (single ended) сигнала в RS-232, передаваемого относительно общей цепи (заземления). Дифференциальная схема подключения использует по две линии для передачи и для приема, что в сравнении RS-232 позволяет повысить помехозащищенность и увеличить длину линии связи.
- RS-485 (ANSI/EIA-485 Standard) – это версия RS-422, допускающая подключение к одному порту до 32-х устройств и определяющая требования к электрическим характеристикам для гарантии получения адекватных уровней напряжения при максимальной нагрузке. Используя многоточечное подключение, вы можете создавать сети устройств, подключенные к одному последовательному порту RS-485. Высокая помехозащищенность и многоточечная схема подключения делают RS-485 привлекательным в индустриальных приложениях, в которых требуется несколько распределенных устройств объединять в сеть с персональным компьютером или другим контроллером для сбора данных и других операций.

F. Программное обеспечение для управления измерительными приборами

Архитектура программного обеспечения LabVIEW для управления измерительными приборами похожа на архитектуру программного обеспечения DAQ. Интерфейсы приборов, такие, как GPIB, содержат набор драйверов. Конфигурирование интерфейсов производится в MAX.

Примечание: Драйверы GPIB находятся на инсталляционном диске LabVIEW, а большинство драйверов GPIB можно загрузить по ссылке `ni.com/support/gpib/versions.htm`. Устанавливайте всегда новые версии драйверов, если только в релизе не указано иное.

MAX (Windows: GPIB)

Для конфигурирования и тестирования интерфейса GPIB используется утилита MAX, которая взаимодействует с различными диагностическими и конфигурационными средствами, инсталлированными с драйверами, а также с системным реестром Windows и менеджером оборудования. Программное обеспечение уровня драйверов представлено в форме DLL и содержит все необходимые функции, которые непосредственно связывают с интерфейсом GPIB. Функции и VI Instrument I/O напрямую вызывают программный драйвер.

Открывается MAX двойным щелчком по иконке на рабочем столе или из меню LabVIEW выбором пункта **Tools»Measurement & Automation Explorer**. В следующем примере показан интерфейс GPIB в MAX после щелчка по кнопке **Scan For Instruments** в линейке инструментов.

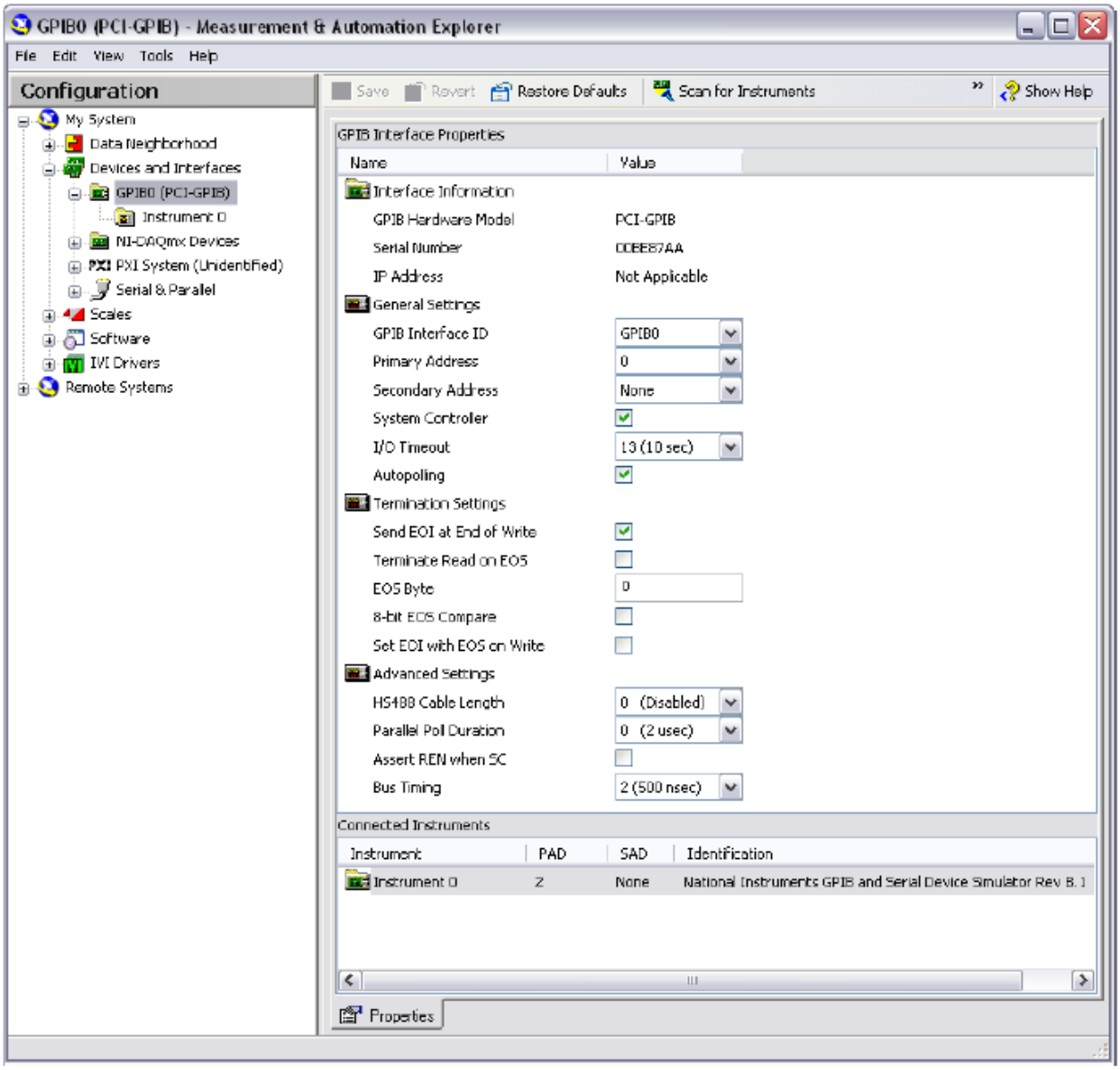


Рисунок 1.6. Интерфейс GPIB в Measurement & Automation Explorer

Объекты, приведенные в списке MAX, конфигурируют двойным щелчком мыши по каждому элементу и выбором нужных вариантов в контекстном меню. Выполнив упражнение, вы научитесь использовать MAX для конфигурирования измерительного прибора с GPIB и обмена информацией с ним.

G. Курсовой проект

Как и во всем учебном курсе, курсовой проект иллюстрирует основные принципы посредством демонстрационных примеров и практических упражнений. Проект должен удовлетворять следующим требованиям:

1. Получать результат измерения температуры 2 раза в секунду
2. Анализировать значение температуры с целью определить, не слишком ли она низка или высока

3. Предупреждать пользователя, если есть опасность теплового удара или замерзания
4. Отображать данные
5. Регистрировать данные, если появляется предупреждение
6. Процесс повторяется, пока пользователь не остановит выполнение программы

В курсовом проекте должны быть предусмотрены следующие входы и выходы:

Входы

- Текущее значение температуры (T)
- Верхний предел температуры (X)
- Нижний предел температуры (Y)
- Стоп

Выходы

- Уровни предупреждения: Предупреждение о тепловом ударе, Нет предупреждения, Предупреждение о замерзании
- Индикатор текущего значения температуры
- Файл регистрации данных

Диаграмма состояний-переходов, показанная на рис. 1-7, выбрана для того, чтобы все студенты могли придерживаться одного и того же набора команд. Эта диаграмма выбрана, поскольку с ее помощью можно решить задачу, а кроме того, она содержит компоненты, которые могут быть эффективно использованы для демонстрации концепции курса. Однако, это может быть не лучшим решением проблемы.

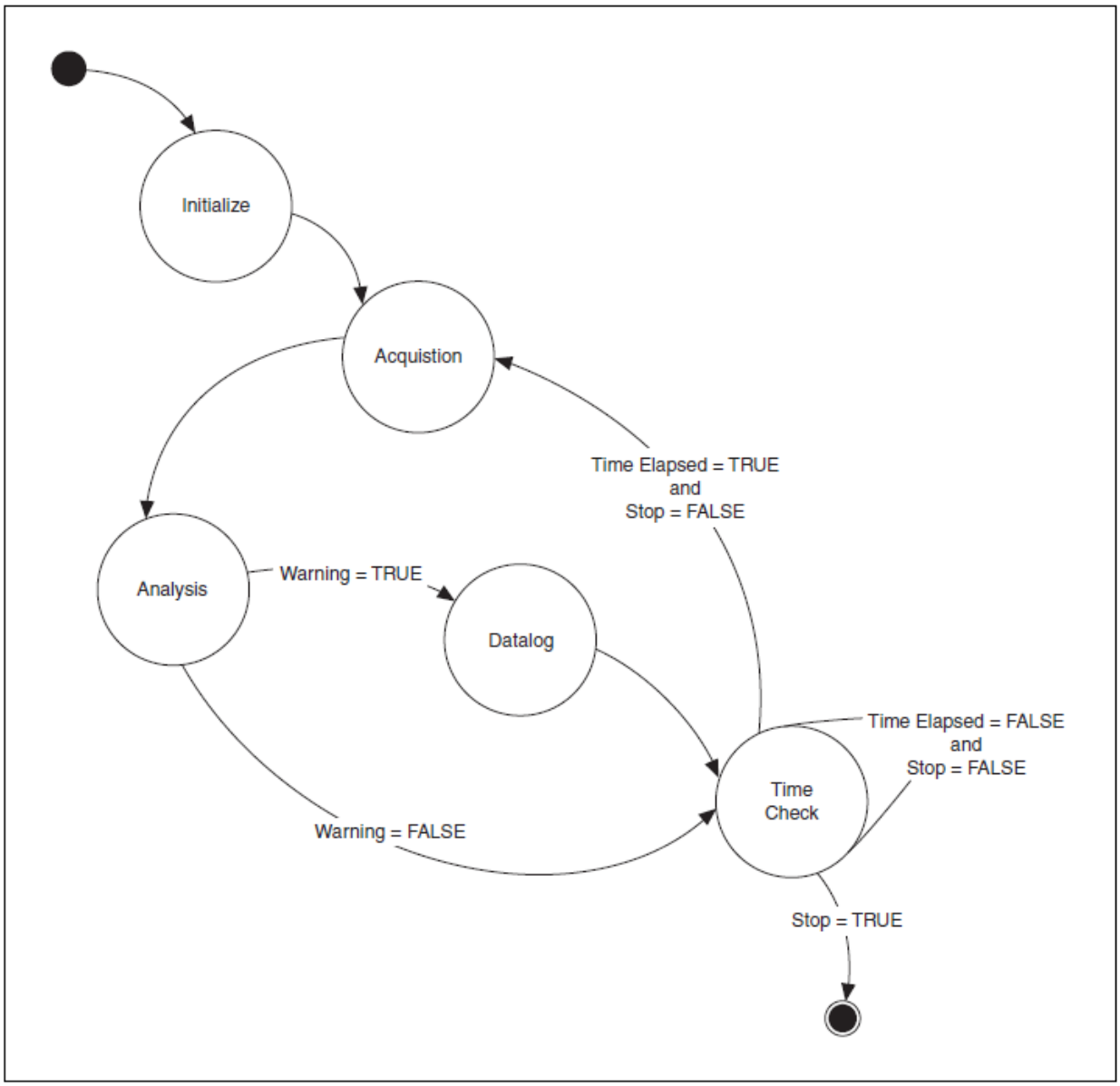


Рисунок 1.7. Диаграмма состояний-переходов для проекта

На рис. 1.8 показан пример альтернативной диаграммы состояний-переходов. Эта диаграмма решает задачу тоже весьма эффективно. Одно из главных отличий этих двух диаграмм состоит в том, насколько они позволяют расширить функциональность в будущем. Диаграмму состояний-переходов на рис. 1-7 вы можете модифицировать для включения состояний предупреждений по другим физическим величинам, например, скорости ветра, давления и влажности. Диаграмму состояний-переходов на рис. 1-8 вы можете дополнить другими уровнями предупреждений по температуре. Предполагаемые в будущем изменения программы влияют на выбор диаграммы.

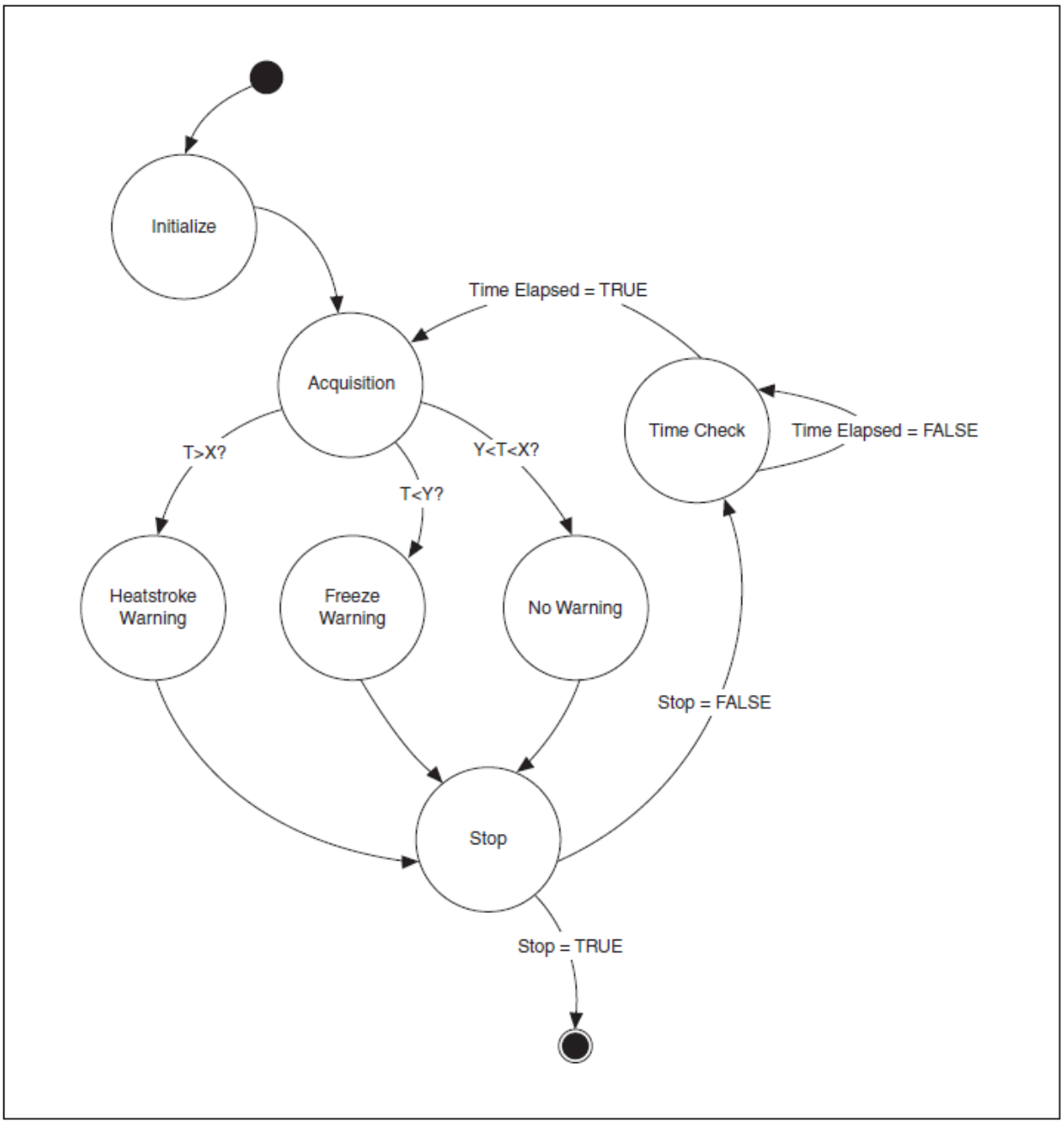


Рисунок 1.8. Альтернативная диаграмма состояний-переходов для проекта

Самопроверка: короткий тест

1. Measurement & Automation Explorer (MAX) можно использовать для изучения, конфигурирования и тестирования DAQ-устройств и приборов с интерфейсом GPIB.
   a. Да
   b. Нет
2. Какими из следующих преимуществ обладают системы на основе программно управляемых измерительных приборов?
   a. Автоматизация процесса измерений
   b. Экономия времени
   c. Одна платформа для многих задач
   d. Ограничение только одним типом прибора

Самопроверка: ответы

1. Measurement & Automation Explorer (MAX) можно использовать для изучения, конфигурирования и тестирования DAQ-устройств и приборов с GPIB интерфейсом.

   a. Да

   b. Нет

2. Какими из следующих преимуществ обладают системы на основе программно управляемых измерительных приборов?

   a. Автоматизация процесса измерений

   b. Экономия времени

   c. Одна платформа для многих задач

   d. Ограничение только одним типом прибора

Заметки

2. Ориентация в LabVIEW

На этом занятии объясняется, как ориентироваться в среде LabVIEW – как использовать меню, панели инструментов, палитры, инструменты, справки и стандартные диалоговые окна LabVIEW. Вы также научитесь запускать VI и получите общее представление о лицевой панели и блок-диаграмме. В конце урока вам предстоит создать простой VI, который выполняет сбор, обработку и представление данных.

План занятия

A. Виртуальные приборы (VI)
B. Состав VI
C. Начинаем проектирование VI
D. Project Explorer
E. Лицевая панель
F. Блок-диаграмма
G. Поиск органов управления, VI и функций
H. Выбор инструмента
I. Поток данных
J. Разработка простого VI

A. Виртуальные приборы (VI)

Программы, спроектированные в LabVIEW, называются виртуальными измерительными приборами, или VI, поскольку их внешний вид и функционирование имитируют реальные измерительные приборы, такие, как осциллографы и мультиметры. LabVIEW содержит полный набор инструментов для сбора, обработки, отображения и регистрации данных, а также средства, которые помогают отладить разрабатываемую программу.

B. Состав VI

VI, созданные в LabVIEW, состоят из трех основных компонентов: окна лицевой панели, блок-диаграммы и иконки/панели подключения.

Окно лицевой панели

Окно лицевой панели является пользовательским интерфейсом VI. На рисунке 2-1 приведен пример окна лицевой панели. Вы создаете окно лицевой панели вместе с органами управления и индикаторами, которые являются соответственно входными и выходными интерактивными терминалами VI.

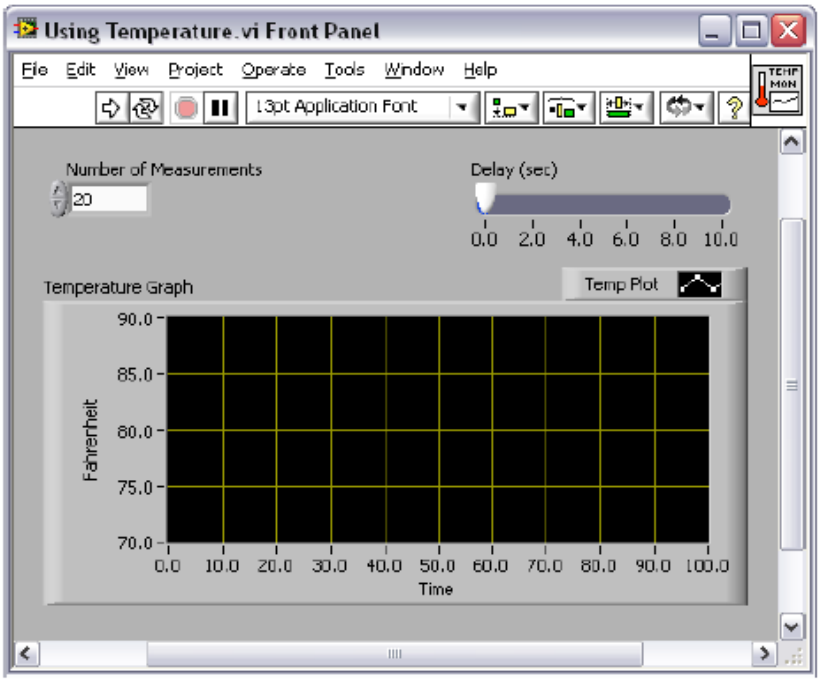


Рисунок 2-1. Лицевая панель VI

Окно блок-диаграммы

После создания лицевой панели в блок-диаграмму добавляется программный код с использованием функций, представленных в графической форме, которые управляют объектами на фронтальной панели. На рисунке 2-2 приведен пример окна блок-диаграммы. В окне блок-диаграммы содержится исходный код программы. Объекты, размещенные на лицевой панели, появляются на блок-диаграмме в виде терминалов.

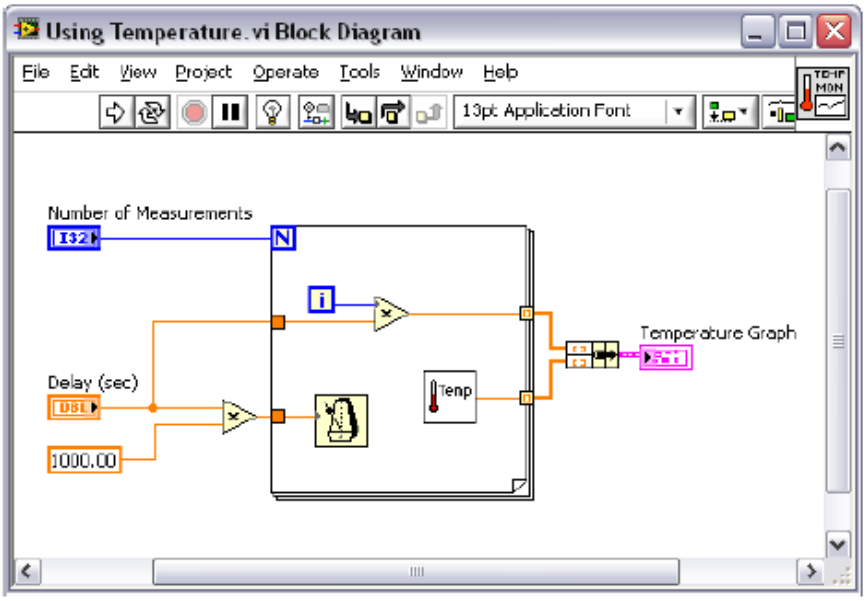


Рисунок 2-2. Блок-диаграмма

Иконка и панель подключения

Иконка и панель подключения позволяют использовать и просматривать VI внутри другого VI. VI, используемый внутри другого VI, называется subVI, и является аналогом функции в текстовом языке программирования. Чтобы использовать VI в качестве subVI, нужно его снабдить иконкой и панелью подключения.

Иконка отображается в верхнем правом углу окна лицевой панели и окна блок-диаграммы каждого VI и является графическим представлением VI. Пример иконки "по умолчанию" показан слева. В иконке может содержаться как текст, так и изображения. Если вы используете VI в качестве subVI, то он обозначается на блок-диаграмме VI иконкой. Иконка "по умолчанию" содержит число, которое показывает, сколько новых VI вы открыли после того, как запустили LabVIEW.

Чтобы использовать VI в качестве subVI, необходимо создать панель подключения, показанную слева. Панель подключения является набором терминалов иконки, которые соответствуют органам управления и индикаторам данного VI, это похоже на список параметров функции, вызываемой в текстовых языках программирования. Доступ к панели подключения осуществляется щелчком правой кнопки мыши по иконке в верхнем правом углу окна лицевой панели. В окне блок-диаграммы панель подключения открыть нельзя.

C. Начинаем проектировать VI

После запуска LabVIEW на экране открывается окно **Getting Started**. Это окно следует использовать для создания новых VI и проектов, выбора

каких-либо из последних открытых файлов LabVIEW, поиска примеров и справочной информации в *LabVIEW Help*. Кроме того, из стартового окна обеспечивается доступ к информации и ресурсам, которые помогут вам изучить LabVIEW, например, к специальным руководствам, разделам справочной системы, а также Интернет-ресурсам по адресу `ni.com/manuals`.

Когда вы открываете существующий файл или создаете новый файл, окно **Getting Started** закрывается. Это окно можно вывести на экран, выбрав команду меню **View»Getting Started Window**.

Рисунок 2-3. Окно LabVIEW Getting Started

Вы можете сконфигурировать LabVIEW так, чтобы при запуске вместо вывода на экран этого окна открывался новый пустой VI. Выберите команду меню **Tools»Options** из списка **Category** выберите пункт **Environment** и установите флажок **Skip Getting Started window on launch**.

Примечание: Содержание окна **Getting Started** зависит от инсталлированной версии LabVIEW и инсталлированных инструментальных средств.

Создание или открытие VI или проекта

Вы можете начать работать в LabVIEW с создания нового VI или проекта, открытия существующего VI или проекта и их изменения, или открытия шаблона, с которого начнете разрабатывать ваш новый VI или проект.

Создание новых проектов и VI

Чтобы открыть новый проект из окна **Getting Started**, выберите пункт **Empty Project** в списке **New**. В этом случае открывается новый проект без имени, в который можно добавить файлы и затем сохранить его.

Чтобы открыть новый пустой VI, который не связан с проектом, выберите пункт **Blank VI** в списке **New** в окне **Getting Started**.

Создание VI на основе шаблона

Чтобы вывести на экран диалоговое окно **New**, в котором приведен список встроенных шаблонов VI, выберите команду меню **File»New**. Открыть диалоговое окно **New** можно также щелчком мыши по ссылке **New** в окне **Getting Started**.

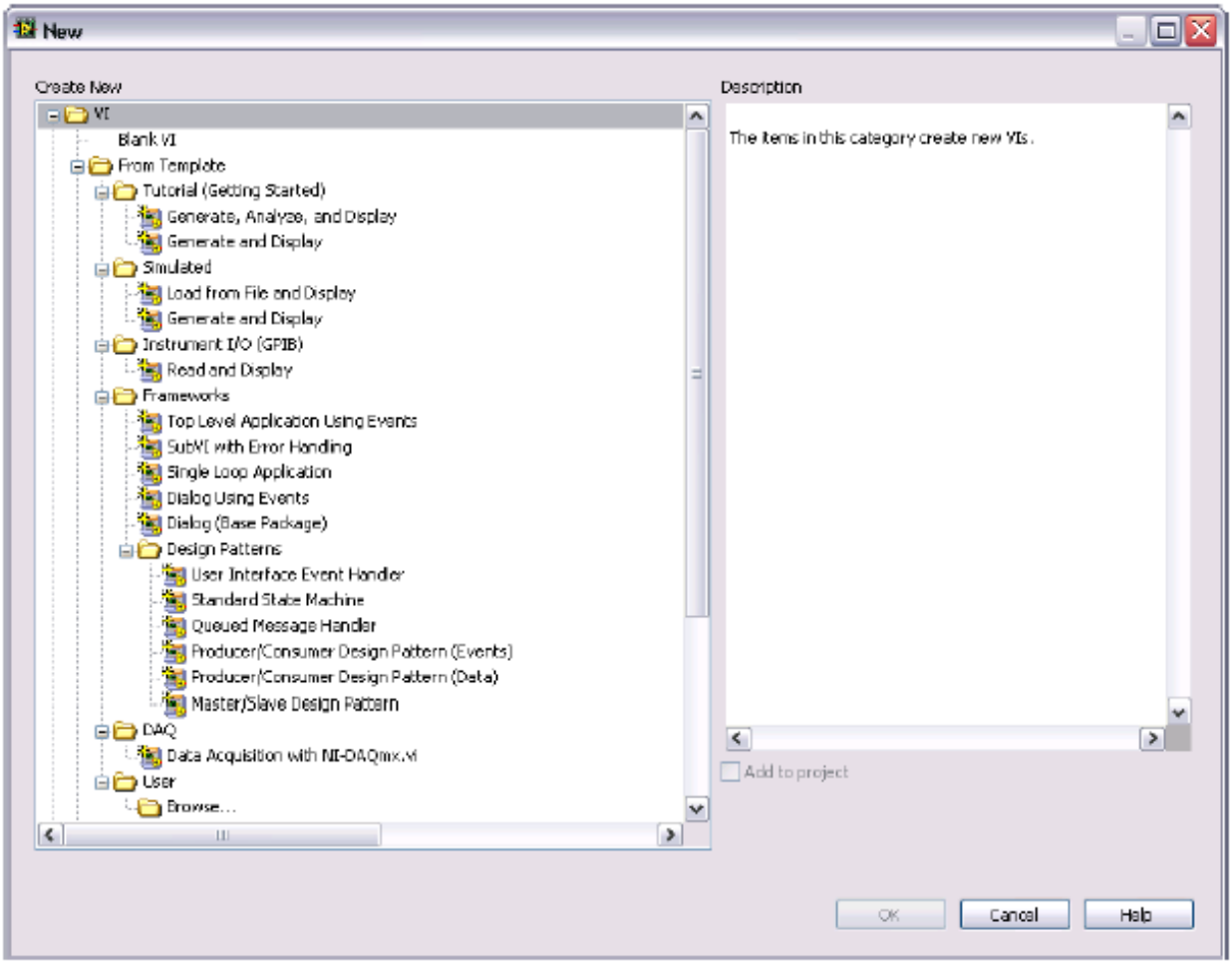


Рисунок 2-4. Новое диалоговое окно

Открытие существующего VI

Выберите пункт **Browse** в списке **Open** в окне **Getting Started**, чтобы выбрать и открыть какой-нибудь существующий VI.

Подсказка: Редактируемые в рамках данного курса VI расположены в папке `<Exercises>\LabVIEW Core 1`.

В процессе загрузки VI может появиться диалоговое окно состояния, похожее на изображенное ниже в качестве примера.

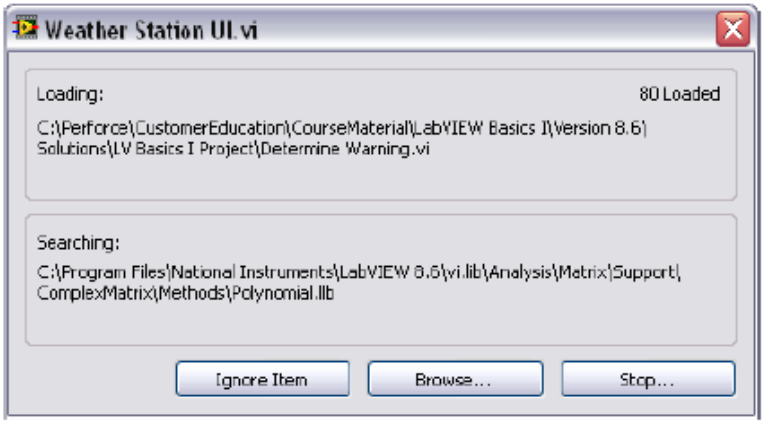


Рисунок 2-5. Диалоговое окно статуса загрузки нового VI

В разделе **Loading** пролистываются subVI, входящие в состав VI, по мере загрузки их в память. Здесь же показывается количество загруженных subVI. В любой момент вы можете отменить загрузку щелчком мыши по кнопке **Stop**.

Если среда проектирования LabVIEW сразу не может обнаружить subVI, то она начинает искать его по всем папкам, указанным в качестве путей поиска. Маршруты поиска VI можно отредактировать, выбрав команду меню **Tools»Options** и пункт **Paths** в списке **Category**.

Вы можете заставить LabVIEW игнорировать subVI щелчком мыши по кнопке **Ignore Item** или щелкнуть мышью по кнопке **Browse** для поиска недостающего subVI.

Сохранение VI

Чтобы сохранить новый VI, выберите команду меню **File»Save**. Если вы уже сохранили ваш VI, для доступа к диалоговому окну **Save As** выберите команду меню **File»Save As**. Из диалогового окна **Save As** можно создать копию VI или удалить исходный VI и заменить его новым.

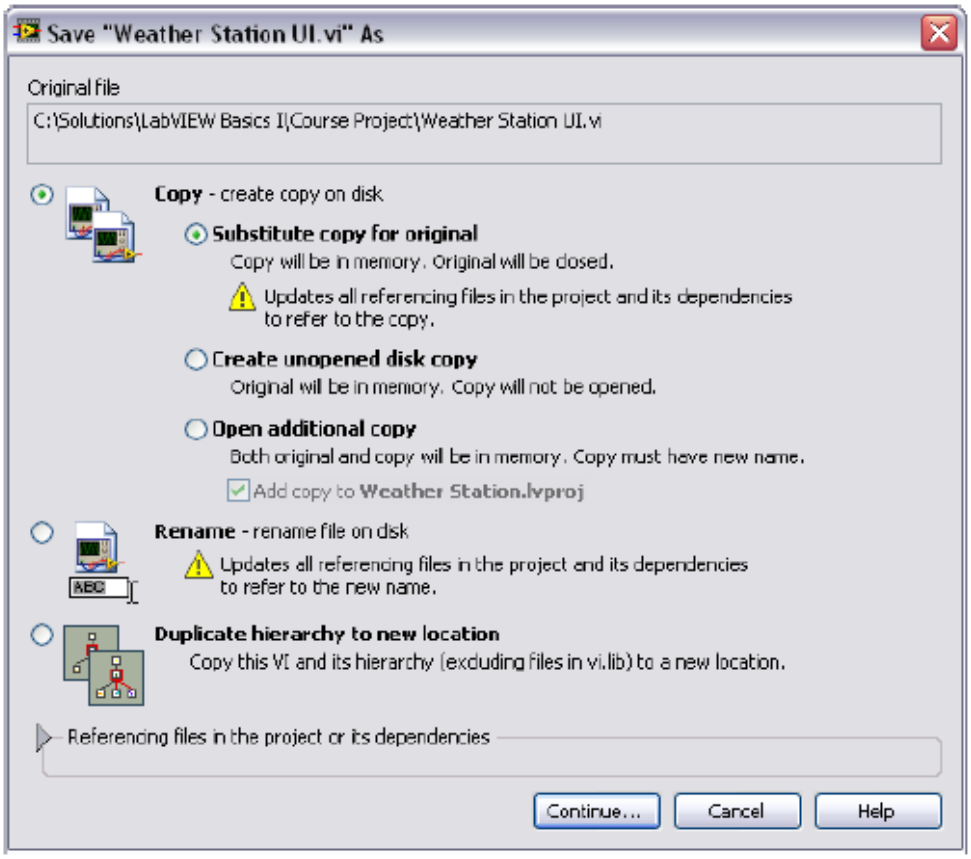


Рисунок 2-6. Диалоговое окно Save As

Примечание: За подробной информацией по каждой опции диалогового окна **Save As** обратитесь к разделу **Save As Dialog Box** справки *LabVIEW Help*.

D. Project Explorer

Проекты предназначены для группирования файлов LabVIEW, а также файлов других типов, создания спецификации построителя приложений, развертывания или загрузки файлов в целевые устройства. Когда вы сохраняете проект, LabVIEW создает файл проекта (`.lvproj`), который включает в себя ссылки на файлы проекта, информацию о конфигурации, сборке приложения, развертывании и т.д.

Проекты необходимы для построения приложений и совместно используемых библиотек. Из проектов работают с различными целевыми устройствами – реального времени (RT), программируемыми логическими интегральными схемами (FPGA) или карманными компьютерами (PDA). Дополнительная информация по применению проектов совместно с модулями LabVIEW Real-Time, FPGA и PDA приведена в соответствующей документации на модули.

Окно Project Explorer

Окно **Project Explorer** служит для создания и редактирования проектов LabVIEW. Для отображения этого окна выберите команду меню **File»New Project** или **Project»New Project** или пункт **Empty Project** в диалоговом окне **New**.

Окно **Project Explorer** состоит из двух страниц: **Items** и **Files.** На странице **Items** отображаются элементы проекта в форме древовидной структуры проекта. На странице **Files** отображаются элементы проекта, для которых есть соответствующий файл на диске. На этой странице можно систематизировать имена файлов и папки. Операции над проектом на странице **Files** отражают и обновляют содержимое на диске. Вы можете переключаться с одной страницы на другую, щелкнув правой кнопки мыши по папке или элементу под названием целевого устройства и выбрав из контекстного меню команду **Show in Items View** или **Show in Files View.**

По умолчанию окно **Project Explorer** состоит из следующих элементов:

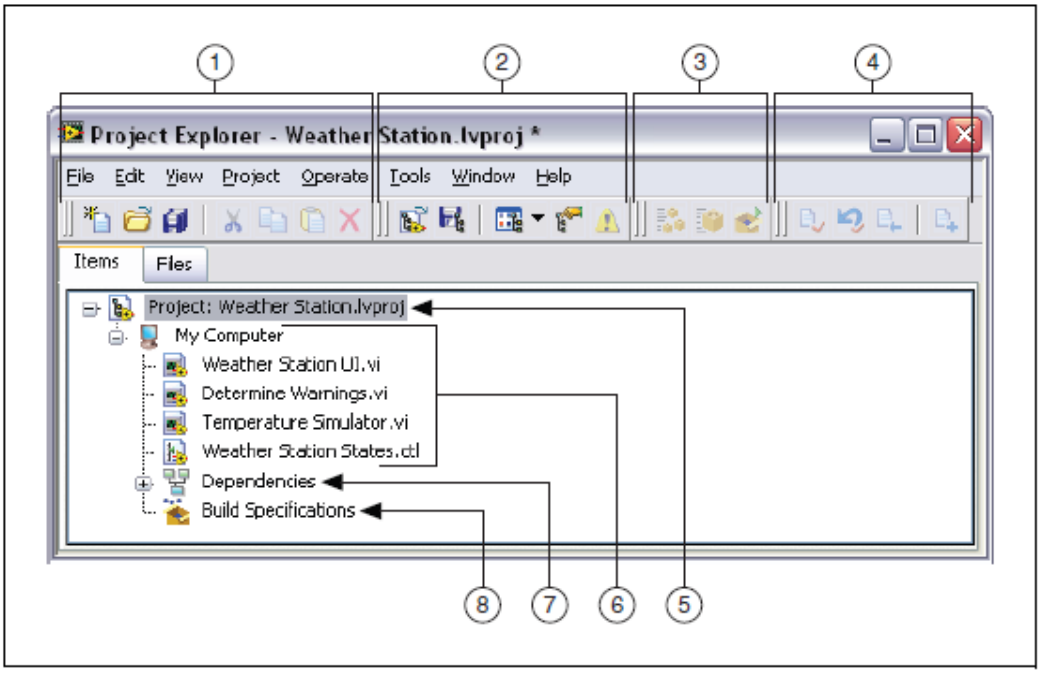
- **Project root** — содержит все остальные элементы окна **Project Explorer.** Данная метка корня проекта включает в себя имя файла проекта.
- **My Computer** — представляет локальный компьютер в качестве целевого устройства проекта.
- **Dependencies** — включает в себя элементы, которые требуются VI, относящимся к целевому устройству.
- **Build specifications** — включает в себя конфигурации компоновки для распространения исходных файлов и других типов компоновок, доступных в комплектах инструментальных средств и модулях LabVIEW. Если у вас инсталлирована система LabVIEW Professional Development System или Application Builder, вы можете использовать **Build Specifications** для конфигурирования создаваемых автономных приложений, совместно используемых библиотек, инсталляторов и zip-файлов.

Подсказка: *Целевым* является любое устройство, на котором может выполняться VI.

Когда вы добавляете целевое устройство в проект, LabVIEW создает дополнительный элемент для представления этого устройства в окне **Project Explorer**. Для каждого целевого устройства создаются также разделы **Dependencies** и **Build Specifications.** Вы можете добавлять файлы для каждого целевого устройства.

Панели инструментов для проекта

Для выполнения операций в проекте LabVIEW следует использовать кнопки **Standard**, **Project**, **Build** и **Source Control** на панели инструментов. Панели инструментов доступны в верхней части окна **Project Explorer** (рисунок 2-7). Чтобы увидеть все кнопки, возможно, потребуется раскрыть окно **Project Explorer**.



1	Standard Toolbar	5	Project Root
2	Project Toolbar	6	Target
3	Build Toolbar	7	Dependencies
4	Source Control Toolbar	8	Build Specifications

Рисунок 2-7. Окно Project Explorer

Совет: Панель инструментов **Source Control** доступна только в том случае, если у вас есть средства управления исходным программным кодом, сконфигурированные в LabVIEW .

Вы можете показывать или скрывать панели инструменты, выбрав команду **View»Toolbars** и панели инструментов, которые хотите показать или скрыть. Можно также щелкнуть правой кнопкой мыши в свободной области на панели инструментов и выбрать панели инструментов, которые вы хотите показать или скрыть.

Создание LabVIEW-проекта

Чтобы создать проект, выполните следующие действия:

1. Выберите команду меню **File»New Project** для вывода на экран окна **Project Explorer**. Для этой же цели можно также выбрать пункт **Project»Empty Project** в диалоговом окне **New**.
2. Добавьте элементы для целевого устройства, которые вы хотите включить в проект.
3. Для сохранения проекта выберите команду меню **File»Save Project**.

Добавление существующих файлов в проект

В проект можно добавлять существующие файлы. Чтобы добавлять в проект LabVIEW файлы VI или текстовые файлы, используйте пункт **My Computer** (или пункт, представляющий другое целевое устройство) в окне **Project Explorer**.

В проект можно добавлять элементы следующими способами:

- Щелкните правой кнопкой мыши по ветви проекта **My Computer** и выберите из контекстного меню команду **Add»File**, чтобы добавить файл. Для этой же цели можно также выбрать команду **Project»Add to Project»File** из меню Project Explorer.
- Чтобы добавить автоматически обновляющуюся папку, щелкните правой кнопкой мыши по ветви проекта, представляющей целевое устройство, и из контекстного меню выберите команду **Add»Folder (Auto-populating)**. Для этой же цели можно выбрать команду меню **Project»Add To Project»Add Folder (Auto-populating)**. LabVIEW непрерывно контролирует и обновляет папку в соответствии с изменениями, вносимыми в проект и на диск.
- Чтобы добавить виртуальную папку, щелкните правой кнопкой мыши по ветки, представляющей целевое устройство, и выберите из контекстного меню команду **Add»Folder (Snapshot)**. Для этой же цели можно выбрать команду меню **Project»Add To Project»Add Folder (Snapshot)**. Когда вы выберете папку на диске, LabVIEW создает в проекте новую виртуальную папку с тем же именем, что и папка на диске. LabVIEW также создает элементы проекта, которые отображают содержимое всей папки, включая файлы и содержимое вложенных папок. При выборе папки на диске добавляется все содержимое папки, включая файлы и содержимое вложенных папок.

Примечание: LabVIEW автоматически не обновляет виртуальную папку в проекте при внесении изменений в папке на диске.

- Чтобы добавить новый пустой VI, щелкните правой кнопкой мыши по элементу, соответствующему целевому устройству, и выберите из контекстного меню команду **New»VI**. Для этой же цели можно также выбрать команду меню **File»New VI** или **Project»Add To Project»New VI**.
- Выберите в правом верхнем углу лицевой панели или блок-диаграммы иконку VI и перетащите ее на элемент, соответствующий целевому устройству.
- Выберите элемент или папку из файловой системы на вашем компьютере и перетащите его/ее на элемент, соответствующий целевому устройству.

Удаление элементов из проекта

Вы можете удалить элементы из окна **Project Explorer** следующими способами:

- Щелкните правой кнопкой мыши по элементу, который нужно удалить, и выберите из контекстного меню команду **Remove from Project.**
- Выделите элемент, который хотите удалить, и нажмите на клавишу <Delete>
- Выделите элемент, который нужно удалить, и щелкните мышью по кнопке **Remove From Project** на панели инструментов Standard

Примечание: При удалении элемента из проекта этот элемент не удаляется с диска.

Организация элементов в проекте

Окно **Project Explorer** включает в себя две страницы: **Items** и **Files**. На странице **Items** отображаются элементы проекта в виде древовидной структуры. На странице **Files** отображаются элементы проекта, для которых есть соответствующие файлы на диске. На этой странице вы можете систематизировать имена файлов и папок. Операции над проектом на странице **Files** одновременно отражают и обновляют содержимое диска. Вы можете переключаться с одной страницы на другую, щелкнув правой кнопкой мыши по папке или элементу, соответствующему целевому устройству, и выбрав из контекстного меню команду **Show in Items Menu** или **Show in Files View.**

Для организации элементов в окне **Project Explorer** следует использовать папки. В проект LabVIEW можно добавлять папки двух типов: виртуальные папки и автоматически обновляющиеся папки. В виртуальных папках систематизируются элементы проекта. Чтобы создать новую виртуальную папку, щелкните правой кнопкой мыши по элементу, соответствующему какому-либо целевому устройству в Project Explorer, и выберите из контекстного меню команду **New»Virtual Folder.** Автоматически обновляющиеся папки обновляются в реальном времени, отражая содержимое папок на диске. Чтобы просмотреть элементы проекта в таком виде, как они появляются на диске, добавьте в проект автоматически обновляющуюся папку.

Автоматически обновляющиеся папки видимы только на странице **Items** в окне **Project Explorer.** Вы можете просмотреть содержимое автоматически обновляющейся папки, однако в них вы не можете выполнять такие дисковые операции как переименование, изменение структуры и удаление элементов проекта. Для выполнения дисковых операций над элементами в автоматически обновляющейся папке следует пользоваться страницей **Files** в окне **Project Explorer.** На странице **Files** отображается размещение папок на диске. Операции над проектом на странице **Files** одновременно отражают и обновляют содержимое папки на диске. Аналогично LabVIEW автоматически обновляет автоматически обновляемую папку в проекте, если вы вносите изменения в соответствующую папку на диске за пределами LabVIEW.

Существует возможность упорядочивать элементы в папке. Чтобы упорядочить элементы в алфавитном порядке, щелкните правой кнопкой по

папке и выберите из контекстного меню команду **Arrange By»Name**. Чтобы упорядочить элементы по типу файла, щелкните правой кнопкой по папке и выберите команду **Arrange By»Type**.

Просмотр файлов в проекте

Когда вы добавляете какой-либо файл в проект, LabVIEW включает в проект ссылку на этот файл на диске. Чтобы открыть файл в редакторе по умолчанию, щелкните по нему правой кнопкой мыши в окне **Project Explorer** и выберите из контекстного меню команду **Open**.

Местонахождение сохраненных на диске файлов, на которые ссылается проект, можно узнать, щелкнув правой кнопкой мыши по проекту и выбрав из контекстного меню команду **View»Full Paths**.

Местонахождение файлов, на которые ссылается проект, на диске и в окне **Project Explorer** можно узнать, воспользовавшись диалоговым окном **Project File Information**. Чтобы открыть окно **Project File Information**, выберите команду **Project»File Information**. Для этой же цели можно также щелкнуть правой кнопкой мыши по проекту и выбрать из контекстного меню команду **View»File Information**.

Сохранение проекта

Вы можете сохранить LabVIEW проект следующими способами:

- Выбрать команду меню **File»Save Project**.
- Выбрать команду меню **Project»Save Project**.
- Щелкнуть правой кнопкой мыши по проекту и выбрать из контекстного меню команду **Save**.
- Щелкнуть мышью по кнопке **Save Project** на панели инструментов **Project**.

Вы должны сохранить новые, несохраненные файлы проекта прежде, чем у вас появится возможность сохранить проект. Когда вы сохраняете проект, LabVIEW не сохраняет зависимые элементы (dependencies), как части файла проекта.

Примечание: Перед тем, как вносить в проект крупные изменения, следует сделать его резервную копию.

E. Лицевая панель

Когда вы открываете новый или существующий VI, на экране появляется окно лицевой панели, которая является пользовательским интерфейсом VI. На рисунке 2-8 приведен пример окна лицевой панели.

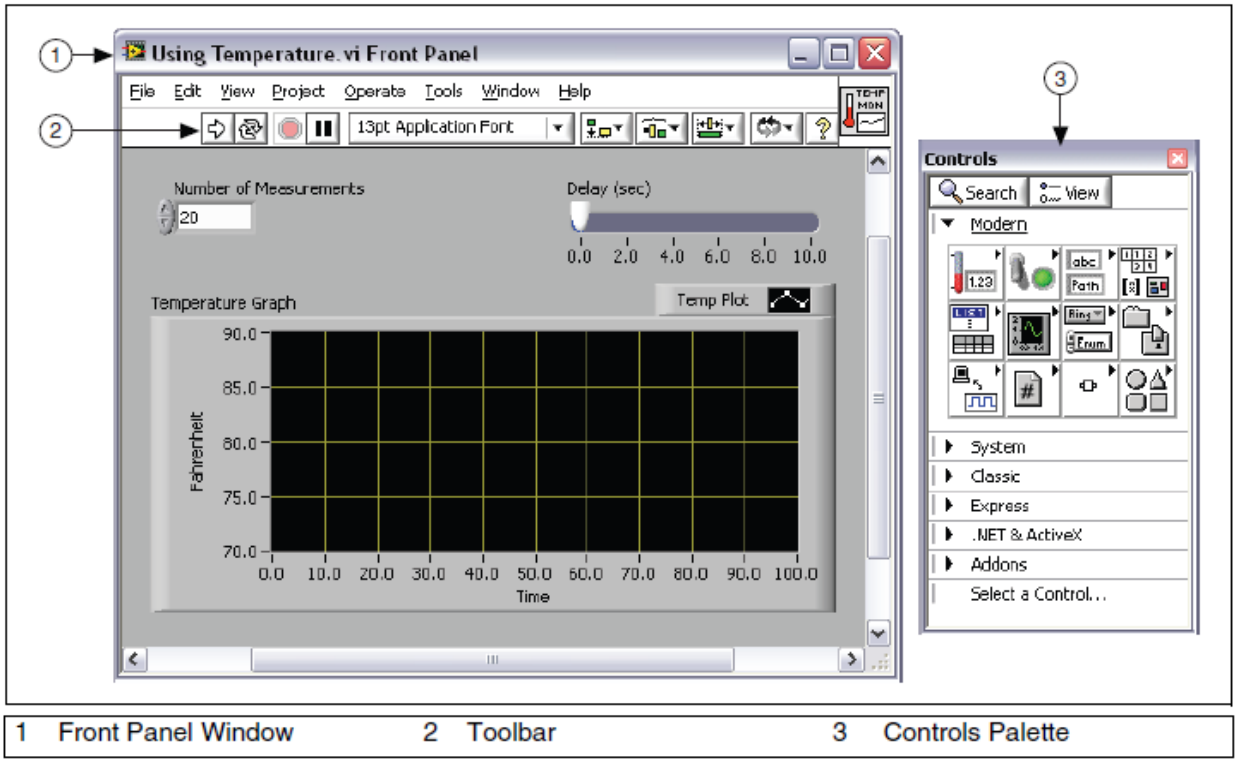


Рисунок 2-8. Пример лицевой панели

Органы управления и индикации

Вы создаете лицевую панель с органами управления и индикаторами, которые являются соответственно интерактивными входными и выходными терминалами VI. К органам управления относятся регуляторы, кнопки, лимбы и другие устройства ввода. К индикаторам относятся графики (графические экраны), светодиодные индикаторы и т.д. Органы управления симулируют входные устройства измерительных приборов и поставляют данные в блок-диаграмму VI. Индикаторы симулируют выходные устройства измерительных приборов, и на них отображаются данные, которые собраны или сгенерированы на блок-диаграмме.

На рисунке 2-8 имеются следующие объекты: два органа управления: **Number of Measurements** и **Delay(sec)**, а также один индикатор: графический экран типа XY, обозначенный как **Temperature Graph.**

Пользователь может изменять значения органов управления **Number of Measurements** и **Delay (sec)**. У него есть также возможность увидеть на индикаторе **Temperature Graph** значения, сгенерированные программным кодом VI, созданным на блок-диаграмме. Об этом вы узнаете в разделе *Числовые органы управления и индикаторы*.

За каждым органом управления или индикатором закреплен некоторый тип данных. Например, горизонтальный движок **Delay(sec)** имеет числовой тип данных. Наиболее часто используемыми типами данных являются числовой, булевский и строковый. О других типах данных вы узнаете на 4-й лекции, *Реализация VI*.

Числовые органы управления и индикации

Числовой тип данных может служить для представления чисел различных типов, например, целочисленного или вещественного. На рисунке 2-9 показаны два обычных числовых объекта: числовой элемент управления и числовой индикатор. Такие объекты, как шкальные индикаторы и лимбы также служат для представления числовых данных.

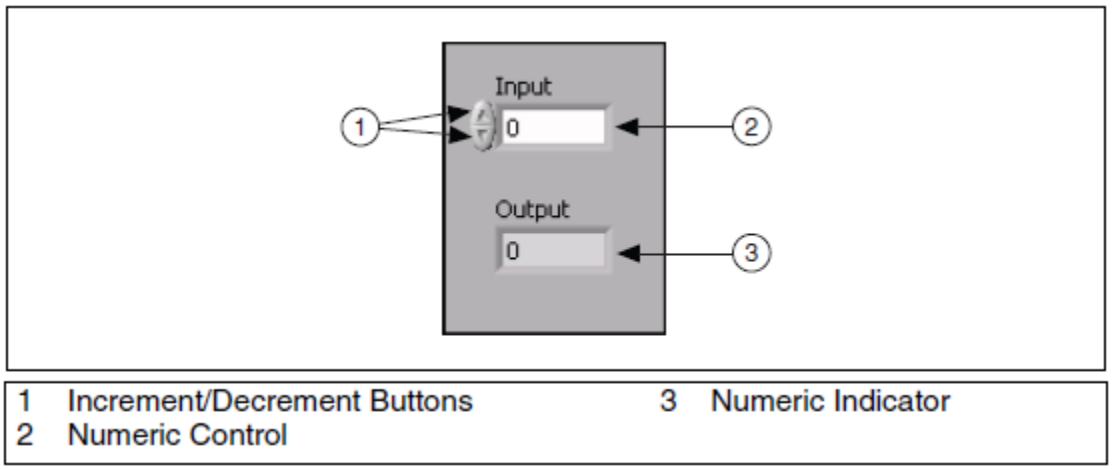


Рисунок 2-9. Числовые элементы управления и индикаторы

Чтобы ввести или изменить значения числового элемента управления, щелкните инструментом Operating по кнопкам инкремента и декремента или щелкните дважды инструментом Labeling либо Operating по числу, а затем введите новое число и нажмите на клавишу <Enter>.

Булевские органы управления и индикации

Булевский тип данных служит для представления данных, которые могут принимать только два значения: TRUE (ON) и FALSE (OFF). Булевские органы управления и индикации применяют для ввода и отображения булевских величин. Булевские объекты симулируют переключатели, кнопки и светодиодные индикаторы. На рисунке 2-10 показаны булевские объекты: вертикальный тумблер и круглый светодиодный индикатор.

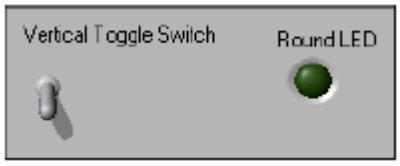


Рисунок 2-10. Булевские элементы управления и индикаторы

Строковые органы управления и индикации

Строковый тип данных представляет собой последовательность ASCII-символов. Строковые органы управления служат для получения такой текстовой информации от пользователя, как пароль или имя пользователя. Используйте строковые индикаторы для того, чтобы вывести пользователю

текстовую информацию. На рисунке 2-11 приведены наиболее распространенные строковые объекты: таблицы и окна для ввода текста.

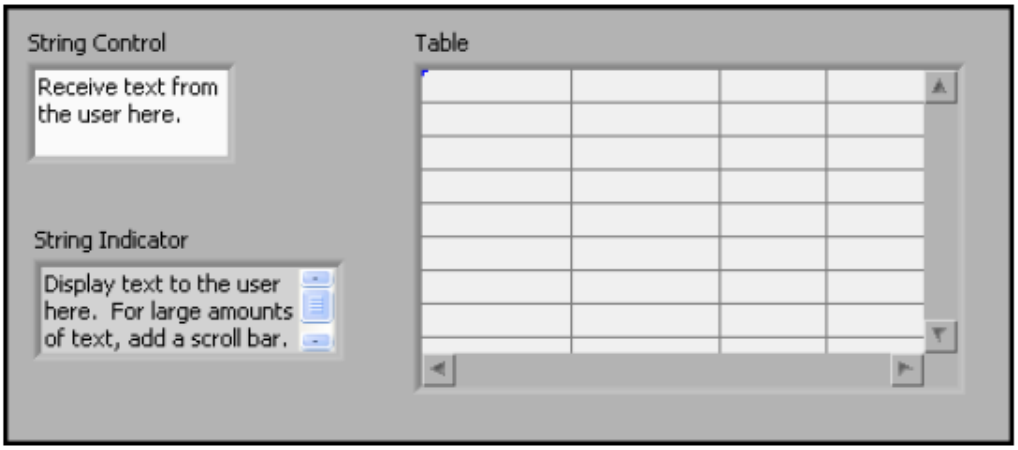


Рисунок 2-11. Строковые элементы управления и индикаторы

Палитра Controls

В палитре **Controls** содержатся органы управления и индикации, которые применяются для создания лицевой панели. Доступ к палитре **Controls** осуществляется из окна лицевой панели с помощью команды меню **View»Controls Palette.** Палитра **Controls** разбита на несколько категорий. При необходимости вы можете сделать видимыми некоторые или все категории. На рисунке 2-12 приведена палитра **Controls**, в которой видны все категории, а категория **Modern** раскрыта. На наших курсах вы будете работать только с объектами категории **Modern.**

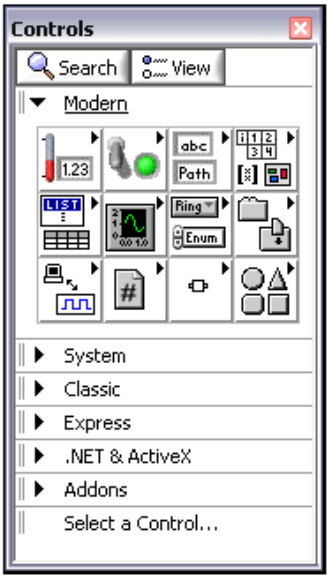


Рисунок 2-12. Палитра Controls

Чтобы сделать видимыми или скрыть категории (суб-палитры), щелкните мышью по кнопке **View** на палитре, затем установите или сбросьте флажки в пункте меню **Always Visible Categories**.

Контекстные меню

Все объекты LabVIEW имеют меню быстрого вызова, также известные как контекстные меню, всплывающие меню или меню по щелчку правой кнопки мыши. В процессе создания VI используйте контекстное меню для изменения внешнего вида или характеристик объектов лицевой панели и блок-диаграммы. Доступ к меню быстрого вызова осуществляется щелчком правой кнопки мыши по объекту.

На рисунке 2-13 показано контекстное меню шкального индикатора.

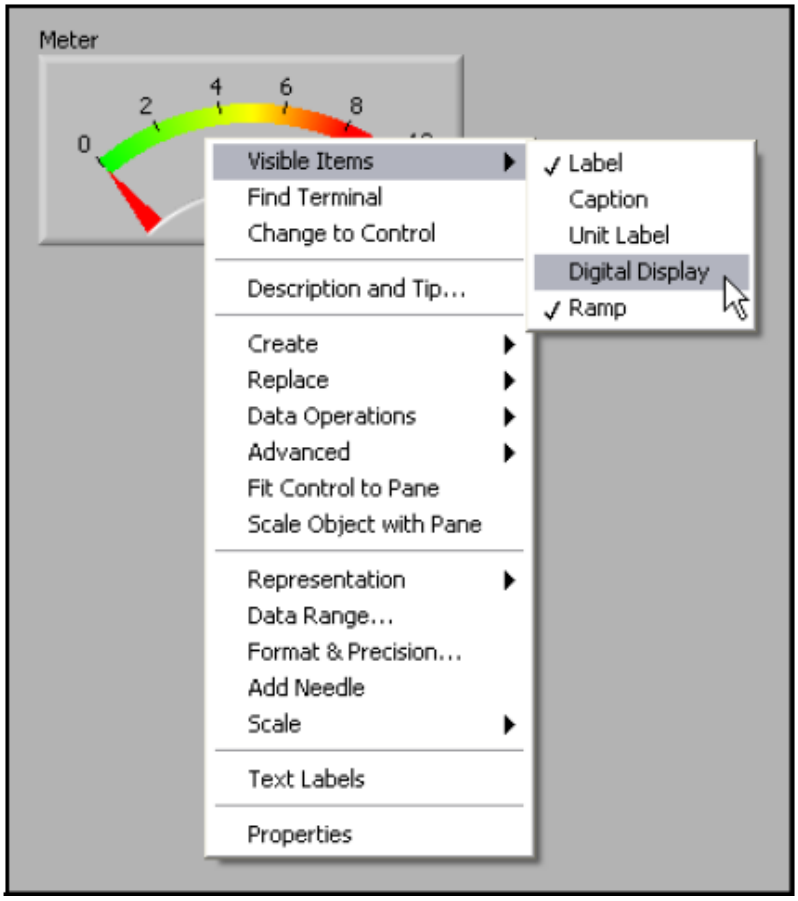


Рисунок 2-13. Контекстное меню шкального индикатора

Диалоговые окна свойств

Объекты лицевой панели имеют также диалоговые окна свойств, с помощью которых можно изменить внешний вид или характеристики объектов. Щелкните правой кнопкой мыши по объекту и выберите из контекстного меню команду **Properties**, чтобы открыть диалоговое окно свойств объекта. На рисунке 2-14 показано диалоговое окно свойств шкального индикатора, изображенного на рисунке 2-13. Варианты, доступные в диалоговом окне свойств некоторого объекта аналогичны вариантам, доступным в контекстном меню для этого же объекта.

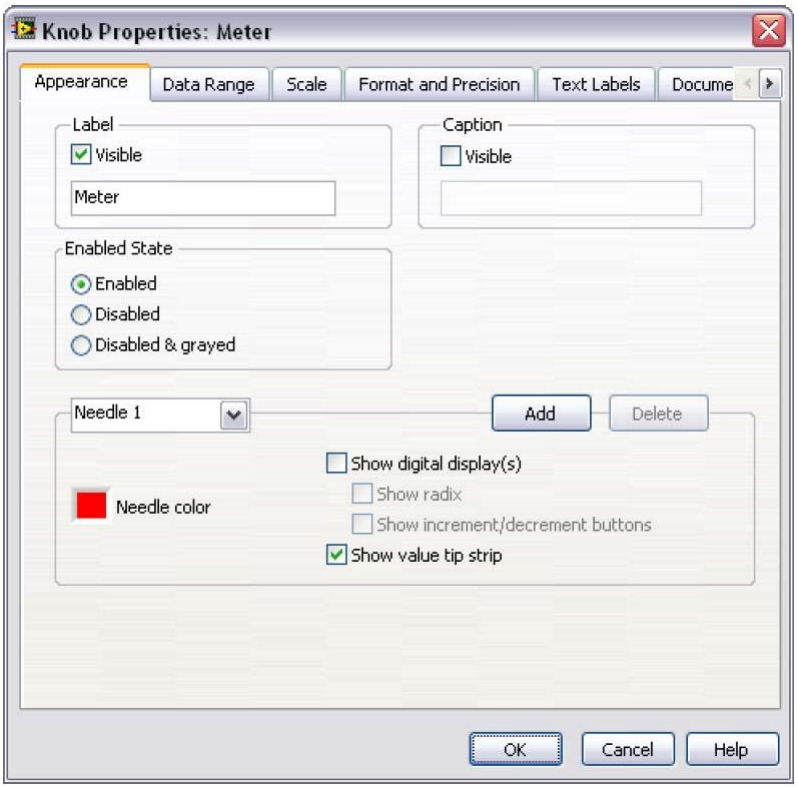


Рисунок 2-14. Диалоговое окно свойств индикатора

Можно выделить несколько объектов на лицевой панели или блок-диаграмме и редактировать любые их общие свойства. Чтобы выделить несколько объектов, с помощью инструмента Positioning растяните прямоугольную область выделения так, чтобы она охватила все объекты, которые желаете редактировать, или, щелкайте мышью по каждому объекту, удерживая при этом нажатой клавишу <Shift>. Чтобы вывести на экран диалоговое окно **Properties**, щелкните правой кнопкой мыши по какому-нибудь выделенному объекту и из контекстного меню выберите команду **Properties**. В диалоговом окне **Properties** отображаются общие для выделенных объектов закладки и свойства. Выделяйте похожие объекты, чтобы вывести больше закладок и свойств. Если вы выделите объекты, у которых нет ни одного общего свойства, в диалоговом окне **Properties** не не будет никаких закладок или свойств.

Панель инструментов окна лицевой панели

В каждом окне есть связанная с ним панель инструментов. Кнопки на панели инструментов лицевой панели служат для запуска и редактирования VI.

В окне лицевой панели появляется следующая панель инструментов:

Чтобы запустить VI, щелкните мышью по кнопке **Run**. Если необходимо, LabVIEW компилирует VI. Вы можете запускать VI, если стрелка на кнопке **Run** сплошь закрашена в белый цвет, как показано слева. Такой вид стрелки показывает также, что VI можно использовать в качестве subVI, если создадите панель подключения VI.

В процессе выполнения VI кнопка **Run** приобретает вид, показанный слева, если это VI верхнего уровня, что означает, что никакая программа его не вызывает, и, следовательно, он не является subVI.

Если выполняющийся VI является subVI, кнопка **Run** становится такой, как показано слева.

Кнопка **Run** выглядит разорванной, если создаваемый или редактируемый VI содержит ошибки. Если кнопка **Run** остается разорванной даже после того, как вы закончили делать соединения на блок-диаграмме, VI является неисправным и неработоспособным. Щелкните мышью по этой кнопке, чтобы вывести на экране окно **Error list**, в которое выводятся все ошибки и предупреждения.

Чтобы запустить VI на исполнение до момента, когда вы прервете или приостановите исполнение, щелкните мышью по кнопке **Run Continuously**. Вы можете еще раз щелкнуть по этой кнопке, чтобы запретить непрерывное исполнение.

В процессе выполнения VI появляется кнопка **Abort Execution**. Щелкните мышью по этой кнопке, чтобы немедленно остановить VI, если нет никаких других способов его остановить. Если VI используется несколькими VI более высокого уровня, кнопка становится недоступной.

Внимание! Кнопка **Abort Execution** немедленно останавливает VI, до того, как VI завершает текущую итерацию. Аварийное прекращение работы VI, который использует внешние ресурсы, такие, как внешние аппаратные средства, может оставить эти ресурсы в неизвестном состоянии из-за невыполнения сброса или некорректного их освобождения. Во избежание этой проблемы следует проектировать VI, в которых есть кнопка останова.

Чтобы приостановить выполнение VI, щелкните мышью по кнопке **Pause**. Когда вы щелкаете по кнопке **Pause**, LabVIEW подсвечивает место на блок-диаграмме, на котором приостановлено исполнение VI, а кнопка **Pause** становится красного цвета. Чтобы продолжить выполнение VI, щелкните еще раз по кнопке **Pause**.

Чтобы изменить настройки шрифта для отдельных фрагментов VI, в том числе размер, стиль и цвет шрифта, выберите выпадающее меню **Text Settings**.

Чтобы выровнять объекты вдоль осей, в том числе по вертикали, по верхнему краю, по левому краю и т.д., выберите выпадающее меню **Align Objects**.

Для выравнивания промежутков между объектами - раздвижения, сжатия и т.д., выберите выпадающее меню **Distribute Objects**.

Чтобы выровнять размер нескольких объектов лицевой панели, выберите выпадающее меню **Resize Objects**

Если у вас есть объекты, которые перекрывают друг друга, и вы хотите определить, какой из них будет впереди или позади другого, выберите выпадающее меню **Reorder.** Выделите один из объектов с помощью инструмента Positioning и затем выберите команду **Move Forward**, **Move Backward**, **Move To Front** и **Move To Back**.

Чтобы включить/выключить отображение окна **Context Help**, воспользуйтесь кнопкой **Show Context Help Window**

Кнопка **Enter Text** появляется, чтобы напомнить о том, что старое значение можно заменить на новое. Эта кнопка исчезнет, если вы щелкните по ней мышью, нажмете на клавишу <Enter> или щелкните в рабочей области лицевой панели или блок-диаграммы.

Совет: Клавиша <Enter> на цифровой клавиатуре завершает ввод текста, в то время, как клавиша <Enter> на основной клавиатуре добавляет новую строку. Изменить результат нажатия кнопок можно, выбрав команду меню **Tools»Options**, пункт **Environment** в списке **Category** и установив флажок **End text entry with Enter**.

F. Блок-диаграмма

К объектам блок-диаграммы относятся терминалы, subVI, функции, константы, структуры, а также проводники, по которым данные передаются между объектами.

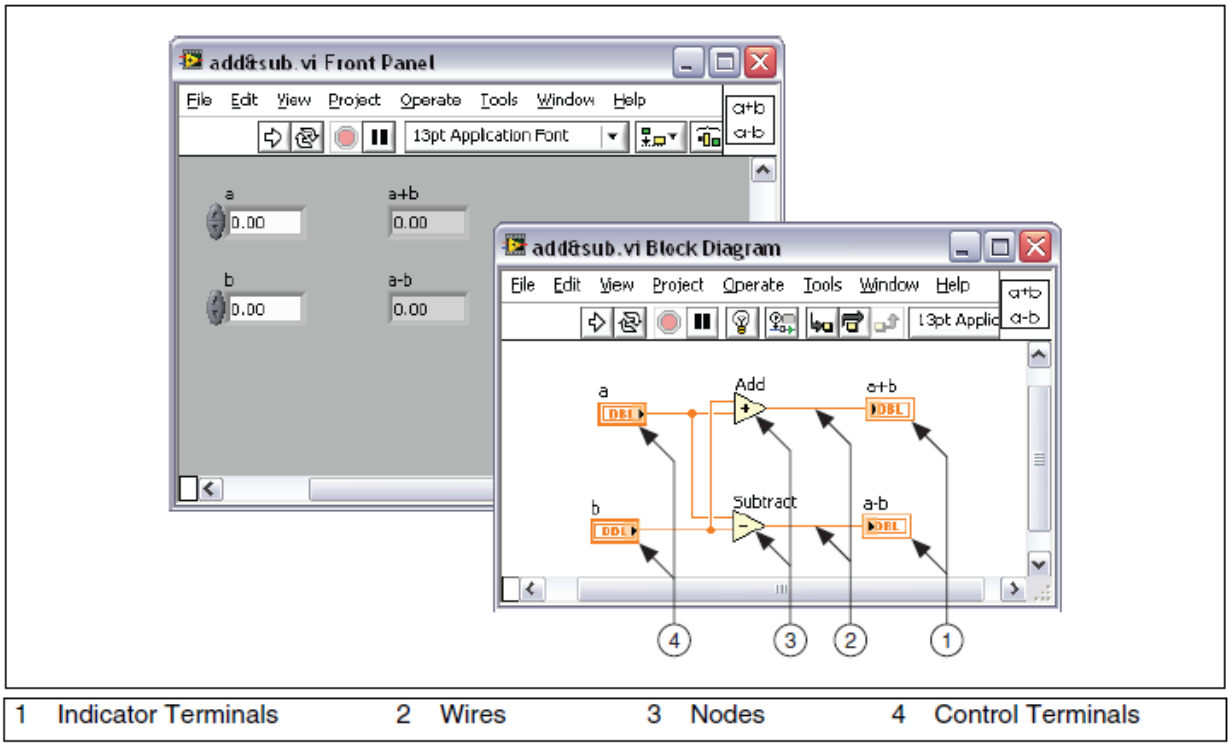


Рисунок 2-15. Пример блок-диаграммы и соответствующей ей лицевой панели

Терминалы

Объекты лицевой панели проявляются на блок-диаграмме в виде терминалов. Терминалы — это входные и выходные порты, через которые осуществляется обмен информацией между лицевой панелью и блок-диаграммой. Терминалы являются аналогами параметров и констант в текстовых языках программирования. Терминалы бывают у элементов управления, элементов индикации, а также у узлов. Терминалы элементов управления и индикации принадлежат соответствующим органам управления и индикации на лицевой панели. Данные, которые вы вводите в элементы управления на лицевой панели (**a** и **b** на рисунке 2-15), поступают через соответствующие терминалы на блок-диаграмму. Здесь данные подаются на входы функций Add и Subtract. Когда эти функции завершают вычисления, они выдают новые значения данных, которые проходят на терминалы индикаторов, обновляя содержимое соответствующих индикаторов на лицевой панели (**a+b** и **a–b** на рисунке 2-15).

Терминалы, изображенные на рисунке 2-15, относятся к четырем органам управления и индикации лицевой панели. Поскольку терминалы служат входами и выходами VI, subVI и функции также имеют терминалы, показанные слева. Например, панели подключения функций Add и Subtract

имеют по три терминала. Чтобы отобразить терминалы функции на блок-диаграмме, щелкните правой кнопкой мыши по узлу функции и выберите из контекстного меню команду **Visible Items»Terminals**.

Элементы управления, индикаторы и константы

Элементы управления, индикаторы и константы являются входами и выходами алгоритма, реализуемого блок-диаграммой. Рассмотрим реализацию алгоритма вычисления площади треугольника:

```
Area = .5 * Base * Height
```

Согласно рисунку 2-16, в данном алгоритме элементы **Base** и **Height** — входы, а элемент **Area** — выход.

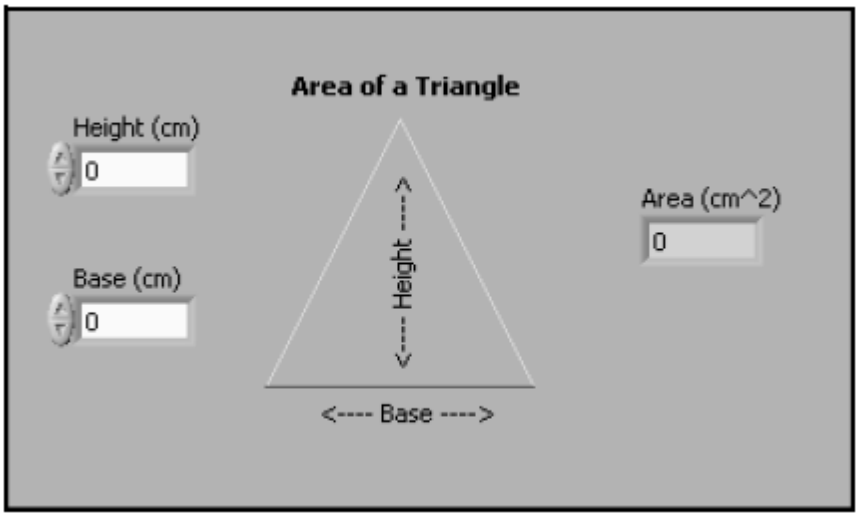


Рисунок 2-16. Лицевая панель Area of a Triangle

Пользователь не может получить доступ к константе 0.5 и не может изменить ее, поскольку она, в соответствии с алгоритмом, не отображается на лицевой панели.

На рисунке 2-17 приведена возможная реализация данного алгоритма блок-диаграммой LabVIEW. На этой блок-диаграмме есть четыре различных терминала, порожденных двумя элементами управления, одной константой и одним индикатором.

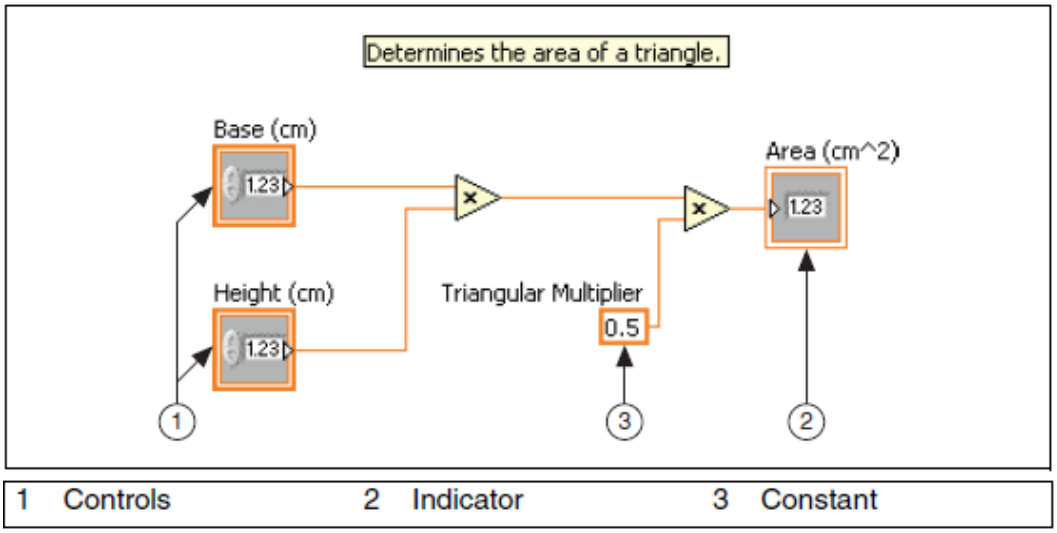


Рисунок 2-17. Блок-диаграмма Area of a Triangle с терминалами в виде иконок

Обратите внимание на то, что внешний вид терминалов **Base (cm)** и **Height (cm)** на блок-диаграмме отличается от внешнего вида терминала **Area (cm^2)**. Есть два отличительных признака между элементом управления и индикатором на блок-диаграмме. Первый из них — стрелка на терминале, которая показывает направление потока данных. Стрелки на терминалах элементов управления показывают направление, по которому данные покидают терминалы, в то время как стрелка на терминалах индикаторов показывает направление, по которому данные поступают в терминалы. Второй отличительный признак — рамка вокруг терминала. Рамка терминалов элементов управления более толстая, чем рамка терминалов индикаторов.

Терминалы можно показывать в виде пиктограммы (иконки) или без использования пиктограммы. На рисунке 2-18 приведена та же самая блок-диаграмма, что и на рисунке 2-17, только терминалы на ней показаны без использования пиктограмм. Тем не менее, отличительные признаки между элементами управления и индикаторами на блок-диаграмме те же самые.

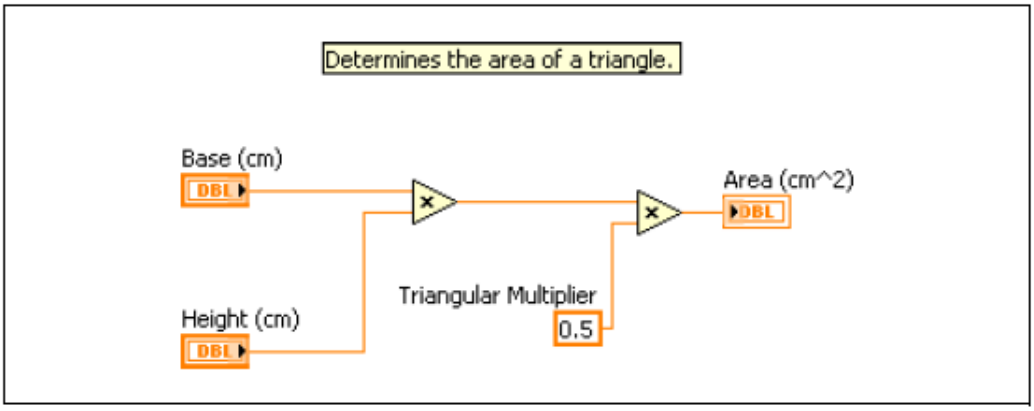


Рисунок 2-18. Блок-диаграмма Area of a Triangle; терминалы показаны не иконками

Узлы блок-диаграммы

Узлы — это объекты блок-диаграммы, у которых есть входы и/или выходы и которые выполняют действия в исполняемом VI. Они являются аналогами выражений, операторов, функций и подпрограмм в текстовых языках программирования. Узлами могут быть функции, subVI и структуры. Структуры являются элементами управления процессом, например, Case, For Loop, While Loop. Функции Add и Subtract на рисунке 2-15 являются узлами функций.

Функции

Функции являются базовыми рабочими элементами LabVIEW. У функций нет лицевых панелей и блок-диаграмм, но у них есть панели подключения. Двойной щелчок мыши по функции только выделяет эту функцию. Иконка функции имеет бледно-желтый фон.

SubVI

SubVI — это VI, которые создаются для использования внутри другого VI, или VI, к которым вы получаете доступ через палитру **Functions**.

Любой VI потенциально может быть использован в качестве subVI. Если щелкнуть дважды по иконке subVI на блок-диаграмме, откроется его лицевая панель. На лицевой панели находятся органы управления и индикации. Блок-диаграмма включает в себя проводники, иконки, функции, возможно и subVI, а также другие объекты LabVIEW. В верхнем правом углу окна лицевой панели и окна блок-диаграммы отображается иконка VI, которая появляется на блок-диаграмме, когда вы помещаете на нее VI в качестве subVI.

