

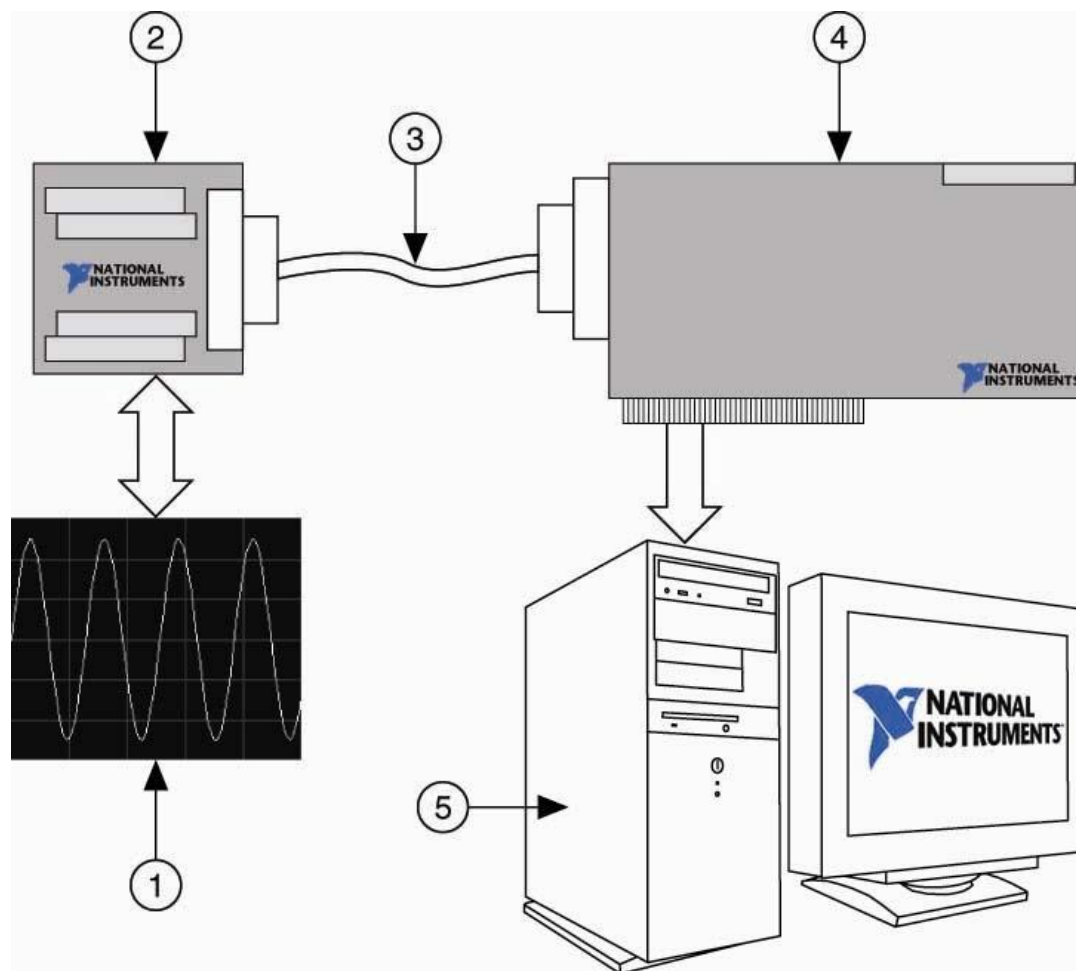
Цели курса

Этот курс подготовит вас для :

- Решения задач с помощью LabVIEW
- Применения средств сбора данных и управления измерительными приборами в приложениях LabVIEW
- Использования опыта модульного программирования
- Разработки, отладки и тестирования LabVIEW VI
- Эффективного использования архитектуры конечных автоматов
- Использования свойств параллелизма и переменных

А. Оборудование для сбора данных (DAQ)

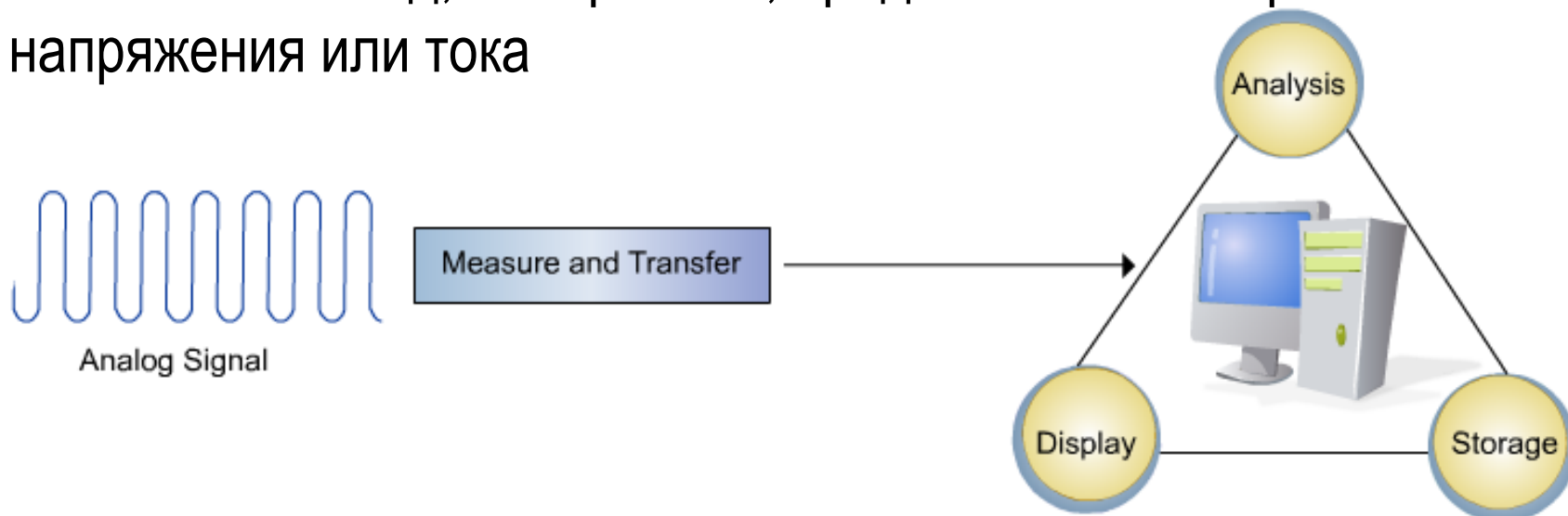
1. Сигнал
2. Коннекторный блок
3. Кабель
4. DAQ устройство
5. Компьютер



Оборудование DAQ – Аналоговый ввод

Процесс измерения аналогового сигнала и передачи результата измерения в компьютер для обработки, визуализации и запоминания

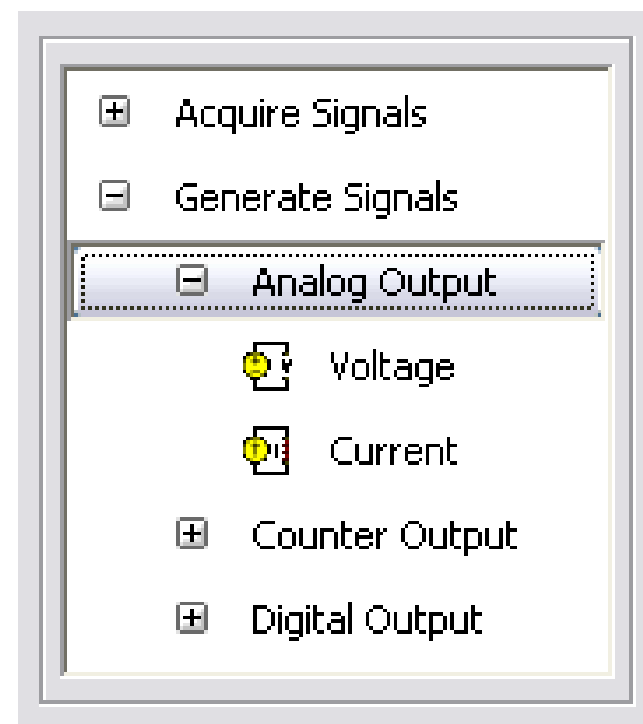
- Сигнал, изменяющийся непрерывно, называется аналоговым
- Аналоговый ввод, как правило, предполагает измерение напряжения или тока



Оборудование DAQ – Аналоговый вывод

Процесс генерации аналогового сигнала с помощью компьютера

- При генерации сигнала выполняется цифро-аналоговое преобразование
- Доступные типы выводимых аналоговых сигналов – напряжение и ток
- Для вывода напряжения или тока должно быть установлено устройство, способное формировать эти типы сигналов



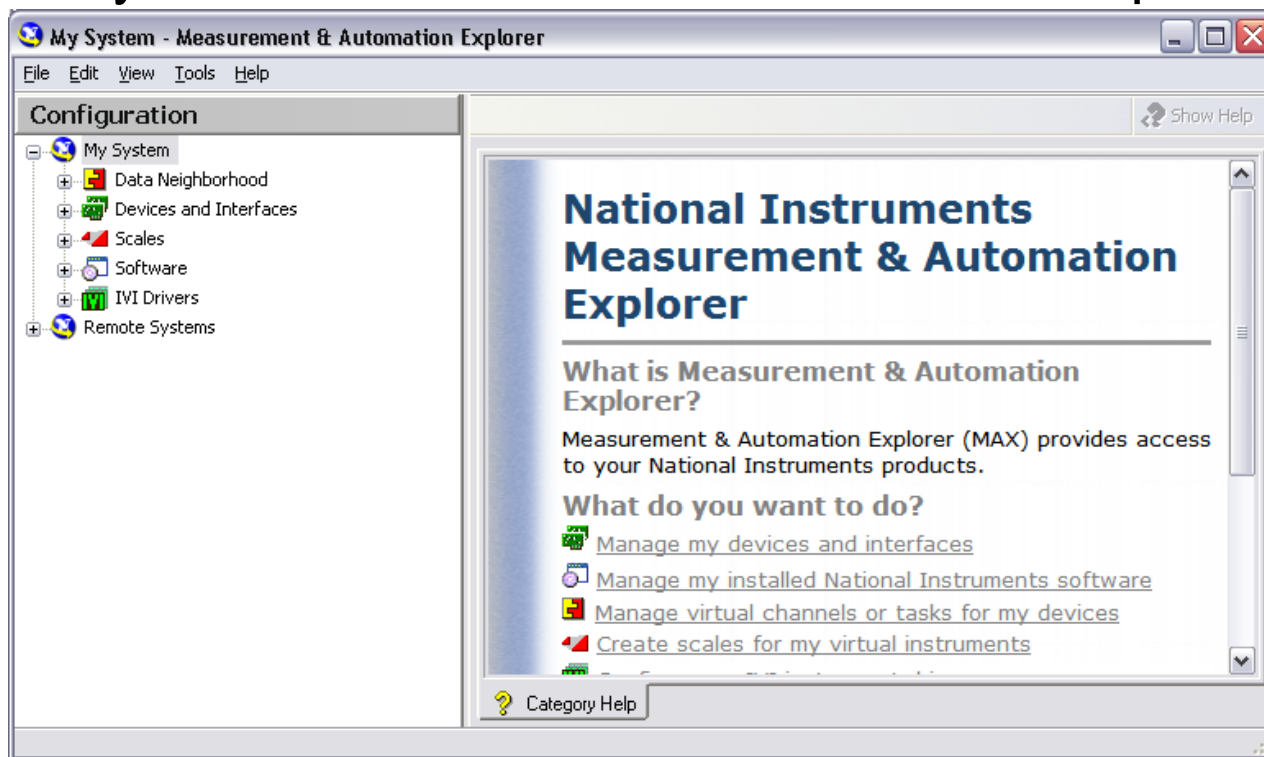
Оборудование DAQ – Цифровой ввод-вывод

- Цифровые сигналы:
 - Электрические сигналы, с помощью которых по проводам передаются цифровые данные (вкл/выкл, высокий/низкий, 1/0)
 - Для формирования или считывания цифровых сигналов применяются цифровые устройства или устройства с конечным числом состояний, такие, как переключатели или светодиоды
 - Для передачи данных используются
 - программируемые устройства
 - коммуникации между устройствами
 - Цифровые сигналы используют как тактовые импульсы или сигналы запуска для управления или синхронизации других видов измерений



В. Программное обеспечение DAQ – Конфигурирование

- Для конфигурирования и тестирования DAQ устройств используется Measurement & Automation Explorer (MAX)

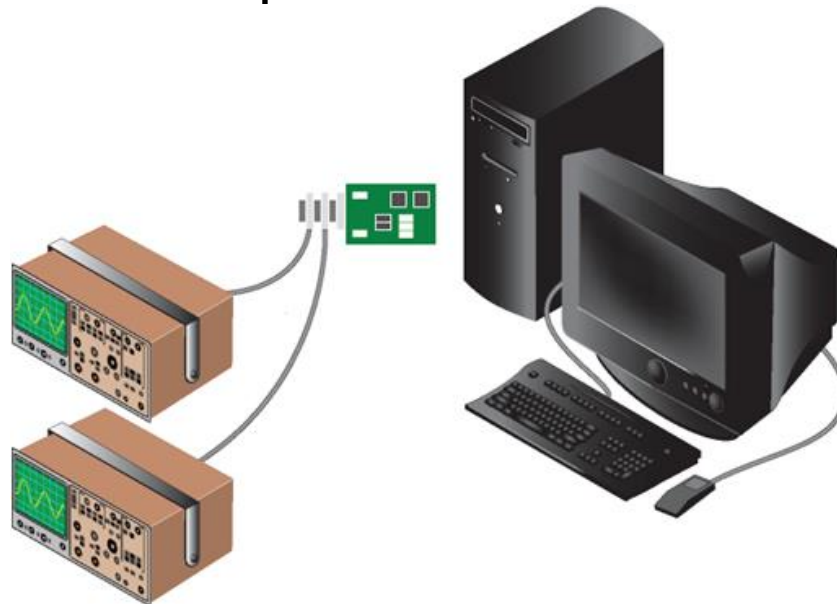


Симуляция DAQ устройств

- Используя симулируемые NI-DAQmx устройства, вы можете опробовать продукцию NI в ваших приложениях без оборудования
- Используя симулируемые NI-DAQmx устройства, вы можете также экспортировать конфигурацию физического устройства в системы, не содержащие физических устройств

С. Управление измерительными приборами

- ПО персонального компьютера используется для управления измерительным прибором через шину управления
- Применяйте и сравнивайте приборы разных категорий
- Разберите свойства приборов, такие, как используемые коммуникационные протоколы



