САМАРСКИЙ
ГОСУДАРСТВЕННЫЙ
АЭРОКОСМИЧЕСКИЙ
УНИВЕРСИТЕТ
имени академика
С.П. КОРОЛЕВА

К.Е. Климентьев

ОСНОВЫ ГРАФИЧЕСКОГО ПРОГРАММИРОВАНИЯ В СРЕДЕ LabVIEW

С А М А Р А 2 0 0 3

УДК 681.3.07

Климентьев К.Е. Основы графического программирования в среде LabVIEW: Учеб. Пособие. Самар. гос. аэрокосм. ун-т. Самара, 2003. 69 с.

ISBN 5-7883-0241-2

Приведена информация об организации и правилах графического программирования в SCADA-системе LabVIEW фирмы National Instruments.

Пособие предназначено для студентов, обучающихся по специальности 220200 - Автоматизированные системы обработки информации и управления, а также может быть полезно студентам смежных специальностей. Разработано на
кафедре информационных систем и технологий.

Табл. 4. Ил. 58. Библиогр.: 10 назв.

Печатается по решению редакционно-издательского совета Самарского государственного аэрокосмического университета имени академика С.П. Королева.

Рецензенты: Д.Л. Головашкин, Л.А. Шерешевский

ISBN 5-7883-0241-2

Оглавление

Введение
1. Первое знакомство с LabVIEW
    1.1. Что такое LabVIEW
    1.2. Виртуальные приборы в LabVIEW
    1.3. Использование LabVIEW
2. Основы разработки виртуальных приборов
    2.1. Разработка модели цифрового термометра
    2.2. Разработка модели температурного монитора
    2.3. Разработка модели анализатора температуры
    2.4. Разработка модели прибора, следящего за поведением температуры
3. Объекты лицевой панели
    3.1. Диаграммы и графики
    3.2. Числовые средства управления и индикаторы
    3.3. Логические средства управления и индикаторы
    3.4. Средства управления и индикаторы прочих типов
        3.4.1. Массивы
        3.4.2. Кластеры
        3.4.3. Строки
        3.4.4. Меню и диалоги
    3.5. Атрибутный узел
4. Объекты блок-схемы
    4.1. Управляющие структуры
        4.1.1. Структура "CASE" ("Выбор")
        4.1.2. Циклы и сдвиговый регистр
        4.1.3. Структура "SEQUENCE" ("Последовательность")
    4.2. Локальные и глобальные переменные
    4.3. Формульный узел
    4.4. ВП файлового ввода-вывода
    4.5. ВП ввода-вывода через интерфейс RS-232
    4.6. Сетевое программирование
5. Расширенные возможности LabVIEW
    5.1. Средства отладки виртуальных приборов
        5.1.1. Поиск и устраенение ошибок
        5.1.2. Встроенный отладчик
    5.2. Узел интерфейса к внешнему коду
    5.3. «Экспресс» - ВП
6. Пример использования LabVIEW
    6.1. Устройство и программирование звуковой карты
    6.2. Системный таймер ПЭВМ
    6.3. Работа с ВП сбора данных
Литература
Приложение А. Сообщения об ошибках
Приложение Б. Библиотечные функции и типы

Памяти В.И. Орищенко

Введение

Одним из главных факторов внедрения средств вычислительной техники во все сферы нашей жизни является увеличение *проблемной ориентированности* этих средств. Например, музыканты-аранжировщики получили возможность общаться с компьютером не в терминах абстрактных языков программирования, но в привычных им терминах нотной грамоты, бухгалтеры и экономисты - в терминах методики управления финансами предприятия, верстальщики - в терминах печатного дела, и так далее.

В сфере технологий разработки программного обеспечения эта тенденция выразилась в возникновении сред *визуального программирования*. Например, популярная среда Borland/Inprise Delphi позволяет легко и просто “нарисовать” внешний интерфейс создаваемой программы, а описывать в текстовом виде требуется лишь алгоритм ее работы.

Дальнейшим развитием идеи визуального программирования явилась концепция *графического программирования*, предусматривающая “рисование” всей программы целиком.

Наиболее последовательное и удачное воплощение концепций визуального и графического программирования для систем сбора данных и управления сложными техническими объектами можно обнаружить в разработках фирмы National Instruments - пакетах LabWindows/CVI и LabVIEW. Данное пособие представляет собой краткое описание возможностей системы LabVIEW, снабженное большим количеством примеров.

Пособие построено по принципу “от простого к сложному”.

В первом разделе обсуждаются основные принципы строения и функционирования LabVIEW.

Второй раздел представляет собой подробное руководство по самостоятельному созданию простейших LabVIEW-приложений.

Третий и четвертый разделы посвящены более глубокому описанию базовых механизмов, позволяющих создавать внешний пользовательский интерфейс и описывать алгоритм работы LabVIEW-приложений.

В пятом разделе рассматриваются некоторые расширенные возможности LabVIEW, позволяющие создавать, тестировать и отлаживать сложные проекты.

Наконец, шестой раздел иллюстрирует некоторые практические аспекты использования LabVIEW на основе несложного примера программирования внешнего устройства ПЭВМ – звуковой платы.

Автор искренне надеется, что пособие позволит начинающим LabVIEW-программистам получить необходимые им первоначальные знания и овладеть базовыми умениями по избранной теме. Непосредственно же пособие адресовано студентам Самарского аэрокосмического университета имени академика С.П. Королева (СГАУ), изучающим на факультете информатики в рамках специальности 220200 “Автоматизированные системы обработки информации и управления” курсы “Системы реального времени”, “Основы автоматизации эксперимента”, “Программные средства АСНИ” и “Проектирование АСНИ”.

Информационной и методической основой для данного пособия послужил перевод на русский язык ряда томов фирменной документации National Instruments, выполненный в 1996-98 гг. группой сотрудников СГАУ под непосредственным руководством большого энтузиаста LabVIEW, доцента кафедры информационных систем и технологий, ныне покойного В.И. Орищенко, которому принадлежала идея написания подобного пособия и которого автор считает своим учителем.

Также автор считает своим долгом выразить признательность другому своему учителю - заведующему кафедрой информационных технологий СГАУ, заслуженному работнику высшей школы РФ, профессору С.А. Прохорову за поддержку при создании данного пособия.

Автор выражает благодарность Д.С. Лежину за ряд ценных замечаний и предложений, высказанных в процессе формирования концепции пособия.

Кроме того, автор считает своим долгом с благодарностью отметить прямое и косвенное участие в формировании текста пособия студентов СГАУ: О. Барышниковой, Н. Кавардаковой, М. Мартьянова, Р. Папировского и Г. Ревякиной.

1. Первое знакомство с LabVIEW

1.1. Что такое LabVIEW

LabVIEW (Laboratory Virtual Instruments Engineering Workshop) – это система программирования, разработанная фирмой National Instruments (США) и ориентированная на создание приложений в области автоматизации научных исследований, управления производством и промышленными установками и т.п. LabVIEW по своим возможностям приближается к системам программирования общего назначения, например к Delphi. Тем не менее, между ними существует ряд важных различий.

1.Система LabVIEW основана на принципах графического программирования.

2.Система LabVIEW основана на принципах объектно-ориентированного программирования.

3.Система LabVIEW является проблемно-ориентированной; она поддерживает программирование множества действий, специфичных для АСНИ, АСУ ТП и АСУП и реализует концепцию виртуальных приборов.

1.2. Виртуальные приборы в LabVIEW

Каждая программа LabVIEW представляет собой отдельный *виртуальный прибор* (ВП), то-есть - программный аналог некоторого реально существующего или воображаемого устройства, состоящий из двух взаимосвязанных частей.

1. Первая часть – «*лицевая панель*» описывает внешний вид ВП и содержит множество средств ввода информации – так называемых *средств управления*, а также множество средств визуализации информации – так называемых *индикаторов*.

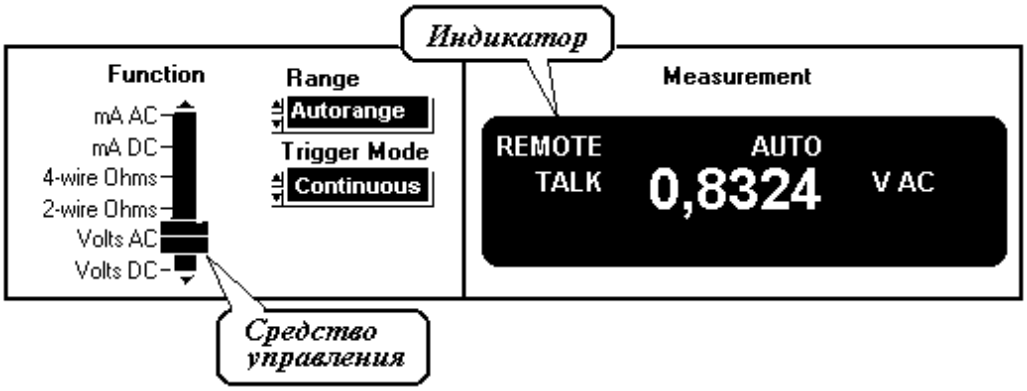


Рисунок 1.1. Лицевая панель ВП – аналога цифрового тестера Fluke 8840A

На рисунке 1.1 к индикаторам относится, например, табло “Measurement”, отображающее разряды числового значения измеряемой величины, а к средствам управления – ползунок “Function”, переключатель диапазона измерений “Range” и переключатель режима измерений “Trigger Mode”.

2. Вторая часть – «*блок-схема*» описывает алгоритм работы ВП.

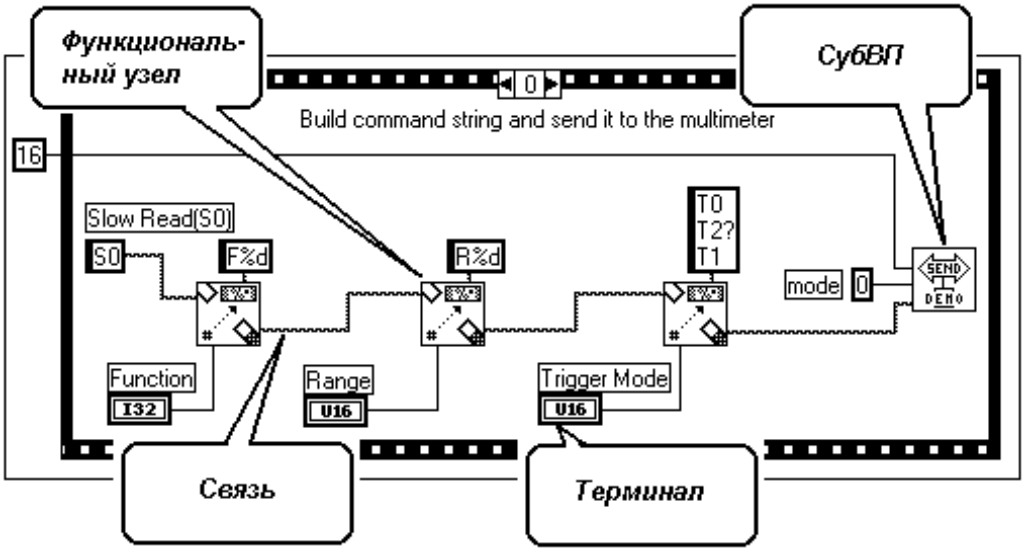


Рисунок 1.2. Блок-схема ВП – аналога цифрового тестера Fluke 8840A

Каждый ВП, в свою очередь, может использовать в качестве составных частей другие ВП, подобно как любая программа, написанная на языке высокого уровня, использует свои подпрограммы. Такие ВП нижнего уровня обычно называются *субВП*. На рисунке 1.2 к субВП относится элемент “Send DEMO” - это ВП, непосредственно реализующий операции по переключению диапазонов, преобразованию сигналов, генерации поразрядного представления результата и т.п.

Также на рисунке можно отметить многочисленные функциональные блоки, играющие роль “задних контактов” для объектов лицевой панели, – это так называемые *терминалы*. Каждому терминалу обязательно соответствует какой-либо индикатор или средство управления, расположенные на лицевой панели.

Важными элементами блок-схемы являются *функциональные узлы* – встроенные субВП, являющиеся частью LabVIEW и выполняющие предопределенные операции над данными. Начиная с версии 7.0 в LabVIEW появились функциональные узлы особого вида – «*экспресс-ВП*». Они представляют собой субВП, реализующие сложные, часто встречающиеся операции, например – настройку и работу с конкретными аппаратными компонентами производства National Instruments.

Данные от терминалов к функциональным узлам и между различными функциональными узлами передаются при помощи *связей*, которые изображены на рисунке разноцветными линиями различной толщины.

Наконец, рамка со скругленными углами, ограничивающая группу соединенных между собой терминалов и функциональных узлов, – это функциональный узел особого вида - *управляющая структура*.

1.3. Использование LabVIEW

Обычному пользователю, как правило, приходится иметь дело с уже готовыми ВП, заранее разработанными другими специалистами. Ему доступна только лицевая панель ВП, в то время как блок-схема ВП скрыта от его глаз. Пользователь снимает какие-либо показания, следит за ходом выполнения какого-нибудь процесса или даже контролирует его ход, используя средства управления передней панели - ручки, тумблеры, кнопки и т.п.

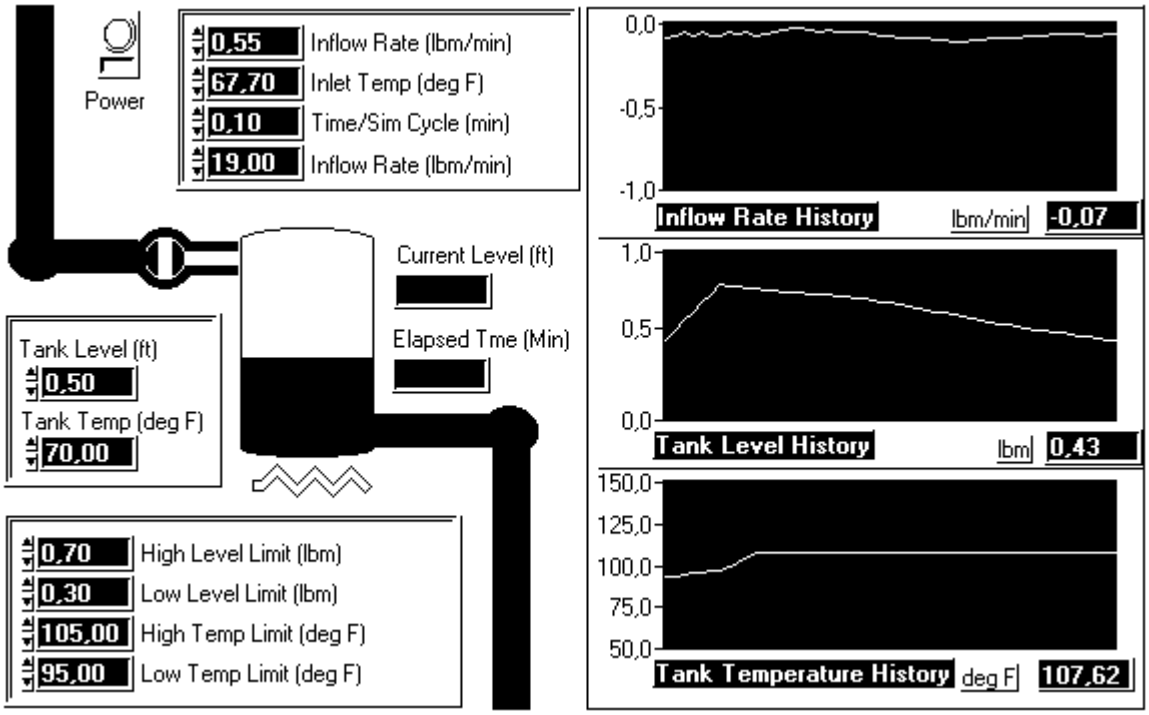


Рисунок 1.3. Пример ВП, предназначенного для управления технологическим процессом

Надо очень четко представлять себе, что ВП - это только модель тех элементов реального прибора или установки, которые гораздо проще и дешевле реализовать в виде программы. Но для того, чтобы ВП можно было использовать как реальный заменитель конкретного осциллографа или распределительного щита, необходимо осуществить связь между объектом физического мира (например, управляемой технологической установкой, см. рисунок 1.3) и программой ВП. Эта связь традиционно осуществляется при помощи специализированных технических средств, - датчиков, аналогово-цифровых и цифро-аналоговых преобразователей, интерфейсов передачи данных и пр., - образующих в совокупности комплекс устройств связи с объектом (УСО). Соответственно, ВП должен иметь выход на программу (драйвер) обслуживания внешнего устройства, являющегося частью УСО (например, на драйвер “измерительной платы”, драйвер контроллера КАМАК и т.п). Обычно в роли такого “связующего звена” выступает функциональный узел блок-схемы ВП или субВП, недекомпозируемый на более мелкие структурные составляющие. Часто он представляет собой фрагмент программного кода, разработанный не средствами LabVIEW, а при помощи языка Ассемблера или Си.

2. Основы разработки виртуальных приборов

В данном разделе на конкретных примерах будут рассмотрены основные понятия и приемы, используемые при разработке собственных ВП средствами LabVIEW for Windows версии 3.x. Полученные в результате знания и умения практически полностью пригодятся при работе с более старшими версиями системы (на момент написания этих строк фирмой National Instruments разработана уже версия 7.1).

Кроме того, для ознакомления с практическими приемами работы в LabVIEW могут быть с успехом использованы демонстрационные версии пакета. Существуют два основных отличия между демонстрационной и полнофункциональной версией:

- с демонстрационными версиями поставляются “укороченные” версии стандартных библиотек ВП;
- демонстрационные версии не позволяют долговременно сохранять результаты работы: все вновь созданные ВП доступны лишь в текущем сеансе работы с LabVIEW, а после выхода из системы они автоматически удаляются.

2.1. Пример 1 – разработка модели цифрового термометра

Предположим, что необходимо измерять температуру среды в экспериментальной камере. Имеется датчик температуры (например, термосопротивление) и устройство аналогово-цифрового преобразования, конвертирующее напряжение на выходных контактах датчика в его числовое значение. Построим модель устройства, воспринимающего и отображающего это значение.

Шаг 1. Запустите установленную на Вашем компьютере версию LabVIEW.

Примечание. В демонстрационном пакете LabVIEW for Windows версий 4.x-6.x, поставляемых на лазерных компакт-дисках, система разработки запускается щелчком по кнопке “Explore LabVIEW for your own applications” (“Изучите работу с LabVIEW на примере собственных приложений”).

Шаг 2. Установите курсор примерно в центре лицевой панели (см. рисунок 2.1). Щелкните правой кнопкой мыши и удерживайте ее до тех пор, пока не появится всплывающие меню “Controls” (“Средства управления”). Вызов всплывающего меню правой кнопкой мыши – наиболее предпочтительный метод создания и изменения объектов LabVIEW.

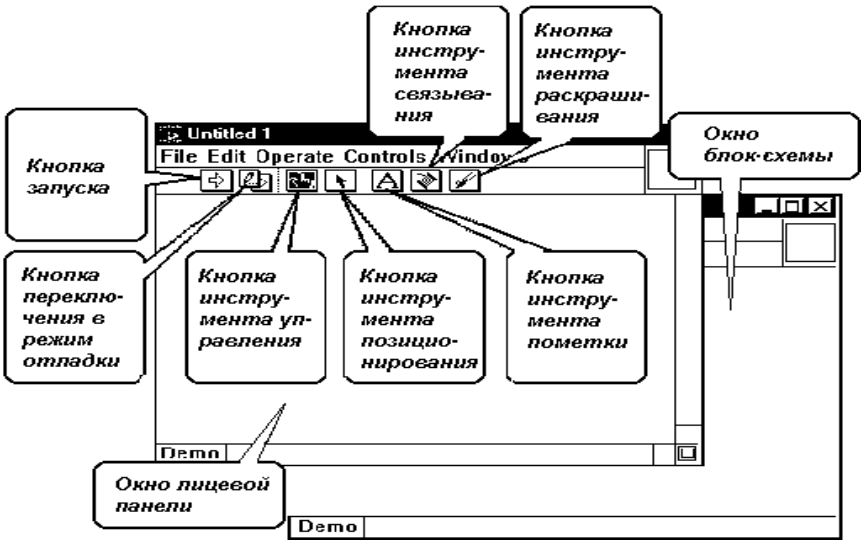


Рисунок 2.1. Вид окон после запуска LabVIEW

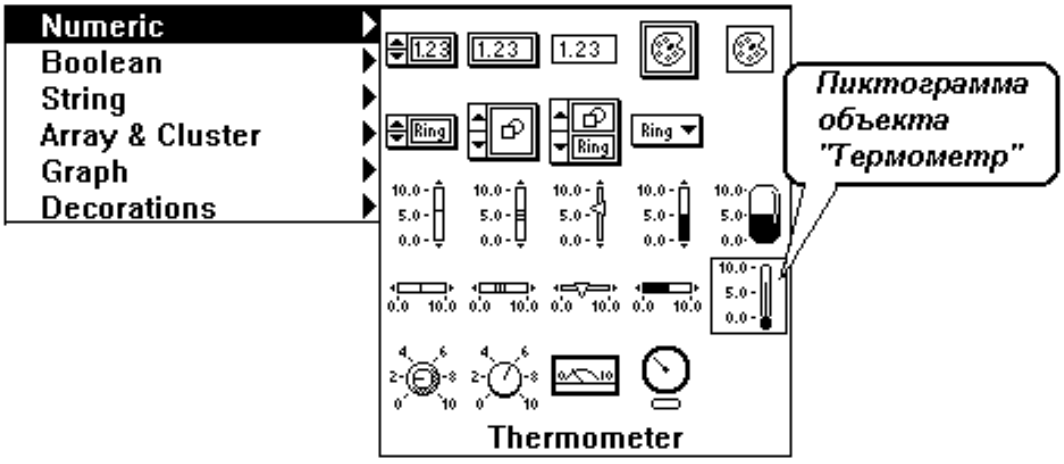


Рисунок 2.2. Выбор объекта “Термометр”

Примечание. В LabVIEW версий 4.x-7.x всплывающее меню содержит не текстовые строки, но пиктограммы со стилизованными изображениями.

Шаг 3. Продолжайте удерживать правую кнопку мыши, перемещая стрелку курсора на элемент “Numeric” (“Числовые”), а также на ту палитру, которая в результате этого появится (см. рисунок 2.2). Когда курсор мыши проходит через каждую пиктограмму, имя соответствующего элемента появляется в нижней части палитры. Отпустите кнопку мыши, когда стрелка окажется на пиктограмме “Thermometer” (“Термометр”).

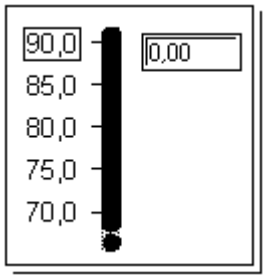
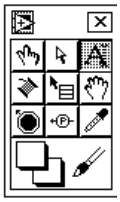


Рисунок 2.3. Пример изменения пределов измерений

Шаг 4. Для изменения пределов измерения на шкале “Термометра”:

- щелчком по пиктограмме инструмента пометки (см. рисунок 2.1) выберите инструмент пометки (“буковку”);
- дважды щелкнув по нижнему пределу измерений на изображении термометра, переведите его в режим редактирования;
- замените строку “0.00” на “70.0” и нажмите клавишу <Enter> на цифровой клавиатуре.

Таким же образом измените “10.0” на “90.0”. LabVIEW автоматически промасштабирует промежуточные значения на шкале, как показано на рисунке 2.3.