Express VI тоже можно использовать в качестве subVI. Express VI — это узлы, для которых требуется минимальное количество соединений, поскольку они конфигурируются с помощью диалоговых окон. Express VI следует использовать для решения стандартных измерительных задач. Вы можете сохранить конфигурацию Express VI как subVI. За более подробной информацией о том, как создать subVI на основе сконфигурированного Express VI, обратитесь к разделу *Express VIs* справки *LabVIEW Help*.

Различить Express VI и другие VI на блок-диаграмме можно по цвету иконки. На блок-диаграмме иконки Express VI появляются на синем поле, а иконки subVI – на желтом.

Сравнение расширяемых узлов и иконок

VI и Express VI можно отображать в виде иконок или в виде расширяемых узлов. Расширяемые узлы имеют вид иконок, окруженных цветным полем. SubVI отображаются на желтом поле, а Express VI – на синем. Используйте иконки, если хотите сэкономить пространство на блок-диаграмме. Применяйте расширяемые узлы, чтобы облегчить выполнение соединений и

документирование блок-диаграмм. По умолчанию subVI на блок-диаграмме отображаются в виде иконок, а Express VI – в виде расширяемых узлов. Чтобы отобразить subVI или Express VI в виде расширяемого узла, щелкните правой кнопкой мыши по subVI или Express VI и снимите флажок рядом с пунктом контекстного меню **View As Icon**.

Вы можете изменять размер расширяемого узла, чтобы упростить выполнение соединений, однако он все равно занимает значительную часть пространства на блок-диаграмме. Чтобы изменить размер узла, выполните следующие действия:

1. Поместите инструмент Positioning на узел. Вверху и внизу этого узла появятся метки для изменения размера.
2. Поместите курсор на метку изменения размера, чтобы изменить тип курсора на курсор изменения размера.
3. С помощью курсора изменения размера потяните границу узла вниз, чтобы стали видимыми дополнительные терминалы.
4. Отпустите кнопку мыши.

Чтобы отменить операцию изменения размера, перед тем, как отпустить кнопку мыши, растяните границу узла за пределы окна блок-диаграммы.

На рисунке 2-19 изображен расширяемый узел Basic Function Generator VI с измененным размером.

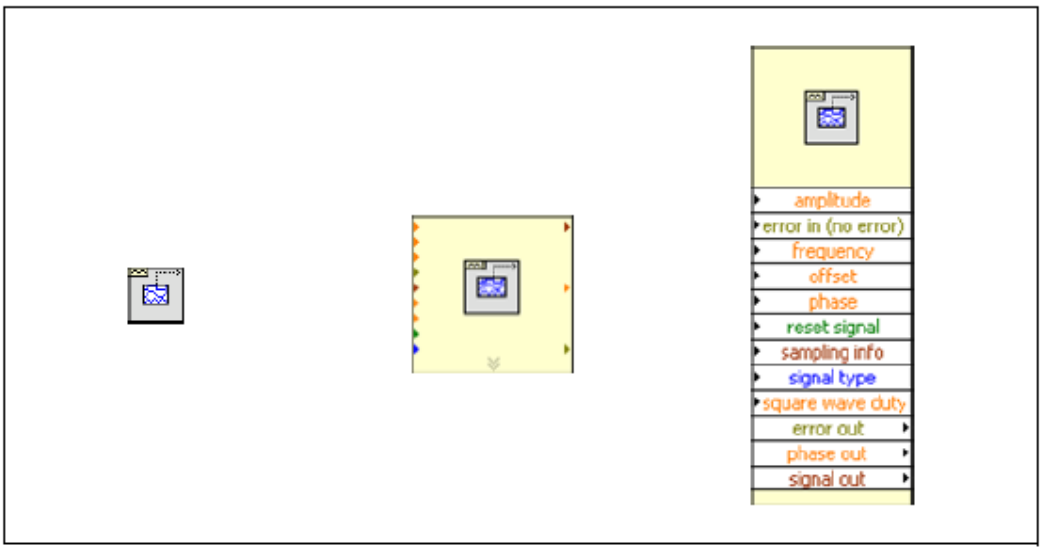


Рисунок 2-19. Basic Function Generator VI в различных режимах отображения

Примечание: Если subVI или ExpressVI отображается в виде расширяемого узла, вы не можете увидеть его терминалы, а кроме того, к такому узлу нельзя разрешить доступ со стороны базы данных.

Проводники

Данные передаются между объектами блок-диаграммы по проводникам. На рисунке 2-15 терминалы элементов управления и индикации подсоединяются к функциям Add и Subtract с помощью проводников. Каждый проводник имеет один источник данных, однако, его можно присоединять ко многим VI и функциям, которые считывают эти данные. Проводники, в зависимости от типов передаваемых по ним данных, бывают различных цветов, стилей и толщины.

Разорванный проводник имеет вид прерывистой черной линии с красным крестиком посередине, как показано слева. Обрывы проводников возникают из-за множества причин, в том числе, при попытке соединить два объекта с несовместимыми типами данных.

В таблице 2-1 приведены наиболее распространенные типы проводников.

Таблица 2-1. Распространенные типы проводников

Тип проводника	Скаляр	1D массив	2D массив	Цвет
Числовой				Оранжевый (с плавающей точкой), Синий (целочисленный)
Логический				Зеленый
Строковый				Розовый

В LabVIEW, чтобы передавать данные в VI, используются проводники, соединяющие между собой множество терминалов. Следует подсоединять проводники к тем входам и выходам, которые совместимы с данными, передаваемыми по этим проводникам. Нельзя, например, соединять выход массива с числовым входом. Кроме того, необходимо, чтобы проводники соединяли только один выход, по крайней мере, с одним входом. Нельзя, например, соединять между собой два индикатора. Компонентами, которыми определяется совместимость соединения, являются тип данных элемента управления и/или индикатора и тип данных терминала.

Типы данных

Типы данных показывают, какие объекты, входы и выходы можно соединять между собой. Если, например, переключатель имеет зеленую рамку, его можно присоединять к любому входу Express VI, отмеченным зеленым цветом. Если у какого-нибудь регулятора отображается в оранжевой рамке, его можно подсоединить к любому входу оранжевого

wdtnf. Однако, нельзя присоединять оранжевый регулятор ко входу, отмеченному зеленым цветом. Обратите внимание на то, что проводники окрашиваются в тот же цвет, что и терминал, к которому они присоединены.

Автоматическое соединение объектов

Если приблизить выделенный объект к другим объектам на блок--диаграмме, LabVIEW нарисует временные проводники, которые показывают корректные соединения. Когда вы отпустите кнопку мыши, чтобы поместить объект на блок-диаграмму, LabVIEW автоматически подключит проводники. Вы можете также автоматически соединять объекты уже на блок-диаграмме. LabVIEW выполняет соединения наиболее совместимых терминалов и не соединяет терминалы, которые несовместимы.

Автоматическое выполнение соединение включается клавишей пробела, когда вы перемещаете объект с помощью инструмента Positioning.

Автоматическое выполнение соединений разрешено по умолчанию, если объект выбирается из палитры **Functions** или при копировании объекта на блок-диаграмме нажатием клавиши <Ctrl> и перетаскиванием объекта. Автоматическое выполнение соединений по умолчанию запрещено, если вы с помощью инструмента Positioning перемещаете объект, уже находящийся на блок-диаграмме.

Установить настройки автоматического выполнения соединений можно, выбрав команду меню **Tools»Options** и пункт **Block Diagram** из списка **Category**.

Соединение объектов вручную

Когда инструмент Wiring перемещается над терминалом, появляется подсказка с именем терминала. Кроме того, терминал мерцает в окне **Context Help** и на иконке, помогая вам убедиться в том, что выполняемое соединение с данным терминалом корректно. Чтобы соединить объекты между собой, щелкните мышью, наведя инструмент Wiring на первый терминал, затем поместите курсор на второй терминал и снова щелкните мышью. После выполнения соединения можно щелкнуть правой кнопкой мыши по проводнику и выбрать из контекстного меню команду **Clean Up Wire**, чтобы LabVIEW автоматически выбрала путь прохождения этого проводника. Если у вас есть разорванные проводники, которые необходимо удалить, нажмите комбинацию клавиш <Ctrl-B>, чтобы удалить все разорванные проводники на блок-диаграмме.

Палитра Functions

Палитра **Functions** содержит VI, функции и константы, которые используются при создании блок-диаграммы. Доступ к палитре **Functions** осуществляется из блок-диаграммы командой меню **View»Functions Palette**. Палитра **Functions** разбита на категории, которые можно скрывать и делать видимыми в соответствии с вашими потребностями. На рисунке 2-20

приведена палитра **Functions** со всеми видимыми категориями и раскрытой разделом **Programming**. В процессе изучения данного курса вам предстоит работать в основном с разделом **Programming**, однако вы также будете использовать и другие категории или субпалитры.

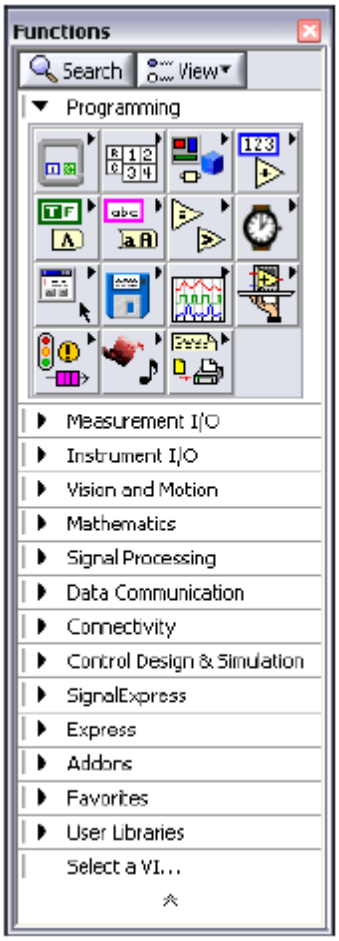


Рисунок 2-20. Палитра Functions

Для того, чтобы сделать видимыми или скрыть категории, щелкните мышью по кнопке **View** на палитре, выберите пункт **Change Visible Categories** и установите или сбросьте соответствующие флажки.

Панель инструментов блок-диаграммы

При запуске VI на панели инструментов блок-диаграммы появляются кнопки, с помощью которых вы можете отлаживать VI. Панель инструментов блок-диаграммы имеет следующий вид:

Щелкните мышью по кнопке **Highlight Execution**, чтобы анимировать исполнение блок-диаграммы при запуске VI. Обратите внимание на движение потока данных по блок-диаграмме. Щелкните еще раз мышью по этой кнопке, чтобы запретить подсветку выполнения.

Щелкните мышью по кнопке **Retain Wire Values** для сохранения значений данных в проводниках в каждой точке выполняемого потока, чтобы при установке пробника на проводник можно было получать последнее значение данных, прошедшее по проводнику. Эти последние значения будут доступны только, если VI успешно выполнится как минимум один раз.

Щелкните мышью по кнопке **Step Into**, чтобы открыть узел и приостановить выполнение. Если вы еще раз щелкните мышью по кнопке **Step Into**, выполнится первое действие, после которого выполнение subVI или структуры вновь приостановится. Аналогично можно нажимать на клавиши <Ctrl> и <стрелка вниз>. Пошаговое выполнение VI происходит от узла к узлу. О готовности каждого узла к исполнению свидетельствует его мерцание.

Щелкните мышью по кнопке **Step Over**, чтобы выполнить узел и приостановиться на следующем узле. Аналогично можно нажимать на клавиши <Ctrl> и <стрелка вправо>. В этом случае узел выполняется без пошагового исполнения внутри него.

Щелкните мышью по кнопке **Step Out**, чтобы завершить исполнение текущего узла и приостановиться. Когда VI завершает исполнение, кнопка **Step Out** становится недоступной. Аналогично можно нажимать на клавиши <Ctrl> и <стрелка вверх>. В этом случае завершается пошаговое исполнение в текущем узле и происходит переход к следующему узлу.

Щелкните мышью по кнопке **Clean Up Diagram**, чтобы автоматически переразвести все имеющиеся проводники и упорядочить объекты на блок-диаграмме, скомпоновав их более аккуратно. Конфигурирование параметров перекомпоновки объектов на блок-диаграмме осуществляется командой меню **Tools»Options**, и выбором пункта **Block Diagram: Cleanup** из списка **Category** в открывающемся диалоговом окне Options.

Кнопка **Warning** появляется, если в VI имеется предупреждение, а в окне **Error List** был установлен флажок **Show Warnings**. Предупреждение указывает на потенциальную проблему в блок-диаграмме, которая, однако, не приводит к прерыванию работы VI.

G. Поиск органов управления, VI и функций

Если вы открываете палитры **Controls** или **Functions** командой меню **View»Controls** или **View»Functions**, вверху палитр появляются две кнопки.

Search — переводит палитру в режим поиска в палитрах элементов управления, VI или функций по текстовому шаблону. Если палитра находится в режиме поиска, щелкните мышью по кнопке Return, чтобы выйти из этого режима и вернуться в палитру.

View — предоставляет возможности выбора формата текущей палитры, состава видимых и скрываемых категорий всех палитр, а также сортировки элементов палитр в алфавитном порядке для форматов **Text** и **Tree**. В диалоговом окне **Options** можно выбрать формат всех палитр, чтобы

открыть это окно, выберите из контекстного меню кнопки **View** команду **Options**. Кнопка **View** появится только в том случае, если вы «прикрепите» палитру, щелкнув мышью по «канцелярской кнопке» в левом верхнем углу палитры.

Пока вы не привыкнете к расположению VI и функций, поиск функции или VI следует осуществлять с помощью кнопки **Search**. Если, например, вы хотите найти функцию Random Number, щелкните мышью по кнопке **Search** на панели инструментов палитры **Functions** и в текстовом поле вверху палитры начните набор текста `Random Number`. LabVIEW отображает все элементы, которые начинаются с набранного вами текста или содержат его. Вы можете щелкнуть мышью по одному из результатов поиска и перетащить его на блок-диаграмму, как показано на рисунке 2-21.

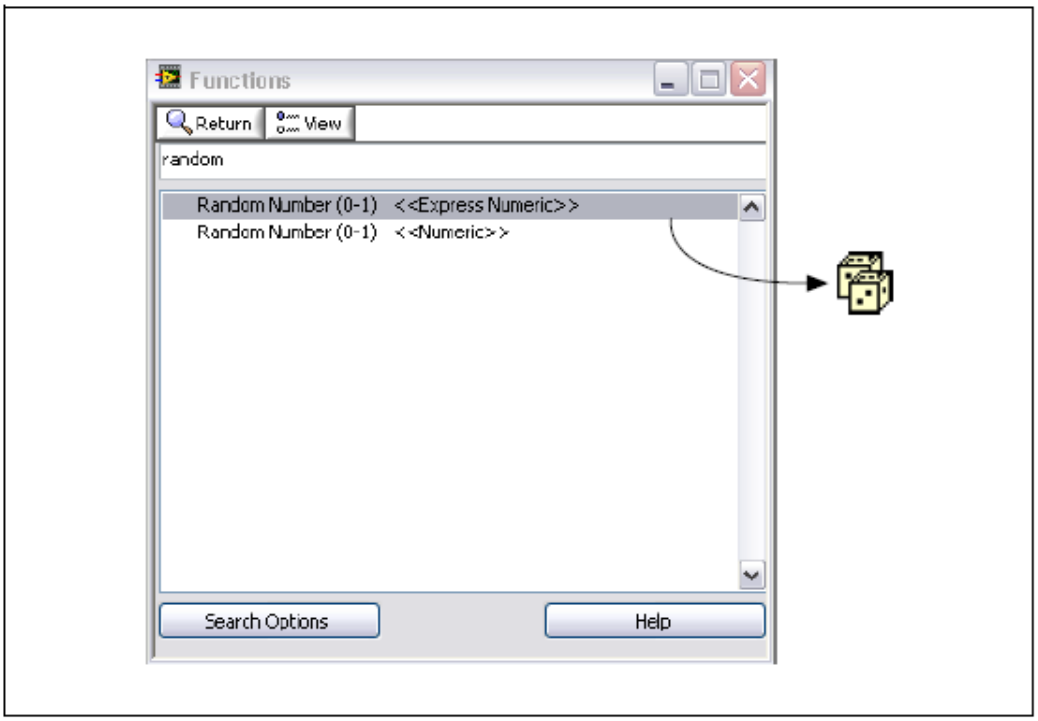


Рисунок 2-21. Поиск объекта в палитре Functions

Дважды щелкните мышью по результату поиска, чтобы найти его место в палитре. Если найденный объект нужен для частого использования, вы можете его добавить в категорию Favorites. Щелкните правой кнопкой мыши по объекту в палитре и, как показано на рисунке 2-22, выберите команду **Add Item to Favorites**.

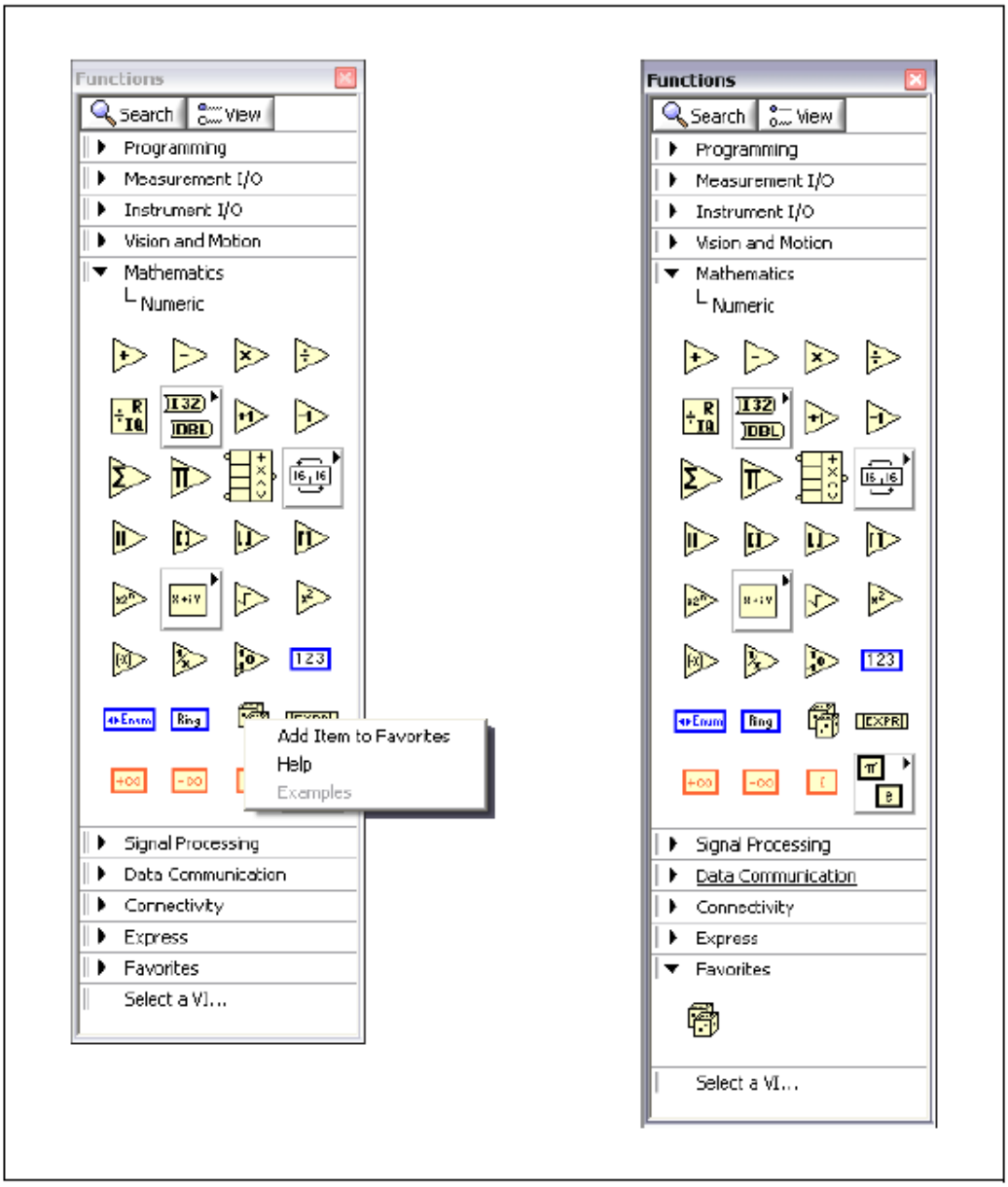


Рисунок 2-22. Включение объекта в категорию Favorites палитры

H. Выбор инструмента

С помощью инструментов LabVIEW можно создавать, модифицировать и отлаживать VI. Инструмент — это специальный режим работы курсора мыши. Режим работы курсора соответствует иконке выбранного инструмента. В зависимости от текущего положения мыши LavVIEW определяет, какой выбрать инструмент.

Рисунок 2-23. Палитра инструментов

Совет: Вы можете выбрать нужный инструмент в палитре **Tools** вручную. Чтобы открыть палитру **Tools**, выберите команду меню **View»Tools Palette**.

Инструмент Operating

Инструмент Operating становится рабочим, когда курсор мыши принимает вид, показанный слева. Этот инструмент используется для изменения значений элемента управления. Например, на рисунке 2-24 инструмент Operating перемещает указатель на элементе управления Horizontal Pointer Slide. Когда курсор мыши наводится на указатель, он автоматически выбирает инструмент Operating.

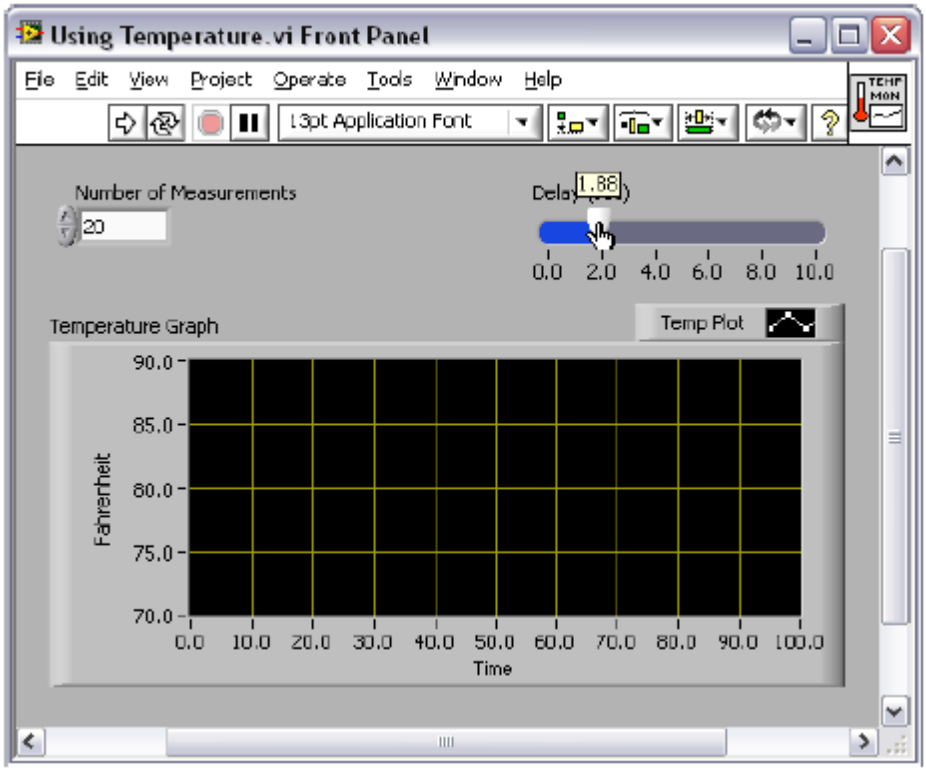


Рисунок 2-24. Использование инструмента Operating

Инструмент Operating чаще всего используется в окне лицевой панели, однако, его можно также использовать и в окне блок диаграммы, чтобы изменить значение константы типа Boolean.

Инструмент Positioning

Инструмент Positioning становится рабочим, когда курсор мыши принимает вид, показанный слева. Используйте этот инструмент для выделения и изменения размеров объектов. Например, на рисунке 2-25 инструмент Positioning выделяет числовой элемент управления **Number of Measurements**. После того, как объект выделен, его можно переместить, скопировать или удалить. Когда курсор мыши наводится на границу объекта, он автоматически превращается в инструмент Positioning.

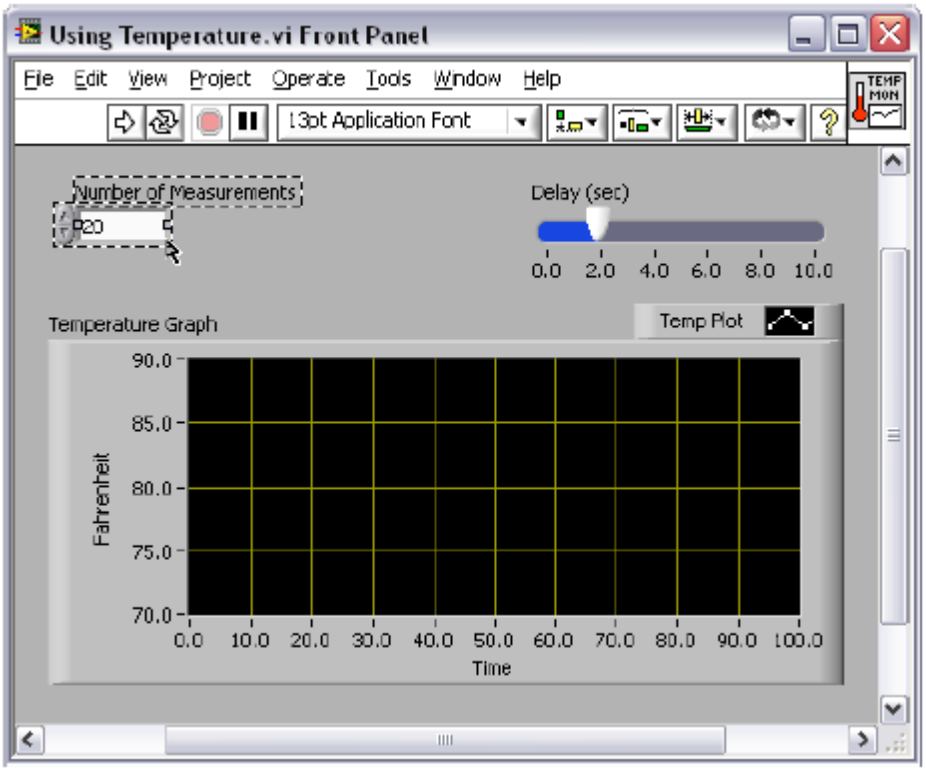


Рисунок 2-25. Использование инструмента Positioning для выделения объекта

Если курсор навести на реперную точку изменения размера объекта, режим курсора изменяется, показывая, что этот размер можно изменять (рисунок 2-26). Обратите внимание – курсор над углом индикатора XY Graph в точке изменения размера превращается в двустороннюю стрелку.

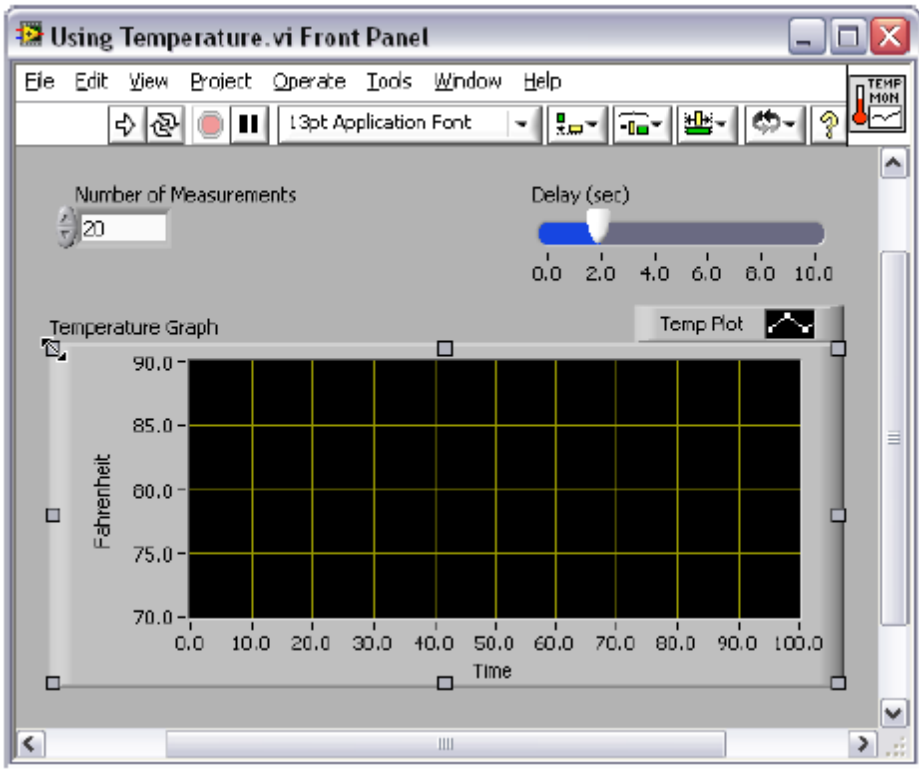


Рисунок 2-26. Использование инструмента Positioning для изменения размеров объекта

Инструментом Positioning можно пользоваться как на лицевой панели, так и на блок-диаграмме.

Инструмент Labeling

Инструмент Labeling становится рабочим, когда курсор мыши принимает вид, показанный слева. Этот инструмент используйте для ввода текста в элемент управления, редактирования текста и создания свободных меток. Например, на рисунке 2-27 с помощью инструмента Labeling вводится текст в числовой элемент управления **Number of Measurements**. Когда курсор мыши наведен на внутреннюю часть этого элемента управления, он автоматически превращается в инструмент Labeling. Щелкните один раз мышью, чтобы поместить курсор внутрь элемента управления. Затем сделайте двойной щелчок, чтобы выделить находящийся в элементе текст.

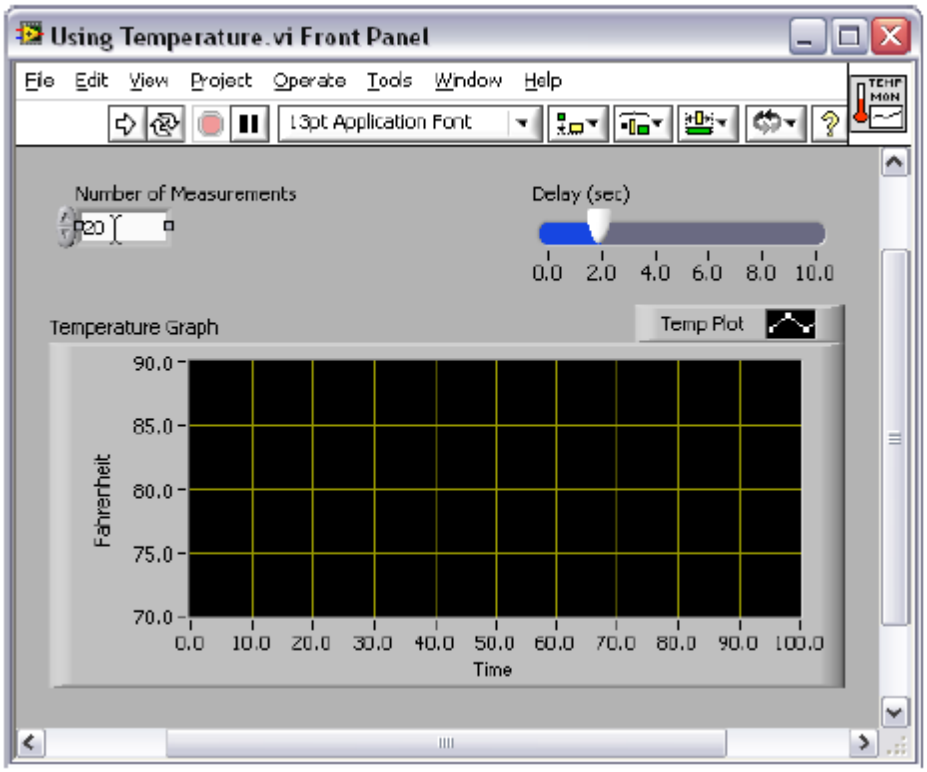


Рисунок 2-27. Использование инструмента Labeling

Когда курсор находится вне определенной области окна лицевой панели или окна блок-диаграммы, в которой возможен выбор конкретного режима использования мыши, курсор принимает вид перекрестия. Если разрешен автоматический выбор инструмента, можно, щелкнув дважды по любому свободному месту, выбрав тем самым инструмент Labeling, и создать свободную метку.

Инструмент Wiring

Инструмент Wiring становится рабочим, когда курсор мыши принимает вид, показанный слева. Этот инструмент используется для соединения объектов на блок-диаграмме между собой. Например, на рисунке 2-28 инструмент Wiring соединяет терминал **Number of Measurements** с терминалом количества итераций структуры For Loop. Когда курсор мыши наведен на выходную или входную точку терминала или на проводник, курсор автоматически превращается в инструмент Wiring.

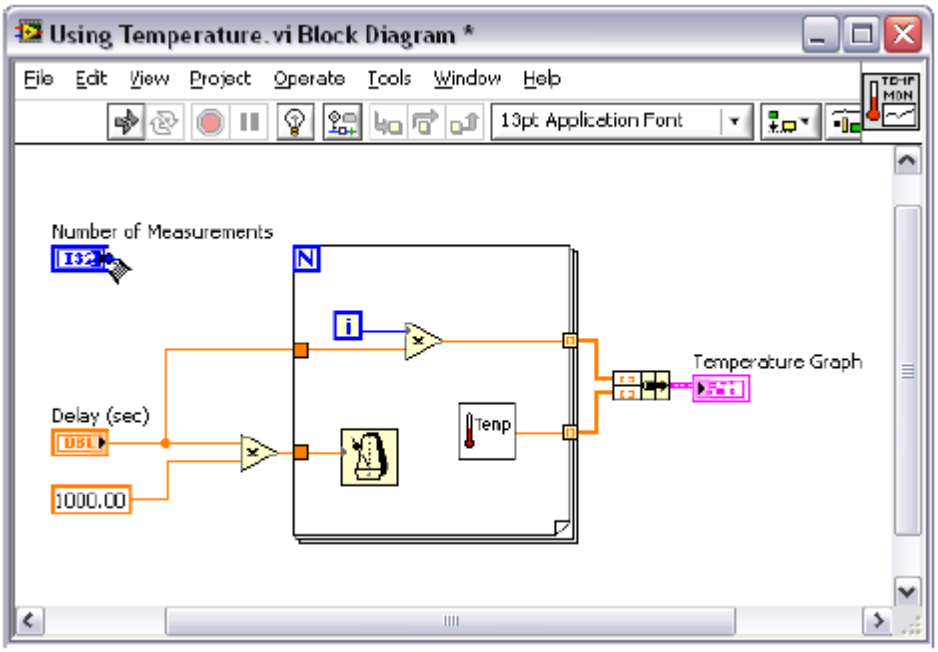


Рисунок 2-28. Использование инструмента Wiring

С инструментом Wiring в основном работают в окне блок-диаграммы и при создании панели подключения в окне лицевой панели.

Другие инструменты палитры Tools

Вы можете выбрать инструменты Operating, Positioning, Labeling и Wiring непосредственно из палитры **Tools**, что является более предпочтительным, чем использование режима автоматического выбора инструмента. Чтобы открыть палитру **Tools**, выберите команду меню **View»Tools Palette**.

Рисунок 2-29. Палитра инструментов

Верхним элементом палитры **Tools** является кнопка Automatic Tool Selection. Когда эта кнопка нажата, LabVIEW автоматически выбирает инструмент в зависимости от положения курсора. Отжав эту кнопку или выбрав в палитре другой инструмент, вы можете отключить автоматический выбор инструмента. В палитре имеется несколько дополнительных инструментов, описанных ниже:

Инструмент Object Shortcut Menu служит для доступа к контекстному меню объекта с помощью левой кнопки мыши.

Инструмент Scrolling предназначен для прокрутки в окнах вместо использования линейки прокрутки.

Инструмент Breakpoint используется для установки контрольных точек в VI, функциях, узлах, на проводниках и структурах, чтобы приостановить в этой точке выполнение VI.

Инструмент Probe применяется для создания пробников на проводниках блок-диаграммы. Этот инструмент служит для проверки промежуточных значений внутри VI, который выдает сомнительные или непредусмотренные результаты.

Инструмент Color Copy используется для копирования цветов и вставки их в инструмент Coloring.

Инструмент Coloring применяется для раскраски объекта. Этот инструмент отображает текущие настройки цвета переднего плана и цвета фона.

I. Потоковое программирование

В LabVIEW VI выполняются под управлением потока данных, и узел блок-диаграммы выполняется только тогда, когда получит все требуемые входные данные. По завершении выполнения узел выдает выходные данные и передает их следующему узлу на пути распространения потока данных. Продвижение данных через узлы определяет порядок выполнения VI и функций на блок-диаграмме.

Программы на Visual Basic, C++, JAVA и в большинстве других текстовых языках программирования выполняются в соответствии с моделью управления потоком команд, согласно которой последовательность программных элементов определяет порядок выполнения программы.

В качестве примера потокового программирования рассмотрим блок-диаграмму, приведенную на рисунке 2-30, которая складывает два числа, и затем из суммы этих чисел вычитает число `50.00`. В этом случае блок-диаграмма выполняется слева направо не из-за того, что объекты размещены в таком порядке, а вследствие того, что функция Subtract не может выполняться до тех пор, пока не выполнится функция Add, которая передаст данные функции Subtract. Следует помнить о том, что узел выполняется только тогда, когда данные доступны на всех его входных терминалах, и выдает данные на выходные терминалы только тогда, когда узел закончит выполнение.

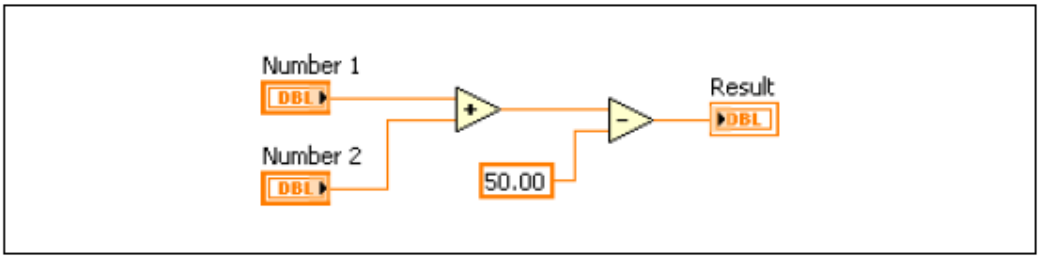


Рисунок 2-30. Пример программы, управляемой потоком данных

Проанализируем, какой сегмент программного кода на рисунке 2-31будет выполняться в первую очередь — функция Add, Random Number или Divide. Вы не сможете это узнать, поскольку данные на входах функций Add и Divide доступны одновременно, а у функции Random Number нет входов. В ситуации, когда один сегмент программного кода должен выполняться раньше другого, причем функции взаимно независимы относительно данных, следует пользоваться другими методами программирования, например, которые используют кластеры ошибок, чтобы принудительно задать порядок выполнения программы. За более подробной информацией о кластерах ошибок обратитесь к разделу *Связываемые данные* лекции 5.

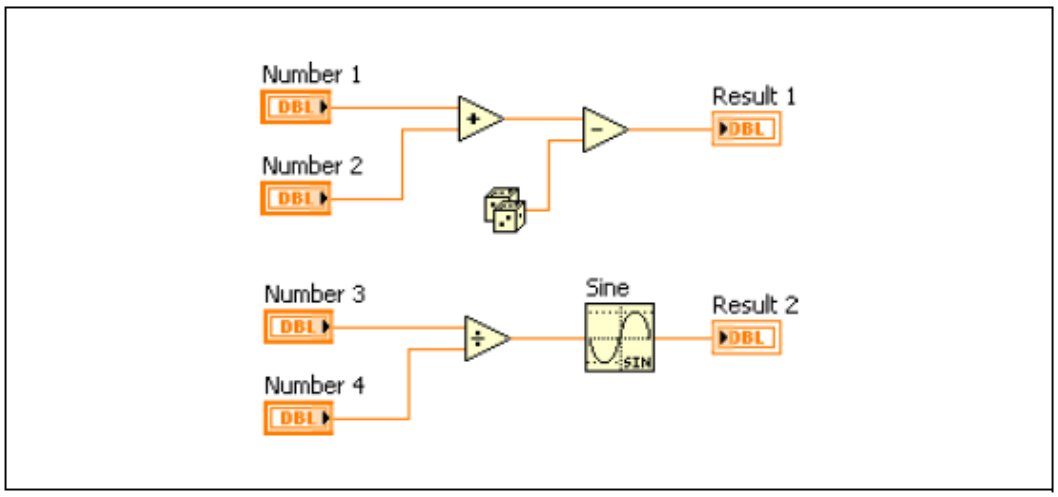


Рисунок 2-31. Пример с несколькими сегментами кода, управляемыми потоком данных

J. Разработка простого VI

Большинство VI, спроектированных в LabVIEW, решают три главных задачи: сбор данных некоторого вида, обработка полученных данных и представление результата. Если каждая из этих задач решается просто, вы можете собрать VI, используя совсем немного объектов блок-диаграммы. Express VI специально предназначены для выполнения стандартных, часто используемых операций. В настоящем разделе вам предстоит узнать о некоторых Express VI, которые выполняют сбор, обработку и представление данных. Затем вы научитесь создавать простой VI, в котором решаются три упомянутые задачи (рисунок 2-32).

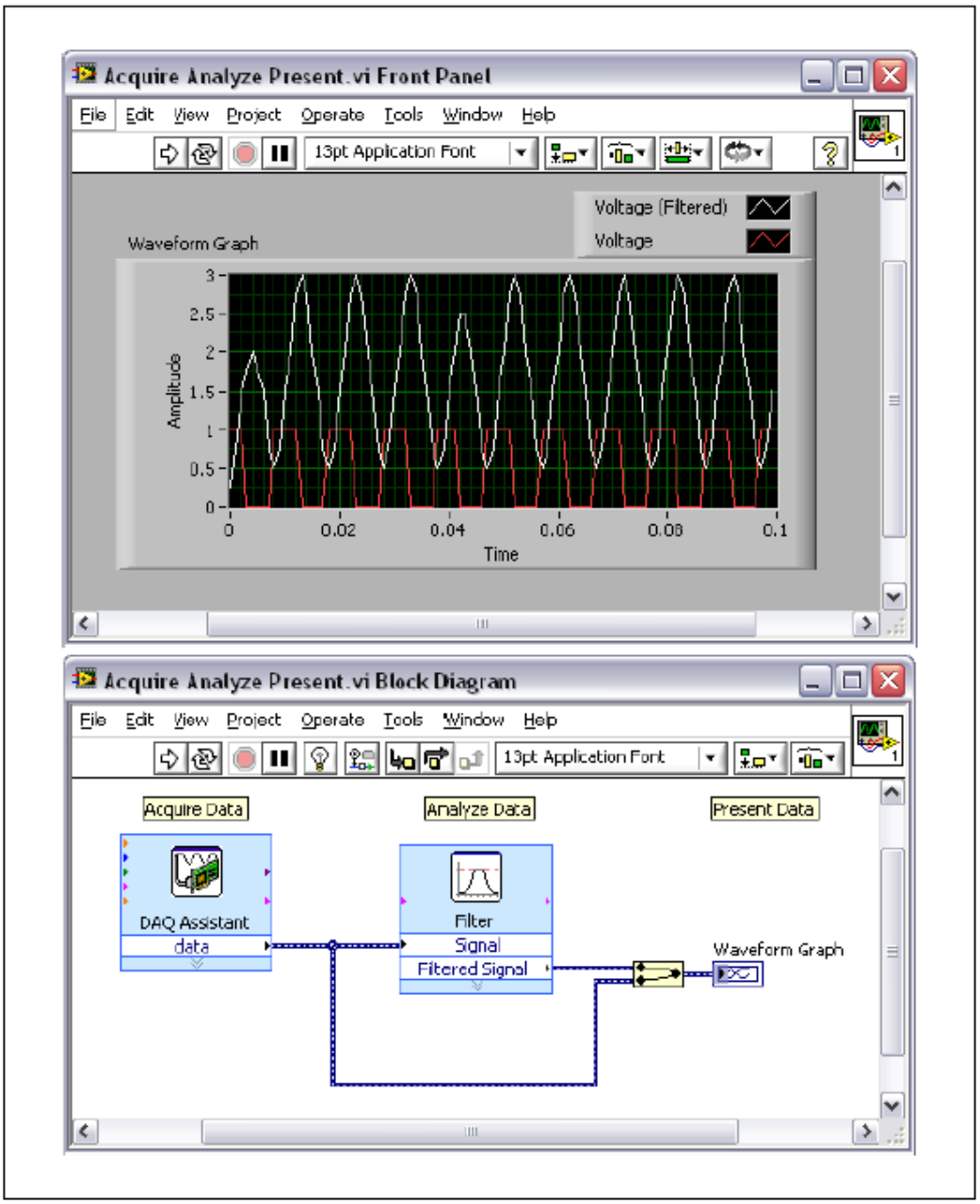


Рисунок 2-32. Лицевая панель и блок-диаграмма примера сбора, обработки и представления данных

Express VI в палитре **Functions** сгруппированы в категории **Express**. Для передачи данных между Express VI применяется динамический тип данных.

Сбор данных

Для решения задачи сбора данных служат следующие Express VI: DAQ Assistant, Instrument I/O Assistant, Simulate Signal и Read from Measurement File.

DAQ Assistant

DAQ Assistant получает данные от устройства сбора данных (DAQ-устройство). В процессе изучения данного курса вам часто придется пользоваться этим Express VI. Пока вы детально не познакомитесь со сбором данных, вы будете использовать только один канал DAQ-устройства – `CH0`. К этому каналу подключен датчик температуры, установленный во вспомогательном сигнальном блоке (DAQ Signal Accessory). Вы можете коснуться датчика, чтобы изменить температуру, измеряемую с его помощью.

Instrument I/O Assistant

Instrument I/O Assistant управляет сбором данных с измерительных приборов, обычно посредством GPIB или последовательного интерфейса

Simulate Signal

Simulate Signal Express VI генерирует смоделированные данные, например, синусоидальный сигнал.

Read From Measurement File

Read From Measurement File Express VI считывает файл в форматах LVM или TDM, который был создан с помощью Write To Measurement File Express VI. Read From Measurement File Express VI не воспринимает ASCII-файлы. Дополнительная информация о считывании данных из файла приведена в лекции 6, *Управление ресурсами*.

Обработка данных

Задачу обработки данных решают следующие Express VI: Amplitude and Level Measurements, Statistics, Tone Measurements и т.д.

Amplitude and Level Measurements

Amplitude and Level Measurements Express VI выполняет измерения параметров напряжения сигнала – постоянной составляющей, среднеквадратического значения, максимального и минимального пиковых значений, размаха, среднего значения за период и среднеквадратического значения за период.

Statistics

Statistics Express VI вычисляет статистические характеристики сигнала, в том числе – среднее значение, сумму, среднеквадратическое отклонение и экстремумы.

Spectral Measurements

Spectral Measurements Express VI выполняет измерения спектральных характеристик сигнала, в том числе амплитудного спектра и спектральной плотности мощности.

Tone Measurements

Tone Measurements Express VI осуществляет поиск гармоники с наивысшей частотой или наибольшей амплитудой. Этот VI определяет также частоту и амплитуду отдельной гармоники.

Filter

Filter Express VI обрабатывает сигнал, пропуская его через фильтры и окна. Среди применяемых фильтров фильтры типа Highpass, Lowpass, Bandpass, Bandstop и Smoothing. Используются окна с фильтрами Butterworth, Chebyshev, Inverse Chebyshev, Elliptical и Bessel.

Представление результатов

Представление результатов осуществляется с помощью Write to Measurement File Express VI или индикаторов, с помощью которых данные выводятся на лицевую панель. Наиболее часто для этой цели используют индикаторы Waveform Chart, Waveform Graph и XY Graph. Стандартные Express VI для представления данных – Write to Measurement File Express VI, Build Text Express VI, DAQ Assistant и Instrument I/O Assistant. В данном случае DAQ Assistant и Instrument I/O Assistant осуществляют вывод данных из компьютера на DAQ-устройство или внешний измерительный прибор.

Write to Measurement File

Write to Measurement File Express VI записывает данные в файл в формате LVM или TDMS. За более подробной информацией о записи результатов измерений в файлы обратитесь к лекции 6, *Управление ресурсами*.

Build Text

Build Text Express VI обычно создает текст для отображения его на лицевой панели или для экспорта его в файл или измерительный прибор. За более подробной информацией о создании строк обратитесь к лекции 6, *Управление ресурсами*.

Запуск VI

После того, как вы сконфигурировали Express VI и соединили их между собой, можно запускать VI на исполнение. Как только вы закончили разработку VI, щелкните мышью по кнопке **Run** на панели инструментов, чтобы запустить VI.

В процессе выполнения VI внешний вид кнопки Run изменяется, как показано слева. После того, как VI завершит работу, кнопка Run принимает исходный вид, а на индикаторах лицевой панели отображаются данные.

Кнопка ошибок при запуске VI

Если VI не запускается, это означает, что он содержит ошибки и не может выполняться. Если VI, который вы создаете и редактируете, содержит ошибки, на кнопке **Run** появляется разрыв.

Если разрыв на данной кнопке остается, даже когда вы закончили выполнять соединения на блок-диаграмме, следовательно, VI неработоспособен и не может быть запущен.

Обычно это означает, что вход, на который обязательно должны подаваться данные, никуда не подключен или проводник оборван. Чтобы получить доступ к окну **Error List**, щелкните мышью по разорванной кнопке запуска. В окне **Error List** указана каждая ошибка и приведено ее описание. Двойным щелчком по строке ошибки в этом окне можно перейти непосредственно к месту, в котором она возникла. Подробно отладка VI рассматривается в лекции 3, *Поиск ошибок и отладка VI*.

Самопроверка: короткий тест

Обратитесь к рисунку 2-33, чтобы ответить на следующие тестовые вопросы.

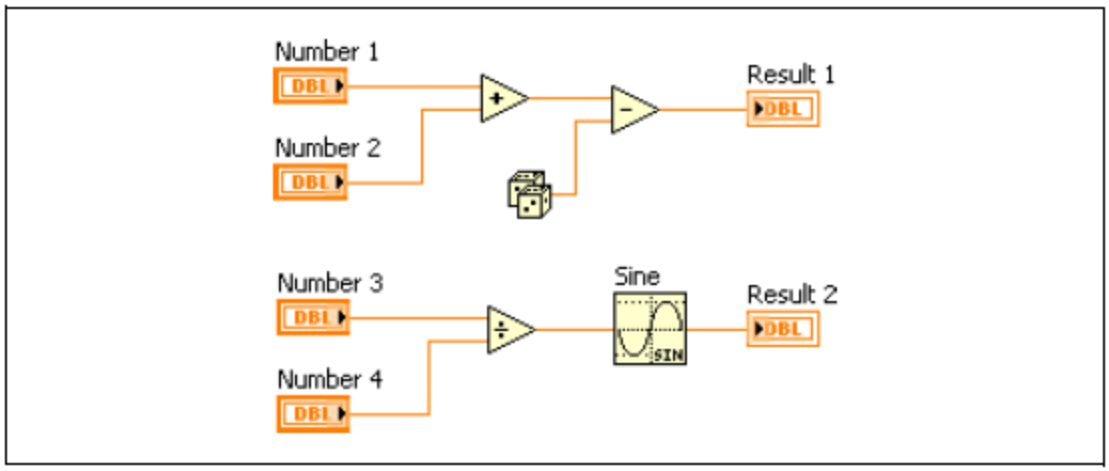
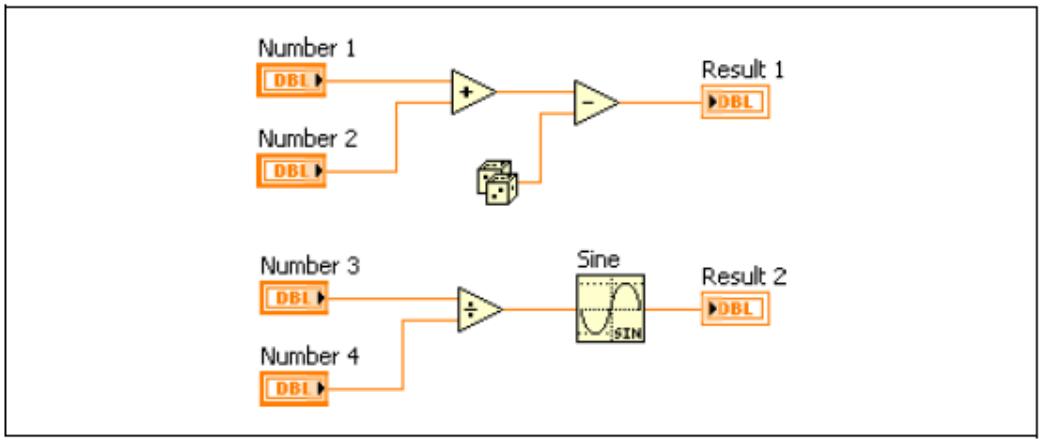


Рисунок 2-33. Вопросы о потоке данных

1. Какая функция выполняется первой: Add или Subtract?
   a. Add
   b. Subtract
   c. Неизвестно какая
2. Какая из функций выполняется первой: Sine или Divide?
   a. Sine
   b. Divide
   c. Неизвестно какая
3. Какая из функций выполняется первой: Random Number, Divide или Add?
   a. Random Number
   b. Divide
   c. Add
   d. Неизвестно какая
4. Какая из функций выполняется последней: Random Number, Subtract или Add?
   a. Random Number
   b. Subtract
   c. Add
   d. Неизвестно какая
5. Из каких трех частей состоит VI?
   a. Лицевой панели

b. Блок-диаграммы
c. Проекта
d. Иконки/панели подключения

Самопроверка: ответы



1. Какая функция выполняется первой: Add или Subtract?
   a. Add
   b. Subtract
   c. Неизвестно какая
2. Какая из функций выполняется первой: Sine или Divide?
   a. Sine
   b. Divide
   c. Неизвестно какая
3. Какая из функций выполняется первой: Random Number, Divide или Add?
   a. Random Number
   b. Divide
   c. Add
   d. Неизвестно какая
4. Какая из функций выполняется последней: Random Number, Subtract или Add?
   a. Random Number
   b. Subtract
   c. Add
   d. Неизвестно какая
5. Из каких трех частей состоит VI?
   a. Лицевой панели
   b. Блок-диаграммы
   c. Проекта
   d. Иконки/панели подключения

Заметки

3

3. Поиск ошибок и отладка VI

Чтобы запустить VI, вы должны соединить все subVI, функции и структуры, используя корректные типы данных для терминалов. Иногда VI выдает непонятные данные или ведет себя непредсказуемым образом. LabVIEW можно использовать для того, чтобы настроить режим выполнения VI и обнаружить проблемы, связанные с организацией блок-диаграммы или с обрабатываемыми данными.

План занятия

A. Справочные утилиты LabVIEW

B. Исправление ошибок в VI

C. Приемы отладки

D. Непонятные или непредвиденные данные

E. Контроль и обработка ошибок

A. Справочные утилиты LabVIEW

Чтобы облегчить создание и редактирование VI, пользуйтесь окном **Context Help**, справочной системой *LabVIEW Help* и поисковиком примеров NI Example Finder. Дополнительную информацию о LabVIEW можно найти в справочной системе *LabVIEW Help* и соответствующих руководствах.

Окно Context Help

Когда вы перемещаете курсор над объектами LabVIEW, основная информация о них выводится в окно **Context Help**. Чтобы открыть окно **Context Help**, выберите команду меню **Help»Show Context Help**, нажмите комбинацию клавиш <Ctrl-H> или щелкните мышью по кнопке **Show Context Help Window** на панели инструментов.

Если перемещать курсор над объектами лицевой панели и блок-диаграммы, в окне **Context Help** отображаются иконки subVI, функций, констант, элементов управления и индикаторов с проводниками, подсоединенными к каждому терминалу. Когда курсор перемещается по опциям в диалоговом окне, в окне **Context Help** отображается описание этих опций.

В окне **Context Help** наименования терминалов, обязательных для подключения, выделены полужирным шрифтом, терминалов, рекомендованных для подключения, – обычным шрифтом, а наименования дополнительных терминалов, которые подключать необязательно, шрифтом серого цвета. Если щелкнуть мышью по кнопке **Hide Optional Terminals and Full Path** в окне **Context Help**, наименования дополнительных терминалов отображаться не будут.

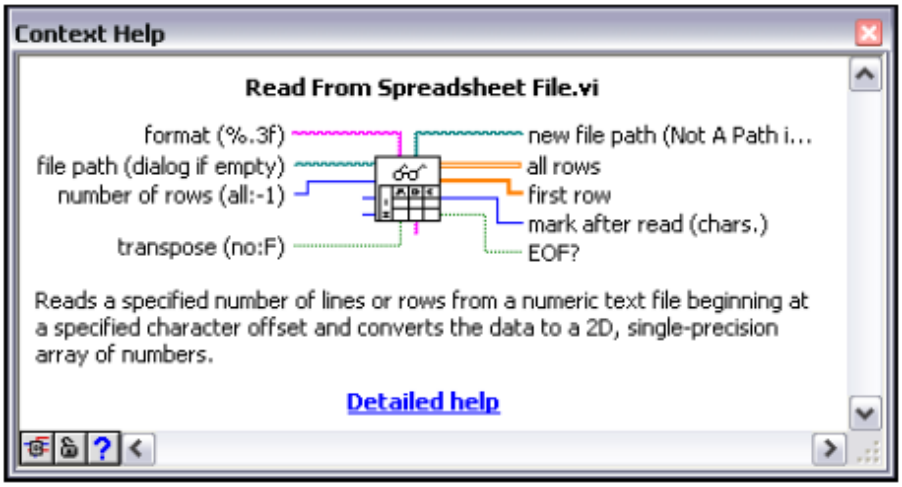


Рисунок 3-1. Окно контекстной справки

Щелкните по кнопке **Show Optional Terminals and Full Path**, расположенной в нижнем левом углу окна **Context Help**, чтобы сделать видимыми дополнительные терминалы панели подключения и полный путь к VI. Дополнительные терминалы показаны с укороченными проводниками, которые указывают на то, что есть еще соединения. В режиме полной

детализации отображаются все терминалы (рисунок 3-2).

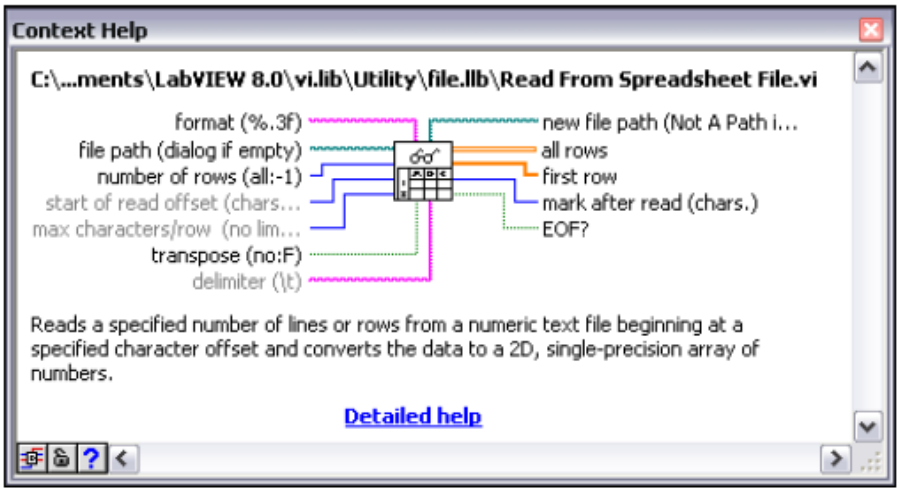


Рисунок 3-2. Окно полной контекстной справки

Чтобы зафиксировать текущее содержимое окна **Context Help**, щелкните по кнопке **Lock Context Help**. В этом случае при перемещении курсора на другой объект содержимое данного окна не изменяется. Чтобы разблокировать окно, щелкните еще раз по этой же кнопке. Блокировкой окна контекстной справки можно также управлять из меню **Help**.

Если для некоторого объекта, который описывается в окне **Context Help**, существует соответствующий раздел в справке *LabVIEW Help*, то в окне **Context Help** появляется ссылка синего цвета **Detailed help**. Кроме того, становится доступной кнопка **More Help**. Чтобы получить подробную информацию об интересующем объекте из справки *LabVIEW Help*, щелкните мышью по ссылке **Detailed help** или по кнопке **More Help**.

Справка LabVIEW Help

Открыть справку *LabVIEW Help* можно, щелкнув мышью по кнопке **More Help** в окне **Context Help**, либо выбрав команду меню **Help»Search the LabVIEW Help,** или щелкнув по ссылке синего цвета **Detailed Help** в окне **Context Help**. Вы можете также щелкнуть правой кнопкой мыши по какому-либо объекту и выбрать из контекстного меню команду **Help**.

Справка *LabVIEW Help* содержит детальные описания большинства палитр, меню, инструментов, VI и функций. В справку включены также пошаговые инструкции по использованию свойств LabVIEW. В *LabVIEW Help* есть ссылки на следующие ресурсы:

- *LabVIEW Documentation Resources*, где есть описания онлайн- и печатных документов, полезных новичкам и опытным пользователям, а также PDF-версии всех руководств по LabVIEW.
- Ресурсы по технической поддержке на Web-сайте компании National Instruments, в том числе NI Developer Zone, KnowledgeBase, Product Manuals Library.

Поисковик NI Example Finder

Поисковик NI Example Finder служит для просмотра или поиска примеров, инсталлированных на вашем компьютере или на странице NI Developer Zone по адресу `ni.com/zone`. Эти примеры демонстрируют, как использовать LabVIEW для решения широкого круга задач тестирования, измерений, управления и проектирования. Чтобы открыть NI Example Finder, выберите команду меню **Help»Find Examples** или щелкните по ссылке **Find Examples** в разделе **Examples** окна **Getting Started**.

Примеры могут подсказать, как использовать отдельные VI или функции. Чтобы открыть раздел справки с ссылками на примеры применения VI или функции, можно щелкнуть правой кнопкой мыши по иконке этого VI или функции на блок-диаграмме или на «пришпиленной» палитре, и выбрать из контекстного меню команду **Examples**. Вы можете модифицировать пример VI,приспосабливая его к вашему приложению, или скопировать и вставить в создаваемый вами VI фрагменты из одного и более примеров.

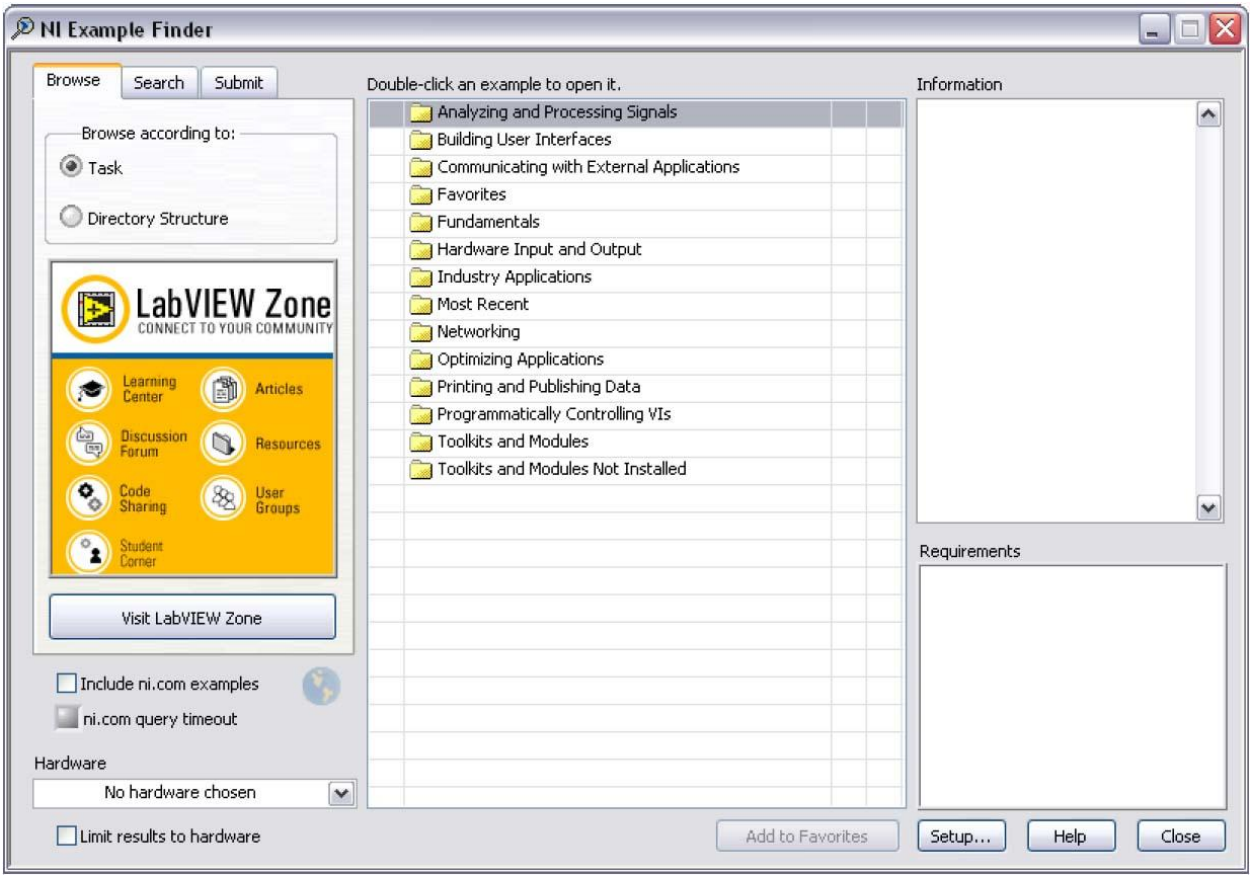


Рисунок 3-3. NI Example Finder

B. Исправление ошибок в VI

Если VI не работает, то он считается неисправным или неработоспособным. На кнопке **Run** появляется разрыв, если создаваемый и редактируемый вами VI содержит ошибки.

Если после завершения соединений на блок-диаграмме на кнопке сохраняется разрыв, VI считается неисправным и не может быть запущен.

Поиск причин неисправностей VI

Предупреждения не препятствуют запуску VI. Их назначение – помочь избежать потенциальных проблем в VI. Зато ошибки могут прервать работу VI. Прежде чем запустить VI, необходимо исправить все ошибки.

Чтобы узнать, почему VI неработоспособен, щелкните по кнопке **Run** с разорванной стрелкой или выберите команду меню **View»Error List**. В окне **Error list** приведены все ошибки. В разделе **Items with errors** содержатся имена всех элементов, находящихся в памяти, таких, как VI и библиотеки проекта, в которых есть ошибки. Если два или более элементов имеют одинаковые имена, то в этом разделе для каждого элемента указывается его конкретное применение. В разделе **errors and warnings** приведены ошибки и предупреждения для VI, выбранного в разделе **Items with errors**. В разделе **Details** приводятся описания ошибок и в некоторых случаях даются рекомендации по их устранению. Чтобы вывести на экран раздел справки *LabVIEW Help*, в котором дается детальное описание ошибки и, кроме того, пошаговые инструкции по ее исправлению, щелкните мышью по кнопке **Help**.

Щелкните один раз по кнопке **Show Error** или дважды щелкните по описанию ошибки, чтобы выделить область на блок-диаграмме или лицевой панели, в которой содержится ошибка.

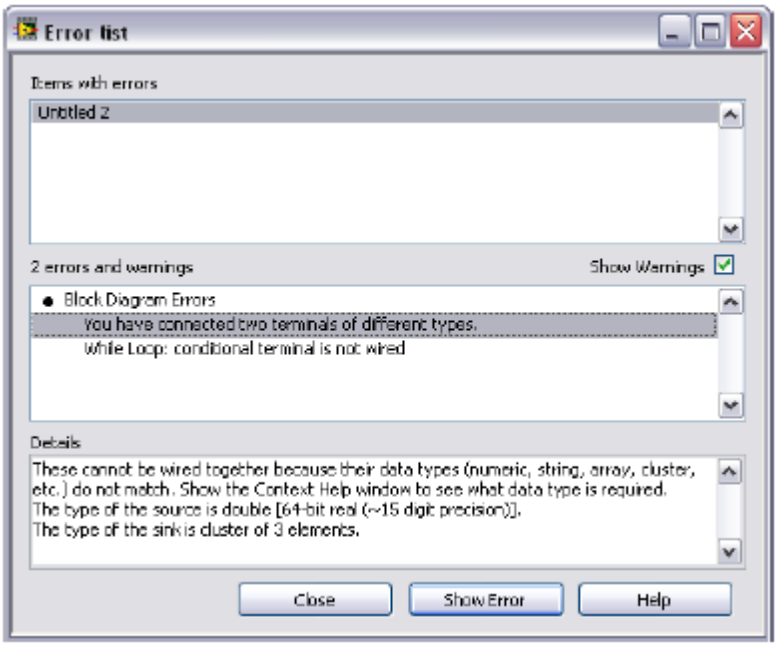


Рисунок 3-4. Пример диалогового окна Error List

Распространенные причины ошибок в VI

Ниже приведены распространенные причины возникновения ошибок в процессе редактирования VI:

- На блок-диаграмме есть оборванный проводник из-за несоответствия типов данных или какой-либо конец проводника никуда не подключен. Информация об исправлении подобных ошибок приведена в разделе *Correcting Broken Wires* справки *LabVIEW Help*.
- Обязательный для подключения терминал никуда не подключен. Информация о назначении обязательными для подключения входов приведена в разделе *Using Wires to Link Block Diagram Objects* справки *LabVIEW Help*.
- Неработоспособен subVI или его панель подключения редактировалась после того, как иконку subVI поместили на блок-диаграмму VI. Информация о subVI приведена в разделе *Creating SubVI* справки *LabVIEW Help*.

C. Техника отладки

Если VI работоспособен, но выдает при этом данные, отличные от ожидаемых, вы можете воспользоваться следующими приемами, чтобы выявить и исправить ошибки, связанные с данным VI или потоком данных на блок-диаграмме:

- Соединяйте входы и выходы error in и error out, расположенные внизу иконок большинства встраиваемых VI и функций. С помощью этих параметров обнаруживаются ошибки, возникающие в каждом из узлов блок-диаграммы, и указывается, в каком месте возникла ошибка. Вы можете пользоваться этими параметрами в создаваемых вами VI.
- Чтобы устранить все предупреждения, связанные с VI, выберите команду меню **View»Error List** и установите флажок **Show Warnings**, это позволит увидеть предупреждения. Затем выясните причины предупреждений и устраните их.
- Чтобы высветить весь маршрут следования проводника и убедиться, что он подключен к правильным терминалам, щелкните по проводнику трижды инструментом Positioning.
- Используйте окно **Context Help** для проверки значений параметров по умолчанию для каждой функции и subVI. Если рекомендуемые или необязательные для подключения входы никуда не подключены, VI и функции получают значения по умолчанию. Например, для входа типа Boolean, если он никуда не подключен, по умолчанию может быть установлено значение TRUE.
- При исправлении ошибок во всем VI используйте диалоговое окно **Find** для поиска subVI, текста и других объектов.
- Чтобы найти никуда не подключенные subVI, выберите команду меню **View»VI Hierarchy**. В отличие от никуда не подключенных функций, неподключенные VI не всегда генерируют ошибки, кроме случая, когда какой-либо вход сконфигурирован, как обязательный для подключения.

Если вы по ошибке поместили никуда не подключенный subVI на блок-диаграмму, он выполняется, пока работает блок-диаграмма. Следовательно, VI в этом случае может выполнять ненужные действия.

- Воспользуйтесь режимом подсветки выполнения, чтобы пронаблюдать продвижение данных по блок-диаграмме.
- Чтобы пронаблюдать каждое действие VI, выполняйте VI по шагам.
- Применяйте инструмент Probe для контроля промежуточных значений данных и выходов ошибок VI и функций, особенно выполняющих операции ввода-вывода.
- Щелкните по кнопке Retain Wire Values на панели инструментов блок-диаграммы, чтобы сохранять значения данных в проводниках для наблюдения их с помощью пробников. Это позволяет легко проверять значения данных, которые последними проходили через любой проводник.
- Для приостановки выполнения используйте контрольные точки, это позволит выполнять VI по шагам и вставлять пробники.
- Приостановите выполнение subVI, чтобы редактировать значения элементов управления и индикаторов, чтобы управлять количеством запусков subVI или вернуться к начальной точке выполнения subVI.
- Выясните, есть ли неопределенные данные, передаваемые через функцию или subVI. Такое часто случается с числами. Например, в какой-то момент VI может выполнить деление на ноль, которое возвращает результат Inf (бесконечность), в то время, как последующие функции или subVI ожидают числа.
- Если VI работает медленнее, чем ожидается, убедитесь в том, что вы отключили внутри subVI подсветку выполнения. Закройте также лицевые панели и блок-диаграммы, если вы их не используете, поскольку открытые окна могут повлиять на быстродействие.
- Проверьте формат представления элементов управления и индикации, чтобы увидеть, возможно ли переполнение из-за преобразования вещественного числа в целое, или целого – в более короткое целое. Например, вы можете подключить 16-разрядное целое к функции, которая воспринимает только 8-разрядные целые. Т.к. функция приводит 16-разрядное целое к 8-разрядному представлению, из-за этого возможна потеря данных.
- Проверьте, нет ли у вас случайно таких циклов For Loop, которые не выполняют ни одной итерации и обрабатывают пустые массивы.
- Проверьте, правильно ли проинициализированы сдвиговые регистры, кроме случая, когда, согласно вашим намерениям, они должны хранить данные, полученные в одной итерации цикла, для последующей.
- Проверьте порядок элементов кластера в точках, соответствующих источнику и приемнику данных. Во время редактирования LabVIEW обнаруживает несоответствие размеров кластеров и типов данных, однако несоответствие элементов одного и того же типа не обнаруживается.
- Проверьте порядок выполнения узлов.

- Убедитесь в том, что VI не содержит скрытых subVI. Вы случайно могли спрятать subVI, наложив один на другой, или уменьшив размер структуры, скрыв subVI под структурой.
- Сопоставьте список subVI, используемых VI, с результатами выполнения команд меню **View»Browse Relationships»This VI's SubVIs** и **View»Browse Relationships»Unopened SubVIs**, чтобы определить, имеются ли какие-нибудь лишние subVI. Откройте также окно VI Hierarchy, чтобы увидеть subVI, входящие в состав VI. Избежать некорректных результатов из-за скрытых VI поможет задание входов этих VI обязательными для подключения.

Подсветка выполнения

Щелчок по кнопке **Highlight Execution** позволяет увидеть анимированное выполнение кода блок-диаграммы.

Подсветка выполнения показывает продвижение данных по блок-диаграмме от одного узла к другому с помощью кружков, продвигающихся по проводникам. Используйте подсветку в сочетании с пошаговым выполнением, чтобы контролировать изменение данных от узла к узлу.

(Модуль MathScript RT) В узлах MathScript подсветка выполнения показывает переход от одной строки к другой с помощью синей стрелки, которая мерцает рядом со строкой, выполняющейся в настоящий момент.

Примечание: Подсветка выполнения существенно снижает скорость работы VI.

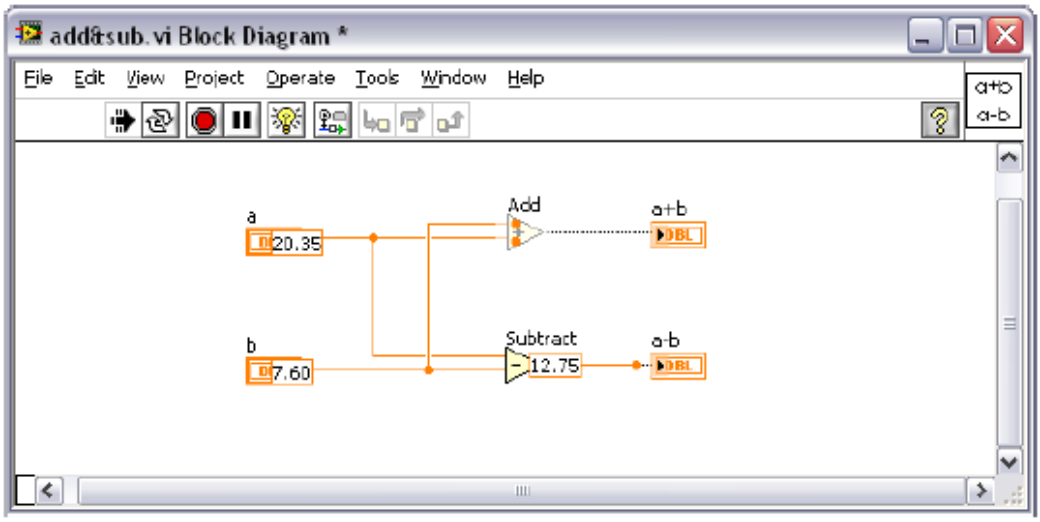


Рисунок 3-5. Пример использования подсветки выполнения

Пошаговое выполнение

Пошаговый режим выполнения VI позволяет видеть результат каждого действия на блок-диаграмме. Ниже изображены кнопки пошагового выполнения, которые влияют на работу VI или subVI только в пошаговом режиме.

Перейдите в пошаговый режим щелчком мыши по кнопке **Step Into** или **Step Over** на панели инструментов блок-диаграммы. Перемещайте курсор над кнопкой **Step Into**, **Step Over** или **Step Out**, чтобы увидеть подсказку с описанием предстоящего шага, который выполнится, если щелкнуть по этой кнопке. Вы можете выполнять subVI в пошаговом или в нормальном режиме.

Когда VI исполняется в пошаговом режиме, мерцание узлов показывает, что они готовы к выполнению. Если выполнять VI в пошаговом режиме с подсветкой (анимацией), на иконках выполняющихся subVI появляется специальный знак.

Инструменты Probe

Воспользуйтесь инструментом Probe (пробником) для контроля промежуточных значения данных, передаваемых по проводникам в процессе работы VI.

Инструментом Probe следует пользоваться, если у вас сложная блок-диаграмма с последовательностью операций, любая из которых может вернуть некорректные данные. Чтобы определить, есть ли некорректные данные и где они имеют место, применяйте инструмент Probe вместе с подсветкой выполнения, пошаговым выполнением и контрольными точками. Если данные доступны, пробник немедленно обновляется и выводит данных в окно **Probe Watch Window** в процессе выполнения с подсветкой, пошагового выполнения или когда вы приостановили VI в контрольной точке. Если выполнение приостанавливается на узле при включенном пошаговом режиме или из-за контрольной точки, вы можете также проконтролировать с помощью пробника значения данных, которые перед этим прошли по проводнику.

Типы пробников

Вы можете контролировать промежуточные значения данных в проводнике во время работы VI с помощью пробника общего назначения, индикатора из палитры **Controls**, специального (Supplied) пробника, адаптируемого пользователем специального пробника или путем создания нового пробника.

Примечание: (Модуль MathScript RT) Вы можете контролировать данные в скрипте узла MathScript Node в процессе работы VI с помощью пробника LabVIEW MathScript.

Пробник общего назначения

Воспользуйтесь пробником общего назначения, чтобы контролировать данные, которые передаются по проводнику. Для этого щелкните правой кнопкой мыши по проводнику и выберите из контекстного меню команду **Custom Probe»Generic Probe**.

На пробнике общего назначения данные просто отображаются. Вы не можете сконфигурировать его для селективного отображения данных.

При выборе команды **Probe** из контекстного меню проводника LabVIEW открывает пробник общего назначения, если ранее вы не использовали пользовательский или специальный пробник для соответствующего типа данных.