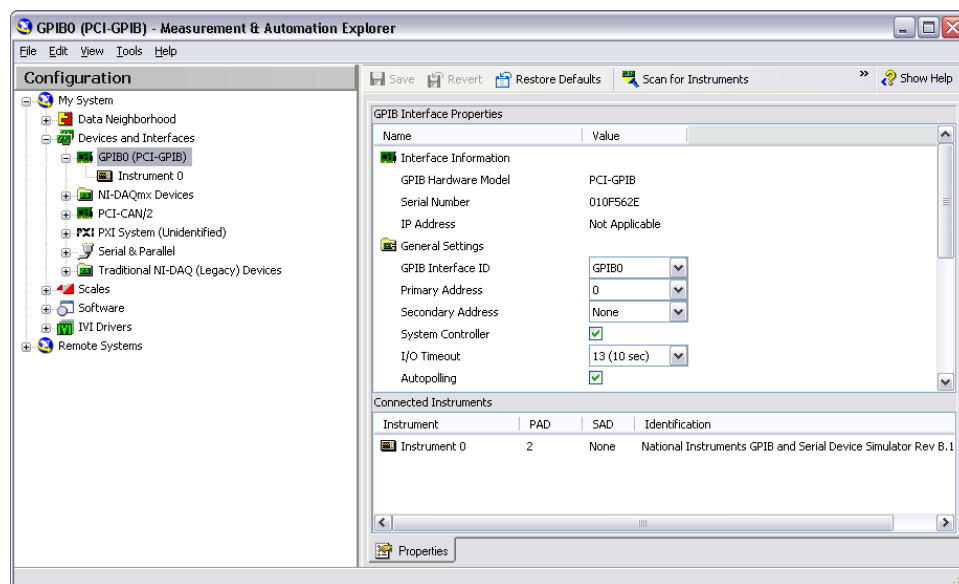
С. Управление измерительными приборами

Достоинства управления измерительными приборами

- Автоматизация процессов
- Экономия времени
- Одна платформа для различных задач
- Простота использования
- Доступны многие типы измерительных приборов

Г. ПО для управления измерительными приборами

- Интерфейсные драйверы: интерфейсы измерительных приборов, как, например, GPIB, включают набор драйверов интерфейса
- Конфигурирование: для конфигурирования интерфейса используется MAX



Заключение – Контрольный вопрос

2. Какие преимущества предоставляет управление измерительными приборами?
- a) Автоматизация процессов
 - b) Экономия времени
 - c) Применение одной платформы для решения многих задач
 - d) Ограничение только одним типом измерительных приборов

Заключение – Ответ на контрольный вопрос

2. Какие преимущества предоставляет управление измерительными приборами?
- a) Автоматизация процессов**
 - b) Экономия времени**
 - c) Применение одной платформы для решения многих задач**
 - d) Ограничение только одним типом измерительных приборов

Лекция 2

Ориентация в LabVIEW

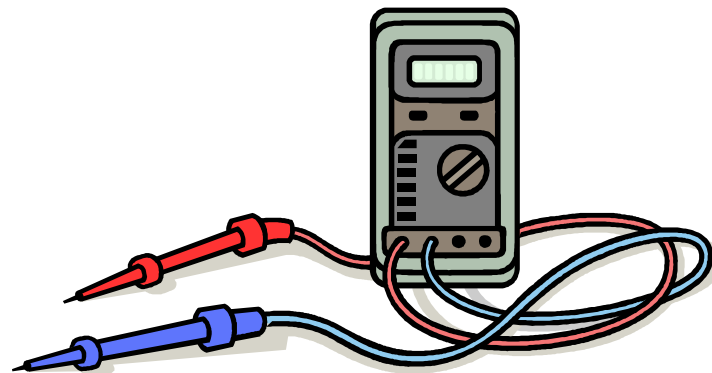
ТЕМЫ

- A. Виртуальные измерительные приборы (VI)
- B. Состав VI
- C. Начинаем проектировать VI
- D. Навигатор по проекту (Project Explorer)
- E. Лицевая панель (Front Panel)
- F. Блок-диаграмма (Block Diagram)
- G. Поиск элементов управления и индикации, VI и функций
- H. Выбор инструмента
- I. Потокное программирование
- J. Разработка простого VI

А. Виртуальные измерительные приборы (VI)

Виртуальный измерительный прибор (VI) – программа в LabVIEW

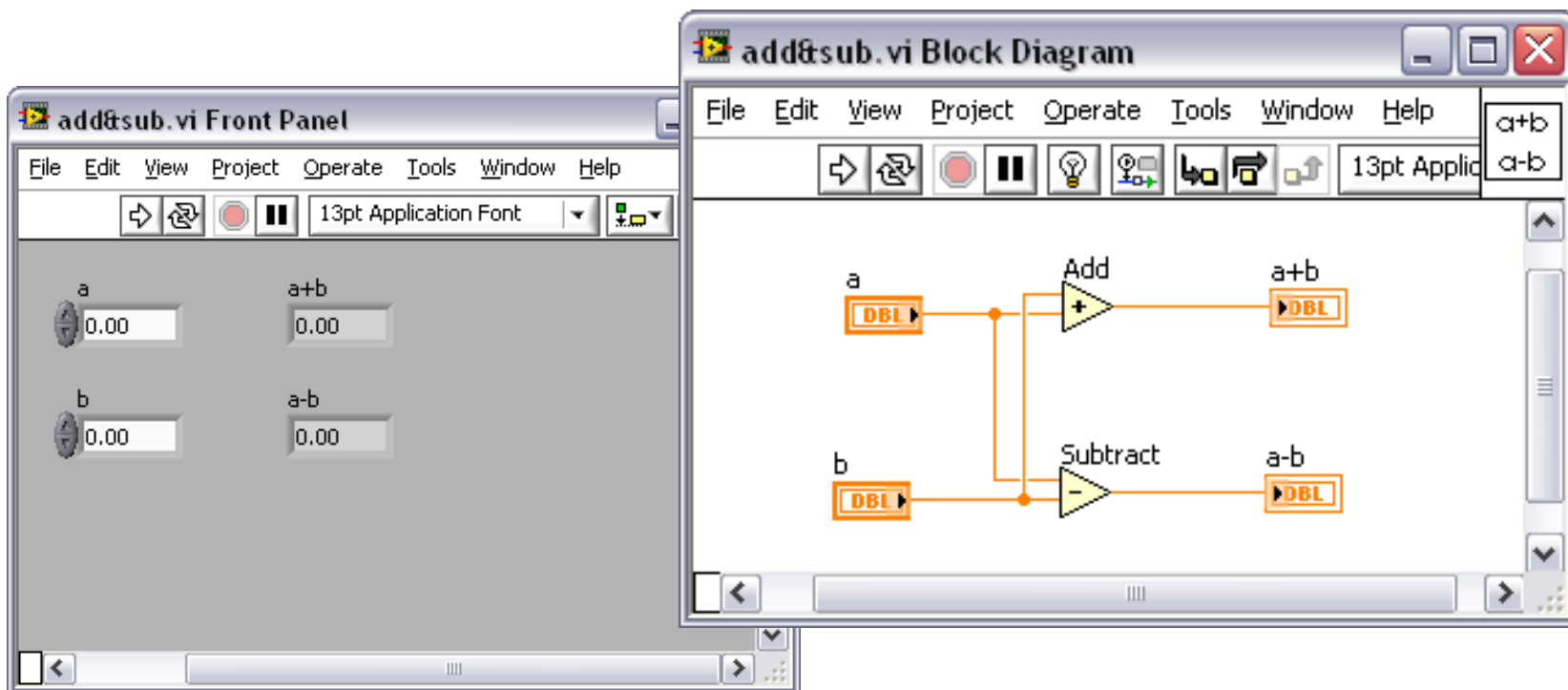
Внешний вид и функционирование VI имитирует настоящие (реальные) приборы, такие, например, как осциллографы и цифровые мультиметры.



В. Состав VI

LabVIEW VI состоят из трех основных компонентов:

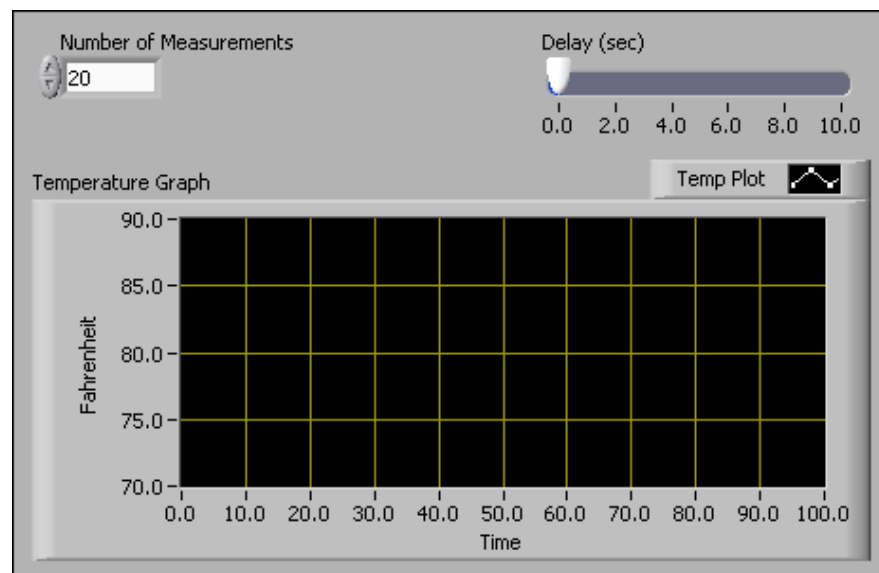
1. Лицевой панели
2. Блок-диаграммы
3. Иконки/Панели подключения



В. Состав VI – Front Panel

Front Panel – лицевая панель - интерфейс пользователя VI

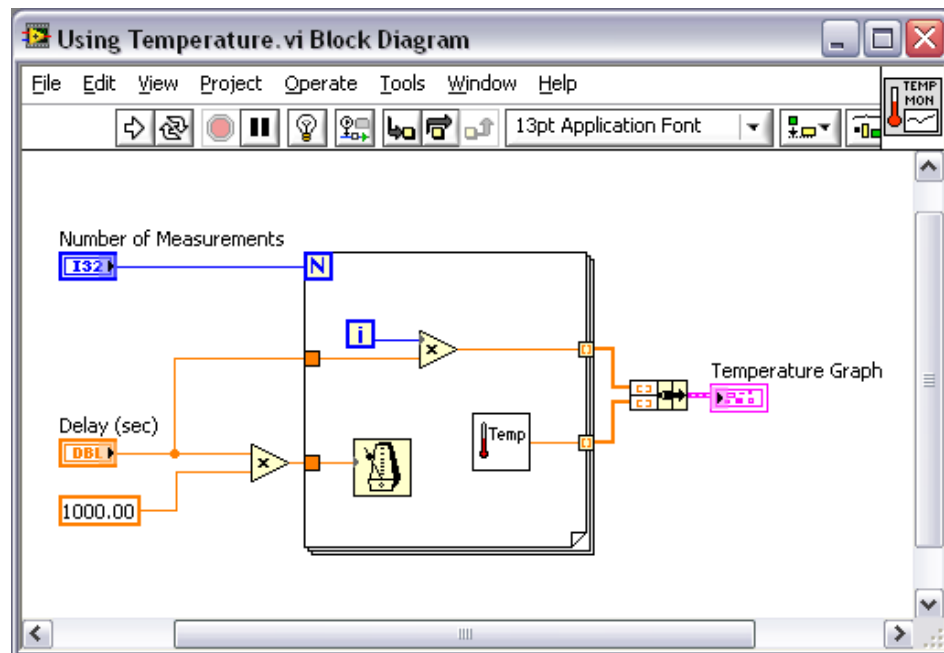
Лицевая панель состоит из элементов управления (входы) и элементов индикации (выходы)



В. Состав VI – Block Diagram

Block Diagram – блок-диаграмма - содержит исходный код в графическом формате

Объекты лицевой панели отображаются на блок-диаграмме в виде терминалов



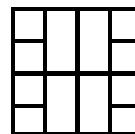
В. Состав VI – Icon/Connector Pane

- Иконка (Icon): графическое представление VI
- Панель подключения (Connector Pane): схема входов и выходов VI
- Иконка и панель подключения необходимы для использования VI в качестве subVI
 - subVI – это VI внутри других VI
 - Аналоги функциям в текстовых языках программирования

Icon



Connector Pane



С. Начинаем проектировать VI



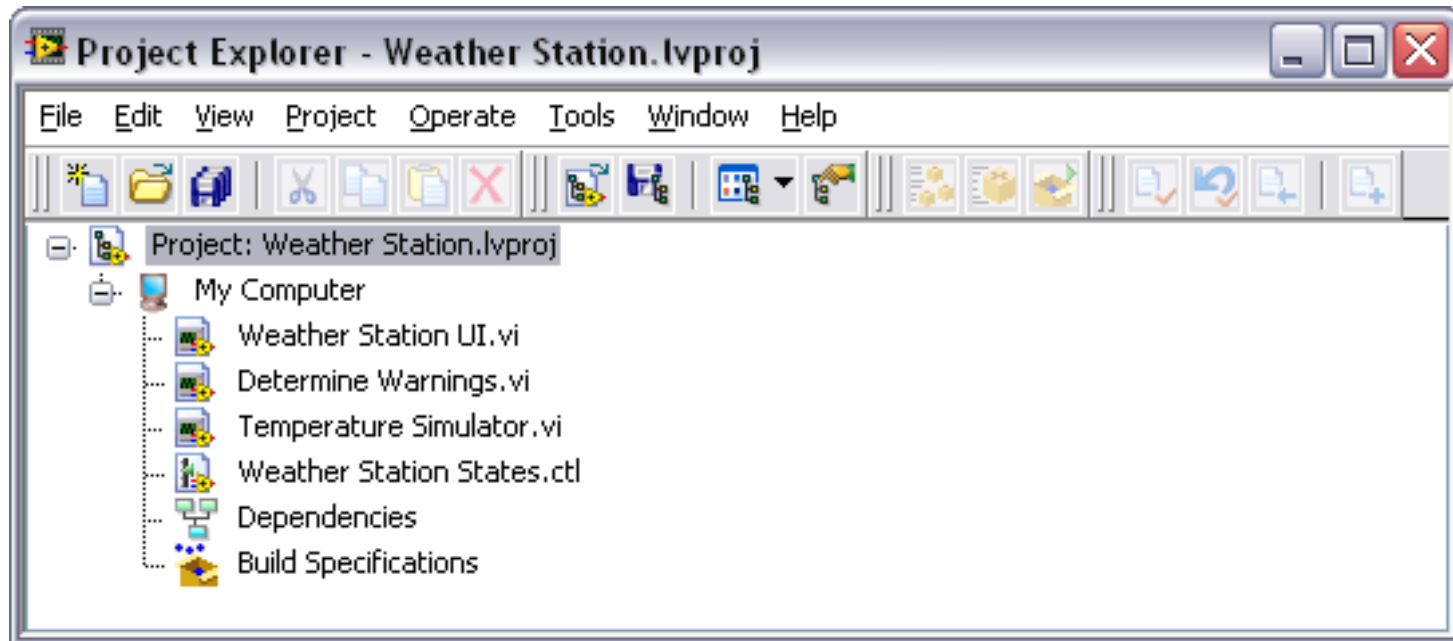
Демонстрируются диалоговые окна **Getting Started** и **New**, используемые в начале проектирования VI.