Примечание. В LabVIEW версий 4.x-6.x пиктограммы инструментов располагаются не в верхней части окна панели, но в отдельном окне. Если это окно не присутствует на экране, то его можно открыть, выбрав в меню “Windows” (“Окна”) пункт “Show Tools palette” (“Показать палитру инструментов”) или нажав Shift + правая кнопка мыши.

Шаг 5. Выберите пункт “Show Diagram” (“Показать блок-схему”) из меню “Windows”, и окно блок-схемы станет активным. Оно уже содержит для индикатора температуры терминал в виде двойного оранжевого прямоугольника с буквами “DBL”.

Оранжевый цвет зарезервирован для обозначения связей и терминалов, оперирующих с данными вещественного типа. Используются следующие цвета для различных типов данных:

- **синий** - для целого типа;
- **оранжевый** - для вещественного типа;
- **зеленый** - для логического (булевского) типа;
- **сиреневый** - для строкового типа;
- **цвет морской волны** - для типа “имя файла”;
- **темно-серый** - для типа “идентификатор открытого файла”;
- **коричневый** - для комбинированных типов.

Буквы “DBL” уточняют, что терминал оперирует с вещественными данными двойной точности, с данными типа “double”. Также используются следующие аббревиатуры для уточнения числовых типов данных:

- **SGL** - для вещественных данных одинарной точности;
- **DBL** - для вещественных данных двойной точности;
- **EXT** - для вещественных данных повышенной точности;
- **I8** – для целых восьмибитовых данных со знаком;
- **I16** – для целых 16-битовых данных со знаком;
- **I32** – для целых 32-битовых данных со знаком;
- **U8** – для целых 8-битовых данных без знака;
- **U16** - для целых 16-битовых данных без знака;
- **U32** - для целых 32-битовых данных без знака;
- **GSG** - для комплексных данных одинарной точности;
- **GDB** - для комплексных данных двойной точности;
- **GXT** - для комплексных чисел расширенной точности.

Шаг 6. Для перемещения объекта и изменения его размеров выберите инструмент позиционирования (“стрелочку”).

Щелкните по терминалу объекта “Термометр” левой кнопкой мыши, и он окажется внутри штриховой рамки. Удерживая нажатой левую кнопку мыши, вы можете “перетащить” выделенный таким образом объект в любую точку панели.

Примечание. Нажав и удерживая нажатой левую кнопку мыши, Вы можете выделить группу рядом расположенных объектов. Кроме того, одновременно нажав и удерживая не только левую кнопку мыши, но и клавишу <Shift>, Вы можете последовательно выделить инструментом позиционирования несколько отдельных объектов.

Вернитесь на переднюю панель и наведите (не нажимая кнопок мыши) инструмент позиционирования на один из углов изображения “Термометра”. Когда угол “раздвоится”, Вы можете нажать левую кнопку мыши и, удерживая ее, изменить размеры объекта.

Примечание. Изменению размеров поддаются почти все объекты передней панели. На панели блок-схем возможно изменять только размеры рамок управляющих структур и некоторых специфических узлов.

Нажатием клавиши <Del> можно удалить выделенный объект или группу объектов. Важная особенность: терминалы объектов лицевой панели не могут быть удалены из блок-схемы. Они уничтожаются только вместе с порождающими их объектами лицевой панели.

Выделенный объект может быть скопирован в буфер промежуточного хранения и вставлен в любой точке панели (см. соответствующие пункты в меню “Edit”).

Выделенный объект может быть продублирован, если перемещать его инструментом позиционирования, одновременно удерживая нажатой клавишу <Ctrl>.

Шаг 7. Снова перейдите на панель блок-схемы, правой кнопкой мыши вызовите палитру, выберите и разместите на панели следующие объекты:

- из палитры “Tutorial” (“Учебник”) пиктограмму объекта “Voltage Read” (“Источник напряжения”);

Примечание. В демонстрационной версии LabVIEW этот объект может называться “Demo Voltage Read” (“Демонстрационный источник напряжения”), а палитра, в которой хранится пиктограмма объекта, может носить наименование “Getting Started” (“Первые шаги”).

- из палитры “Structs & Constants” (“Структуры и константы”) объект “Numeric constant” (“Числовая константа”);

- из палитры “Arithmetic” (“Арифметика”) объект “Multiply” (“Умножение”).

Выберите инструмент пометки и отредактируйте новые объекты следующим образом:

- внутрь объекта “Числовая константа” поместите число 100.00;
- щелкнув правой кнопкой мыши на терминале “Термометра”, вызовите меню свойств объекта, в нем последовательно выберите “Show” (“Показать”) и “Label” (“Метка”), и инструментом пометки внесите внутрь поля метки наименование объекта (например, латинскими буквами - “Temperature” или русскими - “Температура”).

Обратите внимание: в левой верхней части окна на панели инструментов имеется кнопка запуска приложения LabVIEW. Она одновременно играет роль индикатора: пока внутри нее изображена разрушенная стрелка (см. рисунок 2.4), приложение LabVIEW не готово к работе.

Шаг 8. Выберите инструмент связывания (“катушку”). Переместите “катушку” на объект “Voltage Read”. Объект начнет мигать, показывая, что готов к связыванию. Щелкните левой кнопкой мыши и “закрепите” левый конец линии связи.

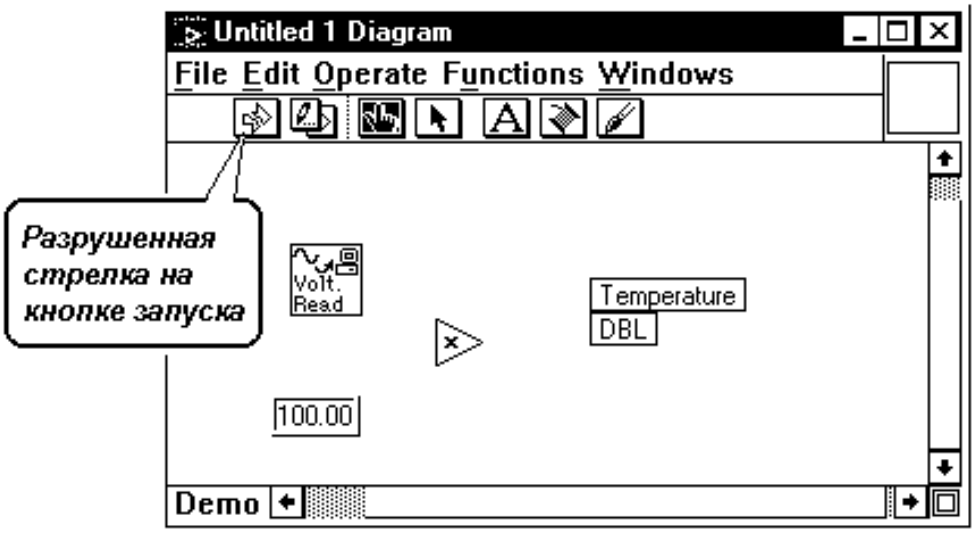


Рисунок 2.4. Отдельные объекты на панели блок-схемы

Перемещайте связь мышью в направлении объекта “Умножение”. Вы можете организовывать на создаваемой линии связи изгибы, останавливаясь и щелкая левой кнопкой мыши. Переместите инструмент связывания на левый верхний угол объекта “Multiply”. Это угол начнет мигать, сигнализируя о том, что вы можете завершить соединение. Нажмите левую кнопку мыши для “закрепления” правого конца линии связи.

Аналогичным образом соедините объект “Числовая константа” с левым нижним углом объекта “Умножение”, а правый угол объекта “Умножение” с терминалом объекта “Температура”.

Обратите внимание: стрелка на кнопке запуска теперь “исправна” - приложение LabVIEW готово к работе.

Шаг 9. Переместитесь на лицевую панель и запустите созданное приложение щелчком по кнопке запуска. Объект “Источник напряжения” сгенерирует одно числовое значение напряжения на выходе некоего датчика (например, термосопротивления), которое будет промасштабировано умножением на 100 и отображено на лицевой панели.

Шаг 10. В правой верхней части лицевой панели располагается квадратное поле, содержащее заготовку пиктограммы создаваемого ВП.

Примечание. В младших версиях демонстрационного пакета LabVIEW это поле может не содержать никаких изображений.

Находясь на лицевой панели, переместите на это поле курсор мыши, нажатием правой кнопки вызовите всплывающее меню и выберите пункт “Edit Icon” (“Редактировать пиктограмму”). На экране появится окно графического редактора, содержащее увеличенное изображение пиктограммы.

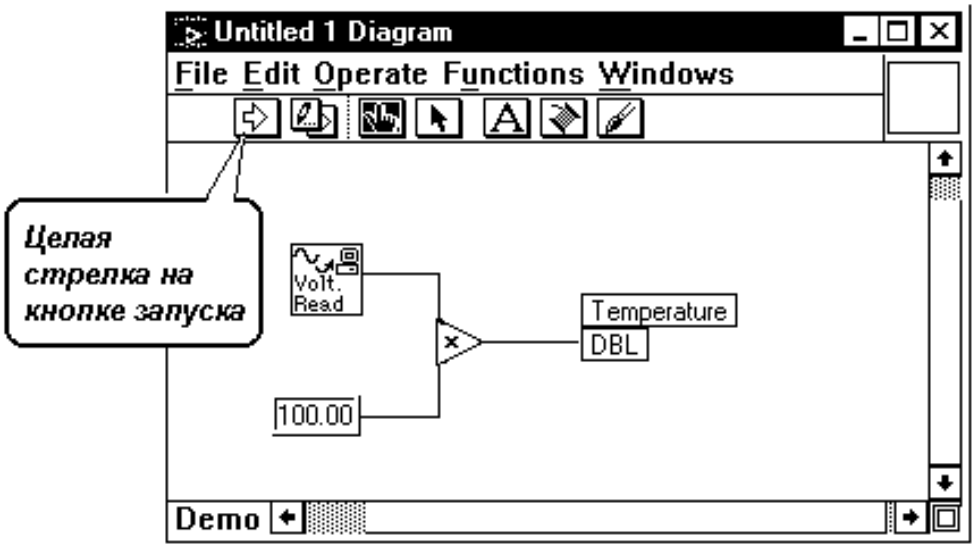


Рисунок 2.5. Связанные объекты на панели блок-схемы

Используя средства, примерно аналогичные используемым в редакторе MS Paint (или MS PaintBrush), нарисуйте свой собственный вариант пиктограммы и нажмите кнопку “OK”.

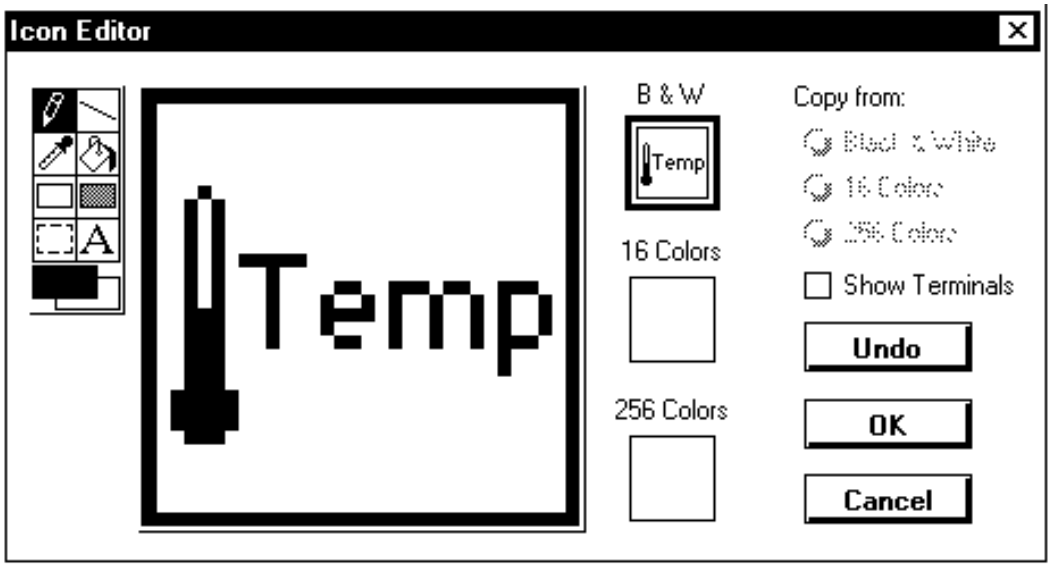


Рисунок 2.6. Редактор пиктограмм

Шаг 11. Построенный нами ВП является моделью цифрового термометра. Другие, более сложные ВП, будут использовать его в качестве своего субВП. По аналогии с процедурами программ, написанных на языках типа Паскаль или Си, необходимо описать входные и выходные параметры построенного субВП.

Повторите начальные действия, использованные Вами на предыдущем шаге, но выберите во всплывающем меню пункт “Show Connector” (“Показать соединитель”). Курсор мыши автоматически переключится в режим инструмента связывания (в режим “катушки”), а поле пиктограммы очистится.

Рисунок 2.7. Палитра образцов

Правой кнопкой мыши вызовите в этом поле всплывающее меню и выберите пункт “Patterns” (“Образцы”). На экране появится палитра, содержащая образцы или заготовки для различных вариантов взаимного расположения и для различных количеств входных и выходных параметров субВП (см. рисунок 2.7). При этом каждая прямоугольная секция образца предназначена для ассоциирования с каким-либо входным или выходным параметром субВП.

Выберите образец, соответствующий случаю единственного параметра (он располагается в левом верхнем углу палитры) и щелкните на нем курсором “катушки”. Выбранный образец переместится в пустое поле.

Примечание. В случае единственного параметра трудно заметить различие между абсолютно пустым полем и полем, заполненным соответствующим образцом. В случае двух, трех и большего количества параметров таких трудностей не возникает.

Поочередно щелкайте курсором “катушки” по объектам лицевой панели и по различным секциям образца (в нашем примере следует щелкнуть по объекту “Термометр” и по единственной секции образца). Секции образца, для которых уже установлено соответствие, окрашиваются в цвета, зависящие от типа передаваемых данных. Важно: если этого не произошло, входы и выходы ВП не определены. Повторите действия Шага 11.

Шаг 12. Сохраните созданный субВП, используя пункт “Save As...” (“Сохранить как...”) меню “File” (“Файл”).

Примечание. В демонстрационных версиях пакета LabVIEW сохранение созданных пользователем ВП и субВП возможно только внутри библиотеки Mywork.LLB.

2.2. Разработка модели температурного монитора

Предположим, что температура в нашей экспериментальной камере изменяется динамически, и необходимо отслеживать и отображать это изменение. Построим соответствующий ВП.

Шаг 1. Создайте лицевую панель для нового ВП. Щелкните правой кнопкой мыши и разместите на лицевой панели (см. рисунок 2.8):

- вертикальный переключатель, выбрав его из палитры “Boolean” (“Логические”) меню “Controls” (“Средства управления”);
- окно “Waveform Chart” (“Диаграмма сигнала”), выбрав его из палитры “Graph” (“Графика”).

При помощи инструмента пометки озаглавьте их соответственно “Power” (“Вкл/Выкл”) и “Temperature History” (“Изменение температуры”).

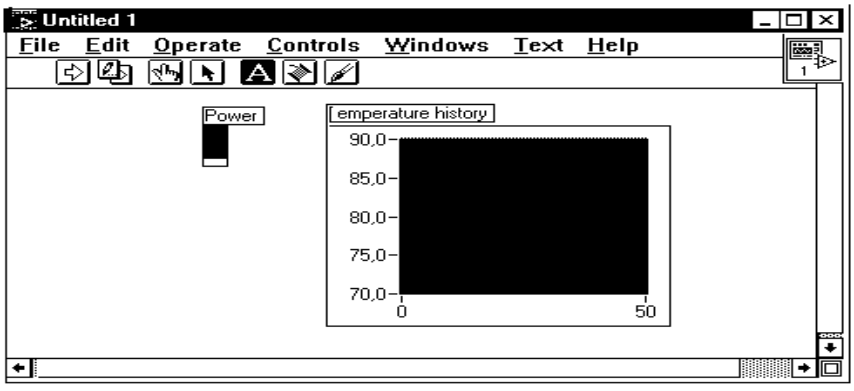


Рисунок 2.8. Создание объектов передней панели

Если Вам не нравится цветовая гамма лицевой панели создаваемого ВП, Вы можете изменить ее при помощи инструмента раскрашивания (“кисточки”). Щелкните правой кнопкой мыши на требуемом объекте лицевой панели, затем, удерживая кнопку нажатой, выберите в цветовой палитре (см. рисунок 2.9) желаемый цвет и отпустите кнопку.

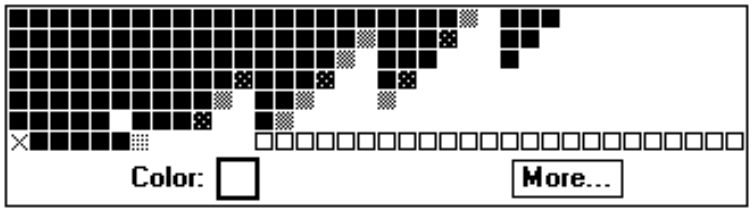


Рисунок 2.9. Цветовая палитра

Шаг 2. Переместитесь на окно блок-схемы и при помощи инструмента позиционирования исправьте местоположение терминалов, если это необходимо. Щелкните правой кнопкой мыши примерно посередине окна, в открывшемся меню выберите пункт “VI...” (“ВП...”) и найдите на диске созданный вами ранее ВП “Цифровой термометр”. Разместите его в окне блок-схемы.

Шаг 3. Измеряемая температура – динамически изменяющийся процесс. Поэтому необходимо, чтобы сбор данных с объекта выполнялся циклически.

В палитре “Structs & Constants” (“Структуры & Константы”) выберите объект “While Loop” (“Цикл While”) и разместите его на свободном участке окна блок-схемы. При помощи инструмента “стрелочка” увеличьте размеры рамки этого объекта. Выделите совместно три ранее созданных объекта и переместите их внутрь рамки. Соедините объекты между собой, как показано на рисунке 2.10.

Фрагмент ВП, размещенный внутри рамки цикла, будет циклически выполняться до тех пор, пока на терминале условия завершения не появится значение “ЛОЖЬ”, сгенерированное при помощи переключателя “Вкл/Выкл”.

Шаг 4. Период итераций организованного на предыдущем шаге цикла зависит лишь от быстродействия компьютера. Необходимо синхронизировать измерения в соответствии с определенным временным расписанием, например, снимать показания раз в секунду.

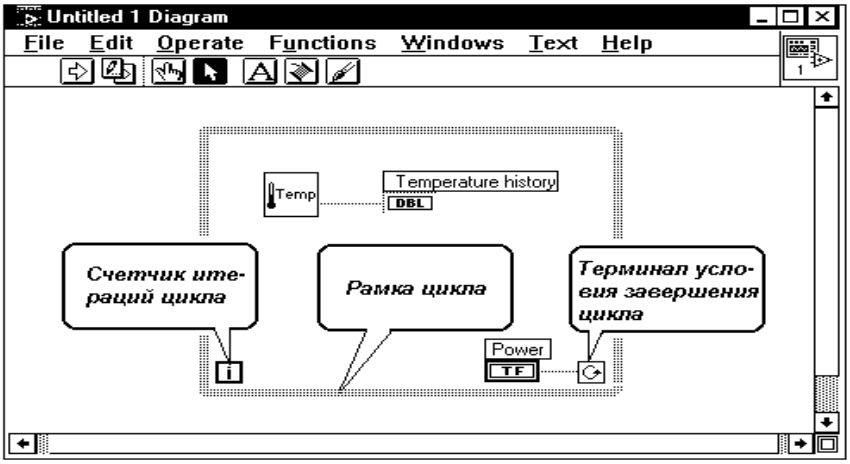


Рисунок 2.10. Внесение объектов внутрь рамки цикла

Выберите в палитре “Structs&Constants” (“Структуры и константы”) числовую константу и установите ее значение в “1000”.

Затем в палитре “Time & Dialog” (“Время и диалог”) выберите объект “Wait Until Next ms Multiple” (“Ожидать, пока не пройдет требуемое количество миллисекунд”). Разместите эти объекты внутри рамки цикла и соедините между собой, как показано на рисунке 2.11.

Шаг 5. Перейдите на лицевую панель ВП, выберите “пальчик”, установите переключатель в положение “включено” и нажмите на кнопку запуска. Прибор начнет выполняться, отображая в окне диаграммы сигнала кривую измеренной температуры. Завершить измерения можно, передвинув переключатель в положение “выключено”.

Примечание. Для принудительного прерывания работы прибора служит также кнопка “Stop”, расположенная на линейке пиктограмм в верхней части окна. Эта кнопка актуальна на этапе разработки и отладки ВП. Существуют режимы работы LabVIEW, в которых эта кнопка недоступна.

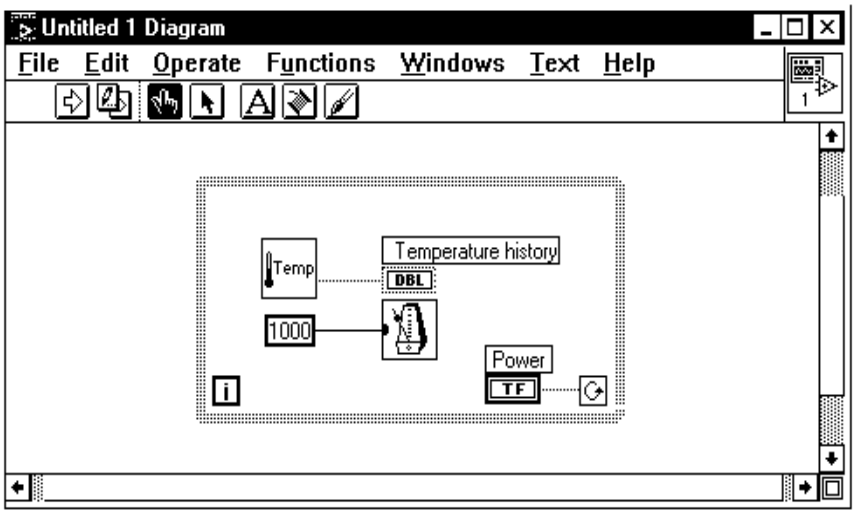


Рисунок 2.11. Блок-схема ВП “Монитор температуры”

Шаг 6. Придумайте и нарисуйте для прибора “Монитор температуры” красивую пиктограмму. Сохраните прибор.

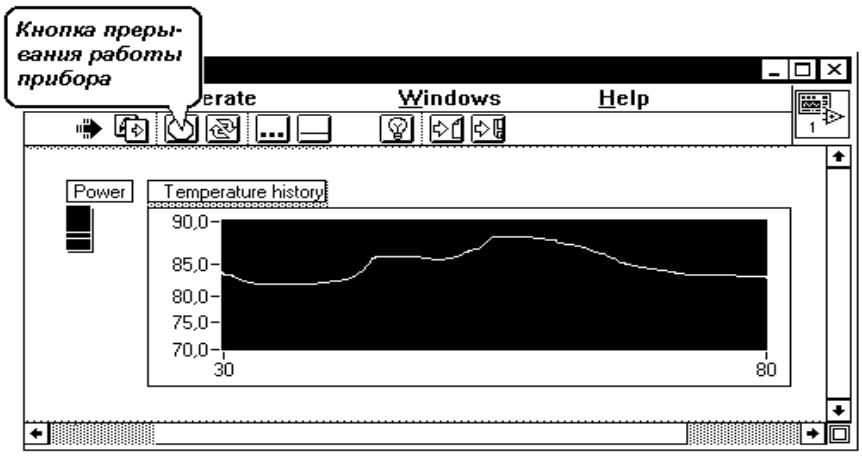


Рисунок 2.12. Передняя панель прибора “Монитор температуры”

2.3. Разработка модели анализатора температуры

Когда измерительная информация в результате воздействия различных влияющих факторов оказывается “зашумлена”, обычно используют различные методы сглаживания выборок. Модифицируем созданный на предыдущем этапе ВП “Монитор температуры” так, чтобы он осуществлял сглаживание измеряемой выборки методом скользящего среднего по трем точкам.