Вы можете отлаживать пользовательский пробник аналогично VI. Однако, вы не можете использовать этот пробник на его же блок-диаграмме или на блок-диаграмме любого из входящих в него subVI. При отладке таких пробников следует пользоваться пробником общего назначения.

Применение индикаторов для просмотра данных

Для наблюдения данных, передаваемых по проводнику, можно также пользоваться индикатором. Если вы, например, просматриваете числовые данные, то можете использовать графический индикатор типа chart. Щелкните правой кнопкой мыши по проводнику, выберите из контекстного меню команду **Custom Probe»Controls** и затем выберите нужный вам индикатор. Вы можете также щелкнуть мышью по иконке **Select A Control** на палитре **Controls** и выбрать любой пользовательский элемент управления или определитель типа данных, сохраненные на компьютере или в общей папке на сервере. LabVIEW воспринимает определитель типов данных, как пользовательский элемент управления, если их использовать для визуального контроля данных.

Если тип данных индикатора, который вы выбрали, не соответствует типу данных проводника, по которому вы сделали щелчок правой кнопкой, LabVIEW не помещает индикатор на проводник.

Специальные пробники

Специальные пробники — это VI, которые выводят на экран полную информацию о данных, передаваемых через проводник. Например, VI Refnum Probe возвращает информацию, которая содержит имя VI, путь к VI и ссылку на него в шестнадцатеричном коде. С помощью специального пробника вы можете также селективно отбирать идущие по проводнику данные. Используйте, например, пробник Error, чтобы получить статус, код,

источник и описание ошибки и задать, хотите ли вы установить условную контрольную точку по появлению ошибки или предупреждения.

Специальные пробники находятся в верхней строчке пункта контекстного меню **Custom Probe.** Чтобы выбрать специальный пробник, щелкните правой кнопкой мыши по проводнику и выберите из контекстного меню команду **Custom Probe.** В контекстном меню по нажатию правой кнопки появляются только пробники, которые соответствуют типу данных проводника.

Пример применения специальных пробников можно посмотреть в Using Supplied Probes VI из библиотеки `labview\examples\general\probes.llb`.

Пользовательские пробники

Чтобы создать пробник на основе существующего пробника или новый пробник, щелкните правой кнопкой мыши по проводнику и выберите из контекстного меню команду **Custom Probe»New**, а затем воспользуйтесь диалоговым окном **Create New Probe**. Пробник следует создавать, когда вы хотите контролировать данные в проводнике полнее, чем со стандартными пробниками LabVIEW. Тип данных создаваемого нового пробника соответствует типу данных проводника, по которому вы щелкнули правой кнопкой. При необходимости отредактировать уже созданный пробник, его необходимо открыть из папки, куда вы его сохранили.

После того, как вы выбрали из контекстного меню **Custom Probe**, перемещайтесь к нему с помощью кнопки палитры **Select a Control** или создайте новый пробник с помощью диалогового окна **Create New Probe.** Этот пробник становится пробником по умолчанию для соответствующего типа данных, и LabVIEW загружает этот пробник, когда вы щелкаете правой кнопкой по проводнику и из контекстного меню выбираете команду **Probe.** LabVIEW загружает только те пробники, которые точно соответствуют типу данных проводника, по которому вы щелкаете правой кнопкой. Следовательно, пробник для представления чисел с плавающей точкой двойной точности не сможет протестировать проводник типа «32-разрядное беззнаковое целое», даже несмотря на то, что в LabVIEW есть возможность преобразования данных.

Примечание: Если вы хотите, чтобы пользовательский пробник стал пробником по умолчанию для определенного типа данных, сохраните его в папке `user.lib\_probes\default`. Не сохраняйте пробники в папке `vi.lib\_probes`, поскольку LabVIEW перезаписывает соответствующие им файлы при обновлении или повторной инсталляции.

Контрольные точки

Воспользуйтесь инструментом Breakpoint, чтобы устанавливать контрольные точки на VI, узлы или проводники, на которых нужно приостановить выполнение VI.

Когда вы устанавливаете контрольную точку на проводник, выполнение VI приостанавливается после того, как данные прошли через проводник и кнопка **Pause** стала красного цвета. Установите контрольную точку на поле блок-диаграммы, чтобы пауза возникла после выполнения всех узлов блок-диаграммы. Рамка блок-диаграммы при этом окрашивается в красный цвет и начинает мерцать, указывая на положение контрольной точки.

При остановке VI в контрольной точке LabVIEW выносит блок-диаграмму на передний план и выделяет узел, проводник или строку скрипта, которые содержат контрольную точку. Если курсор перемещать над имеющейся контрольной точкой, черная область инструмента Breakpoint становится белого цвета.

При достижении контрольной точки VI приостанавливает выполнение, а кнопка **Pause** становится красного цвета. В этот момент вы можете предпринять следующие действия:

- Продолжить выполнение VI в пошаговом режиме с помощью соответствующих кнопок.
- Проконтролировать пробником промежуточные значения данных в проводниках.
- Изменить значения элементов управления на лицевой панели.
- Щелкнуть по кнопке Pause, чтобы продолжить выполнение VI до следующей контрольной точки или до тех пор, пока VI не завершит работу.

Приостановка выполнения

Выполнение subVI приостанавливают, чтобы отредактировать значения элементов управления и индикаторов, настроить количество запусков subVI перед возвратом в точку вызова, или вернуться к началу выполнения subVI. Вы можете настроить LabVIEW так, чтобы пауза возникала перед всеми или определенными вызовами subVI.

Чтобы приостановка происходила при всех вызовах subVI, откройте этот subVI и выберите команду меню **Operate»Suspend when Called**. В этом случае subVI автоматически приостанавливается, когда его вызывает другой VI. Если вы выберите упомянутый пункт меню в режиме пошаговой отладки, приостановка subVI произойдет не сразу, а только при его вызове.

Чтобы приостановить subVI при определенном вызове, щелкните правой кнопкой мыши по узлу subVI на блок-диаграмме и из контекстного меню выберите команду **SubVI Node Setup**. Чтобы приостановить выполнение только для данной копии subVI, установите в контекстном меню флажок **Suspend when called**.

В окне **VI Hierarchy**, которое выводится на экран с помощью команды меню **View»VI Hierarchy**, показывается, приостановлен ли VI при вызове subVI или на контрольной точке. Стрелка показывает, работает ли VI в обычном или в пошаговом режиме.

Символ паузы показывает, что VI приостановлен при вызове или на контрольной точке.

Зеленый символ паузы, или пустой черно-белый символ показывает, что VI приостановлен при вызове. Красный символ паузы, или сплошной черно-белый символ показывает, что в настоящий момент VI приостановлен на контрольной точке. Символ «восклицательный знак» показывает, что subVI приостановлен.

VI может быть одновременно приостановлен при вызове и на контрольной точке.

Определение текущей копии subVI

Когда вы приостанавливаете работу subVI, выпадающее меню **Call list** на панели инструментов отображает цепочку вызовов, начиная с VI самого верхнего уровня до данного subVI. Этот список не совпадает со списком, который отображается при выборе команды **Browse»This VI's Callers**, в котором приводятся все вызывающие VI независимо от того, работают ли они в настоящий момент. Если на блок-диаграмме содержится более одной копии subVI, воспользуйтесь меню **Call list** для определения текущей копии. Если выберете VI из меню **Call list**, откроется его блок-диаграмма и LabVIEW подсвечивает текущую копию subVI.

Просмотреть цепочку точек вызова от текущего VI до VI самого верхнего уровня можно также с помощью функции Call Chain.

D. Неопределенные или неожидаемые данные

Неопределенные данные, такие как `NaN` (не число) или `Inf` (бесконечность), не могут быть обработаны всеми последующими операциями. Эти символические значения, возвращаемые операциями над числами с плавающей точкой, являются признаком недопустимых вычислений или бессмысленных результатов:

- NaN (не число) служит для представления результата выполнения недопустимой операции над числами, например, взятия квадратного корня из отрицательного числа.
- Inf (бесконечность) служит для представления результата (в формате с плавающей точкой) выполнения допустимой операции над числами, например, деления числа на ноль.

LabVIEW не проверяет условия переполнения или потери значимости для целочисленных значений. Переполнение и потеря значимости для чисел с плавающей точкой соответствуют стандарту IEEE 754, *Standard for Binary Floating Point Arithmetic*.

Результаты выполнения операций с плавающей точкой наверняка могут принимать значения `NaN` и `Inf`. И если вы явно или неявно преобразуете

NaN и Inf в целочисленные или булевские значения, эти значения становятся бессмысленными. Например, деление числа 1 на ноль дает Inf. Преобразование Inf в 16-разрядное целое дает число 32767, которое кажется вполне нормальным.

Перед тем, как приводить данные к целочисленным типам, с помощью инструмента Probe проверьте допустимость промежуточных значений с плавающей точкой с помощью функции сравнения Not A Number/Path/Refnum?.

Не стоит полагаться на то, что специальные значения, такие как NaN, Inf или пустые массивы, помогут выяснить, что VI выдает неопределенные данные. Наоборот, убедитесь в том, что VI генерирует определенные данные, сделав так, чтобы VI выдавал сообщение об ошибке, если имеет место ситуация, когда, вероятно, он порождает неопределенные данные.

Если, например, вы создаете VI, в котором входной массив используется для авто-индексации цикла For Loop, определитесь с тем, что, по-вашему, должен делать VI, если этот массив окажется пустым: выдавать код ошибки; заменять определенными данными, которые породил бы сам цикл; или применять структуру Case, которая не допустит выполнения цикла For Loop, если массив пустой.

E. Контроль и обработка ошибок

Как бы хорошо вы не были уверены в созданном VI, вы не можете предугадать все проблемы, с которыми может столкнуться пользователь.

При отсутствии механизма выявления ошибок вы можете узнать только, что VI работает неправильно. Контроль ошибок позволяет получить информацию о том, почему и где возникли ошибки.

Автоматическая обработка ошибок

У каждой ошибки есть свой числовой код и соответствующее сообщение об ошибке.

По умолчанию LabVIEW автоматически обрабатывает любую ошибку, когда VI работает с приостановкой, высвечивая subVI или функцию, в которых возникла ошибка, и выводит на экран диалоговое окно с сообщением об ошибке.

Чтобы запретить автоматическую обработку ошибок для текущего VI, выберите команду меню **File»VI Properties** и выберите пункт **Execution** из выпадающего списка **Category**. Чтобы запретить автоматическую обработку ошибок для любого вновь создаваемого, пустого VI, выберите команду меню **Tools»Options** и выберите в списке **Category** пункт **Block Diagram**. Чтобы запретить автоматическую обработку ошибок для subVI или функции внутри VI, соедините их выход **error out** со входом **error in** другого subVI или функции или индикатором **error out**.

Обработка ошибок вручную

Вы можете выбрать другие методы обработки ошибок. Если, например, у VI ввода-вывода на блок-диаграмме истекает таймаут, то, возможно, вам не захочется останавливать работу всего приложения и выводить на экран диалоговое окно с сообщением об ошибке. Также, вероятно, вы пожелаете, чтобы VI делал повторные попытки ввода-вывода через определенный интервал времени. В LabVIEW можно выполнять такую обработку ошибок на блок-диаграмме VI.

Для обработки ошибок следует использовать VI и функции из палитры **Dialog & User Interface**, а также терминалы **error in** и **error out**, которые есть у большинства VI и функций. Если, например, в LabVIEW возникает ошибка, вы можете вывести сообщение об ошибке в диалоговых окнах различных видов. Для обнаружения и обработки ошибок применяйте механизмы обработки ошибок вместе с отладочными средствами.

VI и функции возвращают ошибки одним из двух способов: с помощью числовых кодов ошибок или с помощью кластера ошибок. Как правило, в функциях используются коды ошибок, а в VI — кластер ошибок, обычно для входов и выходов ошибок. Кластеры ошибок обеспечивают такую же функциональность, как и стандартные **error in** и **error out.**

При выполнении какой-либо операции ввода-вывода следует учитывать возможность возникновения ошибок. Почти все функции ввода-вывода возвращают информацию об ошибках. В VI необходимо предусмотреть проверку наличия ошибок, особенно для операций ввода-вывода (при работе с файлом, последовательным интерфейсом, измерительным прибором, устройством сбора данных и средством связи), и обеспечить соответствующий механизм обработки ошибок.

Для работы с ошибками используйте VI, функции и параметры обработки ошибок в LabVIEW. Если, например, в LabVIEW возникает ошибка, вы можете вывести в диалоговое окно сообщение об ошибке или зафиксировать ошибку программным путем, а затем удалить ее, соединив выход **error out** subVI или функции со входом **error in** VI Clear Errors. Механизмы обработки ошибки следует использовать совместно с отладочными средствами, чтобы обнаруживать и исправлять ошибки. Компания National Instruments настоятельно рекомендует применять механизмы обработки ошибок.

Кластеры ошибок

Чтобы создавать входы и выходы ошибок в subVI, используйте элементы управления и индикаторы кластеров ошибок.

Кластеры **error in** и **error out** состоят из следующих элементов информации:

- **status** — булевская переменная, которая выдает значение TRUE при обнаружении ошибки.

- **code** — 32-разрядное целое число со знаком, которое является числовым идентификатором ошибки. Ненулевой код ошибки в сочетании со значением FALSE элемента status оповещает скорее о предупреждении, чем об ошибке.
- **source** — строка, в которой указывается место возникновения ошибки.

Обработка ошибки в LabVIEW соответствует потоковой модели данных. Информация об ошибке может проходить через VI точно так же, как и поток данных. Передавайте с помощью проводников информацию об ошибке от начала VI до его конца. В конце проектируемого VI включите VI обработки ошибок, чтобы выяснить, работает ли VI без ошибок. Используйте кластеры **error in** и **error out** в каждом используемом VI или проектируйте VI так, чтобы информация об ошибках проходила через него насквозь.

В процессе выполнения VI LabVIEW проверяет наличие ошибок в каждом исполняемом узле. Если LabVIEW не обнаруживаетошибок, узел работает нормально. Если LabVIEW обнаруживает ошибку, узел передает ее следующему узлу, не выполняя ту часть кода, что содержит ошибку. Следующий узел делает то же самое и т.д. В конце выполнения всей этой цепочки LabVIEW выдает сообщение об ошибке.

Объяснение ошибок

При обнаружении ошибки щелкните правой кнопкой мыши внутри кластера и выберите из контекстного меню команду **Explain Error**, чтобы открыть диалоговое окно **Explain Error**, в котором содержится информация об ошибке. Если в VI обнаруживаются не ошибки, а предупреждения, то можно воспользоваться командой контекстного меню **Explain Warning**.

К диалоговому окну **Explain Error** можно также получить доступ из меню **Help»Explain Error**.

Самопроверка: короткий тест

1. Как запретить автоматическую обработку ошибок?
   a. Разрешить выполнение с подсветкой.
   b. Соединить кластер error out одного subVI с кластером error in другого subVI.
   c. Установить флажок **Show Warnings** в окне **Error List**.
2. Какие из следующих компонентов содержатся в кластере ошибок?
   a. Status: Boolean
   b. Error: String
   c. Code: 32-bit integer
   d. Source: String

Самопроверка: ответы

1. Как запретить автоматическую обработку ошибок?
   a. Разрешить выполнение с подсветкой.
   b. Соединить кластер error out одного subVI с кластером error in другого subVI.
   c. Установить флажок **Show Warnings** в окне **Error List**.
2. Какие из нижеследующих компонентов содержатся в кластере ошибок?
   a. Status: Boolean
   b. Error: String
   c. Code: 32-bit integer
   d. Source: String

Заметки

4

4. Реализация VI

На этом занятии вы научитесь программировать в LabVIEW. Навыки по программированию в LabVIEW включают в себя проектирование пользовательского интерфейса, выбор типов данных, документирование программного кода, применение циклических структур, таких, как While Loop и For Loop, добавление программных задержек, отображение данных в виде графика, а также принятие решения в программном коде с помощью Case-структуры.

План занятия

A. Проектирование лицевой панели
B. Типы данных в LabVIEW
C. Документирование программного кода
D. Циклы While
E. Циклы For
F. Задание времени выполнения VI
G. Передача данных от итерации к итерации
H. Вывод данных на графики
I. Case-структуры

A. Проектирование лицевой панели

На этапе проектирования программного обеспечения вы определяете входы и выходы задачи, что является непосредственным началом проектирования лицевой панели.

Исходные данные для задачи можно получить следующими способами:

- в процессе, например, измерения с помощью устройства сбора данных или мультиметра
- чтение данных непосредственно из файла
- манипулирование органами управления

Выходные данные задачи вы можете вывести на графические индикаторы (Graph или Chart), на светодиодные индикаторы (LED) или записать выходные данные в файл. Кроме того, данные можно вывести в некоторое устройство путем генерации сигнала. Позднее в нашем курсе будут занятия по сбору данных, генерации сигнала и записи в файл.

Проектирование органов управления и индикаторов

Выбирая элементы управления и индикации, удостоверьтесь в том, что они соответствуют решаемой задаче. Если, например, вы хотите задавать частоту синусоидального сигнала, то выбирайте элемент управления типа «лимб» (Dial), а если хотите отображать значения температуры, то нужно выбрать индикатор типа «термометр» (Thermometer).

Метки и заголовки

Удостоверьтесь в том, что элементы управления и индикации снабжены понятными метками. Эти метки помогают пользователям определять назначение каждого элемента управления и индикатора. Кроме того, понятные обозначения помогают документировать программный код блок-диаграммы. Метки элементов управления и индикаторов соответствуют именам терминалов на блок-диаграмме.

С помощью заголовков создается описание элемента управления на лицевой панели. Заголовки на блок-диаграмме не появляются, но их использование позволяет документировать пользовательский интерфейс, не загромождая блок-диаграмму дополнительным текстом. Пусть, например, на метеостанции вам необходимо задать верхнюю границу температуры. Если температура превышает это значение, метеостанция выдает предупреждение о тепловом ударе. В этом случае элемент управления можно назвать `Upper Temperature Limit (Celsius)`. Однако такая метка займет лишнее место на блок-диаграмме. Вместо этого используйте заголовок элемента управления `Upper Temperature Limit (Celsius)`, а на блок-диаграмме создайте короткую метку, например, `Upper Temp`.

Дополнительные возможности органов управления и индикации

Для органов управления можно задать значения по умолчанию. На рисунке 4-1 значение по умолчанию элемента управления равно 35 °C. Элементу управления можно присвоить подходящее значение по умолчанию на случай, если пользователь не введет другое значение в процессе работы. Чтобы установить значение по умолчанию, выполните следующие действия:

1. Введите в орган управления желаемое значение.
2. Щелкните правой кнопкой по элементу управления и выберите из контекстного меню команду **Data Operations»Make Current Value Default**.

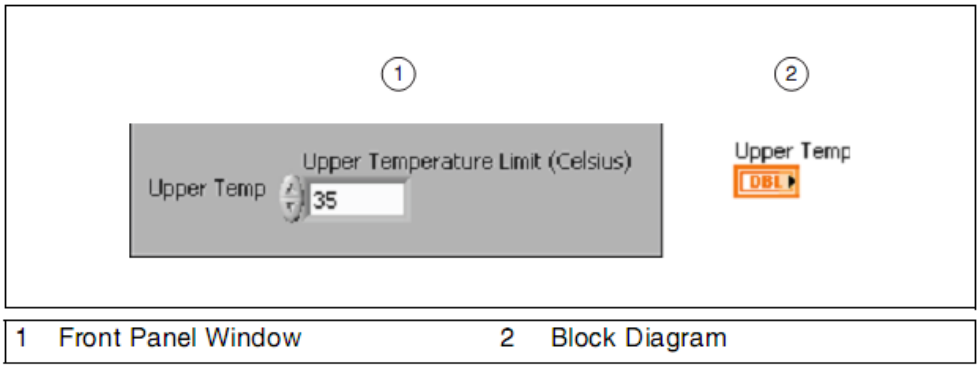


Рисунок 4-1. Установка значения по умолчанию

Вы можете скрывать и делать видимыми отдельные части элементов управления и индикации. Например, на рисунке 4-2 одновременно видны и заголовок, и метка. Однако вам, например, нужно, чтобы был виден только заголовок. Чтобы скрыть метку, щелкните правой кнопкой по элементу управления и выберите команду **Visible Items»Label** (рисунок 4-2).

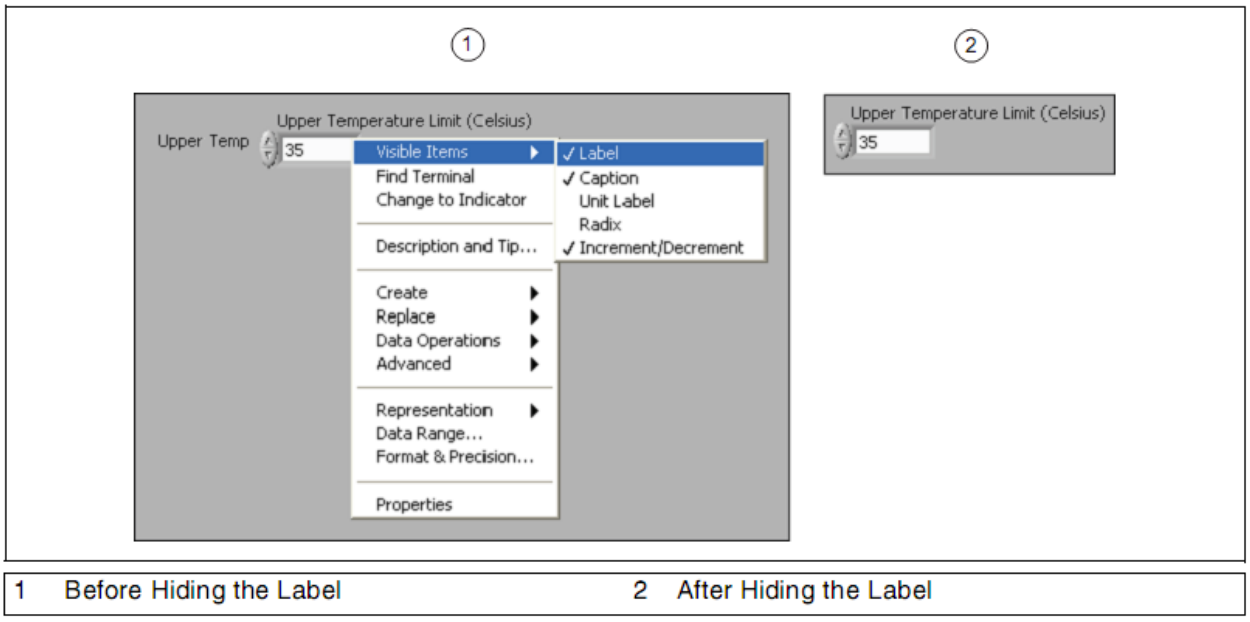


Рисунок 4-2. Делаем метку на лицевой панели невидимой

Использование цветов

Правильное использование цветов способствует улучшению внешнего вида и функциональности пользовательского интерфейса. Но слишком большое количество цветов может привести к цветовой дисгармонии, из-за чего пользовательский интерфейс будет выглядеть слишком перегруженным и сбивающим с толку.

В LabVIEW есть специальное окно, с помощью которого можно выбрать нужные цвета. Чтобы открыть это окно, выберите инструмент Coloring и щелкните правой кнопкой мыши по объекту или рабочей области. Верхняя часть окна выбора цвета содержит спектр шкалы уровней серого цвета и прямоугольник, с помощью которого можно создавать прозрачные объекты. Второй спектр содержит приглушенные цвета, которые хорошо подходят для фона и объектов лицевой панели. Третий спектр содержит цвета, которые хорошо подходят для выделения. Перемещение курсора по вертикали от цветов фона до цветов выделения помогает выбрать цвета выделения, которые будут сочетаться с конкретным цветом фона.

При подборе цветов полезны следующие советы:

- Используйте цветовую гамму LabVIEW по умолчанию. Среда LabVIEW заменяет цвета аналогично тому, как она заменяет шрифты. Если один из цветов VI недоступен, LabVIEW заменяет его наиболее подходящим. Вы можете также пользоваться системными цветами, чтобы согласовать внешний вид лицевой панели с системными цветами любого компьютера, на котором запускается данный VI.
- Начинайте с серой цветовой схемы. Выберите один или два оттенка серого и цвета для выделения, которые хорошо контрастируют с цветом фона.
- Обдуманно добавляйте цвета выделения – на графиках, кнопках прерывания и, возможно, указателях бегунков. Для мелких объектов требуются более светлые и контрастные цвета, чем для более крупных объектов.
- Используйте в большей степени различия в контрасте, чем различия в цветах. Дальтоникам кажется сложным различить объекты, когда различия в цвете больше, чем различия в контрасте.
- Группируйте объекты с интервалами между ними вместо того, чтобы подбирать цвета.
- Подходящими примерами для изучения техники использования цветовых гамм, являются панели «реальных» автономных измерительных приборов, карты и журналы.
- Если вы хотите, чтобы на лицевой панели использовались системные цвета, выберите объекты из раздела **System** палитры **Controls.**

Расстановка интервалов и выравнивание

Расстановка интервалов и выравнивание, вероятно, наиболее важные приемы группировки и разделения объектов. Чем больше объектов видно на одной линии, тем более понятно и более связно организованными они

выглядят. Если элементы выстроены в ряд, то такой порядок естественно воспринимается зрением слева направо или сверху вниз. Это относится и к направлению шрифта. Несмотря на то, что у некоторых народов элементы принято просматривать справа налево, почти все воспринимают порядок элементов сверху вниз.

Следуйте приведенным ниже рекомендациям при разработке лицевой панели, чтобы она служила в качестве пользовательского интерфейса:

- Не размещайте объекты слишком близко друг к другу. Чтобы лицевая панель легче воспринималась, оставьте немного свободного места между объектами, которое, кроме того, предотвращает случайный щелчок мышью по ненужному элементу управления или кнопке.
- Не размещайте одни объекты поверх других объектов. Даже частичное перекрытие элемента управления или индикатора меткой или другим объектом замедляет обновление экрана и может привести к тому, что этот орган управления или индикатор будет мерцать.
- Применяйте разделительные линии между разделами меню (рисунок 4-3), чтобы ускорить поиск нужных элементов в разделах и более тесно их сгруппировать.

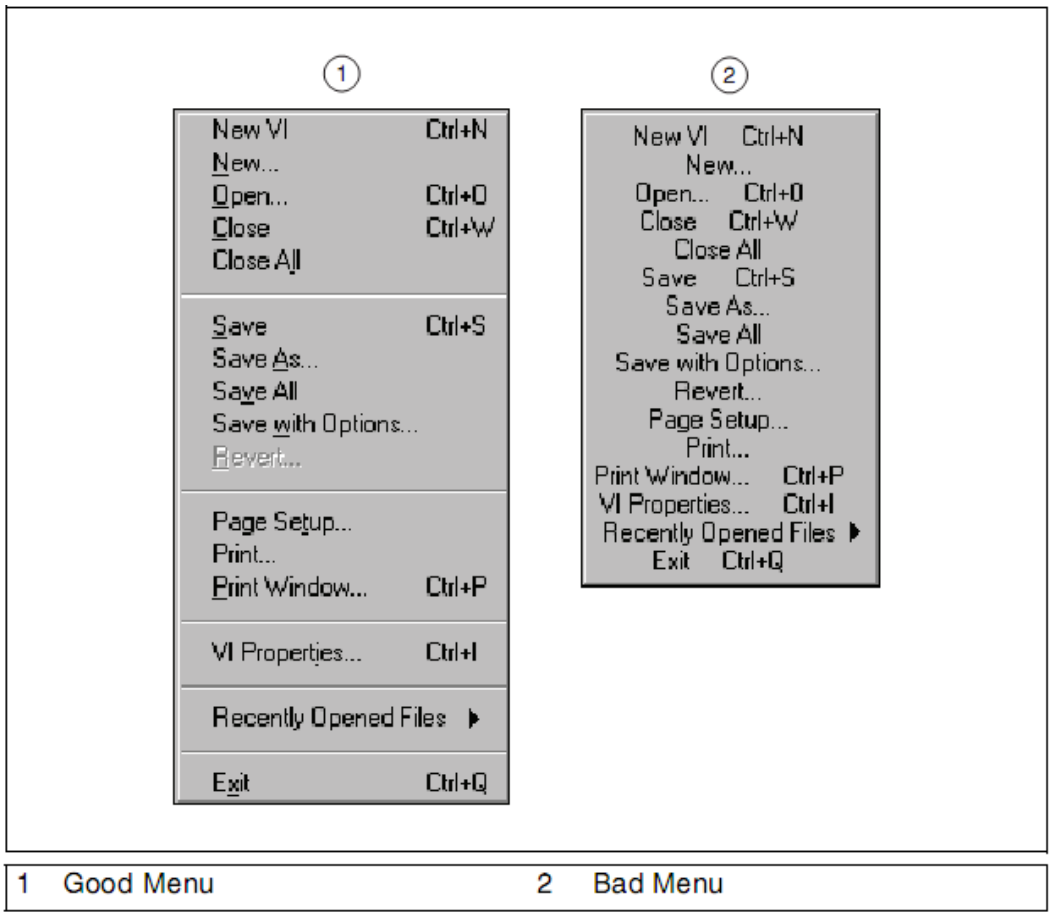


Рисунок 4-3. Пример хорошо и плохо оформленных меню

Текст и шрифты

Текст легче читается, и информация лучше воспринимается, когда выводится на экран в упорядоченном виде. Поэтому пользуйтесь шрифтами

LabVIEW по умолчанию. Например, LabVIEW, по умолчанию, определяет свои встроенные шрифты, как системные. Когда вы переносите VI между платформами, LabVIEW автоматически обновляет свои встроенные шрифты так, чтобы они соответствовали системному шрифту по умолчанию на текущей платформе. Кроме того, если вы пытаетесь открыть VI, в котором применяется недоступный шрифт, LabVIEW заменяет его наиболее подходящим. Например, если вы пытаетесь открыть VI, в котором используется шрифт Arial, на компьютере, на котором нет такого шрифта, LabVIEW заменяет его похожим шрифтом, который инсталлирован на данном компьютере.

Применение слишком большого количества стилей шрифтов может привести к тому, что лицевая панель будет выглядеть загроможденной и неорганизованной. Вместо этого используйте два или три размера одного и того же шрифта. Слова, напечатанные шрифтом Serifs, хорошо распознаются на расстоянии. Если вы применяете более чем один размер шрифта, убедитесь в том, что эти размеры различаются существенно. В противном случае это может плохо интерпретироваться. Если вы используете два различных шрифта, также убедитесь, что они различимы.

Операторские станции промышленного назначения следует проектировать с использованием более контрастного пользовательского интерфейса и с более крупными шрифтами. Нормальные шрифты могут стать трудночитаемыми из-за яркого света при искусственном освещении и при необходимости читать информацию на расстоянии. Также нужно помнить о том, что для сенсорных экранов требуются шрифты больших размеров, и между элементами выбора необходимо оставлять больше свободного места.

Примечание: Если нужных шрифтов нет на целевом компьютере, замена шрифтов может привести к тому, что пользовательский интерфейс будет выглядеть искаженным.

Подсказки и инструменты пользовательского интерфейса

В число встроенных инструментов LabVIEW, применяемых при создании лицевых панелей с дружественным интерфейсом, входят системные элементы управления, закладки, элементы декорации, меню и автоматическое изменение размеров объектов лицевой панели.

Системные элементы управления

Распространенный прием работы с пользовательским интерфейсом заключается в том, что для взаимодействия с пользователем в определенные моменты времени на экране появляются диалоговые окна. Вы можете сделать VI так, чтобы он выглядел, как диалоговое окно. Для этого выберите команду меню **File»VI Properties**, категорию **Window Appearance**, и, наконец, опцию **Dialog**.

В создаваемых вами диалоговых окнах пользуйтесь системными элементами управления и индикаторами, расположенными в палитре

System. Системные элементы управления изменяют свой внешний вид в зависимости от того, на какой платформе запускается VI. Когда вы запускаете VI на другой платформе, цвет и внешний вид системных элементов управления подгоняется под стандартные элементы управления диалогового окна именно этой платформы.

Как правило, системные элементы управления игнорируют все цвета, кроме четко распознаваемых. Если вы встраиваете график или несистемный элемент управления в окна лицевой панели, согласуйте их с этими окнами, скрыв некоторые рамки или выбрав цвета, похожие на системные.

Закладки

«Реальные» измерительные приборы обычно имеют хороший пользовательский интерфейс, и их принципы проектирования следует в большой степени заимствовать, применяя, в то же время, где это допустимо, элементы управления меньших размеров или более эффективные, такие как закладки или кольцевые списки. Закладки служат для размещения перекрывающихся элементов управления и индикаторов на меньшей площади лицевой панели.

Чтобы добавить страницы в элемент управления закладками, щелкните по нему правой кнопкой мыши и выберите из контекстного меню команду **Add Page Before** или **Add Page After**. Затем с помощью инструмента Labeling измените метки на закладках и разместите объекты лицевой панели на соответствующих страницах. Терминалы этих объектов доступны на блок-диаграмме, как терминалы любых объектов лицевой панели (кроме декоративных элементов).

Вы можете подключить к селектору Case-структуры терминал элемента управления закладками (перечислительного типа), чтобы создать более аккуратные блок-диаграммы. Таким способом каждая страница закладки ассоциируется с соответствующей суб-диаграммой Case-структуры. Терминалы элементов управления и индикаторов из каждой страницы помещаются в суб-диаграммы Case-структуры, так же, как узлы и проводники блок-диаграммы, ассоциируемые с этими терминалами.

Элементы декорации

Чтобы сгруппировать или разделить объекты с помощью прямоугольников, линий или стрелок, воспользуйтесь элементами декорации, расположенными в палитре **Decorations**.

Меню

Используйте пользовательские меню, чтобы на сравнительно малой площади представить функциональность лицевой панели более упорядочено. Это позволяет сэкономить место на лицевой панели, освобождая пространство для особо важных элементов управления и индикаторов, элементов, предназначенных для начинающих пользователей, элементов, необходимых для повышения производительности труда, а также

элементов, которые не очень хорошо вписываются в меню. Элементам меню можно также назначить «горячие» клавиши».

Чтобы создать контекстное меню для объектов лицевой панели, щелкните правой кнопкой мыши по объекту и выберите команду **Advanced»Run-Time Shortcut Menu»Edit.** Чтобы создать пользовательское контекстное меню для VI, выберите команду меню **Edit»Run-Time Menu**.

Автоматическое изменение размеров объектов лицевой панели

Воспользуйтесь опциями **Window Size** в меню **File»VI Properties**, чтобы установить минимальный размер окна, сохранить пропорциональность окна при его изменениях, а также установить два различных режима изменения размеров объектов лицевой панели. Когда вы проектируете VI, следует задуматься над тем, сможет ли лицевая панель отображаться на экранах компьютеров с разным разрешениям. Чтобы пропорции лицевой панели устанавливались в соответствии с разрешением экрана, выберите команду меню **File»VI Properties**, затем выберите пункт **Window Size** в выпадающем списке **Category**, и установите флажок **Maintain proportions of window for different monitor resolutions**.

B. Типы данных в LabVIEW

Для представления данных в компьютере существует множество различных типов данных. На втором занятии, *Ориентация в LabVIEW,* вы узнали о числовых, булевских и строковых типах данных. Используются также и другие типы данных – перечислительный, динамический и т.д. Даже среди числовых типов данных различают, например, целые числа или дробные числа.

Терминалы

Терминалы на блок-диаграмме визуально дают пользователю некоторую информацию о соответствующих им типах данных. Например, на рисунке 4-4 тип терминала **Height (cm)** — числовой двойной точности с плавающей точкой, на что указывает оранжевый цвет терминала и нанесенный на него текст **DBL**.

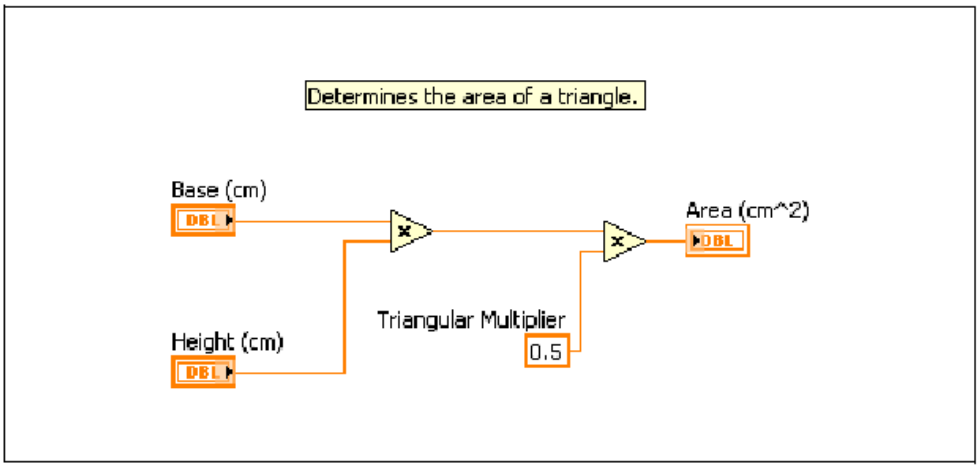


Рисунок 4-4. Пример типа данных терминала

Числовые типы данных

С помощью числовых типов данных представляются числа разных форматов. Чтобы изменить формат представления числа, щелкните правой кнопкой мыши по элементу управления, индикатору или константе и выберите команду **Representation** (рисунок 4-5).

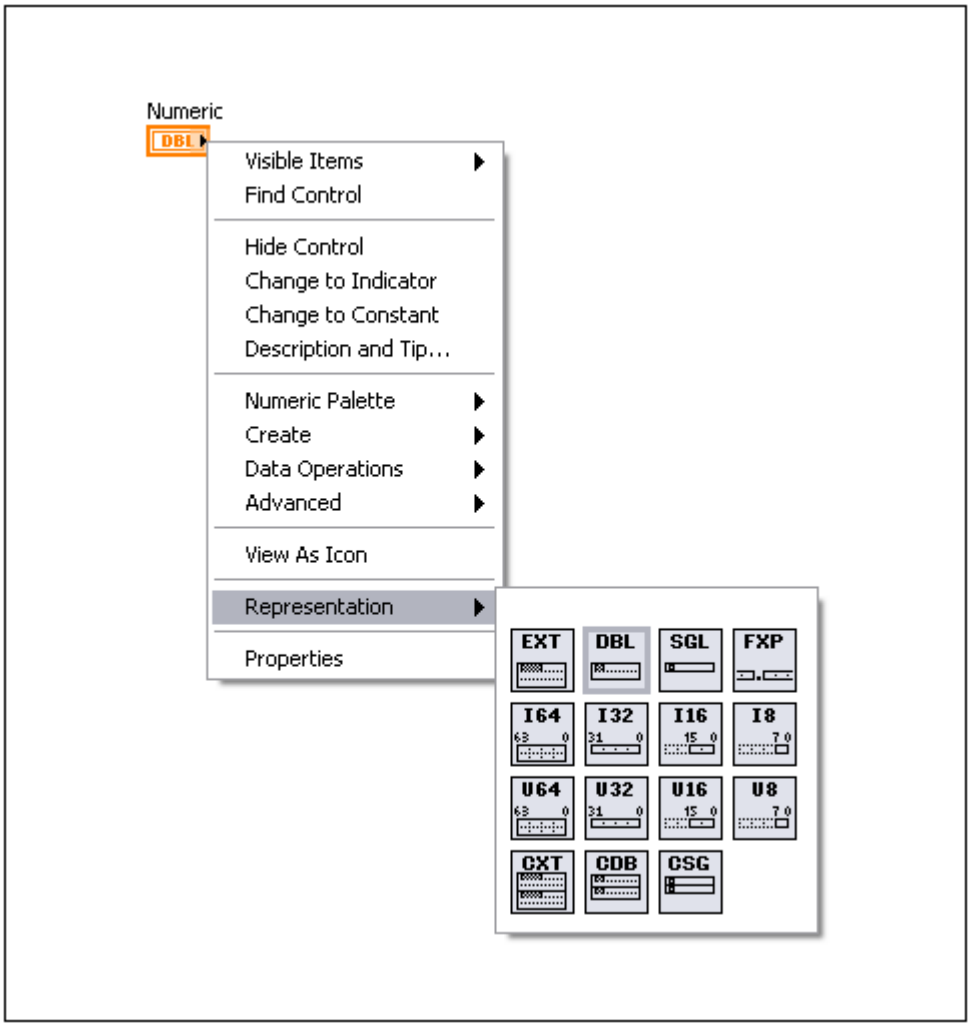


Рисунок 4-5. Форматы представления чисел

Когда на входы функции поступают данные двух или более различных числовых форматов, то, как правило, функция возвращает данные более «длинного» или более «широкого» формата. Перед тем, как выполниться, функции приводят более короткие форматы данных к более широким, а LabVIEW помещает точки приведения на терминалах, где имеют место преобразования типов.

Числовой тип данных включает в себя следующие разновидности: числа с плавающей точкой, целочисленные со знаком, целочисленные без знака и комплексные числа.

Числа с плавающей точкой

Числа с плавающей точкой служат для представления дробных чисел. В LabVIEW числа с плавающей точкой представляются с помощью оранжевого цвета.

Single-precision (SGL) — числа с плавающей точкой в 32-разрядном формате IEEE с одинарной точностью. Такие числа следует использовать для экономии памяти и чтобы избежать переполнения диапазона.

Double precision (DBL) — числа с плавающей точкой в 64-разрядном формате IEEE с двойной точностью. Этот формат используется по умолчанию для числовых объектов. В большинстве случаев используются числа двойной точности с плавающей точкой.

Extended-precision (EXT) — числа с повышенной точностью. Размер памяти и точность представления чисел с повышенной точностью изменяется в зависимости от платформы. В операционной системе Windows эти числа представляются в 80-разрядном IEEE формате с повышенной точностью.

Числа с фиксированной точкой

Числа с фиксированной точкой — это числовой тип данных, который служит для представления множества рациональных чисел с помощью двоичных разрядов (бит). В отличие от типа чисел с плавающей точкой, который позволяет изменять общее количество бит для представления чисел в LabVIEW, числа с фиксированной точкой можно конфигурировать таким образом, чтобы под них всегда отводилось определенное количество бит. Аппаратные средства и целевые устройства, которые могут хранить и обрабатывать только данные с ограниченным или фиксированным количеством бит, работают с числами такого типа. Для чисел с фиксированной точкой можно задать диапазон значений и точность представления.

Примечание: Чтобы представить рациональное число, используя тип данных с фиксированной точкой, знаменатель этого числа должен быть степенью числа 2, поскольку основание двоичной системы счисления равно двум.

Тип данных с фиксированной точкой следует использовать тогда, когда вам не требуется динамическая функциональность чисел с плавающей точкой, или если вы хотите работать с целевым устройством, которое не поддерживает арифметику с плавающей точкой, например, FPGA.

Задайте способ кодирования, длину слова и длину целой части слова, если вы хотите, чтобы число с фиксированной точкой представлялось определенным количеством бит.

Способ кодирования — двоичное кодирование числа с фиксированной точкой. Вы можете выбрать кодирование со знаком или без знака. Если вы выбираете кодирование со знаком, первый бит всегда служит для представления знака, а затем следуют биты, представляющие данные.

Длина слова — общее количество бит, которое используется в LabVIEW для представления всех возможных значений чисел с фиксированной точкой. LabVIEW воспринимает максимальную длину слова в 64 бит. В некоторых целевых устройствах длина слова может быть меньше. Если вы откроете VI для целевого устройства, и в этом VI используются данные с фиксированной точкой с большими значениями длины слова, чем данное устройство может воспринять, в таком VI будут разорванные проводники. Чтобы узнать максимальную длину слова для конкретной целевой платформы, обратитесь к документации по этой платформе.

Длина целой части — количество бит для представления целой части числа, которые LabVIEW использует для представления всех возможных значений данных с фиксированной точкой, или, при заданной начальной позиции слева или справа старшего значащего бита, для представления количества бит, на которое надо сдвинуть двоичную точку, чтобы достичь самого старшего бита. Длина целой части может быть больше, чем длина слова, а также может быть положительной или отрицательной.

Целые числа

Целочисленный тип данных служит для представления целых чисел. Целые числа со знаком могут быть положительными или отрицательными. Если вы уверены в том, что целое число принимает всегда положительные значения, используйте целочисленные типы данных без знака. В LabVIEW целые числа обозначаются синим цветом.

Когда LabVIEW преобразует числа с плавающей точкой в целые числа, VI округляет их до ближайшего четного целого. Например, число 2.5 округляется до 2, а 3.5 округляется до 4.

Byte (I8) — целые числа этого типа занимают в памяти один байт (8 бит) и принимают значения в диапазоне от −128 до 127.

Word (I16) — целые числа этого типа занимают в памяти одно слово (16 бит) памяти и принимают значения в диапазоне –32 768 до 32 767.

Long (I32) — длинные целые числа этого типа занимают в памяти 32 бита и принимают значения в диапазоне −2 147 483 648 до 2 147 483 647. В

большинстве случаев предпочтительнее использовать 32-разрядные целые числа.

Quad (I64) — целые числа учетверенной длины 64 бита и принимают значения в диапазоне от −1e19 до 1e19.

Byte (U8) — целые числа без знака этого типа занимают в памяти один байт (8 бит) памяти и принимают значения в диапазоне от 0 до 255.

Word (U16) — целые числа без знака этого типа занимают в памяти одно слово (16 бит) памяти и принимают значения в диапазоне от 0 до 65 535.

Long (U32) — длинные целые числа без знака этого типа занимают в памяти 32 бита памяти и принимают значения в диапазоне от 0 до 4 294 967 295.

Quad (U64) — целые числа без знака учетверенной длины занимают в памяти 64 бита памяти и принимают значения в диапазоне от 0 до 2e19.

Комплексные числа

Комплексные числа представляются в виде двух значениями, связанными между собой в памяти — одно из них служит для представления действительной части, а другое — для представления мнимой части числа. Поскольку в LabVIEW комплексные числа являются разновидностью чисел с плавающей точкой, они также обозначаются оранжевым цветом.

Complex Single (CSG) — комплексные числа с одинарной точностью состоят из действительной и мнимой части в 32-разрядном IEEE формате с плавающей точкой с одинарной точностью.

Complex Double (CDB) — комплексные числа с двойной точностью состоят из действительной и мнимой части в 64-разрядном IEEE формате с плавающей точкой с двойной точностью.

Complex Extended (CXT) — комплексные числа с повышенной точностью состоят из действительной и мнимой части в IEEE формате с плавающей точкой с повышенной точностью. Размер памяти и точность представления чисел с повышенной точностью изменяется в зависимости от платформы. В операционной системе Windows эти числа представляются в 80-разрядном IEEE формате с повышенной точностью.

Булевские данные

В LabVIEW данные булевского типа хранятся в виде 8-разрядных чисел. Если 8-разрядное число равно нулю, то значение булевской переменной равно FALSE. Любое ненулевое значение служит для представления значения TRUE. В LabVIEW для представления данных булевского типа используется зеленый цвет.

Булевские данные имеют связанные с ними механические действия. К двум основным действиям относятся фиксация и переключение. Вы можете выбрать один из следующих вариантов поведения кнопки:

- **Switch when pressed** — изменяет значение элемента управления при каждом щелчке по нему инструментом Operating. Частота, с которой VI считывает состояние элемента управления, не влияет на его поведение.
- **Switch when released** — изменяет значение элемента управления после щелчка по элементу управления в момент освобождения кнопки мыши. Частота, с которой VI считывает состояние органа управления, не влияет на его поведение.
- **Switch until released** — изменяет значение элемента управления после щелчка по нему мышью и сохраняет новое значение до тех пор, пока кнопку мыши не будет отпущена. В этот момент элемент управления возвращается к своему состоянию по умолчанию, аналогично тому, как это происходит при работе дверного звонка. Частота, с которой VI считывает состояние органа управления, не влияет на его поведение. Такой вариант поведения нельзя выбрать для радио-кнопок.
- **Latch when pressed** — изменяет значение элемента управления после щелчка по нему мышью, и сохраняет новое значение до тех пор, пока VI его однократно не считает. В этот момент элемент управления возвращается к своему состоянию по умолчанию, даже если вы все еще удерживаете кнопку мыши нажатой. Такое поведение аналогично работе прерывателя и является полезным для остановки цикла While или для того, чтобы заставить VI выполнять действие только один раз при каждом воздействии на элемент управления. Такой вариант поведения нельзя выбрать для радио-кнопок.
- **Latch when released** — изменяет значение элемента управления только после того, как вы отпустили кнопку мыши. Когда VI один раз считывает это значение, орган управления возвращается к своему значению по умолчанию. Такое поведение похоже на поведение кнопок в диалоговом окне и системных кнопок. Такой вариант поведения нельзя выбрать для радио-кнопок.
- **Latch until released** — изменяет значение элемента управления при щелчке по нему мышью и сохраняет это значение до тех пор, пока VI его однократно не считает или вы не отпустите кнопку мыши, в зависимости от того, что какое событие произойдет позже. Такой вариант поведения нельзя выбрать для радио-кнопок.

Чтобы узнать больше о механических действиях, поэкспериментируйте с Mechanical Action of Booleans VI из поисковика NI Example Finder.

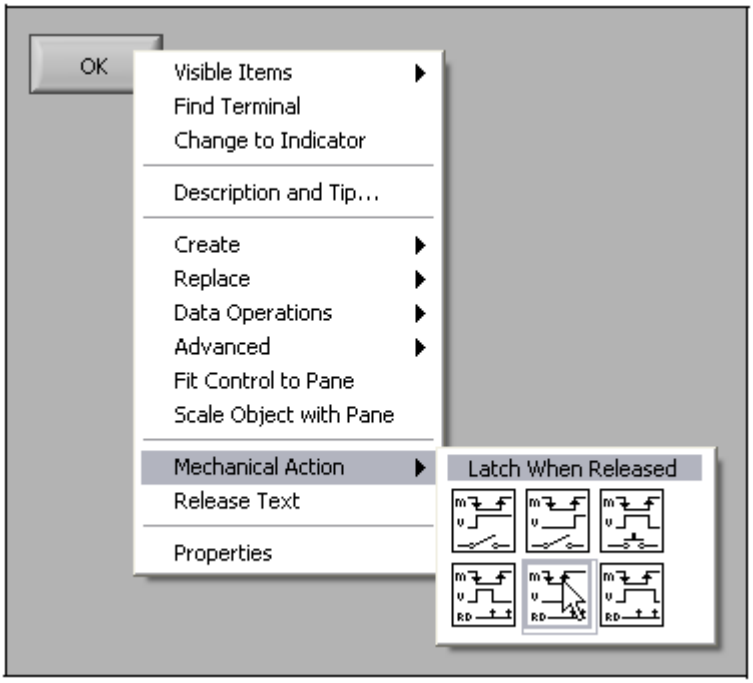


Рисунок 4-6. Механические действия булевских элементов

Строки

Строка — это последовательность отображаемых или не отображаемых ASCII символов. Строки обеспечивают независимый от платформы формат представления информации и данных. Наиболее часто строки применяют в следующих приложениях:

- Создание простых текстовых сообщений.
- Управление измерительными приборами путем посылки текстовых команд прибору и прием из прибора значений данных в виде ASCII строк или в виде двоичных строк, которые затем преобразуются в числовые значения.
- Хранение числовых данных на диске. Чтобы сохранить числовые данные в ASCII файле, то перед записью данных в файл числовые данные должны быть преобразованы в строки.
- Выдача указаний и приглашений пользователю с помощью диалоговых окон.

На лицевой панели строки появляются в виде таблиц, окон для ввода текста и меток. В LabVIEW есть встроенные VI и функции, с помощью которых можно манипулировать строками, в том числе форматировать, разбирать на составляющие и т.д.

Подробная информация об ASCII-кодах и функциях преобразования приведена в разделе «ASCII Codes».

В LabVIEW строковые данные представляются розовым цветом.

Щелкните правой кнопкой мыши по строковому элементу управления или индикатору на лицевой панели, чтобы выбрать один из форматов вывода на

экран, приведенных в нижеследующей таблице. В этой таблице для каждого формата отображения приведен пример сообщения.

Формат отображения	Описание	Сообщение
Normal Display	Отображение печатаемых символов шрифтом элемента управления. Неотображаемые символы обычно выводятся прямоугольниками.	`There are four display types. \ is a backslash.`
'\' Codes Display	Отображение кодов всех непечатаемых символов с обратной косой чертой.	`There\sare\sfour\ sdisplay\stypes.\n \\\sis\sa\sbackslash.`
Password Display	Отображение звездочкой каждого символа, включая пробел.	`**********`
Hex Display	Отображение каждого значения ASCII кода в шестнадцатеричном формате	`5468 6572 6520 6172 6520 666F 7572 2064 6973 706C 6179 2074 7970 6573 2E0A 5C20 6973 2061 2062 6163 6B73 6C61 7368 2E`

LabVIEW хранит строки в виде указателей на структуры, каждая из которых содержит 4-байтовое поле длины строки, за которым следует одномерный массив 1-байтовых целых чисел (8-битовых символов).

Перечислительный тип данных

Перечислительный тип данных (это может быть элемент управления, константа или индикатор) — это комбинация из нескольких типов данных. Перечислительный тип данных (enum) представляется в виде пары значений: строкового типа и числового типа, и значение enum является одним из вариантов списка значений. Если, например, вы создаете перечислительный тип, который имеет имя Month, то для него возможный вариант пар значений будет следующим: January - 0, February – 1 и т.д. до December - 11. На рисунке 4-7 приведен пример таких пар значений в диалоговом окне **Properties** элемента управления перечислительного типа.

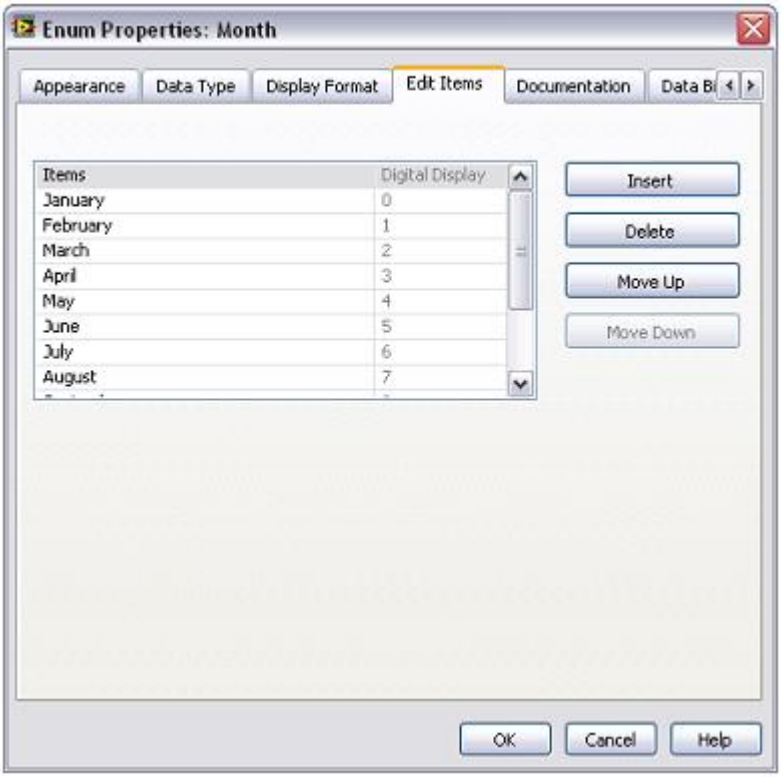


Рисунок 4-7. Свойства элемента управления перечислительного типа Month

Перечислительный тип полезен тем, что на блок-диаграмме он позволяет манипулировать с числами, а это проще, чем со строками. На рисунке 4-8 показан элемент управления перечислительного типа **Month**, выделение пары данных по строке, которой соответствует целое, и соответствующий терминал на блок-диаграмме.

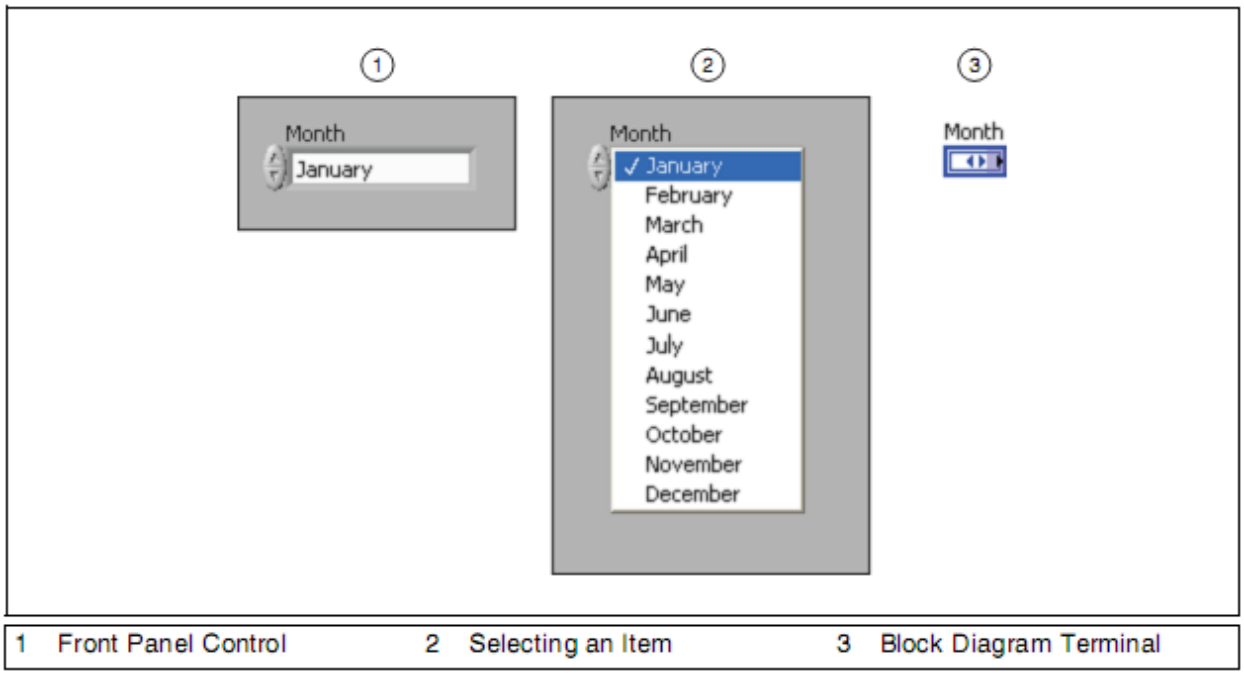


Рисунок 4-8. Элемент управления перечислительного типа Month

Динамический тип данных

Динамический тип данных служит для хранения информации, генерируемую или полученную в Express VI. Динамический тип данных отображается в виде терминала темно-синего цвета, который показан слева. Большинство Express VI получают на вход или возвращают данные динамического типа. Данные динамического типа можно подавать на любой индикатор или вход, который воспринимает данные числового, сигнального или булевского типа. Данные динамического типа следует выводить на индикатор, который отображает их наилучшим образом, таким, как графики типа Graph, Chart или числовые индикаторы.

Большинство других VI и функций в LabVIEW не могут работать с динамическим типом данных. Чтобы воспользоваться встроенным VI или функцией для анализа или обработки данных динамического типа, необходимо применить к этим данным операцию преобразования типа.

Для преобразования данных динамического типа к числовому, булевскому, сигнальному типам данных, а также типу данных «массив», которые могут применяться с другими VI и функциями, следует использовать Convert from Dynamic Data Express VI. Когда вы помещаете этот Express VI на блок-диаграмму, появляется диалоговое окно **Configure Convert from Dynamic Data**. В этом окне отображаются опции, которые позволяют задать желаемый формат данных, возвращаемых Convert from Dynamic Data Express VI.

Если подать данные динамического типа на индикатор массива, LabVIEW автоматически добавляет на блок-диаграмму Convert from Dynamic Data Express VI. Щелкните дважды по этому VI, чтобы открыть диалоговое окно **Configure Convert from Dynamic Data**, и проконтролировать, в каком виде данные появятся в массиве.

C. Документирование программного кода

Профессиональные разработчики, которые поддерживают и модифицируют VI, знают цену хорошей документации. Документирование блок-диаграммы существенно упростит в будущем усовершенствование программного кода. Кроме того, необходимо тщательно документировать окно лицевой панели, чтобы пояснить назначение VI и находящихся на лицевой панели объектов.

Используйте подсказки, описания и свойства VI, методы хорошего проектирования для документирования лицевой панели.

Подсказки и описания

Подсказки – это краткие описания, которые появляются, когда вы перемещаете курсор над элементом управления или индикатором в процессе выполнения VI. Вы можете, например, добавить подсказку, чтобы показать, что температура выражается в градусах Цельсия или объяснить, как используется вход в реализованном алгоритме. В описаниях

предоставляется дополнительная информация об отдельных элементах управления и индикаторах. Они появляются в окне **Context Help**, когда вы перемещаете курсор над объектом. Чтобы добавить подсказки и описания для объектов лицевой панели, щелкните правой кнопкой мыши по элементу управления или индикатору и из контекстного меню выберите команду **Description and Tip**.

Диалоговое окно VI Properties

Чтобы создать описания VI и связать VI с HTML файлами или компилируемыми файлами справки, используйте компонент Documentation в диалоговом окне **VI Properties.** Чтобы открыть окно **VI Properties**, выберите команду меню **File»VI Properties** или щелкните правой кнопкой мыши по иконке VI на лицевой панели или блок-диаграмме и из контекстного меню выберите команду **VI Properties.** Затем из выпадающего меню **Categories** выберите пункт **Documentation.** Во время выполнения VI диалоговое окно **VI Properties** открыть нельзя.

Страница **Documentation** состоит из следующих компонентов:

- **VI description** — содержит текст, который появляется в окне **Context Help**, если вы перемещаете курсор над иконкой VI. Перед любым текстом внутри описания, который вы хотите отформатировать как полужирный, поставьте тег <B>, а после текста — тег </B>. Вы можете также воспользоваться свойством VI Description, чтобы отредактировать описание VI программным путем.
- **Help tag** — содержит имя HTML файла или индексное ключевое слово раздела, которые вы хотите связать с откомпилированным файлом справки. Чтобы настроить тег справки программным путем, вы можете воспользоваться свойством Help:Document Tag.
- **Help path** — содержит путь к HTML файлу или откомпилированному файлу справки, с которым вы хотите установить связь из окна Context Help. Если это поле пустое, ссылка **Detailed help** не появляется в окне Context help, и кнопка **Detailed help** недоступна.
- **Browse** — выводит на экран диалоговое окно работы с файлом, чтобы найти HTML файл или откомпилированный файл справки и получить путь к справке (Help path).

Назначение имен органам управления и индикаторам

Присвоение логичных и наглядных имен элементам управления и индикаторам делает более удобной работу с лицевыми панелями. Если, например, вы называете элемент управления `Temperature`, пользователь может не знать, в каких единицах она выражается. Обозначение элемента управления именем `Temperature °C` делает лицевую панель более информативной. Теперь вы знаете, что значения температуры следует вводить в единицах метрической системы мер.

Графическое программирование

Хотя графический характер среды LabVIEW способствует автоматическому документированию блок-диаграмм, дополнительные комментарии будут полезны при последующей модернизации ваших VI. Различают два типа комментариев на блок-диаграмме: комментарии, которые описывают функцию или работу алгоритмов, и комментарии, которые поясняют назначение данных, которые передаются по проводникам. Оба типа комментариев приведены на блок-диаграмме (рисунок 4-9). Вы можете вставить стандартные метки с помощью инструмента Labeling или свободную метку из суб-палитры **Functions»Programming»Structures» Decorations**. По умолчанию, свободные метки имеют фон желтого цвета.

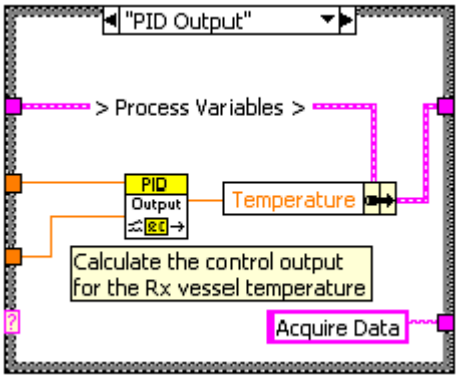


Рисунок 4-9. Документирование на блок-диаграмме

При комментировании VI руководствуйтесь следующими рекомендациями:

- Комментарии должны пояснять, что делает программный код.
- Несмотря на то, что, благодаря графическому характеру среды LabVIEW, программный код обладает свойством самодокументирования, для описания функционирования блок-диаграммы следует использовать свободные метки.
- Не показывайте метки в точках вызова функций и subVI, поскольку они обычно крупные и загроможддают блок-диаграмму. Разработчик, глядя на блок-диаграмму, может отыскать имя функции или subVI с помощью окна **Context Help**.
- Обозначайте с помощью свободных меток малого размера с белым фоном длинные проводники, чтобы указать их назначение. Обозначение метками полезно для проводников, исходящих от сдвиговых регистров и для длинных проводников, которые проходят по всей блок-диаграмме. За более подробной информацией о сдвиговых регистрах обратитесь к разделу *Case-структуры* этой лекции.
- Используйте метки для структур, чтобы точно показать их главное функциональное назначение.
- Обозначайте метками константы, чтобы показать их сущность.
- С помощью свободных меток документируйте алгоритмы, которые реализуются блок-диаграммой. Если вы заимствуете алгоритм из книги или из другого источника, сделайте на него ссылку.

D. Циклы While

Аналогично циклам Do или Repeat-Until в текстовых языках программирования, цикл While, приведенный на рисунке 4-10, выполняет код суб-диаграммы, пока не выполнится условие.

На рисунке 4-10 показаны цикл While в LabVIEW, представление работы данного цикла в виде эквивалентной блок-схемы алгоритма и пример соответствующего псевдокода.

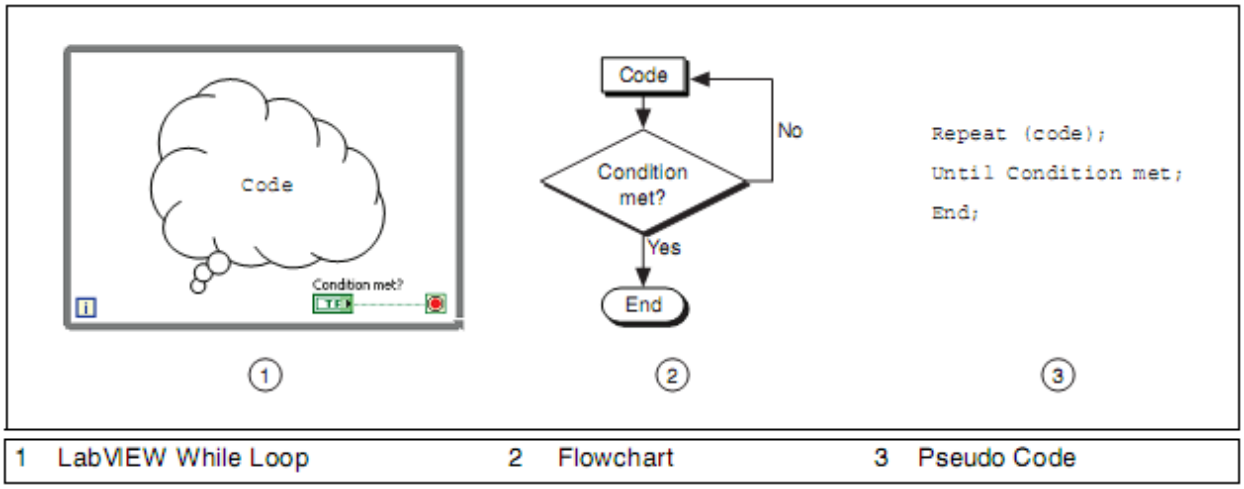


Рисунок 4-10. Цикл While

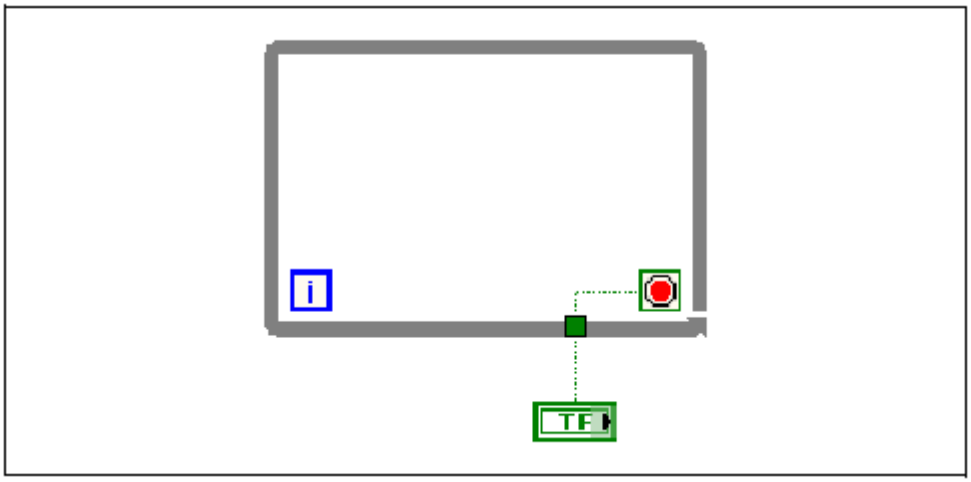
Цикл While находится в палитре **Structures**. Выберите цикл While из палитры, а затем с помощью курсора растяните прямоугольник выделения вокруг фрагмента блок-диаграммы, который должен выполняться неоднократно. Когда вы отпустите кнопку мыши, рамка цикла While охватит выделенный фрагмент. Добавьте объекты, перетаскивая их внутрь цикла.

Подсказка: While всегда выполняется хотя бы один раз

Суб-диаграмма в цикле While выполняется до тех пор, пока на вход терминала условия не поступит определенное булевское значение. Терминал условия в цикле While ведет себя аналогично такому же терминалу в цикле For, сконфигурированного с терминалом условия. Однако, поскольку для цикла For задается число итераций, он не работает бесконечно, если условие никогда не выполняется. Так как у цикла While нет задатчика итераций, он работает бесконечно, если условие никогда не выполняется.

Если в цикле While используется терминал условия **Stop if True** и терминал элемента управления булевского типа размещается за пределами цикла, то при значении элемента управления FALSE в начале работы цикла цикл While будет выполняться бесконечно, как это показано в следующем примере. Цикл также будет выполняться бесконечно, если терминал условия

сконфигурирован, как **Continue if True**, а элемент управления, помещенный вне цикла, установлен в состояние TRUE.



Изменение состояния элемента управления не останавливает бесконечный цикл, поскольку его значение считывается только один раз перед началом работы цикла. Чтобы остановить бесконечный цикл, вам необходимо аварийно прервать выполнение VI щелчком мыши по кнопке **Abort Execution** на панели инструментов.

С помощью терминала условия цикла While можно также выполнять элементарную обработку ошибок. Если подсоединить кластер ошибок к терминалу условия, на терминал передается только булевское значение (True или False) параметра **status**. При этом пункты **Stop if True** и **Continue if True** контекстного меню терминала меняются соответственно на **Stop if Error** и **Continue while Error**.

Терминал итераций является выходным, с которого считывается количество выполненных итераций.

Счет итераций цикла While всегда начинается с нуля.

В блок-диаграмме на рисунке 4-11 цикл While выполняется до тех пор, пока выход функции Random Number не станет больше либо равен `0.9` и элемент управления **Enable** не примет значение `True`. Функция And возвращает значение `True` только в том случае, если на оба входа поданы значения `True`. В противном случае она возвращает значение `False`.

В примере, приведенном на рисунке 4-11, вероятность того, что цикл будет работать бесконечно, велика. В общем случае предпочтительнее, чтобы условие прекращения работы цикла было одно, и чтобы не требовалось выполнения обоих условий одновременно.

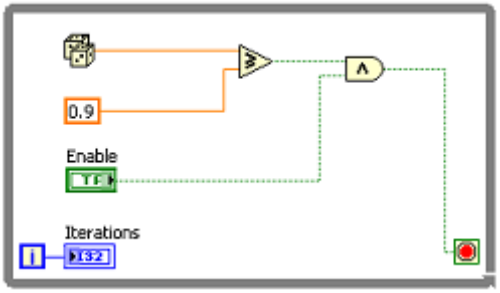


Рисунок 4-11. Возможно, цикл будет выполняться бесконечно

Туннели структур

Через туннели данные передаются в структуру и из структуры. Туннель имеет вид закрашенного квадратика, расположенного на границе цикла While. Цвет туннеля определяется типом данных подключенного к туннелю проводника. Данные выдаются за пределы цикла после того, как цикл закончит работать. Если данные поступают в цикл через туннель, то цикл выполняется только после того, как данные поступят в туннель.

В блок-диаграмме на рисунке 4-12 терминал счетчика итераций присоединен к туннелю. Данные из этого туннеля не проходят на индикатор **Iterations** до тех пор, пока не закончится выполнение цикла While.

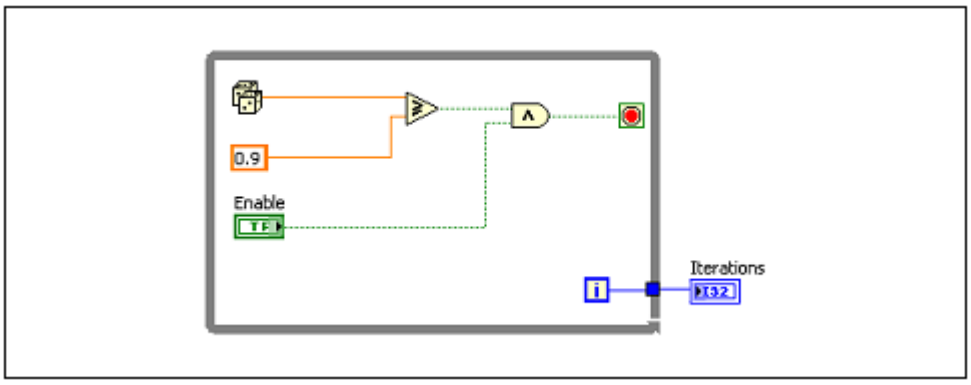


Рисунок 4-12. Туннель цикла While

На индикатор **Iterations** выводится только последнее значение терминала счетчика итераций.

Применение циклов While для обработки ошибок

Вы можете соединить кластер ошибки с терминалом условия цикла While или цикла For, в котором есть терминал условия, чтобы остановить повторения цикла. При этом терминалу условия передается только значение TRUE или FALSE параметра **status** кластера ошибки. При возникновении ошибки цикл останавливается. В цикле For с терминалом условия, чтобы задать максимальное количество итераций, вы также должны подать некоторое значение на терминал задания количества итераций или разрешить авто-индексацию входного массива. Цикл For выполняется до тех пор, пока не возникнет ошибка или не закончится выполнение заданного количества итераций.

Если вы соедините кластер ошибок с терминалом условия, пункты **Stop if True** и **Continue if True** контекстного меню терминала заменятся соответственно на **Stop on Error** и **Continue while Error**.

На рисунке 4-13 момент прекращения работы цикла определяется с помощью кластера ошибки и кнопки останова цикла. Таким способом рекомендуется останавливать большинство циклов.

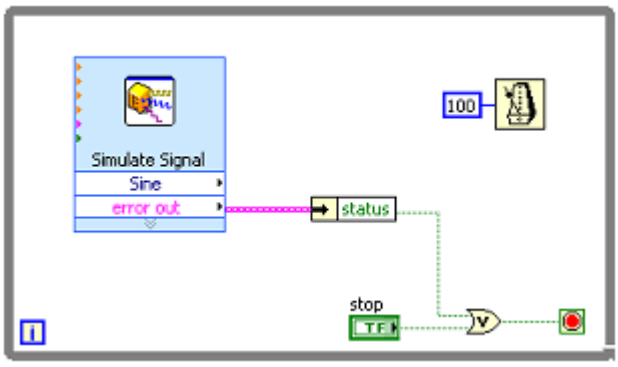


Рисунок 4-13. Остановка цикла While

E. Циклы For

Цикл For выполняет суб-диаграмму заданное количество раз. На рисунке 4-14приведен цикл For в LabVIEW, его эквивалентная блок-схема алгоритма, а также пример эквивалентного псевдокода.

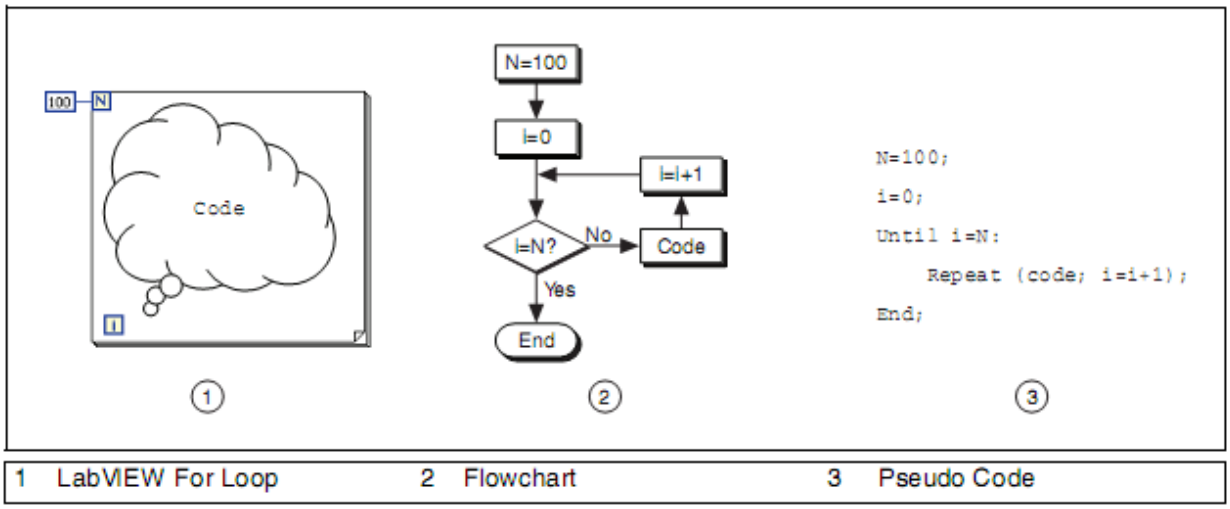


Рисунок 4-14. Цикл For

Цикл For находится в палитре **Structures**. Вы можете также поместить на блок-диаграмму цикл While, щелкнуть правой кнопкой мыши по его границе и заменить его с помощью команды контекстного меню **Replace with For Loop** на цикл For.

Терминал задания количества итераций является входным терминалом, значение которого определяет, сколько раз будет выполняться суб-диаграмма.

Терминал итераций является выходным, с него считывается количество выполненных итераций.

Счет итераций цикла For всегда начинается с нуля.

Цикл For отличается от цикла While тем, что цикл For выполняется заданное количество раз. Цикл While прекращает выполнение суб-диаграммы только в том случае, если на терминал условия приходит соответствующее значение.

На рисунке 4-15 в течение 100 секунд цикл For генерирует каждую секунду случайное число и выводит эти случайные числа на числовой индикатор.

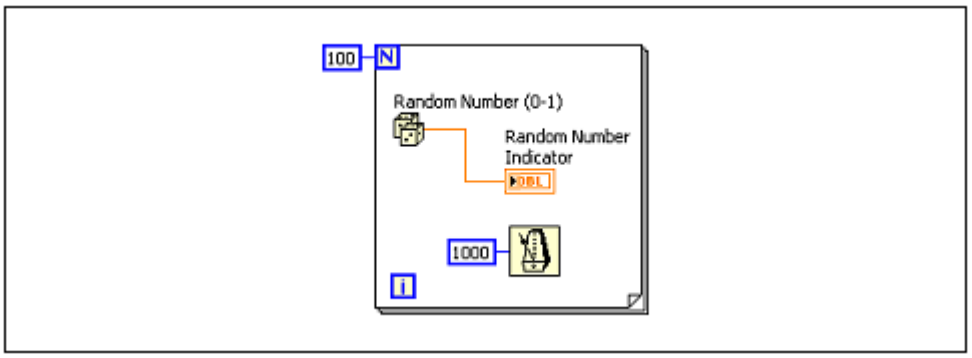


Рисунок 4-15. Пример цикла For

Добавление условного терминала в цикл For

Если необходимо, в цикл For можно добавить терминал условия, чтобы остановить цикл при выполнении логического условия или при возникновении ошибки. Цикл For с терминалом условия выполняется до тех пор, пока не выполнится условие или пока не выполнятся все итерации, т.е. при наступлении того из этих событий, которое произойдет раньше. В терминале задания количества итераций цикла For, сконфигурированного для выхода по условию, появляется красный значок, а в правом нижнем углу структуры For – терминал условия. После того, как цикл For будет сконфигурирован на выход по условию, он выглядит так, как показано на рисунке 4-16.