ДЕМОНСТРАЦИЯ

D. Project Explorer – навигатор по проекту

Проекты LabVIEW используются для:

- Группировки файлов LabVIEW и файлов других типов
- Создания спецификации для сборки
- Развертывания или загрузки файлов в целевые устройства



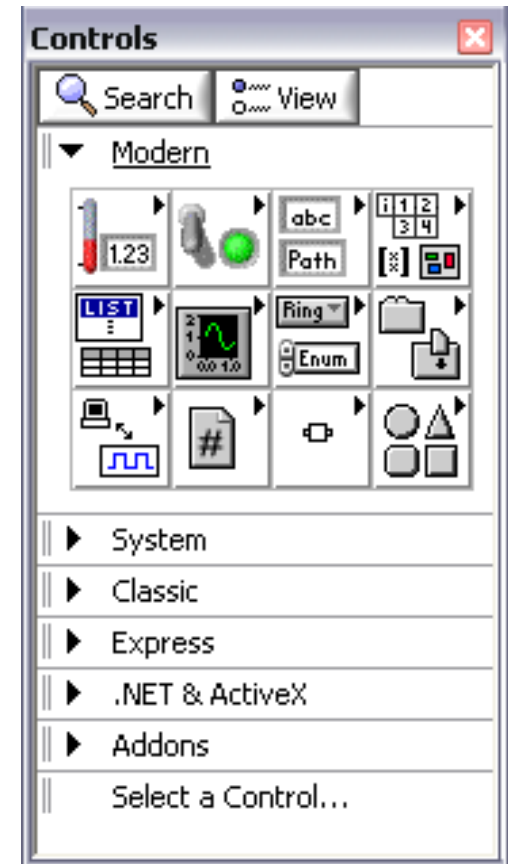
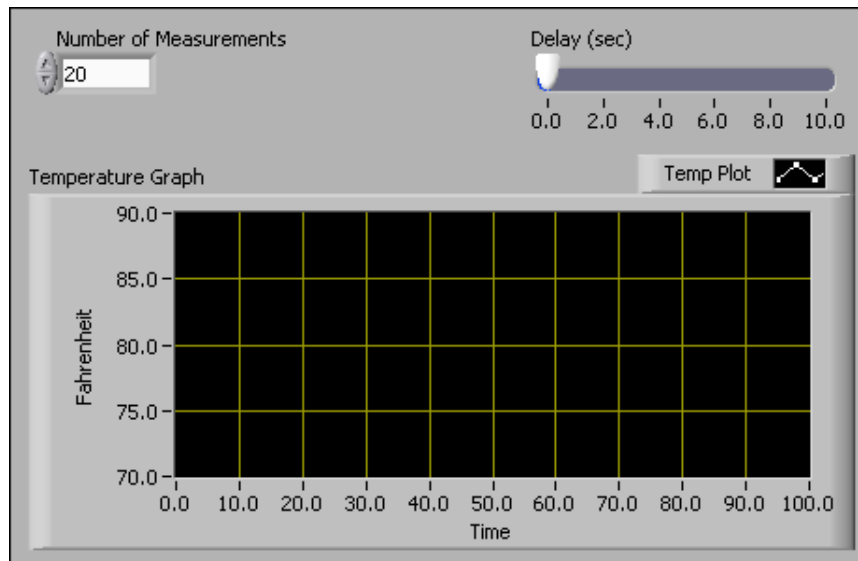
Работа с Project Explorer

Демонстрируются создание проекта, добавление файлов в проект и удаление их из проекта.

ДЕМОНСТРАЦИЯ

E. Front Panel – Палитра Controls

- Содержит элементы управления и индикации, используемые при проектировании лицевой панели
- Из лицевой панели палитру можно открыть, выбрав **View»Controls Palette**



Е. Front Panel – Панель инструментов лицевой панели

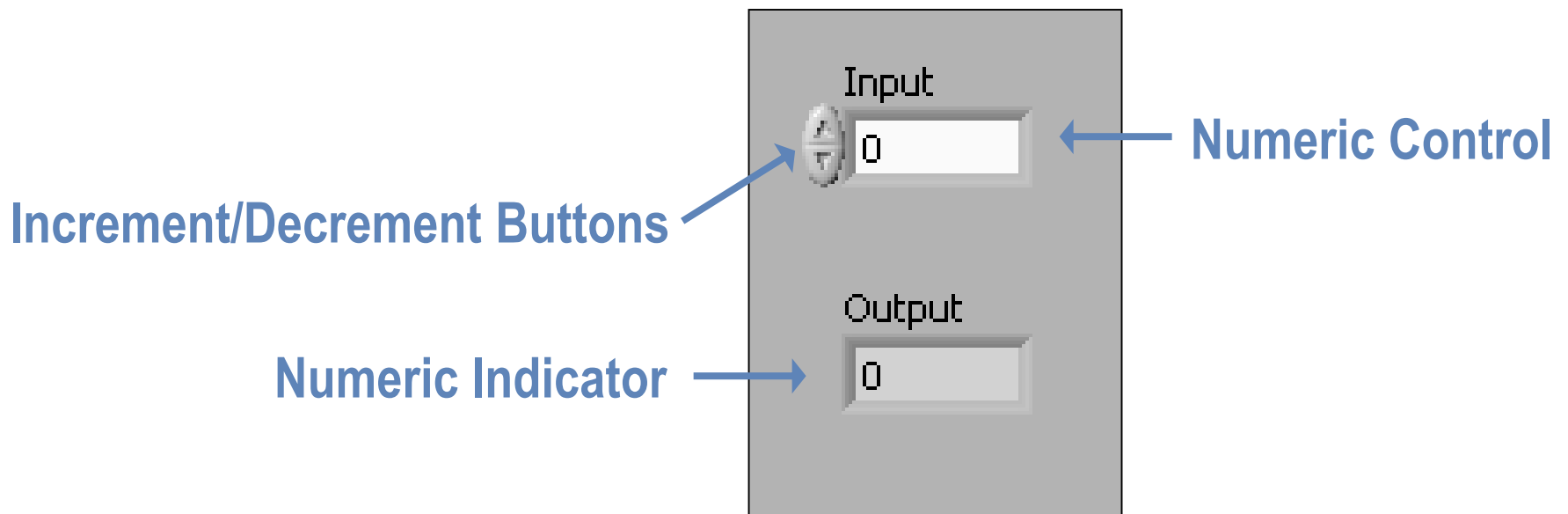


Е. Front Panel – Элементы управления и индикации

- Controls
 - Регуляторы, кнопки, лимбы и другие устройства ввода
 - Симулируют устройства ввода измерительного прибора и являются источниками данных для блок-диаграммы VI
- Indicators
 - Графические экраны, светодиоды и другие элементы индикации
 - Симулируют устройства вывода измерительного прибора и отображают данные, собранные или сгенерированные на блок-диаграмме

E. Front Panel – Числовые элементы управления и индикации

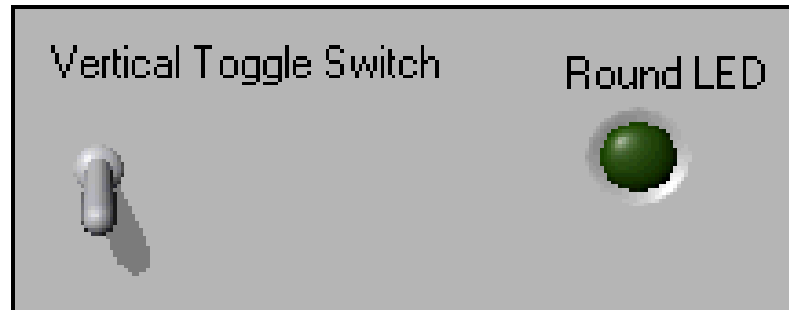
Числовой тип данных представляет различные данные, такие, например, как целые или действительные



Е. Front Panel – Булевские элементы управления и индикации

- Булевский тип представляет данные, имеющие только два значения - True и False или On и Off
- Используйте булевские элементы управления и индикации для ввода и визуализации булевских значений (True или False)
- Булевские объекты симулируют ключи, кнопки и светодиоды

Boolean
Control

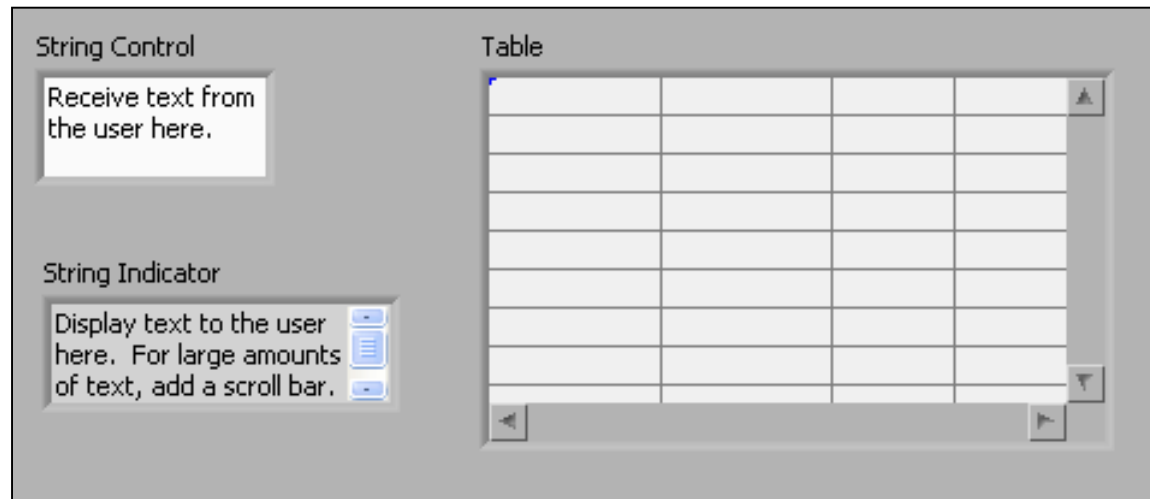


Boolean
Indicator



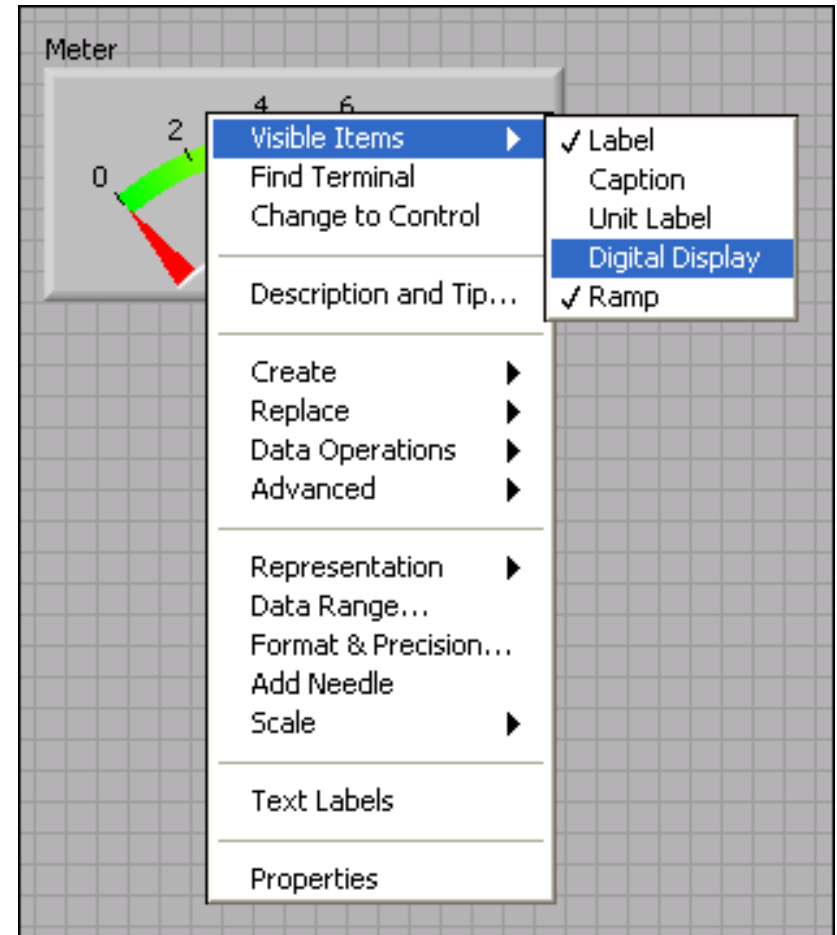
E. Front Panel – Строки

- Строковый тип данных представляет последовательность символов ASCII
- Строковые элементы управления используются для ввода пользователем текста – пароля или имени пользователя
- На строковых индикаторах отображается текст для пользователя



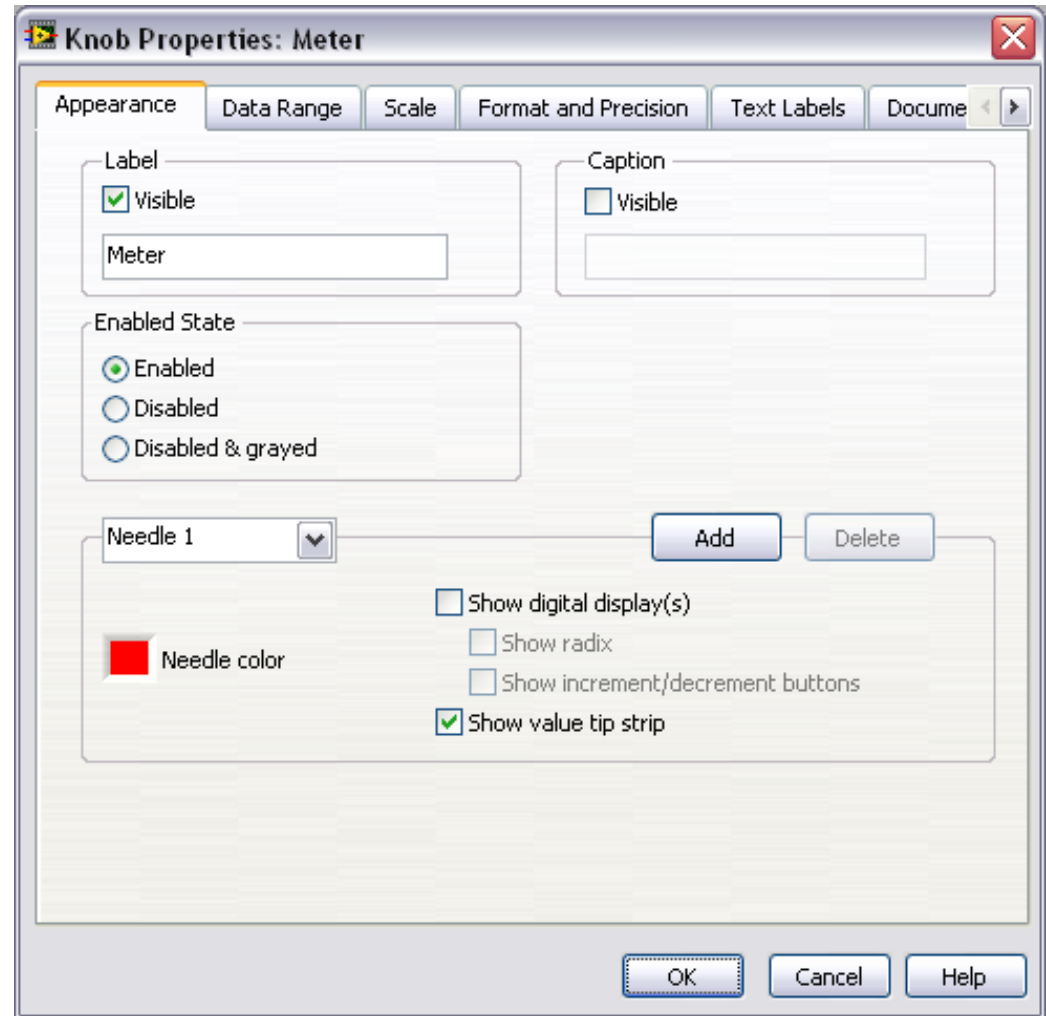
E. Front Panel – Контекстное меню

- Все объекты LabVIEW имеют контекстное меню
- Создавая VI, пользуйтесь элементами контекстного меню для изменения внешнего вида или свойств объектов лицевой панели и блок-диаграммы
- Контекстное меню открывается щелчком правой кнопки мыши по объекту