Шаг 1. Откройте ВП “Монитор температуры” и заранее сохраните его под другим именем при помощи пункта “Save As...” (“Сохранить как...”) меню “File” (“Файл”).

Шаг 2. Перейдите на окно блок-схемы и разместите в нем субВП “3-point average” (“Скользящее среднее по 3-м точкам”) из палитры “Tutorial” (“Учебник”).

Примечание. В демонстрационной версии LabVIEW палитра может носить наименование “Getting Started” (“Первые шаги”). Также в некоторых вариантах инсталляции LabVIEW субВП “3-point average” может отсутствовать. Вы можете создать его самостоятельно, как будет объяснено ниже (см. п. 4.1.2).

Шаг 3. Щелкните правой кнопкой мыши, поместив ее курсор точно на линии связи между субВП “Монитор температуры” и терминалом “Изменение температуры”, и выберите в появившемся меню пункт “Insert” (“Вставить”), а дальше поступайте как обычно. В этом случае вставляемый объект будет автоматически присоединен к левому и правому “отрезкам” линии связи. Использовав этот прием, разместите в указанной позиции узел “Bundle” (“Объединение”) из палитры “Array&Cluster” (“Массив и кластер”).

Примечание. В некоторых версиях LabVIEW палитра “Array&Cluster” (“Массив и кластер”) разделена на две: “Array” (“Массив”) и “Cluster” (“Кластер”). В этом случае объект “Bundle” (“Объединение”) содержится во второй из них.

Узел “Bundle” (“Объединение”) предназначен для объединения нескольких линий связи (в том числе и различных типов) в один общий “кабель”. По умолчанию он рассчитан на одну входную связь. Выберите “стрелочку”, наведите ее на нижний правый угол узла, дождитесь его раздвоения и, удерживая нажатой левую кнопку мыши, “добавьте” к узлу второй вход. Теперь узел “Bundle” (“Объединение”) способен объединять в один общий “кабель” две связи.

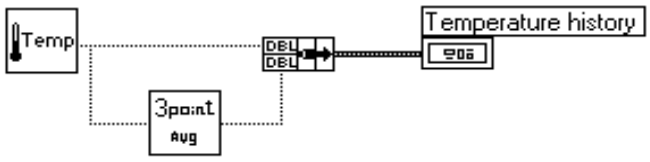


Рисунок 2.13. Модифицированный фрагмент блок-схемы

Шаг 4. Выберите инструмент связывания и модифицируйте фрагмент блок-схемы ВП, как изображено на рисунке 2.13.

Шаг 5. Перейдите на окно лицевой панели. Теперь необходимо сделать так, чтобы на диаграмме одновременно отображались графики двух процессов - исходного и сглаженного. Поместите курсор мыши на объект “Изменение температуры” и нажмите правую кнопку мыши. В появившемся меню выберите сначала “Show” (“Показать”), затем “Legend” (“Легенда”). В правой части диаграммы сигнала появится изображение “легенды”, поясняющей используемые в диаграмме условные обозначения.

Примечание. В некоторых версиях LabVIEW “легенда” (в правой части) отображается вместе с окном диаграммы по умолчанию, также как и палитра свойств (внизу). Для разрабатываемых учебных ВП рекомендуется удалить с лицевой панели изображение палитры свойств.

Выберите “стрелочку” и, используя примерно такую же технику, как и в Шаге 3, добавьте к “легенде” еще одну линию. При помощи “буковки” озаглавьте на “легенде” верхнюю линию “Temperature” (“Температура”), а нижнюю - “3-point average” (“Скользящее среднее по 3-м точкам”). При помощи “кисточки” измените цвета линий на “легенде” (например, на желтый и голубой). Поместите курсор мыши на одну из линий “легенды” и правой кнопкой мыши вызовите всплывающее меню, в котором выберите пункт “Point style” (“Стиль точек”), а вслед за этим - какую-нибудь форму “узелков”. Можете также изменить “Line style” (“Стиль линии”) в этом же всплывающем меню. Имейте в виду, что все выполненные вами назначения отразятся на линиях диаграммы сигнала.

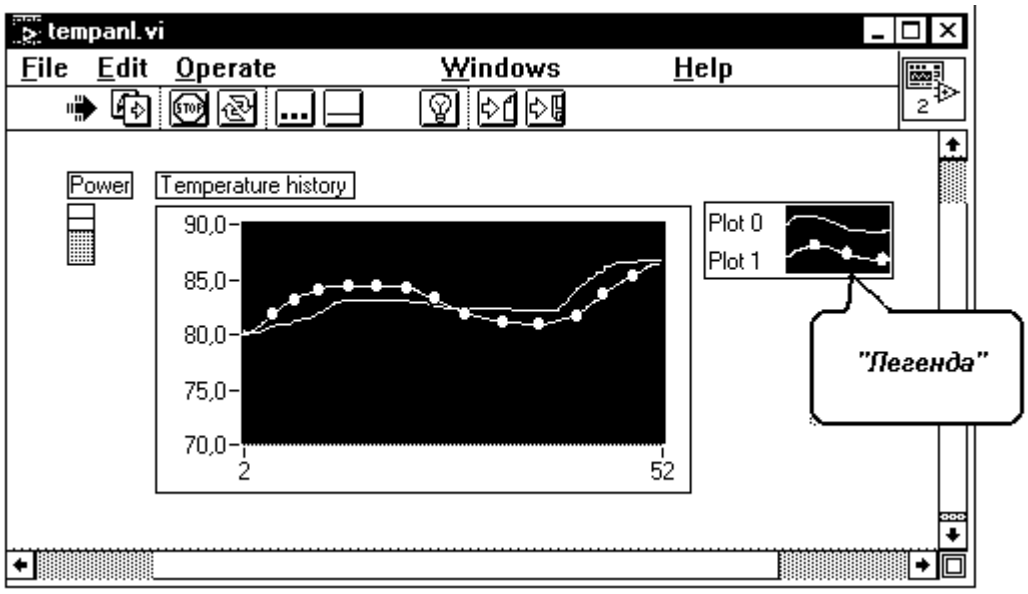


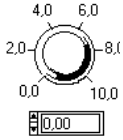
Рисунок 2.14. Лицевая панель “Анализатора температуры”

Шаг 6. Запустите ВП и пронаблюдайте за его работой (см. рис. 2.14). Сохраните ВП.

2.4. Разработка модели прибора, следящего за поведением температуры

Предположим, что необходимо отслеживать поведение измеряемой температуры и предупреждать оператора о превышении наперед заданного порогового значения (“уставки”). Продолжим модифицирование ВП, созданного на предыдущем этапе.

Шаг 1. Откройте ВП “Анализатор температуры” и заранее сохраните его под другим именем при помощи пункта “Save As...” (“Сохранить как...”) меню “File” (“Файл”).



Шаг 2. Создайте на лицевой панели объект типа “Cnob” (“Круглая ручка”), выбрав его из палитры “Controls” (“Средства управления”). Разместите его в окне панели под выключателем питания и озаглавьте “High Limit” (“Верхний предел”). Выберите инструмент пометки и измените пределы значений на шкале ручки так, чтобы они образовали интервал от “70” до “90”.

Выбрав “пальчик” во время работы ВП, Вы можете вращать ручку, изменяя генерируемые объектом значения. Также Вы можете управлять этими значениями, щелкая по маленьким стрелочкам “вверх” и “вниз”, расположенным слева от цифрового поля. Наконец, Вы имеете возможность напрямую внести в цифровое поле желаемое числовое значение.

Шаг 3. Создайте на лицевой панели объект типа “Round LED” (“Круглый светодиод”), выбрав его из палитры “Boolean” (“Логические”). Разместите его в правой части лицевой панели и озаглавьте “Warning” (“Предупреждение”). По умолчанию все логические объекты находятся в состоянии “ЛОЖЬ”. Выберите “кисточку” и покрасьте объект в зеленый цвет. “Пальчиком” переключите объект в положение “ИСТИНА” и покрасьте объект в красный цвет.

Шаг 4. Перейдите на окно блок-схем, выберите инструмент позиционирования и измените размер узла “Bundle” так, чтобы он содержал три входа.

Шаг 5. Создайте объект типа “Greater?” (“Больше?”), выбрав его из палитры “Comparison” (“Сравнение”) и разместите его под узлом “Скользящее среднее по 3-м точкам”. Узел “Больше?” выполняет сравнение двух значений, поданных ему на вход, и возвращает логический результат.

Вообще, в библиотеках LabVIEW содержится большое количество разнообразных функциональных узлов и сервисных субВП. Чтобы получить информацию об их назначении и способах связывания с другими ВП, не обязательно иметь полный комплект справочной документации. Большую помощь может оказать встроенная система подсказок.

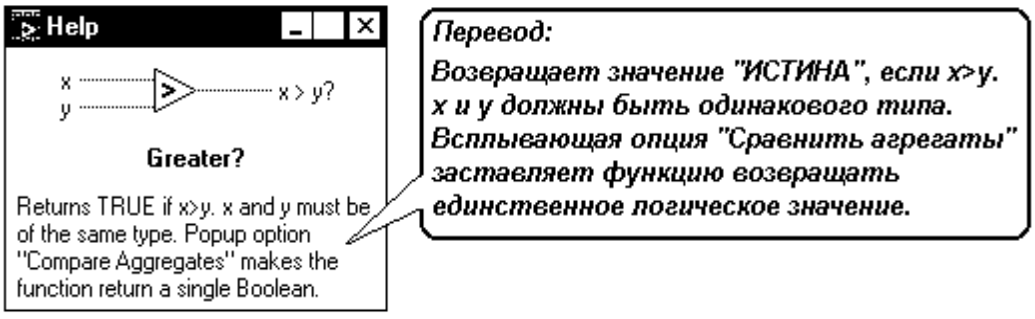


Рисунок 2.15. Окно подсказок

Активируйте окно подсказок при помощи пункта “Show Help” (“Отобразить подсказку”) в системном меню “Help” (“Помощь”). Наведите курсор мыши на интересующий Вас узел блок-схемы (например, на “Больше?”) и задержите его там примерно на секунду. В окне подсказок появится справочная информация об интересующем Вас узле (см. рисунок 2.15).

Примечание. В некоторых версиях LabVIEW активация окна подсказок производится из меню “Windows” (“Окна”).

Шаг 6. Выберите “катушку” и постройте соединения между элементами блок-схемы, как показано на рисунке 2.16.

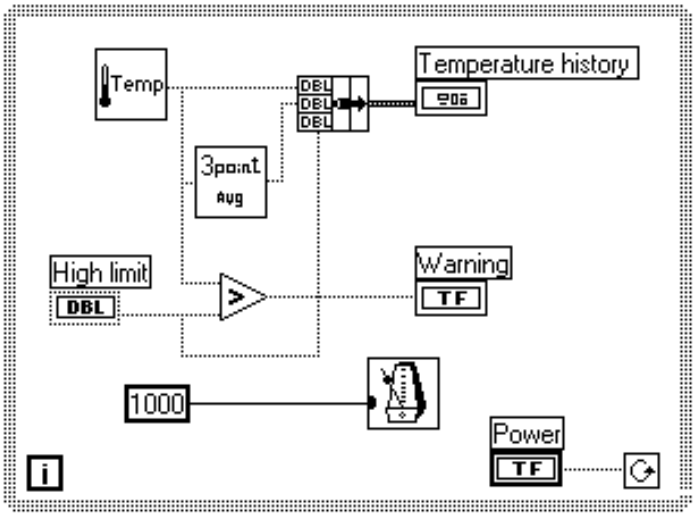


Рисунок 2.16. Блок-схема ВП, следящего за поведением температуры

Во время выполнения сложных соединений Вы можете удалять фрагменты связей, выделив их “стрелочкой”. При этом иногда в окне блок-схемы остаются незаметные глазу оборванные фрагменты связей. ВП, содержащие такие “обрывки”, являются неработоспособными!

Чтобы очистить блок-схему от “плохих” фрагментов, выберите в системном меню “Edit” (“Редактировать”) пункт “Remove Bad Wires” (“Удалить плохие связи”) или нажмите на клавиатуре сочетание Ctrl-B.

Шаг 7. Сохраните ВП. Перейдите на лицевую панель и запустите его. Установите вращающейся ручкой значение предельной температуры, равное “85”. Пронаблюдайте за работой ВП (см. рисунок 2.17).

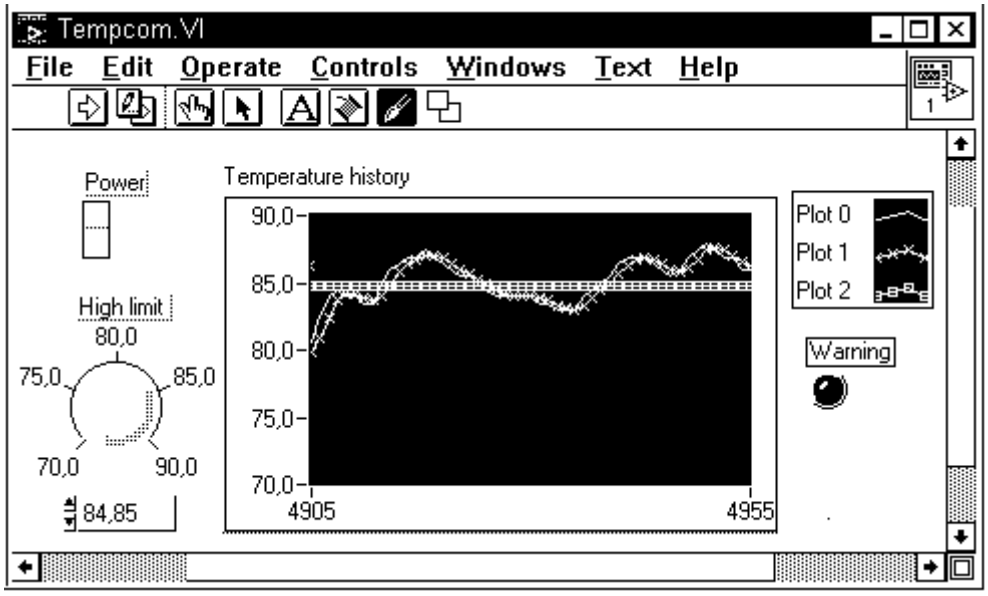


Рисунок 2.17. Лицевая панель ВП, следящего за поведением температуры

3. Объекты лицевой панели

3.1. Диаграммы и графики

В палитре “Graph” (“Графика”) (см. рисунок 3.1) всплывающего меню содержатся несколько различных типов объектов, предназначенных для отображения числовых выборок.

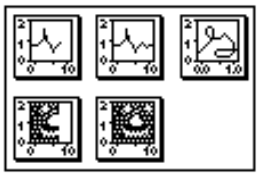


Рисунок 3.1. Палитра “Graph”

1. Объект типа “Waveform Chart” (“Диаграмма сигнала”) отображает эквидистантные числовые последовательности с единичным шагом. Терминал объекта этого типа, располагающийся на блок-схеме, может воспринимать следующие типы данных:

- скаляр (очередной член ряда);
- одномерный массив числовых значений;
- кластер из N массивов (для отображения N линий).

2.Объект типа “Waveform Graph” (“График сигнала”) отображает эквидистантные числовые последовательности, причем абсциссу первого отсчета и шаг по оси абсцисс может задавать пользователь. Терминал объекта этого типа, располагающийся на блок-схеме, может воспринимать следующие типы данных:

- одномерный массив числовых значений (в этом случае предполагается, что шаг по шкале абсцисс равен 1);
- кластер из одномерного массива числовых значений и константы - величины шага по шкале абсцисс;
- кластер из одномерного массива числовых значений и двух констант: величины шага по шкале абсцисс и абсциссы первого отсчета;
- кластер из N вышеописанных массивов или кластеров (для отображения N линий).

3. Объект типа “XY Graph” (“Координатный график”) может отображать графики произвольных (в том числе и неоднозначных) функций. Терминал объекта этого типа, располагающийся на блок-схеме, может воспринимать следующие типы данных:

- кластер из двух массивов: массива значений абсцисс и массива значений ординат;
- кластер из N вышеописанных кластеров (для отображения N линий).

Примеры типичного использования объектов типа “Waveform chart” (“Диаграмма сигнала”), “Waveform graph” (“График сигнала”) и “XY Graph” (“Координатный график”) приведены на рисунке 3.2.

4. Объекты типа “Intensity Chart” (“Диаграмма интенсивности”) и “Intensity Graph” (“График интенсивности”) подобны объектам типов “Waveform Chart” (“Диаграмма сигнала”) и “Waveform Graph” (“График сигнала”), и отличаются от них только тем, что значения ординат отображаются не в виде соединенных линиями точек, но в виде цветов различной интенсивности.

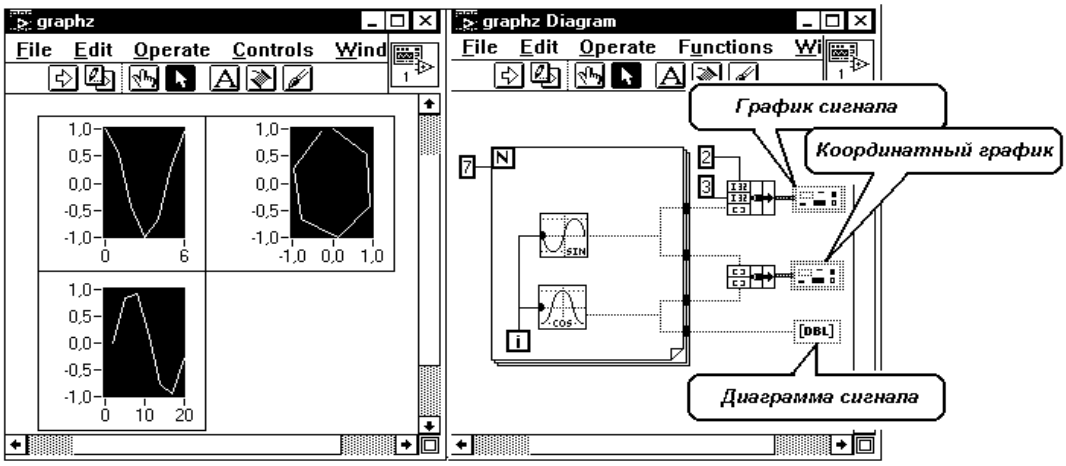


Рисунок 3.2. Использование графиков и диаграмм

Все объекты перечисленных типов обладают общим набором свойств. Доступ к управлению этими свойствами осуществляется при помощи всплывающего меню, которое появляется при нажатии на объекте правой кнопки мыши. Это меню содержит следующие пункты и подпункты (см. рисунок 3.3):

1) Change to Control/Indicator - Переключить объект в режим средства управления или индикатора;

2) Find Terminal - Найти на блок-схеме терминал, соответствующий данному объекту;

3) Show - Отобразить:

- Label - Метку с поясняющим текстом;
- Legend - Легенду (блок условных обозначений);
- Palette - Пульт интерактивного управления свойствами индикатора;
- Digital Display - Цифровой дисплей;
- Scrollbar - Линейку прокрутки;
- X Scale - Шкалу абсцисс;
- Y Scale - Шкалу ординат;

4) Data Operations - Операции с данными;

- Reinitialize to default - Сбросить все установки в “стандартное” состояние;
- Make Current Value Default - Сделать текущие установки “стан-дартными”;
- Cut Data - “Вырезать” данные, сохранив в буфере промежуточного хранения;
- Copy Data - Копировать данные в буфер промежуточного хранения;
- Description - Ввести блок поясняющего текста;
- Clear Chart - Очистить поле графиков;
- AutoScale X - Переключить режим автомасштабирования по оси абсцисс (на статическом изображении);
- AutoScale Y - Переключить режим автомасштабирования по оси ординат (на статическом изображении);
- Update Mode - Выбрать один из режимов обновления изображения на экране (“бегущая линия”, “осциллограф” или “бегущий кадр”);

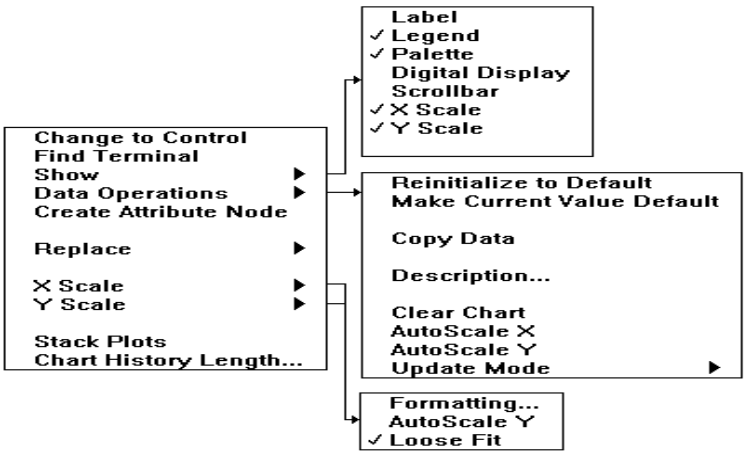


Рисунок 3.3. Структура всплывающих меню объектов типа “Диаграммы и графики”

5) Create Attribute Node - Создать для объекта в окне блок-схемы атрибутный узел;
6) Key Navigation - Переключить управление мышью/клавиатурой;
7) Replace - Заменить любым другим объектом из любой другой палитры;
8) X Scale - Свойства шкалы абсцисс;
- Formatting... - Параметры форматирования изображения;
- AutoScale X - Переключить режим автомасштабирования по оси абс-цисс;
- Loose Fit... - Переключить режим округления пределов шкалы до целых чисел;

9) Y Scale - Свойства шкалы ординат;
- Formatting... - Параметры форматирования изображения;
- AutoScale Y - Переключить режим автомасштабирования по оси ординат;
- Loose Fit... - Переключить режим округления пределов шкалы до целых чисел;

10) Transpose array - Транспонировать массив;
11) Stack Plots - Переключить режим разделения окна индикатора на подокна;
12) Chart History Length - Размер сохраняемой для отображения части выборки.

Примечание. При помощи объектов типа “Атрибутный узел” возможно программное управление свойствами объектов лицевой панели.

3.2. Числовые средства управления и индикаторы

В палитре “Numeric” (“Числовые”) всплывающего меню содержатся средства управления и индикаторы, предназначенные соответственно для ввода и для отображения числовых значений. Они, как правило, являются аналогами элементов лицевых панелей реальных приборов: вариометров, ползунков, стрелочных индикаторов, указателей уровня и т.п. (см. рисунок 3.4.)
Также в палитре “Numeric” (“Числовые”) содержатся цветовые средства управления и индикаторы; они в данном пособии не рассматриваются.

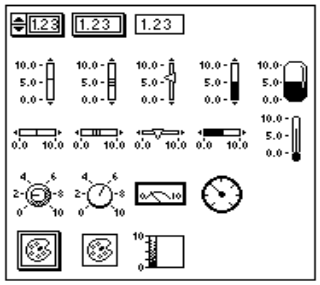
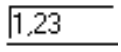
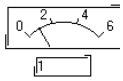


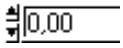
Рисунок 3.4. Палитра “Numeric” (“Числовые”)



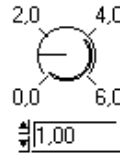
Простейшим средством отображения числовых значений является объект “Digital Indicator” (“Числовой индикатор”). Он состоит только из цифрового дисплея, на котором размещается отображаемое число.