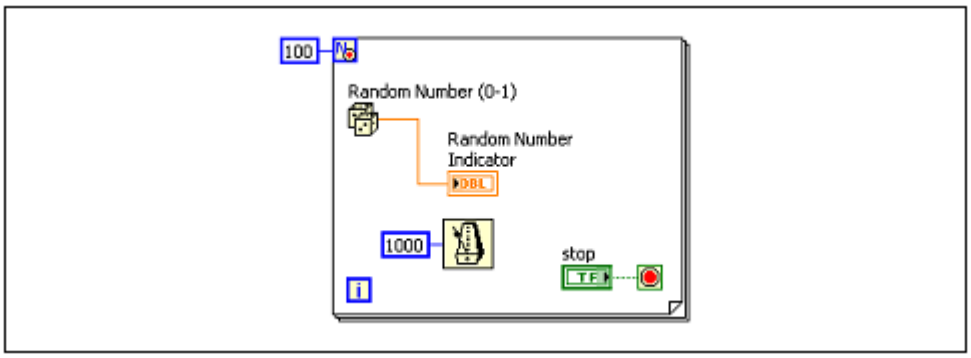


Рисунок 4-16. Цикл For сконфигурирован на выход по условию

Чтобы добавить терминал условия в цикл For, щелкните правой кнопкой мыши по границе цикла и выберите из контекстного меню команду **Conditional Terminal**. Затем выполните подключения терминала условия и терминала задания количества итераций.

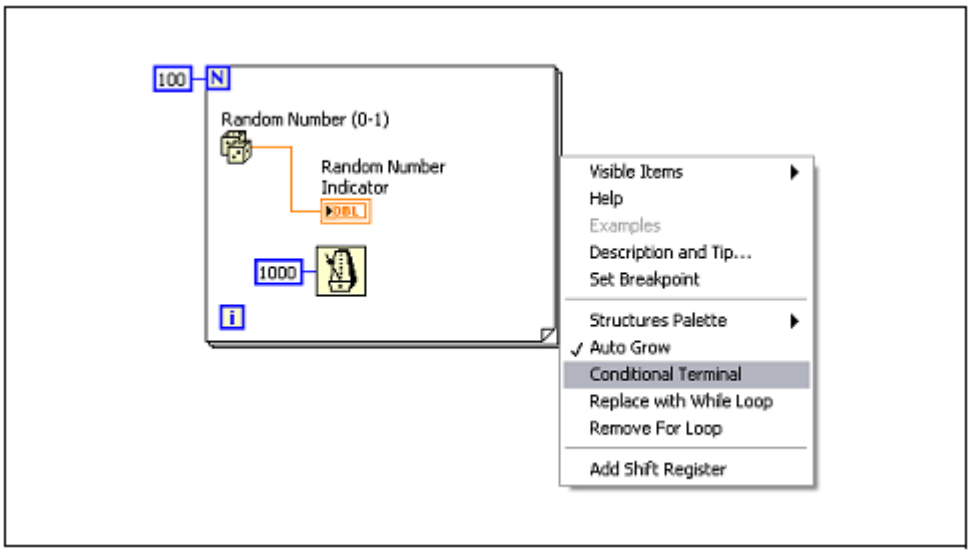


Рисунок 4-17. Добавление терминала условия в цикл For

Преобразование числовых типов данных

LabVIEW позволяет представлять данные таких типов, как целые числа со знаком и без знака, вещественные числа с плавающей точкой или комплексные числа; это обсуждалось в разделе *Типы данных в LabVIEW* настоящего занятия. Обычно, если на входы функции подаются данные разных типов, на выход функция возвращает результат в более «длинном» или «широком» формате. Если вы выполняете действие над целым числом со знаком и целым числом без знака, производится приведение к целому без знака. Если действие выполняется над целым без знака и числом с плавающей точкой, то производится приведение к числу с плавающей точкой. Если действие выполняется над числом с плавающей точкой и комплексным числом, то производится приведение к комплексному числу. Если два числа одного и того же типа состоят из разного количества бит, LabVIEW выполнит приведение результата к числу с наибольшим количеством бит. Если количество бит одно и то же, LabVIEW делает выбор в пользу целого без знака, а не со знаком. Если, например, на функцию Multiply подать данные типа DBL и I32, тип данных результата будет DBL (рисунок 4-18). LabVIEW приводит к вещественному числу с плавающей точкой 32-разрядное целое число со знаком, поскольку в целом числе используется меньше бит, чем в числе с плавающей точкой с двойной точностью. Нижний вход функции Multiply обозначен красной точкой, называемой точкой приведения, которая указывает на то, что в LabVIEW выполнена операция приведения типа данных.

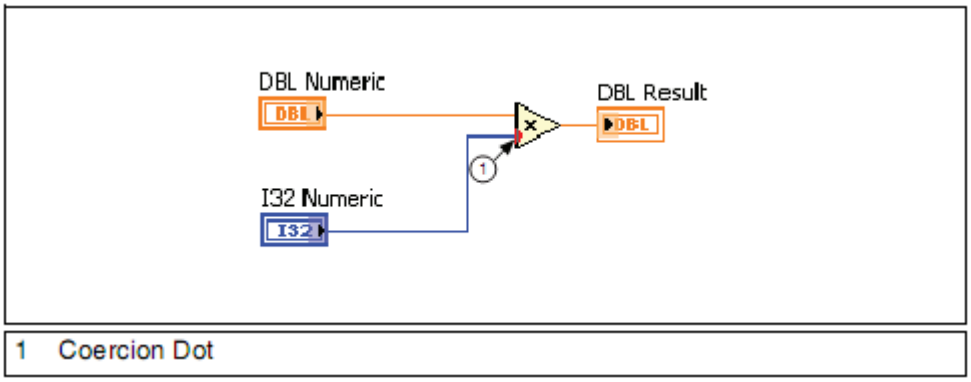


Рисунок 4-18. Пример преобразования типа данных числа

Однако, терминал количества итераций работает противоположным образом. Если вы подадите на этот 32-разрядный терминал число с плавающей точкой двойной точности, LabVIEW приводит более «длинное» числовое значение к 32-разрядному целому числу со знаком. Несмотря на то, что данное преобразование типов противоречит обычным стандартам преобразования, оно является необходимым, поскольку цикл For может выполняться только целое количество раз.

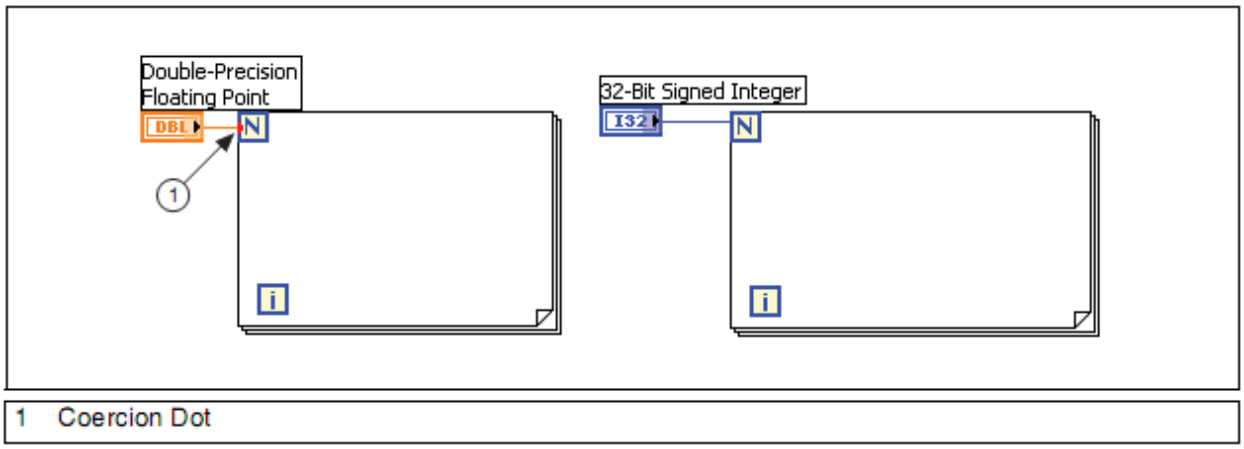


Рисунок 4-19. Приведение типа числа для цикла For

Чтобы обеспечить более высокую производительность, следует избегать приведения типов, используя согласующиеся типы данных (рисунок 4-20, слева), либо преобразуя данные программно в соответствующий тип (рисунок 4-20, справа).

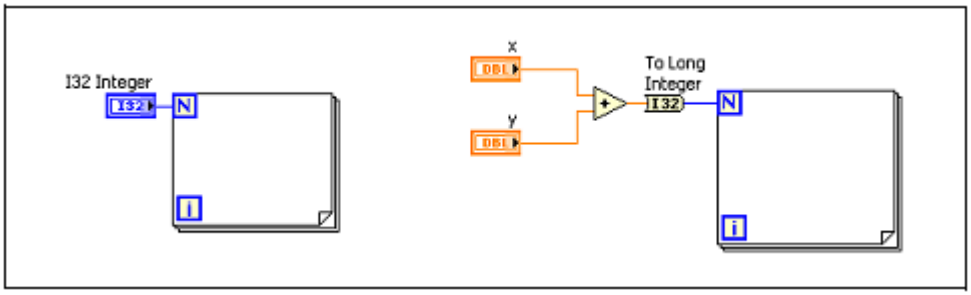


Рисунок 4-20. Избегайте приведения типов чисел, используя согласованные типы данных

F. Тактирование VI

Когда цикл завершает выполнение некоторой итерации, он немедленно приступает к выполнению следующей итерации, пока не выполнится условие останова. Чаще всего бывает необходимо управлять частотой выполнения итераций или тактированием. Если, например, производится сбор данных, которые необходимо получать через каждые 10 секунд, то здесь нужен такой способ тактирования итераций цикла, чтобы итерация цикла выполнялись один раз в 10 секунд.

Даже если вам не требуется, чтобы цикл выполнялся с определенной частотой, все равно необходимо дать процессору время на завершение других задач, таких, как реакция на пользовательский интерфейс. В настоящем разделе представлены некоторые методы тактирования циклов.

Функции ожидания

Поместите внутрь цикла функцию ожидания, чтобы VI мог перейти в режим ожидания в течение заданного интервала времени. Это позволяет процессору перейти к другим задачам во время интервала ожидания. Функции ожидания используют тактовые импульсы операционной системы с периодом повторения одна миллисекунда.

Функция Wait Until Next ms Multiple контролирует счетчик тактовых импульсов с периодом в одну миллисекунду, и ожидает, пока счетчик не достигнет значения, кратного заданному. Используйте эту функцию для синхронизации действий. Поместите данную функцию в цикл для управления частотой выполнения итераций. Чтобы эта функция работала эффективно, время исполнения программного кода должно быть меньше времени, заданного для этой функции. Время выполнения первой итерации цикла неопределено.

Функция Wait (ms) ожидает до тех пор, пока счетчик миллисекунд не досчитает до значения, заданного на входе. Эта функция гарантирует, что частота выполнения итераций цикла будет, по меньшей мере, соответствовать тому значению, которое задано на входе.

Примечание. Time Delay Express VI ведет себя аналогично функции Wait (ms) и, кроме того, в этот VI встроены кластеры ошибок. Дополнительная информация о кластерах ошибок рассматривается в лекции 3, *Поиск ошибок и отладка VI.*

Истекшее время

В некоторых случаях полезно узнать, сколько времени прошло с некоторого момента работы VI. Express VI Elapsed Time выдает количество времени, которое прошло после заданного начального момента. Этот Express VI отслеживает время в процессе работы VI и не предоставляет процессору время на завершение других задач. Вам предстоит воспользоваться Express VI Elapsed Time в курсовом проекте на тему Weather Station (Метеостанция)

G. Передача данных от итерации к итерации

Когда в программах применяют циклы, часто бывает необходимо получать доступ к данным, сформированным на предыдущих итерациях цикла. Если, например, на каждой итерации цикла вы получаете по одному элементу данных, и нужно усреднять каждые пять элементов, то необходимо сохранять данные, полученные в предыдущих итерациях цикла.

Примечание: Другим способом хранения данных от предыдущих итераций является применение узлов обратной связи Feedback Node. За более подробной информацией об этих узлах обратитесь к разделу *Feedback Node* справки *LabVIEW Help*

Сдвиговые регистры аналогичны статическим переменным в текстовых языках программирования.

Воспользуйтесь сдвиговыми регистрами, если нужно передать значения данных из предыдущих итераций цикла к последующей итерации. Сдвиговый регистр появляется в виде пары терминалов, расположенных строго друг против друга на вертикальных границах цикла.

Стрелка в терминале на правой стороне цикла направлена вверх, и этот терминал хранит данные, полученные при завершении итерации. LabVIEW передает данные в следующую итерацию с правой стороны регистра. По окончании выполнения цикла терминал на его правой стороне возвращает последнее значение, сохраненное в сдвиговом регистре.

Создается сдвиговый регистр щелчком правой кнопки мыши по левой или правой границе цикла командой контекстного меню **Add Shift Register**.

Сдвиговый регистр пропускает через себя данные любого типа; регистру автоматически присваивается тип данных первого объекта, подключенного к регистру. Данные, которые подаются на терминалы каждого сдвигового регистра, должны быть одного и того же типа.

В цикле может быть любое количество сдвиговых регистров. Если у вас несколько операций, в которых используются предыдущие значения итераций цикла, применяйте несколько сдвиговых регистров для хранения значений данных от различных процессов в структуре (рисунок 4-21).

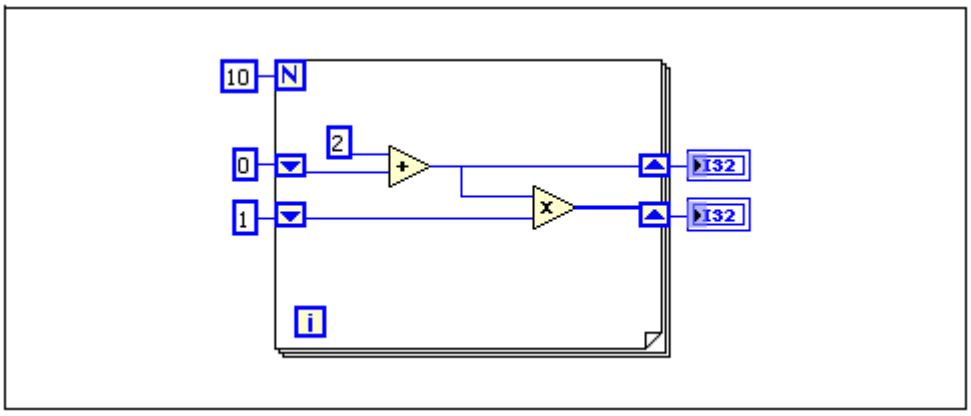


Рисунок 4-21. Несколько сдвиговых регистров

Инициализация сдвиговых регистров

При инициализации сдвигового регистра сбрасывается значение, которое регистр пропускает через себя при выполнении первой итерации цикла. Инициализация осуществляется подключением элемента управления или константы к терминалу сдвигового регистра на левой стороне границы цикла (рисунок 4-22).

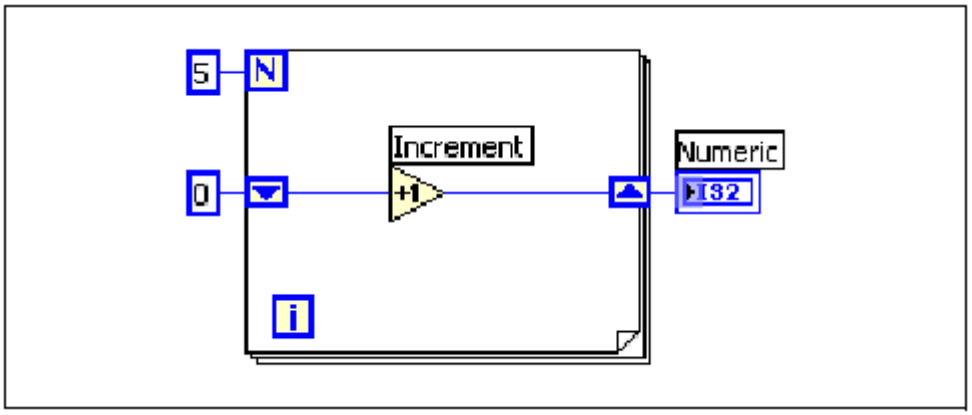


Рисунок 4-22. Инициализированный сдвиговый регистр

На рисунке 4-22 цикл For выполняется пять раз, в каждой итерации увеличивая на единицу значение, передаваемое сдвиговым регистром. После выполнения пяти итераций цикла For сдвиговый регистр передает последнее значение `5` на индикатор, а VI прекращает свою работу. При каждом запуске VI сдвиговый регистр начинает со значения `0`.

Если не проинициализировать сдвиговый регистр, в цикле используется значение, записанное в сдвиговый регистр во время последнего выполнения цикла или, если цикл никогда не выполнялся, значение по умолчанию для соответствующего типа данных.

Неинициализированный сдвиговый регистр используют, чтобы сохранять информацию о состоянии между последовательными выполнениями VI. Сдвиговый регистр без инициализации показан на рисунке 4-23.

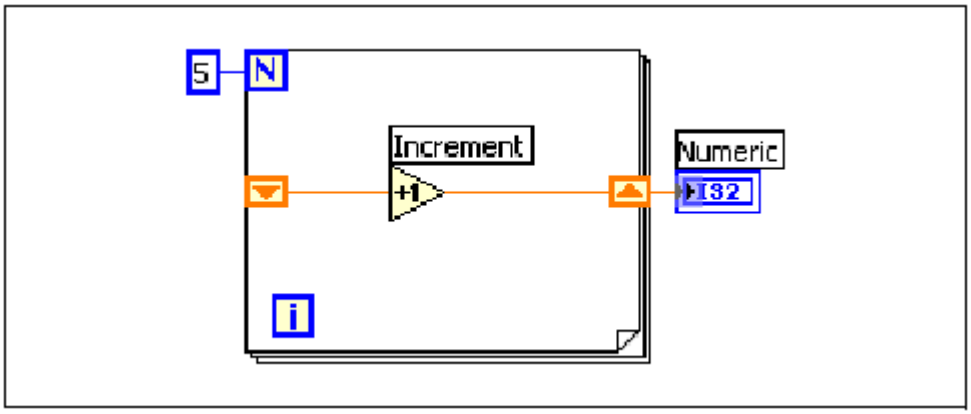


Рисунок 4-23. Инициализированный сдвиговый регистр

На рисунке 4-23 цикл For выполняется пять раз, в каждой итерации увеличивая на единицу значение, передаваемое сдвиговым регистром. При первом запуске VI начальное значение сдвигового регистра равно `0` - это значению по умолчанию для 32-разрядного целого. После пяти итераций цикла For сдвиговый регистр выдает последнее значение, равное `5`, на индикатор, и VI прекращает свою работу. При следующем запуске VI начальное значение сдвигового регистра равно `5` — последнему значению, которое осталось от предыдущего выполнения VI. После пяти итераций сдвиговый регистр выдает на индикатор `10`. Если вы запустите VI снова, сдвиговый регистр начнет со значения `10` и т.д. Неинициализированные сдвиговые регистры хранят значения от предыдущей итерации до тех пор, пока вы не закроете VI.

Стековые сдвиговые регистры

Стековые сдвиговые регистры позволяют получить данные от предыдущих итераций цикла. Такие регистры запоминают значения от нескольких предыдущих итераций и переносят их в следующие итерации. Чтобы создать стековый сдвиговый регистр, щелкните правой кнопкой по левому терминалу и выберите из контекстного меню команду **Add Element**.

Стековые сдвиговые регистры могут находиться только на левой границе цикла, поскольку терминал на правой стороне передает в следующую итерацию данные, сгенерированные только в текущей итерации (рисунок 4-24).

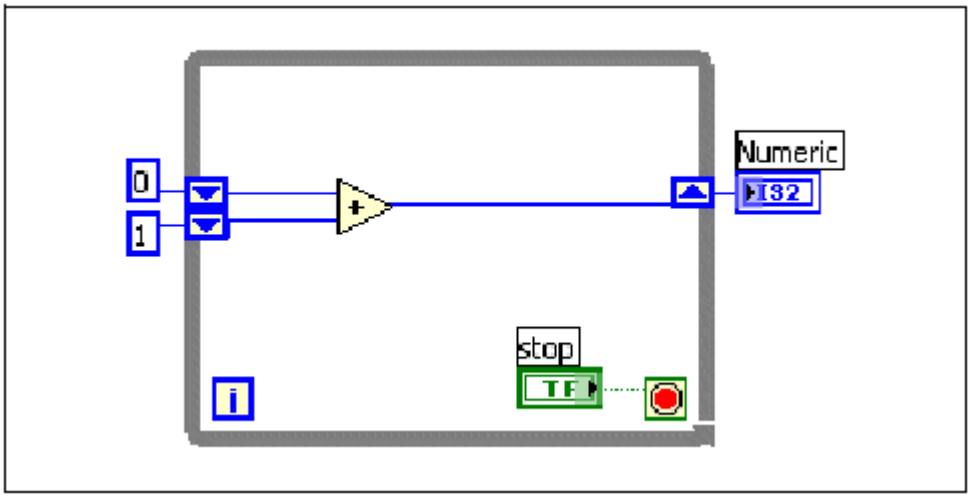


Рисунок 4-24. Стековые сдвиговые регистры

Если на предыдущей блок-диаграмме вы добавите еще один элемент к левому терминалу, то значения последних двух итераций перейдут в следующую итерацию, а значение новой итерации запоминается в верхнем сдвиговом регистре. Нижний терминал хранит данные, переданные ему из предыдущей итерации.

H. Вывод данных на графические индикаторы

Вы уже пользовались графическими индикаторами типа Chart (график диаграмм) и Graph (график осциллограмм) для отображения простых данных. В этом параграфе более подробно объясняются особенности настройки и применения графических индикаторов.

Waveform Charts

Waveform Chart – это специальный тип числового индикатора, на котором отображаются один или более графиков данных, получаемых обычно через постоянный интервал времени. На таких индикаторах может отображаться один или несколько графиков. На рисунке 4-25 показаны элементы графического индикатора диаграмм, на котором отображаются две временные зависимости: Raw Data и Running Avg.

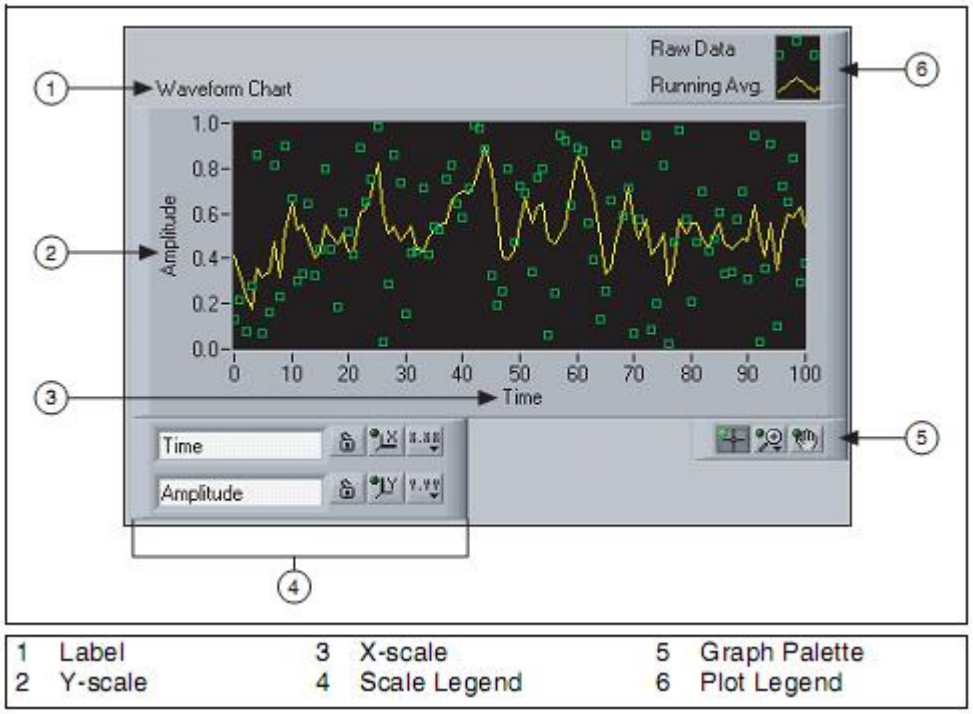


Рисунок 4-25. Индикатор Waveform Chart

Чтобы настроить режим обновления графического индикатора при выводе новых данных, щелкните правой кнопкой мыши по графику и выберите из контекстного меню команду **Advanced»Update Mode.** Можно использовать следующие режимы отображения данных:

- **Strip Chart** — отображаются данные, непрерывно прокручиваемые по индикатору слева направо, причем старые данные отображаются слева, а новые — справа. Такой режим аналогичен регистрации на бумажную ленту в самописце и является режимом обновления данных по умолчанию.
- **Scope Chart** — отображается один элемент данных, например, импульса или колебания, с частичной прокруткой графического индикатора слева направо. Каждое новое значение отображается справа от последнего значения. Когда кривая достигает правой границы области вывода графика, LabVIEW удаляет эту кривую и начинает рисовать заново с левой границы. Повторная прорисовка графика напоминает работу осциллографа.
- **Sweep Chart** — работает аналогично режиму **Scope Chart**, за исключением того, что в режиме **Sweep Chart** старые данные отображаются справа, а новые данные, отделенные вертикальной чертой, - слева. LabVIEW не удаляет кривую в данном режиме при достижении графиком правой границы области вывода графика. Этот режим обновления данных похож на функционирование ЭКГ монитора.

На рисунке 4-26 приведен пример каждого режима обновления диаграммы. В режимах **Scope Chart** и **Sweep Chart** происходит повторная прорисовка данных, как в осциллографе. Поскольку перерисовка графиков в этих режимах требует меньше издержек, они рисуются значительно быстрее, чем в режиме **Strip Chart.**

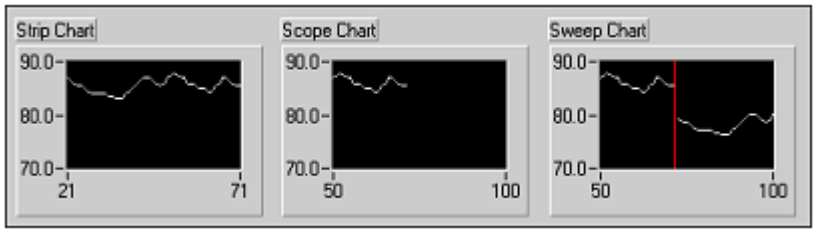


Рисунок 4-26. Режимы обновления графического индикатора Chart

Подключение графического индикатора Waveform Chart

На графическом индикаторе типа Waveform Chart можно отображать скалярную величину. На рисунке 4-27 тип терминала Waveform Chart соответствует типу входных данных.

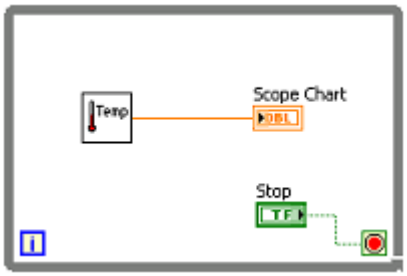


Рисунок 4-27. Подключение данных для отображения одной кривой на графическом индикаторе Waveform Chart

Функция Bundle, расположенная в палитре **Cluster, Class & Variant**, позволяет одновременно отображать несколько кривых на графическом индикаторе Waveform Chart. На рисунке 4-28 функция Bundle объединяет выходы трех VI, чтобы отобразить их на индикаторе Waveform Chart.

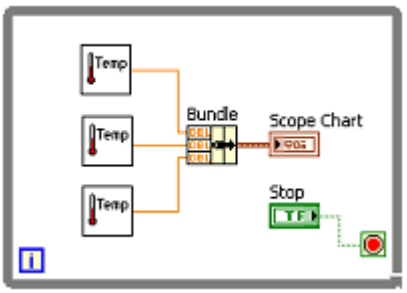
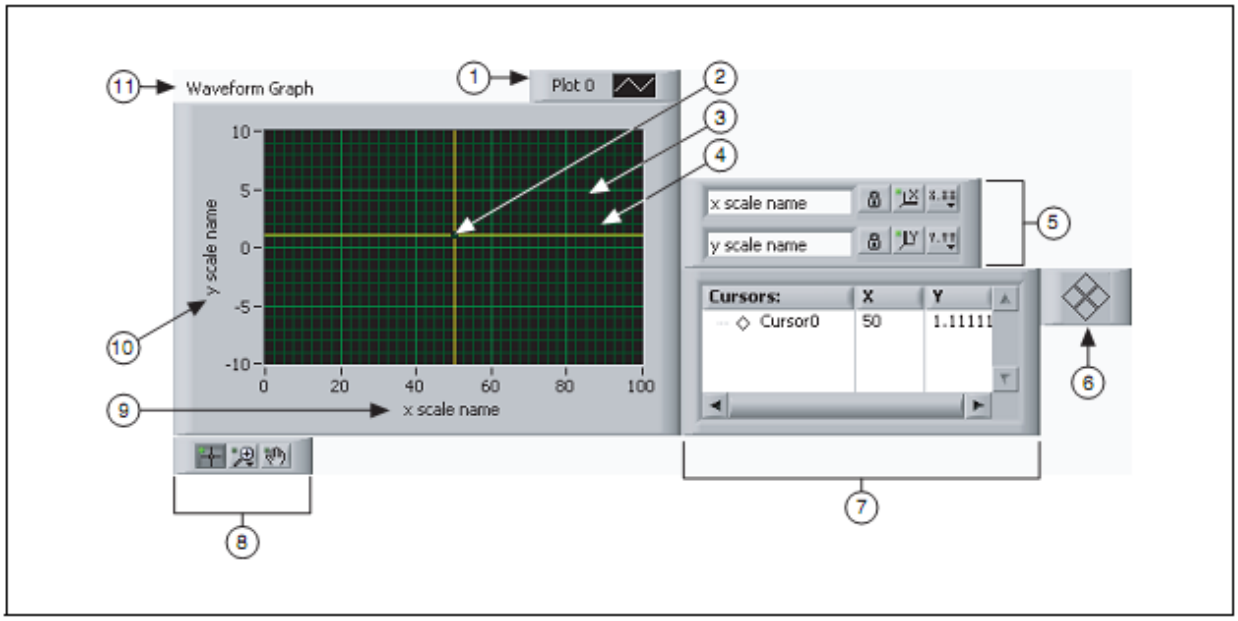


Рисунок 4-28. Подключение нескольких источников данных к графическому индикатору Waveform Chart

Тип данных терминала графического индикатора Waveform Chart изменяется, чтобы соответствовать типу данных с выхода функции Bundle. Чтобы добавить на графический индикатор кривые, измените размер функции Bundle с помощью инструмента Positioning. Подробная информация о функции Bundle приведена в лекции 5, *Связываемые данные*.

Waveform Graph

Как правило, в VI, в котором используется графический индикатор Waveform Graph, данные собираются в массив и затем выводятся на график. На рисунке 4-29 приведены элементы графического индикатора Waveform Graph.



1	Plot Legend	4	Mini-Grid Mark	7	Cursor Legend	10	Y-Scale
2	Cursor	5	Scale Legend	8	Graph Palette	11	Label
3	Grid Mark	6	Cursor Mover	9	X-Scale		

Рисунок 4-29. Waveform Graph

В палитре **Graph Indicators** расположены графические индикаторы Waveform Graph и XY Graph. На Waveform Graph выводятся только однозначные функции типа `y = f(x)`, у которых точки равномерно распределены вдоль оси x, например, измеренные сигналы в зависимости от времени. На графическом индикаторе XY Graph отображаются любые наборы точек, полученных с одинаковым или разным интервалом.

Чтобы отобразить несколько кривых, измените размер легенды графического индикатора. Вывод нескольких зависимостей на один индикатор позволяет сэкономить место на лицевой панели и упростить сравнение кривых. Waveform Graph и XY Graph автоматически подстраиваются для отображения нескольких графиков.

Отображение одной кривой на индикаторе Waveform Graph

Индикатор Waveform Graph для отображения одного сигнала может принимать данные нескольких типов. На индикатор можно подавать отдельный массив значений, которые интерпретируются как точки графика, причем индекс x инкрементируется на единицу, начиная с x=0. На график можно подавать кластер, который состоит из начального значения x, приращения x, и массива данных y. На график можно также подавать данные типа Waveform, которые состоят из самих данных, начального момента времени и приращения времени для сигнала.

Примеры типов данных, которые можно подавать на индикатор Waveform Graph, демонстрируются в Waveform Graph VI из библиотеки `labview\examples\general\graphs\gengraph.llb`.

Отображение нескольких кривых на индикаторе Waveform Graph

На индикаторе Waveform Graph можно отображать несколько графиков, представленных данными нескольких типов. На индикатор можно подать двумерный массив значений, в котором каждая строка — это массив для построения одной кривой. Элементы такого массива интерпретируются как точки графика, причем индекс *x* инкрементируется на единицу, начиная с *x*=0. Чтобы каждый столбец массива воспринимался как кривая, подключите двумерный массив к индикатору Waveform Graph, щелкните по индикатору правой кнопкой мыши и выберите из контекстного меню команду **Transpose Array**. Это особенно полезно при дискретизации данных с нескольких каналов DAQ-устройства, поскольку DAQ-устройство может возвращать данные в виде двумерных массивов, в котором данные каждого канала хранятся в отдельном столбце.

Пример с индикатором Waveform Graph, на который подается двумерный массив, демонстрируется Waveform Graph VI в варианте (Y) Multi Plot 1 graph из библиотеки `labview\examples\general\graphs\gengraph.llb`.

На индикатор Waveform Graph можно также подать кластер, который состоит из начального значения *x*, приращения *x* и двумерного массива данных *y*. Элементы такого массива интерпретируются как точки графика, причем индекс *x* инкрементируется на единицу с начального значения *x*. Этот кластерный тип данных полезен для отображения нескольких сигналов, у которых одна и та же частота дискретизации. Пример с индикатором Waveform Graph, на который подается описываемый кластер, демонстрируется Waveform Graph VI в варианте (Xo=10, dX=2, Y) Multi Plot 2 graph из библиотеки `labview\examples\general\graphs\gengraph.llb`.

На индикатор Waveform Graph можно также подать массив кластеров. Каждый кластер содержит одномерный массив данных. Внутренний массив описывает точки зависимости, а во внешнем массиве для каждой зависимости используется свой кластер. На лицевой панели на рисунке 4-30 показан массив кластеров **y**.

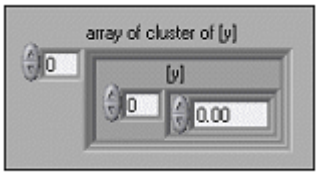


Рисунок 4-30. Массив кластеров **y**

Если количество элементов в каждом графике различно, используйте вместо двумерного массива массив зависимостей. Если вы, например, организуете

измерения по нескольким каналам, причем интервал между измерениями в каждом канале разное, применяйте массив зависимостей вместо двумерного массива, поскольку каждая строка двумерного массива должна иметь одинаковое количество элементов. Количество элементов во внутренних массивах массива кластеров может быть разным. Пример с индикатором Waveform Graph, на который подается описываемый массив кластеров, демонстрируется Waveform Graph VI в варианте (Y) Multi Plot 2 graph из библиотеки `labview\examples\general\graphs\gengraph.llb`.

Кроме того, индикатор Waveform Graph принимает кластер, который состоит из начального значения *x*, приращения *x* и массива кластеров. Каждый кластер состоит из одномерного массива, в котором содержатся данные *y*. С помощью функции Bundle массивы объединяются в кластеры, а с помощью функции Build Array — полученные кластеры объединяются в общий массив. Вы можете также воспользоваться функцией Build Cluster Array, с помощью которой создаются массивы кластеров с заданными входными данными. Пример с индикатором Waveform Graph, на который подаются данные описываемого типа, демонстрируется Waveform Graph VI в варианте (Xo = 10, dX = 2, Y) Multi Plot 3 graph из библиотеки `labview\examples\general\graphs\gengraph.llb`.

На индикатор Waveform Graph можно подавать массив кластеров, каждый из которых состоит начального значения *x*, приращения *x* и массива данных *y*. Это наиболее общий способ отображения нескольких графиков, поскольку он дает возможность отображать каждую зависимость с индивидуальными значениями начальной точки и приращения по оси x. Пример с индикатором Waveform Graph, на который подается данные рассматриваемого типа, демонстрируется Waveform Graph VI в варианте (Xo = 10, dX = 2, Y) Multi Plot 1 graph из библиотеки `labview\examples\general\graphs\gengraph.llb`.

На индикатор Waveform Graph можно подавать данные динамического типа, который предназначен для работы с Express VI. Кроме данных, имеющих отношение к сигналу, данные динамического типа включают в себя атрибуты с информацией о сигнале, такие, как имя сигнала, дата и время, когда данные были измерены. Атрибуты указывают, в каком виде сигнал выводится на индикатор. Если динамический сигнал содержит данные из нескольких каналов, на индикаторе отображается график данных каждого канала, а легенда графиков и метки времени по оси x автоматически форматируется.

Отображение одной кривой на индикаторе XY Graph

На индикаторе XY Graph один график может отображаться данными трех типов. Во-первых, это кластер, который содержит массив *x* и массив *y*.

Пример с индикатором XY Graph, на который подаются данные из такого кластера, демонстрируется вариантом (X and Y arrays) Single Plot graph XY в XY Graph VI из библиотеки `labview\examples\general\graphs\gengraph.llb`.

На индикатор XY Graph также можно подавать массив точек, где в качестве точки рассматривается кластер, который содержит значение *x* и значение *y*. Пример с индикатором XY Graph, на который подаются данные этого типа, демонстрируется вариантом (Array of Pts) Single Plot graph в VI XY Graph в XY Graph VI из библиотеки `labview\examples\general\graphs\gengraph.llb`. Индикатор XY Graph принимает также массив комплексных чисел, действительная часть которых откладывается по оси x, а мнимая часть — по оси y.

Отображение нескольких кривых на индикаторе XY Graph

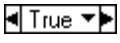
На индикатор XY Graph для отображения нескольких кривых можно подавать данные трех типов. Во-первых, это массив графиков, каждый из графиков представляет собой кластер, который содержит массив *x* и массив *y*. Пример с индикатором XY Graph, на который подаются данные этого типа, демонстрируется вариантом (X and Y arrays) Multi Plot graph в XY Graph VI из библиотеки `labview\examples\general\graphs\gengraph.llb`.

Индикатор XY Graph может также принимать массив кластеров графиков, где график представляет собой массив точек, а в качестве точки рассматривается кластер, который содержит значение *x* и значение *y*. Пример с индикатором XY Graph, на который подаются данные этого типа, демонстрируется вариантом (Array of Pts) Multi Plot graph в XY Graph VI из библиотеки `labview\examples\general\ graphs\gengraph.llb`. На индикатор XY Graph можно подавать массив кластеров графиков, каждый график представляет собой массив комплексных чисел, действительная часть которых откладывается по оси x, а мнимая часть — по оси y.

I. Case-структуры

Case-структура состоит из двух и более суб-диаграмм или фреймов.

В каждый момент времени видима только одна суб-диаграмма, и в каждый момент времени выполняется код только одного фрейма. Значение на входе селектора выбоа фреймов определяет, какая из суб-диаграмм выполняется. Case-структура подобна операторам-переключателям или условным операторам if…then…else в текстовых языках программирования.



В середине переключателя фреймов, расположенного вверху Case-структуры отображается имя состояния переключателя, соответствующее варианту выбора, а по краям переключателя находятся стрелки инкремента и декремента по фреймам.

Щелкая мышью по стрелкам инкремента и декремента, можно просмотреть доступные фреймы. Вы можете также щелкнуть по стрелке вниз рядом с именем фрейма и выбрать фрейм из выпадающего меню.

Присоедините входной параметр к терминалу селектора выбора, чтобы определять, какой фрейм должен выполняться.

На терминал селектора выбора необходимо подавать только значения целочисленного, булевского, строкового или перечислительного типов данных. Вы можете расположить терминал селектора выбора в любом месте на левой границе Case-структуры. Если тип данных терминала селектора булевский, то структура имеет только две фрейма: True и False. Если входной параметр является величиной целочисленного, строкового или перечислительного типа, структура может иметь любое количество фреймов.

Примечание: По умолчанию значения строкового типа, которые вы подаете на терминал селектора фреймов, чувствительны к регистру клавиатуры. Чтобы сделать терминал селектора нечувствительным, подключите строковый параметр к терминалу селектора, щелкните правой кнопкой мыши по границе Case-структуры и выберите из контекстного меню команду **Case Insensitive Match**.

Если вы не определите фрейм по умолчанию для обработки значений, выходящих за границы диапазона параметра, необходимо в явном виде указать все возможные значения входного параметра. Если, например, терминал селектора фреймов целочисленный, и вы определили код фреймов `1`, `2` и `3`, то следует задать фрейм по умолчанию, код которого будет выполняться, если значение входного параметра равно `4` или какому-нибудь другому значению, для которого не определен фрейм.

Примечание: Ветвь по умолчанию задавать нельзя, если вы соединили с терминалом селектора выбора булевский элемент управления. Если вы щелкните правой кнопкой мыши по переключателю фреймов, то увидите, что команда **Make This The Default Case** в контекстном меню отсутствует. Чтобы задать, код какого фрейма будет выполняться, присвойте булевскому элементу управления значение TRUE или FALSE.

Чтобы преобразовать Case-структуру в структуру Stacked Sequence, щелкните правой кнопкой мыши по Case-структуре и выберите из контекстного меню команду **Replace with Stacked Sequence**.

Щелкните правой кнопкой мыши по границе Case-структуры, чтобы добавить, скопировать или удалить фрейм, изменить порядок следования фреймов или выбрать фрейм по умолчанию.

Выбор фрейма

На рисунке 4-31 изображен VI, в котором в Case-структуре выполняется различный код в зависимости от того, какие пользователь выбрал единицы температуры: °C или °F. На вверхней блок-диаграмме на переднем плане показан фрейм True. На средней блок-диаграмме, в середине рисунка, выбирается фрейм False. Чтобы выбрать фрейм, введите значение в идентификатор переключателя фреймов или отредактируйте значения идентификаторов с помощью инструмента Labeling. После того, как вы

выбрали другой фрейм, этот фрейм отображается, как это показано на нижней блок-диаграмме (рисунок 4-31).

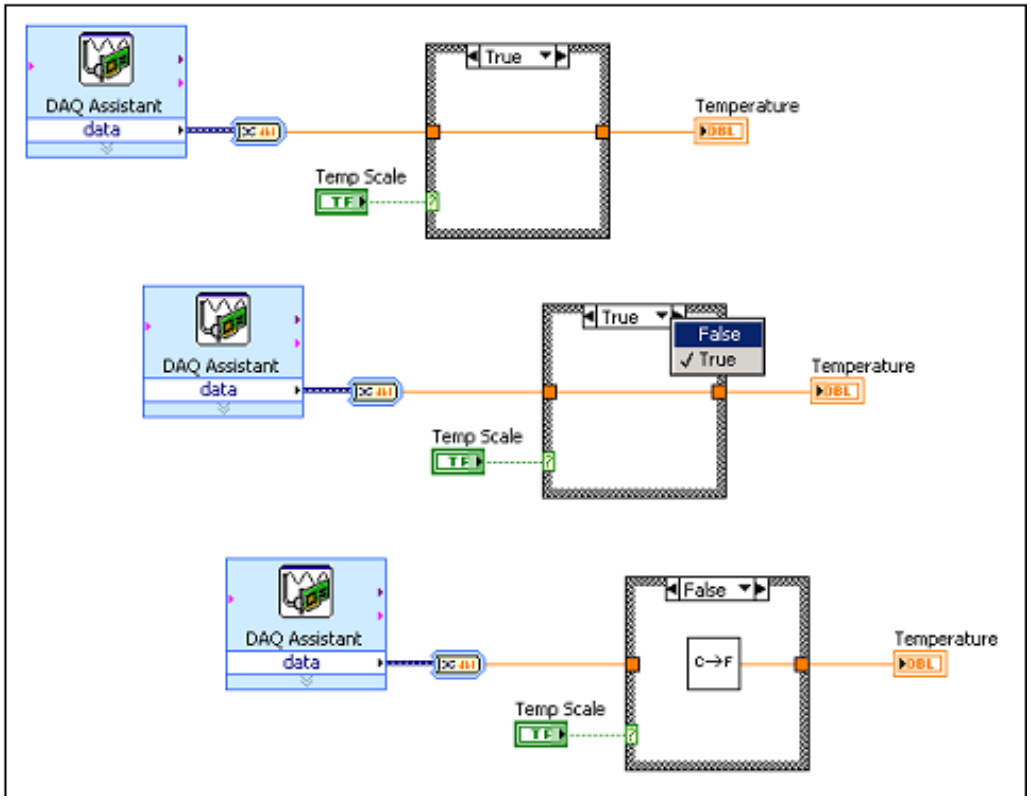


Рисунок 4-31. Выбор фрейма в Case-структуре

Если ввести значение переключателя фреймов, тип которого не совпадает с типом объекта, присоединенного к терминалу селектора выбора, введенное значение становится красного цвета. Это означает, что VI не запустится до тех пор, пока вы не удалите или не отредактируете это значение. Кроме того, поскольку возможная ошибка округления является неотъемлемой частью арифметики с плавающей точкой, вещественные числа с плавающей точкой нельзя использовать в качестве значений селектора фреймов. Если вы подаете на терминал селектора значение с плавающей точкой, LabVIEW округляет его до ближайшего целого. Если вы набираете значение с плавающей точкой внутри переключателя фреймов, оно становится красного цвета, и вам необходимо его удалить или отредактировать до того, как структура сможет выполняться.

Входные и выходные туннели

В Case-структуре можно создавать произвольное количество входных и выходных туннелей. Входные данные доступны во всех фреймах, но необязательно во всех фреймах использовать каждый из входов. Однако выходной туннель необходимо определить в каждом фрейме.

Рассмотрим следующий пример: в Case-структуре есть выходной туннель, однако, по крайней мере, в одном из фреймов к этому туннелю ничего не подключено. Если вы запустите такую структуру, LabVIEW не будет знать, какое значение вернуть на выход. LabVIEW сигнализирует о такой ошибке

белым цветом середины туннеля. Фрейм с неподключенным выходным туннелем может быть невидим в текущий момент на блок-диаграмме.

Чтобы исправить подобную ошибку, переключитесь на фрейм (фреймы), в котором (которых) выходной туннель никуда не подключен и подсоедините к этому туннелю источник данных. Можно определить состояние выходного туннеля во всех фреймах, где этот туннель никуда не подключен, значением по умолчанию для типа данных, уже подключенных к туннелю. Для этого надо щелкнуть правой кнопкой мыши по выходному туннелю и выбрать из контекстного меню команду **Use Default If Unwired.** Если выходной туннель определен во всех фреймах, он окрашен полностью.

Старайтесь не пользоваться опцией **Use Default If Unwired.** Эта опция мешает хорошо документировать блок-диаграмму и может ввести в заблуждение других программистов, работающих с таким программным кодом. Кроме того, выбор данной опции затрудняет отладку программы. Если вы все же используете опцию **Use Default if Unwired**, имейте в виду, что значение по умолчанию определяется типом данных, подключенных к туннелю. Если, например, туннель имеет булевский тип данных, то его значение по умолчанию — `FALSE`. Список значений по умолчанию для разных типов данных приведен в таблице 4-1.

Таблица 4-1. Значения по умолчанию для разных типов данных

Тип данных	Значение по умолчанию
Numeric	0
Boolean	False
String	Empty (" ")

Примеры

В следующих примерах числовые значения подаются через туннели в Case-структуру, и над ними выполняется действие сложения или вычитания в зависимости от значения, которое подается на терминал селектора выбора.

Case-структура с булевским селектором выбора

На рисунке 4-32 приведена Case-структура с булевским селектором выбора. Чтобы рисунок был более наглядным, оба фрейма показаны наложенными друг на друга.

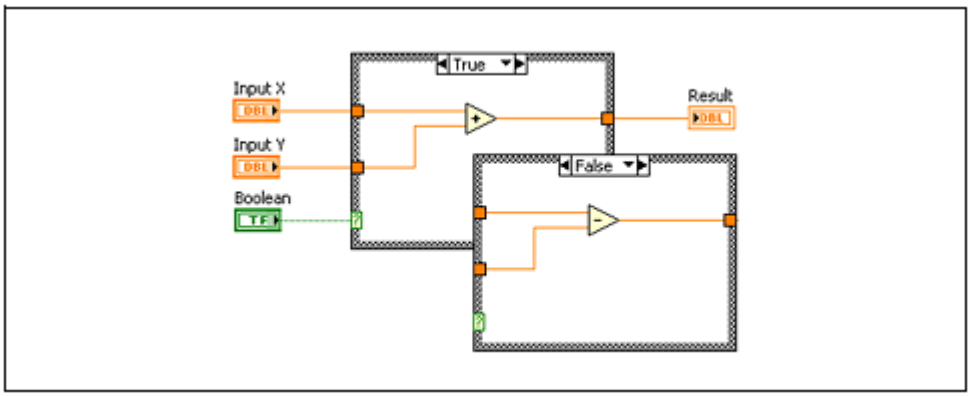


Рисунок 4-32. Case-структура с булевским селектором выбора

Если значение булевского элемента управления, подключенного к селектору, равно True, VI складывает входные числа, в противном случае VI эти числа вычитает.

Case-структура с селектором выбора типа Integer

На рисунке 4-33 приведена Case-структура с целочисленным селектором выбора.

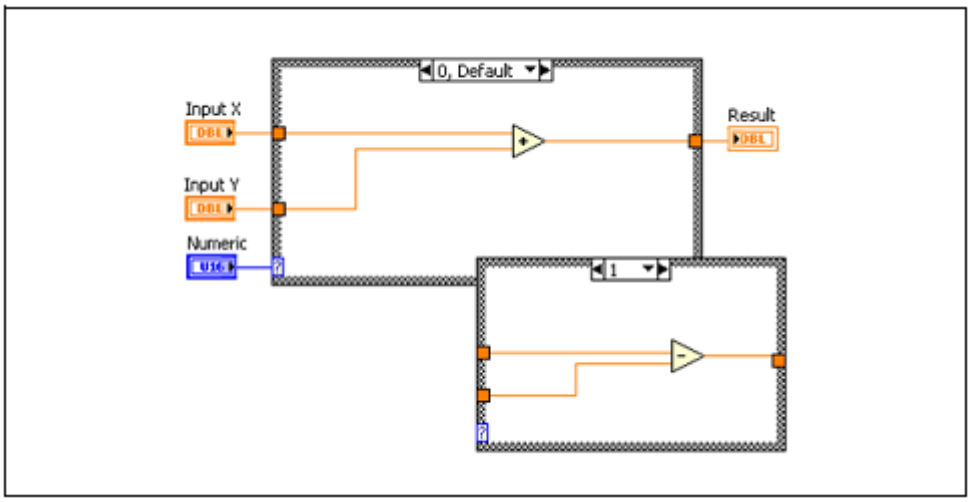


Рисунок 4-33. Case-структура с целочисленным селектором выбора

Numeric является кольцевым элементом управления текстового типа, расположенным на палитре **Text Controls**, который ставит в соответствие тексту числовые значения. Если значение, подаваемое на терминал селектора, равно `0 (add)`, VI выполняет операцию сложения над числами. Если значение терминала равно `1 (subtract)`, VI выполняет операцию вычитания. Если элемент управления Numeric выдает какое-либо другое значение, отличное от `0 (add)` или `1 (subtract)`, выполняется операция сложения, поскольку эта операция определена во фрейме по умолчанию.

Case-структура с селектором выбора типа String

На рисунке 4-34 показана Case-структура со строковым селектором выбора.

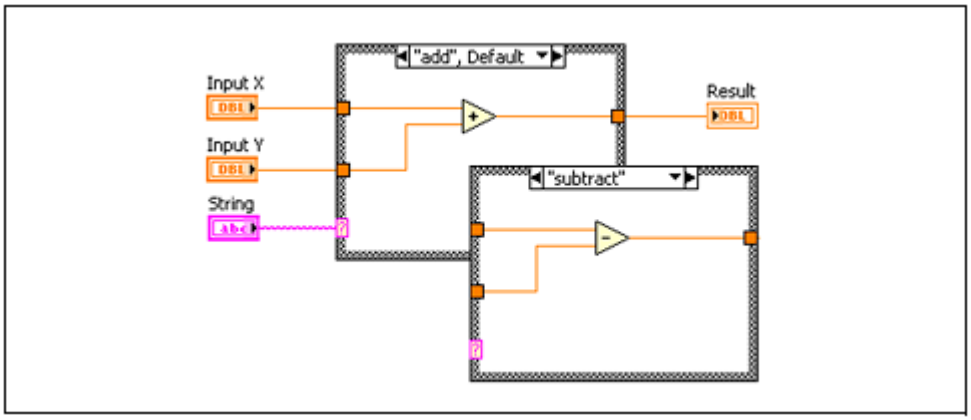


Рисунок 4-34. Case-структура со строковым селектором выбора

Если значение элемента управления **String** равно `add`, то VI выполняет операцию сложения, если оно равно `subtract`, то VI выполняет операцию вычитания.

Case-структура с селектором выбора типа Enum

На рисунке 4-35 приведена Case-структура с селектором перечислительного типа.

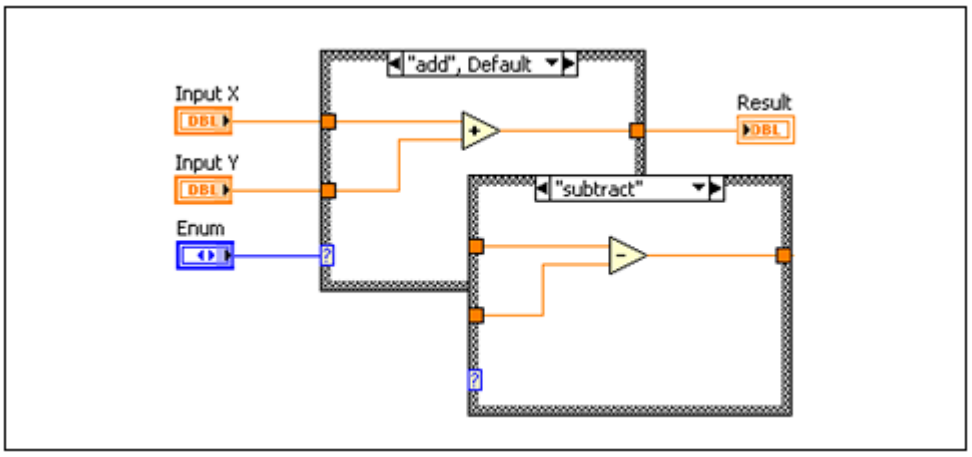


Рисунок 4-35. Case-структура с селектором перечислительного типа

Элемент управления перечислительного типа предоставляет пользователям список элементов, из которого можно выбирать нужный элемент. Такой элемент управления содержит информацию о числовых значениях и соответствующих им строковых обозначениях. Когда вы из контекстного меню Case-структуры выбираете команду **Add Case For Every Value**, в селекторе фреймов отображается обозначение каждого элемента. Case-структура выполняет код того фрейма, который соответствует выбранному текущему значению элемента управления перечислительного типа. В блок-диаграмме на рисунке 4-35, если значение элемента управления **Enum** равно `add`, VI выполняет операцию сложения, если значение этого элемента управления равно `subtract`, VI выполняет операцию вычитания.

Применение Case-структур для обработки ошибок

На рисунках 4-36, 4-37 приведены примеры, где выбор фрейма Case-структуры осуществляется кластером ошибки.

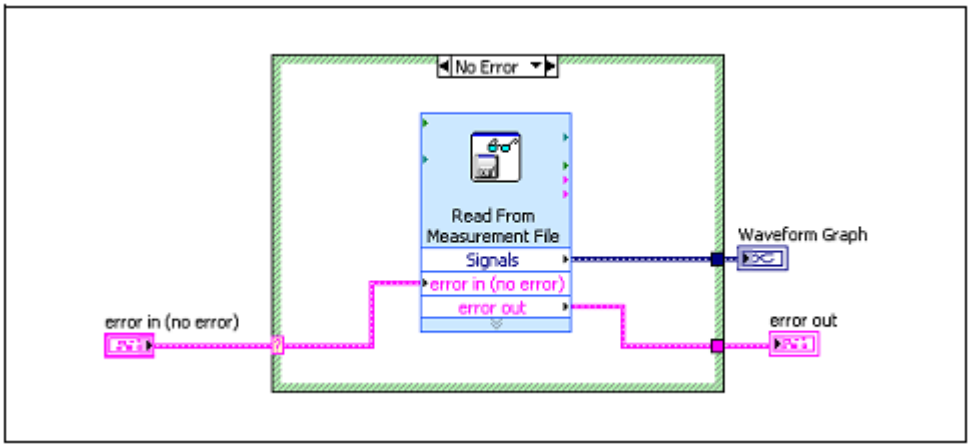


Рисунок 4-36. Фрейм No Error Case-структуры

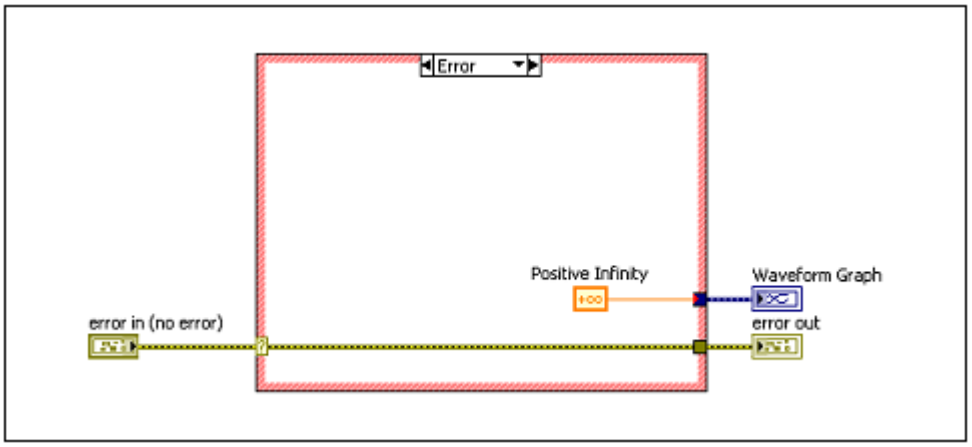


Рисунок 4-37. Фрейм Error Case-структуры

Когда вы присоедините кластер ошибки к терминалу селектора Case-структуры, переключатель фреймов позволяет выбрать один из двух фреймов: `Error` и `No Error`, причем граница структуры меняет цвет, она становится красной для ветви `Error` и зеленой для ветви `No Error`. При возникновении ошибки Case-структура выполняет код суб-диаграммы фрейма `Error`.

При подключении кластера ошибки к терминалу селектора выбора, Case-структура воспринимает в кластере только булевский элемент **status**.

Самопроверка: короткий тест

1. Что является идентификатором элемента управления или индикатора на блок-диаграмме?
   a. Заголовок
   b. Местоположение
   c. Метка
   d. Значение

2. Какая структура обязательно выполняется хотя бы один раз?
   a. Цикл While
   b. Цикл For

3. Какой объект доступен только на блок-диаграмме
   a. Элемент управления
   b. Константа
   c. Индикатор
   d. Панель подключения

4. Как называется механическое свойство булевского элемента управления, если он переключается из состояния False в состояние True при щелчке по нему мышью и остается в этом состоянии до тех пор, пока вы не отпустите кнопку мыши и LabVIEW не прочитает состояние элемента?
   a. Switch until released
   b. Switch when released
   c. Latch until released
   d. Latch when released

Самопроверка: короткий тест

1. Что является идентификатором элемента управления или индикатора на блок-диаграмме?
   a. Заголовок
   b. Местоположение
   c. Метка
   d. Значение

2. Какая структура обязательно выполняется хотя бы один раз?
   a. Цикл While
   b. Цикл For

3. Какой объект доступен только на блок-диаграмме
   a. Орган управления
   b. Константа
   c. Индикатор
   d. Панель подключения

4. Как называется механическое свойство булевского элемента управления, если он переключается из состояния False в состояние True при щелчке по нему мышью и остается в этом состоянии до тех пор, пока вы не отпустите кнопку мыши и LabVIEW не прочитает состояние элемента?
   a. Switch until released
   b. Switch when released
   c. Latch until released
   d. Latch when released

Заметки

5. Связываемые данные

Иногда полезно группировать связанные друг с другом данные. Используйте массивы и кластеры для группировки связываемых данных в LabVIEW. Массивы объединяют в единую структуру данные одного типа, а кластеры – разных типов. Используйте определители типа для определения пользовательских массивов и кластеров. В данной лекции рассматриваются массивы, кластеры, определители типа, а также приложения, где может быть полезно их применение.

План занятия

A. Массивы

B. Кластеры.

C. Определители типа

A. Массивы

Массив состоит из элементов и характеризуются размерностью. Элементы – это данные, из которых состоит массив. Размерность – это длина, высота или глубина массива. Массив может иметь одну или более размерностей и содержать до 2^{31}-1 элементов в каждом измерении, если позволяет память.

Массивы можно создавать из чисел, булевских элементов, строк, путей, сигналов и кластеров. Рассмотрим применение массивов при работе с набором однотипных данных и при выполнении повторяющихся вычислений. Массивы идеально подходят для хранения сигналов или данных, сгенерированных в циклах, в каждой итерации которых формируется один элемент массива.

Примечание: индексация массивов в LabVIEW начинается с нуля. Индекс первого элемента массива равен нулю независимо от размера массива

Ограничения

Вы не можете создать массив массивов. Однако вы можете использовать многомерный массив или создать массив кластеров, в котором каждый кластер содержит один или более массивов. Также вы не можете создать массив элементов управления субпанелями, закладками, элементов управления .NET, ActiveX, графических индикаторов типа Chart или XY с

несколькими графиками. Обратитесь к разделу *Кластеры* настоящей лекции для получения дополнительной информации о кластерах.

Примером простого массива может послужить текстовый массив, в котором перечислены наименования девяти планет нашей Солнечной системы. В LabVIEW он представлен одномерным массивом строк с девятью элементами.

Элементы в массиве упорядочены. Для получения доступа к любому элементу используется индекс. Индексация начинается с нуля, то есть индекс лежит в диапазоне от 0 до $n - 1$, где n - число элементов в массиве. Например, для 12 месяцев в году $n = 12$, а индекс принимает значения от 0 to 11. Третий месяц, март, имеет индекс 2.

На рисунке 5-1 показан пример массива чисел. Первый показанный элемент массива (3.00) имеет индекс 1, а второй элемент (1.00) - индекс 2. Элемент с индексом 0 не виден на рисунке, поскольку переключателем индекса выбран элемент 1. Элемент, на который указывает переключатель индекса, всегда отображается в верхнем левом углу индикатора элементов.

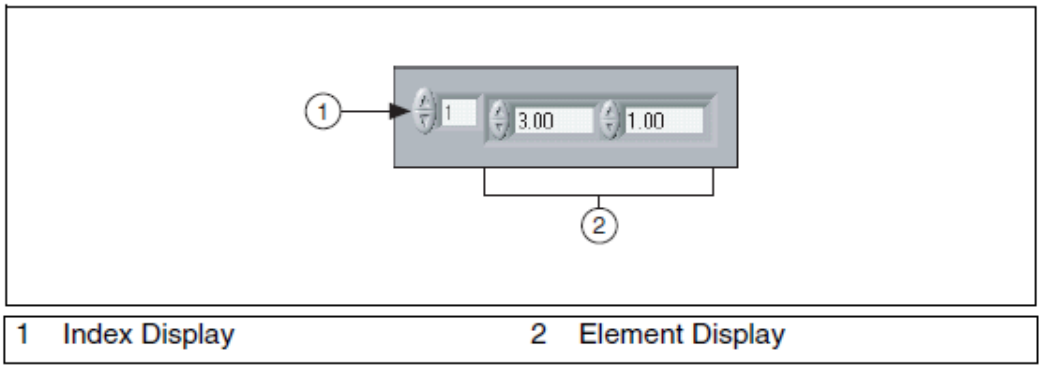


Рисунок 5-1. Массив числовых элементов управления

Создание массивов элементов управления и индикаторов

Массивы элементов управления и индикаторов на лицевой панели создаются путем установки контейнера массива на лицевую панель, как показано на рисунке ниже, и перетаскивания в него объекта данных или элемента. Элемент массива может быть числом, булевским элементом, строкой, путем, ссылкой или кластером элементов управления или индикаторов.

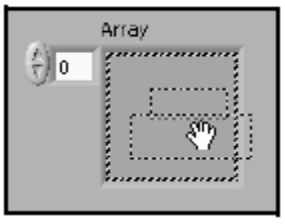


Рисунок 5-2. Помещение числового элемента управления в контейнер массива

Если вы попытаетесь перетащить в контейнер массива неразрешенный элемент управления или индикатор, у вас это не получится.

Вы должны поместить объект в контейнер массива, прежде чем сможете использовать массив на блок-диаграмме. В противном случае терминал массива останется черным, с пустыми скобками, и не будет иметь связанного с ним типа данных.

Двумерные массивы

В предыдущем примере рассматривались одномерные массивы. В двумерном массиве элементы хранятся в сетке. Для определения местоположения элементов используется индекс строки и индекс столбца, нумерация которых начинается с нуля. На рисунке 5-3 показан двумерный массив из 8 столбцов по 8 строк, который содержит 64 элемента.

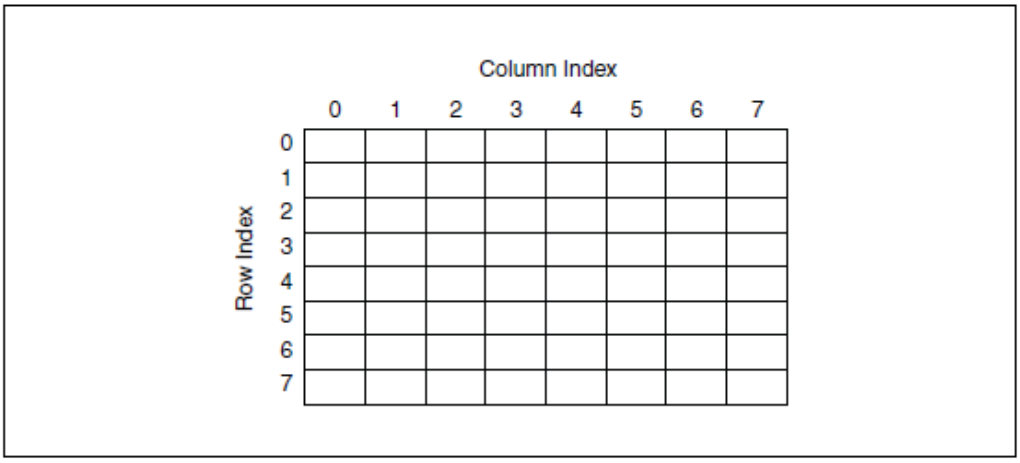


Рисунок 5-3. Двумерный массив

Для создания на лицевой панели многомерного массива щелкните правой кнопкой по переключателю индексов и выберите из контекстного меню **Add Dimension.** Вы можете также растягивать переключатель индексов, пока не получите желаемую размерность.

Инициализация массивов

Вы можете инициализировать массив или оставить его неинициализированным. В инициализированном массиве определено количество элементов в каждой размерности и содержимое каждого элемента. Неинициализированный массив имеет определенную размерность, но не содержит элементов. На рисунке 5-4 показан неинициализированный двумерный массив элементов управления. Обратите внимание, что все элементы выглядят блеклыми. Это указывает на то, что массив неинициализирован.

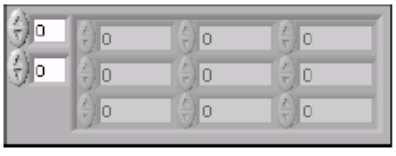


Рисунок 5-4. Двумерный неинициализированный массив

В массиве на рисунке 5-5 проинициализированы 6 элементов. После инициализации элемента в строке двумерного массива остальные элементы строки инициализируются значением по умолчанию для этого типа данных. Например, если вы введете 4 в элемент в первом столбце, третьей строке массива на рисунке 5-5, элементы во втором и третьем столбце третьей строки автоматически заполнятся нулями.

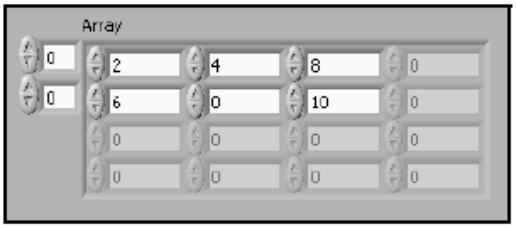


Рисунок 5-5. Инициализированный двумерный массив с шестью элементами

Создание массивов констант

Для создания на блок-диаграмме массива констант выберите в палитре функций константу массива, поместите на блок-диаграмму контейнер массива и поместите строковую, числовую, булевскую или кластерную константу в контейнер. Вы можете использовать массивы констант для хранения неизменных данных или в качестве базы для сравнения с другим массивом.

Автоиндексация массива на входе

При подключении массива ко входу или выходу цикла For или While, вы можете связать каждую итерацию цикла с элементом в этом массиве, разрешив автоиндексацию туннеля. Изображение туннеля изменится с закрашенного квадрата на показанное слева. Щелкните по туннелю правой кнопкой и выберите из контекстного меню **Enable Indexing** или **Disable Indexing** для изменения состояния туннеля.

Входы массива

Если вы разрешите автоиндексацию массива, подключенного ко входному терминалу цикла For, LabVIEW установит счетчик задания количества итераций цикла равным размеру массива, поэтому вам не нужно подключать терминал задания количества итераций. Поскольку вы можете использовать циклы For для обработки массивов по одному элементу в каждой итерации, по умолчанию LabVIEW разрешает автоиндексацию всех массивов, подключенных к циклу For. Вы можете отключить автоиндексацию, если вам не нужно обрабатывать массив поэлементно.

На рисунке 5-6 цикл For выполняется столько раз, сколько элементов в массиве. Обычно, если терминал задания количества итераций цикла For не

подключен, стрелка на кнопке запуска программы "сломана". Однако в данном случае этого не происходит.

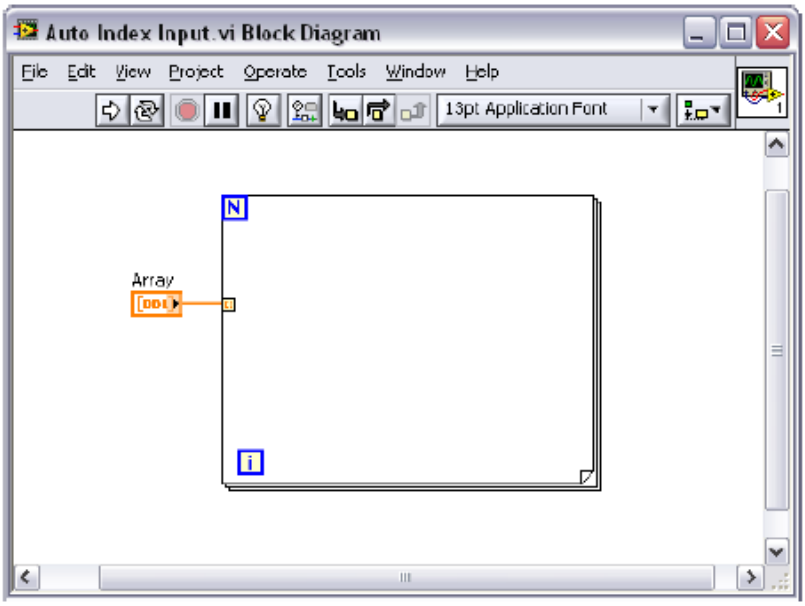


Рисунок 5-6. Массив используется для задания количества итераций

Если вы разрешите автоиндексацию для нескольких туннелей или подключите терминал задания количества итераций цикла Count, количество итераций будет равно наименьшему из заданных значений. Например, если в цикл входят два автоиндексированных массива с 10 и 20 элементами соответственно, а счетчик задания количества итераций цикла равен 15, цикл выполнится только 10 раз, проиндексировав все элементы первого массива и только первые 10 элементов второго.

Выходные массивы

При автоиндексации в массиве выходного туннеля новый элемент в выходной массив записывается в каждой итерации цикла. Таким образом, размер автоидексированных выходных массивов всегда равен количеству итераций.

Провод, соединяющий автоиндексируемый на границе цикла выходной туннель с индикатором массива, становится толще, а выходной туннель содержит квадратные скобки, обозначающие массив, как показано на рисунке 5-7.

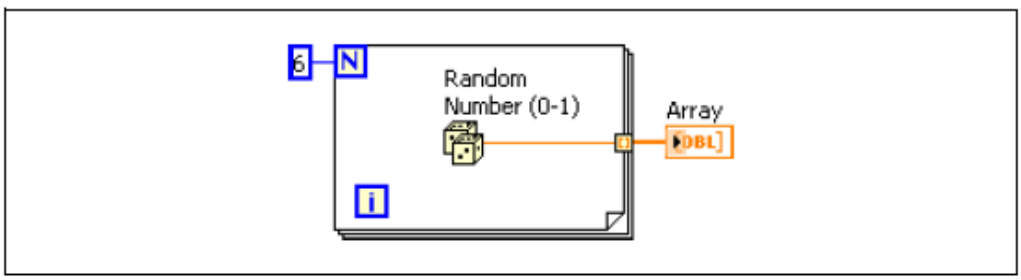


Рисунок 5-7. Автоиндексируемый выход

Щелкните правой кнопкой мыши по туннелю на границе цикла и выберите из контекстного меню **Enable Indexing** или **Disable Indexing** для разрешения или запрещения автоиндексации. По умолчанию в циклах While автоиндексация отключена.

Например, вы можете отключить автоиндексацию, если вам нужно только последнее переданное в туннель значение.

Создания двумерных массивов

Вы можете использовать два вложенных цикла For для создания двумерного массива. Во внешнем цикле For создаются элементы строки, а во внутреннем – столбца, как показано на рисунке 5-8.

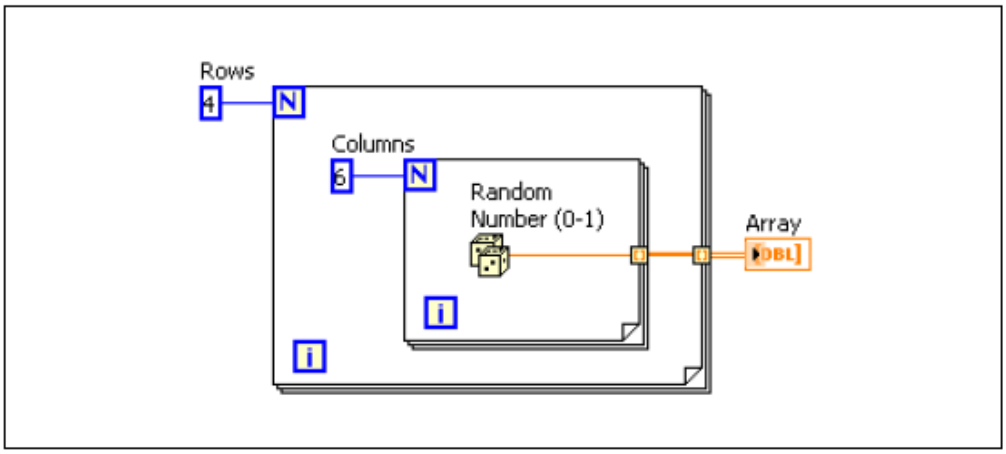


Рисунок 5-8. Создание двумерного массива

B. Кластеры

В кластеры группируют элементы данных разных типов. Примером кластера является кластер ошибок LabVIEW, который объединяет булевский, числовой и строковый элементы. Кластер аналогичен записи или структуре в текстовых языках программирования.

Объединение нескольких элементов данных в кластер уменьшает загромождение проводниками блок-диаграммы, а также количество терминалов панели подключения subVI. Максимальное количество терминалов панели подключения равно 28. Если на лицевой панели вашего VI находится более 28 элементов управления и индикаторов, которые вы хотите передать в другой VI, сгруппируйте часть из них в кластер и назначьте кластеру терминал на панели подключения.

Большинство терминалов кластеров и связанные с ними проводники окрашены на блок-диаграмме в розовый цвет. Терминалы кластеров ошибок и связанные с ними проводники окрашены на блок-диаграмме в горчичный цвет Терминалы кластеров, состоящих из числовых компонентов и называемых иногда точками, а также связанные с ними проводники окрашены на блок-диаграмме в коричневый цвет. Вы можете подключить коричневые числовые кластеры к функциям обработки чисел, например,

Add или Square Root, для выполнения операций одновременно над всеми элементами кластера.

Порядок элементов кластера

Хотя и в кластере, и в массиве элементы упорядочены, вы должны разделять все элементы кластера одновременно, используя функцию Unbundle. Вы можете использовать функцию Unbundle By Name для разделения элементов кластера по имени. При использовании функции Unbundle by Name у каждого элемента кластера должна быть метка.

Кластеры отличаются от массивов фиксированным размером. Подобно массиву, кластер является либо элементом управления, либо индикатором. Кластер не может содержать элементы управления и индикаторы одновременно.

Создание кластеров элементов управления и индикаторов

Для создания на лицевой панели кластера элементов управления или индикаторов поместите на лицевую панель контейнер кластера и перетащите в контейнер объекты данных или элементы, которые могут представлять данные разных типов - числового, булевского, строкового, пути, ссылок, массивы или кластеры элементов управления или индикации.

Изменяйте размер контейнера кластера с помощью курсора мыши в процессе установки элементов в контейнер.

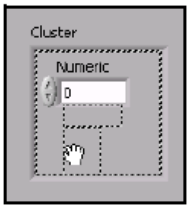


Рисунок 5-9. Создание кластера элементов управления

На рисунке 5-10 показан пример кластера из трех элементов управления: строкового, булевского переключателя и числового элемента. Кластер может включать либо элементы управления, либо индикаторы, но не оба типа объектов одновременно.

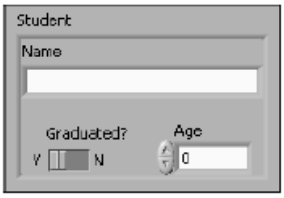


Рисунок 5-10. Пример кластера элементов управления

Создание кластеров констант

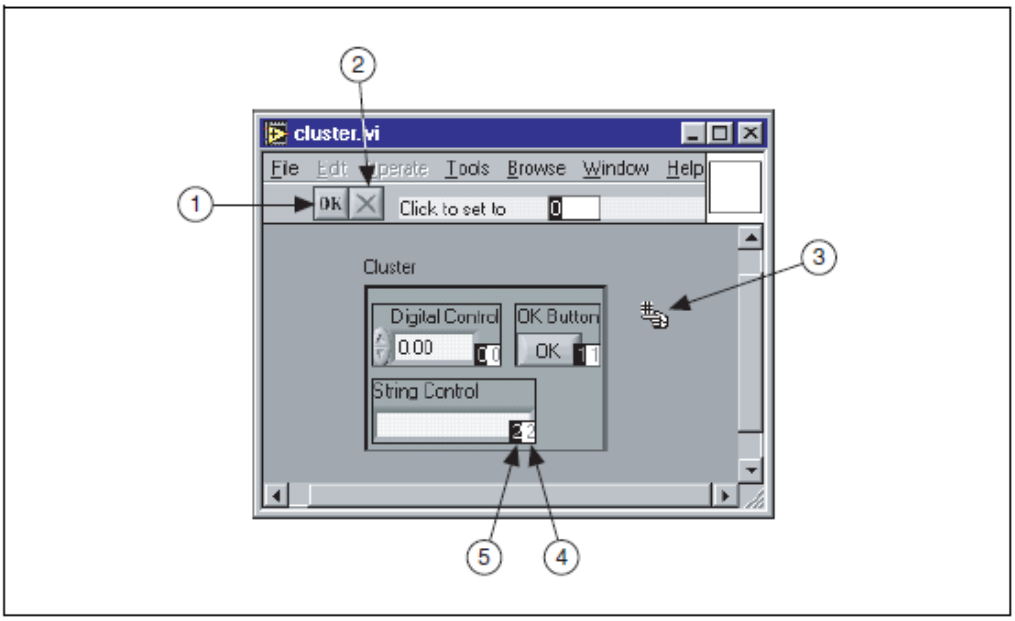
Для создания на блок-диаграмме кластера констант выберите кластер констант в палитре функций, поместите контейнер кластера на блок-диаграмму и добавьте в нее строковую, числовую, булевскую константу или кластер констант. Вы можете использовать кластер констант для хранения постоянных данных или в качестве базы для сравнения с другим кластером.