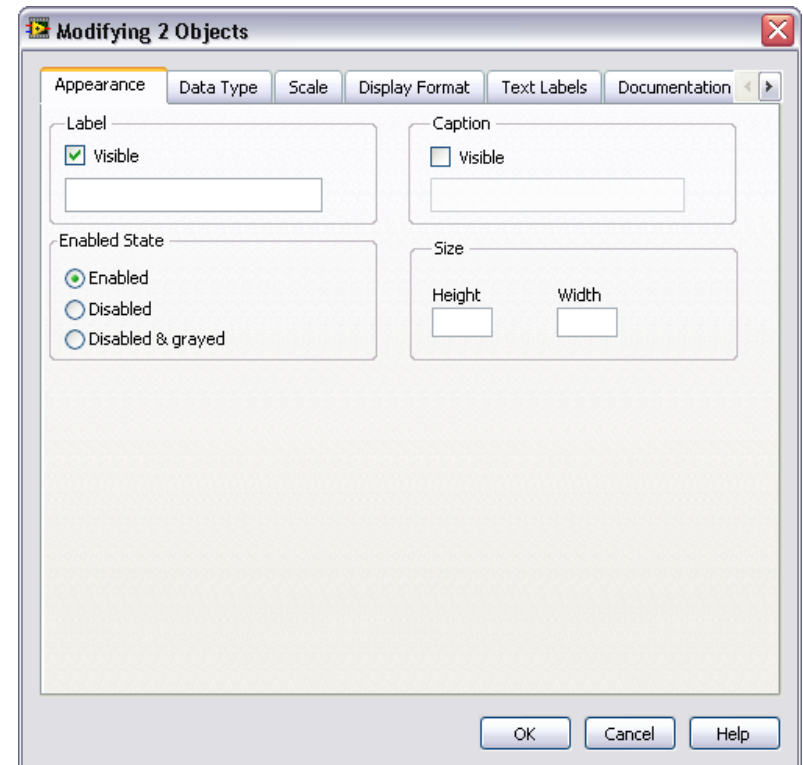
E. Front Panel – Диалоговое окно свойств

- Щелкните правой кнопкой мыши по объекту лицевой панели и выберите Properties (свойства)
- Возможные варианты в диалоговом окне свойств аналогичны вариантам, доступным из контекстного меню этого же объекта

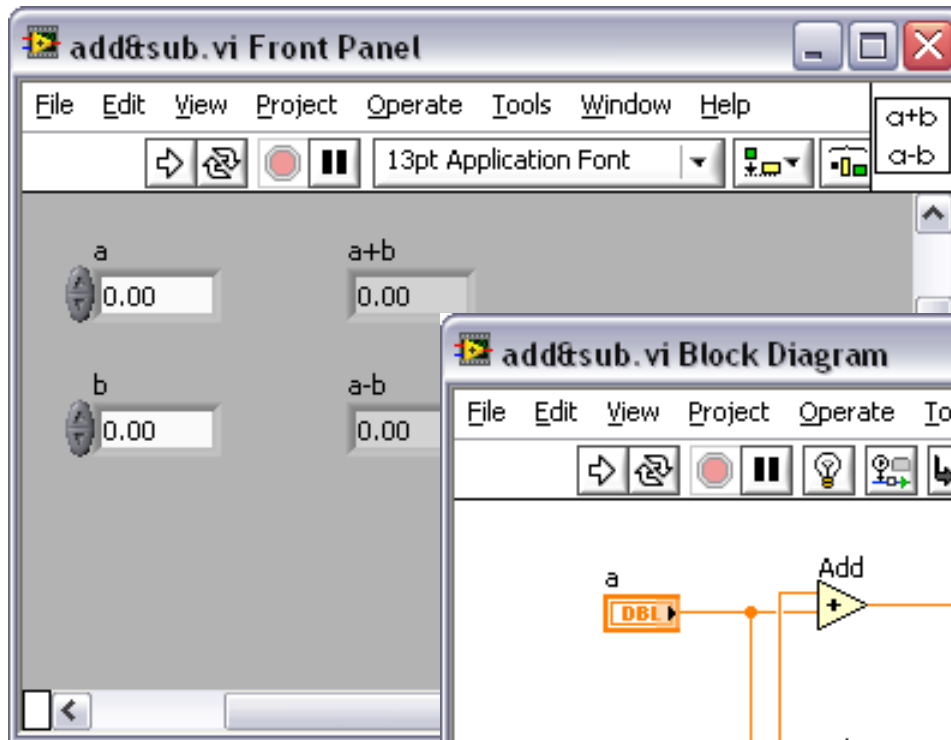


E. Front Panel – Конфигурирование нескольких объектов

- Выберите несколько объектов для одновременного конфигурирования их общих свойств

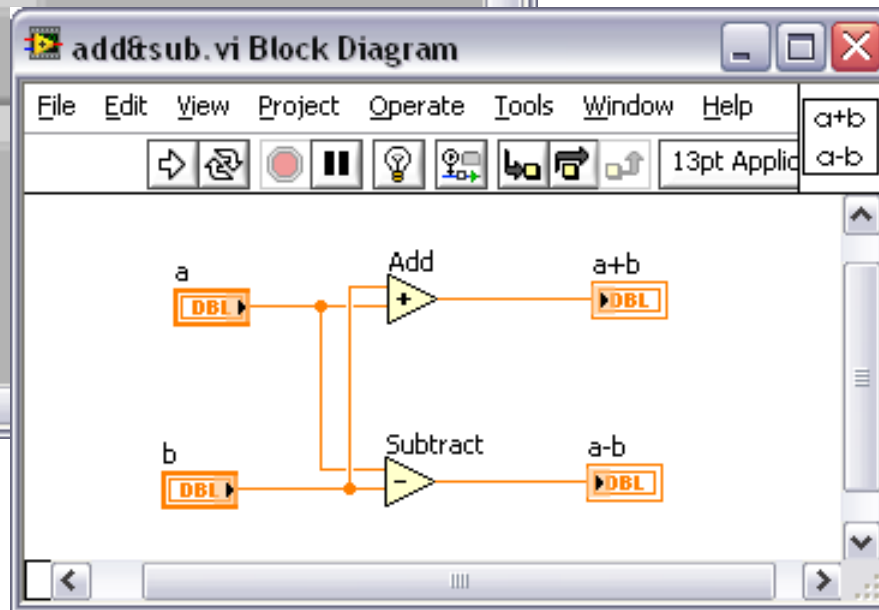


F. Block Diagram



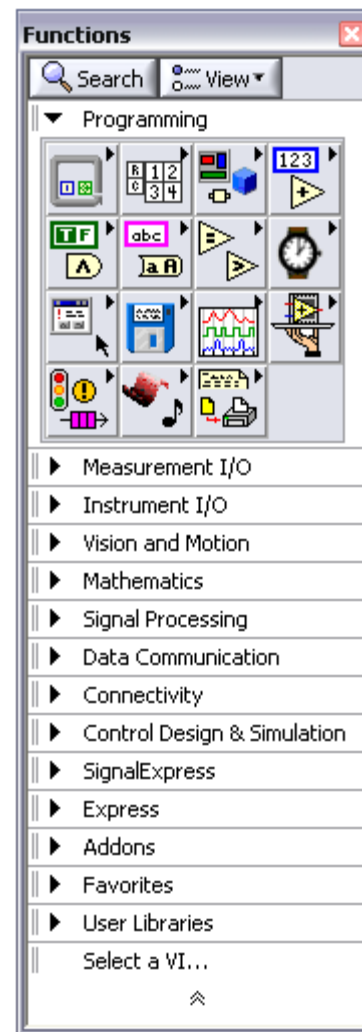
На блок-диаграмме находятся следующие объекты:

- Терминалы
- SubVI
- Функции
- Константы
- Структуры
- Проводники



F. Block Diagram – Палитра функций

Содержит VI, функции и константы, используемые при проектировании блок-диаграммы



F. Block Diagram – Панель инструментов блок-диаграммы



F. Block Diagram – Терминалы

- Терминалы - это:
 - Представление объектов лицевой панели на блок-диаграмме
 - Порты ввода и вывода, через которые осуществляется обмен информацией между лицевой панелью и блок-диаграммой
 - Аналогии параметров и констант в текстовых языках программирования
- Внешний вид терминалов можно изменить, выбрав и переключив пункт **View as Icon** контекстного меню

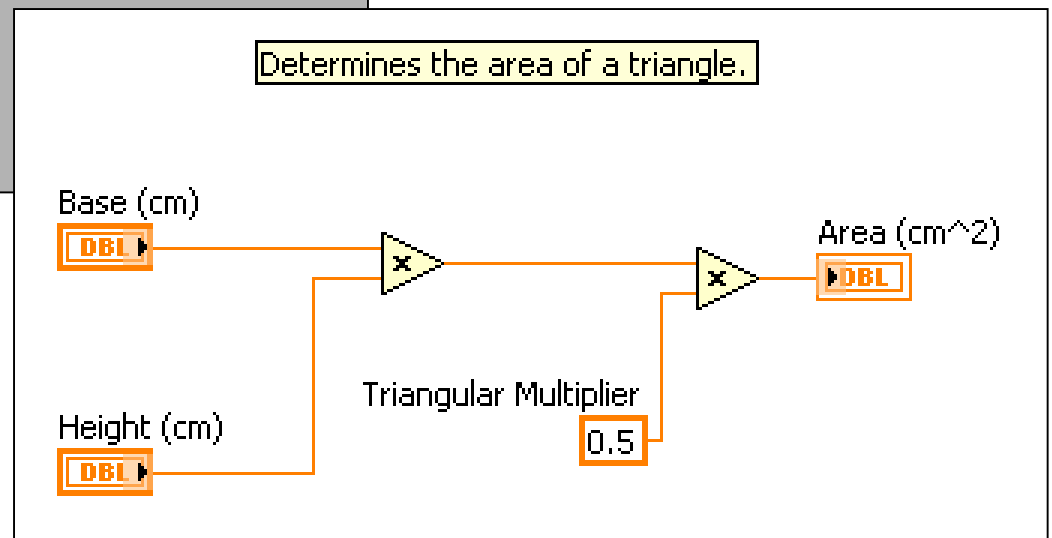
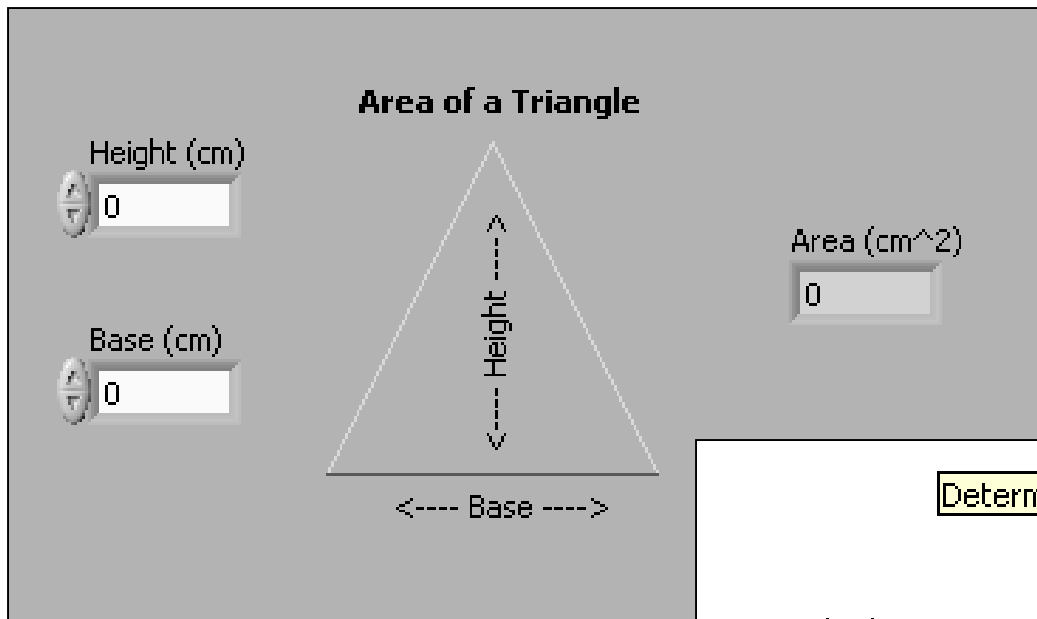
Input



Input

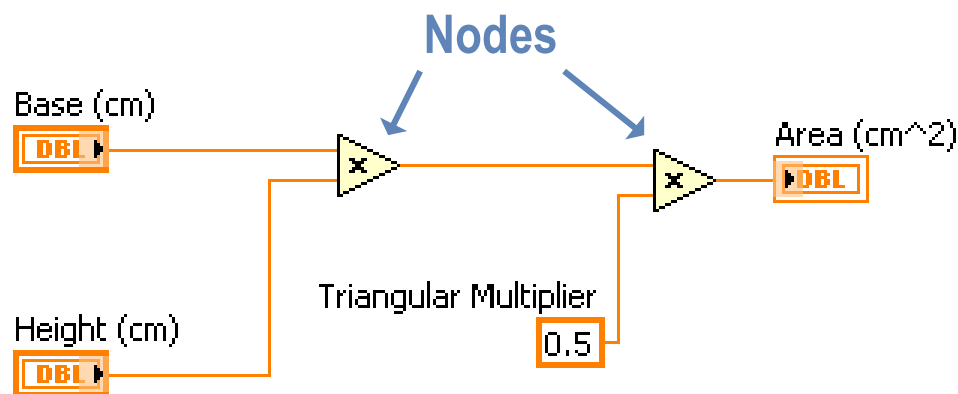


Г. Терминалы блок-диаграммы

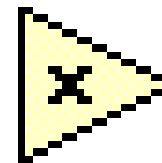


F. Block Diagram – Узлы

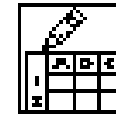
- Объекты блок-диаграммы, у которых есть входы и/или выходы, и которые выполняют операции при запуске VI
- Аналоги высказываний, операторов, функций и подпрограмм в текстовых языках программирования
- Узлами могут быть функции, subVI или структуры



F. Block Diagram – Узлы функций



- Базовые операционные элементы LabVIEW
- Не имеют лицевой панели или блок-диаграммы, но имеют панель подключения
- Двойной щелчок только выделяет функцию, но не раскрывает ее, как в VI
- Фон иконки – бледно-желтый

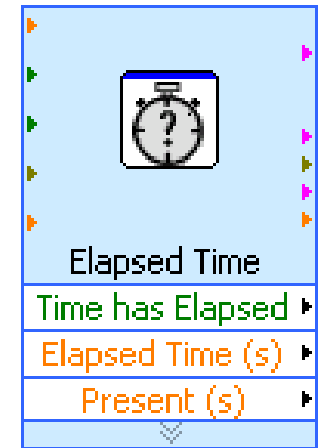


F. Block Diagram – Узлы SubVI

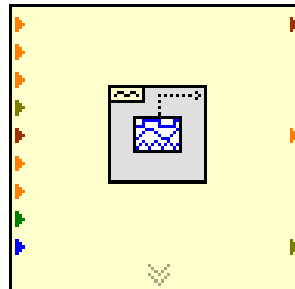
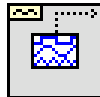
- SubVI – это VI, которые создаются для использования внутри других VI
- Любой VI потенциально может быть использован в качестве subVI
- Если щелкнуть дважды по subVI на блок-диаграмме, то можно увидеть лицевую панель и блок-диаграмму subVI
 - В верхнем правом углу лицевой панели находится иконка текущего VI
 - Эта иконка и появляется на блок-диаграмме, когда VI помещается на блок-диаграмму в качестве subVI