Остальные средства отображения состоят из изображения виртуального индикатора и цифрового дисплея, причем цифровой дисплей предназначен для отображения точного значения числовой величины.



Простейшим средством ввода является объект “Digital Control” (“Числовое средство управления”). Он состоит из поля редактируемого ввода и двух стреловидных кнопок. Вы можете либо напрямую набирать требуемое число в поле ввода при помощи клавиатуры, либо инкрементировать (декрементировать) его, щелкая курсором мыши по стреловидным кнопкам.



Прочие средства ввода чисел состоят из изображения виртуального средства управления и поля редактируемого ввода, сопряженного с парой стреловидных кнопок. Требуемое числовое значение устанавливается либо грубо - при помощи “пальчика”, либо точно - с клавиатуры при помощи “буковки”.

Щелкнув на изображении числового средства управления или индикатора правой кнопкой мыши, можно вызвать всплывающее меню. Его структура в общем соответствует описанной в п. 3.1 структуре меню для диаграмм и графиков. Но имеются и уникальные пункты, рассматриваемые ниже.

1. “Representation” (“Представление”) позволяет изменять тип числовых значений, связанных со средством управления или индикатором. Вид вложенной палитры числовых типов изображен на рисунке 3.5, а перечень соответствующих типов данных приведен в п. 2.1.

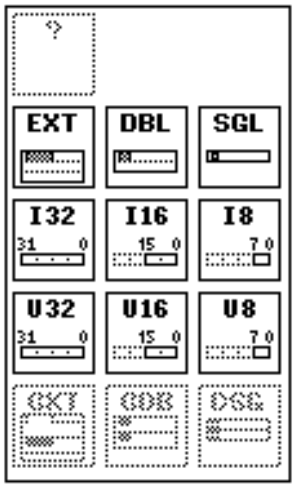


Рисунок 3.5. Палитра числовых типов

2. “Data Range...” (“Пределы числовых значений”) позволяет управлять предельными значениями величин, генерируемых или отображаемых объектом. Вид соответствующей формы ввода приведен на рисунке 3.6.

3. “Format & Precision” (“Формат представления и точность”) позволяет варьировать формой представления отображаемых числовых значений (см. рисунок 3.7).

4. “Add Needle” (“Добавить стрелку”) обеспечивает возможность индикаторам отображать, а средствам управления служить источником для нескольких числовых значений одновременно.

5. Отдельно рассмотрим также пункт меню “Change to Control/Indicator” (“Переключить объект в режим средства управления или индикатора”). Выбрав этот пункт меню, можно, например, объект “Meter” (“Стрелочный индикатор”) заставить работать в качестве средства управления, перемещая стрелку при помощи “пальчика”.

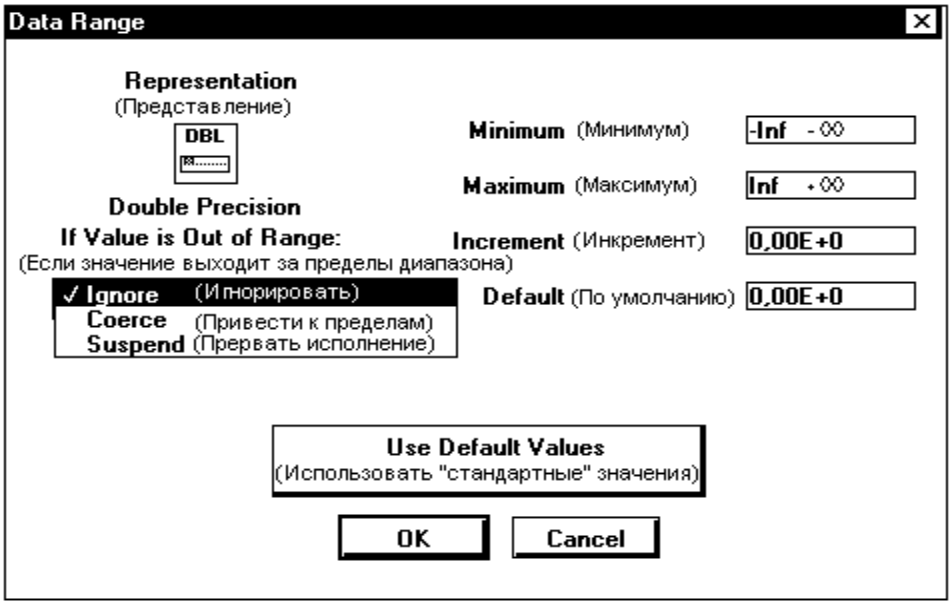


Рисунок 3.6. Форма ввода “Data Range...”

Рисунок 3.7. Форма ввода “Format & Precision”

Примером использования объектов типа “Числовые средства управления и индикаторы” может служить ВП “Имитатор хронометра”, изображенный на рисунке 3.8.

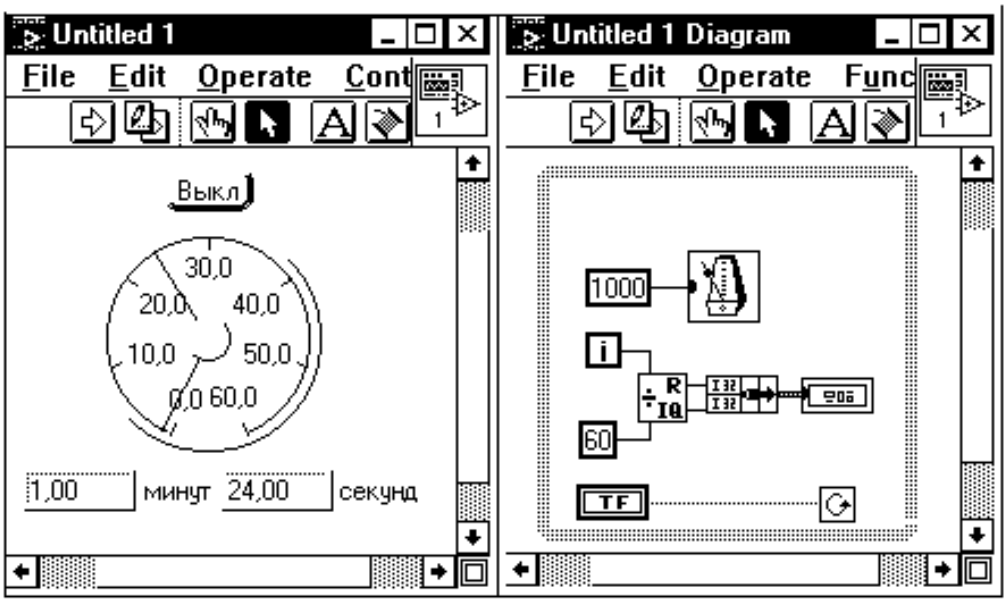


Рисунок 3.8. Виртуальный прибор “Имитатор хронометра”

3.3. Логические средства управления и индикаторы

В палитре “Boolean” (“Логические”) всплывающего меню содержатся средства управления и индикаторы, предназначенные соответственно для ввода и для отображения логических значений “ИСТИНА” или “ЛОЖЬ”.

Они, как правило, являются аналогами элементов лицевых панелей реальных приборов, таких как кнопки, выключатели, тумблеры, сигнальные лампочки, индикаторные светодиоды и т.п. (см. рисунок 3.9)

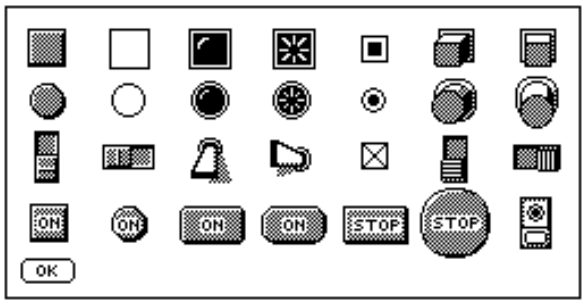


Рисунок 3.9. Палитра “Booleans”

Имеются шесть возможных режимов работы логических средств управления, переключение которых осуществляется в результате выбора пункта “Mechanical Action” (“Механическое действие”) всплывающего меню (см. рисунок 3.10).

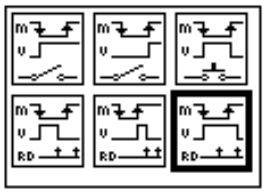


Рисунок 3.10. Вложенная палитра “Механические действия”

1. Switch When Pressed (Переключение при нажатии) инвертирует значение средства управления при каждом нажатии и напоминает работу выключателя света.

2. Switch When Released (Переключение при отпускании) инвертирует значение средства управления в момент отпускания кнопки.

3. Switch Until Released (Переключение до отпускания) инвертирует значение средства управления только после того, как Вы отпускаете кнопку мыши после нажатия.

4. Latch When Pressed (Защелкивание при нажатии) инвертирует значение средства управления при нажатии и сохраняет новое значение, пока ВП не прочитает его.

5. Latch When Released (Защелкивание при отпускании) инвертирует значение средства управления только после того, как Вы отпускаете кнопку мыши, а когда ваш ВП прочитывает значение, средство управления возвращается к своему прежнему состоянию.

6. Latch Until Released (Защелкивание до отпускания) инвертирует значение средства управления, пока Вы нажимаете на него, и сохраняет значение, пока ВП не прочитает это значение или пока Вы не отпустите кнопку мыши.

Как и другие объекты передней панели, логические индикаторы и переключатели обладают набором свойств, изменяемых через всплывающее меню. Изменение механического действия переключателя было рассмотрено выше в данном разделе, а остальные пункты меню в общем традиционны для объектов передней панели (см пп. 3.1 и 3.2). В частности, каждый объект может быть переключен как в режим средства управления, так и в режим индикатора.

Некоторые средства управления кнопочного типа содержат текстовую метку типа “Stop”, “On” или “Off”. Выбрав в качестве операционного инструмента “буковку”, Вы можете изменить текст этой метки.

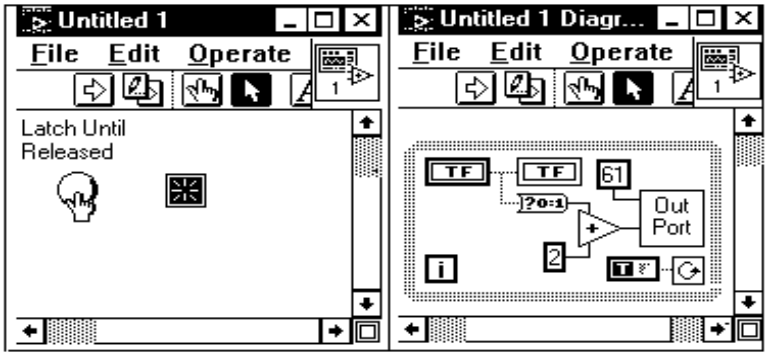


Рисунок 3.11. Имитатор дверного звонка

На рисунке 3.11 изображен ВП “Имитатор дверного звонка”, иллюстрирующий приемы работы с объектами логического типа. Для круглой кнопки установлено механическое действие Latch Until Released (Защелкивается, пока нажато). Константа «61» – 16-ричная. Для организации бесконечного цикла в блок-схеме используется узел “Boolean Constant” (“Логическая константа”). Вы можете переключить этот узел из режима “ИСТИНА” в режим “ЛОЖЬ” при помощи “пальчика”. Узел “Out Port” (“Вывести в порт”) используется для включения бита звучания динамика, а узел “?0:1” предназначен для преобразования логического значения в 0 или 1.

3.4. Средства управления и индикаторы прочих типов

В LabVIEW также предусмотрены средства управления и индикаторы для ввода и отображения данных и других типов, получивших распространение в традиционных языках программирования.

3.4.1. Массивы. Выберите из палитры “Array & Cluster” (“Массив и кластер”) и разместите на лицевой панели пиктограмму объекта типа “Array” (“Массив”). Пока тип элементов массива не определен, объект представляет собой пустую квадратную рамку. Для того, чтобы назначить для объекта конкретный тип данных, щелкните правой кнопкой мыши внутри рамки и выберите из всплывающего меню средство управления или индикатор желаемого типа (см. рисунок 3.12).

Вы можете перемещаться по элементам массива, указав в цифровом поле ввода требуемый индекс, либо щелкая курсором мыши по стреловидным кнопкам этого поля. Основные свойства и приемы работы с отдельными элементами массива не отличаются от рассмотренных в п. 3.2 для числовых средств управления и индикаторов.

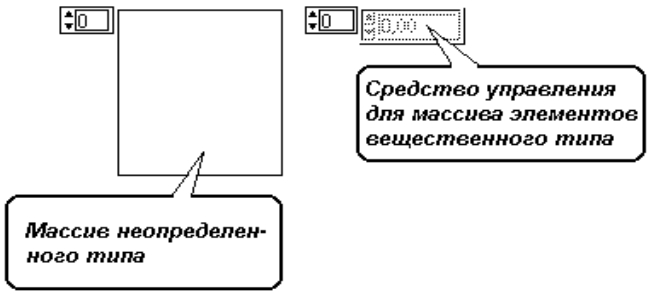


Рисунок 3.12. Этапы определения элементов типа “Массив”

3.4.2. Кластеры. Этот сложный тип представляет собой совокупность элементов различных типов и известен в других языках программирования под наименованиями “структура” (“struct”), “запись” (“record”) или “кортеж”.

Выберите из палитры “Array & Cluster” (“Массив и кластер”) и разместите на лицевой панели пиктограмму объекта типа “Cluster” (“Кластер”). Пока типы элементов кластера не определены, объект представляет собой пустую квадратную рамку. Для того, чтобы “сконструировать” этот сложный тип данных, щелкните правой кнопкой мыши внутри рамки, и выберите из всплывающего меню средство управления или индикатор желаемого типа (рисунок 3.13).

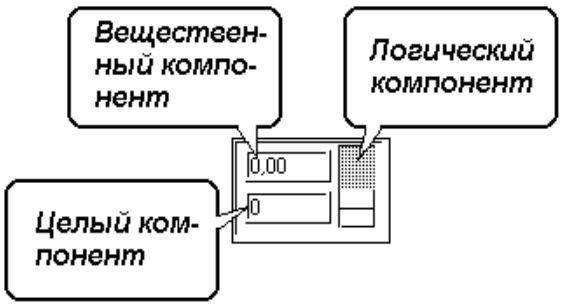


Рисунок 3.13. Индикатор типа “Кластер”

3.4.3. Строки. В палитре “String & Table” присутствуют средства управления и индикаторы, позволяющие вводить с клавиатуры и отображать текстовые строки.

Примечание. Следует отличать строковые средства управления и индикаторы от очень похожих на них внешне объектов передней панели типа “File Path” (“Путь к файлу”). Этот тип используется для задания имени файла при операциях дискового ввода-вывода. Строковый тип для этой цели использоваться не может.

3.4.4. Меню и диалоги. В палитре “List & Ring” (“Список и свиток”) доступны средства управления, предназначенные для организации диалогового взаимодействия с пользователем (см. рисунок 3.14). Все они представляют собой варианты меню, элементами которых могут служить как текстовые строки, так и изображения.

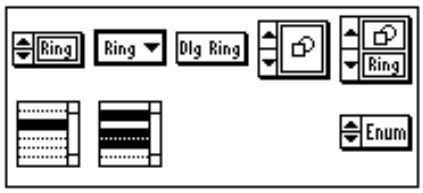


Рисунок 3.14. Палитра “List & Ring”

Принципы работы средств управления этой группы рассмотрим на примере объекта типа “Menu Ring” (“Меню-свиток”). Разместите его на лицевой панели. Он представляет собой “разворачивающийся свиток” пунктов (элементов) меню, с каждым из которых связана текстовая метка, раскрывающая смысл этого пункта. Сразу после создания “свиток” содержит только один пустой элемент. Щелкая правой кнопкой мыши по объекту и выбирая из всплывающего меню “Add Item After” (“Добавить элемент после текущего”), “Add Item Before” (“Вставить элемент перед текущим”) и “Remove Item” (“Удалить элемент”), вы можете отредактировать количество и взаимное расположение пунктов меню. При помощи инструмента “буковка” Вы можете ввести и отредактировать текстовые метки, поставленные в соответствие тому или иному элементу.

После создания объекта типа “меню” в окне блок-схемы появляется его целочисленный терминал, который содержит номер (начиная с 0) выбранного пользователем пункта меню. Во время работы ВП пользователь может выбирать тот или иной пункт меню мышью, при помощи “пальчика”.

3.5. Атрибутный узел

Выше было продемонстрировано, что каждый объект лицевой панели обладает набором свойств, значения которых могут быть изменены вручную при помощи всплывающих меню (см. рис. 3.3, 3.6 и 3.7). Программный доступ к этой возможности осуществляется через *атрибутный узел*.

Примечание. В старших версиях LabVIEW вместо термина «атрибутный узел» (Attribute Node) используется «узел свойств» (Property Node).

Чтобы создать для объекта лицевой панели атрибутный узел, необходимо выбрать во всплывающем меню объекта альтернативу «Создать атрибутный узел» (Create Attribute Node) – см. рис. 3.3. Появившийся список атрибутов объекта может быть увеличен при помощи инструмента «стрелочка» так, чтобы были видны необходимые атрибуты (см. рис. 3.15).

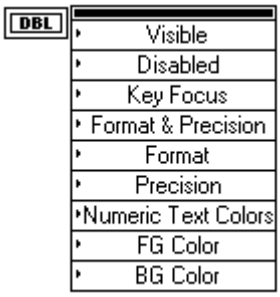


Рис. 3.15. Атрибутный узел числового индикатора

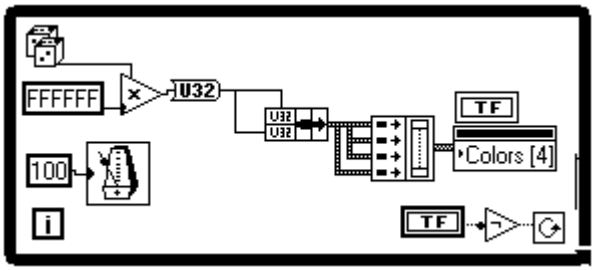


Рис. 3.16. «Разноцветная лампочка»

Каждый элемент списка может находиться как в состояниях «Режим записи» (Write Mode) или «Режим чтения» (Read Mode). На рис. 3.1.5 все атрибуты находятся в режиме записи, о чем информирует маленькая стрелочка в левой части каждой строки. Переключить режим как для отдельного элемента, так и для всего списка можно во всплывающем меню.

Присоединив к элементу списка константу соответствующего типа или средство управления, можно изменять значение атрибута и таким образом, влиять на свойство объекта лицевой панели – его координаты, цвет, видимость, доступность, формат представления данных и т.п. (см. рис. 3.15). На рис. 3.1.6. приведен пример управления объектом «круглая лампочка» - каждые 0.1 секунд случайным образом изменяется ее цвет.

Примечание. Цвет объекта представляет собой 32-битовое целое число, в котором используются 24 младших бита, причем каждые 8 битов кодируют компонент R, G или B. Так же, для задания цвета можно использовать константу типа «Color constant», раскрасив ее при помощи инструмента «кисточка».

4. Объекты блок-схемы

Блок-схема ВП представляет собой множество терминалов и узлов, соединенных при помощи связей. Можно выделить следующие типы узлов:

- функциональные узлы;
- управляющие структуры;
- атрибутные узлы;
- формульные узлы;
- внешние субВП и сервисные субВП-утилиты;
- узлы интерфейса с внешним кодом.

Все эти узлы сгруппированы в палитрах, появляющихся на экране при нажатии правой кнопки мыши внутри окна блок-схемы.

Примечание. Внешний вид палитр, их состав и способ группировки узлов в них сильно зависят от версии LabVIEW и от варианта комплектации пакета.

В палитре “Arithmetic” (“Арифметика”) содержатся функциональные узлы, выполняющие арифметические и логические операции над числовыми данными, например узел “Умножение”.

В палитре “Trig & Log” (“Тригонометрические и логарифмические”) содержатся узлы, реализующие тригонометрические и логарифмические функции различных видов, например узел “Синус от целочисленного аргумента”.

В палитре “Comparison” (“Сравнение”) содержатся узлы, выполняющие операции сравнения значений различных типов, например узел “Сравнение с нулем”.

В палитре “Conversion” (“Преобразование”) содержатся узлы, выполняющие преобразования из одного типа данных в другой, например узел “Преобразование логического значения в соответствующий элемент числового множества {0,1}”.

В палитре “String” (“Строка”) содержатся функциональные узлы, выполняющие различные операции над текстовыми строками, например узел “Выделение подстроки”.

В палитре “Array & Cluster” (“Массив и кластер”) содержатся функциональные узлы, выполняющие операции над матрицами, например узел “Транспонирование матрицы”.

В палитре “File I/O” (“Файловый ввод-вывод”) содержатся функциональные узлы, выполняющие элементарные операции над файлами, например узел “Чтение данных из файла”.

В палитре “Time & Dialog” (“Время и диалог”) сгруппированы функциональные узлы, выполняющие операции синхронизации работы ВП в реальном времени (например, узел “Задержка на требуемое количество миллисекунд”), а также узлы, позволяющие организовывать простейший диалог с оператором.

В палитре “Miscellaneous” (“Разное”) сгруппированы узлы, поддерживающие относительно редко используемые возможности LabVIEW, такие, например, как обращение к программному коду, написанному на языке высокого уровня.

Следующие группы библиотечных компонентов LabVIEW не являются обязательными. Они могут присутствовать или отсутствовать в зависимости от варианта инсталляции системы.

В палитре “Analysis” (“Анализ”) содержатся функциональные узлы, реализующие распространенные алгоритмы обработки и анализа данных, например, узел “Вычисление среднеквадратичного отклонения по выборке”.

В палитре “DAQ” (“Сбор цифровых и аналоговых данных”) содержатся многочисленные узлы, представляющие собой “каркасы” и фрагменты алгоритмов управления устройствами фирмы National Instruments.

Палитра “Instrument I/O” (“Интерфейсы ввода-вывода”) объединяет ряд функциональных узлов, поддерживающих ввод-вывод средствами распространенных интерфейсов GPIB 488.2, RS-232-C и пр., например, узел “Чтение из COM-порта”.

В палитре “Network” (“Сеть”) сгруппированы узлы, поддерживающие взаимодействие систем через распространенные сетевые протоколы TCP/IP, UDP и пр.

Палитра “Tutorial” (“Учебник”) содержит несколько учебных примеров, рассмотренных в фирменной документации National Instruments.

В палитре “Utility” (“Утилиты”) собраны разнородные субВП, обеспечивающие ряд “полезных” (в том числе системных) действий и операций, например, узел “Чтение байта из порта ввода-вывода PC”.

4.1. Управляющие структуры

Для организации вычислительного процесса в LabVIEW имеется набор управляющих структур, аналогичных тем, какие используются в текстовых языках программирования. Они хранятся в палитре “Structs & Constants” (“Структуры и константы”).

4.1.1. Структура “CASE” (“Выбор”). Эта структура служит для построения “развилок” - фрагментов блок-схемы, выполняющихся только при определенном условии. Она состоит из нескольких кадров.

Выполняется только тот фрагмент блок-схемы, который помещен в “текущий” кадр. Таким образом, если количество “кадров” равно двум, структура “CASE” эквивалентна обычному условному оператору “IF” текстовых языков программирования. Если кадров больше, то эквивалентом структуры являются операторы “case” языка Паскаль или “switch” языка Си.