Если у вас есть кластер элементов управления или индикаторов на лицевой панели, и вы хотите создать кластер констант, содержащий такие же элементы на блок-диаграмме, вы можете или перетащить этот кластер с лицевой панели на блок-диаграмму, или щелкнуть правой кнопкой мыши по кластеру на лицевой панели и выбрать из контекстного меню **Create» Constant.**

Упорядочение элементов в кластере

Элементы кластера имеют логический порядок, не связанный с их местом в контейнере. Первый элемент, помещенный вами в кластер – элемент 0, второй – элемент 1, и так далее. При удалении элемента порядок перестраивается автоматически. Порядок элементов кластера определяет очередность, в котором элементы появляются как терминалы функций Bundle и Unbundle на блок-диаграмме. Вы можете просматривать и изменять порядок элементов кластера, щелкнув правой кнопкой мыши по границе кластера и выбрав из контекстного меню **Reorder Controls In Cluster.**

Панель инструментов и кластер изменятся, как показано на рисунке 5-11.



1	Confirm Button	3	Cluster Order Cursor	5	New Order
2	Cancel Button	4	Current Order		

Рисунок 5-11. Изменение порядка элементов кластера

Белое окошко на каждом элементе показывает текущий номер элемента в кластере. В черном окошке показывается его новый номер. Для задания номера элемента кластера введите новый номер в текстовое окно **Click to set to** и щелкните по элементу. Порядковый номер этого элемента станет равным введенному, а номера других элементов подстроятся. Сохраните изменения, нажав кнопку **Confirm** на панели инструментов, либо вернитесь к исходному порядку, нажав кнопку **Cancel**.

Использование функций для работы с кластерами

Используйте функции кластера для создания и манипулирования кластерами. Например, вы можете выполнять следующие операции:

- Извлекать отдельные элементы из кластера
- Добавлять отдельные элементы в кластер
- Разделять кластер на отдельные элементы

Используйте функцию Bundle для сборки кластера, функции Bundle и Bundle by Name - для модификации кластера, а функции Unbundle и Unbundle by Name для разборки кластера.

Вы можете также помещать функции Bundle, Bundle by Name, Unbundle и Unbundle by Name на блок-диаграмму, щелкнув по терминалу кластера правой кнопкой мыши и выбрав из контекстного меню **Cluster, Class & Variant Palette.** Функции Bundle и Unbundle автоматически содержат нужное количество терминалов. Функции Bundle by Name и Unbundle by Name появляются с первым элементом кластера. Используйте инструмент Positioning для изменения размеров функций Bundle by Name и Unbundle by Name для отображения других элементов кластера.

Сборка кластеров

Используйте функцию Bundle для сборки кластера из отдельных элементов или изменения значений отдельных элементов существующего кластера, не задавая новых значений для всех элементов. Используйте инструмент Positioning для изменения размера функции, или щелкните правой кнопкой мыши по входу элемента и выберите из контекстного меню **Add Input.**

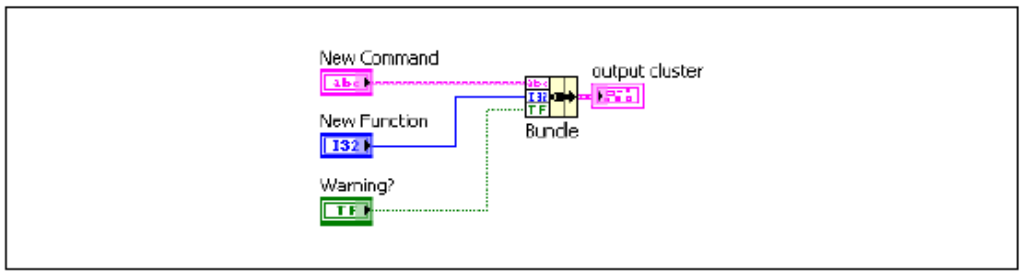


Рисунок 5-12. Сборка кластера на блок-диаграмме

Изменение кластера

При подключении входа кластера вы можете подключать только те элементы, которые хотите изменить. Например, кластер Input Cluster на рисунке 5-13 содержит три элемента управления.

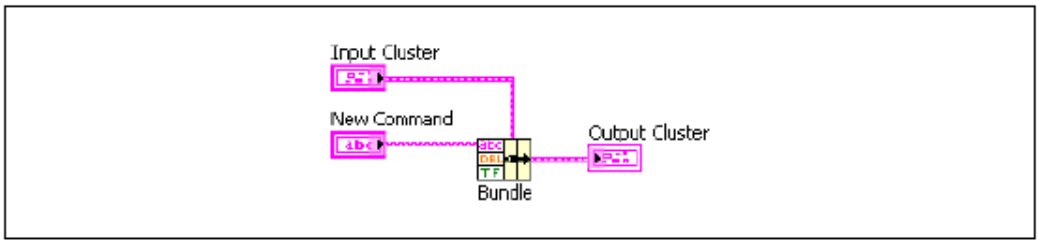


Рисунок 5-13. Использование функции Bundle для изменения кластера

Если вы знаете порядок элементов в кластере, то можете использовать функцию Bundle для изменения значения **Command**, подключив элементы, как показано на рисунке 5-13.

Кроме того, вы можете использовать функцию Bundle by Name для замены или доступа к именованным элементам существующего кластера. Функция Bundle by Name работает аналогично функции Bundle, но вместо обращения к элементам кластера согласно их порядку, она обращается к элементам кластера по их именам. С этой функцией можно работать с элементами кластера только по их именам. Количество входов функции не обязательно должно соответствовать количеству элементов выходного кластера (**output cluster)**.

Щелкните инструментом Operating по входному терминалу и выберите элемент из выпадающего списка. Вы можете также щелкнуть по входу правой кнопкой мыши и выбрать элемент из контекстного меню **Select Item**.

На рисунке 5-14 вы можете использовать функцию Bundle by Name для обновления значений **Command** и **Function** значениями **New Command** и **New Function.**

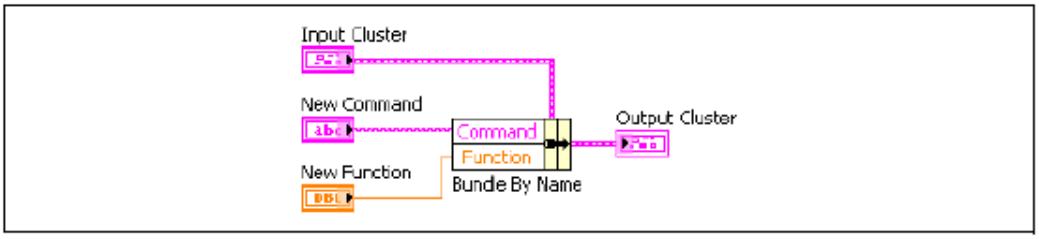


Рисунок 5-14. Использование функции Bundle By Name для модификации кластера

Используйте функцию Bundle by Name для структур данных, которые могут измениться в ходе дальнейшего проектирования. При добавлении нового элемента в кластер или изменении порядка элементов вам не придется переподключать функцию Bundle by Name, потому что имена остаются действительными.

Разборка кластеров

Используйте функцию Unbundle для разделения кластеров на отдельные элементы. Используйте функцию Unbundle by Name для получения элементов кластера с заданными именами. Количество выходных терминалов не зависит от количества элементов входного кластера.

Щелкните инструментом Operating по выходному терминалу и выберите элемент из выпадающего списка. Вы можете также щелкнуть по элементу правой кнопкой мыши и выбрать из контекстного меню **Select Item.**

Например, при использовании функции Unbundle с кластером на рисунке 5-15 у нее появятся четыре выходных терминала, соответствующие четырем элементам кластера. Вы должны знать порядок элементов кластера, чтобы связать правильный булевский терминал неразобранного кластера с соответствующим переключателем в кластере. В этом примере элементы упорядочены сверху вниз, начиная с элемента 0. С функцией Unbundle by Name вы можете использовать любое число выходных терминалов и получать доступ к отдельным элементам в любом порядке.

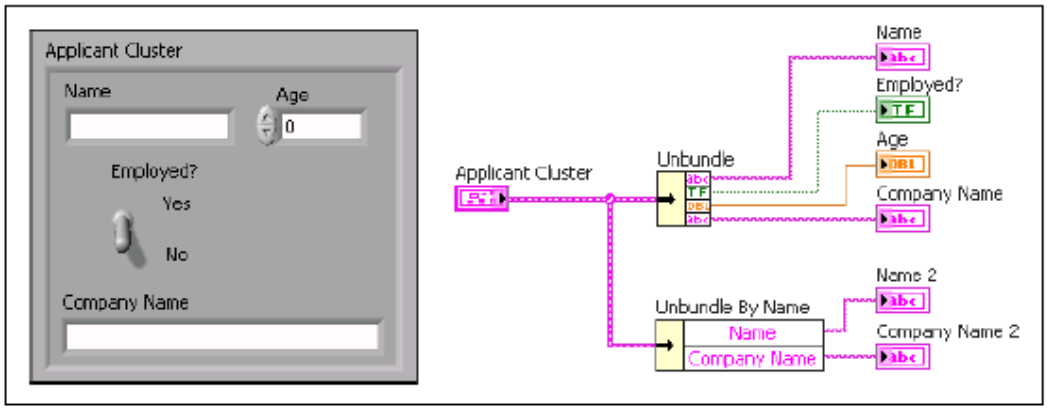


Рисунок 5-15. Функции Unbundle и Unbundle By Name

Кластеры ошибок

В LabVIEW есть специальный кластер, называемый кластером ошибок. LabVIEW использует кластеры ошибок для передачи информации об ошибках. Кластеры ошибок содержит следующие элементы:

- **status** — булевский элемент, принимающий значение TRUE при возникновении ошибки.
- **code** — 32-разрядное целое число со знаком, служащее для идентификации ошибки.
- **source** — строка, определяющая место происхождения ошибки.

Для получения дополнительной информации о кластерах ошибок обратитесь к лекции 3, *«Поиск ошибок и отладка VI»,* данного руководства, а также к разделу *Handling Errors* справки LabVIEW.

C. Определители типа

Вы можете использовать определители типа для пользовательских массивов и кластеров.

Пользовательские элементы управления

Используйте пользовательские элементы управления и индикаторы для расширения имеющегося набора объектов лицевой панели. Вы можете создавать компоненты пользовательского интерфейса, которые отличаются внешним видом от встроенных в LabVIEW элементов управления и индикаторов. Вы можете сохранить созданный вами пользовательский элемент управления или индикатор в папке или библиотеке LLB и использовать его на других лицевых панелях. Вы можете также создать иконку для пользовательского элемента управления или индикатора и добавить ее в палитру **Controls.**

Ознакомьтесь с ограничениями и рекомендациями, прежде чем начнете создавать пользовательские элементы управления и индикаторы.

Обратитесь к разделу *Creating Custom Controls, Indicators, and Type Definitions* справки LabVIEW для получения дополнительной информации о создании и использовании пользовательских элементов управления и определителей типа.

Используйте окно Control Editor (Редактор элементов управления) для создания специализированных элементов управления и индикаторов. Например, вы можете изменить размер, цвет и относительное положение деталей элемента управления или индикатора и импортировать в них изображения.

Вы можете открыть окно Control Editor следующими способами:

- Щелкните правой кнопкой мыши по элементу правления или индикатору на лицевой панели и выберите из контекстного меню **Advanced»Customize**.
- Используйте инструмент Positioning для выбора элемента управления или индикатора на лицевой панели и выберите **Edit»Customize Control**.
- Используйте диалоговое окно **New**.

Окно Control Editor появляется с выбранным объектом лицевой панели. Control Editor имеет два режима: режим редактирования и режим модификации.


На панели инструментов окна Control Editor указывается, в каком режиме вы работаете. Окно Control Editor открывается в режиме редактирования. Нажмите кнопку **Change to Edit Mode** для перехода в режим редактирования. Нажмите кнопку **Change to Customize Mode** для перехода в режим модификации. Также вы можете переключаться между режимами, выбирая в меню **Operate»Change to Customize Mode** или **Operate»Change to Edit**

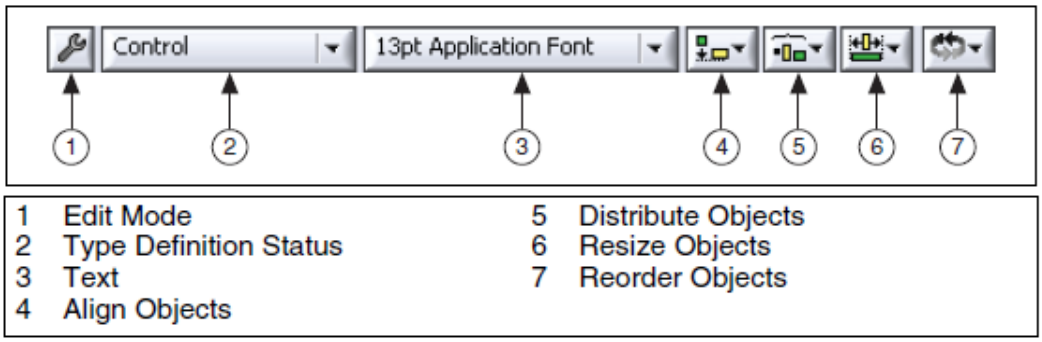
Mode.

Используйте режим редактирования для изменения размера или цвета элемента управления или индикатора и выбора настроек из его контекстного меню, как делаете это в режиме редактирования на лицевой панели.

Используйте режим модификации для выполнения значительных изменений элементов управления или индикаторов путем изменения их отдельные частей.

Режим редактирования

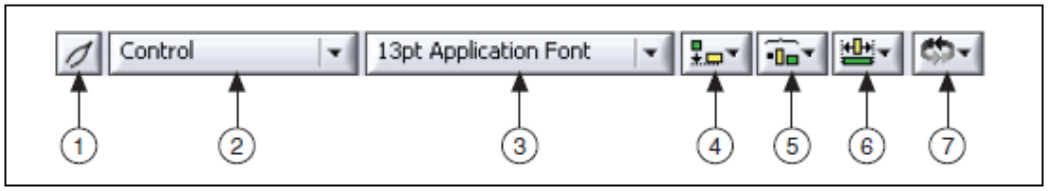
В режиме редактирования вы можете щелкнуть правой кнопкой по элементу управления и изменить его настройки, как и в среде LabVIEW.



1	Edit Mode	5	Distribute Objects
2	Type Definition Status	6	Resize Objects
3	Text	7	Reorder Objects
4	Align Objects		

Режим модификации

В режиме модификации вы можете перемещать отдельные части элемента управления относительно друг друга. Для получения списка доступных частей в режиме модификации, выберите меню **Window»Show Parts Window.**



1	Customize Mode	5	Distribute Objects
2	Type Definition Status	6	Resize Objects
3	Text	7	Reorder Objects
4	Align Objects		

Один из способов создания пользовательских элементов лицевой панели заключается в изменении его статуса определителя типа. Вы можете сохранить элемент как элемент лицевой панели, как определитель типа или как абсолютный определитель типа, в зависимости от выбора параметра **Type Def. Status**. Вариант control (элемент лицевой панели) – такой же элемент управления, как тот, что вы выбираете из палитры **Controls.** Вы

можете изменять его, как вам требуется, и каждая созданная вами копия сохранит свои уникальные свойства.

Сохранение пользовательских элементов управления

После создания пользовательского элемента лицевой панели вы можете сохранить его для дальнейшего использования. Сохраняемые на диск элементы лицевой панели по умолчанию имеют расширение `.ctl`.

Вы можете также использовать Control Editor для сохранения элементов лицевой панели с собственными настройками по умолчанию. Например, вы может использовать Control Editor для изменения настроек графического индикатора, сохранить его, и вызывать в других VI.

Определители типа

Используйте определители типа и абсолютные определители типа для привязки всех экземпляров пользовательского элемента управления и индикатора к сохраненному файлу пользовательского элемента управления или индикатора. Вы можете применить изменения ко всем экземплярам пользовательского элемента управления или индикатора, изменив лишь сохраненный файл. Это полезно, если вы используете один и тот же пользовательский элемент управления или индикатор в нескольких VI.

Когда вы помещаете пользовательский элемент управления или индикатор в VI, между сохраненным вами пользовательским элементом управления или индикатором и экземпляром пользовательского элемента управления или индикатора в VI нет связи. Каждый экземпляр пользовательского элемента управления или индикатора является отдельным, независимым экземпляром. Таким образом, изменения, производимые с файлом пользовательского элемента управления или индикатора, не затрагивают VI, где уже используется этот пользовательский элемент управления или индикатор. Если вы хотите привязать экземпляр пользовательского элемента управления или индикатор к соответствующему ему файлу, сохраните пользовательский элемент управления или индикатор как определитель типа или абсолютный определитель типа. Все экземпляры определителя типа или абсолютного определителя типа связаны с оригинальным файлом, из которого вы их создали.

При сохранении пользовательского элемента управления или индикатора как определителя типа или абсолютного определителя типа, все примененные к нему изменения типа данных затрагивают все экземпляры определителя типа или абсолютного определителя типа во всех VI, где они встречаются. Косметические изменения, примененные к абсолютному определителю типа, также влияют на все экземпляры абсолютного определителя типа на лицевой панели.

Определители типа идентифицируют правильный тип данных для каждого экземпляра пользовательского элемента управления или индикатора. Когда тип данных определителя типа меняется, все экземпляры определителя типа

обновляются автоматически. Другими словами, тип данных экземпляров определителя типа меняется в каждом VI, где применяется этот определитель типа. Однако, поскольку определители типа идентифицируют только тип данных, обновляются только значения, которые являются частью типа данных. Например, в числовых элементах управления диапазон значений не является частью типа данных. Поэтому определители типа числовых элементов управления не определяют диапазон значений экземпляров определителя типа. Также, поскольку наименования элементов в элементах управления ring не определяют тип данных, изменение наименований элемента управления ring в определителе типа не изменят наименования в экземпляре определителя типа. Однако если вы измените наименования в определителе типа элемента enumerated, экземпляры обновятся, потому что наименования являются частью этого типа данных. У экземпляра определителя типа могут быть свои уникальные заголовок, метка, описание, строка подсказки, значение по умолчанию, размер, цвет или стиль элемента управления или индикатора, например, регулятор вместо переключателя.

При изменении типа данных определителя типа LabVIEW преобразует прежнее значение по умолчанию всех экземпляров определителя типа к новому типу данных, если это возможно. LabVIEW не может сохранить значение по умолчанию экземпляра, если тип данных меняется на несовместимый, например, если числовой элемент управления или индикатор заменяется строковым. Когда тип данных определителя типа изменяется на несовместимый с предыдущим, LabVIEW устанавливает значение по умолчанию для экземпляра равным значению по умолчанию, заданному вами в файле `.ctl`. Если вы не задали значение по умолчанию, LabVIEW использует значение по умолчанию для этого типа данных. Например, при смене определителя типа с числового на строковый, LabVIEW заменит все значения по умолчанию, связанные с прежним числовым типом данных, пустыми строками.

Абсолютные определители типа

Абсолютный определитель типа принуждает экземпляр быть полностью идентичным абсолютному определителю типа, за исключением заголовка, метки, описания, строки подсказки и значения по умолчанию. Как и в определителях типа, тип данных абсолютного определителя типа остается неизменным везде, где вы использовали абсолютный определитель типа. Абсолютные определители типа также определяют другие значения, например, диапазон значений числовых элементов управления и наименования элементов типа ring. Единственные свойства сервера VI, доступные для абсолютного определителя типа – те, которые влияют на внешний вид элемента управления или индикатора, например, Visible, Disabled, Key Focus, Blinking, Position и Bounds.

Вы не можете запретить автоматическое обновление экземпляра абсолютного определителя типа, если только не удалите связь между экземпляром и абсолютным определителем типа.

Определители типа и абсолютные определители типа создают пользовательский элемент управления, используя кластер из нескольких

элементов управления. Если вам нужно добавить новый элемент управления и передать новое значение в каждый subVI, вы можете добавить новый элемент управления в пользовательский кластер элементов управления. При этом отпадет необходимость добавлять новый элемент управления на лицевую панель каждого subVI и добавлять новые проводники и терминалы.

Самопроверка: короткий тест

1. Вы можете создать массив массивов.

   a. Да

   b. Нет

2. Два входных массива подключены к циклу For. В обоих туннелях включена автоиндексация. В одном массиве 10 элементов, в другом – 5. К терминалу задания количества итераций подключено значение 7, как показано на рисунке 5-16. Какое значение будет показывать индикатор Iterations после запуска VI?

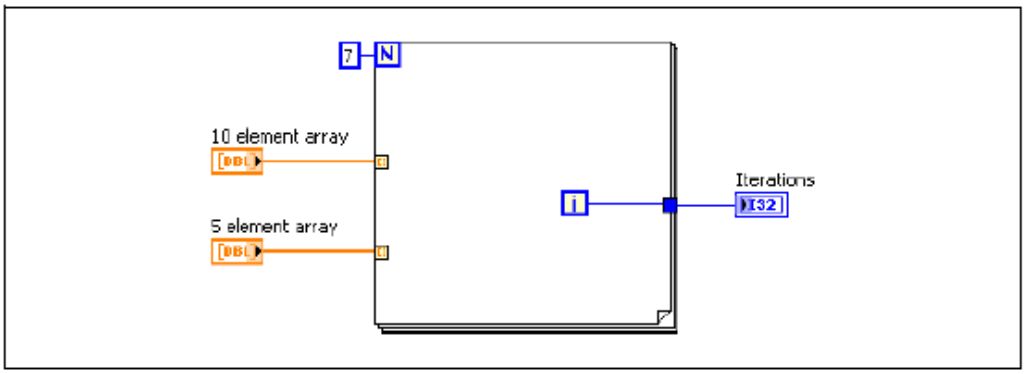


Рисунок 5-16. Что отображается на индикаторе Iteration?

3. Вы создали пользовательский элемент управления, выбрали из выпадающего меню **Control Type** вариант **Control** и сохранили элемент управления в файле с расширением `.ctl`. Далее вы использовали экземпляр пользовательского элемента управления на лицевой панели. Если вы откроете файл `.ctl` и модифицируете элемент управления, отразятся ли изменения на лицевой панели?

   a. Да

   b. Нет

4. Вы вводите данные, представляющие окружность. Данные включают три числа с двойной точностью: координата x, координата y и радиус. В будущем вам может понадобиться расширить все экземпляры данных окружности для включения цвета окружности, задаваемого целым числом. Как вам следует представить окружность на лицевой панели?

   a. Тремя отдельными элементами управления для двух координат и радиуса

   b. Кластером, содержащим все данные

   c. Пользовательским элементом управления, содержащим кластер

   d. Определителем типа, содержащим кластер

   e. Массивом из трех элементов

Самопроверка: ответы

1. Вы можете создать массив массивов

   b. Да

   c. Нет.

   Вы не можете перетащить массив в контейнер массива. Однако вы можете создавать двумерный массив.

2. Два входных массива подключены к циклу For. В обоих туннелях включена автоиндексация. В одном массиве 10 элементов, в другом – 5. К терминалу задания количества итераций подключено значение 7, как показано на рисунке 5-16. Сколько итераций цикла выполнится после запуска VI?

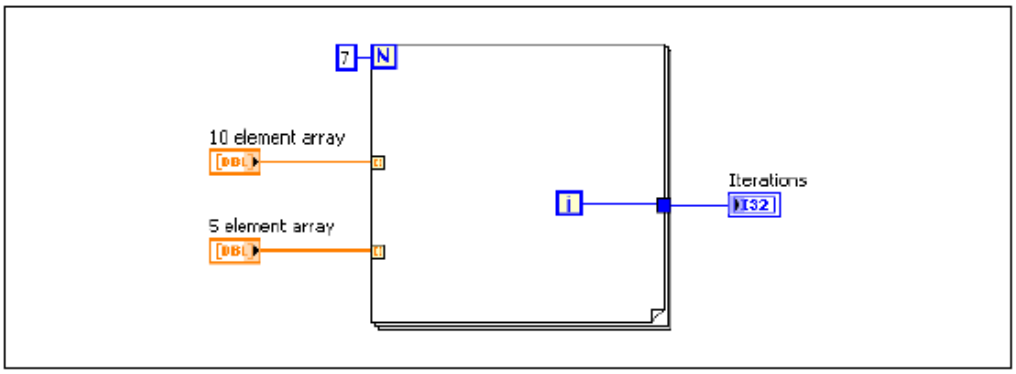


Рисунок 5-17. Что отображается на индикаторе Iteration?

   Значение Iterations = 4

   LabVIEW не допускает выходов за пределы массива. Это помогает предотвратить ошибки программирования. Математические функции LabVIEW работают аналогично – если вы подключите ко входу **x** функции суммирования **Add** массив из 10 элементов, а ко входу **y** этой функции массив из 5 элементов, выходной массив будет содержать 5 элементов.

   Хотя цикл выполнился 5 раз, счет итераций начинается с 0, поэтому на индикаторе Iterations отображается число 4.

3. Вы создали пользовательский элемент управления, выбрали из выпадающего меню **Control Type** вариант **Control** и сохранили элемент управления в файле с расширением `.ctl`. Далее вы использовали экземпляр пользовательского элемента управления на лицевой панели. Если вы откроете файл `.ctl` и модифицируете элемент управления, отразятся ли изменения на лицевой панели?

   a. Да

   b. Нет

4. Вы вводите данные, представляющие окружность. Данные включают три числа с двойной точностью: координата x, координата y и радиус. В будущем вам может понадобиться расширить все экземпляры данных окружности для включения цвета окружности, задаваемого целым числом. Как вам следует представить окружность на лицевой панели?

   a. Тремя отдельными элементами управления для двух координат и радиуса

   b. Кластером, содержащим все данные

   c. Пользовательским элементом управления, содержащим кластер

   d. Определителем типа, содержащим кластер

   e. Массивом из трех элементов

Заметки

6. Управление ресурсами

Вы научились собирать и отображать данные, но важной частью любого проекта является также хранение данных. Кроме того, вы научились устанавливать ваше аппаратное обеспечение и конфигурировать его в Measurement & Automation Explorer. Из этой лекции вы узнаете о хранении данных, программировании базового DAQ-приложения с использованием DAQmx API и об управлении автономными измерительными приборами при помощи VISA API и драйверов измерительных приборов LabVIEW.

План занятия

A. Файловый ввод-вывод

B. Высокоуровневый файловый ввод-вывод

C. Низкоуровневый файловый ввод-вывод

D. Программирование оборудования DAQ

E. Программное управление измерительными приборами

F. Использование драйверов измерительных приборов

A. Файловый ввод-вывод

Функции файлового ввода-вывода выполняют запись данных в файл или чтение их из файла.

Типовая операция файлового ввода-вывода включает следующие процессы:

1. Создание или открытие файла. После открытия файла его представляет уникальный идентификатор, называемый refnum (ссылка)
2. VI или функция файлового ввода-вывода читает из файла или записывает в файл.
3. Закрытие файла.

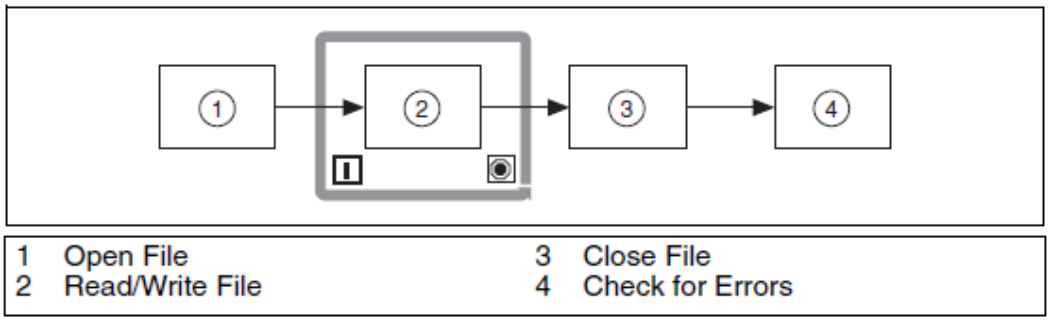


Рисунок 6-1. Этапы типичной операции файлового ввода-вывода

Форматы файлов

В LabVIEW можно использовать или создавать файлы в следующих форматах: двоичные, ASCII, LVM и TDMS.

- **Binary**—двоичные файлы - базовый формат для всех прочих форматов.
- **ASCII**—Файл ASCII представляет собой особый тип двоичного файла, который является стандартом для большинства программ. Он состоит из набора ASCII-кодов. Файлы ASCII также называются текстовыми файлами.
- **LVM**—Файл данных измерений в LabVIEW (.lvm) – текстовый файл с табуляцией в качестве разделителей, который вы можете открыть в табличном или текстовом редакторе. Файл .lvm включает информацию о данных, например, дату и время их генерации. Этот формат – специальный тип ASCII-файла, созданный для LabVIEW.
- **TDMS**—Этот формат - специальный тип двоичного файла, созданный для продуктов National Instruments. Он состоит из двух отдельных файлов – двоичного файла, где хранятся данные и записанные свойства данных, и двоичного файла индексов, который обеспечивает сводную информацию обо всех атрибутах и указателях в двоичном файле.

В этом курсе вы научитесь создавать текстовые (ASCII) файлы. Используйте текстовые файлы, если хотите иметь доступ к файлу из другого приложения, если место на диске и скорость файлового ввода-вывода некритичны, если не нужту произвольный доступ для чтения или записи, и если точность представления чисел не имеет значения.

Вы использовали файл формата LVM в лекции 2, «*Ориентация в LabVIEW*». Для получения дополнительной информации о двоичных файлах и файлах TDMS обратитесь к Справке *LabVIEW Help* или курсу *LabVIE Core 2*.

Папки данных LabVIEW

Вы можете использовать назначаемую по умолчанию папку `LabVIEW Data` для хранения файлов данных LabVIEW, таких, как `.lvm` или `.txt`. LabVIEW инсталлирует папку `LabVIEW Data` в папку данных по умолчанию вашей операционной системы для облегчения организации и нахождения генерируемых LabVIEW данных. По умолчанию Write LabVIEW Measurement File Express VI сохраняет файлы `.lvm`, генерируемые им, в эту папку, а Read LabVIEW Measurement File Express VI читает файлы из этой папки. Константа Default Data Directory и свойство Default Data Directory также по умолчанию возвращают папку `LabVIEW Data`.

Откройте меню **Tools»Options** и выберите **Paths** из списка **Category**, чтобы задать другую папку данных, используемую по умолчанию. Используемая по умолчанию папка данных отличается от папки по умолчанию, которая является папкой, определенной вами для новых VI, пользовательских элементов управления, шаблонов VI и прочих создаваемых документов LabVIEW.

B. Высокоуровневый файловый ввод-вывод

Некоторые VI файлового ввода-вывода выполняют все три этапа процесса файлового ввода-вывода: открытие, чтение/запись и закрытие. Если VI выполняет все три операции, он называется высокоуровневым VI. Однако эти VI могут быть не такими эффективными, как низкоуровневые VI и функции, разработанные для выполнения отдельных частей процесса. Если вы выполняете запись в файл в цикле, используйте VI низкоуровневого файлового ввода-вывода. Если вы записываете данные в файл единой операцией, то можете использовать высокоуровневые VI файлового ввода-вывода.

В LabVIEW есть следующие VI высокоуровневого файлового ввода-вывода

- **Write to Spreadsheet File** (запись в файл электронных таблиц) — преобразует двумерный или одномерный массив чисел двойной точности в текстовую строку и записывает строку в новый ASCII-файл, или добавляет строку в существующий. При этом данные можно транспонировать. VI открывает или создает файл до записи в него, а после записи закрывает. Вы можете использовать этот VI для создания текстового файла, читаемого большинством табличных приложений.

- **Read From Spreadsheet File** (чтение из файла электронных таблиц) — читает заданное количество линий или строк из числового текстового файла, начиная с заданного в символах смещения, и преобразует данные в двумерный массив чисел с двойной точностью. VI открывает файл до чтения из него, а после чтения закрывает. Вы можете использовать этот VI для чтения табличного файла, сохраненного в текстовом формате.
- **Write to Measurement File** (Запись в файл измерений) — Экспресс-VI, который записывает данные в текстовый (`.lvm`) или двоичный (`.tdms`) файл измерений. Вы можете задать метод сохранения, формат файла (`.lvm` или `.tdms`), тип заголовка и разделитель.
- **Read from Measurement File** (чтение из файла измерений) — Экспресс-VI, который читает данные из текстового (`.lvm`) или двоичного (`.tdms`) файла измерений. Вы можете задать имя файла, формат файла и размер сегмента.

Совет: Избегайте установки высокоуровневых VI в цикл, т.к. они выполняют операции открытия и закрытия файлов в каждой итерации цикла.

C. Низкоуровневый файловый ввод-вывод

Низкоуровневые VI файлового ввода-вывода выполняют по одной операции файлового ввода-вывода. Например, существует одна функция для открытия ASCII-файла, одна функция – для его чтения, и одна функция – для закрытия. Используйте низкоуровневые функции, когда файловый ввод-вывод происходит внутри цикла.

Потоковая запись на диск при помощи низкоуровневых функций

Вы можете также использовать функции файлового ввода-вывода для потоковой записи на диск, что сохраняет ресурсы памяти путем уменьшения количества раз, когда функция взаимодействует с операционной системой для открытия и закрытия файла. Потоковая запись на диск - методика сохранения файлов открытыми, пока, например, повторяющиеся операции записи выполняются в цикле.

Подключение константы или элемента управления пути к функциям Write to Text File, Write to Binary File или Write To Spreadsheet File VI добавляет издержки на открытие и закрытие файла каждый раз при выполнении этой функции или VI. VI могут быть более эффективными, если вы будете избегать частого открытия и закрытия одних и тех же файлов.

Во избежание открытия и закрытия одного и того же файла, вы должны передать ссылку (refnum) на файл в цикл. При открытии файла, устройства или сетевого соединения, LabVIEW создает ссылку, связанную с этим файлом, устройством или сетевым соединением. Для всех операций с открытыми файлами, устройствами или сетевыми соединениями используется ссылка для идентификации объекта.

В примерах на рисунках 6-2 и 6-3 показаны преимущества использования потоковой записи на диск. На рисунке 6-2 VI должен открывать и закрывать файл в каждой итерации цикла. На рисунке 6-3 используется потоковая запись на диск для уменьшения количества раз, когда VI должен взаимодействовать с операционной системой для открытия и закрытия файла. Открыв файл один раз до начала цикла и закрыв его после окончания цикла, вы убираете две операции с файлами из каждой итерации цикла.

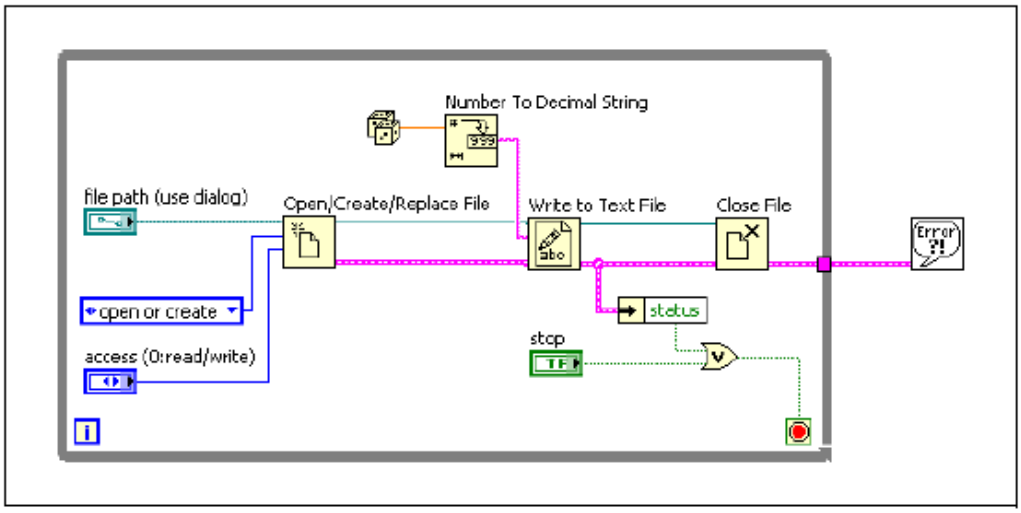


Рисунок 6-2. Пример записи на диск не в режиме потока

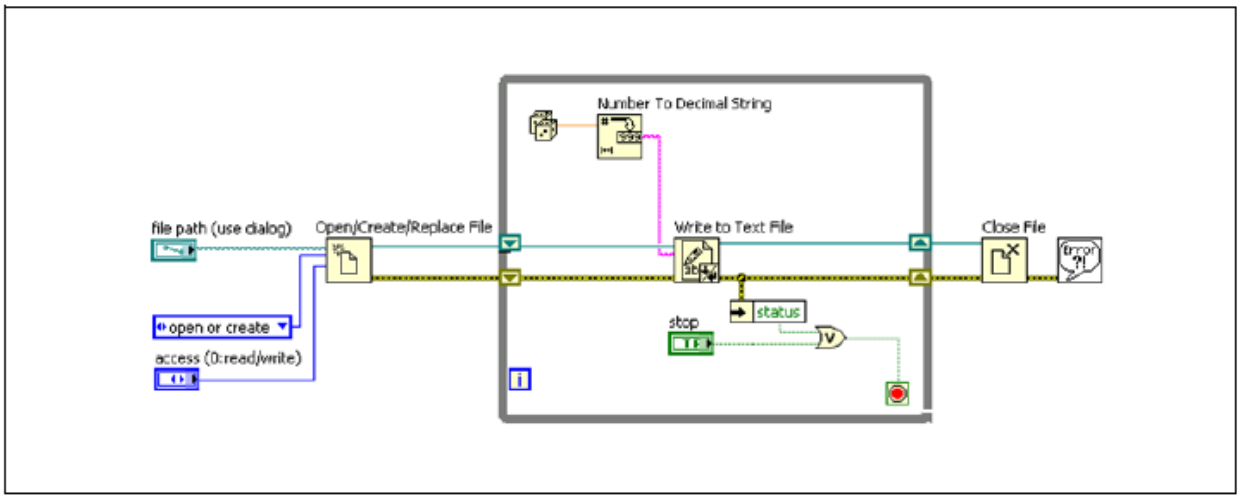


Рисунок 6-3. Пример потоковой записи на диск

D. Программирование оборудования DAQ

Вы уже знакомы с измерительными преобразователями, сигналами, конфигурацией устройств сбора данных и MAX. Теперь вы можете начать изучение применения LabVIEW для разработки приложения сбора данных. NI-DAQmx поддерживают и другие пакеты программного обеспечения, но в данном курсе лекций рассматривается только создание приложений сбора данных в LabVIEW.

DAQmx Name Controls

Палитра DAQmx Name Controls содержит элементы управления для имени задачи, имени канала, физического канала, терминала, имени шкалы, номера устройства и переключателя. Вы можете также создать эти элементы управления, щелкнув правой кнопкой мыши по соответствующему входному терминалу DAQmx VI и выбрав **Create»Control**. Для получения дополнительной информации об этих элементах управления обратитесь к справке *NI-DAQmx Help.*

DAQmx – VI сбора данных

Используйте VI NI-DAQmx с аппаратными средствами NI-DAQ для разработки приложений измерений, сбора данных и управления данными. Обратитесь к документам *DAQ Getting Started Guide* или *NI-DAQ Readme* для получения полного списка устройств, поддерживаемых NI-DAQmx.

Палитра DAQmx - Data Acquisition содержит следующие константы и VI.

Константы

- DAQmx Task Name Constant — создает список всех задач, которые вы создали и сохранили с использованием DAQ Assistant. Щелкните правой кнопкой мыши по константе и выберите I/O Name Filtering из контекстного меню для ограничения задач, отображаемых константой и ограничения того, что можно вводить в константу.
- DAQmx Global Channel Constant — создает список всех глобальных каналов, которые вы создали и сохранили с использованием DAQ Assistant. Щелкните и выберите Browse для выбора нескольких каналов. Щелкните правой кнопкой мыши по константе и выберите I/O Name Filtering из контекстного меню для ограничения каналов, отображаемых константой и ограничения того, что можно вводить в константу.

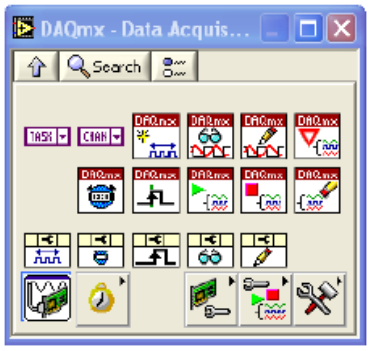
VI

- DAQmx Create Virtual Channel VI — создает виртуальный канал или набор виртуальных каналов и добавляет их в задачу. Копии этого полиморфного VI соответствуют типу ввода-вывода канала (аналоговый ввод, цифровой вывод, выход счетчика); выполняемому измерению или генерации (измерение температуры, генерация напряжения или подсчет

событий); и, в некоторых случаях, используемому датчику (термопары или терморезисторы для измерений температуры).

Эта функция определяет те же настройки, которые вы задаете в MAX, создавая виртуальный канал. Используйте эту функцию, если оператор вашей программы периодически может изменять подключение физического канала, но не другие важные настройки, например, режим конфигурации контакта или используемый масштаб. Используйте выпадающее меню физических каналов для определения номера DAQ-устройства и реального физического канала, к которому подключен ваш сигнал.

- DAQmx Read VI — считывает отсчеты из заданных вами задач или каналов. Экземпляры этого полиморфного VI определяют, в каком формате возвращать отсчеты, один или несколько отсчетов считывать за один раз, и считывать ли данные из одного или нескольких каналов. Вы можете выбрать экземпляр, щелкнув по DAQmx Read VI правой кнопкой и выбрав Select Type для включения дополнительных настроек операции чтения.
- DAQmx Write VI — записывает отсчеты в заданные вами задачи или каналы. Экземпляры этого полиморфного VI определяют, в каком формате записывать отсчеты, записывать один или несколько отсчетов, и записывать ли данные в один или несколько каналов. Вы можете выбрать экземпляр, щелкнув по DAQmx Write VI правой кнопкой и выбрав Select Type для включения дополнительных настроек операции записи.
- DAQmx Wait Until Done VI — ожидает завершения измерения или генерации. Используйте этот VI, чтобы убедиться, что заданная операция завершится, прежде чем вы остановите задачу.
- DAQmx Timing VI — настраивает количество отсчетов для генерации или сбора данных и при необходимости создает буфер. Экземпляры этого полиморфного VI соответствуют типу синхронизации, применяемому в задаче.
- DAQmx Trigger VI — настраивает запуск задачи. Экземпляры этого полиморфного VI соответствуют типу и режимам запуска.
- DAQmx Start Task VI — переводит задачу в состояние выполнения для начала измерения или генерации. Использование этого VI необходимо для некоторых приложений и не обязательно для других.
- DAQmx Stop Task VI — останавливает задачу и возвращает ее в состояние, в каком она была до использования DAQmx Start Task VI или DAQmx Write VI со входом автозапуска, равным TRUE.
- DAQmx Clear Task VI — очищает задачу. До очистки этот VI останавливает задачу, если необходимо, и освобождает занятые задачей ресурсы. Вы не можете использовать задачу после того, как очистили ее, если только не создадите задачу повторно.



E. Программное управление измерительными приборами

VISA – высокоуровневый API, который вызывает низкоуровневые драйвера. VISA может управлять приборами VXI, GPIB, с последовательным портом и осуществлять вызов соответствующих драйверов в зависимости от типа используемого прибора. При отладке коллизий с VISA помните, что наблюдаемая проблема на самом деле может быть проблемой инсталляции одного из вызываемых VISA драйверов.

VISA в LabVIEW – единая библиотека функций для работы с GPIB, с устройствами с последовательным портом, VXI и компьютеризированными измерительными приборами. Вам не нужно использовать отдельные палитры ввода-вывода для программирования прибора. Например, некоторые приборы предоставляют выбор различных интерфейсов. Если драйвер прибора в LabVIEW написан с использованием функций палитры **Instrument I/O»GPIB**, то VI драйвера измерительного прибора не будут работать с прибором с последовательным интерфейсом. VISA решает эту проблему, предоставляя единственный комплект функций для работы с приборами, оснащенными любым типом интерфейса. Поэтому многие драйвера приборов в LabVIEW используют VISA как свой язык ввода-вывода.

Терминология программирования VISA

Приведенная ниже терминология подобна той, что используется в VI драйверов приборов.

- **Ресурс** — любой прибор в системе, в том числе – приборы с последовательным и параллельным портом.
- **Сессия** — вы должны открыть сессию VISA , чтобы ресурс мог с ней связаться (аналогично каналу связи). При открытии сессии работы с прибором LabVIEW возвращает номер сессии VISA, который является уникальной ссылкой на этот прибор. Вы должны использовать номер сессии во всех последующих функциях VISA.
- **Дескриптор прибора** – точное имя ресурса. Дескриптор определяет тип интерфейса (GPIB, VXI, ASRL), адрес устройства (логический или первичный) и тип сессии VISA (INSTR или Event).

Дескриптор прибора аналогичен номеру телефона, ресурс - человеку, с которым вы хотите поговорить, а сессия – телефонной линии. Для каждого

вызова используется собственная линия, а пересечение этих линий приводит к ошибке. В таблице 6-1 показан правильный синтаксис дескриптора прибора.

Таблица 6-1. Синтаксис для различных интерфейсов прибора

Интерфейс	Синтаксис
Асинхронный последовательный	`ASRL[`device`][::INSTR]`
GPIB	`GPIB[`device`]::`primary address`[::`secondary address`][::INSTR]`
Прибор в стандарте VXI со встроенным контроллером или контроллером щины MXI	`VXI[`device`]::`VXI logical address`[::INSTR]`
GPIB-VXI контроллер	`GPIB-VXI[`device`][::`GPIB-VXI primary address`]::`VXI logical address`[::INSTR]`

Вы можете присвоить дескриптору прибора псевдоним VISA в MAX.

Чаще всего из функций VISA используются функции VISA Write и VISA Read. Большинству приборов требуется, чтобы вы послали информацию в виде команды или запроса, прежде чем сможете прочитать информацию из прибора. Таким образом, за функцией VISA Write обычно следует функция VISA Read. Функции VISA Write и VISA Read одинаково работают с любыми типами интерфейса независимо от того, используете ли вы GPIB или последовательную передачу. Однако, поскольку при последовательной передаче от вас требуется настройка дополнительных параметров, вы должны начать передачу по последовательному порту функцией VISA Configure Serial Port VI.

VISA и последовательный порт

VISA Configure Serial Port VI инициализирует порт, идентифицируемый именем ресурса VISA (**VISA resource name**), в соответствии с заданными настройками. **Timeout** устанавливает значение времени ожидания при обмене данными через последовательный порт. **Baud rate** (скорость передачи), **data bits** (биты данных), **parity** (контроль четности) и **flow control** (управление потоком данных) задают соответствующие параметры для последовательного порта. Входной (**error in**) и выходной (**error out**) кластеры ошибок содержат информацию об условиях появления ошибок в этом VI.

На рисунке 6-4 показано, как отправить команду запроса идентификации `*IDN?` прибору, подключенному к последовательному порту COM2. VISA Configure Serial Port VI открывает связь с портом COM2 и устанавливает следующие настройки: 9600 бод, 8 бит данных, контроль нечетности, 1 стоп бит и программное квитирование XON/XOFF. Затем функция VISA Write

отправляет команду. Функция VISA Read считывает до 200 байт в буфер чтения, а Simple Error Handler VI проверяет условия возникновения ошибки.

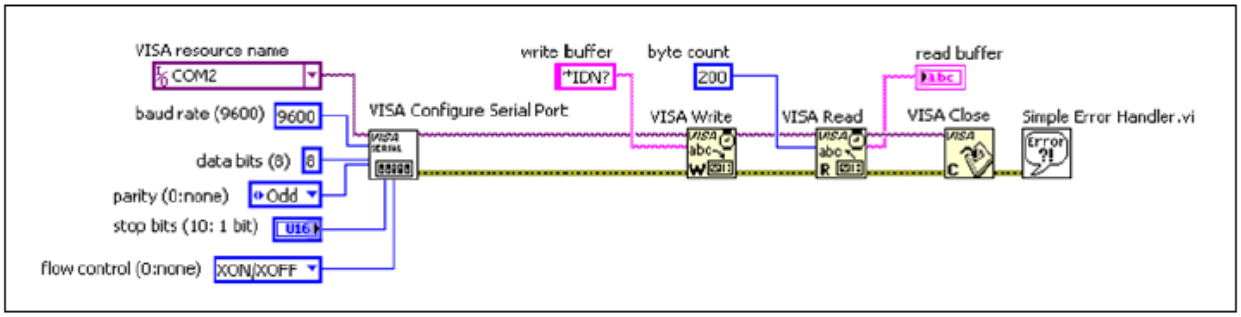


Рисунок 6-4. Пример настройки последовательного порта для VISA

Примечание: Вы можете также использовать VI и функции палитры **Serial** для работы с параллельным портом. Задайте имя ресурса VISA как одного из портов LPT. Например, вы можете использовать MAX для определения того, что LPT1 соответсвует имя ресурса VISA ASRL10::INSTR

F. Использование драйверов измерительных приборов

Представьте следующий сценарий. Вы разработали в LabVIEW VI, который связывается с некоторым осциллографом в вашей лаборатории. К сожалению, осциллограф прекратил работать, и вы должны его заменить. Однако данную модель осциллографов больше не выпускают. Вы хотите приобрести другую модель осциллографа, но ваш VI больше не работает с новым осциллографом. Вам приходится переделывать VI.

Используемый вами драйвер измерительного прибора содержит программный код, специфичный для данного прибора.Поэтому при замене прибора вы должны заменить только драйвера измерительного прибора, что значительно сокращает время переработки программного обеспечения. Драйвер прибора помогает сделать тестовое приложение проще для сопровождения, поскольку драйвер содержит все функции ввода-вывода прибора в одной библиотеке, отдельно от остального кода. При замене оборудования обновление программного приложения упрощается, т.к. драйвер прибора содержит весь специфичный для данного прибора код.

Драйвера измерительных приборов

Драйвер измерительного прибора LabVIEW Plug and Play - это набор VI, управляющих программируемым измерительным прибором. Каждый VI предназначен для программно реализуемой операции, такой, как конфигурирование и запуск прибора, а также чтение из него данных. Драйвера измерительных приборов помогают пользователям начать использовать свои приборы с персональным компьютером, экономят время разработки и средства, поскольку пользователям не приходится учить программный протокол каждого прибора. Драйвера измерительным приборов являются открытыми и хорошо задокументированными

программными средствами, и вы можете настраивать их для улучшения производительности. Модульный дизайн облегчает настройку драйверов.

Местонахождение драйверов измерительных приборов

Вы можете найти большинство драйверов измерительных приборов LabVIEW Plug and Play при помощи поисковика Instrument Driver Finder. Доступ к поисковику Instrument Driver Finder в LabVIEW осуществляется из меню **Tools»Instrumentation»Find Instrument Drivers** или **Help»Find Instrument Drivers**. Instrument Driver Finder подключает вас к сайту `ni.com` для поиска драйверов приборов. При инсталляции драйвера в NI Example Finder добавляется пример программы с использованием драйвера.

Пример использования драйвера измерительного прибора

Блок-диаграмма на рисунке 6-5 инициализирует цифровой мультиметр Agilent 34401 (DMM), используя VI конфигурирования для выбора разрешающей способности и диапазона, выбора функции, разрешения или запрещения автоматического выбора диапазона, использует VI для чтения результата однократного измерения, закрывает прибор и проверяет ошибки.

Каждое приложение, использующее этот драйвер, состоит из одинаковой последовательности операций: инициализация, конфигурирование, чтение данных, закрытие.

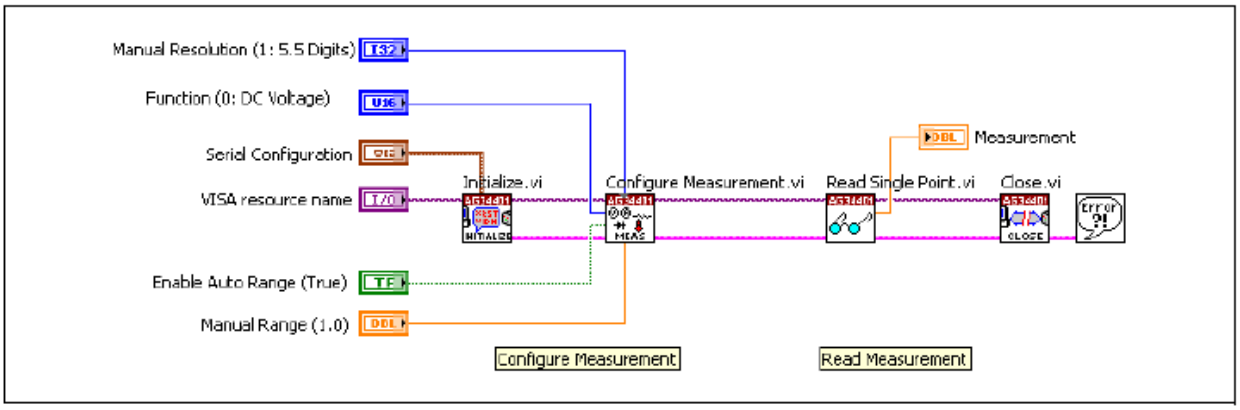


Рисунок 6-5. Пример драйвера измерительного прибора Agilent 34401 DMM

Этот пример доступен в NI Example Finder, после установки драйвера LabVIEW Plug and Play для измерительного прибора Agilent 34401 DMM.

Многие программируемые приборы имеют большое количество функций и режимов. Поэтому необходимо обеспечить единообразную модель разработки, чтобы помочь и разработчикам драйверов измерительных приборов, и конечным пользователям, разрабатывающим приложения с использованием этих приборов. Модель драйверов измерительных приборов LabVIEW Plug and Play содержит рекомендации, как по их внутренней, так и по внешней структуре. Внешняя структура отвечает за взаимодействие драйвера измерительного прибора с пользователем и другими программными компонентами системы. Внутренняя структура отвечает за

внутреннюю организацию программного модуля драйвера измерительного прибора.

С точки зрения внешней структуры драйвера измерительного прибора пользователь взаимодействует с драйвером при помощи API или интерактивного интерфейса. Как правило, интерактивный интерфейс используется для тестирования либо конечными пользователями. Доступ к API осуществляется через LabVIEW. Драйвер измерительного прибора взаимодействует с прибором через VISA.

С точки зрения внутренней структуры VI функции драйвера измерительного прибора организованы в шесть категорий, которые перечислены в приведенной ниже таблице.

Категория	**Описание**
Инициализация	Initialize VI устанавливают связь с прибором и являются первым VI, вызываемым драйвером измерительного прибора.
Конфигурирование	Configure VI – подпрограммы, которые настраивают прибор для выполнения требуемых операций. После вызова этих VI прибор готов ко снятию измерений или стимуляции системы.
Действие/Состояние	Action/Status VI посылают прибору команду выполнить действие (например, осуществить запуск), получают текущее состояние прибора или переводят в состояние ожидания.
Данные	DataVI передают данные в или из прибора.
Утилиты	Utility VI реализуют вспомогательные действия, например, сброс или самопроверку.
Закрытие	Close VI разрывают программную связь с прибором. Это последний VI, вызываемый драйвером измерительного прибора.

Самопроверка: короткий тест

1. Непрерывно работающая программа тестирования сохраняет в одном файле результаты всех тестов, выполняющихся в течение одного часа по мере их получения. Если основной целью является скорость выполнения программы, какие функции файлового ввода-вывода нужно использовать?

   a. VI низкоуровневого ввода-вывода

   b. VI высокоуровневого ввода-вывода

2. Если вы хотите видеть данные в текстовом редакторе, подобном Notepad, какой формат файла нужно использовать при сохранении данных?

   a. ASCII

   b. TDMS

3. Какая из следующих цепочек соответствует основному алгоритму программирования DAQmx?

   a. Create Task»Configure Task»Acquire/Generate Data»Start Task

   b. Acquire/Generate Data»Start Task»Clear Task

   c. Start Task»Create Task»Configure Task»Acquire/Generate Data»Clear Task

   d. Create Task»Configure Task»Start Task»Acquire/Generate Data»Clear Task

4. VISA – это высокоуровневые API, вызывающие драйверы низкого уровня.

   a. Да

   b. Нет

Самопроверка: ответы

1. Непрерывно работающая программа тестирования сохраняет в одном файле результаты всех тестов, выполняющихся в течение одного часа по мере их получения. Если основной целью является скорость выполнения программы, какие функции файлового ввода-вывода нужно использовать?

   a. VI низкоуровневого ввода-вывода

   b. VI высокоуровневого ввода-вывода

2. Если вы хотите видеть данные в текстовом редакторе, подобном Notepad, какой формат файла нужно использовать при сохранении данных?

   a. ASCII

   b. TDMS

3. Какая из следующих цепочек соответствует основному алгоритму программирования DAQmx?

   a. Create Task»Configure Task»Acquire/Generate Data»Start Task

   b. Acquire/Generate Data»Start Task»Clear Task

   c. Start Task»Create Task»Configure Task»Acquire/Generate Data»Clear Task

   d. Create Task»Configure Task»Start Task»Acquire/Generate Data»Clear Task

4. VISA – это высокоуровневые API, вызывающие драйверы низкого уровня.

   a. Да

   b. Нет

Заметки

7. Разработка модульных приложений

Данная лекция посвящена разработке модульных приложений. Преимущество LabVIEW лежит в иерархической природе его VI. Создав VI, вы можете использовать его на блок-диаграмме другого VI, причем количество уровней вложенности неограниченно. Модульное программирование помогает управлять изменениями и быстро отлаживать блок-диаграммы.

План занятия

A. Модульное программирование

B. Создание иконки и панели подключения

C. Использование SubVI

A. Модульное программирование

Модульность – уровень компоновки программы из дискретных модулей, при котором изменения в одном модуле минимально влияют на другие модули. Модули LabVIEW называются subVI.

SubVI – это VI внутри других VI. SubVI соответствует подпрограмме в текстовых языках программирования. При двойном щелчке по subVI вместо диалогового окна, где вы можете выбирать настройки, появляются его лицевая панель и блок-диаграмма. На лицевой панели находятся элементы управления и индикаторы. Блок-диаграмма содержит проводники, иконки объектов лицевой панели, функции, возможно, subVI, и другие объекты LabVIEW, с которыми вы уже знакомы.

В правом верхнем углу лицевой панели и блок-диаграммы находится иконка VI. Эта же иконка появляется при помещении VI на блок-диаграмму.

При создании VI вы можете заметить, что часто выполняете определенную операцию. Рассмотрим использование subVI или циклов для повторного выполнения этой операции. Например, следующая блок-диаграмма содержит две идентичные операции.

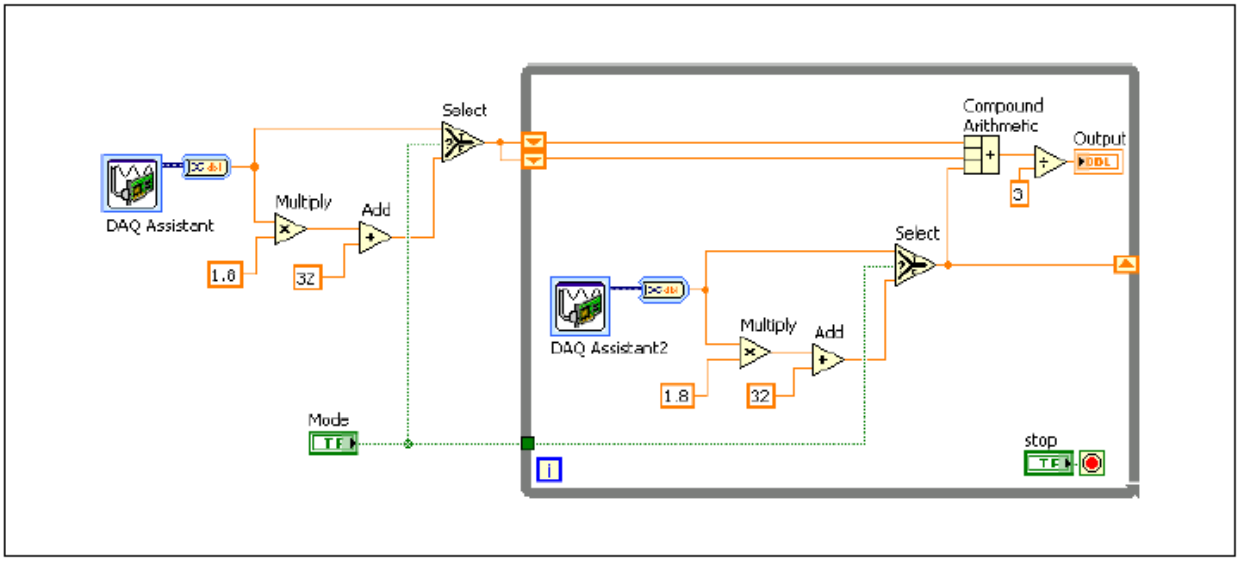


Рисунок 7-1. Блок-диаграмма с двумя одинаковыми операциями

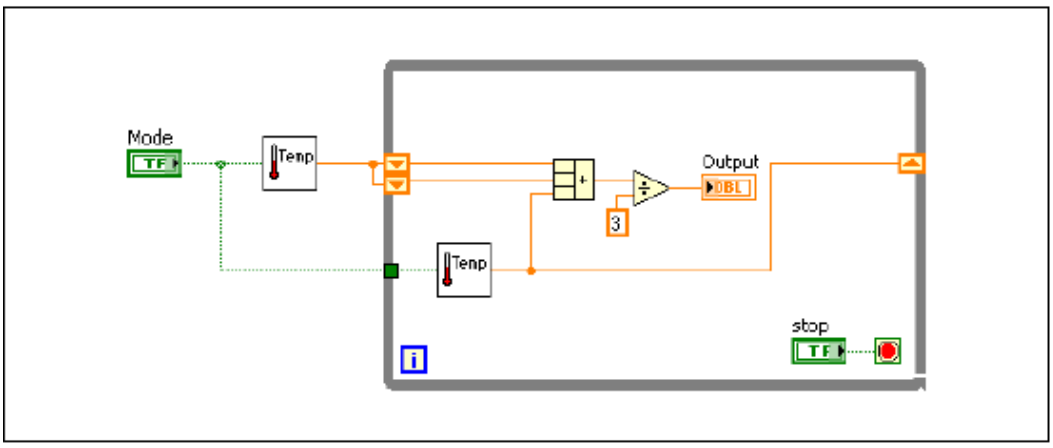


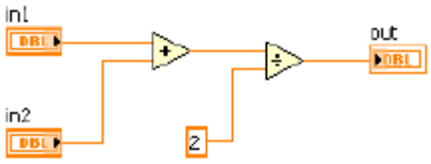
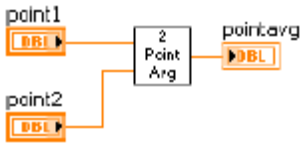
Рисунок 7-2. Блок-диаграмма с SubVI для одинаковых операций

В примере дважды вызывается Temperature VI на блок-диаграмме как subVI, а функциональность примера такая же, как у предыдущего. Вы можете также использовать этот subVI в других VI.

Следующий псевдокод и блок-диаграммы показывают аналогию между subVI и подпрограммами.

Код функции	**Код вызывающей программы**

`function average (in1, in2, out)`
`{`
`out = (in1 + in2)/2.0;`
`}`
	`main`
	`{`
	`average (point1, point2, pointavg)`
	`}`

Блок-диаграмма SubVI	**Блок-диаграмма вызывающего VI**
	

B. Создание иконки и панели подключения

После разработки лицевой панели и блок-диаграммы VI, создайте иконку и панель подключения, чтобы VI можно было использовать в качестве subVI. Иконка и панель подключения соответствуют прототипу функции в текстовых языках программирования. Иконка VI находится в правом верхнем углу окон лицевой панели и блок-диаграммы.

Иконка VI - это графическое представление VI. Она может включать текст, изображения или комбинацию текста и изображений. При использовании VI в качестве subVI, иконка идентифицирует subVI на the блок-диаграмме. При добавлении VI в палитру иконка VI также появляется в палитре **Functions**.

Вы можете дважды щелкнуть по иконке на лицевой панели или блок-диаграмме для ее редактирования.

Примечание: Настройка иконок рекомендуется, но не является обязательной. Использование иконки по умолчанию не повлияет на функциональность

Также вам потребуется создать панель подключения для использования VI в качестве subVI.

Панель подключения – это набор терминалов, соответствующих элементам управления и индикаторам VI, аналогично списку параметров вызова функции в текстовых языках программирования. Панель подключения определяет входы и выходы, которые вы можете подключить к VI для использования в качестве subVI. Панель подключения получает данные из входных терминалов, передает их коду блок-диаграммы через элементы управления лицевой панели и получает результаты через выходные терминалы с индикаторов лицевой панели.

Создание иконки

Иконки – графические представления VIs.

Иконка отображается в правом верхнем углу окна лицевой панели и блок-диаграммы каждого VI.

Иконка VI по умолчанию содержит номер, отображающий, сколько новых VIs, вплоть до 9, вы открыли после запуска LabVIEW. Для отключения нумерации, выберите меню **Tools»Options»Front Panel** и снимите флажок **Use numbers in icons of new VIs (1 through 9)** . Иконка может содержать текст или изображения. При использовании VI в качестве subVI, иконка идентифицирует subVI на блок-диаграммеVI. При добавлении VI в палитру иконка VI также появится в палитре **Functions**.

Используйте диалоговое окно редактора иконок **Icon Editor** для редактирования иконки VI. Для отображения диалогового окна **Icon Editor** дважды щелкните по иконке в верхнем правом углу окон лицевой панели или блок-диаграммы.

Иконки создаются для графического представления VI или пользовательского элемента лицевой панели. Используйте диалоговое окно Icon Editor для создания или редактирования иконок.

Вы можете использовать баннеры для идентификации родственных VIs. National Instruments рекомендует создавать и сохранять баннер как шаблон. Далее вы можете использовать шаблон для иконки родственного VI и модифицировать тело иконки для предоставлении информации о конкретном VI.

Сохранение баннера как шаблона

Выполните следующие шаги для сохранения баннера как шаблона иконки VI.

1. Дважды щелкните по иконке в правом верхнем углу окна лицевой панели или блок-диаграммы или щелкните правой кнопкой мыши по иконке и выберите из контекстного меню **Edit Icon** для отображения диалогового окна **Icon Editor.**
2. Нажмите <Ctrl-A> для выбора всех слоев иконки и нажмите клавишу <Delete>, чтобы удалить выделенное. Иконка по умолчанию – единственный пользовательский слой, называемый **VI Icon.**
3. На странице **Templates** выберите шаблон `_blank.png` из категории **VI»Frameworks.** Вы можете просматривать шаблоны по категории или по ключевым словам.
4. Используйте инструмент Fill на правой стороне диалогового окна Icon **Editor** для заливки баннера иконки цветом.
5. Используйте инструмент Text для ввода текста в баннер иконки. Когда текст активен, вы можете перемещать его нажатием клавиш со стрелками.
6. Выберите **File»Save As»Template** для отображения диалогового окна **Save Icon As** и сохраните иконку как шаблон для дальнейшего использования. LabVIEW сохраняет шаблоны иконок как файлы `.png` с 256 цветами.

Создание иконки из шаблона

Для создания иконки VI на основе шаблона выполните следующие действия.

1. Нажмите клавиши <Ctrl-A> для выбора всех слоев иконки и нажмите <Delete>, чтобы удалить выбранное.
2. На странице **Templates** выберите созданный вами шаблон. Вы можете просматривать шаблоны по категории или по ключевым словам.
3. На странице **Icon Text** введите до четырех строк текста в тело иконки. Вы можете настроить шрифт, выравнивание, размер и цвет текста. Если вы установите галочку **Center text vertically**, диалоговое окно **Icon Editor** отцентрирует текст иконки по вертикали.
4. На странице **Glyphs** перетащите глифы в зону **Preview**. Нажмите клавишу <F> или <R> для разворота глифа по горизонтали или поворота по часовой стрелке соответственно, если нужно – переместите глиф. Вы можете также дважды щелкнуть по глифу, чтобы поместить его в левый верхний угол иконки. Вы можете просматривать глифы по категории или по ключевым словам.
5. Используйте инструмент Move для перемещения глифа. Каждый глиф находится в отдельном слое и потому перемещается отдельно. Обратите внимание, что при выборе символа остальная часть иконки окрашивается серым, чтобы было понятно, что именно вы перемещаете.

6. Используйте инструменты редактирования, расположенные с правой стороны диалогового окна **Icon Editor**, при необходимости дальнейшего редактирования иконки.

7. В диалоговом окне Icon Editor создается новый слой для каждого непоследовательного использования инструментов редактирования. Выберите **Layers»Create New Layer** для создания нового слоя при последовательном использовании инструментов редактирования.

Примечание: Вы не можете изменять шаблон иконки или текст иконки инструментами редактирования, расположенными с правой стороны диалогового окна **Icon Editor.** Используйте страницы **Templates** и **Icon Text** для изменения шаблона и текста иконки соответственно.

7. (Не обязательно) Выберите **Layers»Show Layers Page**, чтобы отобразить страницу **Layers.** Используйте эту страницу для настройки наименования, прозрачности, видимости и порядка следования слоев иконки.
8. Нажмите кнопку **OK** для сохранения информации об иконке в VI и закрытия диалогового окна **Icon Editor.**

Вы можете также перетащить рисунок откуда-нибудь из файловой системы и поместить в верхний правый угол лицевой панели для использования рисунка в качестве иконки. Вы можете перетаскивать файлы с расширениями `.png`, `.bmp`, или `.jpg`.

Примечание: Если вы измените иконку путем перетаскивания рисунка из файловой системы, LabVIEW создаст новый слой под названием VI Icon для рисунка и удалит всю прочую информацию об иконке из диалогового окна Icon Editor.

Настройка панели подключения

Задайте подключения, назначив каждому терминал панели подключения элемент управления или индикатор лицевой панели. Щелкните правой кнопкой мыши в верхнем правом углу лицевой панели и выберите из контекстного меню Show Connector для отображения панели подключения. Панель подключения отобразится вместо иконки. При просмотре панели подключения в первый раз вы увидите шаблон подключения. Вы можете выбрать другой шаблон, щелкнув по панели подключения правой кнопкой мыши и выбрав Patterns из контекстного меню.

Каждый прямоугольник панели подключения представляет терминал. Используйте прямоугольники для назначения входов и выходов. Шаблон панели подключения по умолчанию: 4 × 2 × 2 × 4. Если вы полагаете, что при дальнейшей модификации VI вам потребуются новые входы или выходы, оставьте дополнительные терминалы шаблона панели подключения по умолчанию неподключенными.

На лицевой панели на рисунке 7-3 имеются четыре элемента управления и один индикатор, поэтому LabVIEW отображает четыре входных и один выходной терминал на панели подключения.

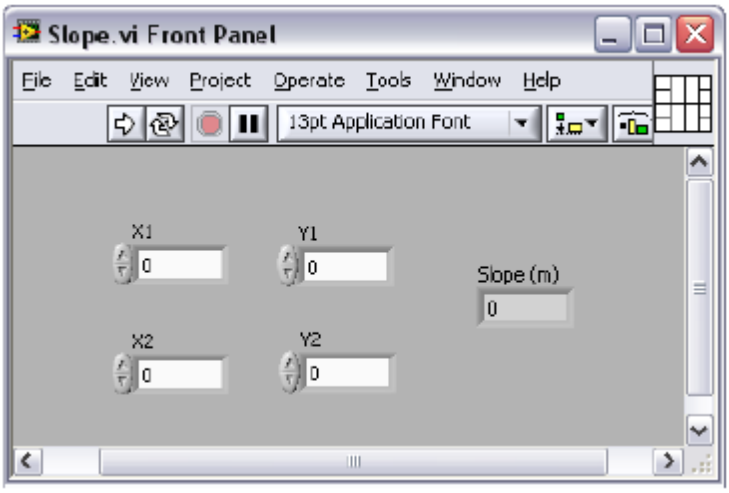


Рисунок 7-3. Лицевая панель Slope VI

Выбор и модификация шаблонов терминалов

Выберайте различные шаблоны терминалов VI, щелкнув правой кнопкой мыши по панели подключения и выбрав **Patterns** из контекстного меню. Например, вы можете выбрать шаблон панели подключения с дополнительными терминалами, не подключая их, пока они вам не понадобятся. Подобная гибкость позволяет вам вносить изменения, минимально воздействуя на иерархию VI.

На лицевой панели также быть больше элементов управления и индикаторов, чем терминалов.

Шаблон, использующийся в данный момент, выделен жирной границей. Вы можете назначить панели подключения до 28 терминалов.

Слева показан наиболее часто используемый шаблон, принятый в качестве стандарта и упрощающий подключения.

На рисунке 7-4 показан пример стандартной схемы размещения терминалов шаблона. Верхние входы и выходы, как правило, используются для передачи ссылок, а нижние входы и выходы – для обработки ошибок.

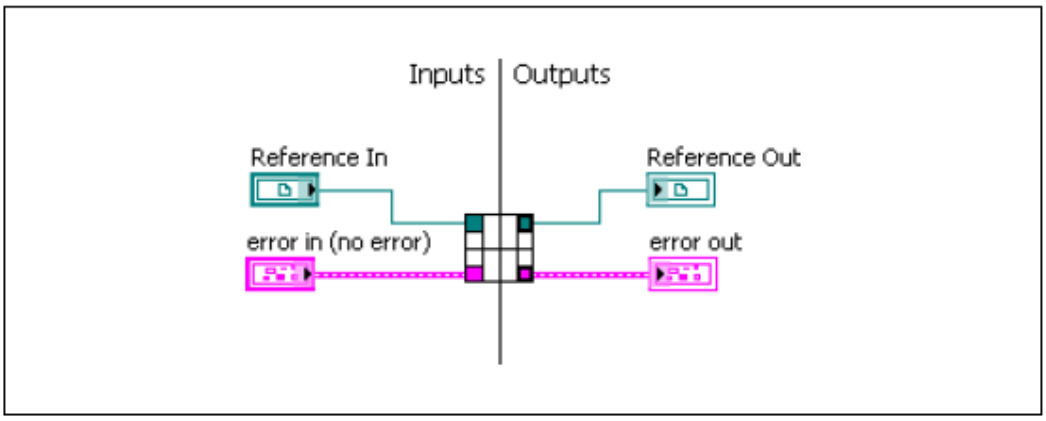


Рисунок 7-4. Пример схемы размещения терминалов шаблона

Примечание: Избегайте использования панелей подключения с количеством терминалов больше 16. Хотя шаблоны с большим числом терминалов могут показаться полезными, их очень трудно подключать. Если вам требуется передать больше данных, используйте кластеры.

Назначение терминалов элементам управления и индикаторам

После выбора шаблона панели подключения, вы можете назначить каждому из терминалов панели подключения элемент управления или индикатор лицевой панели. Помещайте входы слева, а выходы – справа, чтобы избежать путаницы при подключении.

Для назначения терминала элементу управления или индикатору лицевой панели щелкните сначала по терминалу на панели подключения, а потом по элементу управления или индикатору лицевой панели, который хотите назначить этому терминалу. Щелкните по пустому месту на лицевой панели. Терминал окрасится цветом, соответствующим типу данных выбранного элемента, что показывает, что вы подключили терминал.

Вы можете также выбрать сначала элемент управления или индикатор, а потом терминал.

Примечание: Хотя вы используете инструмент Wiring для назначения терминалов панели подключения элементам управления и индикаторам лицевой панели, между панелью подключения и этими элементами управления и индикаторами не протягиваются проводники.

C. Использование SubVIs

Для помещения subVI на блок-диаграмму щелкните кнопку **Select a VI** на палитре **Functions**. Выберите VI, который хотите использовать в качестве subVI, и дважды щелкните мышкой для помещения его на блок-диаграмму.

Вы можете также поместить открытый VI на блок-диаграмму другого открытого VI. Щелкните инструментом Positioning по иконке в верхнем правом углу лицевой панели или блок-диаграммы VI, который хотите использовать в качестве subVI , и перетащите его иконку на блок-диаграмму другого VI.

Открытие и редактирование SubVIs

Чтобы отобразить лицевую панель subVI из VI верхнего уровня, дважды щелкните по subVI на блок-диаграмме инструментом Operating или Positioning. Для отображения блок-диаграммы subVI удерживайте при этом клавишу <Ctrl>.

Вы можете редактировать subVI, щелкнув дважды по subVI на блок-диаграмме инструментом Operating или Positioning. При сохранении subVI изменения применяются ко всем вызовам этого subVI, а не только к текущему экземпляру.

Задание обязательных, рекомендованных и необязательных входов и выходов

В окне контекстной справки **Context Help** наименования обязательных терминалов выделены жирным шрифтом, рекомендованных - обычным, необязательных – серым. Названия необязательных терминалов не будут отображаться, если вы нажмете кнопку **Hide Optional Terminals and Full Path** в окне **Context Help.**

Вы можете назначать каждый вход обязательным, рекомендованным и необязательным в вашем subVI, чтобы пользователь не забыл подключить терминалы subVI.

Щелкните правой кнопкой по терминалу на панели подключения и выберите из контекстного меню **This Connection Is**. Флажок означает настройку терминала. Выберите **Required, Recommended** или **Optional**. Можете также вызвать из меню **Tools»Options»Front Panel** и поставить флажок **Connector pane terminals default to required**. Эта настройка устанавливает все терминалы панели подключения обязательными, а не рекомендованными. Это относится к подключениям, созданным инструментом wiring, и к subVI, созданным с помощью **Create SubVI**.

Примечание: Вы можете выбрать **Dynamic Dispatch Input (Required)** или **Dynamic Dispatch Output (Recommended)** для динамически назначаемого члена класса VI.

Для входных терминалов «обязательный» означает, что если вы не подключите этот вход subVI, стрелка запуска блок-диаграммы, на которые вы его поместили, будет сломана. Для выходных терминалов опция «обязательный» недоступна. Для входов и выходов «рекомендованный» или «необязательный» означает, что блок-диаграмма, на которой расположен

subVI может выполняться, даже если вы не подключите эти терминалы. Если вы не подключите эти терминалы, VI не выдаст предупреждений.

Входы и выходы VI в библиотеке `vi.lib` уже помечены как **Required**, **Recommended** или **Optional.** По умолчанию LabVIEW назначает входам и выходам созданных вами VI статус **Recommended**. Назначайте терминал обязательным, только если такое свойство этого входа необходимо, чтобы VI работал правильно.

Обработка ошибок в SubVIs

Передача ошибок в и из subVI происходит через кластеры ошибок. При помощи структуры Case ошибки, передаваемые в subVI, обрабатываются во фреймах No Error и Error.

Кадр No Error содержит код для нормальной работы subVI, как показано на рисунке 7-5.