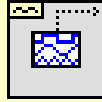
F. Block Diagram – Узлы SubVI

- Express VI – специальный тип subVI
 - Требуют минимума соединений, поскольку их конфигурируют с помощью диалоговых окон
 - Конфигурацию Express VI можно сохранить, как subVI
- Иконки Express VI на блок-диаграмме окружены голубым полем



F. Block Diagram – Иконки/Расширяемые узлы



	
▶	amplitude
▶	error in (no error)
▶	frequency
▶	offset
▶	phase
▶	reset signal
▶	sampling info
▶	signal type
▶	square wave duty
	error out ▶
	phase out ▶
	signal out ▶

F. Block Diagram – Проводники

- Данные между объектами блок-диаграммы передаются по проводникам
- Проводники имеют разный стиль, цвет и толщину, которые зависят от типа данных
- Оборванный проводник выглядит, как черная пунктирная линия с красным X посередине



	DBL Numeric	Integer Numeric	String
Scalar			
1D Array			
2D Array			

F. Block Diagram – Советы для соединений

- Нажмите <Ctrl>-B, чтобы удалить все разорванные проводники
- Щелкните правой кнопкой мыши и выберите **Clean Up Wire** для изменения маршрута, по которому проходит проводник

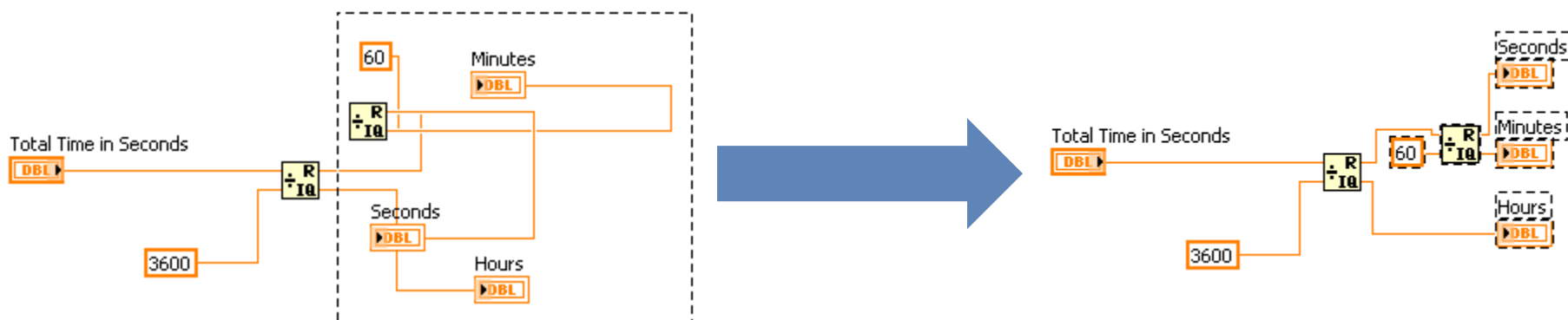


F. Block Diagram – Советы для соединений



Используйте инструмент Clean Up Diagram (привести в порядок диаграмму) для упорядочения проводников и объектов с целью улучшения читаемости

1. Выделите фрагмент блок-диаграммы
2. Щелкните по кнопке Clean Up Diagram на панели инструментов блок-диаграммы



Упражнение 2-1

Тема: Исследование VI

ДОМАШНЕЕ ЗАДАНИЕ

Определите компоненты существующего VI.

GOAL

ЦЕЛЬ

Упражнение 2-1

Тема: Исследование VI

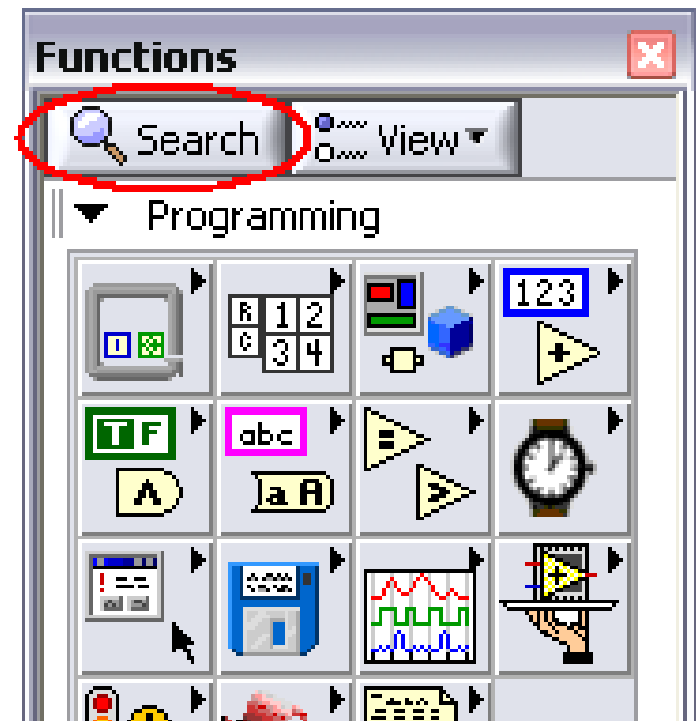
- Что такое константы и когда их нужно использовать?
- Что такое свободные метки и когда их нужно использовать?

ДИСКУССИЯ

DISCUSSION

G. Поиск элементов управления и индикации, VI и функций

Для поиска элементов управления и индикации, функций и VI используется кнопка **Search** палитр **Controls** и **Functions**.



Упражнение 2-2

Тема: Ориентация в палитрах

Научитесь искать в палитрах элементы управления и индикации, функции и VI.

GOAL

ЦЕЛЬ

Упражнение 2-2

Тема: Ориентация в палитрах

- Почему желательно добавлять функции в категорию **Favorites** палитры Functions?

ДИСКУССИЯ

DISCUSSION

Н. Выбор инструмента

- Создают, модифицируют и отлаживают VI с помощью инструментов LabVIEW
- Инструмент – это специальный режим работы курсора мыши
- Режим работы курсора соответствует иконке выбранного инструмента
- В режиме автоматического выбора инструмента LabVIEW выбирает инструмент, исходя из текущего положения курсора мыши



Упражнение 2-3

Тема: Выбор инструмента

Приобретите опыт использования режима автоматического выбора инструмента.

GOAL

ЦЕЛЬ

Упражнение 2-3

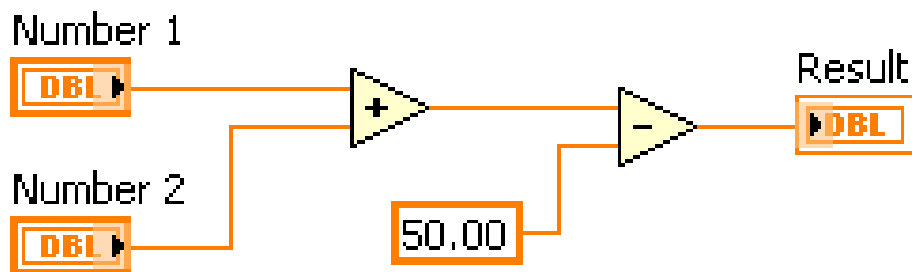
Тема: Выбор инструмента

- Как включить режим автоматического выбора инструмента?

I. Потокное программирование

LabVIEW использует модель потока данных для управления исполнением VI

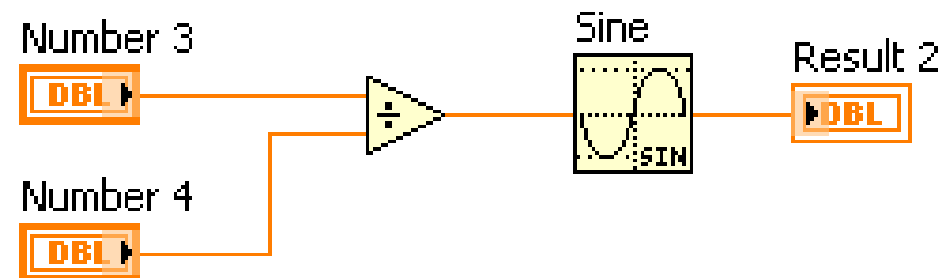
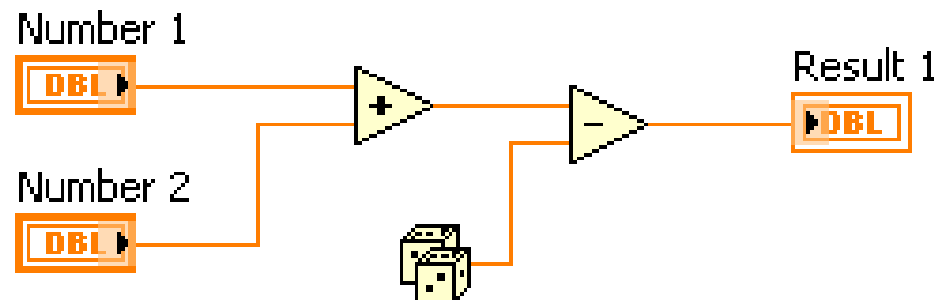
- Узел выполняется только, когда данные доступны на всех его входных терминалах
- Узел передает данные на выходные терминалы только когда завершается исполнение узла



I. Потокное программирование – Контрольный вопрос

Какой узел выполняется первым?

- a) Add
- b) Subtract
- c) Random Number
- d) Divide
- e) Sine

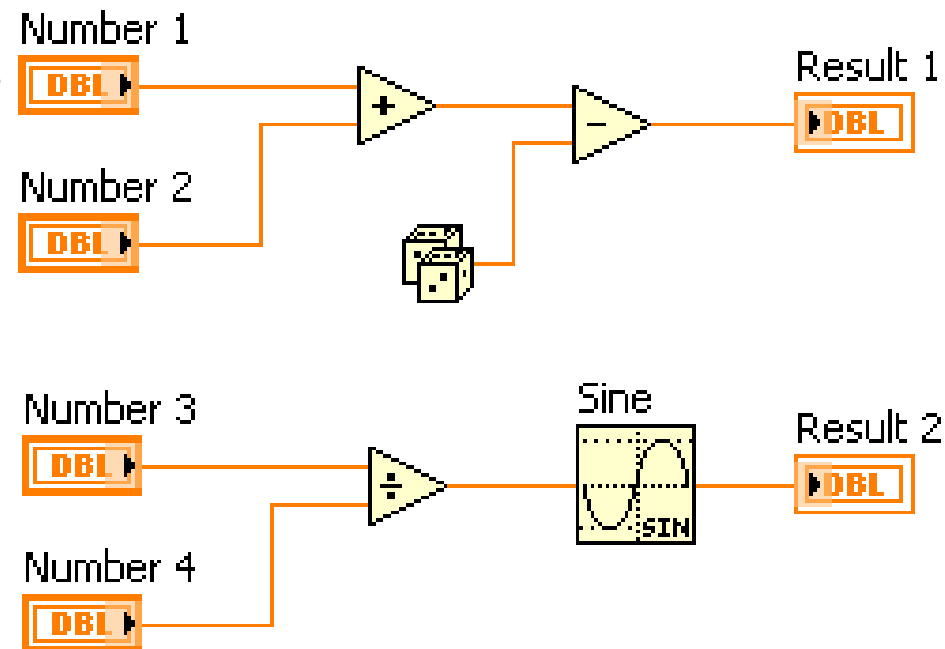


I. Потокное программирование – Ответ на контрольный вопрос

НЕТ КОРРЕКТНОГО ОТВЕТА

Какой узел выполняется первым?

- a) *Add – возможно*
- b) *Subtract – определенно нет*
- c) *Random Number – возможно*
- d) *Divide – возможно*
- e) *Sine – определенно нет*



Упражнение 2-4

Тема: Потокное программирование

Понять, как поток данных определяет порядок выполнения VI.

GOAL

ЦЕЛЬ

Упражнение 2-4

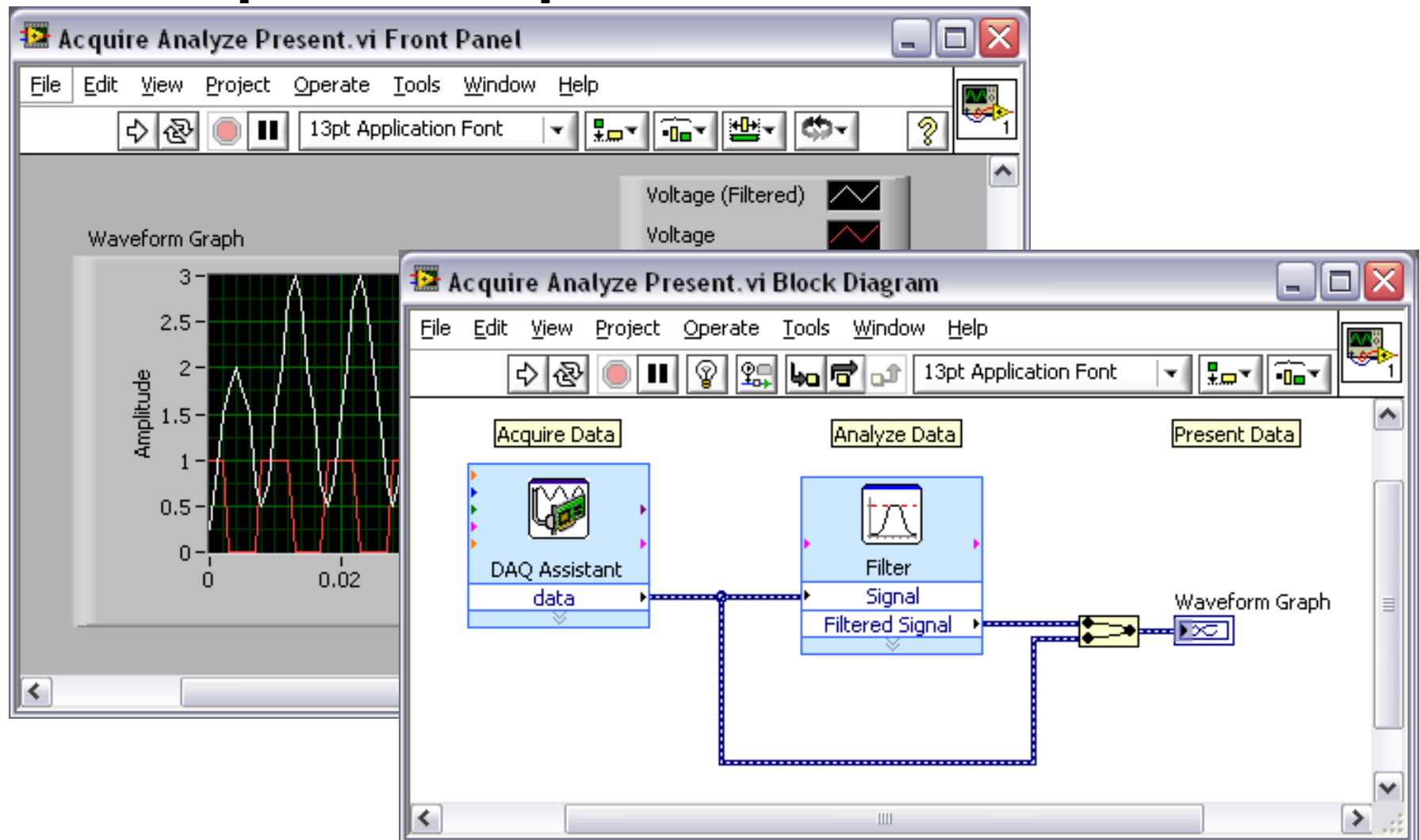
Тема: Потокное программирование

- Должна ли хорошо спроектированная блок-диаграмма «течь» в определенном направлении?