“Текущий” кадр определяется значением, подаваемым на терминал условия структуры, на прямоугольный элемент, помеченный знаком “?”, расположенный на левой стороне рамки структуры. Если количество кадров равно двум, то этот терминал воспринимает логические значения “TRUE” (“ИСТИНА”) и “FALSE” (“ЛОЖЬ”), в противном случае он воспринимает целые числа - номера кадров.

Щелкните правой кнопкой мыши на рамке структуры и в появившемся всплывающем меню выберите:

- Add Case After (“Добавить кадр после”), если Вы хотите добавить новый кадр после текущего;
- Add Case Before (“Вставить кадр перед”), если Вы хотите вставить новый кадр перед текущим;
- Duplicate Case (“Размножить кадр”), если Вы хотите создать копию текущего кадра вместе с находящимся внутри него фрагментом блок-схемы;
- Remove Case (“Удалить кадр”), если вы хотите удалить кадр.

При редактировании блок-схемы Вы можете “перелистывать” кадры, щелкая по стреловидным кнопкам в верхней части рамки структуры.

Фрагменты блок-схемы, находящиеся внутри рамок структуры “CASE”, обмениваются данными с внешними объектами через шлюзы данных, расположенные на рамках структуры. Такие шлюзы создаются автоматически при проведении связи “катушкой” через рамку структуры. Чтобы создать такой шлюз принудительно, подведите “катушкой” связь к рамке структуры и щелкните левой кнопкой мыши. Для удаления шлюза разметьте его “стрелочкой” и нажмите клавишу “Del”.

Примечание. Если на рамке структуры “CASE” присутствует шлюз данных, то связи с ним должны быть определены во всех кадрах структуры.

На рисунке 4.1 представлен простой пример использования структуры “CASE” - фрагмент блок-схемы, расставляющий два числа (X и Y) “по ранжиру”.

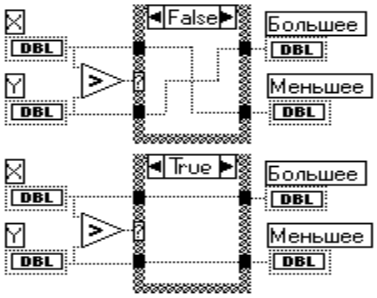


Рисунок 4.1. Использование структуры “CASE”

4.1.2. Циклы и сдвиговый регистр. Циклы “WHILE” (“Пока”) и “FOR” (“Для”) служат для организации итеративного (повторяющегося) выполнения фрагмента блок-схемы, помещенного внутрь их рамки. Размер их рамки может быть изменен при помощи “стрелочки”.

Цикл “WHILE” обязательно содержит внутри рамки два предопределенных компонента - терминал условия завершения и счетчик итераций.

Терминал условия завершения - это маленькая квадратная пиктограмма зеленого цвета в правом нижнем углу рамки цикла. Он способен воспринимать только логические значения “TRUE” (“ИСТИНА”) и “FALSE” (“ЛОЖЬ”) и предназначен для управления выполнением цикла: фрагмент блок-схемы внутри рамки цикла будет повторяться до тех пор, пока на этот терминал подается значение “TRUE” (“ИСТИНА”).

Счетчик итераций (квадратная пиктограмма с буквой “i” внутри) содержит целое число - номер текущего повторения фрагмента блок-схемы, заключенного в рамке цикла.

Цикл “FOR” содержит внутри рамки цикла аналогичный счетчик итераций, а в левом верхнем углу рамки находится терминал числового предела (квадратная пиктограмма с буквой “N” внутри). Этот терминал воспринимает целочисленное значение, поступающее извне рамки цикла, - это количество повторений, которое должен совершить цикл.

Фрагменты блок-схемы, располагающиеся внутри рамок циклов “WHILE” и “FOR”, могут обмениваться данными с внешними объектами через шлюзы данных, расположенные на рамках структуры. Шлюзы данных создаются автоматически при проведении инструментом “катушка” связей через рамку цикла. Следует иметь в виду, что порожденные внутри цикла скаляры во время его (цикла) работы накапливаются на его (цикла) рамке и после завершения последней итерации образуют массивы значений.

Кроме того, при помощи сдвиговых регистров возможна передача данных из одной итерации в другую.

Щелкните правой кнопкой мыши на рамке цикла и выберите во всплывающем меню пункт “Add Shift Register” (“Добавить сдвиговый регистр”). На левой и правой сторонах рамки цикла появятся симметрично расположенные шлюзы сдвиговых регистров. Правый шлюз, содержащий изображение направленной вверх стрелочки, позволяет передавать данные из текущей итерации в следующую. Левый шлюз, содержащий изображение направленной вниз стрелочки, позволяет в текущей итерации принимать данные из предыдущей.

Также при помощи сдвиговых регистров возможен обмен данными между “далеко отстоящими” итерациями. Щелкните правой кнопкой мыши на одном из шлюзов сдвигового регистра и выберите из всплывающего меню “Add Element” (“Добавить элемент”). К шлюзу будет добавлен еще один элемент, позволяющий “заглянуть” на одну итерацию дальше в “прошлое” или “будущее”, в зависимости от того, на левом или правом шлюзе выполнялась эта операция. Удалять элементы можно при помощи пункта “Remove Element” (“Удалить элемент”) того же меню. Весь сдвиговый регистр можно удалить при помощи пункта “Remove ALL” (“Удалить все”).

На рисунке 4.2 приведен пример использования циклов и сдвиговых регистров для сглаживания экспериментальных данных методом скользящего среднего по 3-м точкам.

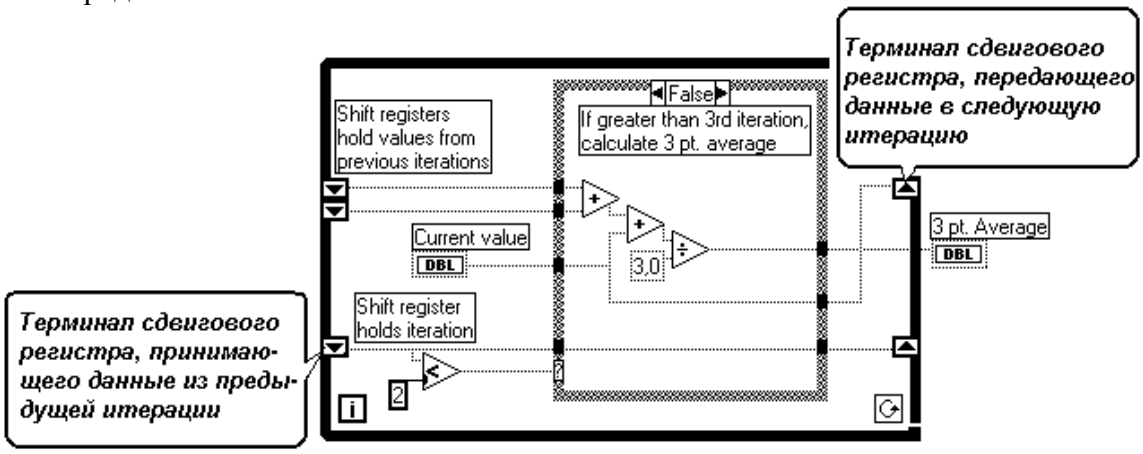


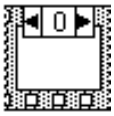
Рисунок 4.2. Использование цикла “WHILE” и сдвиговых регистров в ВП “Скользящее среднее по 3-м точкам”

4.1.2. Структура “SEQUENCE” (“Последовательность”). В LabVIEW применена схема, использующая для организации вычислительного процесса принцип *управления потоками данных*. Это означает, что

> **ни один узел, ни один фрагмент блок-схемы не начнет свое выполнение до тех пор, пока на всех его входных связях не появятся требуемые данные**.

С другой стороны,

> **узлы и фрагменты, не зависящие друг от друга по данным, запускаются в произвольном порядке и выполняются параллельно**.



Чтобы обеспечить необходимую последовательность исполнения не зависящих друг от друга фрагментов блок-схемы, в LabVIEW используется структура “SEQUENCE” (“Последовательность”). Она состоит из нескольких последовательно пронумерованных кадров (фреймов). Сначала выполняется фрагмент блок-схемы, размещенный в нулевом кадре, потом в первом, и т.д.

Щелкните правой кнопкой мыши на рамке структуры и в появившемся всплывающем меню выберите:

- Add Frame After (“Добавить кадр после”), если Вы хотите добавить новый кадр после текущего;
- Add Frame Before (“Вставить кадр перед”), если вы хотите вставить новый кадр перед текущим;
- Duplicate Frame (“Размножить кадр”), если хотите создать полную копию текущего кадра, вместе с расположенным внутри фрагментом блок-схемы;
- Make This Frame (“Присвоить кадру номер”), если Вы хотите перенумеровать кадры;
- Remove Frame (“Удалить кадр”), если вы хотите удалить кадр.

При редактировании блок-схемы Вы можете “перелистывать” кадры, щелкая по стреловидным кнопкам в верхней части рамки структуры.

Для передачи данных из одного кадра структуры “SEQUENCE” в другой используются *локальные переменные последовательности*. Щелкните правой кнопкой мыши на рамке структуры и из всплывающего меню выберите Add Sequence Local (“Добавить локальную переменную последовательности”). На рамке текущего кадра (и всех последующих кадров) появится прозрачный прямоугольный шлюз локальной переменной. Когда Вы соедините шлюз с каким-нибудь источником или приемником данных, внутри шлюза появится изображение стрелки, показывающей направление передачи данных.

Примечание. Шлюз локальной переменной виден также на рамках предыдущих кадров последовательности, но там он недоступен для связывания.

Для удаления локальной переменной структуры “SEQUENCE” щелкните правой кнопкой мыши на ее терминале и выберите Remove (“Удалить”).

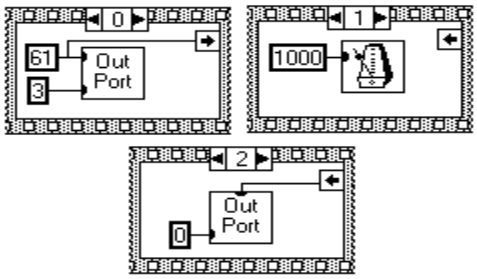


Рисунок 4.3. Использование структуры “SEQUENCE”

На рисунке 4.3 приведен пример использования структуры “SEQUENCE” - фрагмент ВП, производящего односекундное звучание встроенного динамика компьютера. Локальная переменная последовательности в этом примере использована для передачи номера порта ввода-вывода (все константы – шестнадцатеричные) из нулевого кадра в третий.

4.2. Локальные и глобальные переменные

Для облегчения создания сложных блок-схем с большим количеством разнонаправленных связей в LabVIEW предусмотрены механизмы *локальных* и *глобальных переменных*. По сути они представляют собой “удаленные копии” терминалов для объектов лицевой панели. Правила их построения и использования рассмотрим на примере локальных переменных.

LOCAL

Шаг 1. Щелкните правой кнопкой мыши в любой позиции окна блок-схемы и в палитре “Structs & Constants” (“Структуры и константы”) выберите узел “Local Variable” (“Локальная переменная”).

Шаг 2. Щелкните правой кнопкой мыши на пиктограмме появившегося узла и во всплывающем меню выберите пункт “Select Item” (“Выбрать объект”). В появившемся меню появится список **поименованных** объектов лицевой панели. Выберите объект, с которым вы хотите ассоциировать локальную переменную.

Примечание. Если Вы не присвоили объекту лицевой панели имени на этапе его создания, это можно сделать в любой момент, включив отображение текстовой метки и внеся в нее информацию при помощи инструмента “буковка”.

После создания локальной переменной Вы можете перемещать ее в любую позицию блок-схемы и “размножать”, после чего использовать в качестве “копии” терминала объекта передней панели.

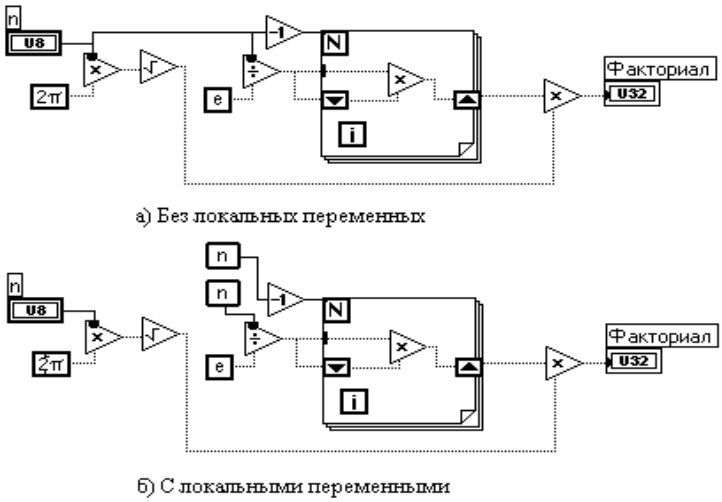


Рисунок 4.4. Варианты ВП для расчета факториала

Примечание. Во всплывающем меню локальной переменной имеется пункт “Change To Read/Write Local” (“Сделать локальную переменную доступной для чтения/записи”), позволяющий придать ей статус источника или приемника данных вне зависимости от типа связанного с ней объекта передней панели.

Преимущества использования локальных переменных продемонстрируем на примере ВП (см. рисунок 4.4), вычисляющего значение факториала по приближенной формуле Стирлинга: $n! \approx \sqrt{2\pi\, n}\left(n/e\right)^n$.

GLOB Глобальные переменные отличаются от локальных тем, что при помощи пункта “Open Front Panel” (“Открыть лицевую панель”) всплывающего меню для них создается отдельный субВП, содержащий средства управления и индикаторы. Этот субВП потом может быть подключен к нескольким ВП одновременно, а терминалы размещенных на его лицевой панели объектов использоваться как общий разделяемый ресурс.

4.3. Формульный узел

Структура под названием *формульный узел* используется в LabVIEW для упрощения создания блок-схем, реализующих вычисления по сложным математическим формулам.

Шаг 1. Щелкните правой кнопкой мыши в любой позиции окна блок-схемы и в палитре “Structs & Constants” (“Структуры и константы”) выберите узел “Formula Node” (“Формульный узел”). Увеличьте размеры появившейся рамки до необходимой величины при помощи инструмента “стрелочка”.

Шаг 2. Щелкните правой кнопкой мыши на левой и правой сторонах рамки, во всплывающем меню выберите пункты “Add Input” (“Добавить вход”) и (или) “Add Output” (“Добавить выход”). Озаглавьте созданные таким образом шлюзы входных и выходных переменных при помощи инструмента “буковка”.

Шаг 3. При помощи инструмента “буковка” введите внутрь рамки текст формулы, содержащий в качестве составных элементов имена входных и выходных переменных.

Синтаксис формул похож на используемый в языке Си и требует использования следующих обозначений:

- “=” - присвоение;
- “**? :**” - условное выражение в стиле языка Си (например, **y=x>0?1:0**);
- “||” - логическое “ИЛИ”;
- “&&” - логическое “И”;
- “**!**=” и “==” - неравенство и равенство;
- “<”, “>”, “<=” и “>=” - другие отношения: “меньше”, “больше”, “меньше или равно”, “больше или равно”;
- “+” и “**-**” - сложение и вычитание;
- “*****” и “/” - умножение и деление;
- “+”, “**-**”, “**!**” - унарные операции: положительность, отрицательность, логическое “НЕ”;
- “^” - возведение в степень.

Разрешено использование нескольких операторов, разделенных символом “;” (точка с запятой). Возможно использование следующих предопределенных функций:

- **abs(x)** абсолютная величина x;
- **acos(x)** арккосинус x в радианах;
- **acosh(x)** гиперболический арккосинус x в радианах;
- **asin(x)** арксинус x в радианах;
- **asinh(x)** гиперболический арксинус x в радианах;
- **atan(x,y)** арктангенс для x/y в радианах;
- **atanh(x)** гиперболический арктангенс x в радианах;
- **ceil(x)** ближайшее целое, большее чем x;
- **cos(x)** косинус x в радианах;
- **cosh(x)** гиперболический косинус x в радианах;
- **cot(x)** котангенс x в радианах;
- **csc(x)** косеканс x в радианах;
- **exp(x)** значение e, возведенное в степень x;
- **explm(x)** значение e, возведенное в степень x минус единица;
- **floor(x)** ближайшее целое, меньшее чем x;
- **getexp(x)** порядок;
- **getman(x)** мантисса;
- **intrz(x)** ближайшее к x целое, размещенное между 0 и x;
- **ln(x)** натуральный логарифм x;
- **lnpl(x)** натуральный логарифм для (x+1);
- **log(x)** десятичный логарифм x;
- **log2(x)** двоичный логарифм x;

- **max(x,y)** максимум из x и y;
- **min(x,y)** минимум из x и y;
- **mod(x,y)** вещественный остаток от деления x на y;
- **rand()** случайное число, равномерно распределенное на интервале от 0 до 1;
- **rem(x,y)** целый остаток от деления x на y;
- **sec(x)** вычисляет секанс x в радианах;
- **sign(x)** знак числа x;
- **sin(x)** синус x в радианах;
- **sinc(x)** синус x, деленный на x в радианах;
- **sinh(x)** гиперболический синус x в радианах;
- **sqrt(x)** квадратный корень x;
- **tan(x)** тангенс x в радианах;
- **tanh(x)** гиперболический тангенс x в радианах;
- **x^y** значение x в степени y.

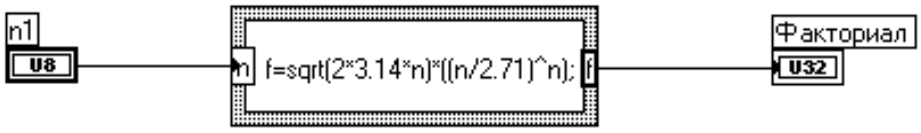


Рисунок 4.5. Использование формульного узла

На рисунке 4.5 приведен пример использования формульного узла для вычисления факториала по приближенной формуле Стирлинга (см. п. 4.3).

4.4. ВП файлового ввода-вывода

LabVIEW поддерживает три режима работы с файлами:

- двоичный режим - самый быстрый и универсальный метод файлового ввода-вывода, при котором файлы рассматриваются как наборы байтов;
- текстовый режим, при котором файлы рассматриваются как потоки символов ASCII;
- datalog - это режим, специально разработанный фирмой National Instruments для сохранения больших объемов типизированной информации и последующего удобного доступа к ней.

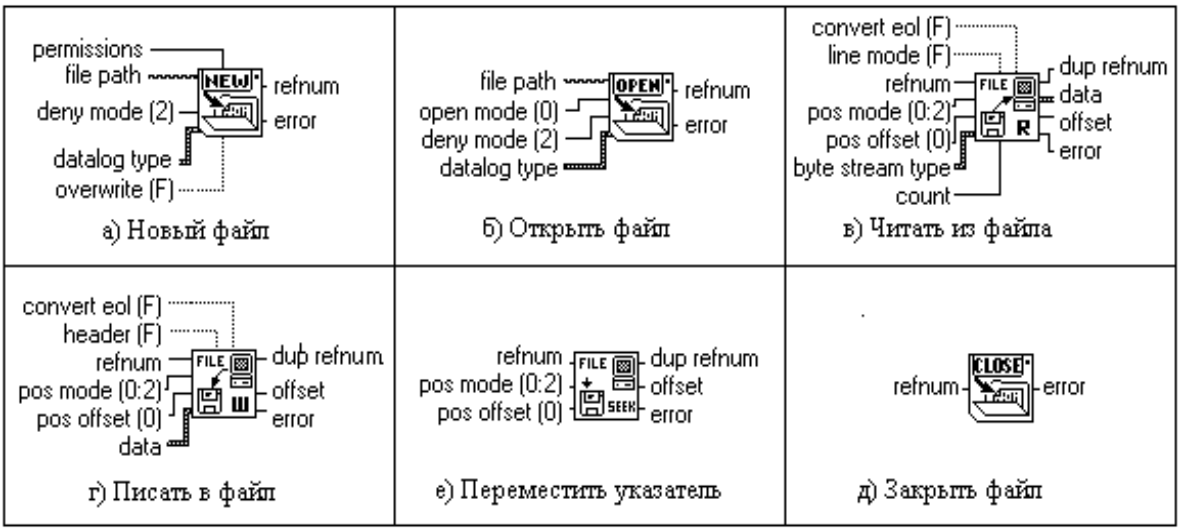


Рисунок 4.6. Некоторые ВП файлового ввода-вывода

Большинство файловых ВП сконцентрировано в палитре “File I/O” (“Файловый ввод-вывод”). Наиболее важные из них изображены на рисунке 4.6.

В LabVIEW используется традиционная схема доступа к файлам. Перед работой с файлом его необходимо открыть при помощи ВП “Open File” (“Открыть файл”) или создать помощи ВП “New File” (“Новый файл”). Этим ВП необходимо передать имя файла (“file path”), причем оно обязательно должно быть типа “Путь доступа к файлу” (“Path”), но не “Строка” (“String”).

На вход “режим открытия” (“open mode”) этих ВП может быть подан один из следующих кодов доступа к файлу: 0 - разрешены чтение и запись (этот режим устанавливается по умолчанию, если ко входу ничего не присоединено); 1 - только чтение; 2 - только запись; 3 - только запись в принудительно очищенный файл. Если ко входу “тип записи” (“datalog type”) присоединено произвольное значение какого-либо типа (целого, вещественного или сложного), то автоматически включается datalog-режим. При успешном выполнении эти ВП возвращают числовой идентификатор (“refnum”) открытого файла.

Собственно ввод-вывод осуществляется при помощи ВП “Read File” (“Читать из файла”) и “Write File” (“Писать в файл”), которым нужно передать числовой идентификатор открытого файла (“refnum”). Данные передаются-принимаются через входные или выходные связи, помеченные как “data”. Если ко входу “тип потока байтов” (“byte stream type”) присоединено значение какого-либо типа, то эти операции будут выполняться над блоками данных соответствующего типа. Также эти ВП имеют “сквозной” выход числового идентификатора (“dup refnum”) для организации последовательных цепочек ВП.

Позицию чтения-записи в файле можно напрямую изменить при помощи ВП “Seek” (“Переместить указатель”). На вход “относительная позиция” (“pos offset”) этого ВП необходимо подать значение смещения, а на вход “режим позиционирования” (“pos mode”) - код способа определения нового местоположения указателя: 0 - относительно начала файла; 1 – относительно конца файла; 2 - относительно текущей позиции.

После окончания работы с файлом его необходимо закрыть при помощи ВП “Close File” (“Закрыть файл”).

Все вышеупомянутые ВП возвращают код ошибки (“error”), соответствующий результату выполнения файловой операции: 0 - отсутствие ошибки; 1 - ошибка в аргументах; 4 - достигнут конец файла; 5 - попытка повторного открытия файла; 6 - ошибка ввода-вывода; 7- файл не найден; 8 - операция запрещена; 9 - диск полон и т.п. Поэтому целесообразно использование ВП “Simple Error Handler” (“Простой обработчик ошибок”), который прерывает выполнение ВП и отображает в отдельном окне причину возникшей ошибки.

Пример блок-схемы ВП, выполняющего побайтовое чтение из двоичного файла, приведен на рисунке 4.7.