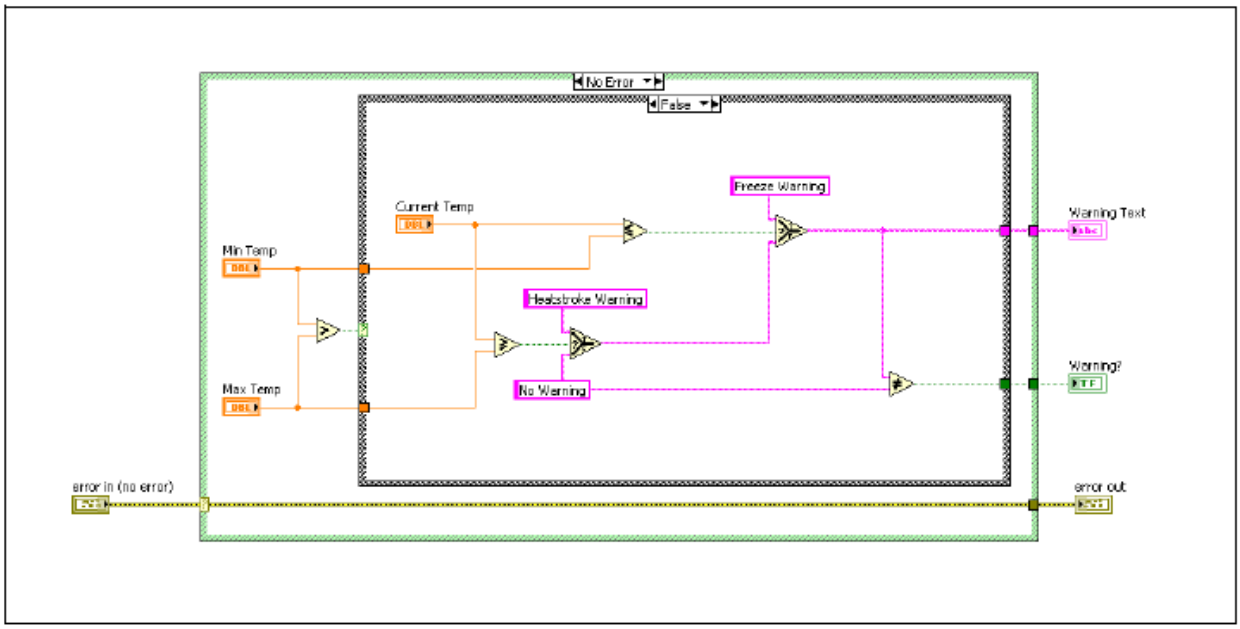


Рисунок 7-5. Фрейм No Error SubVI

Кадр Error, как правило, передает ошибку из входного кластера ошибок **error in** в выходной **error out** и содержит код обработки ошибки, как показано на рисунке 7-6.

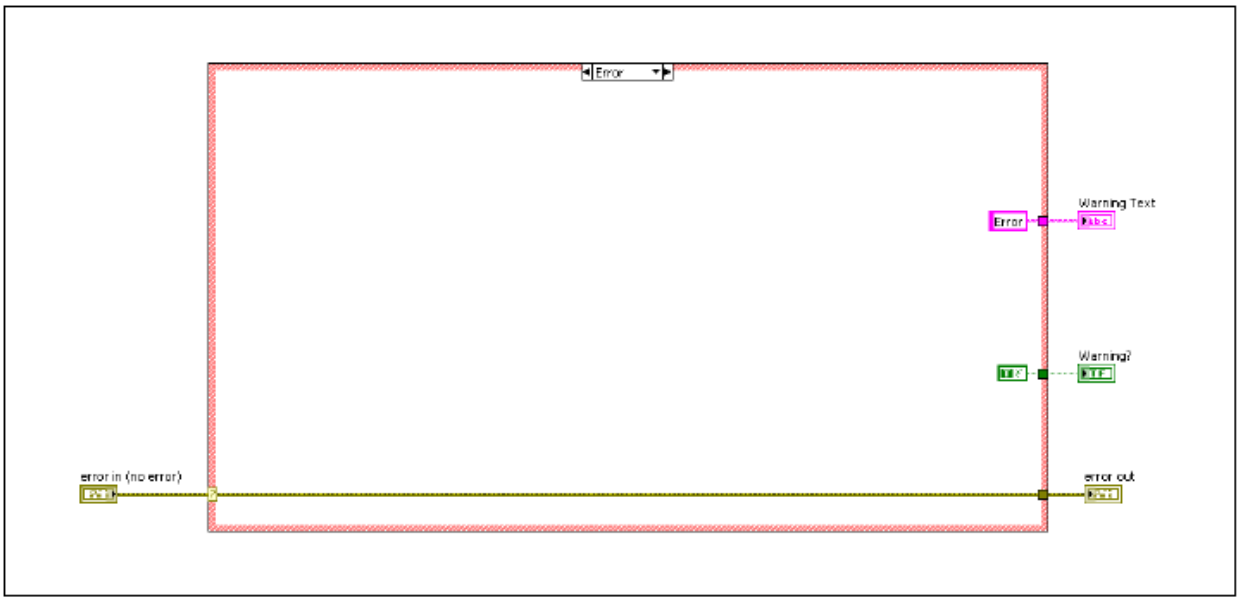


Рисунок 7-6. Кадр Error SubVI

Избегайте использования Simple Error Handler VI и General Error Handler VI внутри subVIs. При необходимости используйте их в главном VI, как показано на рисунке 7-7.

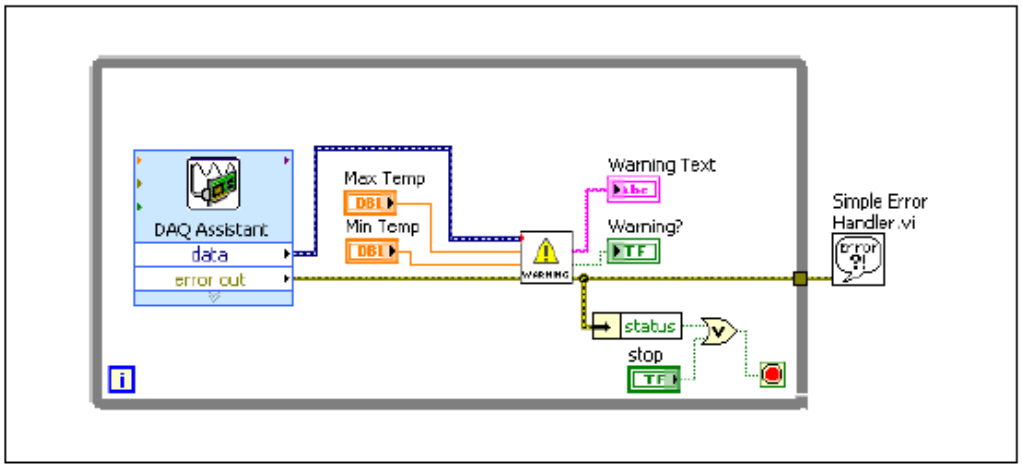


Рисунок 7-7. Блок-диаграмма вызывающего VI

Создание SubVI из существующего VI

Вы можете упростить блок-диаграмму VI, преобразовав ее фрагменты в subVI. Используйте инструмент Positioning для выбора фрагмента блок-диаграммы, которую хотите использовать повторно, и вызовите из меню **Edit»Create SubVI**. На месте выбранного фрагмента блок-диаграммы появится иконка нового subVI. LabVIEW создает элементы управления и индикаторы для нового subVI, автоматически настраивает панель подключения в соответствии с количеством выбранных вами терминалов элементов управления и индикаторов и подключает subVI к существующим проводникам.

На рисунке 7-8 показано преобразование выбранного фрагмента в subVI.

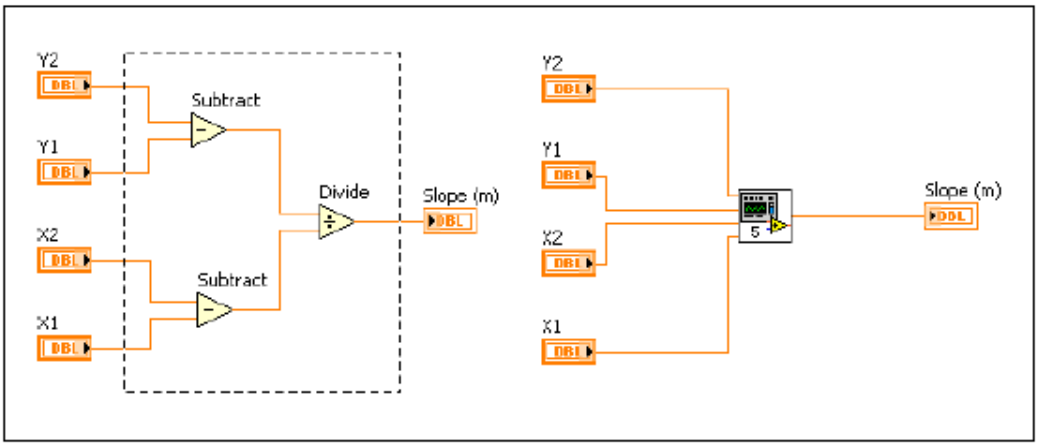


Рисунок 7-8. Создание нового SubVI

В новом subVI применяется шаблон панели подключения и иконка, используемые по умолчанию. Дважды щелкните по subVI для редактирования панели подключения и иконки и сохранения subVI.

Примечание: Не выделяйте более 28 объектов при создании subVI, поскольку 28 – максимальное число терминалов на панели подключения. Если вам нужно использовать более 28 элементов управления и индикаторов лицевой панели, сгруппируйте некоторые из них в кластер и назначьте кластеру терминал на панели подключения.

Самопроверка: короткий тест

1. Какая настройка терминала subVI может послужить причиной ошибки, если этот терминал не подключен?

   a. Required (Обязательный)

   b. Recommended (Рекомендованный)

   c. Optional (Необязательный)

2. Вы должны создать специальную иконку для использования VI в качестве subVI.

   a. Да

   b. Нет

Самопроверка: ответы

1. Какая настройка терминала subVI может послужить причиной ошибки, если этот терминал не подключен?

   a. Required (Обязательный)

   b. Recommended (Рекомендованный)

   c. Optional (Необязательный)

2. Вы должны создать специальную иконку для использования VI в качестве subVI.

   a. Да

   b. Нет

   Нет необходимости создавать специальную иконку для использования VI в качестве subVI, но это настоятельно рекомендуется для того, чтобы ваш код был более читабельным.

Заметки

8. Общепринятая методика проектирования и шаблоны

Первым шагом создания проекта является изучение существующих в LabVIEW архитектур. Архитектуры жизненно необходимо для разработки успешного программного дизайна. Наиболее распространенные архитектуры обычно группируются в шаблоны проектирования.

По мере распространения шаблон проектирования, становится легче распознавать, где он был использован. Это поможет вам и другим разработчика читать и модифицировать VI, основанных на определенных шаблонах проектирования.

Для VI LabVIEW существует много шаблонов проектирования. В большинстве приложений используется, по крайней мере, один шаблон. В данном курсе вы изучите шаблон проектирования под названием конечный автомат. Вы можете узнать больше о шаблонах проектирования из курса *LabVIEW Core 2*.

План занятия

A. Программирование последовательностей
B. Программирование состояний
C. Конечный автомат
D. Параллелизм

A. Программирование последовательностей

Многие VI, созданные вами в LabVIEW, выполняют задачи последовательно. Способы программирования этих задач могут значительно различаться. Рассмотрим блок-диаграмму на рисунке 8-1. На ней измеряется напряжение, диалоговое окно предлагает пользователю включить питание, снова измеряется напряжение, пользователю предлагается выключить питание. Однако в этом примере ничто не задает порядок выполнения этих операций, и любое из них может выполниться первым.

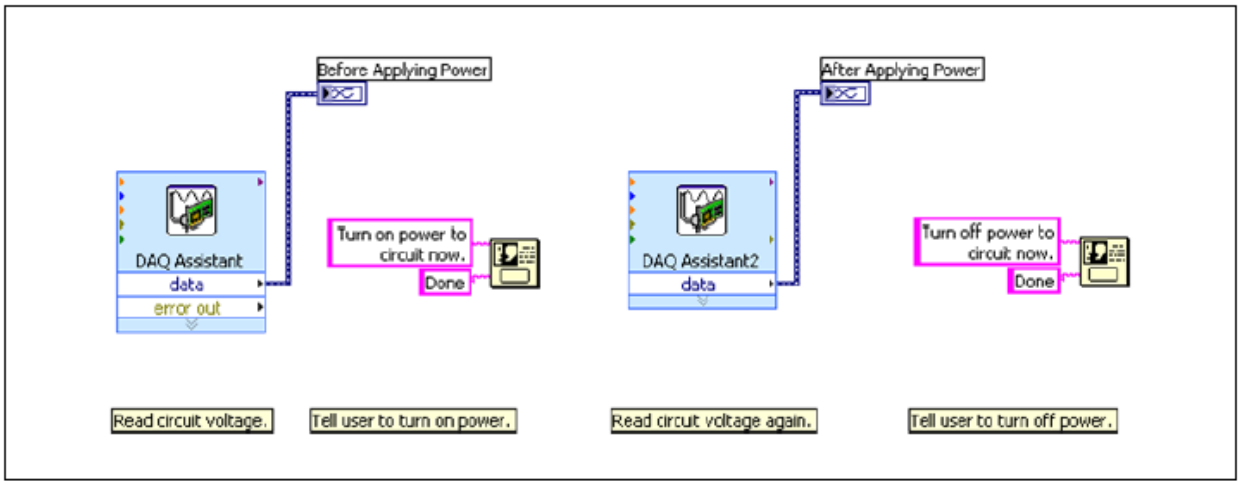


Рисунок 8-1. Задачи, выполняемые не последовательно

В LabVIEW вы можете задать последовательное выполнение задач, поместив каждую задачу в отдельный subVI и соединив кластеры ошибок subVI в соответствии с желаемым порядком выполнения. Однако в этом примере только у двух задач есть кластеры ошибок. Вы сможете заставить выполняться в нужном порядке два DAQ Assistant, но не функции One Button Dialog, как показано на рисунке 8-2.

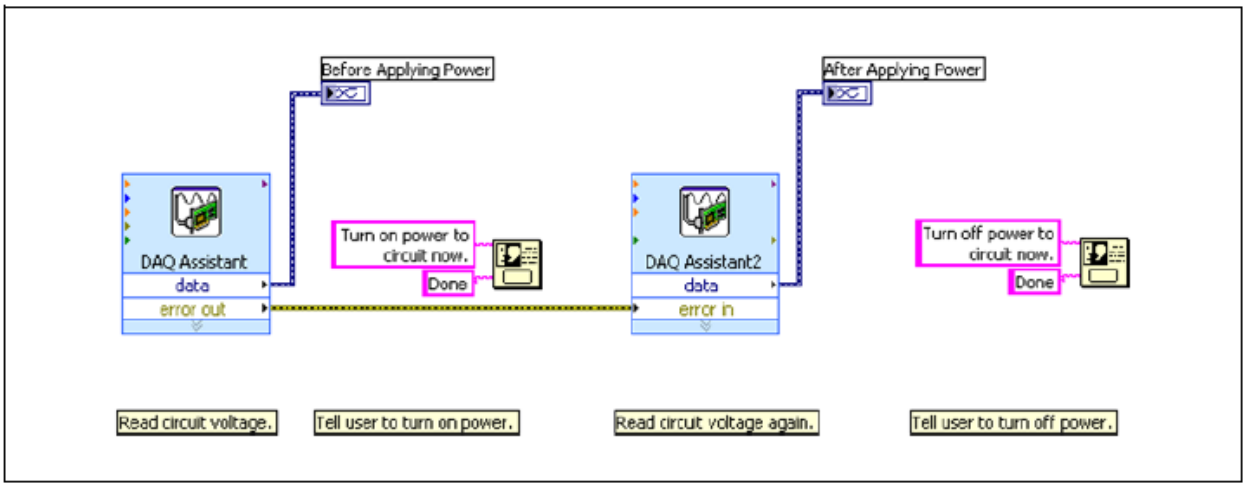


Рисунок 8-2. Задачи, частично выполняемые последовательно

Вы можете использовать структуру Sequence, чтобы задать порядок выполнения объектов на блок-диаграмме. Структура Sequence состоит из

одной или несколько субдиаграмм, или фреймов, которые выполняются последовательно; второй фрейм не начнет выполняться, пока не будет выполнено все, что требуется в первом фрейме. На рисунке 8-3 показан пример VI с использованием структуры Sequence для определения порядка выполнения.

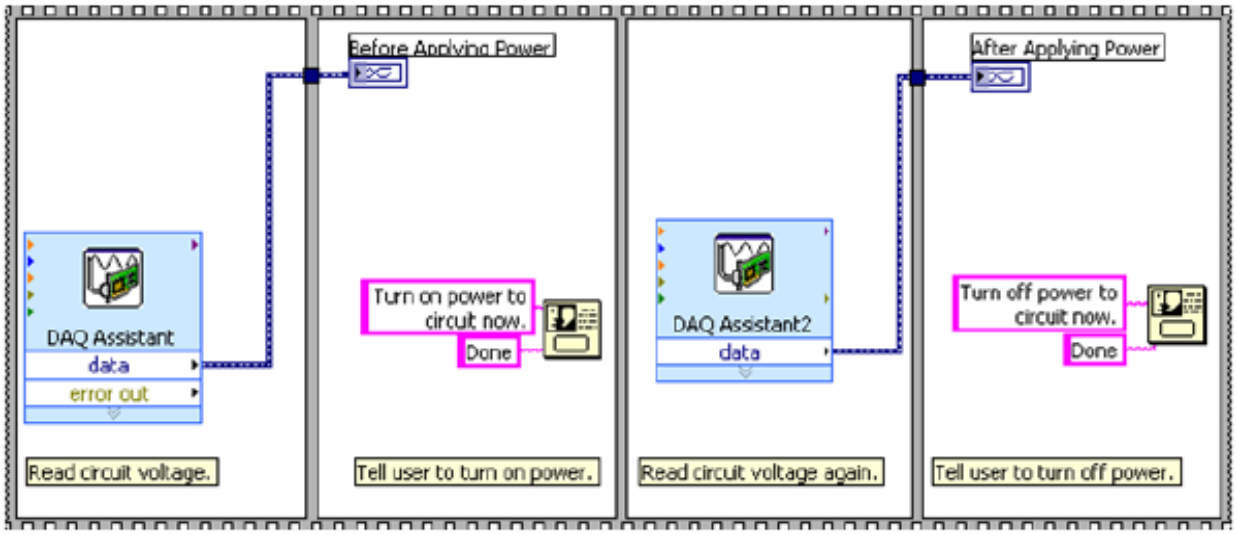


Рисунок 8-3. Задачи, упорядоченные с помощью структуры Sequence

Чтобы воспользоваться преимуществами свойственного LabVIEW параллелизма, не злоупотребляйте использованием структур Sequence. Структуры Sequence гарантируют порядок выполнения, но запрещают параллельное выполнение операций. Еще один минус использования структур Sequence в том, что остановить выполнение последовательности на каком-либо фрейме нельзя. Хороший пример использования структур Sequence показан на рисунке 8-4.

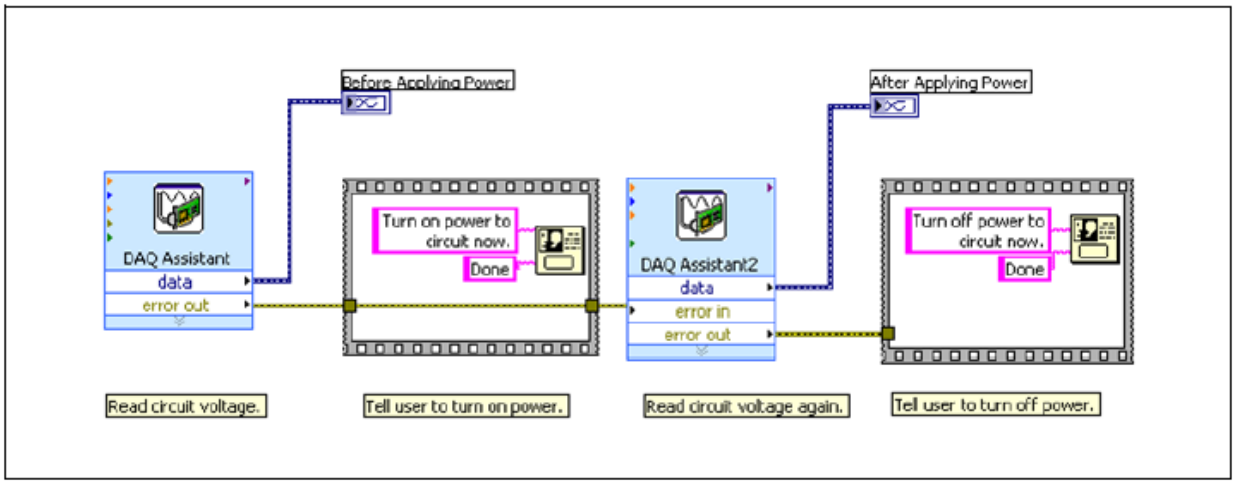


Рисунок 8-4. Задачи, упорядоченные с помощью структур Sequence и кластеров ошибок

Однако лучший способ разработки такого VI – заключить функции One Button Dialog 2 в структуру Case, соединяя кластеры ошибок с селектором выбора структуры Case.

Используйте структуры Sequence расчетливо, потому что они не выполняют принудительной проверки ошибок и продолжают выполняться даже при обнаружении ошибок. Для управления порядком выполнения полагайтесь больше на поток данных, а не на структуры sequence.

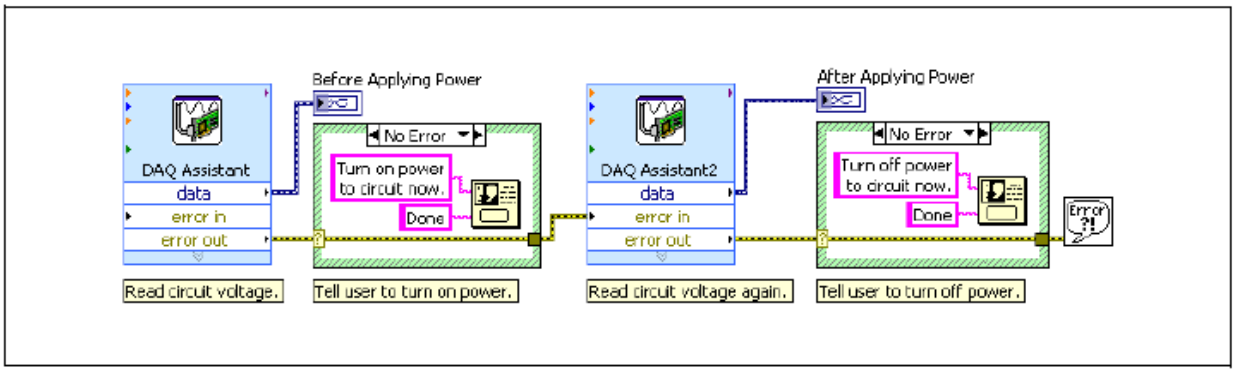


Рисунок 8-5. Задачи, упорядоченные с помощью кластера ошибок и структур Case

B. Программирование состояний

Хотя структуры Sequence или последовательно соединенные subVI решают задачу программирования последовательности, однако часто при реализации VI требуются более сложные способы программирования:

- Что делать, если вам необходимо изменять порядок выполнения последовательности?
- Что делать, если вам необходимо повторять один из элементов последовательности более часто, чем другие?
- Что делать, если некоторые элементы последовательности нужно выполнять только при определенных условиях?
- Что делать, если вам необходимо остановить программу немедленно, до завершения выполнения всей последовательности?

Хотя в вашей программе может не требоваться ничего из вышеперечисленного, всегда существует возможность модификации программы в будущем. Поэтому программирование на основе архитектуры конечных автоматов является хорошим выбором, даже если пока достаточно и программирования с использованием последовательной структуры.

C. Конечные автоматы

Конечный автомат – широко распространенный и очень полезный шаблон разработки в LabVIEW. Вы можете использовать конечный автомат для реализации любого алгоритма, который можно описать диаграммой состояний или блок-схемой. Конечный автомат обычно реализует умеренно сложный алгоритм принятия решений, например, программу диагностирования или наблюдения за процессом.

Конечный автомат состоит из набора состояний и функций перехода из текущего состояния в следующее. Существует много разновидностей

конечных автоматов. Наиболее распространенные из них – автомат Мили и автомат Мура. Автомат Мили выполняет действие при каждом переходе. Автомат Мура выполняет определенное действие в каждом состоянии в диаграмме состояний-переходов Шаблон проектирования конечных автоматов в LabVIEW реализует любой алгоритм, описываемый машиной Мура.

Применение конечных автоматов.

Используйте конечные автоматы в приложениях, где существуют различимые состояния. Из каждого состояния можно пеерйти в одно или несколько состояний, либо завершить поток процесса. В конечном автомате используются данные, введенные пользователем, или вычисленные для определения следующего состояния. Во многих приложениях требуется начальное состояние, за которым следует состояние по умолчанию, в котором могут производиться различные действия. Выполняемые действия могут зависеть как от предыдущих, так и от текущих значений входов и состояний. Состояние выключения часто производит также процедуру очистки.

Конечные автоматы часто используются для создания интерфейса пользователя. Различные действия оператора переводят пользовательский интерфейс в различные фрагменты обработки, выполняющие роль состояний конечного автомата. Каждый фрагмент может привести к другому фрагменту для продолжения обработки или подождать другого действия пользователя. В приложениях с пользовательским интерфейсом конечный автомат непрерывно наблюдает за действиями, которые выполняет пользователь.

Конечный автомат часто используют также в процедурах тестирования, в которых каждый этап процесса представляет собой состояние. В зависимости от результатов тестирования каждого состояния, может быть вызвано другое состояние. Это может происходить непрерывно, что позволяет провести углубленный анализ тестируемого процесса.

Преимущество использование конечных автоматов в том, что вы с легкостью создадите VI в LabVIEW после разработки диаграммы переходов.

Инфраструктура конечного автомата

Для перевода диаграммы переходов в блок-диаграмму LabVIEW требуются следующие инфраструктурные составляющие:

- **While Loop** — непрерывное выполнение различных состояний
- **Case Structure** — содержит фрейм с исполняемым кодом для каждого состояния
- **Shift Register** — содержит информацию о переходах
- **State Functionality Code** — реализует функциональность состояния
- **Transition Code** — определяет следующее состояние в последовательности

На рисунке 8-6 показана базовая структура реализованного в LabVIEW конечного автомата системы измерения температуры.

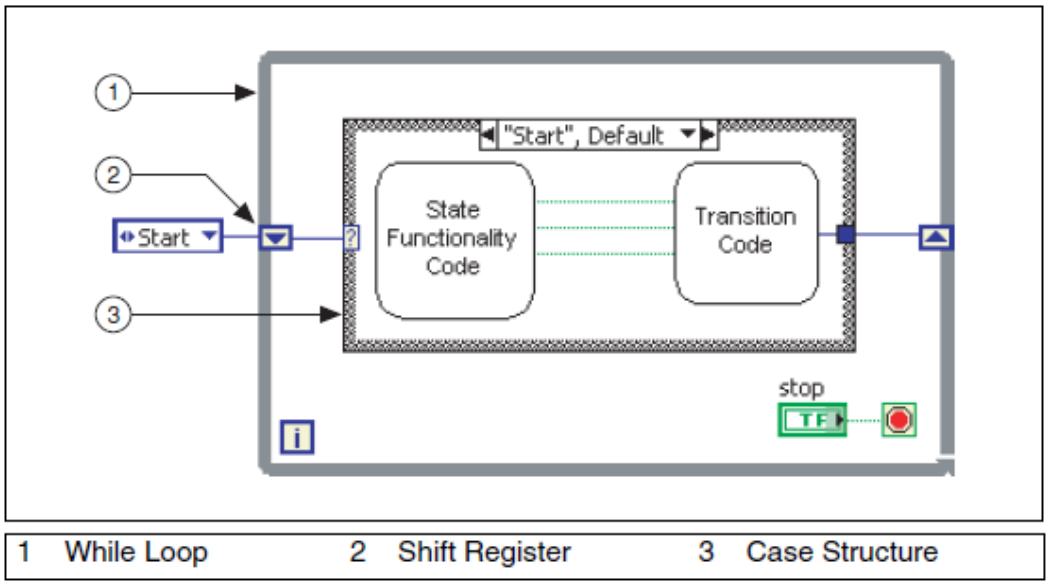


Рисунок 8-6. Базовая инфраструктура конечного автомата LabVIEW

Поток диаграммы перехода реализован циклом While. Отдельные состояния представлены фреймами структуры Case. Сдвиговый регистр в цикле While хранит текущее состояния и передает его на вход структуры Case.

Управление конечными автоматами

Лучший способ управления инициализацией и переходами конечного автомата – использование элемента управления типа enumerated. Enums широко используются как селекторы выбора в конечном автомате. Однако, если пользователь предпринимает попытку добавить или удалить состояние из элемента управления типа enumerated, оставшиеся проводники, подключенные к копиями этого элемента управления, рвутся. Это одно из наиболее распространенных затруднений при реализации конечных автоматов на основе элементов управления типа enumerated. Одно из решений этой проблемы – создать определитель типа enumerated. Это заставляет все экземпляры элемента управления типа enumerated автоматически обновляться при добавлении или удалении состояния.

Переходы в конечных автоматах

Существует много способов управления тем, какой фрейм структуры Case будет выполняться в конечном автомате. Выбирайте метод, лучше всего подходящий для функциональности и сложности вашего конечного автомата. Из всех способов реализации переходов в конечном автомате наиболее простой и распространенный – единственная структура Case, которая может быть использована для переходов между любым количеством состояний. Этот способ обеспечивает масштабируемую, легкочитаемую и развиваемую архитектуру конечного автомата. Другие методы могут оказаться полезными в определенных ситуациях, поэтому вам имеет смысл ознакомиться и с ними.

Переход по умолчанию

Для перехода по умолчанию не требуется кода для определения следующего состояния, поскольку далее возможно только одно состояние. На рисунке 8-7 показан шаблон разработки, в котором реализован переход по умолчанию для системы измерения температуры.

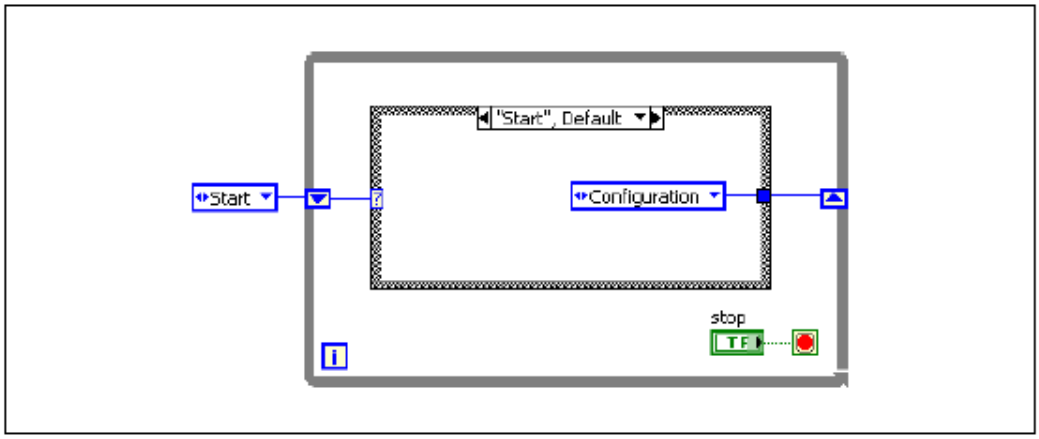


Рисунок 8-7. Единственный переход по умолчанию

Переход между двумя состояниями

Данный метод включает принятие решения о переходе между двумя состояниями. Существуют несколько шаблонов для реализации этого метода. На рисунке 8-8 показана функция Select, используемая для перехода между двумя состояниями.

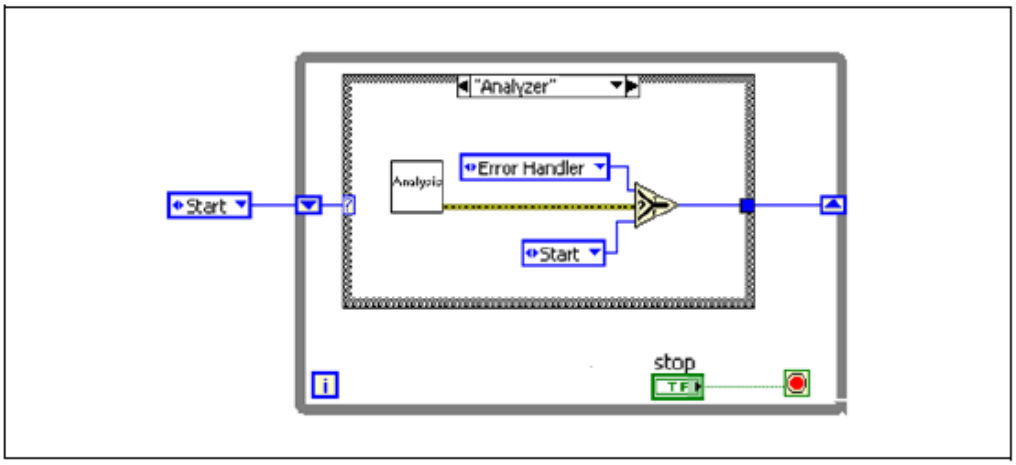


Рисунок 8-8. Код перехода с функцией Select

Этот метод хорошо работает, если вы точно знаете, что в определенном состоянии всегда выполняется переход в два других состояния. Однако этот метод ограничивает масштабируемость приложения. Если вам придется модифицировать приложение для перехода между более чем двумя состояниями, оно перестанет работать и потребует глобальной модификации кода перехода.

Переход между двумя и более состояниями.

Для создания более масштабируемой архитектуры используйте один из следующих методов перехода между состояниями.

- **Case Structure** — используйте структуру Case вместо функции Select для реализации кода перехода. На рисунке 8-9 показан переход с использованием структуры Case в системе измерения температуры.

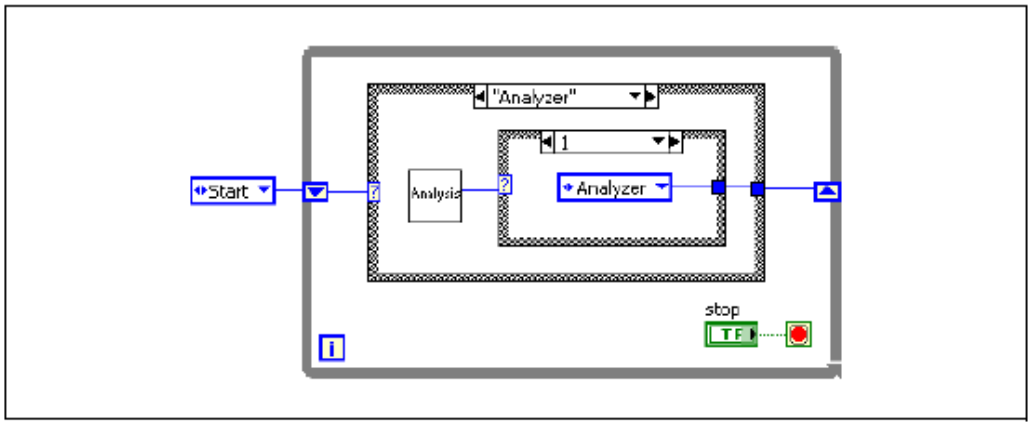


Рисунок 8-9. Код перехода со структурой Case

Преимуществом использования структуры Case является самодокументированный код. Поскольку каждый фрейм структуры Case соответствует элементу в enum, код легок для чтения и понимания. Структура Case также масштабируется, поэтому по мере расширения приложения вы можете добавлять больше переходов в определенное состояние, добавляя для этого состояния фреймы в структуру Case. Недостатком использования структуры Case является невозможность увидеть весь код сразу. Из-за природы структуры Case невозможно сразу увидеть всю функциональность кода перехода.

- **Transition Array** (Массив переходов) – если вам необходимо увидеть больше кода, чем позволяет структура Case, вы можете создать массив переходов для всех переходов. На рисунке 8-10 показан массив переходов для системы измерения температуры.

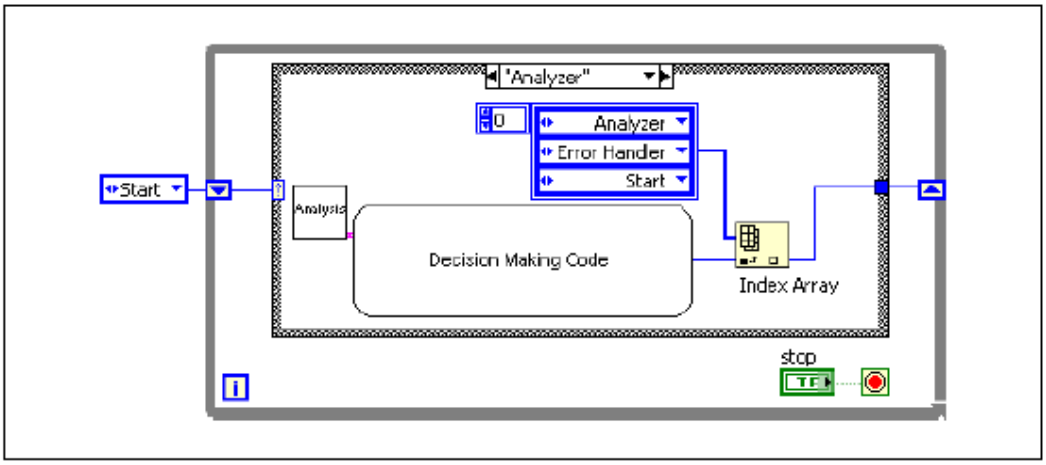


Рисунок 8-10. Код перехода с массивом переходов

В этом примере код принятия решений предоставляет индекс, который описывает следующее состояние. Например, если далее код должен перейти в состояние Error Handler, код принятия решений выдает 1 на индексный вход функции Index Array. Этот шаблон разработки делает код перехода масштабируемым и более простым для чтения.

Недостаток данного подхода в том, что вы должны разрабатывать код перехода с осторожностью, поскольку индексация в массиве начинается с нуля.

Изучение структуры Case: курсовой проект

В данном курсовом проекте каждые полсекунды выполняется измерение температуры, анализируется, не является ли температура слишком высокой или слишком низкой и выдается предупреждение пользователю, если существует опасность теплового удара или обморожения. Программа записывает данные при возникновении предупреждения. Если пользователь не нажал кнопку «Стоп», весь процесс повторяется. На рисунке 8-11 представлена диаграмма переходов для реализации курсового проекта

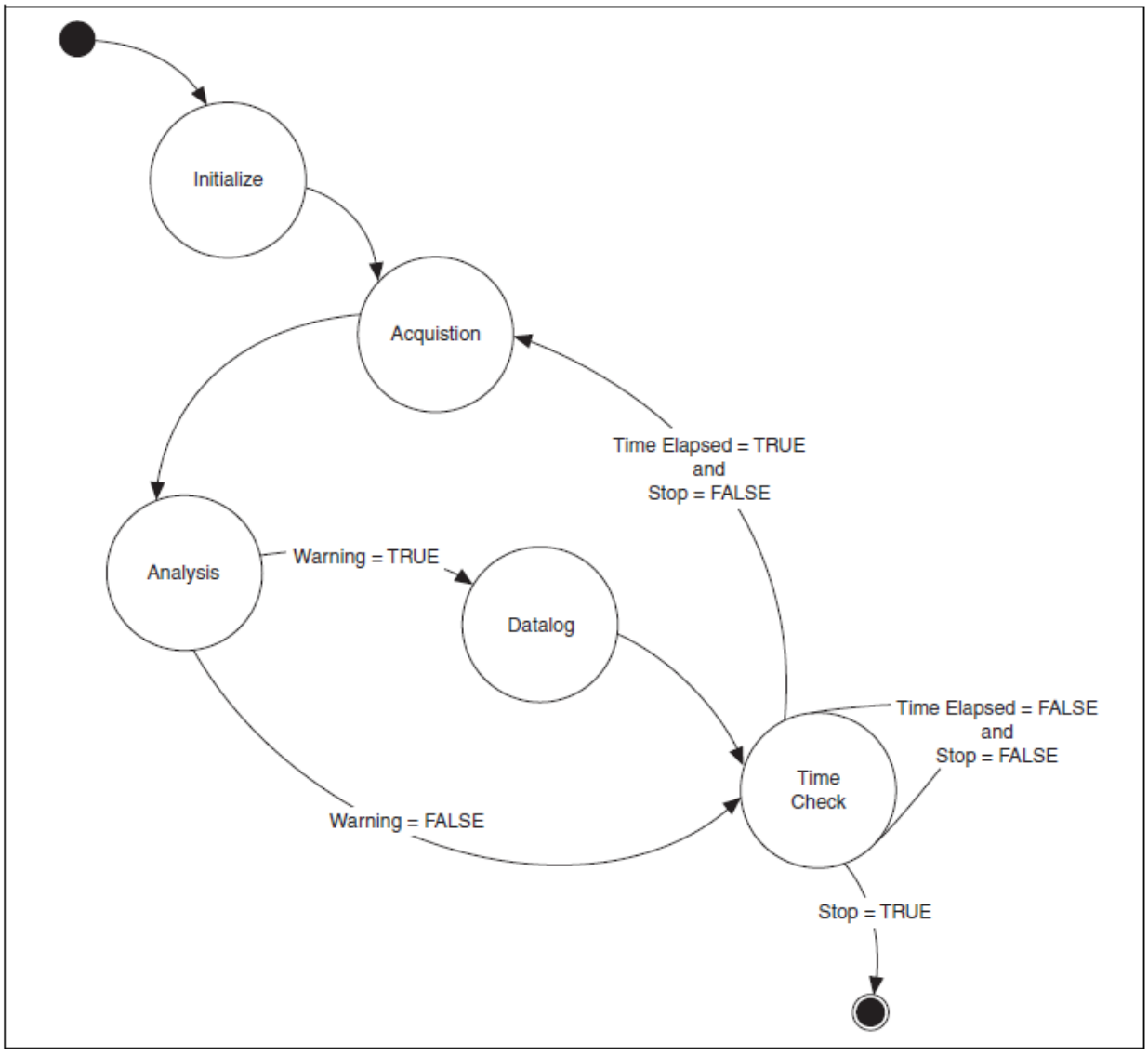


Рисунок 8-11. Диаграмма переходов для реализации курсового проекта

На рисунках с 8-12 по 8-15 представлены состояния конечного автомата, реализующего показанную на рисунке 8-11 диаграмму переходов. Если вы инсталлировали упражнения и решения, проект находится в папке `<Exercises>\LabVIEW Core 1\Course Project`, и вы можете подробно изучить этот конечный автомат.

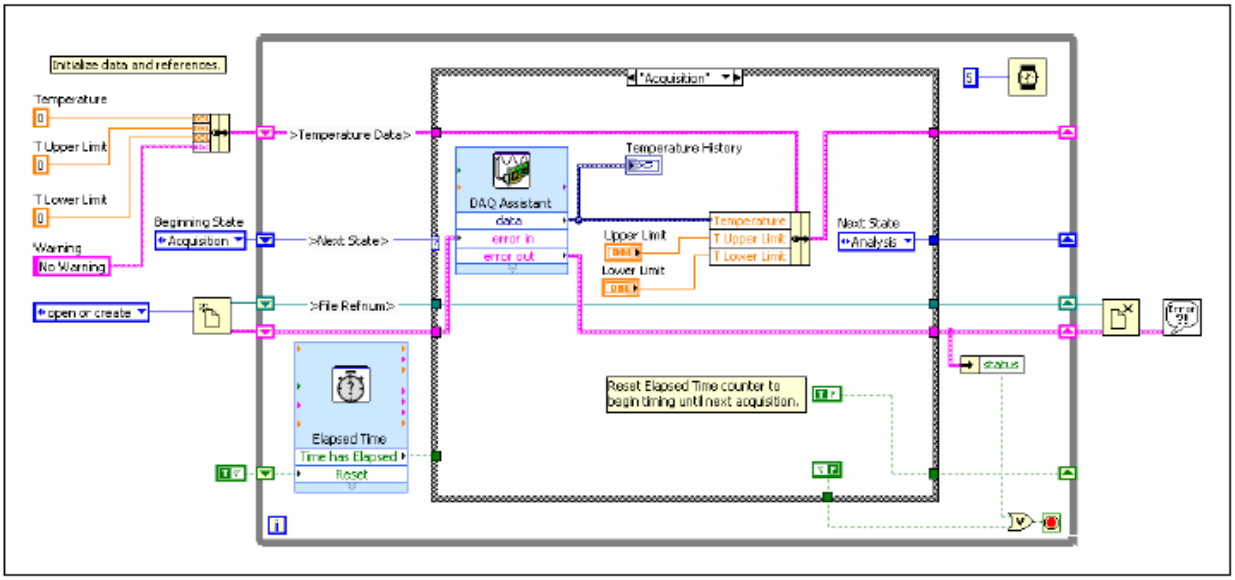


Рисунок 8-12. Состояние измерения

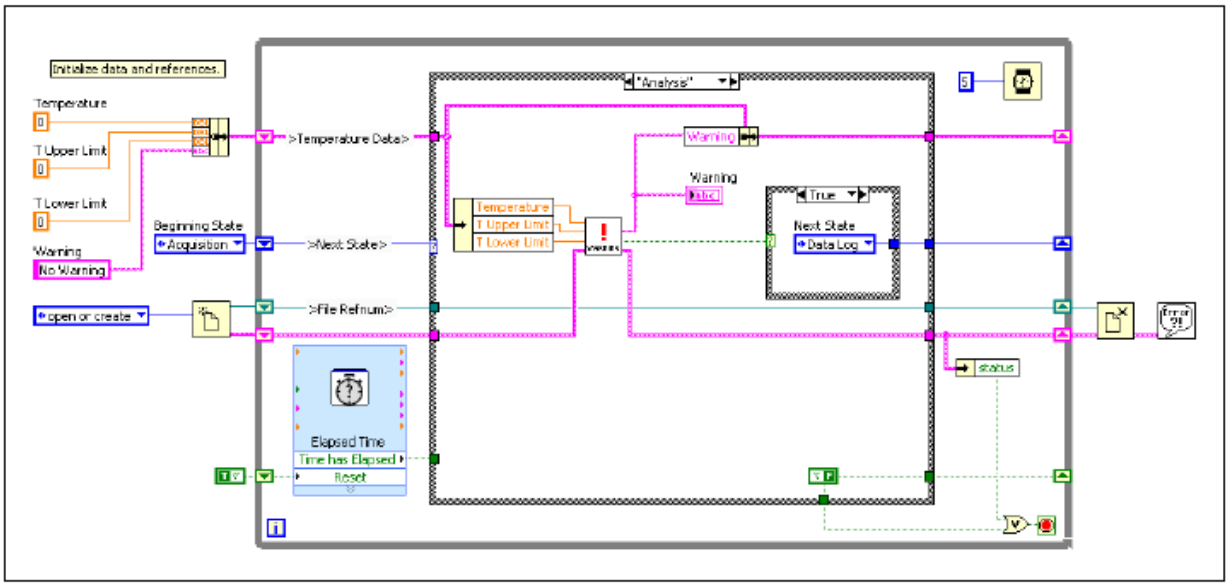


Рисунок 8-13. Состояние анализа

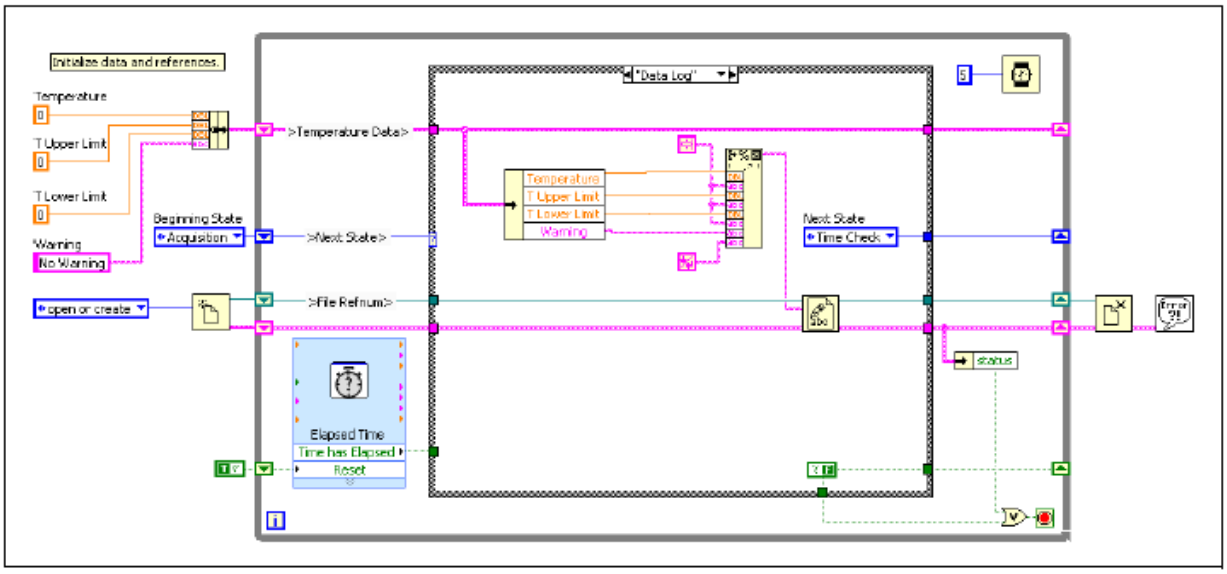


Рисунок 8-14. Состояние записи данных

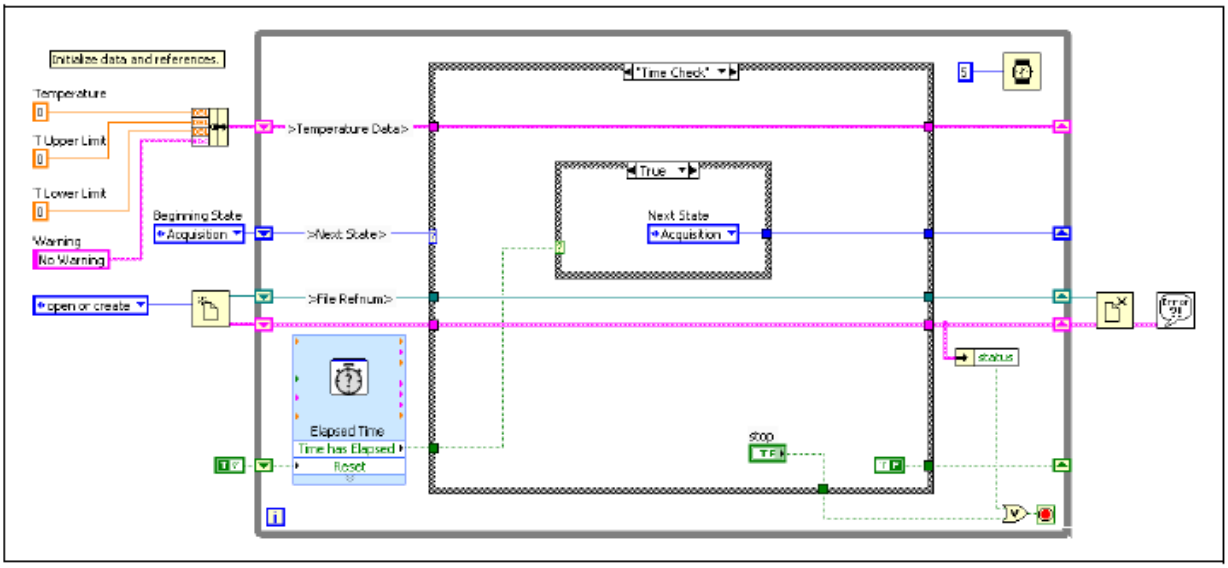


Рисунок 8-15. Состояние проверки времени

D. Параллелизм

Часто вам необходимо запрограммировать несколько задач, которые должны выполняться одновременно. В LabVIEW задачи могут выполняться параллельно, если между ними нет зависимости по данным, и если они не используют один и тот же общий ресурс. Пример общего ресурса – файл или измерительный прибор.

Вы узнаете о шаблонах проектирования LabVIEW для выполнения нескольких задач параллельно в курсе LabVIEW Core 2. Этот шаблон проектирования включает параллельные циклы, архитектуру ведущий/ведомый и изготовитель/потребитель.

Самопроверка: короткий тест

1. При использовании структуры Sequence вы можете остановить выполнение на любой стадии последовательности.

   a. Да

   b. Нет

2. Какие из следующих преимуществ дает применение конечного автомата взамен последовательной структуры?

   a. Вы можете изменять порядок выполнения в последовательности

   b. Вы можете повторять отдельные элементы последовательности

   c. Вы можете задать условия, которые определяют, когда должен выполняться элемент последовательности

   d. Вы можете остановить выполнение программы в любом месте последовательности.

Самопроверка: ответы

1. При использовании структуры Sequence вы можете остановить выполнение на любой стадии последовательности.

   a. Да

   b. Нет

2. Какие из следующих преимуществ дает применение конечного автомата взамен последовательной структуры?

   a. Вы можете изменять порядок выполнения в последовательности

   b. Вы можете повторять отдельные элементы последовательности

   c. Вы можете задать условия, которые определяют, когда должен выполняться элемент последовательности

   d. Вы можете остановить выполнение программы в любом месте последовательности

Заметки

9. Использование переменных

В этой лекции вы научитесь использовать переменные для передачи данных между циклами и VIs. Также вы узнаете о проблемах и способах их преодоления при программировании с использованием переменных.

План занятия

A. Параллелизм

B. Переменные

C. Функциональные глобальные переменные

D. Состязания

A. Параллелизм

В рамках данного курса параллелизм относится к выполнению нескольких задач в одно и то же время. В качестве примера можно привести формирование и отображение двух синусоид с различными частотами. При Используя параллелизм вы помещаете одну синусоиду в один цикл, а вторую – в другой.

Проблемы при программировании параллельных задач – передача данных между циклами без создания зависимости между этими данными. Например, если вы передадите данные по проводнику, циклы больше не будут параллельными. В примере с несколькими синусоидами вам может понадобиться единый механизм останова, как показано на рисунке 9-1.

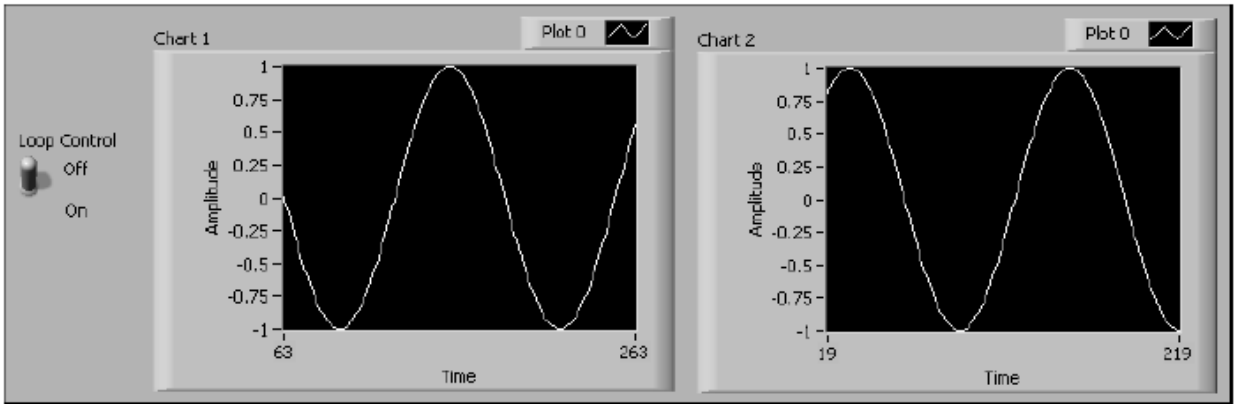


Рисунок 9-1. Лицевая панель прибора с параллельными циклами

Рассмотрим, что произойдет, если вы попытаетесь передать данные между параллельными циклами, пользуясь различными методами.

Метод 1 (неверный)

Поместите терминал **Loop Control** снаружи обоих циклов и подключите его к обоим терминалам условия, как показано на рисунке 9-2. Loop control – входной сигнал для обоих циклов, поэтому значение терминала **Loop Control** считывается только один раз, до начала выполнения обоих циклов While. Если в цикл передается False, циклы While будут выполняться бесконечно. Выключение тумблера не останавливает VI, поскольку во время итераций циклов значение тумблера не считывается.

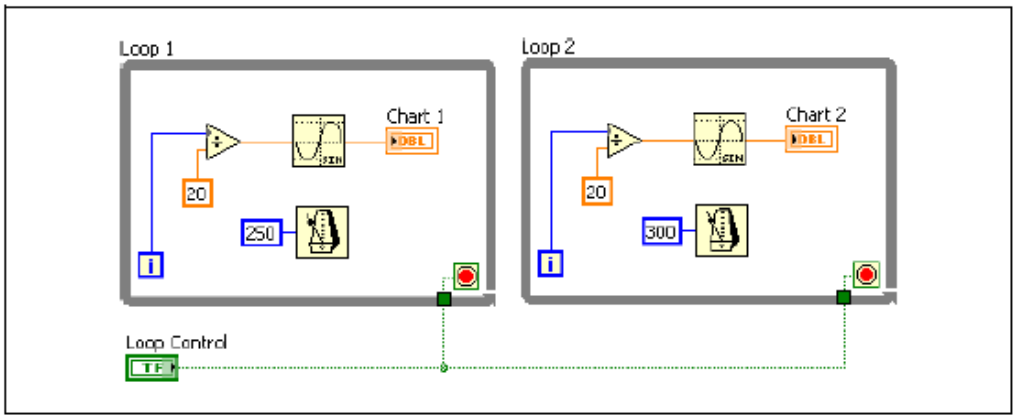


Рисунок 9-2. Параллельные циклы, метод 1

Метод 2 (неверный)

Поместите терминал **Loop Control** в цикл 1, чтобы он считывался в каждой итерации цикла 1, как показано на приведенной блок-диаграмме. Хотя цикл 1 завершается как надо, цикл 2 не начинает выполняться, пока не получит все входные данные. Цикл 1 не передает данные из цикла, пока цикл не остановится, поэтому цикл 2 должен ждать последнее значение **Loop Control**, доступное только по завершении цикла 1. Таким образом, циклы не выполняются параллельно. Цикл 2 выполняет только одну итерацию, потому что его терминал условия получает значение True с переключателя **Loop Control** из цикла 1.

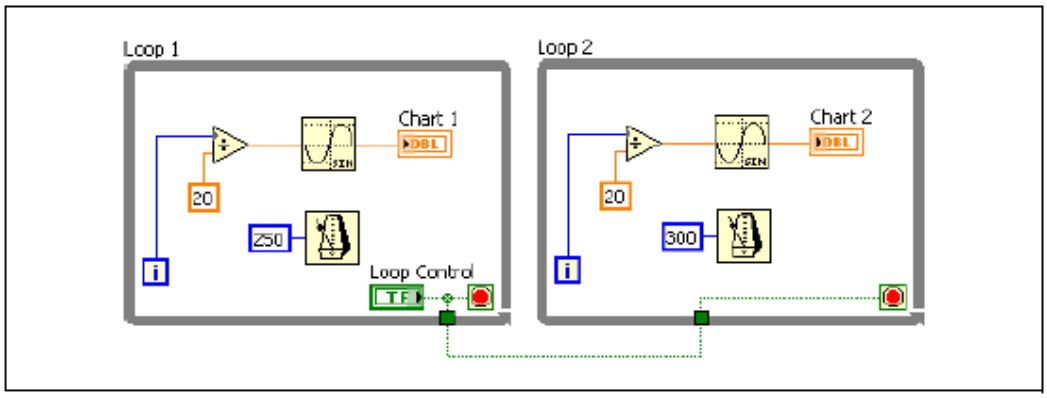


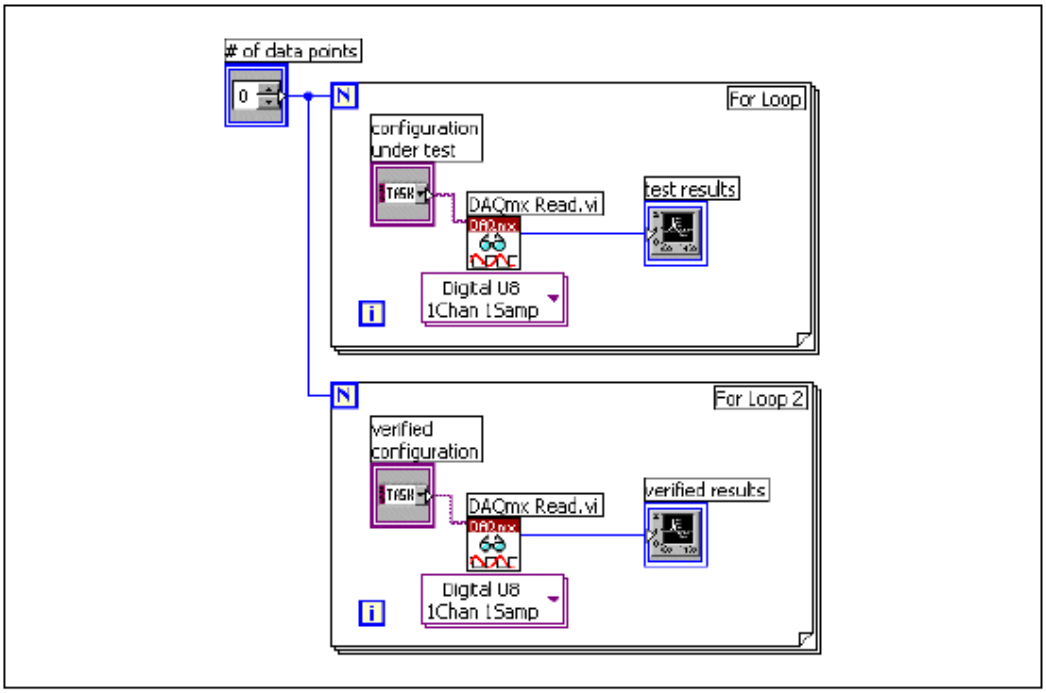
Рисунок 9-3. Параллельные циклы, метод 2

Метод 3 (Решение)

Если бы вы могли считать значение Loop control из файла, зависимости между потоками данных между циклами больше не существовало бы, поскольку каждый цикл осуществляет доступ к файлу независимо. Однако операции чтение и записи файлов требуют времени, по меньшей мере - времени процессора. Другой способ - найти место, где данные Loop control могли бы храниться в памяти и напрямую считывались бы оттуда. Далее в этой лекции будет показаны методы решения этой проблемы.

B. Переменные

В LabVIEW именно поток данных, а не последовательный порядок команд определяет последовательность выполнения элементов на блок-диаграмме. Благодаря этому можно создать блок-диаграмму, на которой операции выполняются одновременно. Например, вы можете одновременно запустить два цикла For и отобразить результаты на лицевой панели, как показано на приведенной блок-диаграмме.



Однако если вы будете использовать проводники для передачи данных между параллельными блок-диаграммами, они больше не будут параллельными. Параллельными блок-диаграммами могут быть два параллельных цикла на одной блок-диаграмме без зависимости потока данных, либо два отдельных VI, вызываемых одновременно.

В блок-диаграмме на рисунке 9-4 два цикла не выполняются параллельно из-за проводника между двумя subVIs.

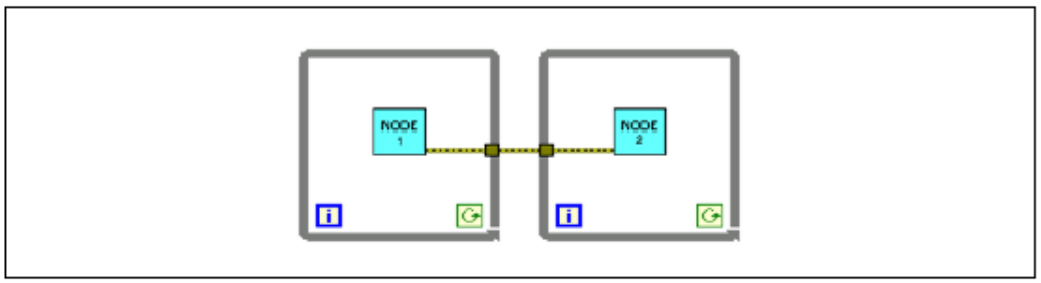


Рисунок 9-4. Зависимость между данными, вызванная проводником

Проводник создает зависимость данных, потому что второй цикл не начинается, пока первый не закончится и не передаст данные через туннель.

Чтобы два цикла работали параллельно, удалите проводник. Используйте другой способ для передачи данных между subVIs, например, переменную.

В LabVIEW *переменные (variables)* – элементы блок-диаграммы, позволяющие вам получать доступ к данным или сохранять данные в другом месте. Настоящее местоположение данных меняется в зависимости от типа переменной. Локальные переменные (Local Variables) хранят данные в элементах управления и индикации лицевой панели. Глобальные переменные (Global Variables) и однопроцессные переменные общего доступа(Single-Process Shared Variables) хранят данные в специальных репозитариях, которые могут быть доступны из разных VI. Функциональные глобальные переменные (Functional Global Variables) хранят данные в сдвиговых регистрах цикла While. Независимо от того, где хранятся данные, все переменные позволяют обойти проблему нормального потока данных путем передачи данных из одного места в другое без проводников. Поэтому переменные полезны в параллельной архитектуре, но они имеют и определенные недостатки, например, открывают возможность возникновения состязаний.

Переменные, используемые в одном VI

Локальные переменные используются для обмена данными в одном VI.

В LabVIEW вы считываете или записываете данные из объекта лицевой панели, используя его терминал на блок-диаграмме. Однако объект лицевой панели имеет только один терминал, а вашему приложению может понадобиться обратиться к данным в этом терминале более чем в одном месте.

Локальные и глобальные переменные передают информацию между точками в приложении, которые вы не можете соединить проводником. Используйте локальные переменные для доступа к объектам лицевой панели из нескольких мест в одном VI. Используйте глобальные переменные для доступа к данным и передачи их между несколькими VIs.

Используйте Feedback Node для запоминания данных из предыдущего VI или итерации цикла.

Создание локальных переменных

Щелкните правой кнопкой мыши по объекту лицевой панели или терминалу блок-диаграмме и выберите из контекстного меню **Create»Local Variable** для создания локальной переменной. Иконка локальной переменной появится на блок-диаграмме.

Вы можете также выбрать локальную переменную из палитры функций и поместить ее на блок-диаграмму. Узел локальной переменной еще не связан с элементов управления или индикатором.

Чтобы связать локальную переменную с элементом управления или индикатором, щелкните правой кнопкой мыши по локальной переменной и

выберите из контекстного меню **Select Item**. В раскрытом контекстном меню содержатся все элементы лицевой панели, имеющие собственные метки.

LabVIEW использует собственные метки для связи локальных переменных с объектами лицевой панели, поэтому присваивайте элементам управления и индикаторам лицевой панели содержательные метки.

Чтение и запись переменных

После создания переменной вы можете читать из нее данные или записывать в нее данные. По умолчанию новая переменная настроена на прием данных. Такая переменная работает как индикатор и является локальной или глобальной переменной для записи. При записи новых данных в локальную или глобальную переменную, обновляется связанный с ней элемент управления или индикатор лицевой панели.

Вы можете также настроить переменную для работы в качестве источника данных. Щелкните правой кнопкой мыши по переменной и выберите из контекстного меню **Change To Read,** чтобы переменная вела себя, как элемент управления. Когда этот узел выполняется, VI считывает данные из связанного с переменной объекта лицевой панели.

Для настройки переменной на прием данных с блок-диаграммы щелкните по переменной правой кнопкой мыши и выберите из контекстного меню **Change To Write.**

На блок-диаграмме вы можете различить переменные для чтения от переменных для записи так же, как отличаете элемент управления от индикатора. У переменной чтения более толстая граница, как у элемента управления, а у переменной записи – более тонкая, как у индикатора.

Пример использования локальных переменных

В разделе *Параллелизм* данной лекции вы видели пример VI с параллельными циклами. На лицевой панели находится единственный переключатель, который останавливает генерацию данных, отображаемых на двух графиках. На блок-диаграмме данные для каждого графика генерируются в отдельном цикле While для использования собственных временных параметров в каждом цикле. В этом примере два цикла используют один переключатель для одновременного останова циклов.

Чтобы оба графика обновлялись должным образом, циклы While должны работать параллельно. Соединение циклов While проводником для передачи значений переключателя приведет к последовательному выполнению циклов. На рисунке 9-5 приведена блок-диаграмма этого VI с использованием локальной переменной для передачи данных.

Цикл 2 считывает значение связанной с переключателем локальной переменной. При установке переключателя на лицевой панели в положение False, терминал переключателя в цикле 1 записывает это значение в терминал условия цикла 1. Цикл 2 считывает локальную переменную **Loop**

Control и записывает False в терминал условия цикла 2. Таким образом, циклы работают параллельно и одновременно выключаются при изменении положения переключателя на лицевой панели.

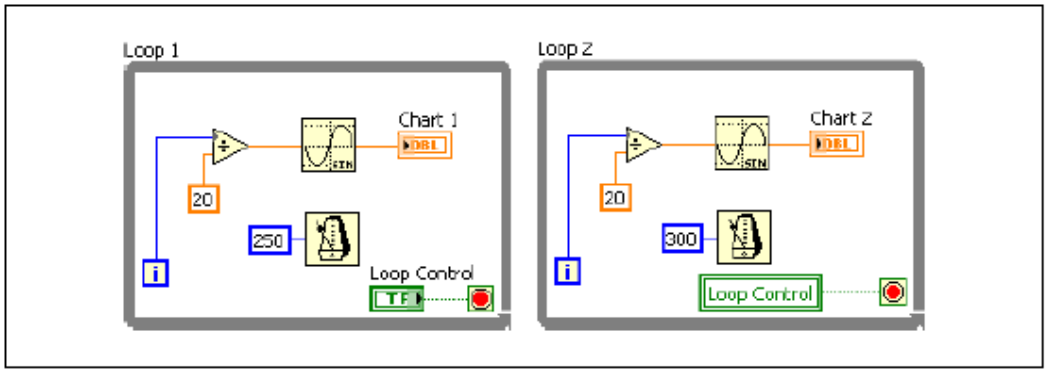


Рисунок 9-5. Использование локальной переменной для останова параллельных циклов.

При помощи локальной переменной вы можете записывать или читать элемент управления или индикатор лицевой панели. Запись в локальную переменную аналогична передаче данных на любой другой терминал. Однако при помощи локальной переменной вы можете записывать в нее значения, даже если это элемент управления, и читать данные, даже если это индикатор. То есть при помощи локальной переменной вы можете обращаться к объекту лицевой панели и как ко входу, и как к выходу.

Например, если интерфейс пользователя требует входа в систему, вы можете очищать поля **`Login`** и **`Password`** каждый раз при входе в систему нового пользователя. Используйте локальную переменную для чтения данных из строковых элементов управления **Login** и **Password** при входе пользователя в систему и записи пустых строк в эти элементы правления при выходе из системы.

Переменные для обмена данными между VIs

Вы можете также использовать переменные для передачи данных между несколькими одновременно выполняющимися VIs. Локальная переменная передает данные в пределах одного VI. Глобальная переменная передает данные между несколькими VI. Предположим, у вас есть два одновременно запущенных VI, каждый из которых содержит цикл While и выводит данные на график. Первый VI содержит булевский элемент управления для останова обоих VI. Вы можете использовать глобальную переменную для останова обоих циклов единственным булевским элементом управления. Если бы оба цикла были бы на блок-диаграмме одного и того же VI, вы могли бы использовать для этого локальную переменную.

Вы можете также использовать вместо глобальной переменной переменную общего доступа типа Single Process. Переменная общего доступа (Shared Variable) подобна локальной и глобальной переменной, но позволяет передавать данные по сети. Переменная общего доступа может быть использована на локальном компьютере (single-process) или публикуемой в сети (network-published). Хотя публикуемые в сети переменные общего

доступа не рассматриваются в данном курсе, вы всегда можете преобразовать используемую вами переменную Single Process в переменную, публикуемую в сети Network-Published.

Глобальные переменные применяют для обмена данными между VI, выполняющимися на одном компьютере, особенно, если не используется файл проекта. Однопроцессные переменные общего доступа применяют, если в будущем потребуется разделять информацию переменных между VI, выполняющимися на нескольких компьютерах.

Создание глобальных переменных

Глобальные переменные используются для передачи данных между несколькими одновременно выполняющимися VI. Глобальные переменные – встроенные объекты LabVIEW. При создании глобальной переменной LabVIEW автоматически создает специальный глобальный VI, у которого есть лицевая панель, но нет блок-диаграммы. Добавьте элементы управления и индикаторы на лицевую панель глобального VI, чтобы определить типы данных содержащихся в нем глобальных переменных. Другими словами, эта лицевая панель – контейнер, из которого несколько VI могут получать доступ к данным.

Предположим, у вас есть два одновременно запущенных VI, каждый из которых содержит цикл While и выводит данные на график. Первый VI содержит булевский элемент управления для останова обоих VI. Вы можете использовать глобальную переменную для останова обоих циклов единственным булевским элементом управления. Если бы оба цикла находились на блок-диаграмме одного и того же VI, вы могли бы использовать для этого локальную переменную.

Выберите глобальную переменную из палитры функций и поместите ее на блок-диаграмму.

Дважды щелкните по узлу глобальной переменной, чтобы отобразить лицевую панель глобального VI. Поместите туда элементы управления и индикаторы, как на обычную лицевую панель.

LabVIEW использует собственные метки для идентификации глобальных переменных, поэтому присваивайте элементам управления и индикаторам лицевой панели содержательные метки.

Вы можете создать несколько отдельных глобальных VI, каждый с одним объектом на лицевой панели, либо, если хотите сгруппировать похожие переменные, создать один глобальный VI с несколькими объектами лицевой панели.

Глобальная переменная с несколькими объектами более эффективна, поскольку вы можете группировать родственные переменные. Блок-диаграмма VI может содержать несколько узлов глобальной переменной, связанных с элементами управления и индикаторами на лицевой панели глобальной переменной. Эти узлы глобальной переменной - или копии первого узла, помещенного вами на блок-диаграмму глобального VI, или

узлы глобальной переменной другого глобального VI , которые вы поместили в текущий VI. Глобальные VI помещаются в другие VI точно так же, как subVI. Каждый раз при помещении на блок-диаграмму нового узла глобальной переменной, LabVIEW создает новый VI, связанный только с этим узлом глобальной переменной и его копиями.

На рисунке 9-6 показана лицевая панель глобальной переменной с числовым элементом, строкой и кластером, содержащим число и булевский переключатель. На панели управления не показываются кнопки **Run, Stop** и прочие кнопки стандартной панели управления.

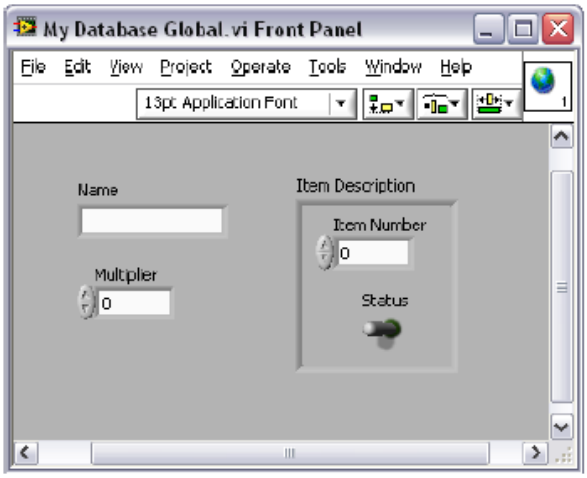


Рисунок 9-6. Окно лицевой панели глобальной переменной

Поместив желаемые объекты на лицевую панель глобального VI, сохраните его и вернитесь на блок-диаграму исходного VI. Далее вы должны выбрать объект в глобальном VI, к которому хотите получить доступ. Щелкните по узлу глобальной переменной и выберите объект лицевой панели из контекстного меню. В контекстном меню перечислены все объекты лицевой панели глобального VI с присвоенными метками. Вы можете также щелкнуть правой кнопкой мыши по узлу глобальной переменной и выбрать объект лицевой панели из контекстное меню **Select Item.**

Также вы можете щелкнуть по узлу глобальной переменной инструментами Operating или Labeling и выбрать объект лицевой панели из контекстного меню.

Если вы хотите использовать эту глобальную переменную в других VI, щелкните **Select a VI** на палитре функций. По умолчанию глобальная переменная связана с первым объектом лицевой панели с собственной меткой, который вы поместили в глобальный VI. Дважды щелкните по узлу глобальной переменной на блок-диаграмме и выберите объект лицевой панели из контекстного меню **Select Item** для связи глобальной переменной с данными из другого объекта лицевой панели.

Создание переменных типа Single Process

Для использования переменной общего доступа необходим файл проекта. Для создания переменной общего доступа типа Single Process щелкните правой кнопкой по строке **My Computer** в окне **Project Explorer** и выберите

New»Variable. Появится диалоговое окно **Shared Variable Properties**, показанное на рисунке 9-7.

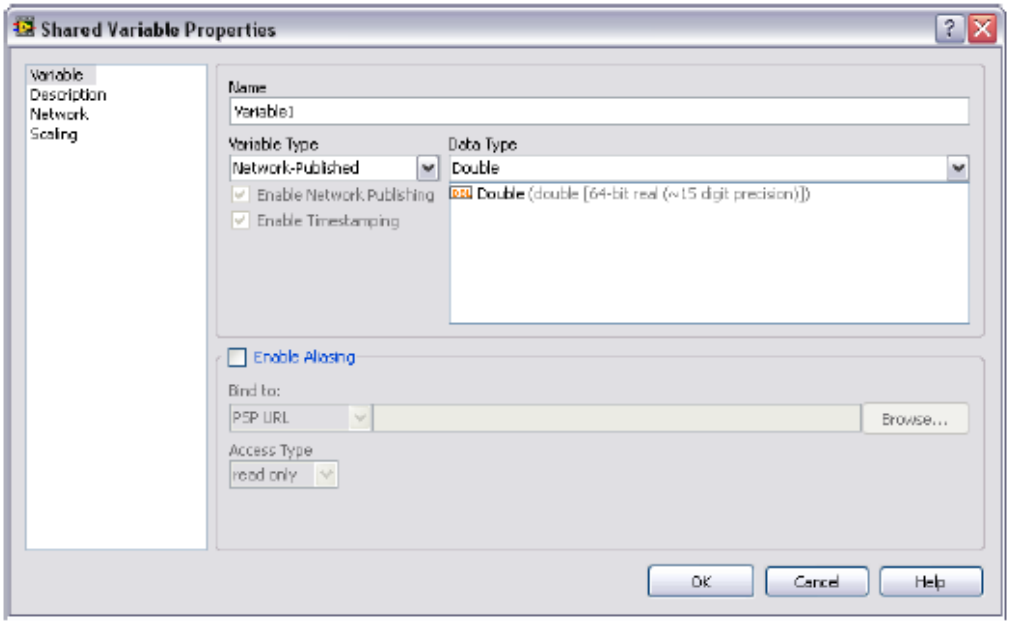


Рисунок 9-7. Диалоговое окно Shared Variable Properties

На вкладке **Variable Type** выберите **Single Process.** Задайте имя переменной (name) и тип данных (data type). После создания переменной общего доступа она автоматически появится в новой библиотеке в файле проекта. Сохраните эту библиотеку. При необходимости вы можете добавить в нее другие переменные общего доступа. Вы можете перетащить переменную из списка в окне **Project Explorer** прямо на блок-диаграмму. Используйте контекстное меню для переключения между чтением и записью. Используйте кластеры ошибок переменной для управления потоком данных.

Используйте переменные осторожно

Локальные и глобальные переменные – продвинутые концепции LabVIEW. Изначально они не являются частью модели потока данных LabVIEW. Блок-диаграмма может стать сложной для чтения при использовании переменных, так что вы должны использовать их осторожно. Неправильное использование локальных и глобальных переменных, например, использование их вместо панели подключения или для доступа к значениям в каждом кадре структуры Sequence, может привести к непредсказуемому поведению VI. Чрезмерное использование локальных и глобальных переменных, например, чтобы избежать длинных проводников на блок-диаграмме, или использование их вместо потока данных, снижает производительность.