ДИСКУССИЯ

DISCUSSION

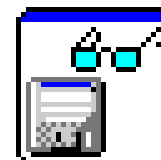
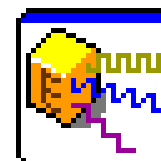
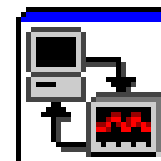
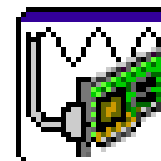
J. Разработка простого VI



J. Разработка простого VI – Получение данных

Express VI получения данных:

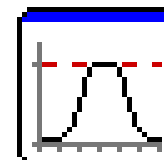
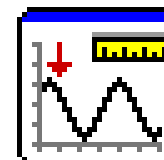
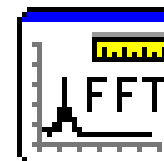
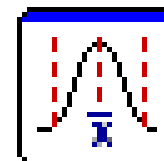
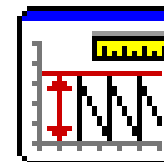
- DAQ Assistant Express VI
- Instrument I/O Assistant Express VI
- Simulate Signal Express VI
- Read from Measurement File Express VI



Ж. Разработка простого VI – Обработка

Express VI обработки:

- Amplitude and Level Measurements Express VI
- Statistics Express VI
- Spectral Measurements Express VI
- Tone Measurements Express VI
- Filter Express VI



Ж. Разработка простого VI – Представление данных

- Задачи представления решаются Express VI, которые выполняют некоторые функции, или индикаторами, которые отображают данные на лицевой панели VI
- Индикаторы – это Waveform Chart (графики диаграмм), Waveform Graph (графики осциллограмм) и XY Graph
- Express VIs – это Write to Measurement File Express VI (запись в файл результатов измерений), Build Text Express VI (компоновка текста), DAQ Assistant Express VI (помощник при работе с оборудованием сбора данных) и Instrument I/O Assistant Express VI (помощник при работе с измерительными приборами)

J. Разработка простого VI – выполнение

1. Поместите Express VI на блок-диаграмму
2. Сконфигурируйте Express VI в открывшемся диалоговом окне
3. Соедините Express VI друг с другом
4. Сохраните VI и запустите на исполнение

Кнопка Run выглядит разорванной, если созданный или редактируемый вами VI содержит ошибки



Упражнение 2-5

Простой VI сбора, обработки и представления данных

Создать простой VI сбора и обработки данных, а также представления результатов.

GOAL

ЦЕЛЬ

Упражнение 2-5

Простой VI сбора, обработки и представления данных

- Как определить путь к создаваемому текстовому файлу?

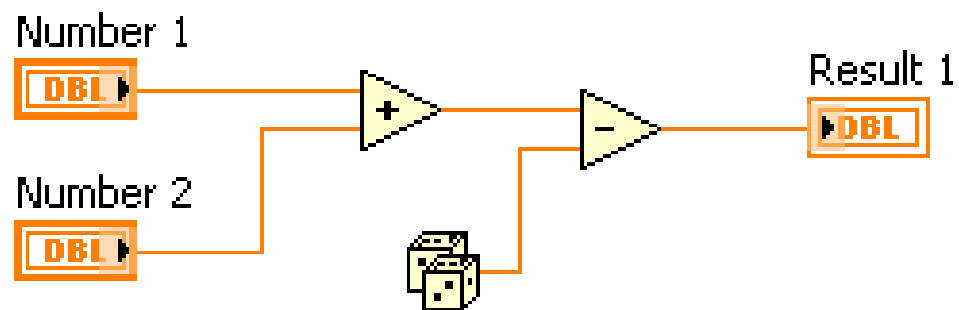
ДИСКУССИЯ

DISCUSSION

Заключение – Контрольный вопрос

1. Какая функция выполняется первой: Add или Subtract?

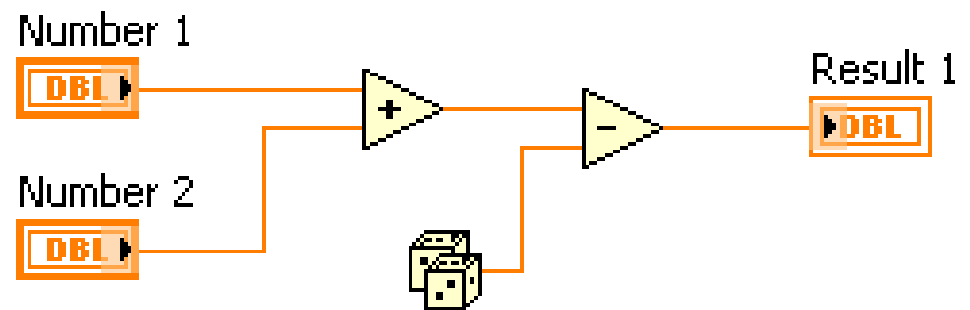
- a) Add
- b) Subtract
- c) Unknown (неизвестно)



Заключение – Ответ на контрольный вопрос

1. Какая функция выполняется первой: Add или Subtract?

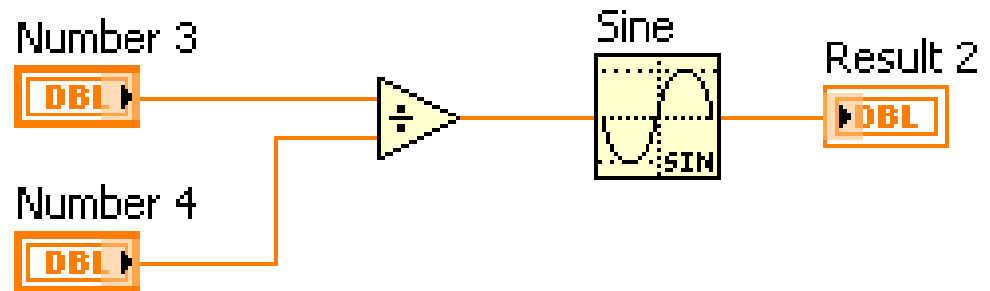
- a) **Add**
- b) Subtract
- c) Unknown (неизвестно)



Заключение – Контрольный вопрос

2. Какая функция выполняется первой: Sine или Divide?

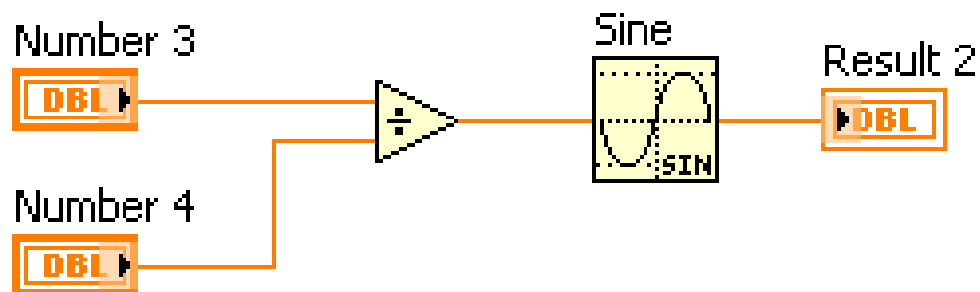
- a) Sine
- b) Divide
- c) Unknown (неизвестно)



Заключение – Ответ на контрольный вопрос

2. Какая функция выполняется первой: Sine или Divide?

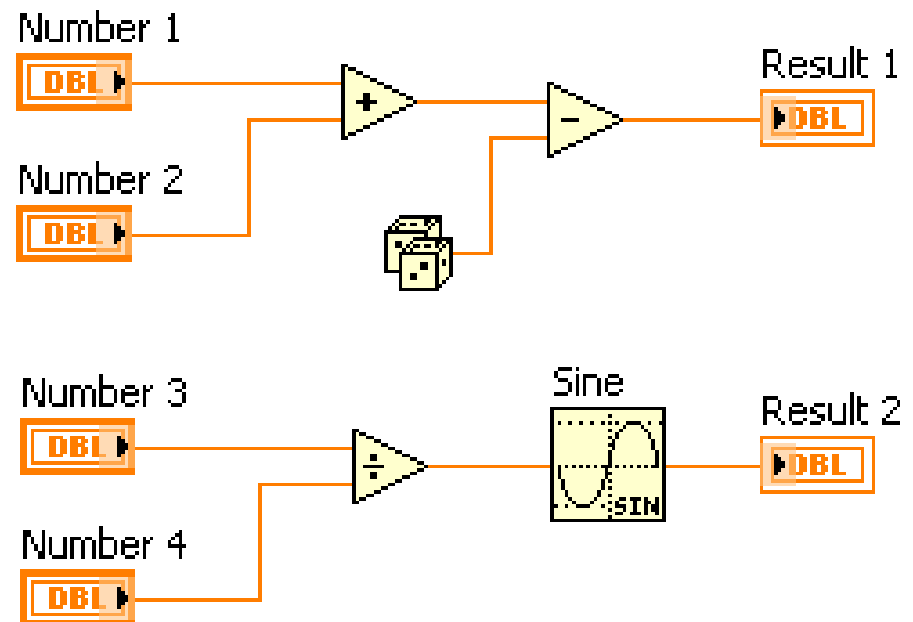
- a) Sine
- b) **Divide**
- c) Unknown (неизвестно)



Заключение – Контрольный вопрос

3. Какая из следующих функция выполняется раньше: Random Number, Add или Divide?

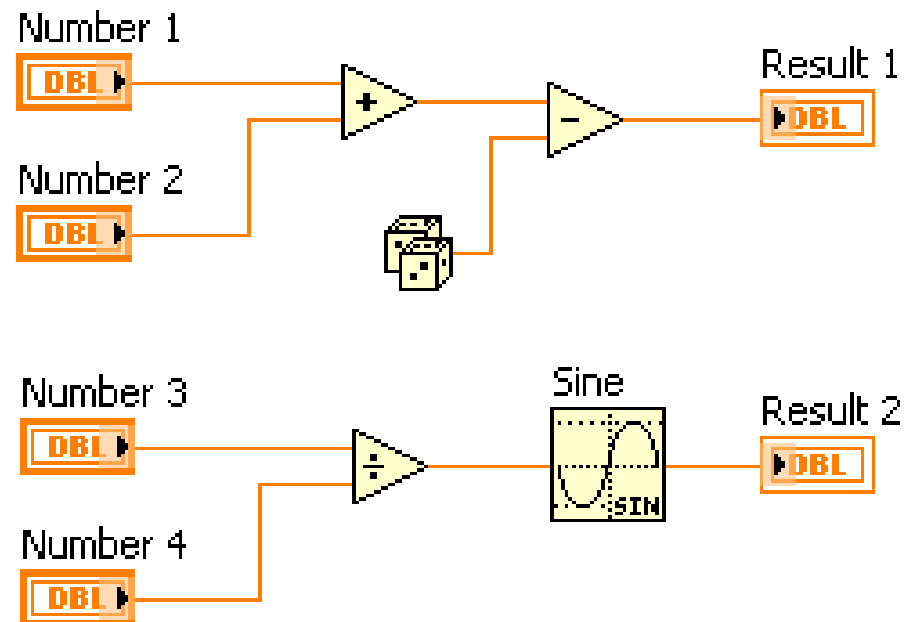
- a) Random Number
- b) Divide
- c) Add
- d) Unknown (неизвестно)



Заключение – Ответ на контрольный вопрос

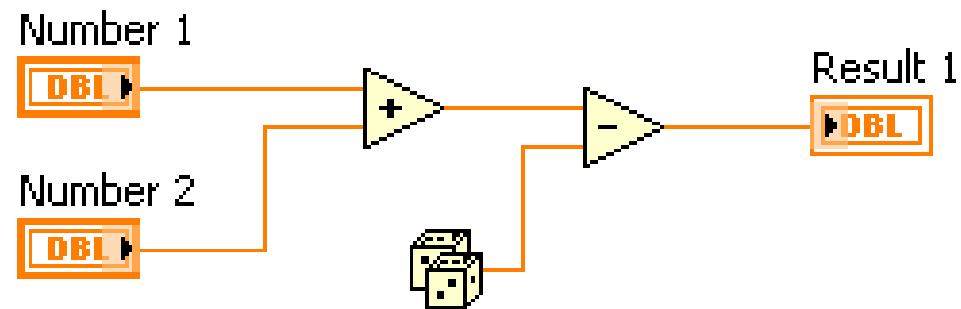
3. Какая из следующих функция выполняется раньше: Random Number, Add или Divide?

- a) Random Number
- b) Divide
- c) Add
- d) **Unknown (неизвестно)**



Заключение – Контрольный вопрос

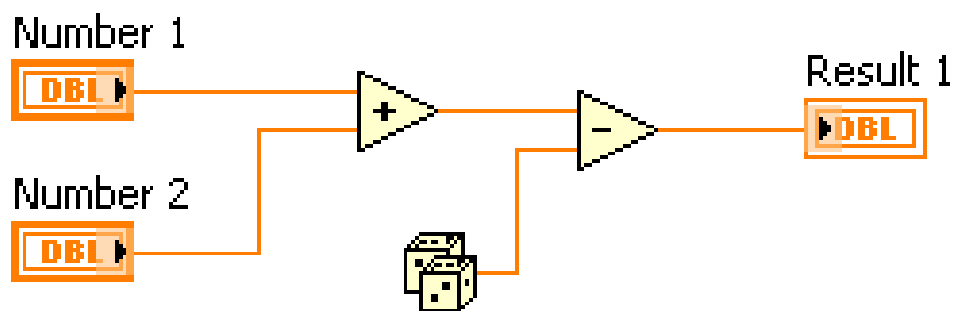
4. Какая из следующих функция выполняется последней: Random Number, Subtract or Add?
- a) Random Number
 - b) Subtract
 - c) Add
 - d) Unknown (неизвестно)



Заключение – ответ на контрольный вопрос

4. Какая из следующих функция выполняется последней: Random Number, Subtract or Add?

- a) Random Number
- b) **Subtract**
- c) Add
- d) Unknown (неизвестно)



Заключение – Контрольный вопрос

5. Из каких трех частей состоит VI?