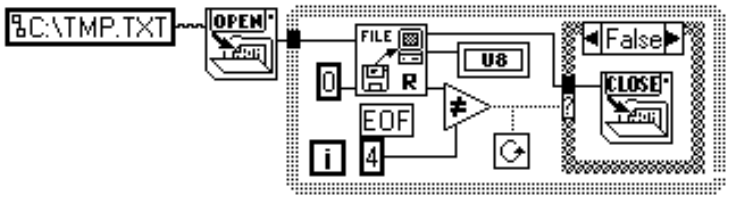


Рисунок 4.7. Побайтовое чтение из двоичного файла

В палитре “Utility/File” (“Файловые утилиты”) содержится большое количество ВП, выполняющих типовые высокоуровневые файловые операции. Например, ВП “Write Characters To File” (“Писать символы в файл”) реализует высокоуровневую обобщенную операцию открытия-записи-закрытия по отношению к текстовому файлу. Если ко входу “File Path” (“Имя файла”) этого ВП ничего не присоединено, то автоматически открывается диалоговое окно, позволяющее найти требуемый файл на диске или ввести имя нового файла с клавиатуры.

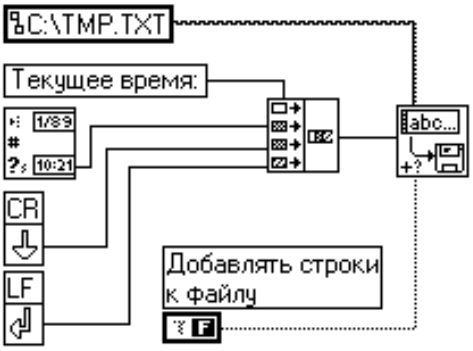


Рисунок 4.8. Запись строки в текстовый файл

Пример применения этого ВП приведен на рисунке 4.8. В этом примере использованы также ВП “Concatenate Strings” (“Соединить строки”), “Get Data/Time String” (“Получить дату и время в виде строки”), текстовая константа “Текущее время:”, а также предопределенные текстовые константы “CR” (“Возврат каретки”) и “LF” (“Перевод строки”).

4.5. ВП ввода-вывода через интерфейс RS-232

Последовательный интерфейс RS-232 (устройства на его основе известны также под наименованиями “COM-порт”, “асинхронный адаптер AUX”, “стык-2” и пр.) часто используется для последовательной передачи цифровых данных между двумя устройствами со скоростями от 110 бит/с до 115 Кбит/с на расстояние до 15 метров. Интерфейс способен функционировать в синхронном режиме, используя 25 сигналов различного назначения. Но в большинстве конкретных устройств (например, в COM-портах ПЭВМ) реализован асинхронный режим, использующий всего 9 линий. Также очень часто встречается передача данных всего по 3 линиям в режиме “нуль-модема”.

LabVIEW поддерживает связь со внешними устройствами через RS-232 при помощи ВП, сконцентрированных в разделе “Serial” (“Последовательный интерфейс”) палитры “Instrument I/O” (“Приборный ввод-вывод”). Наиболее важные из них изображены на рисунке 4.9.

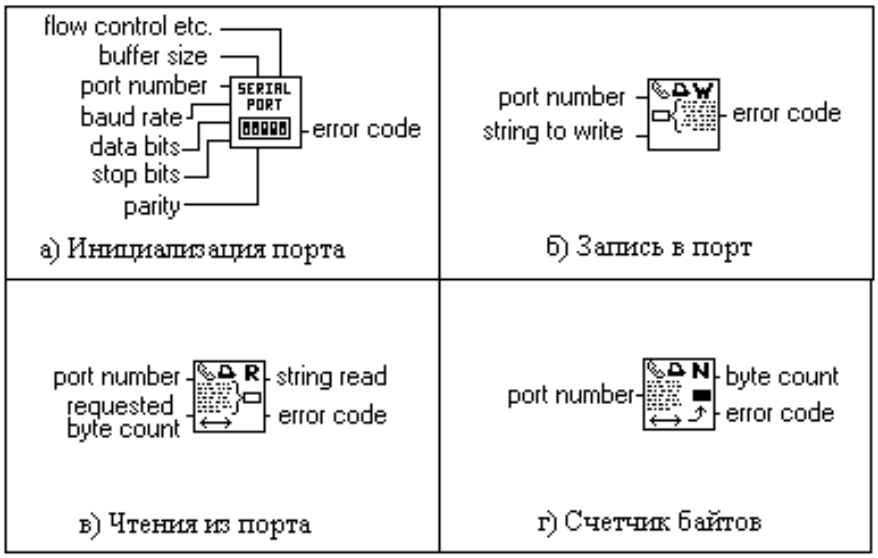


Рисунок 4.9. ВП для работы с RS-232

Используемый порт в ВП идентифицируется номером (“port number”): 0 - COM1, 1-COM2 и т.д.

Примечание. Номера, начинающиеся с 10, зарезервированы для использования не последовательными, но параллельными портами. Например, 10 соответствует LPT1, 11 - LPT2 и т.д. Это позволяет использовать некоторые из перечисленных ВП для работы с принтером.

Интерфейсы на разных концах линии связи должны быть согласованы по режиму передачи данных. Для настройки COM-порта ПЭВМ на конкретный режим используется ВП “Serial Port Init” (“Инициализация порта”). Параметрами настройки являются:

- скорость передачи данных (“baud rate”), которая выбирается из следующего ряда допустимых значений: 110, 150, 300, 600, 1200, 2400, 4800, 9600 (по умолчанию), 19200, 38400, 57600, 115200 бит/с.;
- количество битов, передаваемых в одном кадре (“data bits”): 5, 6, 7 или 8 (по умолчанию);
- количество стоповых битов (“stop bits”) в кадре: 1 (по умолчанию) или 2;
- бит четности (“parity”): 0 или 1.

Также ВП “Serial Port Init” позволяет определить размер промежуточного буфера (“buffer size”) для накопления передаваемых байтов (по умолчанию этот размер - 1024 байта).



Запись данных в порт осуществляется при помощи ВП “Serial Port Write” (“Запись в последовательный порт”), а чтение из него - при помощи ВП “Serial Port Read” (“Чтение из последовательного порта”). При приеме данных может быть полезен ВП “Bytes At Serial Port” (“Счетчик байтов”), который возвращает количество байтов, находящихся в буфере.

Пример работы с последовательным интерфейсом приведен на рисунке 4.10.

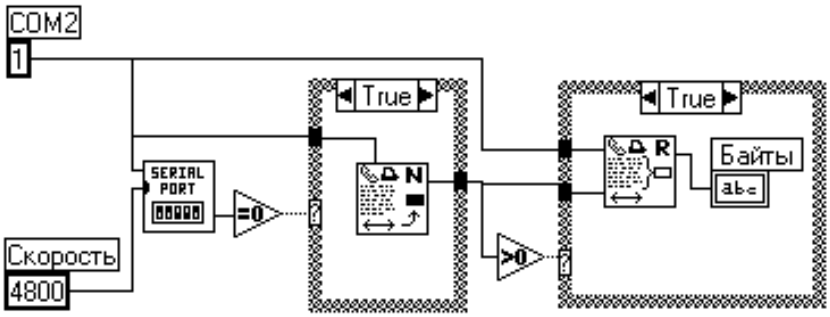


Рисунок 4.10. Чтение данных из последовательного порта COM-2

4.6. Сетевое программирование

Под аббревиатурой “TCP/IP” обычно понимается семейство сетевых технологий, основанных на двух протоколах:

- TCP - Transmission Control Protocol (протокол управления передачей), который отвечает за формирование и доставку “пакетов” данных (datagrams) от одного узла сети к другому;
- IP - Internet Protocol (межсетевой протокол), который отвечает за маршрутизацию в сети, состоящей из множества узлов.

Каждый узел сети (компьютер, прибор и т.п.) идентифицируется уникальным 4-байтовым IP-адресом, который традиционно записывается в виде AAA.BBB.CCC.DDD. Информационное взаимодействие между узлами обычно осуществляется по схеме “клиент-сервер”, причем на одном узле возможна одновременная работа нескольких серверов, различающихся уникальным числовым идентификатором (номером порта). Номера портов для серверов стандартных служб и протоколов фиксированы: 7 - Echo, 80 - HTTP, 110 - POP3 и т.п., поэтому для пользовательских нужд рекомендуется использовать значения в интервале 4097-65535.

ВП LabVIEW, обеспечивающие программирование сетевых приложений в рамках технологий TCP/IP, сконцентрированы в разделе “TCP” палитры “Network” (“Сеть”). Наиболее важные ВП приведены на рисунке 4.11.

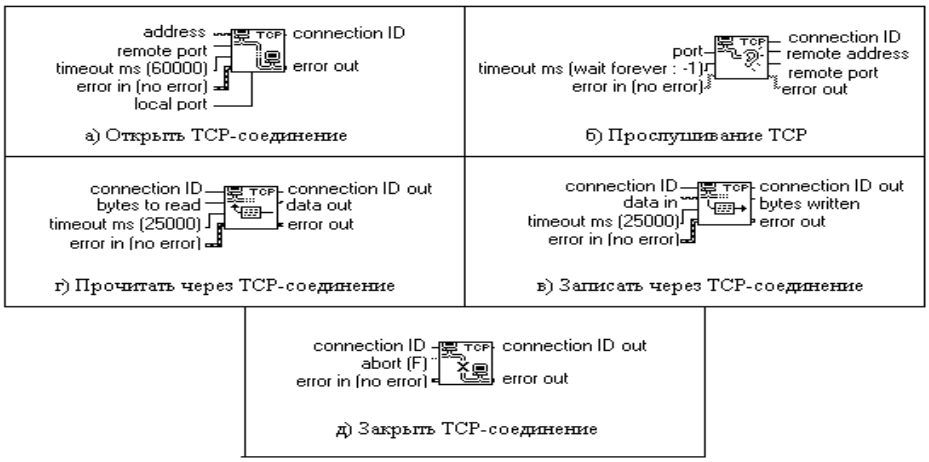


Рисунок 4.11. ВП для работы через TCP/IP

ВП “TCP Open Connection” (“Открыть TCP-соединение”) пытается открыть TCP-соединение. Входными параметрами этого ВП являются: IP-адрес (“address”) узла, с которым предполагается соединение; номер порта (“remote port”) для службы, с которой предполагается соединение; тайм-аут (“time out ms”), в течение которого производится попытка открыть соединение (по умолчанию его значение составляет 60000 мс=1 сек). При успешном открытии ВП возвращает числовой идентификатор установленного соединения (“connection ID”).

ВП “TCP Listen” (“Прослушивание TCP”) пытается обнаружить запрос на открытие TCP-соединения, исходящий от какого-то другого узла. Входными параметрами этого ВП являются: номер порта (“port”) протокола или службы, которая ожидает запрос; тайм-аут (“time out ms”), в течение которого выполняется прослушивание (по умолчанию это время бесконечно). При успешном обнаружении запроса ВП возвращает числовой идентификатор установленного соединения (“connection ID”) и TCP-адрес узла, выполнившего этот запрос (“remote address”).

После установления TCP-соединения узлы могут обмениваться информацией при помощи ВП “TCP-read” (“Прочитать через TCP-соединение”) и “TCP-write” (“Записать через TCP-соединение”). Этим ВП необходимо передать числовой идентификатор соединения и значение тайм-аута (“time out ms”), которое по умолчанию составляет 25 с. Входные и выходные данные передаются через параметры “data in” и “data out” в текстовой форме, т.е. в виде потока байтов. Для преобразования данных любого вида в текстовую форму целесообразно использование ВП “Type Cast” (“Преобразование типов”), который находится в палитре “Miscellaneous” (“Разное”).

После завершения обмена данными необходимо закрыть TCP-соединение при помощи ВП “TCP Close”. (“Закрыть TCP-соединение”)

Типичными действиями сетевого сервера являются: 1) ожидание запроса на соединение; 2) обмен данными с клиентом; 3) закрытие соединения. Пример простого ВП, иллюстрирующего эту последовательность действий, приведен на рисунке 4.12.

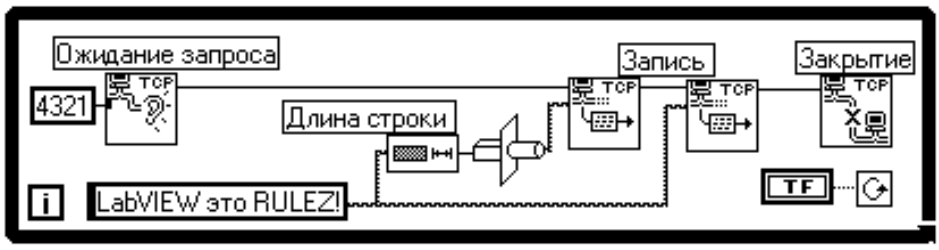


Рисунок 4.12. Пример сетевого сервера

Типичными действиями сетевого клиента являются: 1) попытка открыть соединение; 2) обмен данными с сервером; 3) закрытие соединения. Пример простого ВП, иллюстрирующего эту последовательность действий, приведен на рисунке 4.13.

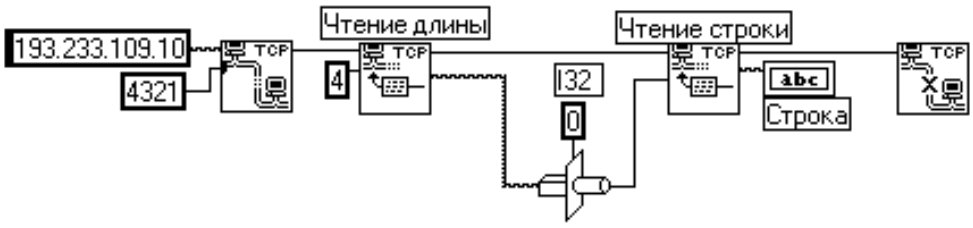


Рисунок 4.13. Пример сетевого клиента

Примечание. Установление соединения между клиентом и сервером еще не гарантирует успешного обмена данными, поэтому указанный этап рекомендуется повторять в цикле, каждый раз контролируя код ошибки, возвращаемый на выходе “error out”.

5. Расширенные возможности LabVIEW

5.1 Средства отладки виртуальных приборов

5.1.1 Поиск и устранение ошибок. Признаком работоспособного ВП является изображение “целой” стрелочки на кнопке запуска. Наоборот, если стрелочка “разрушена”, то ВП содержит какие-либо ошибки, препятствующие его нормальному функционированию.

Для того, чтобы определить местоположение и вид ошибки, щелкните левой кнопкой мыши по разрушенной стрелочке. На экране появится дополнительное окно “Error List” (“Список ошибок”) с меню, содержащим в качестве элементов наименования имеющихся ошибок. Выберите мышью интересующую Вас ошибку, и в том же окне появится текст, объясняющий причину ошибочной ситуации, а местоположение ошибки в окне блок-схемы будет помечено мерцанием. Перевод текстов сообщений об ошибочных ситуациях приведен в Приложении А.

5.1.2 Встроенный отладчик. В составе LabVIEW имеется встроенный отладчик графических блок-схем. Режим отладки включается и выключается специальной кнопкой на панели кнопок “быстрого доступа” LabVIEW (см. рисунки 2.1 и 4.6).

1. Для включения режима отладки нажмите на кнопку, содержащую изображение “карандашика” и “стрелочки”. Когда на этой кнопке “стрелочка” располагается поверх “карандашика”, то включен режим отладки; в противном случае LabVIEW находится в “нормальном” режиме создания, редактирования и выполнения ВП.

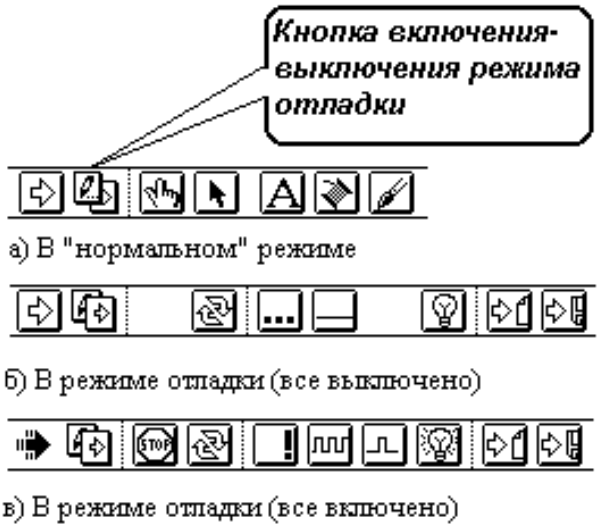


Рисунок 4.6. Панель кнопок “быстрого доступа” в разных режимах

2. Порядок работы фрагментов блок-схемы удобно проверять в *режиме пошагового исполнения*. Этот режим включается и выключается специальной кнопкой на панели отладки. В выключенном состоянии на этой кнопке изображена прямая линия, во включенном - меандр.

После запуска ВП при включенном режиме пошагового исполнения на панели появляется дополнительная кнопка с изображением одиночного “прямоугольного импульса”. Щелчок мышью по этой кнопке запускает на исполнение очередной фрагмент блок-схемы. При этом исполняемый в данный момент фрагмент блок-схемы выделяется мерцанием.

Примечание. Начиная с версии 4.0, в LabVIEW отсутствуют кнопки включения и выключения режима пошагового исполнения. При включенной отладке этот режим доступен по умолчанию, а запуск очередного шага вызывается нажатием специальных кнопок. Первая из них позволяет последовательно исполнять все узлы и структуры блок-схемы, а вторая - только узлы (фрагмент блок-схемы, заключенный внутри рамки какой-либо структуры рассматривается как целостный узел).

3. Промежуточные результаты вычислений, полученные при работе ВП, можно отслеживать в *режиме анимации*. Этот режим включается и выключается при помощи кнопки, содержащей изображение “лампочки”. В этом режиме ВП работает “сам”, но в замедленном темпе. При этом текущее местоположение вычислительного процесса помечается круглой “фишкой”, а промежуточные значения отображаются в текстовых “метках”, связанных с функциональными узлами (см. рисунок 4.7).

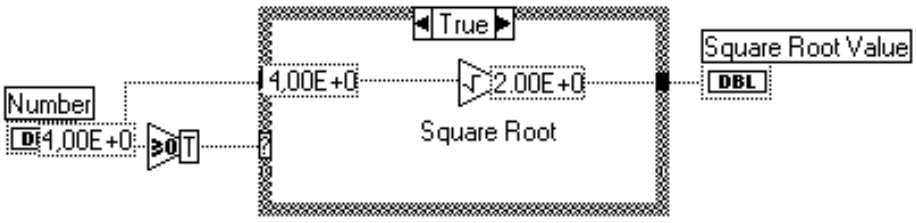


Рисунок 4.7. Работа ВП в режиме анимации

4. Для отладки ВП, содержащих большое количество субВП, удобно использовать механизм *контрольных точек*. Работающий ВП будет приостанавливать свое выполнение на тех субВП, у которых в панели средств отладки кнопка “многоточие” переключена в состояние “восклицательный знак”.

5. Также полезен для целей отладки *режим непрерывного исполнения*, при котором ВП, завершивший свою работу, немедленно запускается снова. Этот режим включается специальной кнопкой (“круговорот”).

Примечание. Для аварийного прерывания работы ВП служит кнопка “Стоп”. Кроме того, начиная с версии 4.0 в панели отладки LabVIEW присутствует кнопка “Пауза”.

5.2. Узел интерфейса к внешнему коду

В случае управления нестандартным оборудованием или применения сложных алгоритмов обработки данных возникает необходимость использования в ВП фрагментов, написанных на языках высокого уровня, например, на Си. Эта возможность доступна в LabVIEW благодаря узлу “Code Interface Node” (“Узел интерфейса к внешнему коду”). Английская аббревиатура термина - CIN, поэтому узлы интерфейса к внешнему коду часто называют CIN-узлами.

Примечание. Существует также возможность обращения к коду, оформленному в виде внешних библиотек динамической компоновки (DLL).

Правила конфигурирования CIN-узлов рассмотрим на конкретном примере.

Требуется создать функциональный узел, выполняющий чтение содержимого из двоичного файла, побайтовую сортировку данных и запись результата в другой файл. Входными параметрами для узла являются имена входного и выходного файлов.

Шаг 1. Создайте новый субВП. Разместите на лицевой панели два средства управления типа “Path” (“Путь доступа к файлу”).

Шаг 2. В окне блок-схемы разместите объект типа “Code Interface Node” (“Узел интерфейса к внешнему коду”), выбрав его из палитры “Miscellaneous” (“Разное”). По умолчанию он имеет вход только для одного параметра. Инструментом “стрелочка” увеличьте размеры пиктограммы узла так, чтобы нашлось место для двух параметров (см. Рисунок 4.8).

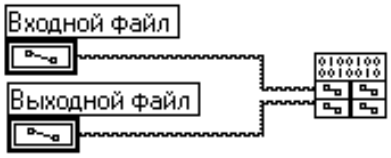


Рисунок 4.8. Связывание терминалов со входами CIN-узла

Примечание. Пиктограмма CIN-узла имеет вид “таблички”. Каждая строка таблички разделена на два поля, но соответствует одному параметру.

Шаг 3. При помощи инструмента “катушка” свяжите терминалы средств управления с соответствующими входами узла.

Шаг 4. Щелкните правой кнопкой мыши на пиктограмме CIN-узла и выберите из всплывающего меню пункт “Create .c file” (“Создать файл с текстом на языке Си”). В появившейся форме определите для этого файла имя (например, **example.c**) и местоположение (например, в каталоге c:\example).

Примечание. В LabVIEW версии 3.0 этот пункт меню называется “Create .h file” (“Создать заголовочный файл”). Файл с текстом программы на языке Си пользователь должен создавать самостоятельно.

Шаг 5. Обратитесь ко вновь созданному файлу. Он имеет следующий вид (комментарии переведены на русский язык):

```
/* Исходный код для CIN-узла */
#include "extcode.h"

/* "Заглушки" для необязательных функций */
UseDefaultCINInit
UseDefaultCINDispose
UseDefaultCINAbort
UseDefaultCINLoad
UseDefaultCINUnload
UseDefaultCINSave

CIN MgErr CINRun(Path _, Path _);

CIN MgErr CINRun(Path _, Path _) {

/* ВВОДИТЕ СВОЙ КОД СЮДА */

 return noErr;
}
```

“Главная” функция CIN-узла носит наименование **CINRun()**. Именно в нее требуется поместить текст алгоритма на языке Си, выполняющий требуемые действия.

Шаг 6. В тексте, описывающем алгоритм работы CIN-узла, **запрещено** использование стандартных для языка Си функций и описателей типов переменных. Вместо них необходимо использовать функции и типы, описанные в Приложении Б. Вот как будет выглядеть текст процедуры **CINRun()** с учетом этих ограничений:

```
CIN MgErr CINRun(Path s1, Path s2) {

 File  f1, f2; /* Хэндлы открытых файлов */
 int32 q;      /* Счетчик прочитанных и записанных байтов */
 uInt8 *b;     /* Указатель на буфер */
 uInt8 i,j,tmp;
```

```
b = (uInt8 *) DSNewPtr(BUFSIZE);
FMOpen(&f1,s1,openReadOnly,denyReadWrite);
FcreateAlways(&f2,s2,0x80,openReadWrite,denyReadWrite,"");
FMRead  (f1, BUFSIZE, &q, b);

for (i=2;i<q;i++) {
 j=i;
 while ((j>1)&&(b[j-1]>b[j]))  {
   tmp=b[j];
   b[j]=b[j-1];
   b[j-1]=tmp;
   j--;
 }
}

FMWrite (f2, BUFSIZE, &q, b);
FMClose (f1);
FMClose (f2);
DSDisposePtr(b);
return noErr;
}
```

Шаг 7. Откомпилируйте полученный файл **example.c**.

Компиляция 16-битового CIN-кода для LabVIEW 3.x и 4.x может осуществляться только при помощи компилятора фирмы Watcom. Предположим, что этот компилятор установлен в каталоге c:\watcom, а система LabVIEW - в каталоге c:\labview. Для описания Вашего проекта создайте файл **example.lvm** со следующим содержимым:

```
name=EXAMPLE
type=CIN
codeDir=C:\EXAMPLE
wcDir=C:\WATCOM
cinToolsDir=C:\LABVIEW\CINTOOLS
!include $(cinToolsDir)\generic.mak
```

Запустите компиляцию командой **wmake /f example.lvm**.