Часто в использовании переменных нет необходимости. На рисунке 9-8 показан конечный автомат для управления движением на перекрестке. В каждом состоянии обновляется значение для следующего этапа последовательности световых сигналов. В показанном состоянии восточному и западному направлению дан зеленый свет, а северному и

южному – красный. Состояние ждет 4 секунды, что показывает функция Wait (ms).

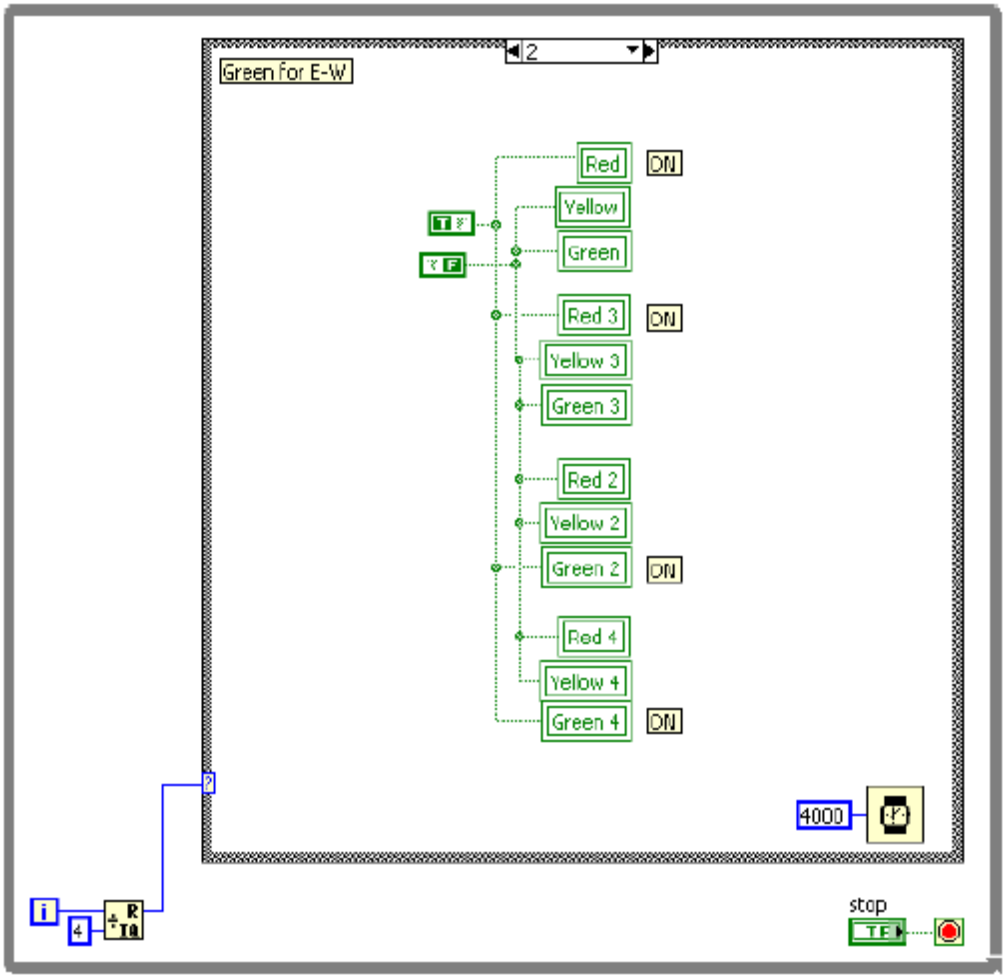


Рисунок 9-8. Использовано слишком много переменных

В примере, показанном на рисунке 9-9, выполняется та же задача, но более эффективно и использован лучший вариант реализации. Обратите внимание, что этот код значительно проще для чтения и понимания, чем показанный в предыдущем примере, в основном благодаря уменьшению количества используемых переменных. Благодаря помещению в цикл While снаружи структуры Case, индикаторы могут обновляться после каждого состояния без использования переменных. Этот пример легче модифицировать при дальнейшей разработке, например, при добавлении сигнала «поворот налево», чем предыдущий.

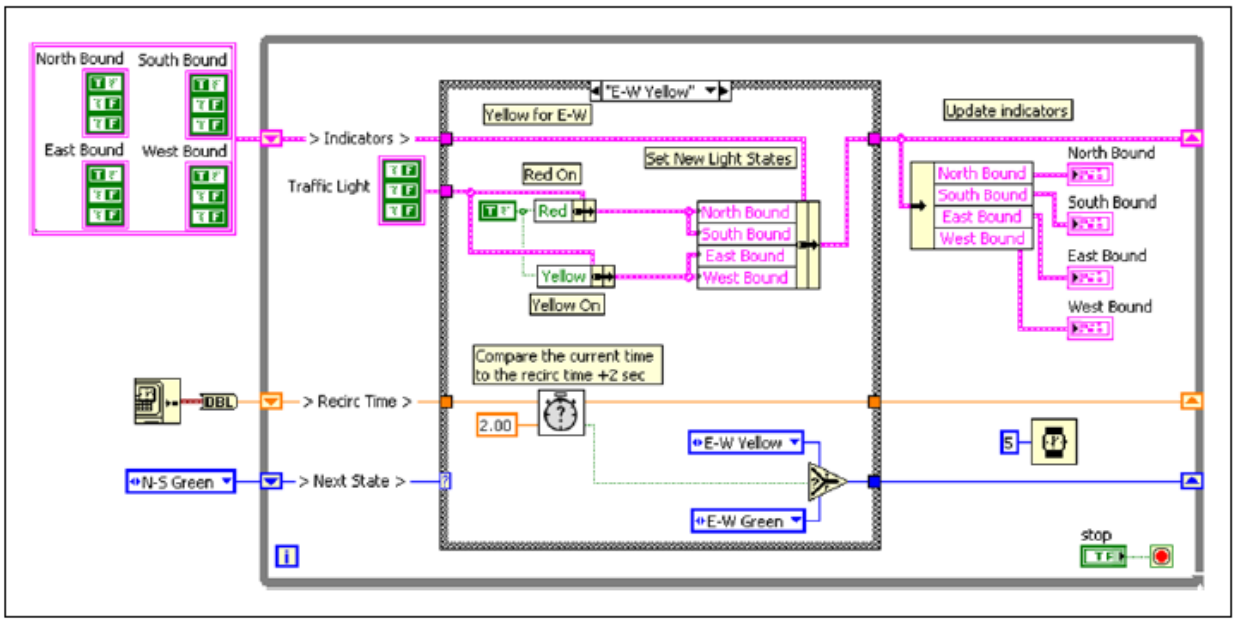


Рисунок 9-9. Уменьшение количества переменных

Инициализация переменных

Для инициализации локальной или глобальной переменной убедитесь, что до запуска VI переменная содержит известные значения. В противном случае переменные могут содержать данные, могущие привести к неправильной работе VI. Если в переменной используется результат вычислений как начальное значение, убедитесь, что LabVIEW записывает это значения в переменную, прежде чем предпринимает попытку обратиться к переменной с другими целями. Подключение записи в переменную параллельно с остальным VI может привести к возникновению состязаний.

Чтобы быть уверенными, что запись выполняется первой, вы можете изолировать код, записывающий начальное значение в переменную в первом кадре структуры Sequence или в subVI и подключить subVI первым в потоке данных на блок-диаграмме.

Если не проинициализировать переменную прежде, чем VI прочитает ее в первый раз, переменная будет содержать значение по умолчанию связанного с ней объекта лицевой панели.

На рисунке 9-10 показана распространенная ошибка при использовании переменных. Переменная общего доступа синхронизирует условия останова двух циклов. При первом запуске этот VI выполняется, поскольку значение по умолчанию булевского элемента равно False. Однако при каждом запуске этого VI после нажатия кнопки **Stop** в переменную записывается значение True. Поэтому при повторном и последующих запусках этого VI, нижний цикл останавливается после первой же итерации, хотя верхний цикл обновляет переменную достаточно быстро.

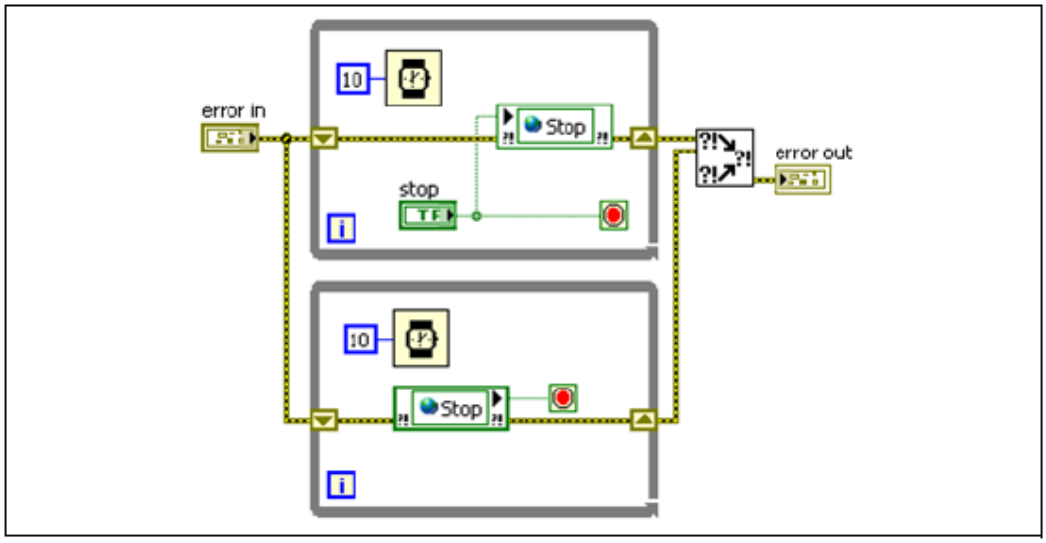


Рисунок 9-10. Ошибка при инициализации переменной общего доступа

На рисунке 9-11 показан скорректированный код данного VI с добавлением инициализации переменной общего доступа. Инициализируйте переменную до начала цикла, чтобы убедиться, что второй цикл не остановится сразу же.

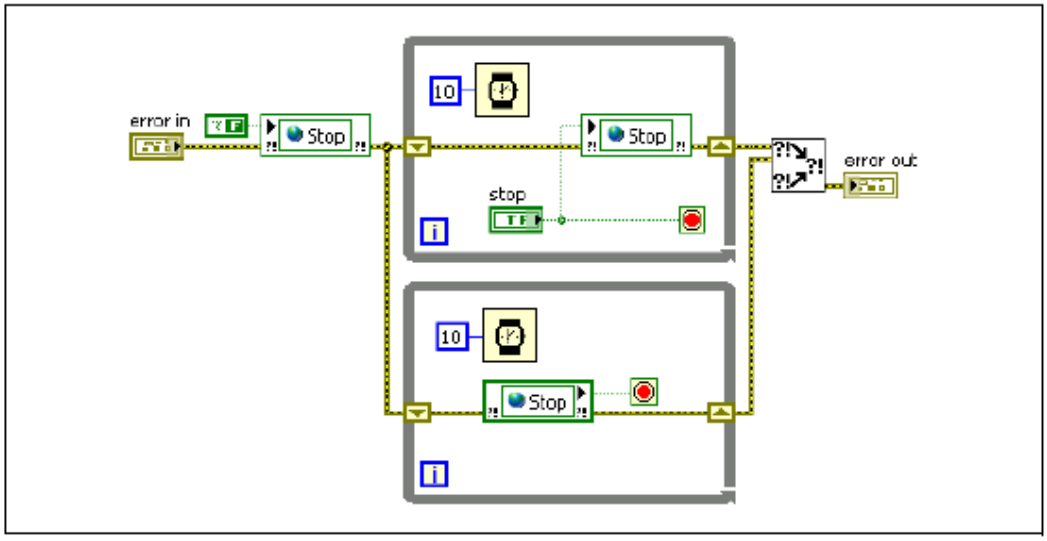


Рисунок 9-11. Правильная инициализация переменной общего доступа

C. Функциональные глобальные переменные

Вы можете использовать неинициализированные сдвиговые регистры в циклах For или While для сохранения данных до тех пор, пока VI не будет выгружен из памяти. В сдвиговом регистре хранится последнее состояние сдвигового регистра. Поместите цикл While в subVI и используйте сдвиговые регистры для хранения данных, чтобы в них можно было писать и из них можно было читать так же, как и в случае использования глобальных переменных. Такой метод обмена данными часто называют методом функциональной глобальной переменной. Преимущество применения этого метода по сравнению с методом глобальной переменной состоит в том, что вы можете управлять доступом к данным в сдвиговом

регистре. В обобщенном виде функциональная глобальная переменная содержит неинициализированный сдвиговый регистр в цикле For или While, который выполняется один раз, как показано на рисунке 9-12.

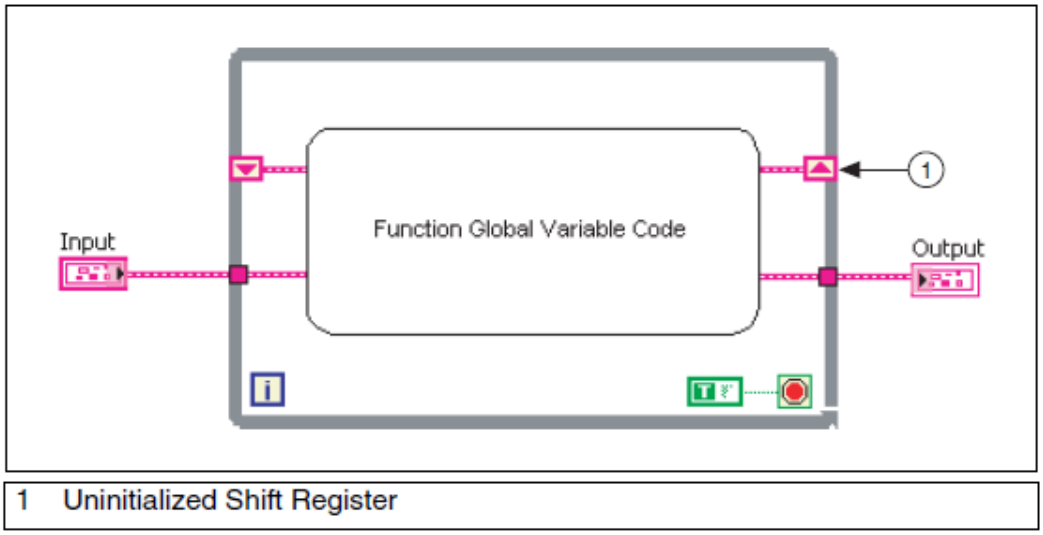


Рисунок 9-12. Формат функциональной глобальной переменной

Функциональная глобальная переменная обычно имеет входной параметр **action**, которым определяется задача, выполняемая VI. VI использует неинициализированный сдвиговый регистр в цикле While для хранения результата операции

На рисунке 9-13 показана простая функциональная глобальная переменная с возможностями установки и извлечения.

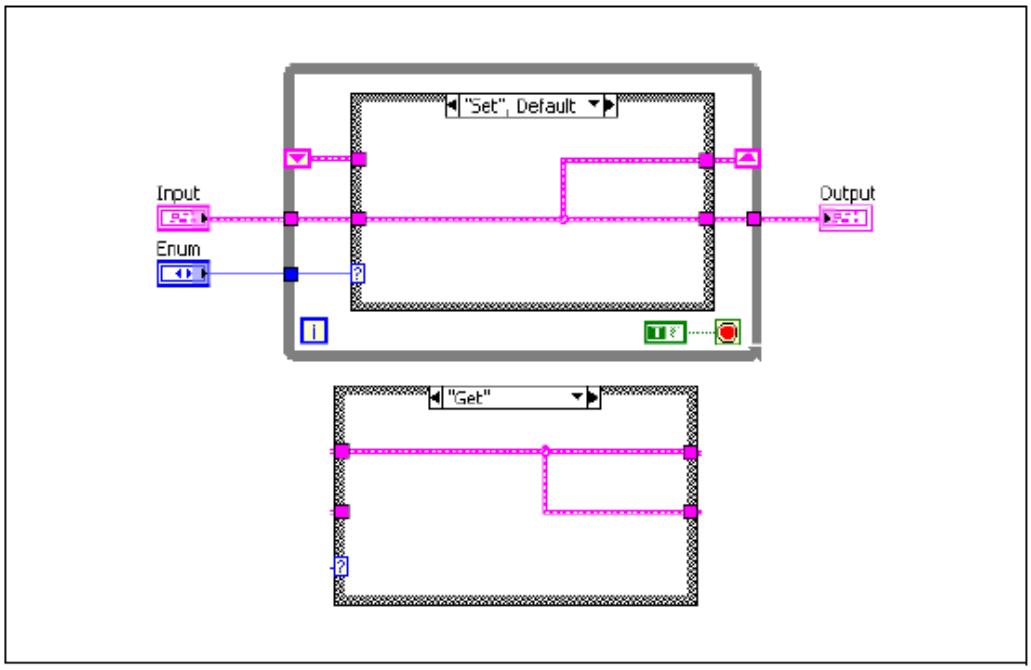


Рисунок 9-13. Функциональная глобальная переменная с возможностями установки и извлечения

В этом примере данные передаются в VI и хранятся в сдвиговом регистре, если вы установите перечислительный тип данных в режим Set. Данные извлекаются из сдвигового регистра, когда перечислительный тип данных настроен на Get.

Хотя вы можете использовать функциональные глобальные переменные для реализации простых глобальных переменных, как показано в предыдущем примере, они особенно полезны при реализации более сложных структур данных, например, стека или буфера очереди. Вы можете также использовать функциональные глобальные переменные для защиты доступа к глобальным ресурсам, таким, как файлы, измерительные приборы и устройства сбора данных, которые не можете представить глобальной переменной.

Примечание: Функциональная глобальная переменная является нереентерабельным subVI. Это означает, что при вызове subVI из нескольких мест, используется та же копия subVI. Таким образом, одновременно может происходить только один вызов subVI.

Использование функциональных глобальных переменных для настройки временных параметров

Эффективное приложение функциональных глобальных переменных – управление временными параметрами VI. Многие VI для измерения и автоматизации требуется некая форма синхронизации. Часто прибору или аппаратному устройству необходимо время для инициализации, и вы должны в явном виде задать временные параметры VI для принятия во внимание физического времени, требуемого для инициализации системы. Вы можете создать функциональную глобальную переменную для измерения времени между запусками VI, как показано на рисунке 9-14.

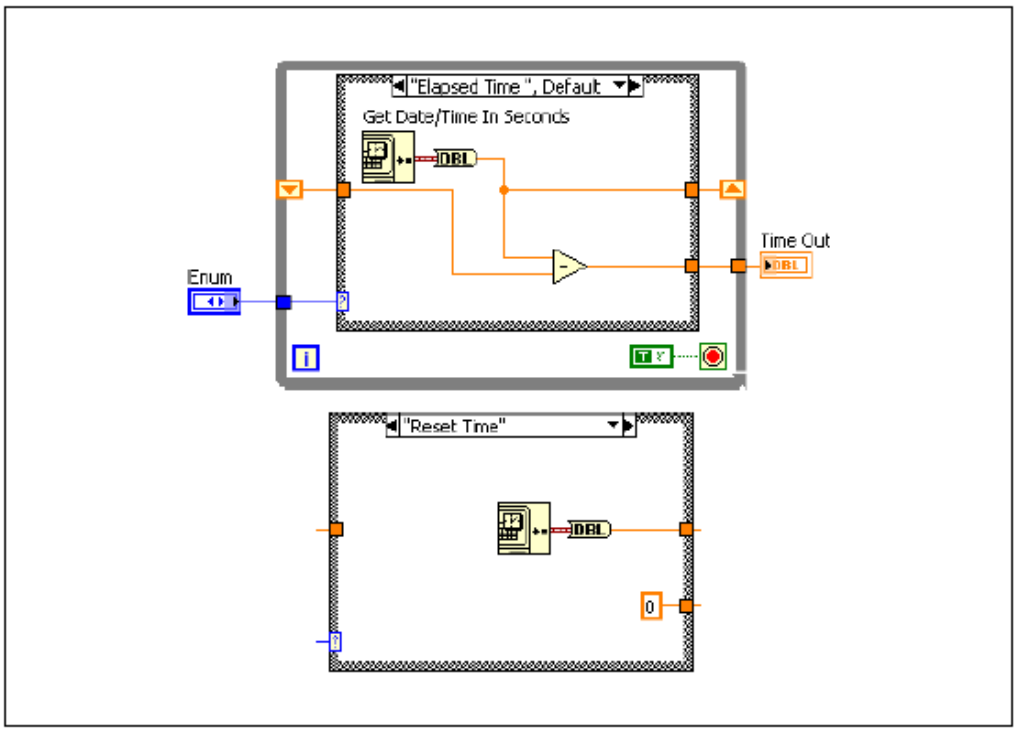


Рисунок 9-14. Функциональная глобальная переменная для определения прошедшего времени

Фрейм Elapsed Time получает текущую дату и время в секундах и вычитает полученный результат из времени, запомненного в сдвиговом регистре. Фрейм Reset Time инициализирует функциональную глобальную переменную известным значением времени.

Elapsed Time Express VI реализует функциональность, аналогичную данной функциональной глобальной переменной. Преимущество использования функциональной глобальной переменной в том, что вы можете проще настраивать проект, например, добавить возможность задания паузы.

D. Состязания

Состязания - ситуация, когда время возникновения событий или график выполнения задач может непредумышленно воздействовать на значения выходных данных. Состязания являются общей проблемой программ, в которых параллельно выполняются несколько задач, обменивающиеся между собой данными. Рассмотрим пример на рисунках 9-15 и 9-16.

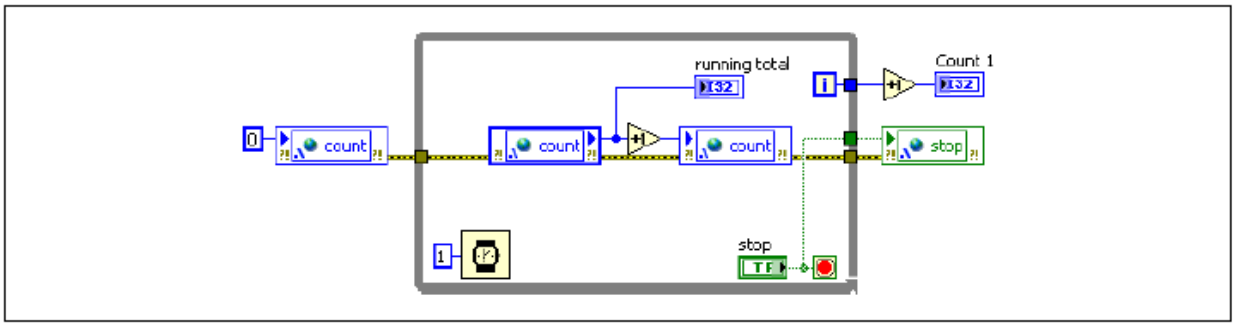


Рисунок 9-15. Пример возникновения состязаний: цикл 1

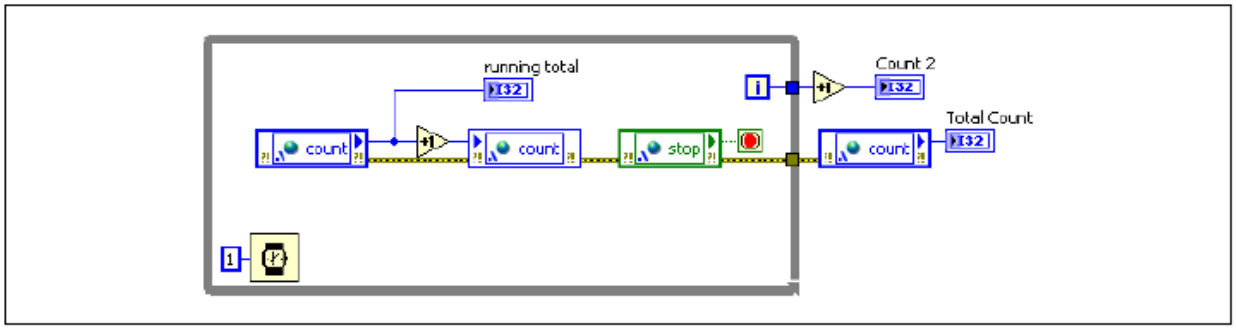


Рисунок 9-16. Пример возникновения состязаний: цикл 2

Каждый из циклов увеличивает значение переменной общего доступа с каждой итерацией. Ожидается, что после нажатия кнопки **Stop** значение **Total Count** равно сумме **Count 1** и **Count 2**. При запуске VI в течение короткого периода времени, скорее всего, так и будет. Однако, если VI выполняется длительное время, значение **Total Count** будет меньше суммы **Count** 1 и **Count 2**, потому что в этом VI возникает состязание.

На компьютере с одним процессором действия в подобной многозадачной программе на самом деле происходят последовательно, но LabVIEW и операционная система переключаются между задачами так, что задачи эффективно выполняются в одно и то же время. Состязание в этом примере случается, когда переключение с одной задачи на другую происходит в определенный момент. Обратите внимание, что оба цикла выполняют следующие операции:

- Чтение переменной общего доступа
- Увеличение значения счетчика чтений на единицу
- Запись инкрементированного значения в переменную общего доступа

Теперь рассмотрим, что произойдет, если операции цикла выполнятся в следующем порядке:

1. Цикл 1 считывает переменную общего доступа.
2. Цикл 2 считывает переменную общего доступа
3. Цикл 1 увеличивает считанное значение на единицу.
4. Цикл 2 увеличивает считанное значение на единицу..
5. Цикл 1 записывает увеличенное значение в переменную общего доступа.
6. Цикл 2 записывает увеличенное значение в переменную общего доступа.

В этом примере оба цикла записывают значения в одну и ту же переменную, и инкремент второго цикла записывается поверх инкремента первого цикла. Это порождает состязание, что может вызвать серьезные проблемы, если вы хотите, чтобы программа вычислила точное значение.

В приведенном примере между чтением переменной общего доступа и записью в нее имеются несколько инструкций. Поэтому менее вероятно, что VI переключится между циклами в неправильное время. Это объясняет, почему этот VI работает правильно, если работает не долго и теряет лишь несколько единиц при длительном выполнении.

Состязания очень сложно идентифицировать и отладить, потому что результат зависит от порядка, в каком операционная система выполняет задачи и временных параметров внешних событий. Способ взаимодействия задач друг с другом и операционной системой, а также случайные временные параметры внешних событий, делает этот порядок, в конечном счете, случайным. Часто код, в котором реализуются условия появления состязаний, может при тестировании тысячи раз возвращать одни и те же результаты, но существует возможность, что он выдаст результат, отличный от ожидаемого.

Избежать состязаний можно путем:

- Управления разделяемыми ресурсами и ограничения их использования
- Идентифицируя и защищая критические участки кода
- Задавая порядок выполнения

Управление и ограничение разделяемых ресурсов

Состязания обычно часто возникают, когда две задачи имеют доступ и по чтению, и по записи к одному ресурсу, как показано в предыдущем примере. Ресурс – это некоторый объект, являющийся общим для процессов. Чаще всего разделяемые ресурсы – это хранилища данных, например, переменные, а также файлы и ссылки на аппаратные ресурсы.

Доступ на изменение ресурса из нескольких источников часто приводит к возможности возникновения состязаний. Таким образом, идеальным способом избежать состязаний является минимизация общих ресурсов и ограничение количество источников для записи в оставшиеся общие ресурсы. Вообще, наличие нескольких «читателей» или «наблюдателей»

общего ресурса не приносит вреда. Однако старайтесь использовать только один источник для записи или контроллер при работе с общим ресурсом. Как правило, состязания возникают только, когда в ресурс обращаются по записи несколько источников.

В предыдущем примере вы может уменьшить зависимость от разделяемых ресурсов, заставив каждый цикл выполнять счет локально. Затем объедините финальные значения счетчиков после нажатия кнопки **Stop.** При этом потребуется единственное чтение и единственная запись в общий ресурс, что исключает возможности состязаний. Если у всех разделяемых ресурсов только один источник для записи или контроллер и в VI хорошо организован последовательный порядок выполнения, состязаний не возникает.

Защита критических фрагментов кода

Критический фрагмент кода – это код, который должен вести себя устойчиво в любых обстоятельствах. При работе с многозадачными программами одна задача может прерывать другую в процессе выполнения. Это происходит практически во всех современных операционных системах. Как правило, это не оказывает влияния на выполняющийся код, но, если прерывающая задача изменяет разделяемый ресурс, который прерываемая задача считает постоянным, возникает состязание.

На рисунках 9-15 и 9-16 показаны фрагменты критического кода. Если один цикл прерывается другим циклом, когда он выполняет критический фрагмент кода, возникают условия, при которых могут возникнуть состязания. Один из способов устранения возможности состязаний – идентификация и защита критических фрагментов вашего кода. Существует много методов для защиты критических фрагментов кода. Наиболее эффективные из них – функциональные глобальные переменные и семафоры.

Функциональные глобальные переменные

Один из способов защиты критических фрагментов кода – помещение их в subVIs. Нереентерабельный subVI в каждый момент времени можно вызвать только из одного места. Таким образом, помещение критического фрагмента кода в нереентерабельный subVI не дает коду быть прерванным другими процессами, вызывающими этот subVI. Использование архитектуры функциональных глобальных переменных для зашиты критических фрагментов особо эффективно, поскольку сдвиговые регистры могут заменить менее защищенные методы хранения, например, глобальные переменные или переменные общего доступа типа Single Process. Функциональные глобальные переменные способствуют также созданию многофункциональных subVI, обрабатывающих все задачи, связанные с конкретным ресурсом.

После определения всех критических фрагментов кода в VI сгруппируйте фрагменты в соответствии с используемыми ресурсами и создайте для каждого ресурса функциональную глобальную переменную. Каждый критический фрагмент, выполняющий отдельную операцию, может стать

командой для функциональной глобальной переменной. Вы можете сгруппировать критические фрагменты, выполняющие одну и ту же операцию, в одну команду, то есть повторно использовать код.

Вы можете применить функциональные глобальные переменные для защиты критических фрагментов кода на рисунках 9-15 и 9-16. Для устранения возможности возникновения состязаний, замените переменные общего доступа функциональной глобальной переменной и поместите код инкрементирования значения счетчика внутрь функциональной глобальной переменной, как показано на рисунках 9-17, 9-18 и 9-19.

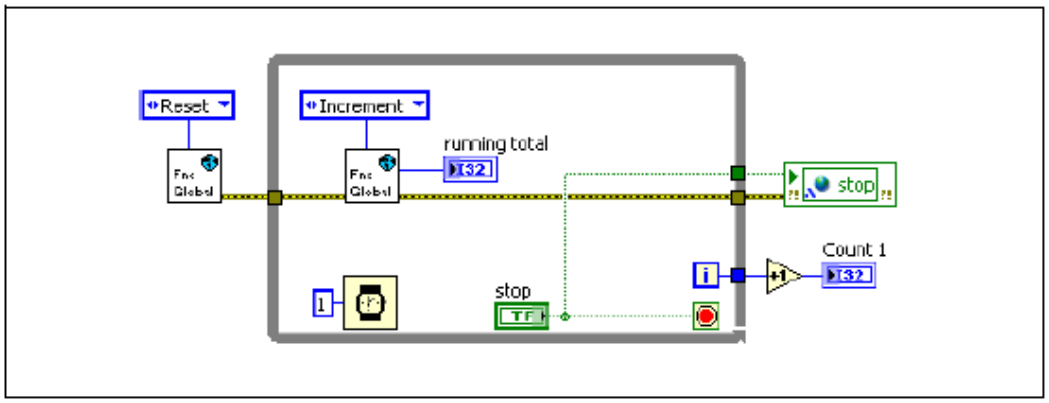


Рисунок 9-17. Использование функциональных глобальных переменных для защиты критического фрагмента кода в цикле 1

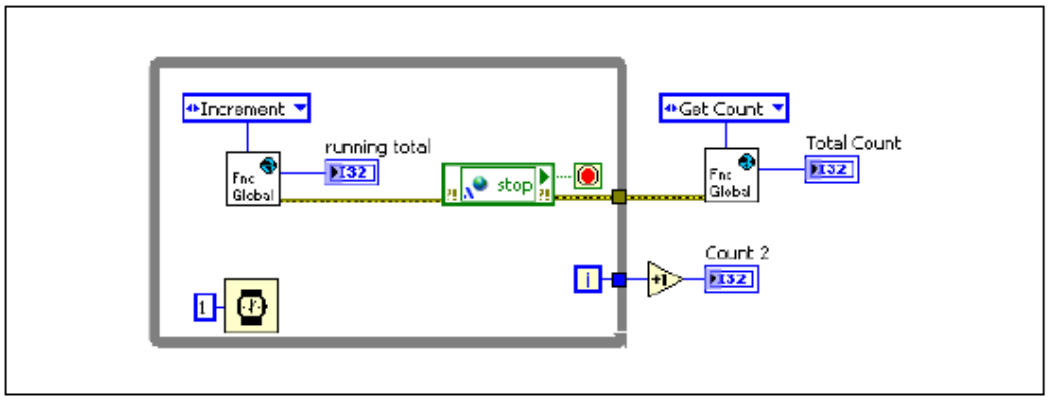


Рисунок 9-18. Использование функциональных глобальных переменных для защиты критического фрагмента кода в цикле 2

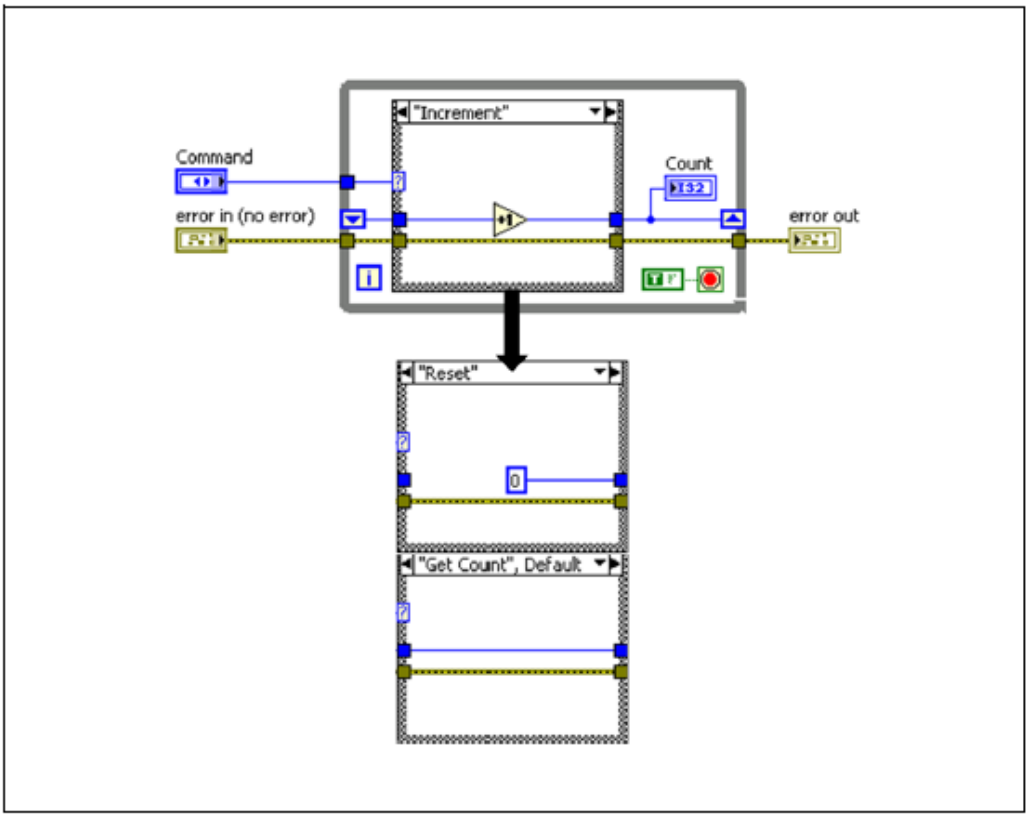


Рисунок 9-19. Функциональная глобальная переменная исключает возможность возникновения состязаний

Семафоры

Семафоры – механизмы синхронизации, специально разработанные для защиты ресурсов и критических фрагментов кода. Вы можете запретить критическим фрагментам кода прерывать друг друга, поместив каждый из них между Acquire Semaphore и Release Semaphore VI. По умолчанию семафор в каждый момент времени позволяет выполняться только одной задаче. Таким образом, после того как одна из задач начнет выполнять критический фрагмент кода, другая задача не сможет начать выполнения своего критического фрагмента, пока не завершится первая задача. При правильной реализации это исключает возможность возникновения состязаний.

Вы можете использовать семафоры для защиты критических фрагментов VI, показанных на рисунках 9-15 и 9-16. Именованный семафор позволяет вам разделять семафоры между VI. Вы должны открыть семафор в каждом VI, затем захватить его до начала критического фрагмента кода и освободить после критического фрагмента. На рисунках 9-20 и 9-21 показано, как избежать возникновения состязаний при помощи семафоров.

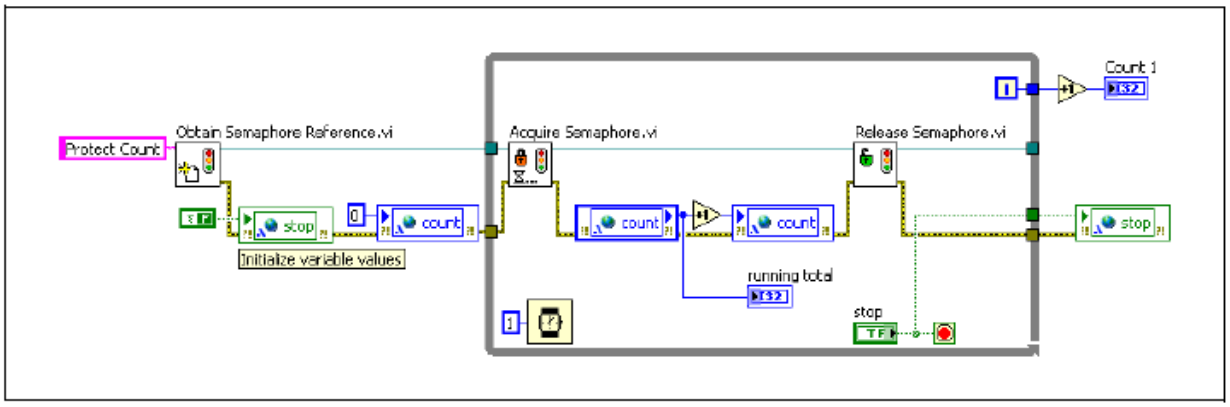


Рисунок 9-20. Защита критического фрагмента кода семафором в цикле 1

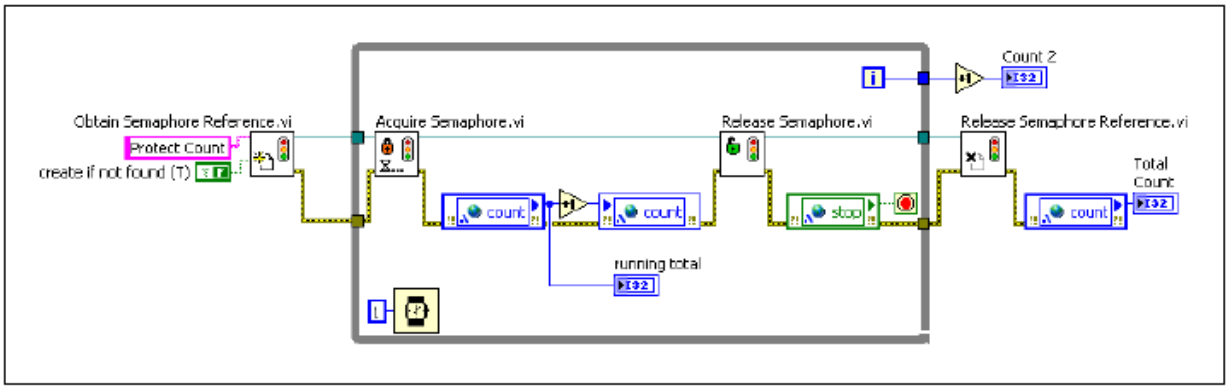


Рисунок 9-21. Защита критического фрагмента кода семафором в цикле 2

Определение порядка выполнения

Код с ненадлежащим использованием потока данных для управления порядком выполнения может привести к возникновению состязаний. Когда не установлена зависимость данных, LabVIEW может выполнять задачи в любом порядке, что может привести к возникновению состязаний, если задачи зависят друг от друга. Рассмотрим пример на рисунке 9-22.

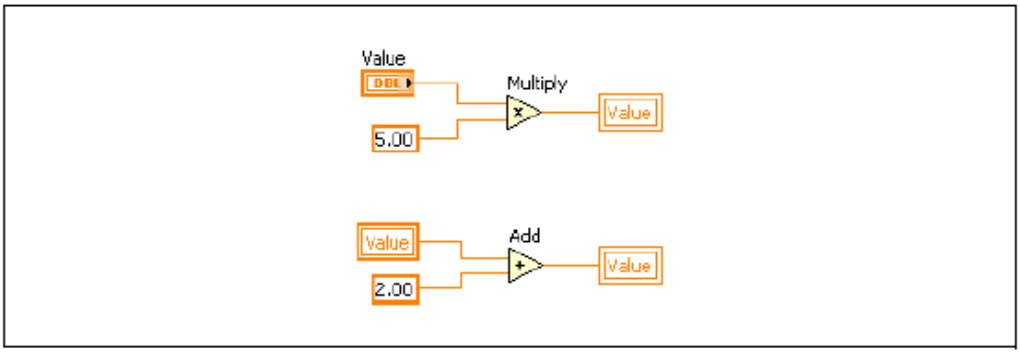


Рисунок 9-22. Простой пример возникновения состязаний

Код в этом примере может привести к четырем возможным результатам в зависимости от порядка выполнения операций.

Результат 1: Value = (Value Ч 5) + 2

1. Терминал считывает значение Value.
2. Value × 5 записывается в Value.
3. Локальная переменная считывает значение Value × 5.
4. (Value × 5) + 2 записывается в Value.

Результат 2: Value = (Value + 2) x 5

1. Локальная переменная считывает значение Value.
2. Value + 2 is записывается в Value.
3. Терминал считывает значение Value+2.
4. (Value + 2) × 5 записывается в Value.

Результат 3: Value = Value x 5

1. Терминал считывает значение Value.
2. Локальная переменная считывает значение Value.
3. Value + 2 записывается в Value.
4. Value × 5 записывается в Value.

Результат 4: Value = Value + 2

1. Терминал считывает значение Value.
2. Локальная переменная считывает значение Value.
3. Value × 5 записывается в Value.
4. Value + 2 записывается в Value.

Хотя в этом коде присутствует состязание, как правило, он ведет себя более предсказуемо, чем первый пример с состязаниями, поскольку обычно LabVIEW назначает последовательный порядок операций. Однако вы должны избегать ситуаций, подобных этой, потому что порядок выполнения и поведение VI может изменяться. Например, порядок может измениться при запуске VI при других условиях или при переходе на новую версию LabVIEW. К счастью, состязания подобной природы легко устраняются управлением потоком данных.

Самопроверка: короткий тест

1. Вы должны использовать переменные, где только возможно.
   a. Да
   b. Нет

2. Какие из объектов не могут передавать данные?
   a. Семафоры
   b. Функциональные глобальные переменные
   c. Локальные переменные
   d. Переменные общего доступа типа Single Process

3. Какие из объектов нужно использовать в проекте?
   a. Локальные переменные
   b. Глобальные переменные
   c. Функциональные глобальные переменные
   d. Переменные общего доступа типа Single Process

4. Какие из объектов не могут быть использованы для обмена данными между несколькими VI?
   a. Локальные переменные
   b. Глобальные переменные
   c. Функциональные глобальные переменные
   d. Переменные общего доступа типа Single Process

Самопроверка: ответы

1. Вы должны использовать переменные, где только возможно.
   a. Да
   b. Нет

2. Какие из объектов не могут передавать данные?
   a. Семафоры
   b. Функциональные глобальные переменные
   c. Локальные переменные
   d. Переменные общего доступа типа Single Process

3. Какие из объектов нужно использовать в проекте?
   a. Локальные переменные
   b. Глобальные переменные
   c. Функциональные глобальные переменные
   d. Переменные общего доступа типа Single Process

4. Какие из объектов не могут быть использованы для обмена данными между несколькими VI?
   a. Локальные переменные
   b. Глобальные переменные
   c. Функциональные глобальные переменные
   d. Переменные общего доступа типа Single Process

Заметки

A

Приложение A. Анализ и обработка числовых данных

Обычно пользователи начинают работу со сбора данных в приложение или программу, поскольку выполнение их задачи, как правило, требует взаимодействия с физическими процессами. Чтобы извлечь из данных полезную информацию, принять решение о процессе и получить результаты, данными необходимо управлять и анализировать их.

Как инструмент для решения инженерных задач, LabVIEW предоставляет сотни аналитических функций для исследователей, ученых, инженеров, а также для студентов и преподавателей. Они могут встраивать эти функции прямо в свои приложения для проведения интеллектуальных измерений и быстрого получения результатов.

План занятия

A. Выбор правильного метода анализа

B. Категории анализа

A. Выбор правильного метода анализа

Пользователи встраивают анализ в свои приложения и программы разными способами. Существуют ряд соображений, которые могут помочь определить способ, каким образом требуется проводить анализ.

Сравнение оперативного анализа и автономного анализа

Оперативный анализ (inline analysis) подразумевает, что данные анализируются в том же приложении, где и собираются. Как правило, он относится к приложениям, в которых необходимо принимать решения в процессе выполнения, а результаты оказывают непосредственное воздействие на процесс через изменение параметров или выполнения действий. Это типичный случай управляющих приложений. При выполнении оперативного анализа важно учитывать объем собранных данных и конкретные программы анализа данных. Необходимо найти подходящий баланс между ними, поскольку вычислительная сложность программ может стать настолько большой, что неблагоприятно скажется на производительности приложения.

Другие примеры применения оперативного анализа – приложения, где параметры измерений должны подстраиваться под характеристики измеряемого сигнала. Например, когда необходимо записывать один или несколько сигналов, которые изменяются очень медленно, за исключением неожиданных вспышек высокочастотной активности. Чтобы уменьшить объем записываемых данных, приложение должно быстро определить, когда требуется увеличить частоту и когда ее уменьшить по завершении вспышки. Измеряя и анализируя определенные аспекты сигналов, приложение может адаптироваться к условиям измерений и выбрать подходящие параметры выполнения. Хотя это лишь один пример, существуют тысячи приложений, где требуется определенная степень интеллектуальности – способности принимать решения на основании различных состояний - и адаптивности, что можно обеспечить, только реализуя в приложении алгоритмы анализа.

Принятие решений на основе собранных данных не всегда выполняется автоматически. Часто принятие решения связано с процессом, который наблюдает за выполнением и определяет, выполняется ли он, как ожидалось, или необходимо подстроить одну или несколько переменных. Хотя нередко пользователи записывают данные, азатем извлекают их из файлов или баз данных и автономно (offline) анализируют для корректировки процесса, частые изменения вынуждают проводить анализ в процессе работы. В подобных случаях приложение должно обрабатывать данные процесса, манипулировать ими, упрощать, форматировать и представлять данные в пригодном для пользователя виде. Пользователи LabVIEW могут воспользоваться множеством объектов визуализации для представления данных в наиболее компактной и полезной форме.

LabVIEW предлагает программы анализа и математические программы, естественно взамодействующие с функциями сбора данных и отображения, которые легко встроить в любое приложение. Кроме того, LabVIEW предлагает программы для поточечной обработки данных (point-by-point

execution); эти программы специально разработаны для удовлетворения необходимости оперативного анализа в приложениях реального времени. Пользователи должны учитывать некоторые аспекты при принятии решений об использовании программ, выполняющихся по точкам.

Автономные приложения (offline applications) обычно не требуют получения результатов в реальном времени для принятия решений о процессе. Подобным приложениям анализом нужны только достаточные вычислительные ресурсы. Главной целью таких приложений является определение причины и следствия влияния переменных на процесс, путем оценки корреляции между несколькими наборами данных. Как правило, эти приложения импортируют данных из пользовательских двоичных или ASCII-файлов и коммерческих баз данных наподобие Oracle, Access и других, поддерживающих стандарт SQL/ODBC. Как только данные импортированы в LabVIEW, пользователи могут выполнять множество доступных программ анализа, управлять данными и выводить их в заданном формате для отчетов.

LabVIEW предоставляет функции для доступа к любому формату файлов и баз данных, естественного подключения к мощным инструментам создания отчетов наподобие NI DIAdem и Report Generation Toolkit for Microsoft Office, и реализации современных технологий совместного использования данных наподобие XML, презентаций данных в Web и ActiveX.

Сравнение программного и интерактивного анализа

Пользователи LabVIEW, инженеры и ученые, хорошо знакомы с различными способами получения данных из сотен устройств. Они встраивают интеллектуальность в свои приложения для оперативного анализа и представления результатов в темпе выполнения приложения. Они знают также, что сбора данных и их визуализации в реальном времени недостаточно. Обычно пользователи сохраняют сотни и тысячи мегабайт данных на жестких дисках и в базах данных. После нескольких (от одного до ста) запусков приложения пользователи извлекают информацию для принятия решений, сравнивают результаты и вносят в процесс необходимые изменения до достижения желаемых результатов.

Относительно просто собрать настолько большие объемы данных, что они быстро станут неуправляемыми. Имея быструю плату сбора данных и достаточное количество каналов, можно собрать тысячи значений всего за несколько миллисекунд. Извлечь из всех этих данных нужную информацию – непростая задача. От инженеров и ученых, как правило, ожидается предоставление отчетов, создание графиков и, в конечном счете, подтверждения некоторых оценок и заключений по эмпирическим данным. Без необходимых инструментов это было бы чрезвычайно трудной задачей, приводящей к потере продуктивности.

Чтобы упростить анализ результатов измерений, программисты LabVIEW создали приложения, предоставляющие диалоги и интерфейсы, которые могут использовать другие. В соответствии со значениями входных параметров наборы данных обрабатываются специальными программами анализа. Создавая подобное приложение, пользователи встраивают в него

определенную интерактивность. Чтобы это было эффективно, программист должен обладать обширными знаниями об информации и типах анализа, которые необходимы пользователю.

В LabVIEW вы легко можете значительно уменьшить объем данных и отформатировать их до сохранения на диск, чтобы после извлечении собранных данных для анализа их было легче обрабатывать. LabVIEW также предоставляет бесчисленные функции для генерации отчетов на основании результатов измерений и информации, полученных из собранных данных.

B. Категории анализа

LabVIEW предлагает сотни встроенных функций анализа, покрывающих различные области и методы извлечения информации из собранных данных. Вы можете использовать эти функции без изменений или модифицировать их и настраивать под конкретные нужды. Эти функции объединены в следующие группы: Measurement, Signal Processing, Mathematics, Image Processing, Control, Simulation и Application Areas.

- Measurement
  - Amplitude and Level
  - Frequency (Spectral) Analysis
  - Noise and Distortion
  - Pulse and Transition
  - Signal and Waveform Generation
  - Time Domain Analysis
  - Tone Measurements
- Signal Processing
  - Digital Filters
  - Convolution and Correlation
  - Frequency Domain
  - Joint Time-Frequency Analysis (Signal Processing Toolset)
  - Sampling/Resampling
  - Signal Generation
  - Super-Resolution Spectral Analysis (Signal Processing Toolset)
  - Transforms
  - Time Domain
  - Wavelet and Filter Bank Design (Signal Processing Toolset)
  - Windowing

- Mathematics
  - Basic Math
  - Curve Fitting and Data Modeling
  - Differential Equations
  - Interpolation and Extrapolation
  - Linear Algebra
  - Nonlinear Systems
  - Optimization
  - Root Finding
  - Special Functions
  - Statistics and Random Processes
- Image Processing
  - Blob Analysis and Morphology
  - Color Pattern Matching
  - Filters
  - High-Level Machine Vision Tools
  - High-Speed Grayscale Pattern Matching
  - Image Analysis
  - Image and Pixel Manipulation
  - Image Processing
  - Optical Character Recognition
  - Region-of-Interest Tools
- Control
  - PID and Fuzzy Control
- Simulation
  - Simulation Interface (Simulation Interface Toolkit)
- Application Areas
  - Machine Condition Monitoring (Order Analysis Toolset)
  - Machine Vision (IMAQ, Vision Builder)
  - Motion Control
  - Sound and Vibration (Sound and Vibration Analysis Toolset)

Полный список функций анализа в LabVIEW можно найти на сайте `ni.com/analysis`.

Заметки

B

Приложение B. Основы измерений

В настоящем приложении объясняются понятия, не зная которых нельзя эффективно измерять и генерировать сигналы. На этих понятиях концентрируется осмысление назначения и функционирования компонентов измерительной системы, которые находятся вне компьютера. Вам предстоит изучить следующие вопросы: датчики, источники сигналов, согласование (кондиционирование) сигналов, заземление измерительной системы, способы повышения качества измерений. Настоящее приложение дает базовое представление о перечисленных понятиях.

План занятия

A. Применение компьютеризированных измерительных систем

B. Основы измерений

C. Повышение качества измерений

A. Применение компьютерных измерительных систем

Основной задачей всех измерительных систем является измерение и/или генерация реальных физических сигналов. Измерительные устройства помогают получать, анализировать и представлять результаты измерений.

Пример измерительной системы приведен на рисунке B-1. Прежде чем компьютерная измерительная система сможет измерить некоторую физическую величину, например, температуру, физический сигнал с помощью датчика или измерительного преобразователя должен быть преобразован в электрический – ток или напряжение. Чтобы улучшить качество измерений, возможно, потребуется выполнить согласование (кондиционирование) электрического сигнала, которое может включать фильтрацию с целью подавления шума, усиление или ослабление сигнала, чтобы привести его к допустимому диапазону измерений. После кондиционирования сигнал измеряется, и результат измерений вводится в компьютер.

В настоящем курсе изучаются два способа ввода измеряемого электрического сигнала в компьютер — с помощью устройства сбора данных (DAQ-устройства) или с помощью автономного измерительного прибора. Программное обеспечение осуществляет управление всей системой, которая выдает необработанные данные, обрабатывает их и представляет пользователю результаты. Путем указанных действий

физическую величину, которую желаете измерить, можно ввести в компьютер для анализа и отображения.

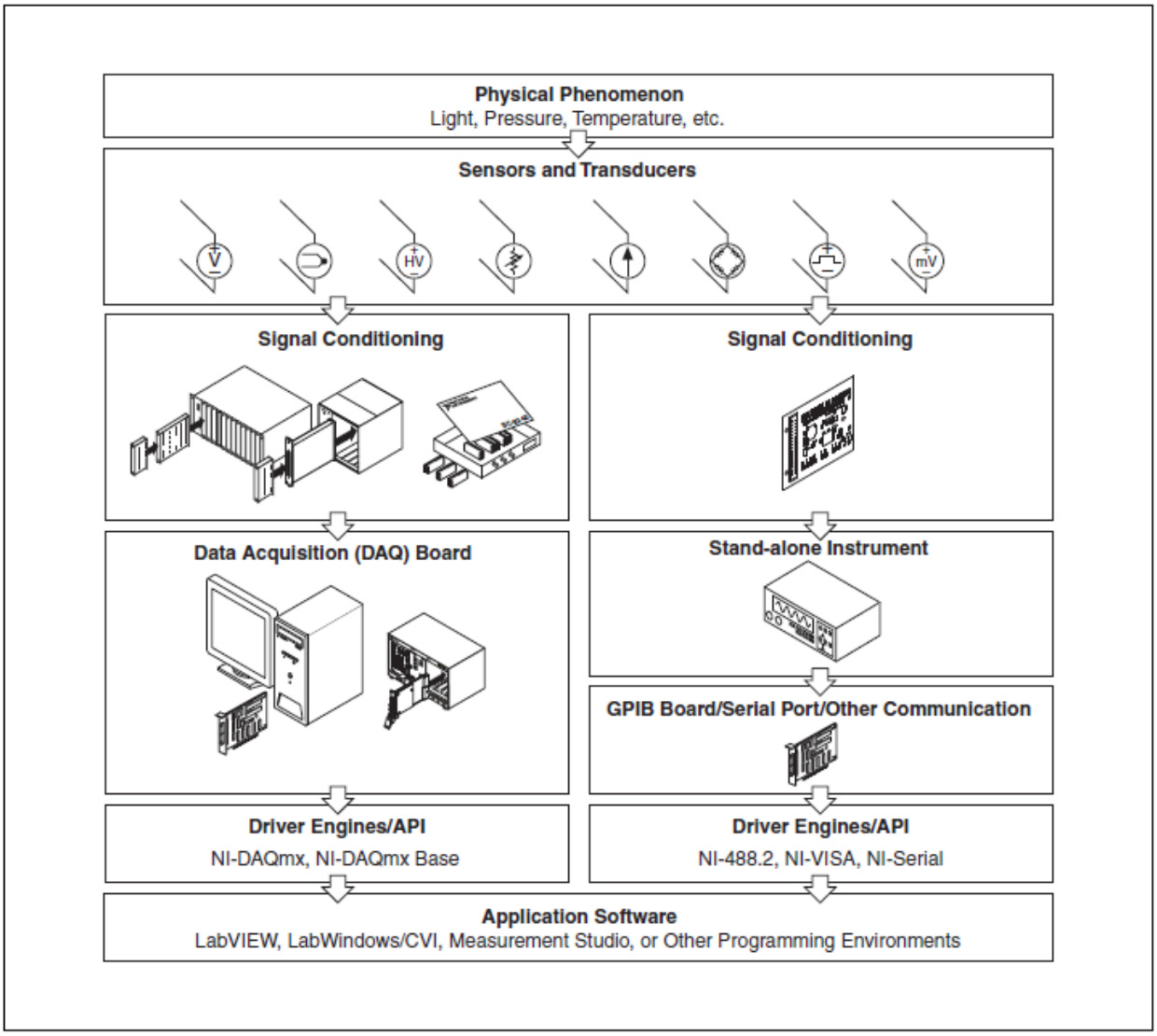


Рисунок B-1. Системы измерений

B. Основы измерений

В настоящем параграфе приводится информация об общих понятиях, с которыми нужно познакомиться прежде, чем приступить к измерениям с помощью DAQ-устройств и автономных измерительных приборов.

Сбор данных о сигналах

Под сбором данных будем понимать процесс измерения – преобразования физической величины в данные, с которыми может работать компьютер. Измерение начинается с преобразования физической величины в электрический сигнал. Измерительные преобразователи формируют электрические сигналы при измерении таких величин, как температура, сила, звук, свет и т.п. В таблице B-1 приведены некоторые распространенные типы преобразователей (датчиков).

Таблица B-1. Физические величины и датчики

Физическая величина	Датчики
Температура	Термопары Термометры сопротивления Термисторы Интегральные датчики
Свет	Фотодатчики на электронных лампах Фоторезисторы
Звук	Микрофоны
Сила и давление	Тензодатчики Пьезоэлектрические преобразователи Датчики нагрузки
Положение (перемещение)	Потенциометры Линейные дифференциальные трансформаторы Оптические кодеры
Поток жидкости	Манометры-расходомеры Турбинный (механический) расходомер Ультразвуковые расходомеры
pH	Электроды для измерения pH

Источники сигналов

Устройства ввода аналоговых сигналов в компьютер работают с заземленными и "плавающими" источниками сигнала.

Заземленные источники сигнала

Заземленный источник — это такой источник, выходное напряжение которого снимается относительно заземления системы, например, относительно шины заземления здания (рис. B-2). Подобный источник имеет общую "землю" с измерительным прибором. Наиболее распространенными заземленными источниками сигналов являются устройства, которые через настенную розетку питания подключаются к заземлению здания, например, генераторы сигналов и источники питания.

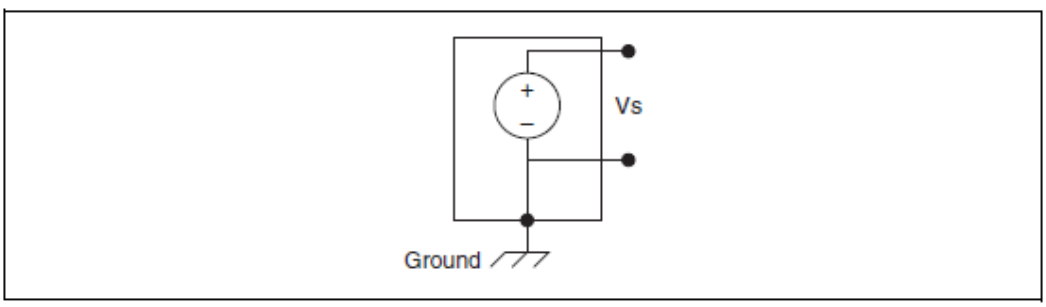


Рисунок B-2. Заземленный источник сигнала

Примечание. Обычно общий контакт ("заземления") двух независимо заземленных источников имеют разные потенциалы. Разность потенциалов между "землями" двух приборов, подключенных к системе заземления одного и того же здания, составляет от 10 до 200 мВ. Если силовая подводка электроэнергии выполнена неправильно, эта разность может быть и выше. В подобных ситуациях говорят о паразитных контурах заземления

Плавающие источники сигнала

У плавающего источника сигнала выходное напряжение не связано с общей цепью заземления (рис. B-3). Распространенными примерами плавающих источников являются гальванические элементы, термопары, трансформаторы и изолирующие усилители. Обратите внимание на то, что на рис. B-3. ни один из выводов источника не подключен к цепи заземления, как у источника на рис. B-2, поэтому выходной сигнал плавающего источника не зависит от системы заземления.

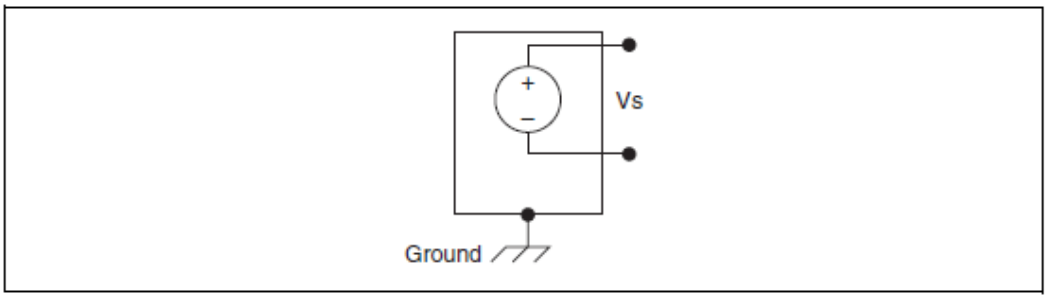


Рисунок B-3. Плавающий (изолированный) источник сигнала

Кондиционирование сигналов

Под кондиционированием сигналов следует понимать процесс предварительной обработки сигналов с целью улучшения точности измерений, качества изоляции цепей (развязки), фильтрации и т.д. У многих автономных измерительных приборов и DAQ-устройств есть встроенные средства кондиционирования сигналов. Кондиционирование сигналов можно также выполнять за пределами измерительного прибора или DAQ-устройства, если спроектировать схему кондиционирования или применить устройства, специально предназначенные для этой цели. Для кондиционирования сигналов компания National Instruments выпускает SCXI-устройства и другие изделия. Далее на примере различных DAQ-устройств и SCXI-устройств проиллюстрированы вопросы кондиционирования сигналов.

Чтобы измерять сигналы с датчиков, необходимо преобразовать их в форму, которую может воспринять измерительный прибор. Например, у большинства термопар выходное напряжение очень мало и соизмеримо с шумом. Следовательно, перед оцифровкой такого сигнала его необходимо усилить. Усиление является одной из форм кондиционирования. К другим типовым разновидностям кондиционирования сигналов относятся линеаризация, возбуждение датчика, развязка.

На рис. B-4. показаны некоторые распространенные типы датчиков и сигналов и требуемые для них виды кондиционирования.

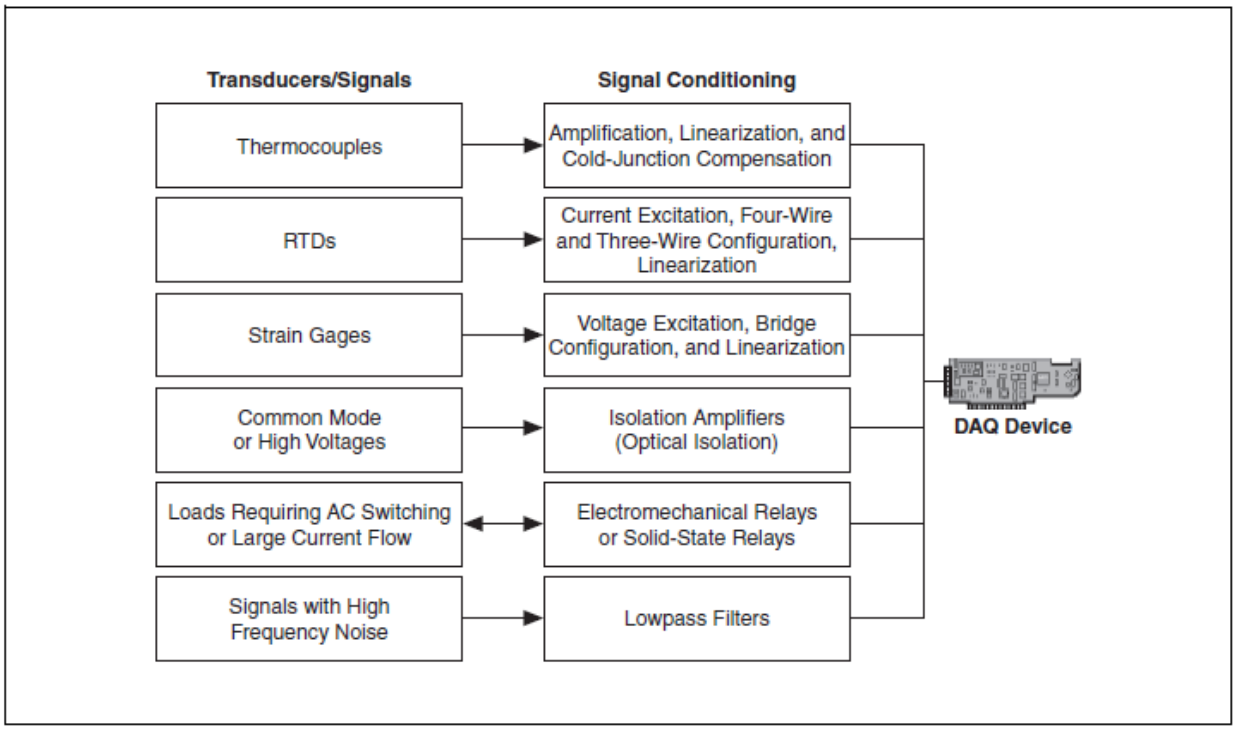


Рисунок B-4. Распространенные типы датчиков и разновидности преобразования сигналов

Усиление

Усиление является наиболее часто применяемой разновидностью кондиционирования сигналов, позволяющей уменьшить влияние помех и повысить точность представления сигнала после оцифровки.

Для увеличения отношения сигнал/помеха усиление сигналов должно производиться как можно ближе к источнику сигнала, поскольку вместе с сигналом усиливается и любая помеха. Расположение усилителя вбдизи источника сигнала увеличивает отношение сигнал/шум. При этом наивысшая точность измерений может быть достигнута, если диапазон изменения усиленного напряжения соответствует максимальному диапазону входных напряжений аналого-цифрового преобразователя (АЦП).

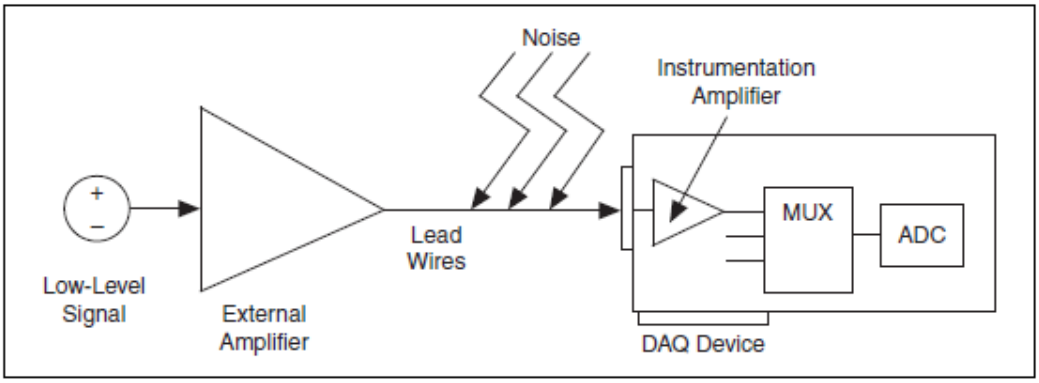


Рисунок B-5. Усиление сигналов

Если сигнал усиливается в устройстве сбора данных (DAQ), то аналого-цифровому преобразованию подвергается сумма сигнала и помехи, которая может быть наведена на соединительные проводники, поэтому отношение сигнал/помеха уменьшается. Однако если сигнал усиливать непосредственно у источника с помощью модуля кондиционирования (SCXI), помеха искажает сигнал в меньшей степени и можно получить лучшее цифровое представление значений слабого сигнала. Подробную информацию об аналоговых сигналах можно найти на сайте National Instruments `ni.com/info` по ключу `exd2hc`.

Линеаризация

Многие датчики, такие как термопары, обладают нелинейной зависимостью выходного сигнала от измеряемой величины. LabVIEW позволяет линеаризовать напряжение, поступающее от датчиков, так что можно легко отмасштабировать значения напряжения в единицах измеряемой величины. В LabVIEW имеются функции, обеспечивающие масштабирование сигналов от датчиков деформации, терморезисторов, термопар и термисторов.

Возбуждение датчика

Системы кондиционирования сигналов могут вырабатывать возбуждающие воздействия, которые требуются для функционирования некоторых датчиков. Так, при измерении физических величин с помощью тензодатчиков и терморезисторов, необходимо подать внешнее напряжение или ток на измерительную схему, в которую включены датчики. Это напоминает радиоприемник, которому для приема и преобразования аудиосигналов нужен источник питания.

Развязка

Для обеспечения надежности применяется такая разновидность кондиционирования сигналов, как изоляция (развязка) выходных цепей датчика от компьютера.

Внимание! Источник сигнала нельзя подключать непосредственно к устройству сбора данных без какой-либо развязки, если контролируемый сигнал содержит большие выбросы напряжения, которые могут вывести из строя компьютер или опасны для оператора

Изоляцию датчика от компьютера можно использовать, чтобы исключить влияние разности потенциалов в контурах заземления на результаты измерений с помощью устройства сбора данных. Если устройство сбора данных и источник сигнала заземлены не в одной и той же точке, может появиться паразитный контур заземления, который послужит причиной дополнительных погрешностей измерения. Большая разность потенциалов между точками заземления источника сигнала и устройства сбора данных может даже вывести из строя измерительную систему. Для устранения

паразитных контуров заземления и повышения точности измерения сигнала используется гальваническая развязка.

Измерительные системы

Измерительную систему необходимо сконфигурировать в соответствии с решаемыми задачами и с учетом используемого оборудования.

Измерительные системы с дифференциальными входами

Измерительные системы с дифференциальными входами используются совместно с плавающими (незаземленными) источниками сигнала. Ни один из дифференциальных входов измерительной системы не соединен с общей шиной или заземлением здания. Примерами подобных систем являются портативные приборы с батарейным питанием и устройства сбора данных с инструментальными (измерительными) усилителями.

Типичное устройство производства National Instruments, в котором реализована 8-канальная измерительная система с дифференциальными входами, показано на рис. B-6. Аналоговые мультиплексоры (MUX) в измерительных цепях позволяют увеличить количество измерительных каналов, при этом достаточно одного инструментального усилителя. На рис. B-6 вывод AIGND (заземление цепей аналогового ввода) является заземлением измерительной системы.

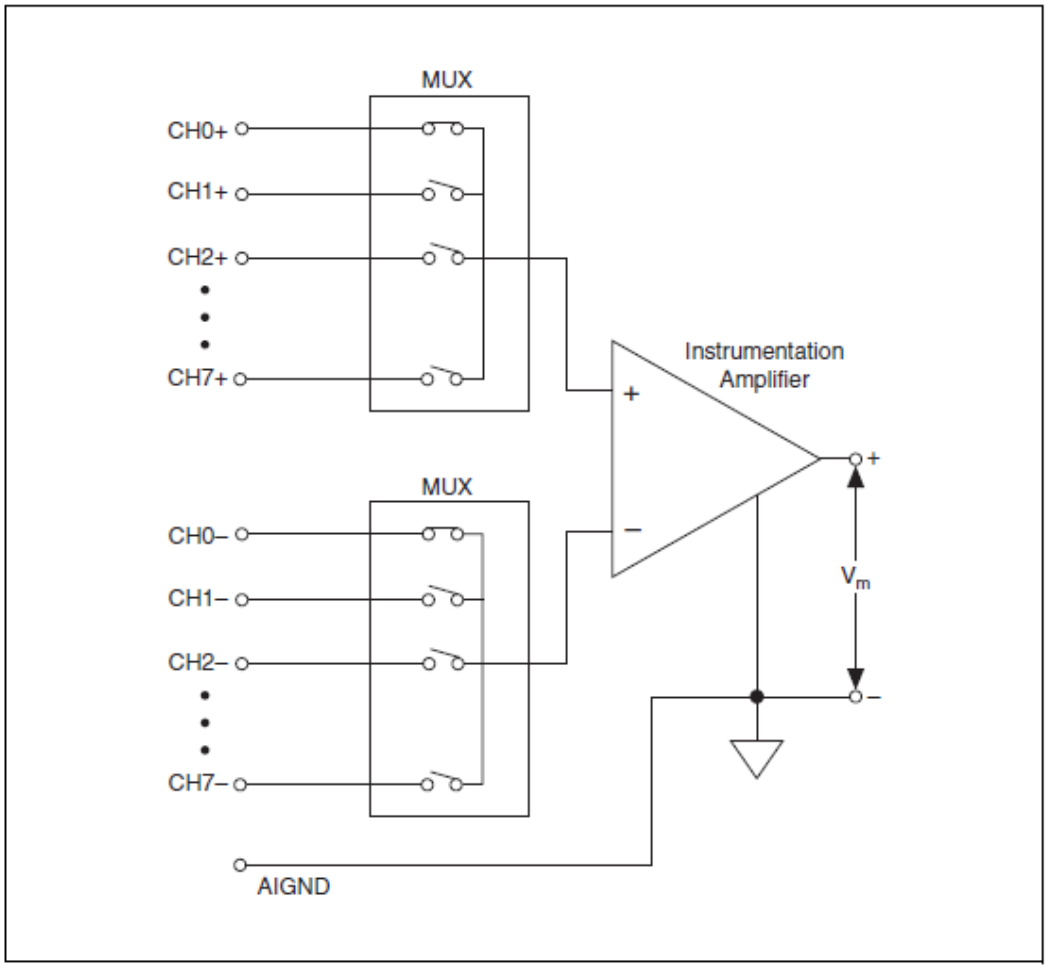


Рисунок B-6. Типичная измерительная система с дифференциальным входом

Заземленные и незаземленные измерительные системы с несимметричным входом

Заземленные и незаземленные измерительные системы с несимметричным входом используются с источниками сигналов с общей точкой и измерения в них выполняются относительно общей точки. Если система с несимметричным входом заземлена (Referenced Single-Ended Measurement Systems – RSE), то напряжение измеряется относительно вывода заземления аналогового ввода AIGND, непосредственно соединенного с заземлением самой системы. На рис. B-7 показана 16-канальная измерительная система с несимметричным входом и заземленным общим проводом.

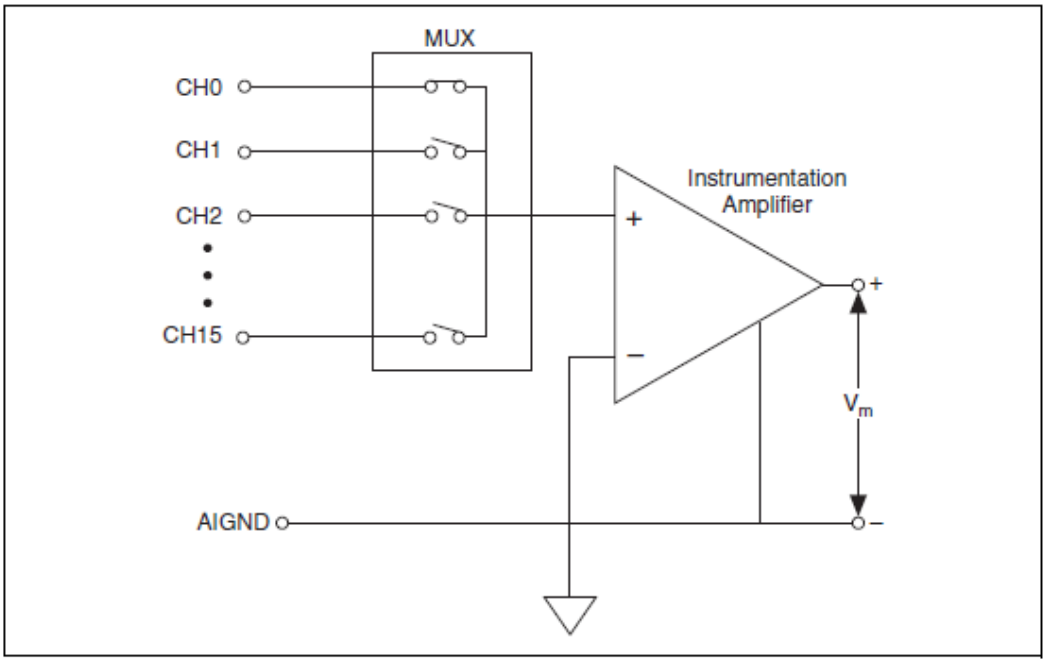


Рисунок B-7. Типичная измерительная система с несимметричным входом и заземленным общим проводом

В устройствах сбора данных часто применяются схемы измерений с несимметричными входами без заземления общего провода (Non-Referenced Single-Ended Measurement Systems – NRSE). Схема реализации подобных подключений является вариантом схемы с заземленными несимметричными входами (рис. B-8).

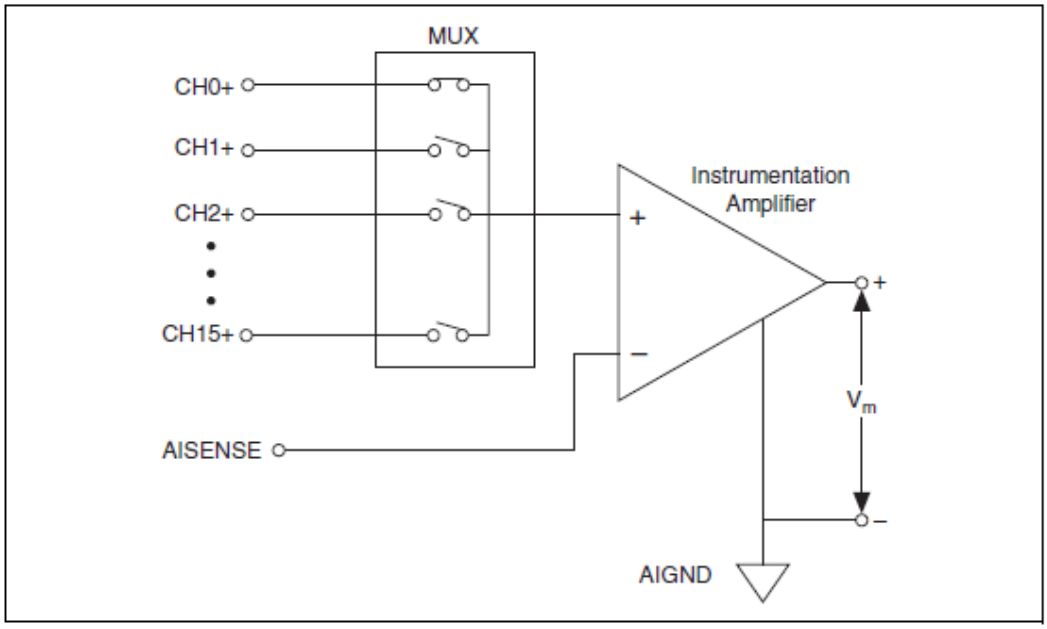


Рисунок B-8. Типичная измерительная система с несимметричным входом и незаземленным общим проводом

В системах типа NRSE все измерения проводятся относительно общего вывода аналоговой части схемы (в устройствах сбора данных E серии он обозначен AISENSE), однако потенциал этого вывода может значительно отличаться от потенциала заземления системы (AIGND). Одноканальная

система с незаземленным несимметричным входом аналогична одноканальной дифференциальной системе.