- a) Front Panel
- b) Block Diagram
- c) Project
- d) Icon/Connector Pane

Заключение – Ответ на контрольный вопрос

5. Из каких трех частей состоит VI?

a) **Front Panel**

b) **Block Diagram**

c) Project

d) **Icon/Connector Pane**

Лекция 3

Поиск ошибок и отладка VI

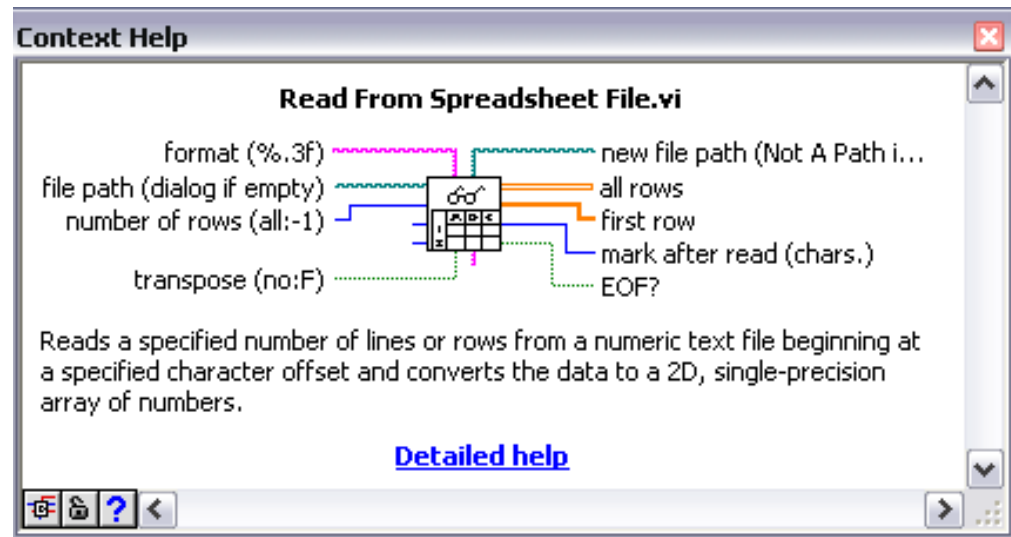
ТЕМЫ

- A. Справочные утилиты LabVIEW
- B. Исправление ошибок в VI
- C. Техника отладки
- D. Непонятные или неожиданные данные
- E. Контроль и обработка ошибок

A. Справочные утилиты LabVIEW

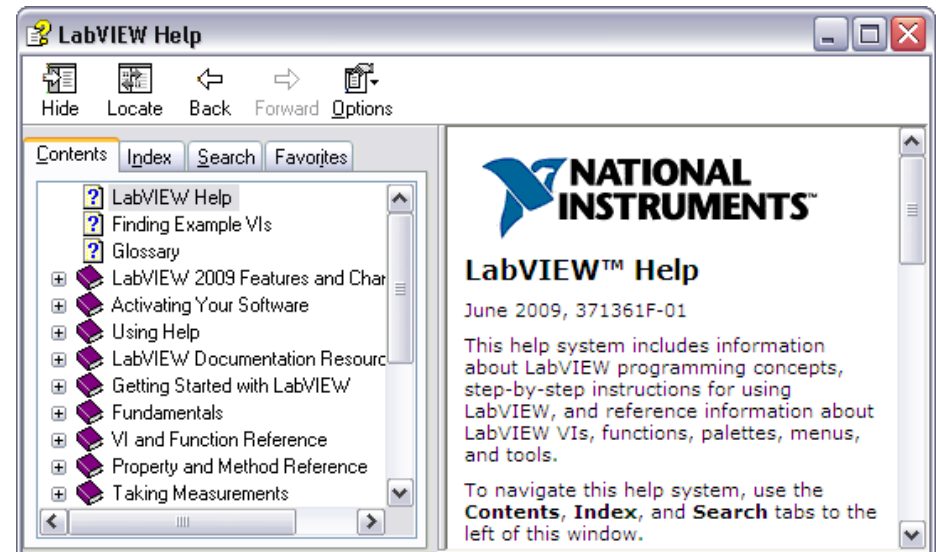
– Context Help (контекстная справка)

- Отображают основную информацию об объектах LabVIEW при наведении курсора на каждый объект
- Выберите **Help»Show Context Help**, нажмите <Ctrl-H> или щелкните по кнопке **Show Context Help Window** на панели инструментов

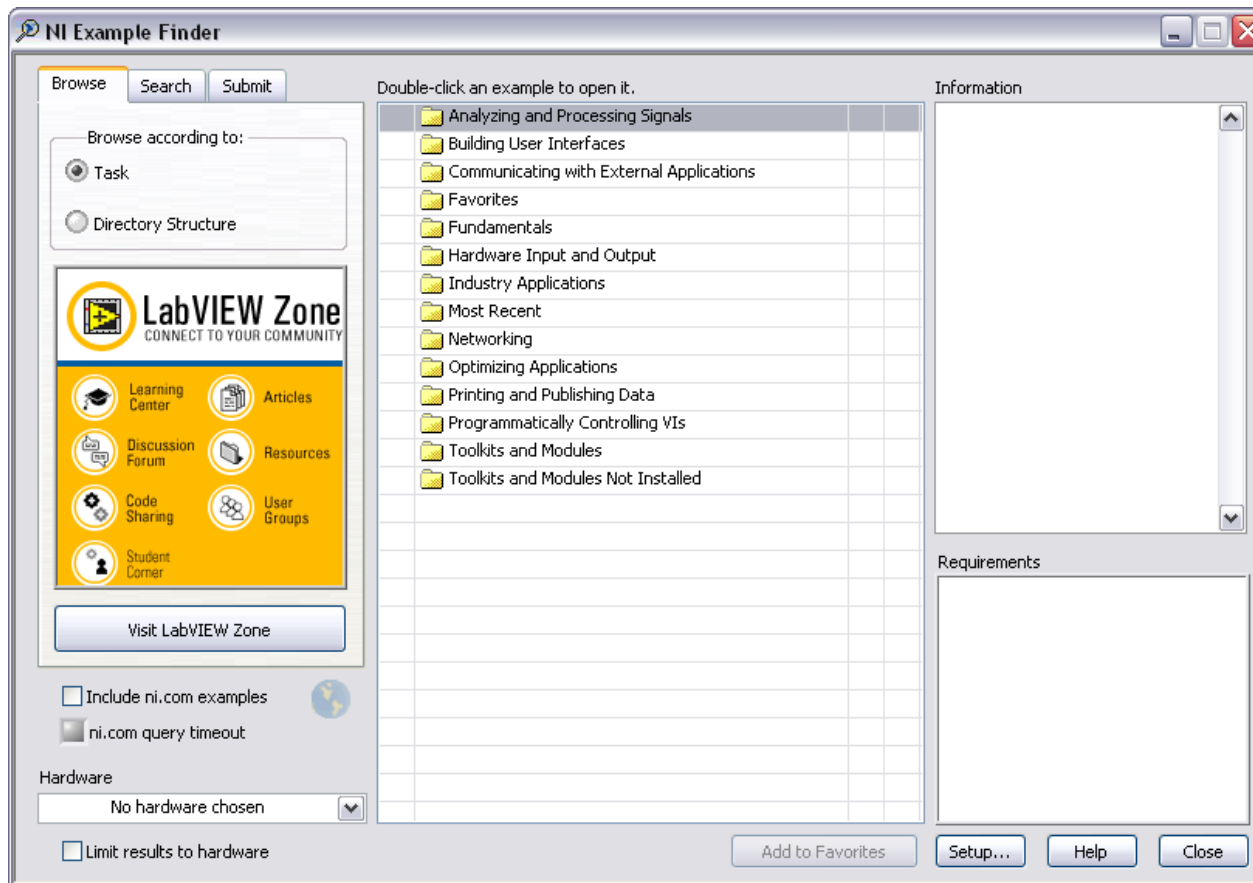


А. Справочные утилиты LabVIEW – LabVIEW Help

- Подробно описывают большинство палитр, меню, инструментов, VI и функций, а также инструкции по их использованию LabVIEW
- Вход в *LabVIEW Help*:
 - Выберите **Help»Search the LabVIEW Help**
 - Используйте ссылку **Detailed help** или кнопку в окне **Context Help**
 - Щелкните правой кнопкой по объекту и выберите **Help** в контекстном меню



A. Справочные утилиты LabVIEW – NI Example Finder (поисковик примеров)



Упражнение 3-1

Тема: использование справочной системы

ДОМАШНЕЕ ЗАДАНИЕ

Познакомиться с использованием окна **Context Help**, справочной системы *LabVIEW Help* и поисковика примеров NI Example Finder.

GOAL

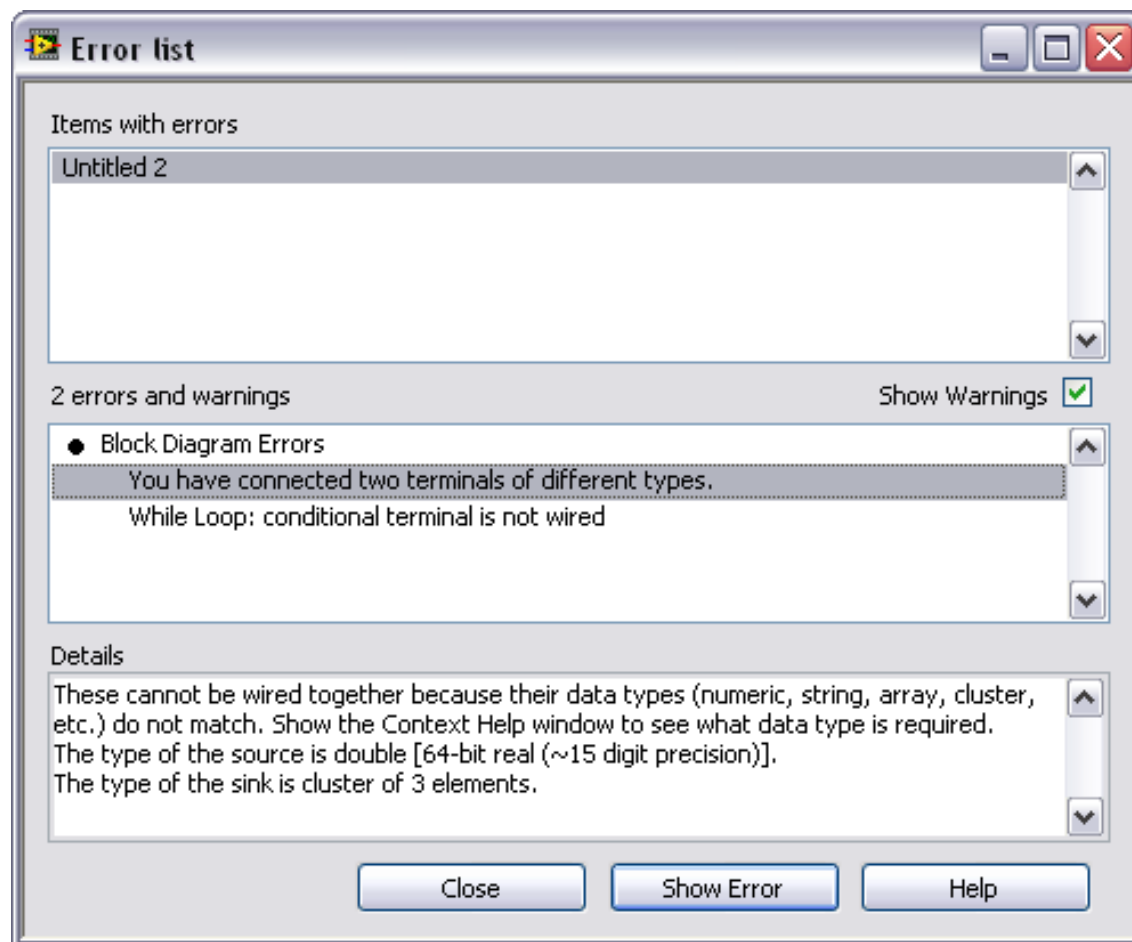
ЦЕЛЬ

Упражнение 3-1

Тема: Использование справочной системы

- Вы работаете с VI, в котором содержатся незнакомые вам функции. Как вы определите функциональность блок-диаграммы?

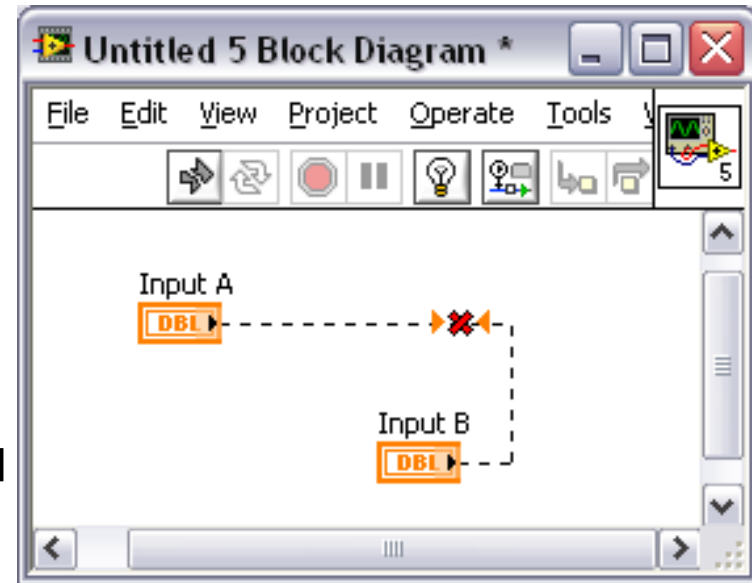
В. Исправление ошибок в VI



В. Исправление ошибок в VI

Распространенные проблемы

- Разорванные проводники
 - Вы соединили булевский элемент управления со строковым индикатором
 - Вы соединили числовой элемент управления с числовым элементом управления
- Не подключен обязательный для подключения терминал на блок-диаграмме
- Использован "неисправных" subVI или панель подключения subVI редактировалась после установки иконки subVI на блок-диаграмму VI



С. Техника отладки

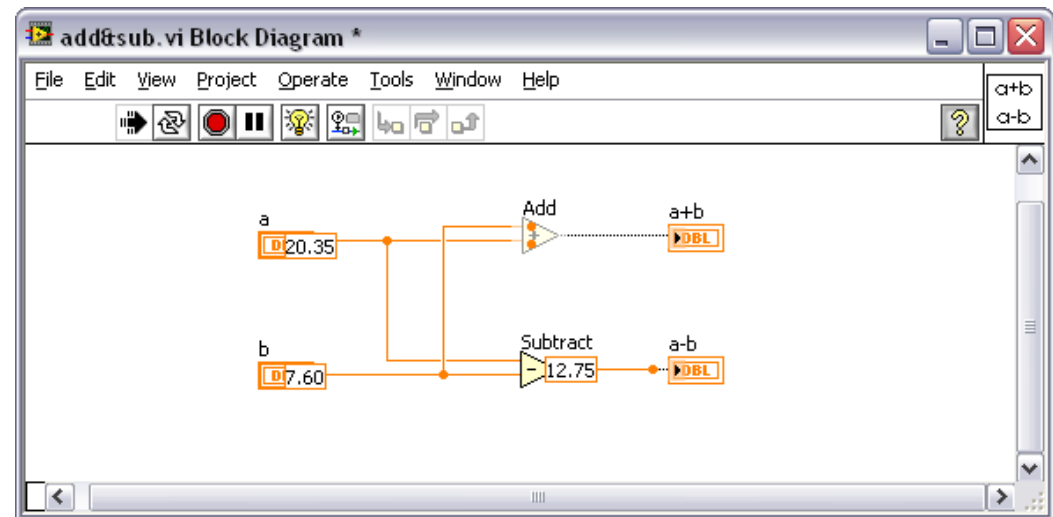
Ваш VI «исправен», но вы получаете данные или характеристики, отличные от тех, которые должны быть

- Какой-то subVI не подключен или невидим?
- Используются некорректные данные по умолчанию?
- Пропускаются неопределенные данные?
- Представление чисел корректно?
- Порядок выполнения узлов правильный?