Компиляция 32-битового CIN-кода для LabVIEW 5.x и 6.x может осуществляться при помощи компилятора Visual C/C++ фирмы Microsoft. Предположим, что этот компилятор установлен в каталоге c:\vc, а система LabVIEW - в каталоге c:\labview. Проследите, чтобы в глобальной системной среде имелось определение CINTOOLSDIR=C:\LABVIEW\CINTOOLS. Для описания Вашего проекта создайте файл **example.lvm** со следующим содержимым:

```
name=EXAMPLE
type=CIN
!include $(CINTOOLSDIR) \ntlvsb.mak
```

Запустите компиляцию командой **nmake /f example.lvm**.

Шаг 8. Результатом успешной компиляции должен быть файл с расширением .lsb, например **example.lsb**. Щелкните правой кнопкой мыши на пиктограмме CIN-узла, выберите в меню пункт “Load Code Resource” (“Загрузить кодовый ресурс”) и укажите в появившейся форме Ваш файл **example.lsb**. Сохраните ВП.

Примечание. В отличие от “обычных”, ВП с CIN-узлами не могут быть перенесены из одной версии LabVIEW в другую.

5.3. «Экспресс» - ВП

Начиная с версии 7.0, в LabVIEW появились стандартные узлы специального вида – «экспресс»-ВП. Это заранее разработанные ВП, реализующие сложные, часто возникающие на практике задачи, например:
генерация сигналов различной формы, ввод-вывод с использованием оборудования от National Instruments, типовые методы цифровой обработки сигналов в режиме реального времени и пр. При этом пользователь освобождается от решения технических проблем, с которыми он столкнулся бы при использовании «традиционных» методов программирования в среде LabView, таких как: привязка данных к меткам реального времени, буферизация данных, синхронизация работы процедур ввода-вывода и обработки и т.п. Принципы работы с «экспресс»-ВП рассмотрим на конкретном примере:

Требуется сгенерировать (и отобразить в режиме реального времени) синусоидальный сигнал с частотой 1 Гц и диапазоном значений -1..1 ед, искаженный аддитивным шумом, равномерно распределенным на интервале от –0.5 до +0.5 ед.

Варианты решения «традиционными средствами» представлены на рис. 4.9.

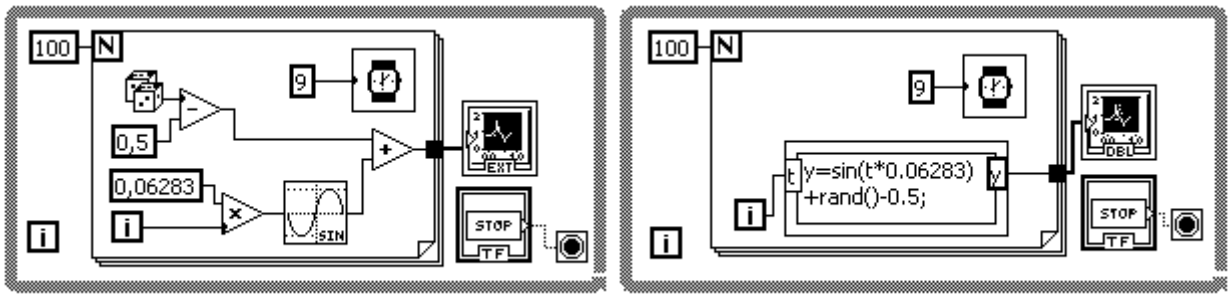


Рис. 4.9. «Традиционные» варианты генерации сигнала

Примечание. В современных версиях Windows реальная продолжительность задержки, выполняемой «Часиками», составляет N+1 миллисекунд..

Вариант решения с использованием «экспресс»-ВП «Simulate signal» («Имитация сигнала») изображен на рис. 4.10. Очевидно, он гораздо более компактен и прост при создании.

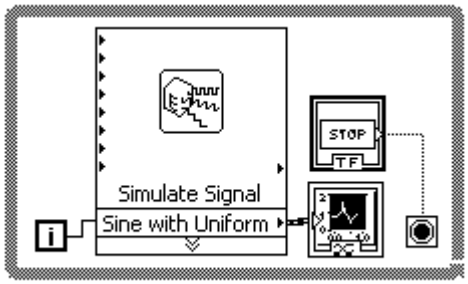


Рис. 4.10. средствами «экспресс»-ВП.

Гибкая настройка «экспресс» - ВП на условия конкретной задачи (выбор формы сигнала, частоты дискретизации, амплитуда шума и т.п.) выполняется не программно, а путем заполнения специальных форм (см. рис. 4.11).

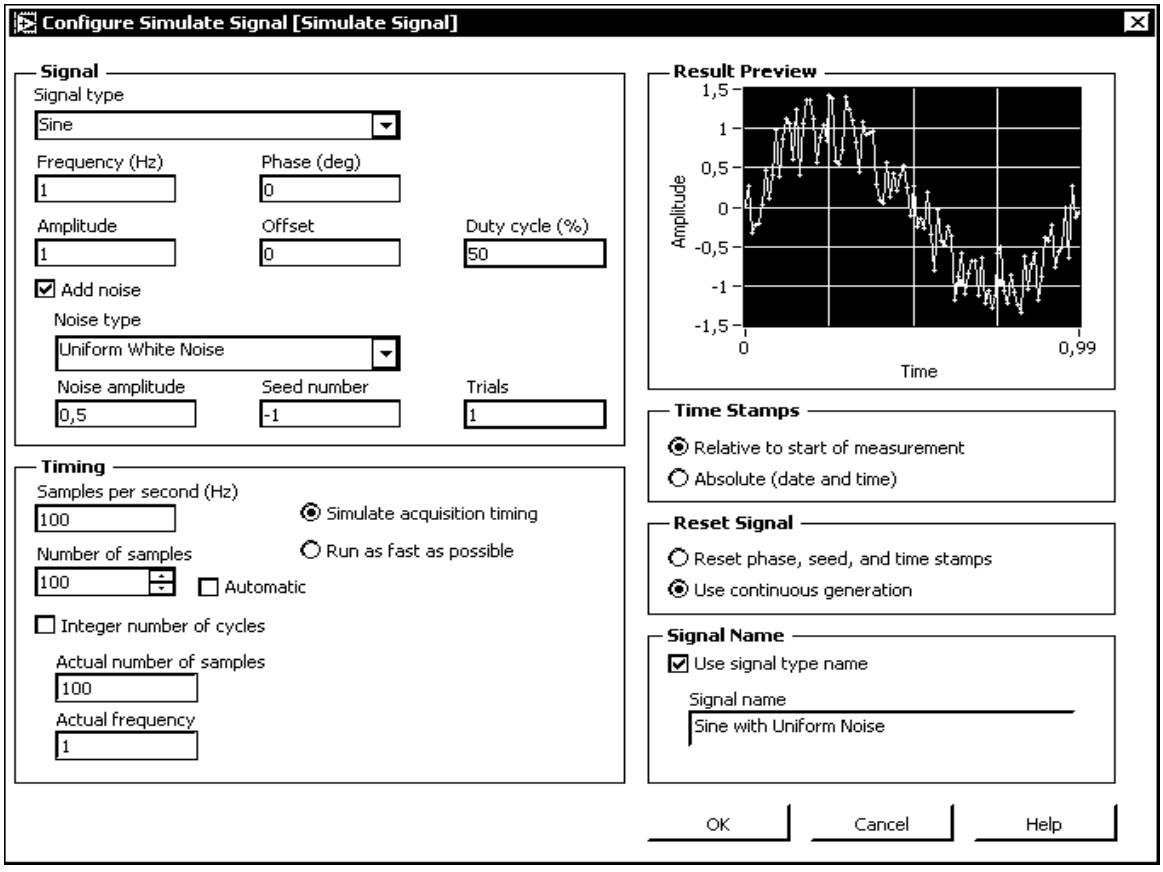


Рис. 4.11. Форма для выбора параметров «экспресс»-ВП

Форма позволяет определить:

- Signal type - вид сигнала (Sine – синусоидальный, Square – меандр, Triangle – треугольный, Sawtooth – пилообразный, DC - шум);
- Frequency - частоту изменения гармонического чигнала;
- Amplitude – максимальное отклонение значения сигнала от среднего;
- Noise type – тип шума (Uniform White Noise – равномерно распределенный белый шум, Gaussian White Noise – нормально распределенный белый шум и т.п.);

- Noise amplitude - максимальное отклонение значения шума от среднего;
- Samples per second – количество отсчетов сигнала, генерируемых в течение 1 сек;
- Number of samples – количество отсчетов сигнала, генерируемых в одной «пачке»;
- Simulate acquisition timing – признак привязки отсчетов к меткам реального времени;
- Run as fast as possible – признак генерации сигнала без привязки к реальному времени, с максимально возможной скоростью.
- Integer number of cycles – признак необходимости «склеивания» данных из различных «пачек» с целью имитации гладкого сигнала;
- и некоторые другие признаки.

«Экспресс»-ВП генерируют и используют данные сложного типа «массив кластеров», поэтому для работы с ними нельзя применять «традиционные» средства управления и индикаторы. Палитры LabVIEW предоставляют два комплекта объектов лицевой панели – «традиционных» и «экспресс»-совместимых.

6. Пример использования LabVIEW

Ряд типичных приемов, используемых при программировании на LabVIEW задач реального времени, проиллюстрируем на конкретном примере.

Требуется осуществить цифровую регистрацию с частотой 5 кГц случайного процесса, представляющего собой динамически изменяющееся в диапазоне 0-2В напряжение. В качестве средства измерения использовать звуковую карту.

Встроенный таймер LabVIEW имеет разрешение 1 мс, что позволяет организовывать сбор данных только с частотой порядка 1 кГц. Поэтому для решения задачи целесообразно использование возможностей, предоставляемых аппаратным таймером ПЭВМ.

6.1. Устройство и программирование звуковой карты

Современные звуковые карты (но не устройства с интерфейсом AC97) представляют собой сложные многофункциональные устройства, содержащие высокоточные (разрядность 16-24 бита) быстродействующие (время преобразования 5-25 мкс) АЦП и ЦАП (см. рисунок 6.1).

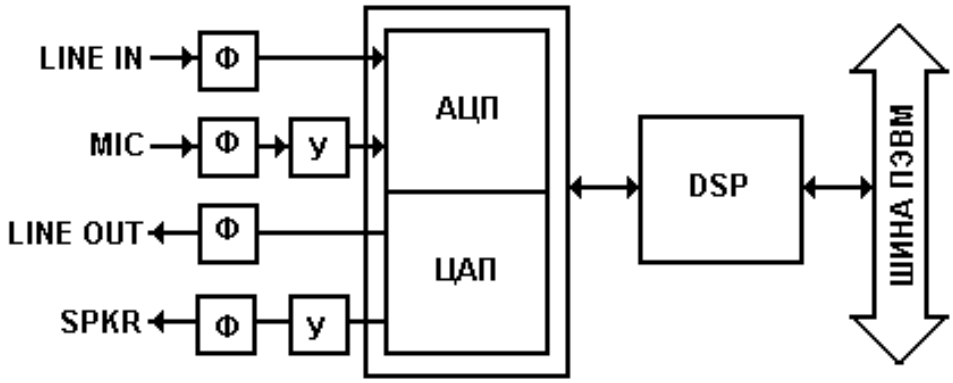


Рисунок 6.1. Функциональная схема звуковой карты

На этом рисунке: “LINE IN” - линейный вход, “MIC” - вход с микрофона, “LINE OUT” - линейный выход, “SPKR” - выход на наушники или внешние динамики, “Ф” - аналоговые фильтры, “У” - усилители аналоговых сигналов, “DSP” - процессор цифровых сигналов. На DSP возложено большое количество задач по управлению процессами аналогово-цифрового и цифро-аналогового преобразования, по синтезу и обработке цифровых сигналов, по обмену данными с ПЭВМ и пр. В частности, DSP поддерживает ряд сохранившихся еще с младших моделей Creative Soundblaster режимов работы звуковой карты, предусматривающих непосредственный 8-битовый одноканальный ввод-вывод.

Программное управление режимами работы DSP осуществляется через порты ввода-вывода (см. таблицу 6.1) при помощи специальных команд (см. таблицу 6.2). По умолчанию базовый адрес области портов ввода-вывода звуковой карты BASE=220h.

Таблица 6.1

Некоторые порты ввода-вывода звуковой карты

Адрес порта	Назначение
BASE+6	Регистр управления DSP
BASE+0Ah	Буферный регистр для чтения данных
BASE+0Ch	1. Буферный регистр для записи данных и команд 2. Регистр состояния буфера записи (если бит 7 сброшен, то разрешена запись в буферный регистр)
BASE+0Eh	Регистр состояния буфера чтения (если бит 7 установлен, то разрешено чтение из буферного регистра)

Таблица 6.2

Некоторые команды управления DSP

Команда	Описание
10h	Передача байта на ЦАП
20h	Чтение байта с АЦП
0D1h	Разрешить ввод-вывод
0D3h	Запретить ввод-вывод

Инициализация DSP состоит из следующих действий:

- послать значение 1 в порт BASE+6;
- выполнить временную задержку;
- послать значение 0 в порт BASE+6.

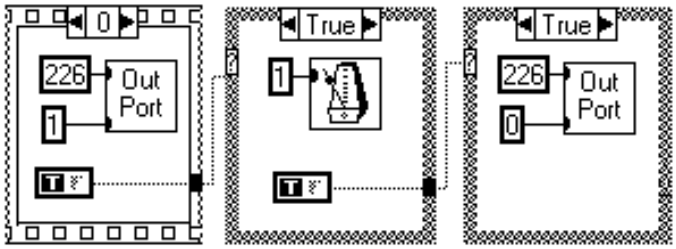


Рисунок 6.2. СубВП инициализации DSP

Чтение из DSP заключается в том, чтобы:

- в цикле ожидать готовности DSP (установки 7-го бита в порту BASE + 0Eh);
- прочитать байт из порта BASE+0Ah.

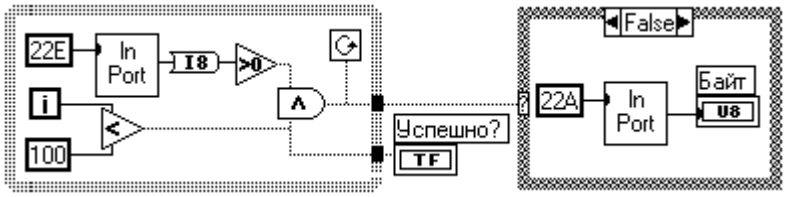


Рисунок 6.3. СубВП чтения из DSP

Аналогично выглядит запись в DSP:

- в цикле ожидать готовности DSP (сброса 7-го бита в порту BASE +0Ch);
- записать байт в порт BASE+0Ch.

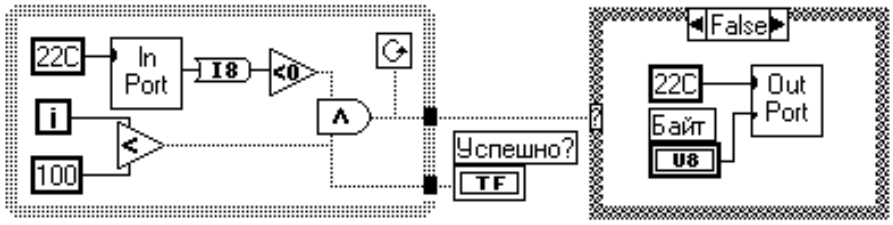


Рисунок 6.4. СубВП записи в DSP

Разрешение ввода-вывода заключается в посылке в DSP команды 0D1h, а запрещение ввода-вывода - в посылке команды 0D3h.

Чтение данных из АЦП звуковой карты складывается из следующих действий:

- послать в DSP команду с кодом 20h;
- прочитать байт данных из DSP.

В противоположность этому, посылка данных на ЦАП звуковой карты заключается в том, чтобы:

- послать в DSP команду с кодом 10h;
- послать в DSP байт данных.

6.2. Системный таймер ПЭВМ

Системный таймер современных ПЭВМ - это устройство, позволяющее вычислительной системе решать задачи *реального времени*, а именно:

- отслеживать моменты наступления программно-аппаратных событий;
- самостоятельно генерировать такие события в требуемые моменты времени.

Функционально системный таймер может быть представлен в виде совокупности трех “каналов”, работающих независимо друг от друга (см. рисунок 6.4). Работа каждого из каналов синхронизирована с тактовыми импульсами, поступающими с кварцевого генератора с частотой 1,193180 МГц.

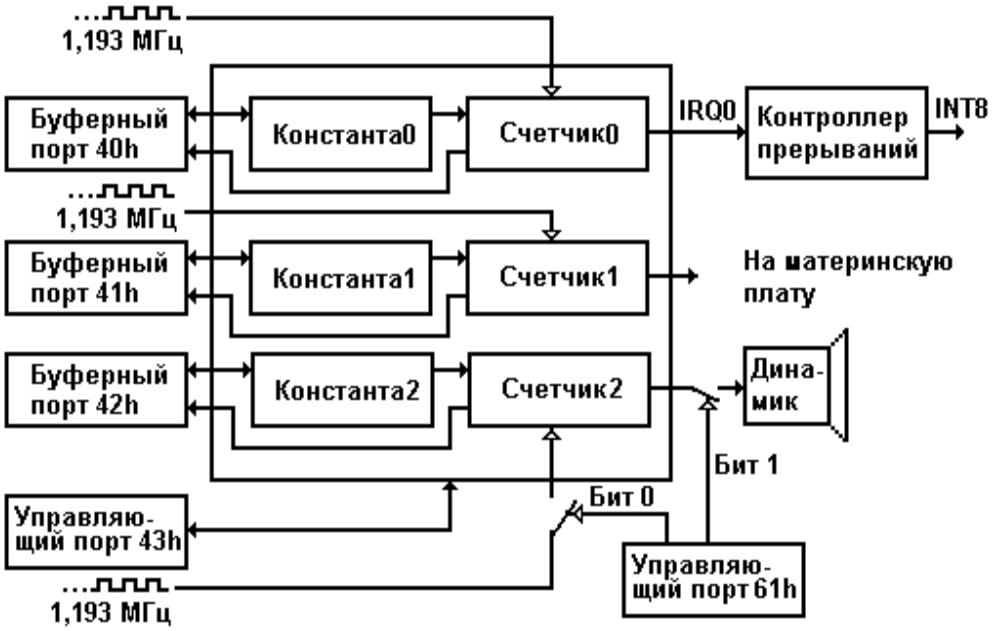


Рисунок 6.5. Функциональная схема системного таймера

Каждый из каналов может работать в одном из шести режимов. По умолчанию для всех каналов установлен режим №3. В этом режиме каналы работают следующим образом (см. рисунок 6.6):

- в счетчик таймера загружается некоторая константа пересчета в диапазоне 1..65535 (по умолчанию она имеет максимально возможное значение);
- по фронту каждого приходящего тактового импульса из счетчика вычитается значение 2;
- при достижении счетчиком значения 0 на выходе у канала таймера инвертируется значение сигнала “OUT”, кроме того в счетчик вновь загружается исходная константа и работа счетчика продолжается;
- по фронту сигнала “OUT” (т.е. при каждом втором обнулении счетчика) генерируется некоторое “событие”.

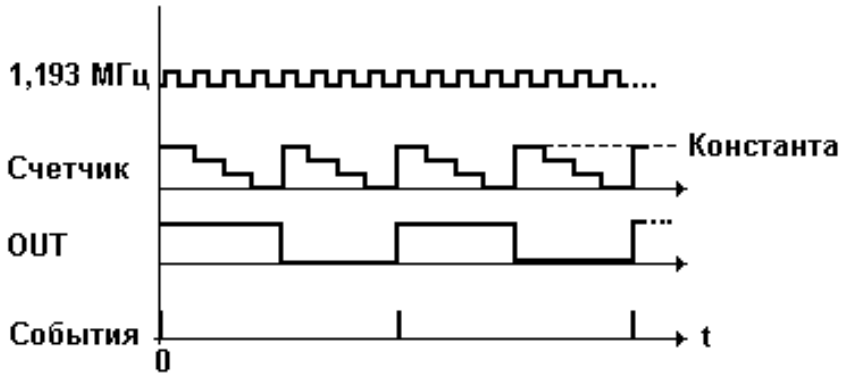


Рисунок 6.6. Работа каналов таймера в режиме №3

“События” канала 0 заключаются в том, что на контроллер прерываний подаются запросы с приоритетом Irq0 и, соответственно, в системе возникают прерывания с номером 8. Прерывание 8 используется операционной системой Windows для организации вытесняющей многозадачности.

“События” канала 1 заключаются в генерации тактовых импульсов, синхронизирующих системные процессы на материнской плате ПЭВМ.

“События” канала 2 представляют собой прямоугольные импульсы, подающиеся на вход встроенного компьютерного динамика. Управление работой канала и звучанием динамика осуществляется через порт 61h: бит 0 маскирует поступление в счетчик канала тактовой частоты, а бит 1 - маскирует передачу выходного сигнала канала на динамик.

Все три канала являются программируемыми: возможно переключение режимов работы канала, изменение константы пересчета, считывание и загрузка значений счетчиков и пр. Управление таймером осуществляется через порт 43h (см. таблицу 6.3).

Таблица 6.3

Управляющий регистр системного таймера

Биты	Назначение
0	Тип счета: 0 - двоичный , 1- двоично-десятичный
1..2	Режим работы канала: 011 - режим №3.
4..5	Команда: 00 - зафиксировать значение счетчика в буферном регистре; 01 - подготовить чтение/загрузку старшего байта счетчика; 10 - подготовить чтение/загрузку младшего байта счетчика; 11 - подготовить последовательное чтение/загрузку младшего и старшего байтов счетчика.
6..7	Номер канала: 00 - нулевой, 01-первый, 10-второй.

Обмен данными с каналами счетчика выполняется через порты 40h (нулевой канал), 41h (первый канал), 42h (второй канал).

Идея реализации сбора данных в режиме реального времени заключается в том, чтобы запрограммировать канал №2 таймера на работу с требуемой частой, сканировать в цикле значения счетчика и выполнять опрос АЦП в момент его (счетчика) перезагрузки.

Инициализация счетчика заключается в следующем (см. рисунок 6.6.):

- записать в порт 43h управляющее слово со значением 0B6h;
- последовательно записать в буферный регистр 42h младший и старший байты константы пересчета, соответствующей требуемой частоте опроса (0.0002с*1193180*2 1/с = 478);
- запустить счет (с одновременным отключением динамика) записью в порт 61h значения 1.

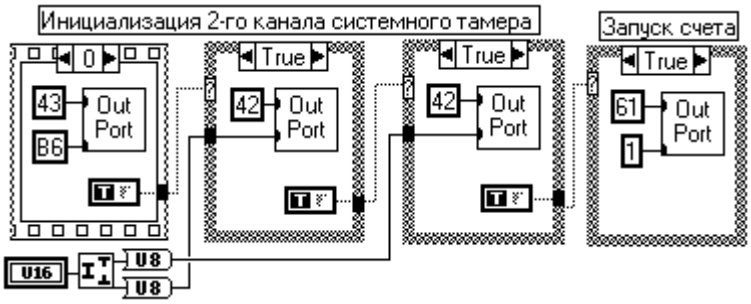


Рисунок 6.7. СубВП инициализации второго таймерного канала

Чтение текущего значения счетчика второго канала таймера можно выполнить так:

- подготовить чтение, послав в порт 43h управляющий байт со значением 86h;
- последовательно прочитать из буферного порта 42h сначала младший, а затем старший байты значения счетчика.