Выводы по способам подключения источников сигналов к измерительным системам

На рис. B-9 обобщены способы подключения источника сигнала к измерительной системе.

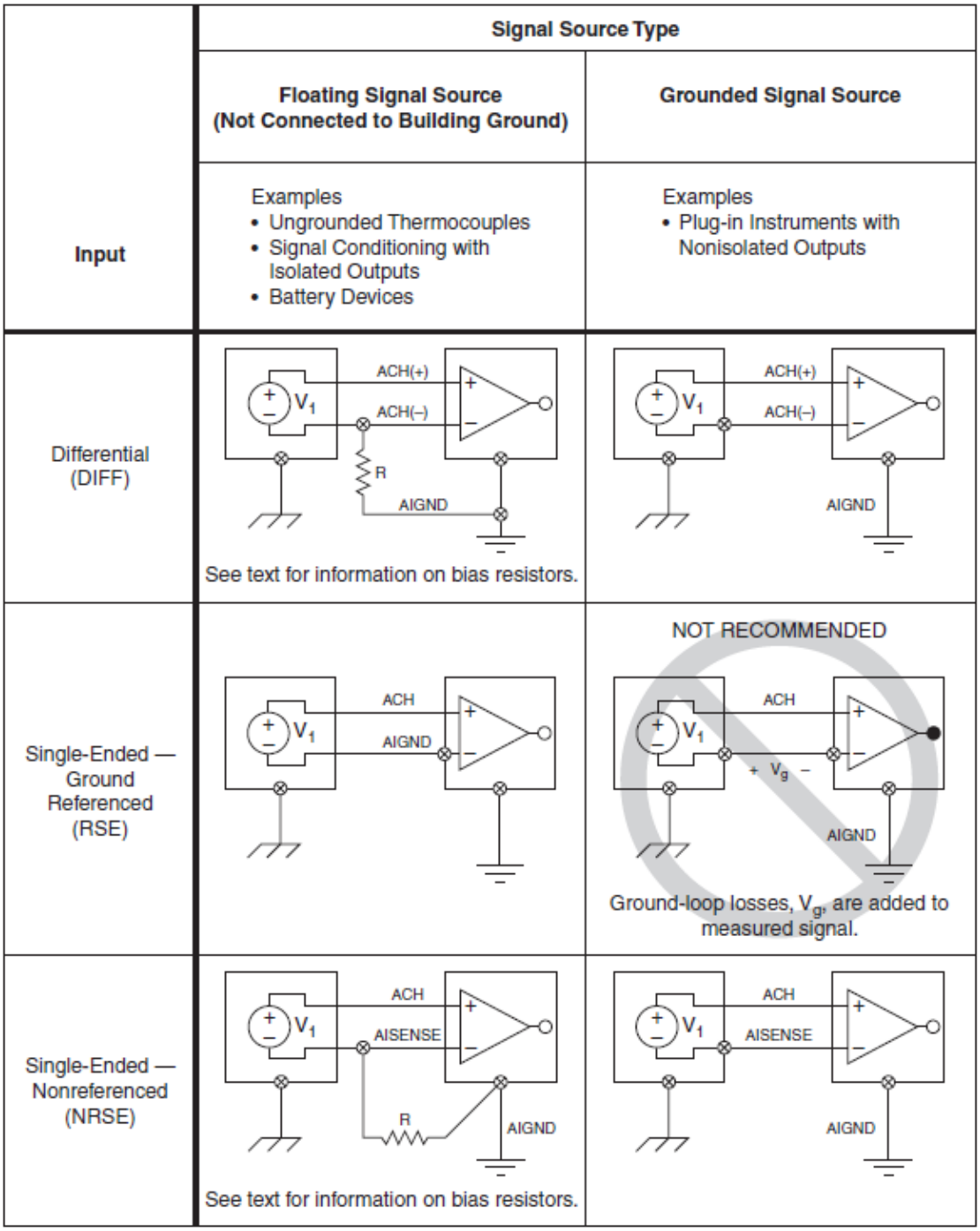


Рисунок B-9. Источники сигналов и системы измерений

C. Повышение качества измерений

При проектировании измерительной системы может оказаться, что она не удовлетворяет требованиям по качеству. Быть может, вам захочется зафиксировать самое маленькое изменение уровня напряжения. Также, вероятно, что у вас не получится определить, какая форма у сигнала: треугольная или пилообразная, и в таком случае вы пожелаете увидеть более качественное изображение формы сигнала. Зачастую возникает потребность уменьшить содержание шума в сигнале. Настоящий параграф знакомит с тремя методами улучшения упомянутых характеристик измерений.

Обнаружение наименьшего изменения сигнала

На уровень наименьшего изменения напряжения сигнала, которое можно обнаружить, влияют следующие факторы:

- Разрешающая способность и диапазон входного напряжения АЦП
- Коэффициент усиления инструментального усилителя
- Цена младшего значащего разряда, которая вычисляется на основании разрешения, диапазона входных напряжений и коэффициента усиления

Разрешающая способность

Разрешающая способность — это наименьшее значение изменения входного сигнала, которое может быть зафиксировано каким-либо устройством или датчиком. Разрешающая способность АЦП определяется количеством бит, используемых для представления аналогового сигнала. Можно провести аналогию между разрешающей способностью средства измерений и количеством меток на линейке. Чем больше меток, тем более точны будут измерения. Точно так же, чем выше разрешающая способность, тем больше количество уровней квантования, на которые измерительная система может разбить диапазон значений на входе АЦП, и, следовательно, тем меньше обнаруживаемое изменение входного напряжения.

Например, у 3-разрядного АЦП диапазон входного напряжения делится на $2^3 = 8$ уровней квантования. Каждому уровню соответствует своя двоичная кодовая комбинация в диапазоне от 000 до 111. АЦП преобразует каждый отсчет измеряемого аналогового сигнала в одну из кодовых комбинаций. На рис. B-10 изображен результат аналого-цифрового преобразования синусоидального сигнала с помощью 3-разрядного АЦП. Очевидно, что такой цифровой сигнал не является адекватным представлением исходного сигнала, поскольку преобразователь имеет слишком мало уровней квантования для представления изменяющихся напряжений аналогового сигнала. Однако при увеличении разрядности до 16 бит, количество уровней квантования АЦП возрастает с 8 до 65536 (2^{16}). В этом случае АЦП обеспечивает весьма точное представление аналогового сигнала.

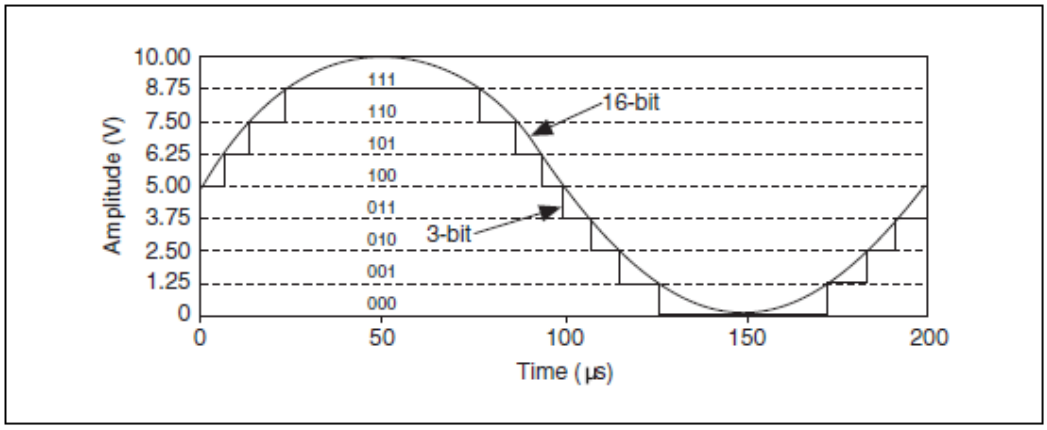


Рисунок B-10. Пример 3-битного и 16-битного разрешения

Диапазон измерений

Диапазон измерений — минимальный и максимальный уровни аналогового сигнала, которые АЦП может преобразовать в цифровую форму. Большинство измерительных приборов позволяют выбирать один из нескольких диапазонов (как правило, от 0 до 10 В или от –10 до 10 В) путем изменения режима с униполярного на биполярный или путем выбора одного из коэффициентов усиления, что позволяет полностью использовать разрешающую способность (шкалу) АЦП для оцифровки сигнала.

Униполярный и биполярный режимы

В униполярном режиме измерительное устройство поддерживает диапазон измерений от 0 В до +*X* В. В биполярном режиме оно поддерживает диапазон измерений от –X В до +X В. Некоторые устройства поддерживают только один из режимов, в то время как другие устройства могут переключаться с униполярного на биполярный режим.

Внимание! Устройства, которые могут переключаться с униполярного на биполярный режим, способны выбирать такой режим, который больше всего подходит для измеряемого сигнала. График 1 на рисунке B-11 иллюстрирует униполярный режим для 3-разрядного АЦП. Такой АЦП имеет восемь уровней квантования в диапазоне измерений от 0 до 10 В. В биполярном режиме, как показано на графике 2 на рисунке B-11, диапазон измерений — от –10.00 В до 10.00 В. Теперь тот же самый АЦП разбивает на восемь уровней квантования уже диапазон напряжений 20 В. Наименьшая фиксируемая разность напряжений увеличивается с 1.25 В до 2.5 В, и теперь представление сигнала получается значительно менее точным. Устройство выбирает наиболее подходящий режим в зависимости от диапазона входного сигнала, который задается при создании виртуального канала.

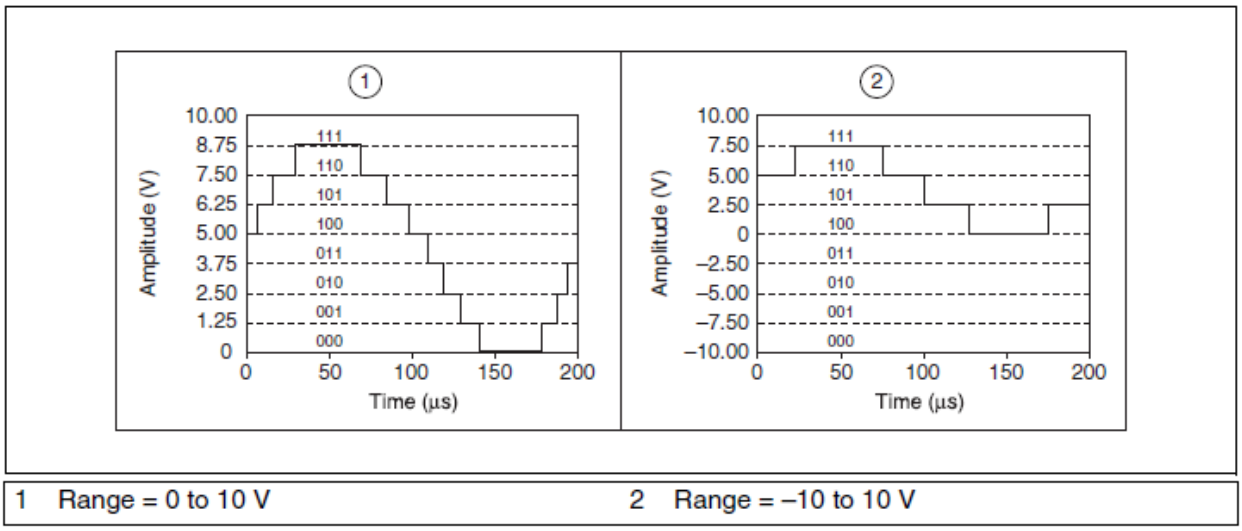


Рисунок B-11. Пример диапазона

Усиление

Чтобы улучшить представление сигнала, перед аналого-цифровым преобразованием он подвергается усилению или ослаблению, что позволяет эффективно использовать диапазон входных напряжений АЦП, в результате чего будет использоваться максимально возможное количество уровней квантования для представления сигнала.

Например, на рисунке B-12 показано, как влияет усиление на сигнал, который изменяется в диапазоне от 0 до 5 В, при использовании 3-разрядного АЦП с диапазоном входных напряжений от 0 до 10 В. При отсутствии усиления (при единичном коэффициенте усиления), АЦП использует в процессе преобразования только четыре из восьми уровней квантования. В результате усиления сигнала в два раза АЦП использует уже восемь уровней квантования, и соответствующее представление в цифровой форме становится более точным. Это справедливо при условии, что допустимый диапазон входных напряжений — от 0 до 10 В, поскольку любой сигнал, который превышает 5 В, при усилении в два раза дает напряжение на входе АЦП более 10 В.

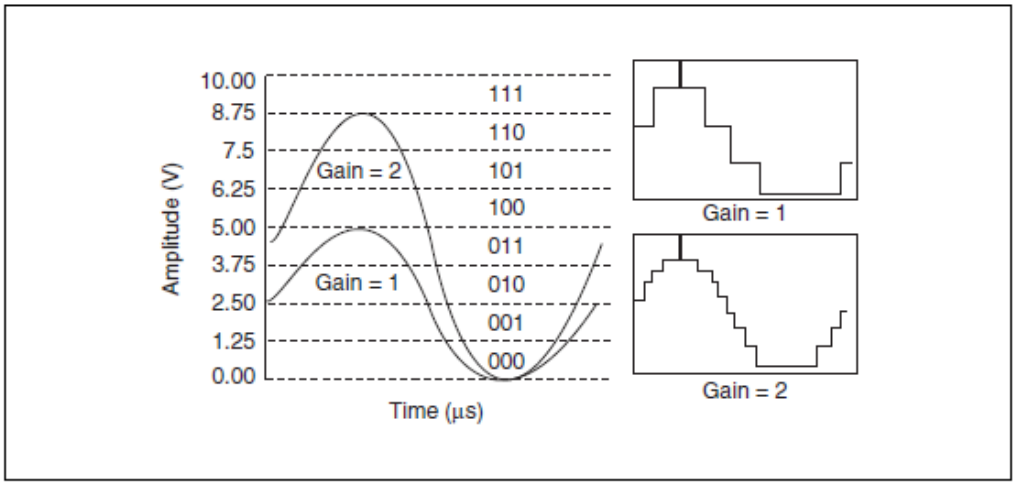


Рисунок B-12. Пример усиления

Диапазон измерений, разрешающая способность и коэффициент усиления DAQ-устройства определяют наименьшее фиксируемое изменение входного напряжения, которое соответствует цене младшего значащего разряда (МЗР) цифрового кода.

Цена МЗР

Цена МЗР — наименьшее изменение сигнала, которое может зафиксировать измерительная система. Она вычисляется по следующей формуле:

$$C = D \cdot \frac{1}{(2^R)}$$

где D — диапазон входных напряжений, R — разрядность в битах.

Диапазон напряжений на входе устройства получается путем умножения диапазона напряжений на входе АЦП на коэффициент усиления. Если, например, диапазон напряжений на входе АЦП от –10 до +10 В (peak to peak) и коэффициент усиления равен 2, диапазон напряжений на входе устройства составляет от –5 до +5 В (peak to peak), т.е. размах напряжений составляет 10 В.

Чем меньше цена МЗР, тем с более высокой точностью устройство может измерить сигнал. Приведенная формула подтверждает все то, что вы уже узнали о разрешающей способности, диапазоне измерений и коэффициенте усиления:

- Чем выше разрешающая способность, тем меньше цена МЗР, и тем выше точность (меньше погрешность) измерения сигнала
- Чем больше коэффициент усиления, тем меньше цена МЗР, и выше точность (меньше погрешность) измерения сигнала
- Чем шире диапазон измерений, тем больше цена МЗР, и тем ниже точность (больше погрешность) измерения сигнала

Цена МЗР является важным фактором при выборе DAQ-устройства. Например, 12-разрядное DAQ-устройство с диапазоном входных напряжений от 0 до 10 В и единичным усилением фиксирует изменение напряжения 2.4 мВ, в то время как такое же устройства с диапазоном входных напряжений от –10 В до 10 В смогло бы только зафиксировать изменение напряжения 4.9 мВ.

$$C = D \cdot \frac{1}{(2^R)} = 10 \cdot \frac{1}{(2^{12})} = 2.4mV$$

$$C = D \cdot \frac{1}{(2^R)} = 20 \cdot \frac{1}{(2^{12})} = 4.9mV$$

Повышение качества восстановления измеренного сигнала

Наиболее эффективным способом повышения качества восстановления измеренного сигнала является уменьшение цены МЗР и увеличение частоты дискретизации. Чтобы эффективно измерить частоту сигнала, частота дискретизации должна превосходить частоту сигнала как минимум в два раза.

Согласно теореме Найквиста для точного измерения наивысшей компоненты частотного спектра измеряемого сигнала частота дискретизации должна быть не менее, чем вдвое больше этой компоненты. Другими словами, высокочастотная область спектра накладывается на ту полосу частот, в которой мы хотим проводить измерения.

Теорема Найквиста записывается следующим образом в виде формулы:

$$f_{sampling} > 2 \cdot f_{signal}$$

где $f_{sampling}$ — частота дискретизации, а f_{signal} — наивысшая компонента частотного спектра измеряемого сигнала.

Чтобы показать, насколько высокой нужно выбирать частоту дискретизации, на рис. B-13 показано, что можно получить при различных частотах дискретизации сигнала. В примере A синусоида частотой *f* дискретизируется с той же частотой – восстановленный по таким отсчетам сигнал выглядит, как уровень постоянного напряжения. Если увеличить частоту сбора данных до *2f*, оцифрованный сигнал будет содержать равное с исходным сигналом число периодов, т.е. частота будет определена корректно, однако форма дискретизированного сигнала будет треугольной, как показано в примере B. В примере C частота дискретизации составляет *4f/3*. Частота Найквиста в этом случае равна *(4f/3)/2 = 2f/3*. Поскольку в этом случае частота *f* больше частоты Найквиста, дискретизированный сигнал воспроизводится искаженным и по частоте, и по форме. Путем увеличения частоты дискретизации удается повысить точность воспроизведения формы сигнала. Однако, имеющиеся в наличии аппаратные средства, как правило, накладывают ограничения на частоту дискретизации.

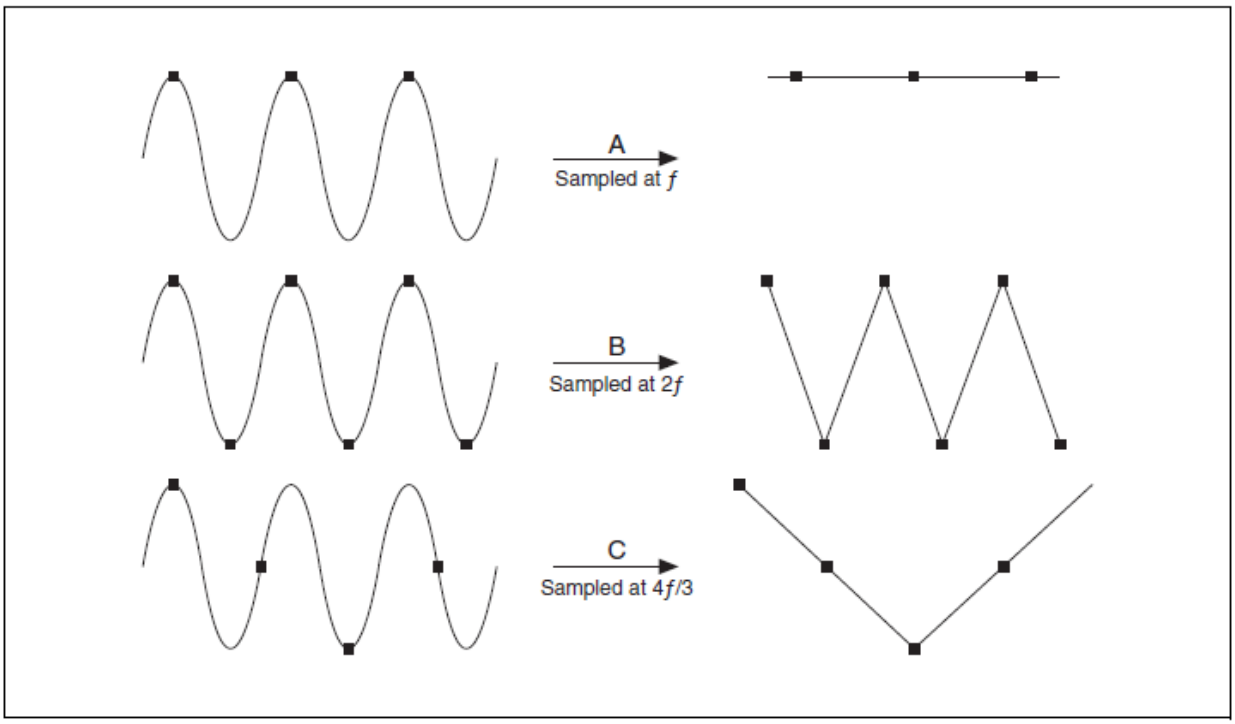


Рисунок B-13. Влияние частоты дискретизации

Уменьшение уровня помехи

Для уменьшения уровня помехи следует руководствоваться следующими рекомендациями:

- Применять экранированные кабели или кабели типа «витая пара».
- Делать как можно короче подводящие провода, чтобы на них было как можно меньше наводок.
- Прокладывать сигнальные проводники подальше от кабелей питания переменного тока и мониторов, чтобы уменьшить влияние наводки 50 или 60 Гц.
- Увеличивать отношение сигнал/помеха, усиливая сигнал поближе к источнику.
- Измерять сигнал с более высокой, чем требуется, частотой дискретизации, а затем усреднять полученные данные, поскольку в этом случае уровень помехи стремится к нулю.

Тест для самоконтроля

1. Вычислите цену МЗР для сигнала, измеренного с помощью 16-разрядного DAQ-устройства, у которого предел измерений равен 5 В.

2. Измеряется треугольный сигнал, частота которого равна 1100 Гц. Какую следует использовать частоту дискретизации, чтобы наилучшим образом восстановить форму этого сигнала?

   a. 1 кГц

   b. 10 кГц

   c. 100 кГц

   d. 1000 кГц

3. Измеряется треугольный сигнал, частота которого равна 1100 Гц. Какую минимальную из приведенных ниже частот дискретизации следует использовать, чтобы достоверно измерить основную компоненту частотного спектра сигнала?

   a. 1 кГц

   b. 10 кГц

   c. 100 кГц

   d. 1000 кГц

Тест для самоконтроля: ответы

1. Вычислите цену МЗР для сигнала, измеренного с помощью 16-разрядного DAQ-устройства, у которого предел измерений равен 5 В.

$$C = D \cdot \frac{1}{(2^R)} = \left(5 \cdot \frac{1}{(2^{16})}\right) = 76.29\mu V$$

2. Измеряется треугольный сигнал, частота которого равна 1100 Гц. Какую следует использовать частоту дискретизации, чтобы наилучшим образом восстановить форму этого сигнала?

   a. 1 кГц

   b. 10 кГц

   c. 100 кГц

   d. 1000 кГц

3. Измеряется треугольный сигнал, частота которого равна 1100 Гц. Какую минимальную из приведенных ниже частот дискретизации следует использовать, чтобы достоверно измерить основную компоненту частотного спектра сигнала?

   a. 1 кГц

   b. 10 кГц

   c. 100 кГц

   d. 1000 кГц

Заметки

C

Приложение C. CAN-интерфейс

Интерфейсная шина Controller Area Network (далее CAN-интерфейс) представляет собой высокоинтегрированную систему последовательных шин для объединения в сеть интеллектуальных устройств. CAN-шины и CAN-устройства получили распространение в автомобильных и промышленных системах. При работе с устройством, имеющим CAN-интерфейс, можно разрабатывать приложения в LabVIEW, с помощью которых будет осуществляться обмен информацией с CAN сетью.

План занятия

A. История развития CAN-интерфейса

B. Основные сведения о CAN-интерфейсе

C. Конфигурирование каналов

D. CAN API

E. Программирование CAN-интерфейса в LabVIEW (Channel API)

A. История развития CAN-интерфейса

В последние несколько десятилетий в связи с необходимостью внедрения усовершенствований в технологию производства автомобилей электронные системы управления стали чаще применяться для таких функций как тактирование двигателя, антиблокировочная система (АБС) тормозов и зажигание без распределителя.

Изначально электронные устройства в автомобилях соединялись с помощью систем проводников типа «точка-точка». Все большее количество электроники в автомобилях привело к появлению в них громоздких, тяжелых и дорогих монтажных жгутов. Чтобы избавиться от соединений типа «точка-точка», производители автомобилей заменили специализированную проводку внутри-автомобильными сетями, за счет чего уменьшилась стоимость, сложность и масса автомобиля. В 1985 году фирмой Bosch был разработана сеть Controller Area Network (CAN) — асинхронная последовательная коммуникационная шина, использующая в качестве среды передачи витую пару проводов, которая стала стандартной автомобильной сетью.

CAN-интерфейс обеспечивает недорогую надежную сеть, которая позволяет устройствам общаться через электронный блок управления (Electronic Control Unit — ECU блок). Согласно протоколу CAN блок ECU оснащен одним CAN-интерфейсом, а не отдельными каналами аналогового ввода для каждого устройства в составе системы. За счет этого уменьшается полная стоимость и масса автомобилей. В состав каждого из устройства сети входит микросхема CAN-контроллера, и поэтому такое устройство считается интеллектуальным. Все передаваемые сообщения «видимы» всеми устройствами в сети. Каждое устройство способно определить, имеет ли сообщение отношение к нему или это сообщение можно проигнорировать.

Поскольку CAN сети все чаще применяются в автомобильной промышленности, был разработан международный стандарт ISO 11898 высокоскоростного CAN-интерфейса. Позднее для кузовной электроники был внедрен низкоскоростной CAN-интерфейс. Наконец, однопроводной CAN-интерфейс был введен для некоторых устройств кузовной электроники и устройств поддержания комфортных условий. Микросхемы CAN-контроллеров выпускаются ведущими производителями полупроводниковой техники как Intel, Motorola и Philips.

До середины 90-х годов прошлого века CAN-протокол являлся основой многих промышленных сетевых протоколов, в том числе DeviceNet и CANOpen.

Применение CAN-интерфейса в автомобилях

К примерам автомобильных устройств с CAN-интерфейсом можно отнести контроллер двигателя, коробку передач, АБС тормозов, фары, стеклоподъемники, усилитель руля, приборную панель и т.д.

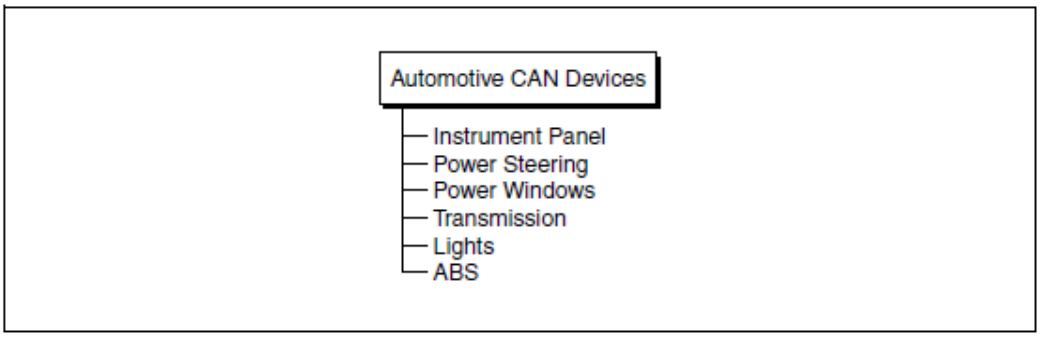


Рисунок C-1. Автомобильные CAN-устройства

Рынки других CAN-устройств

Сравнение требований к автомобильным и промышленным сетям выявило между ними следующие сходства:

- переход от выделенных сигнальных линий
- низкая стоимость
- устойчивость к жестким внешним условиям
- способность работать в реальном времени

Вследствие идентичности свойств CAN-интерфейс стали применять а промышленности, в том числе при производстве текстильного оборудования, в упаковочных машинах, а также в конвейерном оборудовании, таком, как фотоэлектрические датчики и устройства перемещения.

В связи с растущей популярностью CAN-интерфейса в автомобилях и на производстве, этот интерфейс все больше используется в огромном множестве приложений. Применение этого интерфейса в сельскохозяйственном оборудовании, навигационных системах, медицинской аппаратуре, оборудовании для производства полупроводников, авиационной радиоэлектронике, механических станках свидетельствует об универсальности CAN.

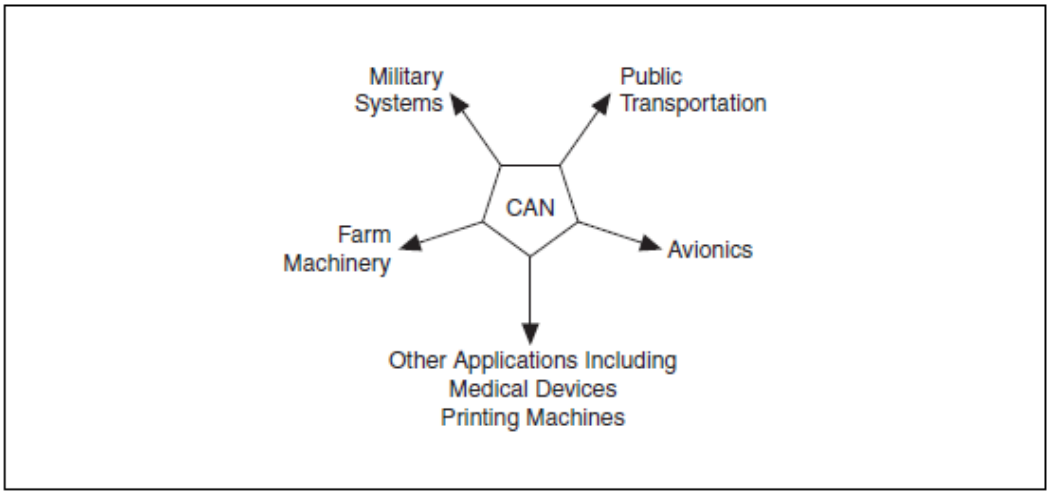


Рисунок C-2. Рынки CAN устройств

B. Основные сведения о CAN-интерфейсе

Преимущества CAN-интерфейса

- Более низкая стоимость в связи с меньшим количеством проводов по сравнению с двухпроводным соединением «точка-точка»
- Высокая помехозащищенность протокола
  - Внутренне присущий детерминизм
  - Отказоустойчивость
  - Надежность — более десяти лет применяется в автомобильной промышленности

Спецификации CAN-протокола

- Кадр для передачи данных — до 8 байт
- Максимальная скорость передачи данных — 1 Мбит/с
- Расстояние, на которое передаются данные на скорости 1 Мбит/с — 40 метров
- Расстояние, на которое передаются данные на скорости 10 кбит/с — 6 км
- Предельное (теоретически) количество узлов на шине — 2032
  - Предельное количество узлов (практически) — около 100 с учетом передатчика
  - Количество узлов, используемое на большинстве шин: от 3 до 10
- Поля арбитража ID (11 бит или 29 бит)
  - Отражает приоритет сообщения

Типы CAN-протоколов

- Высокоскоростной CAN-протокол
  - Скорость передачи данных — до 1 Мбит/с
- Низкоскоростной/отказоустойчивый CAN-протокол
  - Скорость передачи данных — до 125 кбит/с
  - Отказоустойчивый
- Однопроводной CAN-протокол
  - Скорость передачи данных — до 83.3 кбит/с
  - Выдает импульс высокого напряжения, который активизирует устройства, находящиеся в неактивном состоянии

Интерфейсные CAN-устройства производства NI

Компания National Instruments поставляет интерфейсные CAN-устройства четырех типов, которые работают под управлением программного интерфейса NI-CAN API, описываемого в настоящем приложении. Устройства каждого типа выпускаются в нескольких конструктивных исполнениях и в модификациях с одним портом или с двумя портами.

Типы CAN-устройств

Высокоскоростной CAN-интерфейс

- 1 и 2 порта
- Максимальная скорость передачи данных — 1 Мбит/с

Низкоскоростной CAN-интерфейс

- 1 и 2 порта
- Максимальная скорость передачи данных — 125 кбит/с

CAN-интерфейс с программным выбором

- 1 и 2 порта (каждый порт может работать в высокоскоростном, низкоскоростном или однопроводном режиме)

Однопроводной CAN-интерфейс

- 1 и 2 порта
- Максимальная скорость передачи данных — 83.3 кбит/с

Кадр CAN

CAN-устройства посылают данные через CAN-сеть в виде пакетов, называемых кадрами. Типовой кадр состоит из поля арбитража ID, поля данных, кроме того используется кадр удаленного запроса, кадр ошибки и кадр перезагрузки.

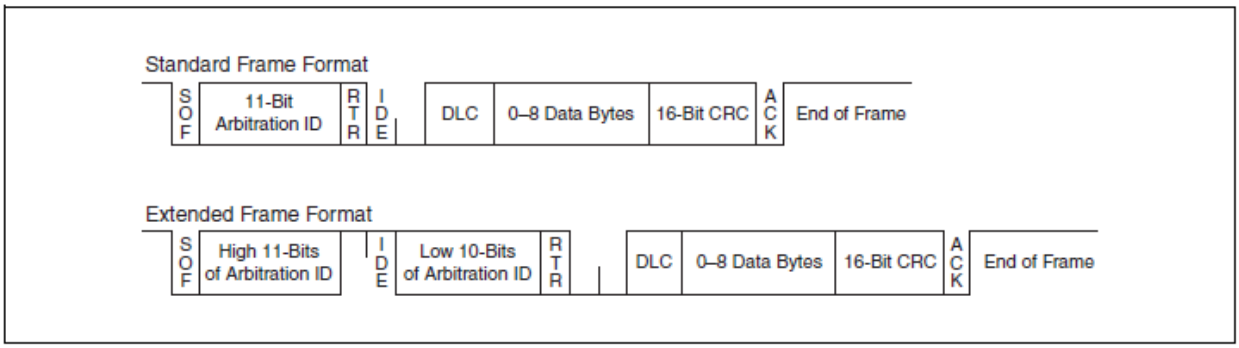


Рисунок C-3. Стандартный и расширенный кадры CAN

Поле арбитража ID

Поле арбитража определяет приоритет сообщений на шине. Если несколько узлов одновременно пытаются передать сообщение на CAN-шину, доступ к ней автоматически получает узел с наивысшим приоритетом (у которого идентификатор приоритета имеет наименьшее значение). Узлы с более низким приоритетом должны ожидать, пока шина не освободится, и только затем делать новую попытку передачи данных. Ожидание длится до того, как будет обнаружен признак конца кадра.

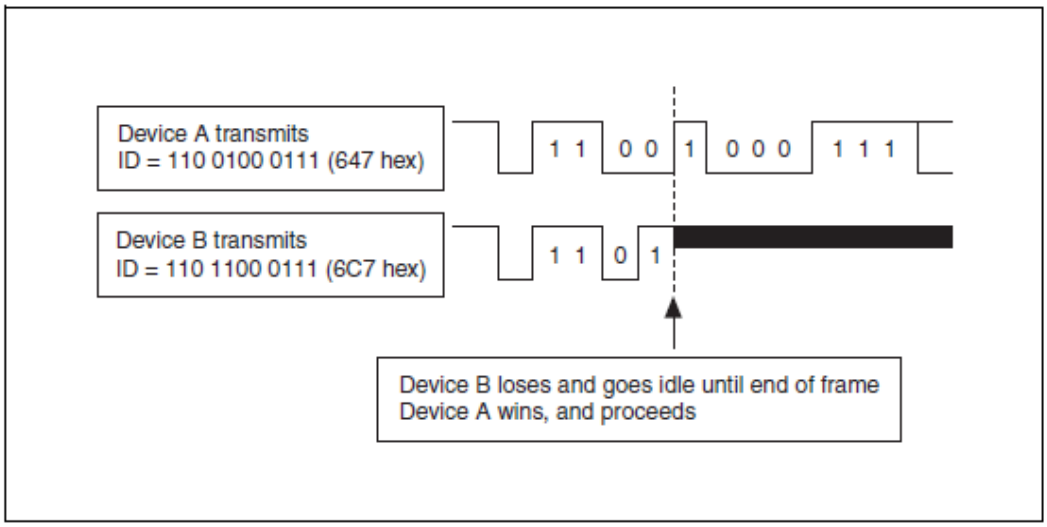


Рисунок C-4. Арбитраж в CAN

Поле данных

В поле данных содержатся фактически передаваемые данные. Согласно спецификациям Bosch Version 2.0 CAN-протокол поддерживает два формата поля данных. Главное различие между ними — по длине поля арбитража ID. Для стандартного формата кадра (имеющего обозначение 2.0A) длина этого поля — 11бит, а для расширенного формата (имеющего обозначение 2.0B) — 29 бит.

Согласно терминологии для CAN-протокола, принятой компанией NI, поле данных, для которого описаны отдельные поля, называют сообщением. Сообщение может состоять из одного или более каналов, которые содержат данные и другую имеющую к ним отношение информацию. На рисунке C-5 приведено сообщение от АБС, которое состоит из двух каналов: Temperature и Torque.

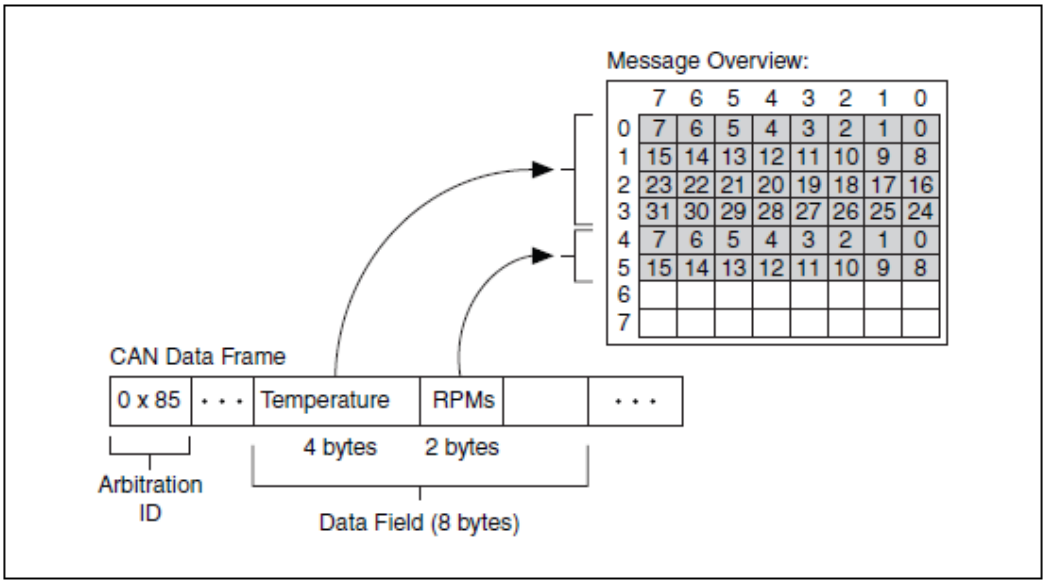


Рисунок C-5. Сообщение CAN из двух каналов

C. Конфигурация канала

С помощью утилиты MAX можно сконфигурировать каналы таким образом, чтобы среда LabVIEW могла интерпретировать поля данных. MAX позволяет загрузить набор каналов из файла базы данных или сконфигурировать каналы вручную.

Базы данных CAN-устройств

Чтобы перетранслировать содержимое поля данных в формат, пригодный для использования, CAN-устройство поставляется вместе с файлом базы данных, в котором описываются каналы, имеющиеся в сообщении. Файл базы данных CAN-устройства (CANdb) представляет собой текстовый файл, который содержит описание каналов. Это дает возможность искать данные в кадре и преобразовывать их в формат с соответствующими единицами измерения. Для каждого канала в базах данных хранятся следующие данные:

```
Channel name
Location (Start bit) and size (number of bits) of
the
Channel within a given Message
Byte Order (Intel/Motorola)
Data type (signed, unsigned, and IEEE float)
Scaling and units string
Range
Default Value
Comment
```

В большинстве компьютерных программ приходится вручную преобразовывать поле данных в полезные данные с помощью информации, которая хранится в файле базы данных. Однако при использовании программного обеспечения компании National Instruments, такие преобразования выполняются автоматически. Можно загрузить конфигурацию канала из файла в утилиту MAX, и затем применять ее в приложении посредством драйвера NI-CAN.

Загрузка каналов из базы данных

Чтобы импортировать конфигурации каналов из файла Vector CANdb в MAX, щелкните правой кнопкой по заголовку **CAN Channels** и выберите команду **Import from CANdb File**. Чтобы выделить несколько каналов, щелкните мышью, удерживая клавишу <Shift>, а затем выберите команду **Import**. Если нужно выделить другой набор каналов, вы можете выделить требуемые каналы и затем снова их импортировать. После этого щелкните мышью по кнопке **Done**, чтобы вернуться в MAX.

Примечание. Вы можете также получить доступ к базе данных CAN-устройства непосредственно из LabVIEW CAN API. Однако загрузка каналов в MAX избавляет от необходимости задавать путь в программах LabVIEW и позволяет считывать данные из каналов и записывать их туда с помощью тестовых панелей утилиты MAX.

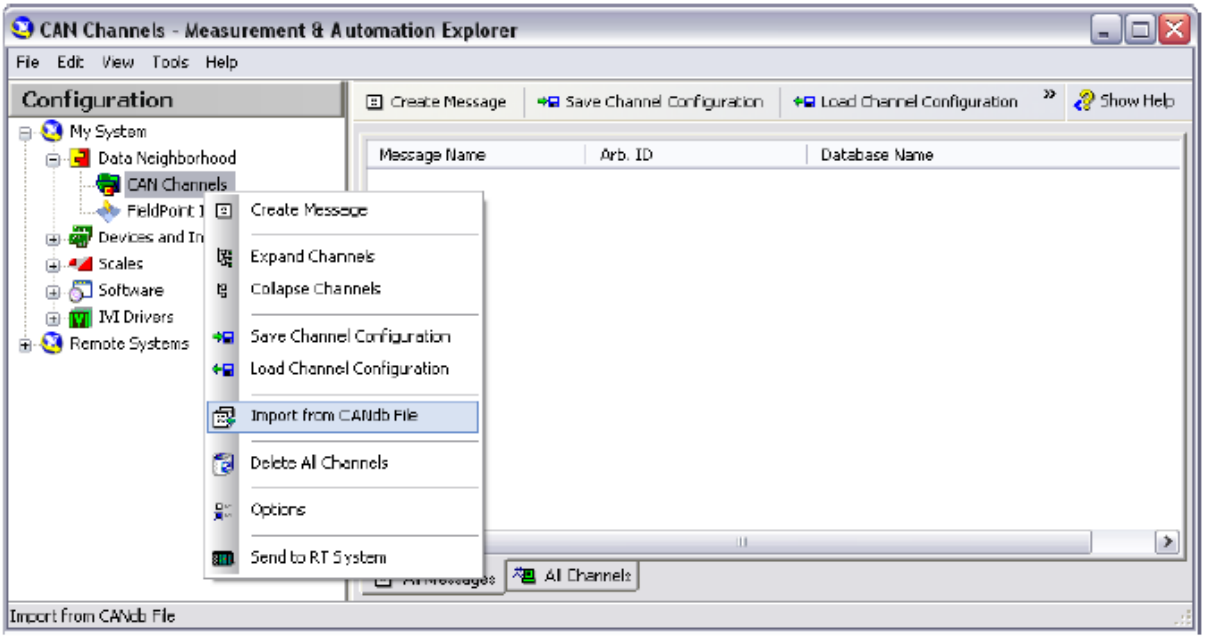


Рисунок C-6. Импорт файла базы данных CAN в MAX

Явное создание канала

Если файл базы данных отсутствует, можно создавать каналы в MAX. Для этого необходимо выполнить следующие действия:

1. Щелкните правой кнопкой по заголовку **CAN Channels** в разделе **Data Neighborhood** и выберите команду **Create Message**.
2. Введите свойства сообщения и щелкните по кнопке **OK**. Созданное сообщение появляется в разделе CAN Channels.
3. Щелкните правой кнопкой по имени сообщения и выберите команду **Create Channel**. На экране появится окно конфигурации, похожее на то, что приведено на рисунке C-7. После того, как вы введете информацию о старт бите и количестве бит соответственно в поля **Start Bit** и **No. of Bits**, матрица заполнится затемненными прямоугольниками, которые укажут на биты, уже использованные в определениях канала для данного сообщения. Синие прямоугольники показывают биты, задаваемые в текущий момент.

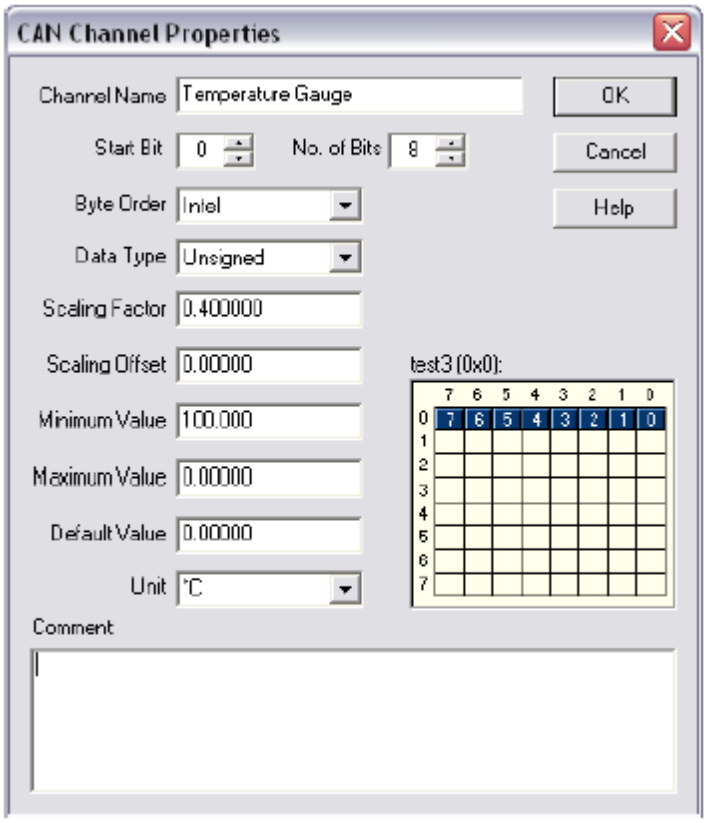


Рисунок C-7. Явное конфигурирование канала в MAX

4. Введите свойства канала и щелкните по кнопке **OK**.
5. Снова щелкните правой кнопкой и выберите команду **Create Channel** для каждого канала, который есть в сообщении.
6. Чтобы сохранить конфигурации каналов в файл, щелкните правой кнопкой по заголовку **CAN Channels** и выберите команду **Save Channel Configuration**.

При сохранении конфигурации каналов создается свой файл базы данных для вашего устройства. Результирующий файл базы данных NI-CAN имеет расширение *.ncd. Доступ к базе данных NI-CAN осуществляется также как и к любой другой базе данных CAN-устройства. Инсталлируя файл базы данных NI-CAN вместе с приложением, это приложение можно распространять среди множества пользователей.

Вы можете протестировать созданные описанным образом каналы с помощью тестовой панели Channel Test Panel.

D. CAN API

Есть два разновидности API, которые можно использовать с оборудованием NI-CAN: Channel и Frame:

- Channel API
  - Программный интерфейс высокого уровня
  - Удобно работать с блоками оборудования
  - Простая синхронизация CAN и DAQ оборудования

    - Несовместимость с драйверами NI-CAN 1.6 и ниже
- Frame API
    - Программный интерфейс более низкого уровня, усовершенствованный API
    - Протокол «команда/ответ»
    - Усовершенствованная синхронизация CAN и DAQ оборудования
    - Совместимость со всеми версиями NI-CAN

Для каждого порта NI-CAN, например порта CAN0, в каждый момент времени можно использовать только один API-интерфейс. Если у вас, например, есть одно приложение, которое работает с Channel API, и еще одно приложение, которое работает с Frame API, порт CAN0 нельзя одновременно использовать с обоими приложениями. Если у вас имеется 2-портовая CAN-карта, то вы можете подсоединить порты CAN0 и CAN1 к одной и той же CAN-сети, и затем использовать порт CAN0 с одним приложением, а порт CAN1 — с другим приложением. Возможен другой вариант, когда порт CAN0 используется в обоих приложениях, но каждое из приложений запускается в разное время. В большинстве случаев необходимо пользоваться интерфейсом Channel API, поскольку это облегчает программирование и сокращает время разработки. Тем не менее, в некоторых случаях без интерфейса Frame API не обойтись, например:

- При отладке приложения, разработанного с использованием NI-CAN версии 1.6 и младше. Интерфейс Frame API совместим с программным кодом ранних версий.
- При необходимости разработки протокола «команда/ответ», согласно которому устройству посылается команда, а оно, в свою очередь, посылает ответ. Как правило, в таких протоколах, применяется пара жестко заданных идентификаторов для каждого устройства, причем идентификатор не определяет смысловое значение байтов данных.
- Если для устройств требуется использовать кадры удаленного запроса. Channel API не поддерживает кадры удаленного запроса, однако у Frame API есть широкие возможности для передачи и приема кадров удаленного запроса.
- При синхронизации передачи данных по CAN-шине и сбора данных с DAQ-карты. Frame API предоставляет доступ к функциям интерфейса RTSI более низкого уровня, чем Channel API, и поэтому лучше подходит для более продвинутой синхронизации.
- При использовании одного из интерфейсных устройств CAN типа NI USB-847x. Изделия семейства USB-CAN не поддерживают объекты CAN или Channel API. Вы все же можете использовать Frame API, однако существуют некоторые ограничения на функции, доступные для изделий USB-CAN. Чтобы ознакомиться со списком этих функций, обратитесь приведенному ниже разделу *Ссылки*.

Примечание. Настоящий курс охватывает только Channel API.

E. Программирование CAN-интерфейса в LabVIEW (Channel API)

Простейшая программа на основе NI-CAN включает в себя инициализацию задачи, старт, чтение или запись данных и очистку задачи. Для доступа к списку имен каналов в базе данных часто используется Get Names VI.

Для связи между функциями служит ссылка на задачу. CAN-задача представляет собой набор CAN-каналов, которые работают синхронно в одном направлении: на чтение или на запись. Задача может включать в себя несколько сообщений, однако все они должны передаваться через один и тот же интерфейс или порт.

CAN Init Start VI

CAN Init Start VI инициализирует задачу для заданного списка каналов и начинает обмен данными. Вход mode определяет, на какой режим настроена задача: чтение или запись.

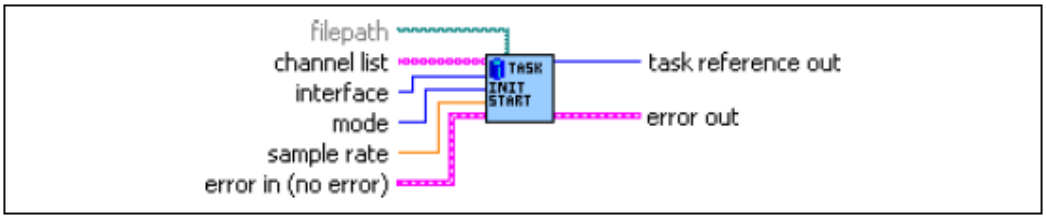


Рисунок C-8. CAN Init Start VI

CAN Get Names VI

CAN Get Names VI извлекает массив имен CAN-каналов или имен сообщений из файла базы данных CAN. Если вход **file path** не подключен, имена каналов берутся из MAX. В противном случае, они извлекаются из указанного файла базы данных. Вход **mode** определяет, к чему осуществляется доступ: к именам каналов или именам сообщений.

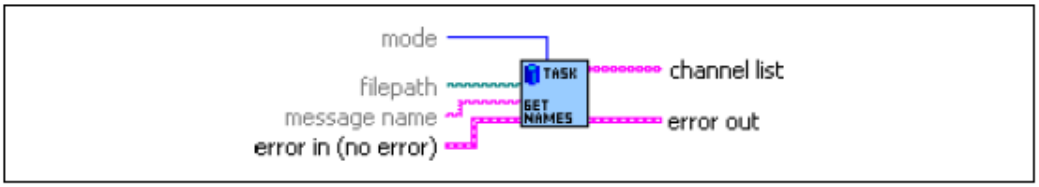


Рисунок C-9. CAN Get Names VI

Существуют три варианта доступа к каналам из приложения:

- задать имя канала, импортированного в MAX
- задать имя файла базы данных и имена каналов без использования MAX
- получить доступ ко всем каналам в файле базы данных с помощью CAN Get Names VI

Чтобы получить доступ к CAN-каналу непосредственно из базы данных CAN, задайте имя канала с указанием в качестве префикса пути к базе данных. Если вы, например, используете канал с именем `Switch0` в файле базы данных `C:\CAN Demo Box.DBC`, подайте строку `C:\CAN Demo Box.DBC::Switch0` на вход CAN Init Start VI, как показано на рисунке C-10. На этом рисунке также показано, как считывать канал, доступный в MAX и как извлекать все каналы из файла базы данных.

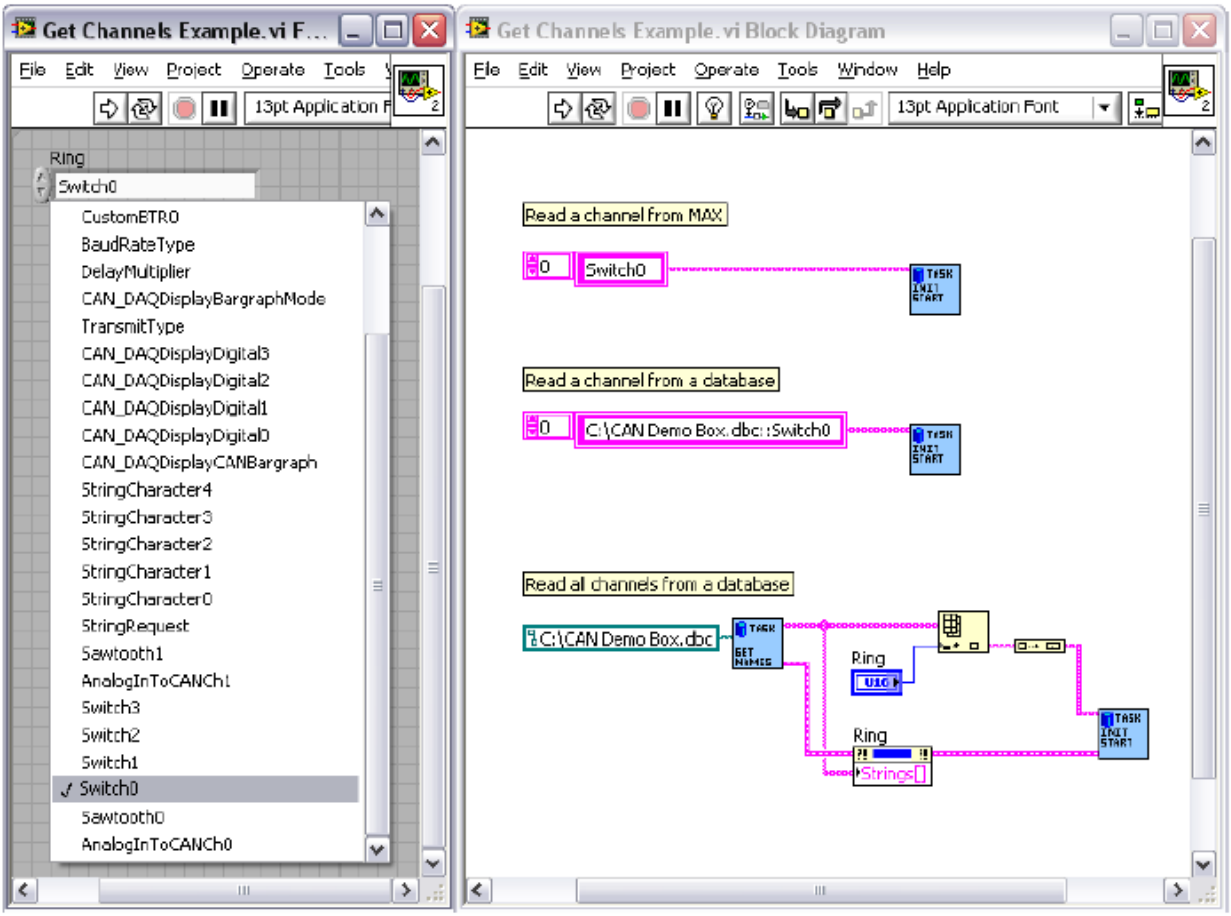


Рисунок C-10. Задание каналов в LabVIEW

CAN Read VI

CAN Read VI читает отсчеты из задачи ввода через CAN-интерфейс. Отсчеты извлекаются из принятых CAN-сообщений. Чтобы выбрать тип данных для принимаемых отсчетов, щелкните правой кнопкой по иконке VI и выберите из контекстного меню команду **Select Type**.

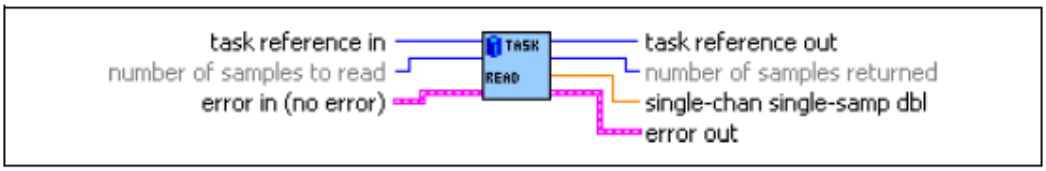


Рисунок C-11. CAN Read VI

CAN Write VI

CAN Write VI записывает отсчеты в задачу вывода через CAN-интерфейс. Эти отсчеты помещаются в передаваемые CAN-сообщения. Чтобы выбрать тип данных для передаваемых отсчетов, щелкните правой кнопкой по иконке VI и выберите из контекстного меню команду **Select Type**.

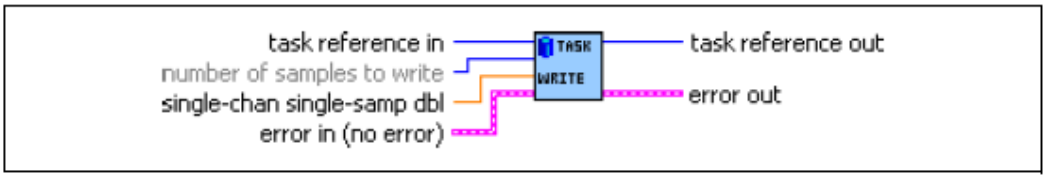


Рисунок C-12. CAN Write VI

CAN Clear VI

CAN Clear VI останавливает обмен данными для задачи и очищает конфигурацию.

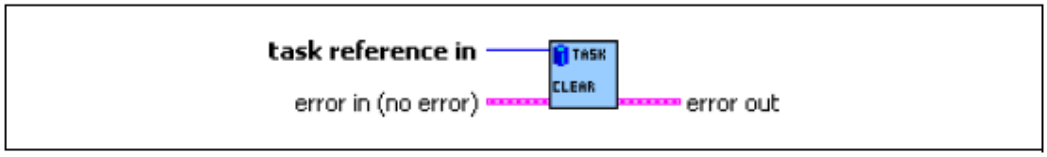


Рисунок C-13. CAN Clear VI

Самопроверка: короткий тест

□

1. Какая скорость передачи данных является максимальной для низкоскоростного или отказоустойчивого CAN-интерфейса?
   a. 1 Мбит/с
   b. 83.3 кбит/с
   c. 256 кбит/с
   d. 125 кбит/с

2. По какой из причин используются каналы NI-CAN?
   a. Они позволяют ассоциировать биты/байты CAN-сообщения/кадра с осмысленными именами и информацией о масштабировании.
   b. Они предоставляют полный доступ к CAN сообщению/кадру.
   c. Они предоставляют доступ к CAN-каналу на физическом уровне.
   d. Они позволяют ассоциировать различные типы CAN-кадров с пользовательскими именами.

3. Какими из способов можно получить доступ к CAN-каналам из вашего приложения?
   a. задать имя канала, импортированного в MAX
   b. задать имя файла базы данных и имена каналов без использования MAX
   c. получить доступ ко всем каналам в файле базы данных с помощью CAN Get Names VI

Самопроверка: ответы

1. Какая скорость передачи данных является максимальной для низкоскоростного или отказоустойчивого CAN-интерфейса?
   a. 1 Мбит/с
   b. 83.3 кбит/с
   c. 256 кбит/с
   d. 125 кбит/с

2. По какой из причин используются каналы NI-CAN?
   a. Они позволяют ассоциировать биты/байты CAN-сообщения/кадра с осмысленными именами и информацией о масштабировании.
   b. Они предоставляют полный доступ к CAN сообщению/кадру.
   c. Они предоставляют доступ к CAN-каналу на физическом уровне.
   d. Они позволяют ассоциировать различные типы CAN-кадров с пользовательскими именами.

3. Какими из способов можно получить доступ к CAN-каналам из вашего приложения?
   a. Задать имя канала, импортированного в MAX
   b. Задать имя файла базы данных и имена каналов без использования MAX
   c. Получить доступ ко всем каналам в файле базы данных с помощью CAN Get Names VI

Заметки

Приложение D. Дополнительная информация и ресурсы

В этом приложении содержится дополнительная информация о технической поддержке National Instruments и ресурсах LabVIEW.

Техническая поддержка National Instruments

Обратитесь к следующим разделам отмеченного наградами веб-сайта National Instruments `ni.com` для получения технической поддержки и профессиональных услуг:

- **Support (Поддержка)** — техническая поддержка по адресу ni.com/support включают следующие разделы:
  - **Self-Help Technical Resources (Технические ресурсы для самостоятельного решения проблем)** — обратитесь за ответами и решениями на сайт `ni.com/support`, где находятся программные драйвера и их обновления, База знаний с возможностью поиска, руководства по продукции NI, мастера по пошаговому поиску устранению неисправностей, тысячи образцов программ, учебных пособий, драйверов измерительных приборов и т.д. Зарегистрированные пользователи получают также доступ к дискуссионным форумам NI по адресу `ni.com/forums`. Специалисты по применению NI гарантируют ответ в режиме он-лайн на каждый заданный вопрос.
  - **Standard Service Program Membership (Членство в стандартной программе обслуживания)** — эта программа позволяет ее участникам обращаться непосредственно к специалистам по применению NI в режиме «тет-а-тет» по телефону и электронной почте для получения технической поддержки, а также обеспечивает эксклюзивный доступ по требованию к учебным модулям через Services Resource Center (Центр ресурсов сервиса). NI предлагает дополнительное членство в течение года после покупки, затем вы можете его продлить.

    Для получения информации о других возможностях технической поддержки в вашем регионе, посетите сайт `ni.com/services` или обратитесь в местный офис по `ni.com/contact`.
- **System Integration (Системная интеграция)** — если вы столкнулись с ограничениями по времени, техническим ресурсам и иными сложностями при работе над проектом, члены National Instruments Alliance Partner (Альянс партнеров NI) могут вам помочь. NI Alliance Partner объединяет системных интеграторов, консультантов и

поставщиков технических средств, которые проводят экспертизу и предоставляют пользователям всестороннюю помощь. Для получения дополнительной информации, свяжитесь с местным офисом NI или посетите сайт ni.com/alliance.

Если вы провели поиск по сайту ni.com и не нашли ответа на свои вопросы, обратитесь в ваш местный офис или в центральный офис NI. Номера телефонов наших офисов во всем мире можно найти в начале данного руководства. Вы можете также посетить раздел Worldwide Offices на сайте ni.com/niglobal для доступа к веб-сайтам филиалов, где имеется обновляемая контактная информация, телефоны службы поддержки, адреса электронной почты и информация о текущих событиях.

Другие учебные курсы National Instruments

National Instruments предлагает несколько учебных курсов для пользователей LabVIEW. Эти курсы продолжают обучение, начатое вами в настоящем курсе и продвигают его на новый уровень. Посетите сайт ni.com/training для приобретения материалов курса или записи на практические курсы с инструктором, проводимые по всему миру.

Сертификация National Instruments

Сертификат NI подтверждает вашу квалификацию в работе с продукцией и технологиями NI. Автоматизированные и измерительные производства, ваши работодатели, клиенты и коллеги распознают ваш сертификат NI как признак знаний и навыков, которые вы получили на практике. Посетите сайт ni.com/training для получения дополнительной информации о программе сертификации NI.

Ресурсы LabVIEW

Ниже описано, как вы можете получить дополнительную информацию о LabVIEW.

Публикации о LabVIEW

Книги LabVIEW

О программировании и применении LabVIEW написаны много книг. Веб-сайт National Instruments содержит список всех книг о LabVIEW с ссылками на места, где их возможно приобрести. Посетите сайт http://zone.ni.com /devzone/cda/tut/p/id/5389 для получения дополнительной информации.

Заметки

Глоссарий

A

automatic scaling (автоматическое масштабирование)	Возможность шкал автоматически изменяться, подстраиваясь под диапазон отображаемых значений. В частности, автомасштабирование шкал графика определяет их максимальное и минимальное значения.

B

block diagram (блок-диаграмма)	Графическое представление программы либо алгоритма. Блок диаграмма состоит из исполняемых узлов (node), изображаемых иконками, и проводников (wire), передающих данные от узла к узлу. Блок диаграмма является исходным кодом VI и отображается в специальном окне – окне блок-диаграмм.
Boolean controls/indicators (логические элементы управления/индикации)	Объекты лицевой панели, предназначенные для управления и отображения логических данных (ИСТИНА или ЛОЖЬ)
broken **Run** button («сломанная» кнопка запуска)	Кнопка, заменяющая кнопку запуска Run при невозможности выполнения VI вследствие ошибок в VI.
broken VI («неисправный» VI)	VI, который не может выполняться вследствие возникновения ошибок. Свидетельством этого является разорванная стрелка на кнопке запуска VI.

C

channel (канал)	1. Физический – терминал или контакт, на который подается аналоговый либо цифровой сигнал при измерении или генерации. Один физический канал может содержать более одного терминала, как в случае дифференциальной конфигурации канала аналогового ввода или цифрового порта, состоящего из восьми линий. Наименование физического канала счетчика (counter) представляет собой исключение, поскольку не является именем терминала, через который счетчик получает или генерирует цифровой сигнал. 2. Виртуальный – совокупность настроек свойств, таких как имя, физический канал, подключения входных терминалов, тип измерения либо генерации и информация о масштабе. Виртуальные каналы NI-DAQmx можно определить либо вне какой-либо задачи (глобальный канал), либо как часть

	конкретной задачи (локальный канал). Настройка виртуальных каналов не является обязательной процедурой при использовании драйвера Traditional NIDAQ (Legacy) и более ранних версий, но она является составной частью любого измерения, выполняемого с помощью драйвера NI-DAQmx. При использовании Traditional NI-DAQ настройка каналов происходит в утилите MAX. В NI-DAQmx вы можете настраивать виртуальные каналы либо в MAX, либо непосредственно в программе, и при этом настройка может являться как частью задачи, так и выполняться отдельно. 3. Канал переключателя (Switch) – представляет любую точку подключения переключателя. Канал может состоять из одного или нескольких сигнальных проводов (обычно один, два или четыре) в зависимости от топологии переключателя. Для канала переключателя нельзя создать виртуальный канал. Каналы переключателя могут использоваться только совместно с функциями и VI из библиотеки NI-DAQmx Switch.
checkbox	В диалоговом окне небольшое квадратное поле (чекбокс), в которое можно ставить либо снимать отметку. Обычно служат для независимого выбора из некоторого набора опций. Возможен выбор более, чем для одного чекбокса.
conditional terminal (терминал условия)	Терминал цикла While, принимающий логическое значение, которое определяет, будет ли цикл выполняться еще раз.
Context Help window (окно контекстной справки)	Окно, отображающее основную информацию об объекте LabVIEW, над которым в данный момент находится курсор мыши. Такими объектами могут быть VI, функции, константы, структуры, палитры, свойства, методы, события или компоненты диалоговых окон.
control (элемент управления)	Объект лицевой панели для ввода данных в VI интерактивно или в SubVI - программно. Примерами таких объектов являются ручки управления, кнопки, лимбы и т.д.
Controls palette (Палитра элементов управления)	Палитра, содержащая элементы управления, индикаторы и элементы оформления лицевой панели.
current VI	VI, чья лицевая панель, блок-диаграмма или редактор иконки (Icon Editor) является активным окном.

D

DAQ	*См.* data acquisition (сбор данных).
DAQ Assistant (помощник по сбору данных)	Графический интерфейс для настройки измерительных задач, каналов и масштабов.

DAQ device (устройство сбора данных)	Устройство для получения или генерации данных. Может содержать несколько каналов и схемы преобразования. DAQ устройствами являются встраиваемые платы, PCMCIA карты, а также DAQPad устройства, подключаемые к компьютеру через порт USB или IEEE 1394 (FireWire). Модули SCXI также считаются DAQ устройствами.
data acquisition (DAQ) (сбор данных)	1. Получение и измерение аналоговых или цифровых электрических сигналов от датчиков, измерительных преобразователей, пробников или зажимов. 2. Генерация аналоговых или цифровых электрических сигналов.
data flow (поток данных)	Принцип программирования, состоящий в том, что исполняемые узлы начнут выполняться только после получения всех необходимых входных данных. По завершении выполнения узлы автоматически генерируют выходные данные. Среда LabVIEW – это система программирования в соответствии с потоком данных. Порядок выполнения VI и функций на блок-диаграмме определяется продвижением данных через узлы.
data type (тип данных)	Формат представления информации. В среде LabVIEW для большинства VI и функций приняты следующие типы данных: числовой (numeric), массив (array), строка (string), логический (Boolean), путь (path), ссылка (refnum), перечисление (enumeration), сигнал (waveform) и кластер (cluster).
default	Значение, принятое по умолчанию. Многие входы VI используют значения по умолчанию в случае, если они не заданы.
device (устройство)	Измерительный прибор либо контроллер, к которому вы можете обращаться как к отдельному объекту, предназначенному для управления или наблюдения за реальными процессами. Устройство часто подключается к хост-компьютеру через некоторую коммуникационную сеть для обмена данными. *См. также* DAQ device и measurement device.
drag	Способ использования курсора мыши для выбора, перетаскивания, копирования или удаления объектов.
driver	Программное обеспечение для управления определенным техническим устройством, таким как DAQ-устройство.
dynamic data type (динамический тип данных)	Тип данных, который используется в экспресс-VI и состоит из выборок сигнала и атрибутов. Атрибуты содержат определенную информацию о сигнале, такую, как имя сигнала либо дату и время получения сигнала. Они определяют вид сигнала на графическом индикаторе осциллограмм или диаграмм.

E

Error list window (окно списка ошибок)	Окно, в котором отображаются ошибки и предупреждения, возникающие в VI, и, в некоторых случаях, рекомендации по устранению ошибок.
error message (сообщение об ошибке)	Сообщение о неправильной работе программного обеспечения или аппаратных средств либо о попытке подачи на вход функции недопустимых данных.
Express VI (экспресс-VI)	VI, предназначенный для решения стандартных измерительных задач. Настраивается с помощью диалогового окна настройки.

F

For Loop (цикл For)	Циклическая структура, выполняющая свою субдиаграмму (тело цикла) заданное количество раз. Эквивалентна следующему коду в текстовых языках программирования: `For i = 0 to n – 1, do....`
front panel (лицевая панель)	Интерактивный интерфейс пользователя VI. Вид лицевой панели имитирует панель управления реальных измерительных приборов, таких, как осциллограф и мультиметр.
function (функция)	Встроенный исполняемый элемент, похожий на оператор, функцию или процедуру в текстовых языках программирования.
functions palette (палитра функций)	Палитра, содержащая VI, функции, структуры блок-диаграммы и константы.

G

General Purpose Interface Bus (Интерфейсная шина общего назначения)	GPIB. Является синонимом HP-IB. Стандартная шина для управления электронными измерительными приборами с помощью компьютера. Известна также под именем шины IEEE 488, поскольку определяется стандартами ANSI/IEEE 488-1978, 488.1-1987 и 488.2-1992.
graph (графический индикатор)	Двумерное отображение одной или нескольких кривых. График получает и отрисовывает данные как единое целое.

I

I/O (ввод/вывод)	Передача данных в/из компьютерной системы через коммуникационные каналы, устройства ввода команд оператором и/или интерфейсы сбора данных и управления.
icon (иконка)	Графическое представление узла на блок-диаграмме.

indicator (индикатор)	Объект лицевой панели для отображения результатов операции, например, графический или светодиодный индикатор.
instrument driver (драйвер прибора)	Набор высокоуровневых функций для управления и взаимодействия с техническими средствами измерительного прибора в системе.
Instrument I/O Assistant	Дополнительное приложение, которое запускается при выборе Instrument I/O Assistant Express VI, предназначенное для коммуникации с приборами, управляемыми с помощью текстовых сообщений, и анализа ответной информации.

L

label (метка)	Текстовый объект, используемый для наименования или описания объектов или областей лицевой панели или блок-диаграммы.
LabVIEW (среда разработки виртуальных измерительных приборов)	Laboratory Virtual Instrument Engineering Workbench – графический язык программирования, в котором для создания программы вместо строк текста используются иконки.
LED	Светодиод.
legend (панель редактирования)	Объект, принадлежащий графическому индикатору осциллограмм или диаграмм, для отображения имени и стиля кривых.

M

MAX	*См.* Measurement & Automation Explorer.
Measurement & Automation Explorer (Проводник по средствам измерений и автоматизации)	Windows-приложение для конфигурирования и диагностирования оборудования компании National Instruments.
measurement device	Устройство сбора данных, такое, как многофункциональная плата ввода-вывода E серии, SCXI модули согласования сигналов и модули коммутаторов.
menu bar (панель меню)	Горизонтальная линейка со списком наименований основных меню приложения. Расположена ниже строки заголовка. Каждое приложение имеет свою линейку меню, однако некоторые меню и команды являются общими для большинства приложений.

N

NI-DAQ	Программные драйверы, поставляемые в комплекте со всеми устройствами NI DAQ и компонентами для согласования сигналов. NIDAQ представляет собой обширную библиотеку VI и функций ANSI C, которые можно вызывать из различных сред разработки приложений, например, LabVIEW, для программирования измерительных устройств NI, модулей согласования сигналов и коммутаторов.
NI-DAQmx	Новая версия драйверов NI-DAQ с новыми VI, функциями и средствами проектирования для управления измерительными устройствами. Преимуществами NIDAQmx над предыдущими версиями NI-DAQ являются наличие помощника DAQ Assistant для конфигурирования каналов и измерительных задач вашего устройства в средах разработки LabVIEW, LabWindows™/ CVI™ и Measurement Studio; возможности имитации большого числа поддерживаемых устройств с целью тестирования и модификации приложений в случае отсутствия реальных устройств; а так же более простой и интуитивно понятный программный интерфейс (API) для создания измерительных приложений с использованием меньшего количества функций и VI по сравнению с более ранними версиями NI-DAQ.
node (узел)	Исполняемый элемент программы. Узлы аналогичны операторам, функциям и подпрограммам в текстовых языках программирования. На блок-диаграмме узлами являются функции, структуры и subVI.
numeric controls and indicators (числовые элементы управления и индикаторы)	Объекты лицевой панели для ввода и отображения числовых данных.

O

object (объект)	Общее название любого элемента лицевой панели или блок-диаграммы, в т.ч. - элементы управления, индикаторы, структуры, узлы, проводники и импортированные рисунки.
Operating tool (инструмент Управление)	Инструмент для ввода данных или операций над ними.

P

palette (палитра)	Отображает объекты или инструменты, которые используются для создания лицевой панели или блок-диаграммы.
plot (кривая)	Графическое представление массива данных на графическом индикаторе осциллограмм или диаграмм.

Positioning tool (инструмент Перемещение)	Инструмент для перемещения и изменения размеров объектов.
project (проект)	Набор файлов LabVIEW и файлов других типов, которые можно использовать для создания спецификаций построения приложений и развертывания или загрузки файлов на целевые устройства.
Project Explorer window (окно обозревателя проекта)	Окно, в котором вы можете создавать и редактировать проекты LabVIEW.
Properties dialog boxes (диалоговые окна свойств)	Диалоговые окна, доступ к которым осуществляется из контекстного меню элемента управления или индикатора. Используются для настройки внешнего вида элемента управления/индикатора.
Pull-down menus (выпадающие меню)	Выпадающие меню, доступные из строки меню. Пункты выпадающего меню обычно являются общими.
PXI	PCI eXtensions for Instrumentation (Расширение шины PCI для измерительных приложений) – модульная компьютерная измерительная платформа.

S

sample (выборка)	Одна точка данных при вводе или выводе аналогового или цифрового сигнала.
scale (шкала)	Элемент графического индикатора осциллограмм, диаграмм и некоторых числовых элементов управления и индикаторов, содержащий набор маркеров или точек с известными интервалами, соответствующими единицам измерения.
shortcut menu (контекстное меню)	Меню, вызываемое щелчком правой кнопкой мыши на объекте. Это меню является специфичным для каждого объекта.
string (строка)	Текстовое представление данных.
structure (структура)	Элемент управления выполнением программы, такой как Flat Sequence structure (плоская структура последовательности), Stacked Sequence structure (стековая структура последовательности), Case structure (структура выбора), For Loop (цикл For), While Loop (цикл While) или Timed Loop (тактируемый цикл).
subpalette (субпалитра)	Палитра, доступ к которой осуществляется из другой палитры, расположенной выше в иерархическом дереве.
subVI (подVI)	VI, используемый на блок-диаграмме другого VI. Аналог подпрограммы.

T

task (задача)	Совокупность из одного или более каналов, параметров синхронизации, запуска и других свойств в NI-DAQmx. Задача представляет собой выполняемую операцию измерения или генерации сигнала.
template VI (шаблон VI)	VI, содержащий наиболее распространенные элементы управления и индикаторы, с помощью которого вы можете создать различные VI с похожей функциональностью. Шаблоны VI можно найти в диалоговом окне New.
terminal (терминал)	Объект или область в узле, на который подаются данные.
tip strip (подсказка)	Небольшое текстовое поле желтого цвета, идентифицирующее имя терминала и, таким образом, облегчающее подключение к нему проводника.
tool (инструмент)	Специальный режим работы курсора для выполнения определенных операций.
toolbar (линейка инструментов)	Линейка, содержащая кнопки управления для запуска и отладки VI.
Traditional NI-DAQ (Legacy)	Устаревшая версия драйвера с API для разработки приложений сбора данных и управления на базе устаревших моделей DAQ устройств National Instruments. Используются только в случае особой необходимости. Дополнительную информацию об использовании драйвера Traditional NI-DAQ (Legacy), а также полный список поддерживаемых устройств, операционных систем, версий программного обеспечения и языков программирования можно найти в документе *NI-DAQ Readme*.

V

VI	*См*. virtual instrument (VI).
virtual instrument (VI) (виртуальный прибор – VI)	Программа, спроектированная в среде LabVIEW, которая моделирует внешний вид и функциональность реального измерительного прибора.

W

waveform (осциллограмма)	Набор значений напряжений, полученных с заданной частотой дискретизации.
waveform chart (развертка осциллограмм)	Графический индикатор для отображения данных, считанных с одинаковым временным интервалом.

While Loop (цикл While)	Циклическая структура, повторяющая участок кода (тело цикла) до тех пор, пока не будет выполнено определенное условие.
wire (проводник)	Тракт для пересылки данных между узлами.
Wiring tool (инструмент Соединение)	Инструмент для задания пути пересылки данных между терминалами.