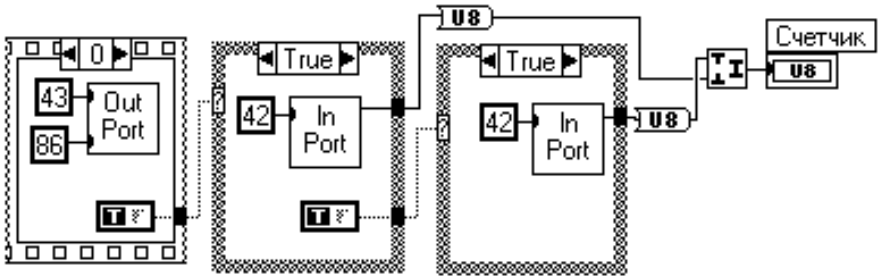


Рисунок 6.8. СубВП чтения счетчика второго таймерного канала

6.3. Работа с ВП сбора данных

Используя разработанные ранее и описанные выше субВП, можно скомпоновать ВП сбора аналоговых данных с использованием звуковой карты (см. рисунки 6.9-6.11).

Примечание. Лицевые панели большинства СубВП, использованных при построении демонстрационного ВП, имеют тривиальный вид, и поэтому на рисунках не отображены.

Правила работы с этим ВП следующие.

Шаг 1. Подключите на вход звуковой карты источник аналогового сигнала (например, микрофон). Также для проверки результатов работы подключите к выходу звуковой карты наушники или динамик.

Шаг 2. Загрузите ВП. Убедитесь, что на вход звуковой карты поступает информативный сигнал и запустите ВП. Через некоторое время, после заполнения буфера данными, на табло появится график оцифрованного сигнала, а в динамике прозвучит “эхо”.

Примечание. На входе звуковых карт ставятся фильтры нижних частот, поэтому использование их АЦП целесообразно только для измерения сигналов с частотами выше 100 Гц.

Рисунок 6.9. Лицевая панель демонстрационного ВП

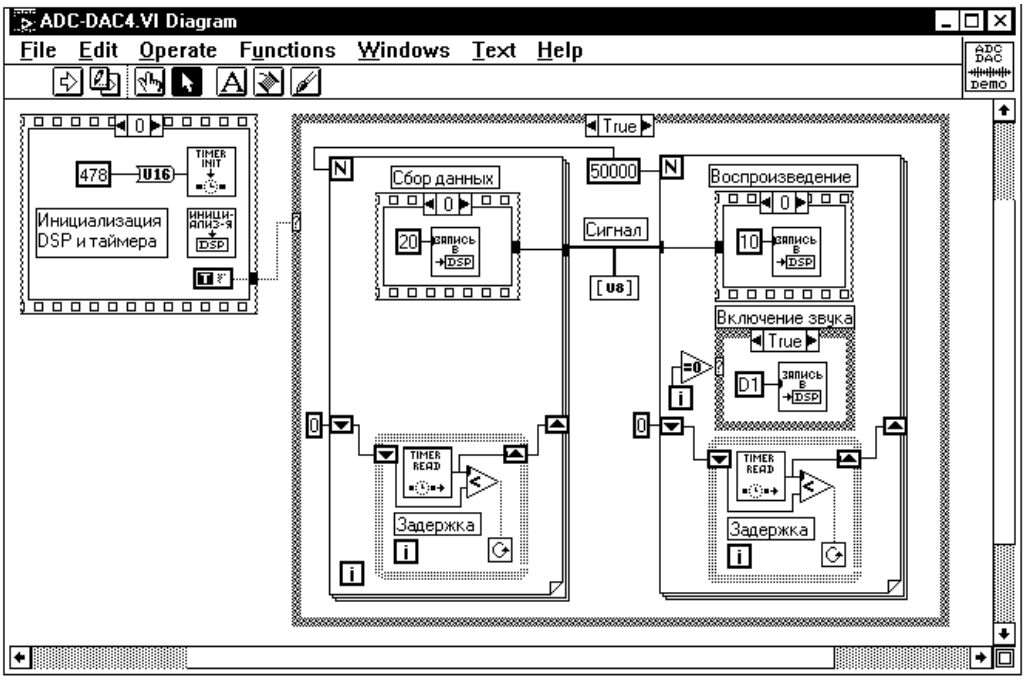


Рисунок 6.10. Блок-схема демонстрационного ВП

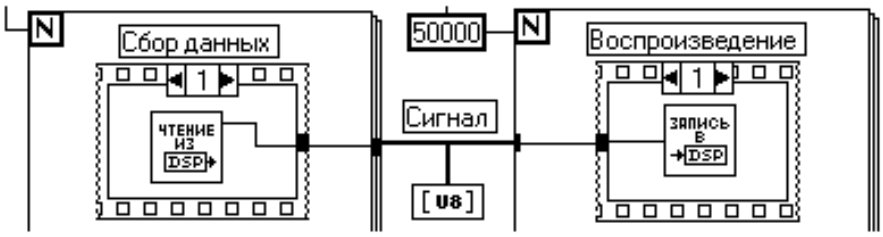


Рисунок 6.11. “Скрытые” кадры структур “Сбор данных” и “Воспроизведение”

Литература

1. Жарков Ф.П., Каратаев В.В., Никифоров В.Ф., Панов В.С. Использование виртуальных приборов LabVIEW. - М.:Солон-Р, 1999. - 268 с.

2. Муравьев С.В., Токарев С.В. Новый стиль в измерительном программировании // Приборы и системы управления. - 1997, №10.- С. 40-47.

3. Соболев В.С. Программное обеспечение современных систем сбора и обработки измерительной информации // Приборы и системы управления. - 1998, №1. - С. 55-63.

4. Программирование в среде Си для ПЭВМ ЕС / Романовская Л.М., Русс Т.В., Свитковский С.Г. - М.: Финансы и статистика, 1991. - 352 с.

5. Фролов А.В., Фролов Г.В. Аппаратное обеспечение IBM PC: В 2-х ч.- М.: Диалог-МИФИ, 1992. - 208 с.

6. Wells Lisa K., Travis Jeffrey. LabVIEW For Everyone: graphical programming made even easier. - Prentice-Hall, 1997. - 586 p.

7. LabVIEW Code Interface Reference Manual. - National Instruments Corp., 1993.

8. LabVIEW User Manual. - National Instruments Corp., 1993.

9. LabVIEW Tutorial. - National Instruments Corp., 1993.

10. LabVIEW for Windows. Demonstration Guide. - National Instruments Corp., 1992.

Приложение А. Сообщения об ошибках

<Function name>: contains unwired or bad terminal. (<Имя функции>: содержит неприсоединенный или плохой терминал). Вход ни к чему не присоединен или присоединен к связи неверного типа. Необходимо соединить требуемые входы с данными соответствующих типов.

<Code Interface Node>: object code not loaded. (<Узел интерфейса с кодом>: объектный код не загружен). Компоновка CIN-узла не завершена. Необходимо загрузить кодовый ресурс.

<Control>: control does not match its type definition. (<Средство управления>: средство управления не соответствует определению его типа). Отредактируйте средство управления, чтобы согласовать типы, либо отключите явное определение типа.

Enumeration has duplicate entries. (Перечисление содержит двойные вхождения). Все элементы в средстве управления перечислимого типа должны быть уникальны.

More Errors... (Имеются еще ошибки...). LabVIEW показывает не более 100 ошибок одновременно. Информация об остальных ошибках появится после того, как Вы устраните предыдущую сотню.

For Loop: N is unwired and there are no input indexing tunnels. (Цикл FOR : Параметр N ни к чему не присоединен, и отсутствует индексация входных туннелей). Если параметр N ни к чему не присоединен или размеры входного массива неизвестны, то цикл не может определить необходимое количество итераций.

Global or Local Variable: named component doesn't exist. (Глобальная или Локальная Переменная: именованный компонент не существует). Имя локальной или глобальной переменной изменилось со времени последней загрузки ВП, и теперь оно больше не соответствует никакому объекту лицевой панели. Выберите другое имя.

Global Variable: subVI is missing. (Глобальная Переменная: субВП отсутствует). LabVIEW не может найти субВП глобальной переменной, возможно потому, что имя субВП изменилось. Исправьте имя или создайте новый субВП.

<Node>: A subroutine priority VI cannot contain an asynchronous node. (<Узел>: ВП с приоритетом подпрограммы не может содержать асинхронного узла). Невозможно использовать асинхронную функцию (например, Wait) в ВП, который имеет уровень приоритета, соответствующий подпрограмме.

<Object>: Object is hidden. (<Объект>: объект скрыт). Объект скрыт, и это позже может привести к конфликтам.

Right Shift Register: type is undefined. (Правый сдвиговый регистр: тип неопределен). Вы должны удалить неиспользуемые сдвиговые регистры.

Right Shift Register: some but not all left sides are wired. (Правый сдвиговый регистр: некоторые левые шлюзы ни к чему не присоединены). Сдвиговый регистр должен иметь входы для всех левых шлюзов, либо ни для одного из них.

Sequence: One or more sequence locals were never assigned. (Структура SEQUENCE : Одна или более локальных переменных ни к чему не присоединена). Вы забыли в структуре Sequence присоединить локальную переменную. Удалите ее.

<SubVI name>: bad linkade to subVI. (<Имя субВП>: плохое соединение с субВП). Структура входов и выходов субВП изменилась с тех пор, как Вы последний раз загружали субВП, или Вы забыли определить входные и выходные объекты передней панели субВП. В первом случае Вы можете использовать команду Relink (Перестроить соединения) во всплывающем меню свойств субВП.

<SubVI name>: recursive references (dispose it). (<Имя субВП>: присутствуют рекурсивные ссылки (удалите их)). Вы изменили имя вызывающего ВП так, что оно стало таким же, как имя одного из вызываемых субВП, и LabVIEW теперь считает, что ВП вызывает сам себя рекурсивно. (ВП и субВП, вероятно, располагаются в различных каталогах). В LabVIEW запрещена рекурсия, то-есть попытка поместить ВП в собственную блок-схему.

<SubVI name>: LV Subroutine link error>. (<Имя субВП>: Ошибка связывания подпрограммы). Существует проблема компоновки с подпрограммой, написанной на языке высокого уровня.

<SubVI name>: A Subroutine priority VI cannot call a non-subroutine priority subVI. (<Имя субВП>: Подпрограмма с приоритетом ВП не имеет права

вызывать субВП с приоритетом не-подпрограммы). Вы должны заставить субВП выполняться с приоритетом подпрограммы или обратиться из ВП к чему-нибудь другому, чей приоритет равен приоритету подпрограммы.

<SubVI name>: subVI is already running. (<Имя субВП>: субВП уже выполняется). Нельзя запускать ВП, если один из его субВП уже выполняется как субВП другого ВП.

<SubVI name>: subVI is in either panel order or cluster order mode. (<Имя субВП> : субВП находится в режиме переупорядочивания панелей или кластеров). Вы не можете выполнять ВП во время изменения порядка панелей или кластеров одного из субВП.

<SubVI name>: subVI is in interactive retrieval mode. (<Имя субВП>: субВП находится в режиме интерактивного поиска). Нельзя запускать ВП, пока один из его субВП находится в режиме интерактивного поиска.

<SubVI name>: subVi is missing. (<Имя субВП>: субВП отсутствует). LabVIEW не может найти субВП после загрузки вызывающего ВП, возможно потому, что Вы изменили имя субВП. Замените плохой субВП на хороший, или укажите LabVIEW путь к отсутствующему субВП, если Вы можете найти его.

<SubVI name>: subVI is not executable. (<Имя субВП>: субВП не является исполнимой программой). СубВП неисправен. Откройте его и устраните ошибки.

Terminal: The associated array or cluster on the front paneel has no elements; its type is underined. (Терминал : связанный с ним массив или кластер на передней панели не имеет элементов; его тип неопределен). Вы должны поместить внутрь массива или кластера средство управления или индикатор.

Type Definition: Can't find valid type definition. (Определение типа: Невозможно найти допустимое определение типа). LabVIEW не может найти определение типов ВП после загрузки вызывающего ВП, возможно потому, что Вы изменили имя. Откройте отсутствующее определение типов ВП, если Вы можете найти его, спозиционируйтесь на плохом средстве управления и отсоедините его от определения типов или замените его на другое средство управления.

(Un) Bundle By Name: Empty cluster, or some components are unnamed. (Объединение по имени: Пустой кластер или некоторые компоненты не имеют имен). Входной кластер, присоединенный к функции Bundle By Name, или пуст, или содержит компоненты без текстовых меток.

Unit : bad unit syntax. (Модуль : плохой синтаксис модуля). Текст в узле содержит ошибку. Вопросительный знак помещается непосредственно перед нераспознанным символом.

Приложение Б. Библиотечные типы данных и функции

Библиотека LVLIB.LIB и включаемый файл EXTCODE.H предназначены для поддержки создания CIN-узлов на языке Си и содержат несколько сотен специфических объектов - функций, макросов, констант и типов данных. Далее кратко описывается небольшое подмножество этих, наиболее часто используемых, объектов.

1. Типы данных

Таблица А.1

Типы данных LabVIEW

Тип	Длина в байтах	Описание
Bool32	4	Логическое значение: 1 - “ИСТИНА”, 0 - “ЛОЖЬ”
LVBoolean	2	Логическое значение: <0 - “ИСТИНА”, ≥0 - “ЛОЖЬ”
int8	1	Целое число со знаком
int16	2	-"-
int32	4	-"-
uInt8	1	Целое число без знака
uInt16	2	-"-
uInt32	4	-"-
float32	4	Вещественное число с плавающей запятой
float64	8	-"-
floatExt	10	-"-
cmplx64	8	Комплексное число с вещественными полями re и im
cmplx128	16	-"-
cmplxExt	20	-"-
uChar	1	Символ
CStr	-	Массив символов, кончающийся байтом 0
PStr	-	Массив символов, длина хранится в байте с номером 0
LStr	-	Массив символов, длина хранится в первых четырех
CPStr	-	Список переменных типа PStr
Path	-	Путь и имя файла
Uptr	-	Указатель на байт
MgErr	4	Целое число, содержащее код ошибки
File	-	Дескриптор открытого файла

2. Функции для работы с динамической памятью

UPtr DSNewPtr(int32 size) возвращает указатель на распределенный блок памяти длиной **size** байтов.

UPtr DSNewPClr(int32 size) аналогично **DSNewPtr**, но память предварительно очищается.

MgErrDSDisposePtr(Uptr p) освобождает блок памяти по адресу **p**.

3. Функции для работы с файлами

MgErrFCreate(File *fdp, Path path, int32 permissions, int32 openMode, int32 denyMode, PStr group) - создает отсутствующий файл с именем **path**, атрибутами доступа **permissions** (если бит 7 сброшен, то после закрытия у файла будет установлен атрибут “только для чтения”), режимом доступа со стороны текущего процесса **openMode** (openReadOnly - запись в файл невозможна, openWriteOnly - чтение из файла невозможно, openReadWrite - разрешены чтение и запись)**,** режимом доступа со стороны других процессов **denyMode** (denyReadWrite - чтение и запись запрещены, denyWriteOnly - запись запрещена, denyNeither - полный доступ), UNUX-группой **group** (в Windows игнорируется) и возвращает дескриптор в **fdp**.

MgErrFCreateAlways(File *fdp, Path path, int32 permissions, int32 openMode, int32 denyMode, PStr group) - аналогично **MgErrFCreate**, но если файл уже существует, то он очищается.

MgErrFMOpen(File *fdp, Path path, int32 openMode, int32 denyMode) - открывает файл с именем **path**, режимом доступа со стороны текущего процесса **openMode**, режимом доступа со стороны других процессов **denyMode** и возвращает дескриптор в **fdp**.

MgErrFMClose(File *fd) - закрывает файл с дескриптором **fd**.

MgErrFMRead(File *fd, int32 inCount, int32 *outCountp, uPtr buffer) - читает **inCount** байтов из файла с дескриптором **fd** в буфер **buffer**, возвращая число реально прочитанных байтов в **outCountp**.

MgErrFMWrite(File *fd, int32 inCount, int32 *outCountp, uPtr buffer) - записывает **inCount** байтов в файл с дескриптором **fd** из буфера **buffer**, возвращая число реально записанных байтов в **outCountp**.

MgErrFMSeek(File fd, int32 ofst, int32 mode) - перемещает указатель чтения/записи в файле с дескриптором **fd** на позицию **ofst** в соответствии с режимом **mode** (fStart - относительно начала файла, fCurrent - относительно текущей позиции, fEnd - относительно конца файла).

MgErr FMTell(File fd, int32 *ofstp) - возвращает для файла с дескриптором **fd** текущую позицию чтения/записи в **ofstp**.

MgErrFGetEOF(File fd, int32 *sizep) - возвращает в **sizep** текущую длину файла с дескриптором **fd**.

MgErrFSetEOF(File fd, int32 size) - устанавливает для файла с дескриптором **fd** новую длину **size**.

MgErrFMove(Path oldPath, Path newPath) - присваивает файлу с именем **oldPath** новое имя **newPath** (возможно, изменяя его местоположение на носителе).

MgErrFRemove(Path path) - удаляет файл с именем **path**.

int32 FExists(Path path) - проверяет существование файла с именем **path**.

4. Функции для работы со временем

uInt32 MilliSecs() - возвращает текущее количество миллисекунд.

uInt32 TimeInSecs() - возвращает количество секунд, прошедшее с 0 часов 0 минут 0 секунд 1.1.1904 г.

CStr DateString(uInt32 secs, int32 fmt) - возвращает указатель на текстовую строку, описывающую дату, определяемую числом секунд **secs**, прошедших с 0 часов 0 минут 0 секунд 1.1.1904 г., в соответствии с форматом **fmt**: 0 - “mm/dd/yy”, 1 - со включением полных наименований дня недели и месяца; 2 - со включением сокращенных наименований.

CStr TimeString(uInt32 secs, int32 fmt) - возвращает указатель на текстовую строку, описывающую время суток, определяемое числом секунд **secs**, прошедших с 0 часов 0 минут 0 секунд 1.1.1904 г., в соответствии с форматом **fmt**: 0 - “hh:mm”, 1- “hh:mm:ss”.

CStr ASCIITime(uInt32 secs) - возвращает указатель на текстовую строку, описывающую дату и время суток, определяемые числом секунд **secs**, прошедших с 0 часов 0 минут 0 секунд 1.1.1904 г.

5. Функции и макросы для работы с символами

boolean IsDigit(uChar c) - возвращает результат проверки символа **c** на принадлежность множеству цифр.

boolean IsAlpha(uChar c) - возвращает результат проверки символа **c** на принадлежность множеству букв латинского алфавита.

boolean IsUpper(uChar c) - возвращает результат проверки символа **c** на принадлежность множеству прописных букв латинского алфавита.

boolean IsLower(uChar c) - возвращает результат проверки символа **c** на принадлежность множеству строчных букв латинского алфавита.

uChar ToUpper(uChar c) - возвращает результат преобразования строчной буквы латинского алфавита в прописную.

uChar ToLower(uChar c) - возвращает результат преобразования прописной буквы латинского алфавита в строчную.

int32 HexChar(int32 n) - возвращает код символа 16-ричной цифры, соответствующей числу **n**.

6. Функции для работы со строками

int32 StrLen(CStr s) - возвращает длину строки **s** в байтах.

int32 StrCat(CStr s1, CStr s2) - добавляет строку **s2** в конец строки **s1**, возвращая новую длину строки **s1**.

int32 StrNCmp(CStr s1, CStr s2, int32 n) - возвращает результат лексикографического сравнения **n** первых символов строк **s1** и **s2** (например, если подстроки совпадают, то функция возвращает 0).

int32 StrNCaseCmp(CStr s1, CStr s2, uInt32 n) - работает аналогично **StrNCmp**, но различие строчных и прописных букв латинского алфавита игнорируется.

Примечание. Для функций **StrLen**, **StrCat**, **StrNCmp** и **StrNCase** в библиотеке имеются функциональные аналоги, работающие с другими типами строк. Например, функция **PStrLen** манипулирует строками типа **PStr**, **LStrLen** - строками типа **LStr**, а **CPStrLen** - строками типа **CPStr**.

PtoCStr(PStr pstr, CStr cstr) - преобразует строку типа **PStr** в строку типа **CStr**.

CToPStr(CStr cstr, PStr pstr) - преобразует строку типа **CStr** в строку типа **PStr**.

PToLStr(PStr pstr, LStrPtr lstrp) - преобразует строку типа **PStr** в строку типа **LStr**.

LToPStr(LStrPtr lstrp, PStr pstr) - преобразует строку типа **LStr** в строку типа **PStr**.

7. Математические функции и макросы

int32 Max(int32 n, int32 m) - возвращает большее из чисел **n** и **m**.

int32 Min(int32 n, int32 m) - возвращает меньшее из чисел **n** и **m**.

int32 Pin(int32 i, int32 low, int32 high) - возвращает значение **low** если **i**<**low**, значение **high**, если **i**>**high,** и значение **i** в противном случае.

void RandomGen(float64 *xp) - возвращает в **xp** значение псевдослучайной величины, равномерно распределенной на интервале [0,1].

Примечание. Кроме того разрешено использование следующих стандартных для языка Си математических функций: **atan()**, **cos()**, **exp()**, **fabs()**, **log()**, **sin()**, **sqrt()**, **tan()**, **acos()**, **asin()**, **atan2()**, **ceil()**, **cosh()**, **floor()**, **fmod()**, **frexp()**, **ldexp()**, **log10()**, **modf()**, **pow()**, **sinh()**, **tanh()**.

8. Функции и макросы для работы с переменными и памятью

int16 HiWord(int32 x) - возвращает старшее слово 32-битовой переменной **x**.

int16 LoWord(int32 x) - возвращает младшее слово 32-битовой переменной **x**.

int8 HiByte(int16 x) - возвращает старший байт 16-битовой переменной **x**.

int8 LoByte(int16 x) - возвращает младший байт 16-битовой переменной **x**.

uInt8 HiNibble(uInt8 x) - возвращает старшие 4 бита 8-битовой переменной **x**.

uInt8 LoNibble(uInt8 x) - возвращает младшие 4 бита 8-битовой переменной **x**.

int16 Word(int8 hi, int8 lo) - компонует 16-битовое слово из байтов **hi** и **lo**.

Int32 Long(int16 hi, int16 lo) - компонует 32-битовое двойное слово из 16-битовых слов **hi** и **lo**.

int32 Cat4Chrs(uInt8 a, uInt8 b, uInt8 c, uInt8 d) - компонует 32-битово двойное слово из байтов **a**, **b**, **c** и **d**.

int16 Offset(type, field) - возвращает байтовое смещение поля **field** в сложной структуре **type**.

9. Функции и макросы общего назначения

int32 BinSearch(arrayp, n, elmtSize, key, compareProcP()) - выполняет двоичный поиск по ключу **key** в массиве **arrayp**, каждый из **n** элементов которого имеет размер в **elmtSize** байтов, причем правило сравнения элемента с ключом задается при помощи определяемой пользователем функции **compareProcP**.

void QSort(arrayp, n, elmtSize, compareProcP()) - выполняет быструю сортировку (Дейкстры) массива **arrayp**, каждый из **n** элементов которого имеет размер в **elmtSize** байтов, причем правило сравнения элементов задается при помощи определяемой пользователем функции **compareProcP**.

DbgPrintf(uChar *formats, ...) - открывает дополнительное окно и отображает в нем отладочные сообщения пользователя, организованные по правилам стандартной для языка Си функции **printf**.