С. Техника отладки – Execution Highlighting (подсветка выполнения)



- Используйте подсветку выполнения для визуального наблюдения потока данных на блок-диаграмме
- Если VI выполняется намного медленнее, чем ожидалось, убедитесь, что вы отключили подсветку выполнения в subVI

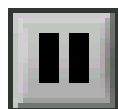


С. Техника отладки – Single Stepping (пошаговое выполнение)

Пошаговое выполнение применяется для наблюдения результатов каждого действия VI на блок-диаграмме

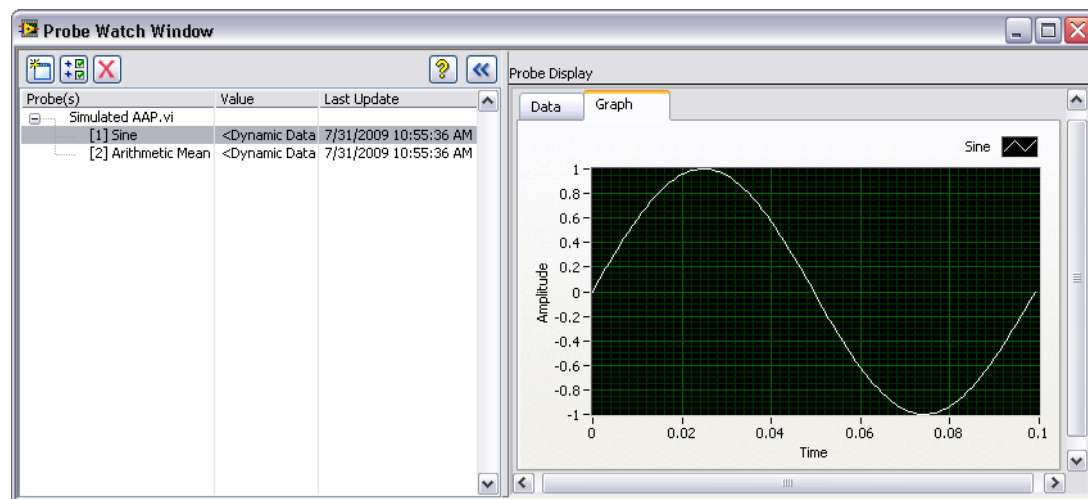
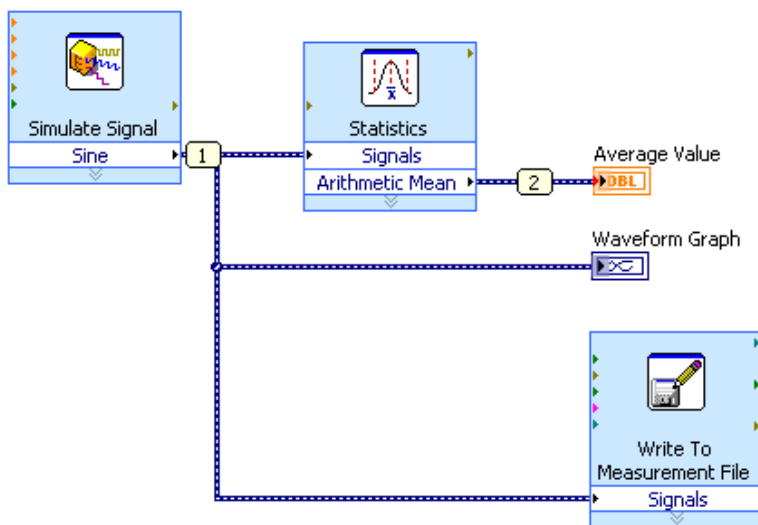
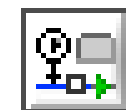
Приостанавливает выполнение subVI для изменения значения элементов управления и индикации, чтобы контролировать количество шагов выполнения или для возврата к началу выполнения subVI

- Откройте subVI и выберите в контекстном меню пункт **Operate»Suspend When Called**



С. Техника отладки – Probes (пробники)

- Используйте инструмент Probe для контроля промежуточных значений данных и проверки выходов ошибок VI и функций, особенно тех, которые выполняют операции I/O
- Сохраняют значения в проводниках, так что вы можете видеть данные после завершения выполнения



С. Техника отладки – Breakpoints (контрольные точки)



- Если во время выполнения достигается контрольная точка, выполнение VI приостанавливается и кнопка **Pause** становится красной
- При останове в контрольной точке можно выполнять следующие действия:
 - Контролировать VI по шагам с помощью кнопок пошагового выполнения
 - Наблюдать пробники для контроля промежуточных значений
 - Изменять значения в элементах управления на лицевой панели
 - Щелкнуть по кнопке **Pause** для продолжения выполнения до следующей контрольной точки или до завершения работы VI

D. Непонятные или неожиданные данные

- ∞ (Inf)
 - Infinity (бесконечность)
 - Деление числа на ноль?
- NaN
 - Not a number (не число)
 - Выполняется запрещенная операция, например, извлечение квадратного корня из отрицательного числа
- Контролируйте появление непредвиденных значений Inf или NaN при выполнении математических операций

Е. Контроль и обработка ошибок

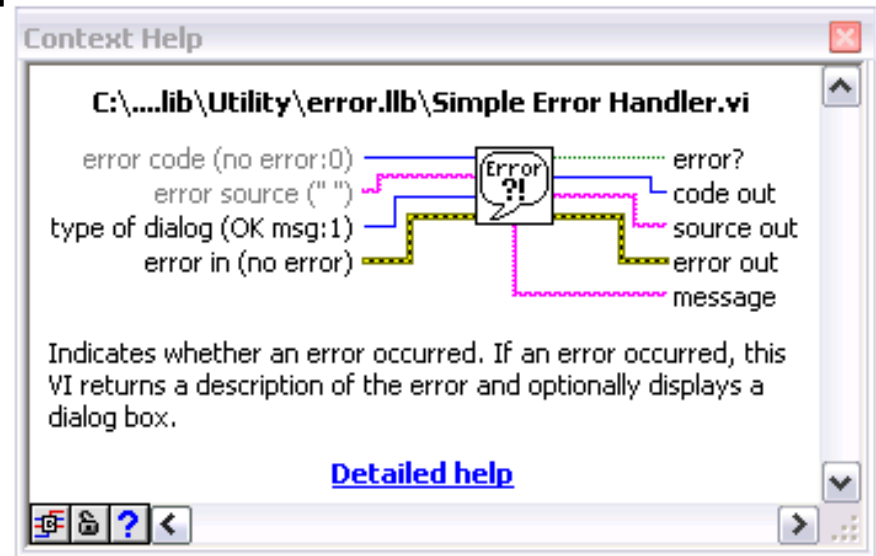
- Независимо от того, как создавался VI, вы не можете предвидеть все проблемы, с которыми может столкнуться пользователь
- Без механизма контроля ошибок, вы можете узнать только, что VI работает неправильно
- Контроль ошибок сообщит вам, где и почему возникла ошибка
 - Автоматическая обработка ошибок
 - Ручная обработка ошибок

Е. Контроль и обработка ошибок – Автоматическая обработка ошибок

- LabVIEW автоматически обрабатывает любые известные ошибки в процессе выполнения VI :
 - приостанавливая выполнение,
 - подсвечивая subVI или функцию, в которых обнаружена ошибка,
 - отображения диалогового окна Error
- Выберите **File»VI Properties**, а затем из выпадающего меню Category - пункт **Execution** для отключения автоматической обработки ошибок в конкретном VI

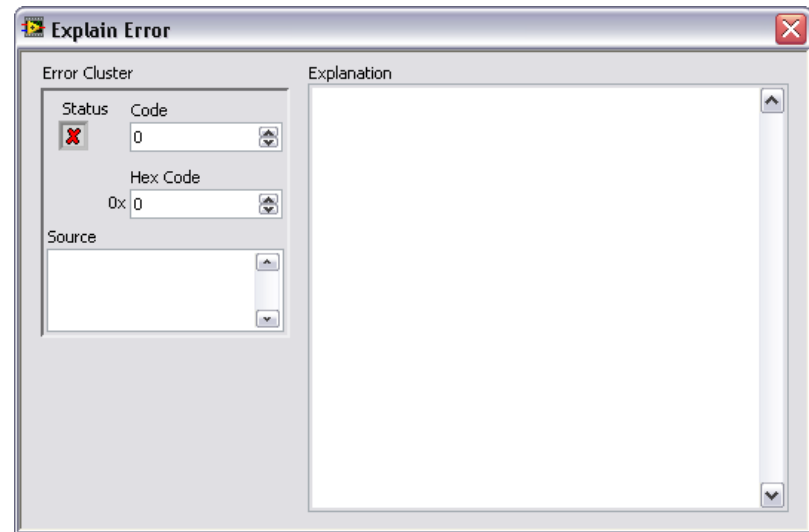
Е. Контроль и обработка ошибок – Ручная обработка ошибок

- Для отключения автоматической обработки ошибок subVI или функции, соедините выход кластера **error out** с входным кластером **error in** другого subVI или функции или с индикатором ошибки
- Используйте VI, функции и параметры обработки ошибок для управления ошибками



Е. Контроль и обработка ошибок – Error Clusters – кластеры ошибок

- Используйте индикаторы и элементы управления кластера ошибок для создания входов и выходов ошибок subVI
- Кластеры **error in** и **error out** состоят из следующих информационных компонентов :
 - Status (состояние)
 - Code (код)
 - Source (источник)



Упражнение 3-2

Тема: Отладка

Используйте встроенные в LabVIEW средства отладки.

GOAL

ЦЕЛЬ

Упражнение 3-2

Тема: Отладка

- Если у вашего VI стрелка Run изломана, что нужно сделать прежде всего?
- После того, как вы «исправили» изломанную стрелку Run, VI выдает непредвиденные данные. Что теперь вы должны делать?

ДИСКУССИЯ

DISCUSSION

Заключение – Контрольный вопрос

1. Как запретить автоматическую обработку ошибок?
 - a) Включить подсветку выполнения
 - b) Соединить выходной кластер ошибки subVI со входным кластером ошибки другого subVI
 - c) Установить флажок в поле **Show Warnings** диалогового окна **Error List**

Заключение – Ответ на контрольный вопрос

1. Как запретить автоматическую обработку ошибок?
 - a) Включить подсветку выполнения
 - b) Соединить выходной кластер ошибки subVI со входным кластером ошибки другого subVI**
 - c) Установить флажок в поле **Show Warnings** диалогового окна **Error List**

Заключение – Контрольный вопрос

2. Из каких компонентов состоит кластер ошибок?

- a) Status: Boolean
- b) Error: String
- c) Code: 32-bit integer
- d) Source: String

Заключение – Ответ на контрольный вопрос

2. Из каких компонентов состоит кластер ошибок?

- a) **Status: Boolean**
- b) Error: String
- c) **Code: 32-bit integer**
- d) **Source: String**

Лекция 4

Реализация VI

ТЕМЫ

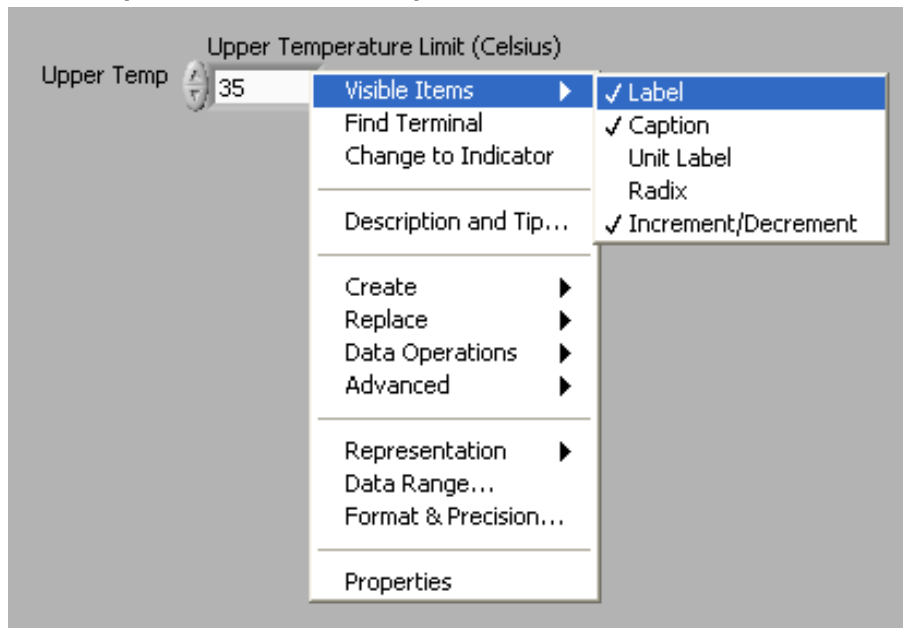
- A. Проектирование лицевой панели
- B. Типы данных LabVIEW
- C. Документирование кода
- D. Циклы While
- E. Циклы For
- F. Тактирование VI
- G. Передача данных между итерациями
- H. Вывод данных на графические индикаторы
- I. Структуры Case

А. Проектирование лицевой панели

- Проектирование лицевой панели сводится к созданию ВХОДОВ и ВЫХОДОВ
- Входные данные получают следующими способами:
 - От устройства сбора данных
 - Непосредственным чтением из файла
 - Манипуляциями с элементами управления
- Выходные данные получают следующими способами :
 - Отображением на индикаторах
 - Сохранением в файле
 - Выводом через устройства вывода

A. Проектирование лицевой панели – Labels/Captions (метки/заголовки)

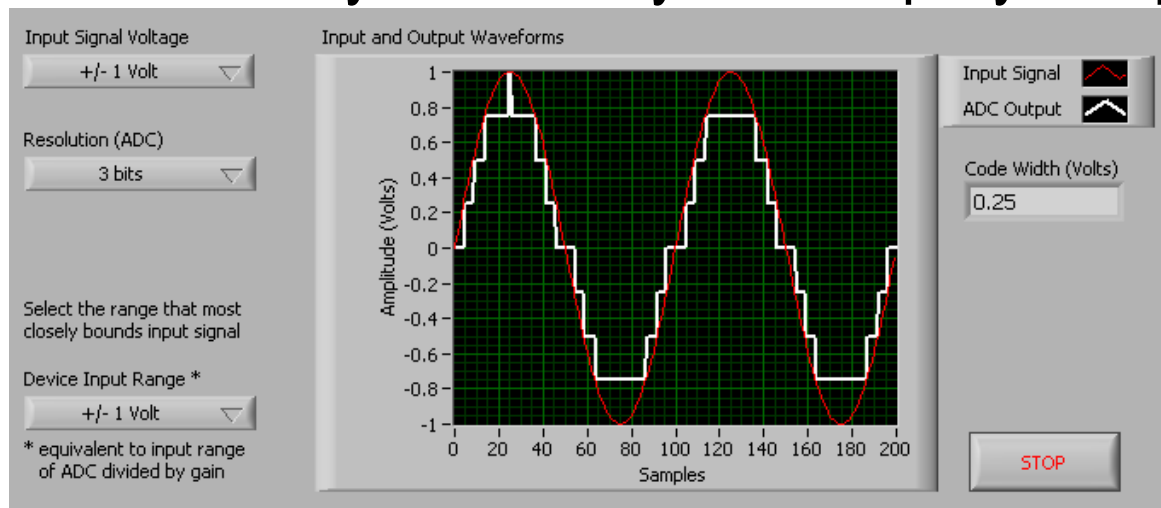
- Labels – короткие описания
- Captions – длинные описания
- Captions (заголовки) не видны на блок-диаграмме



А. Проектирование лицевой панели – советы по использованию цвета

Начинайте с использования серой гаммы

- Выберите один или два оттенка серого
- Экономно добавляйте цвета подсвечивания (более яркие) для важных настроек – графиков, кнопки завершения и бегунков ползунковых регуляторов



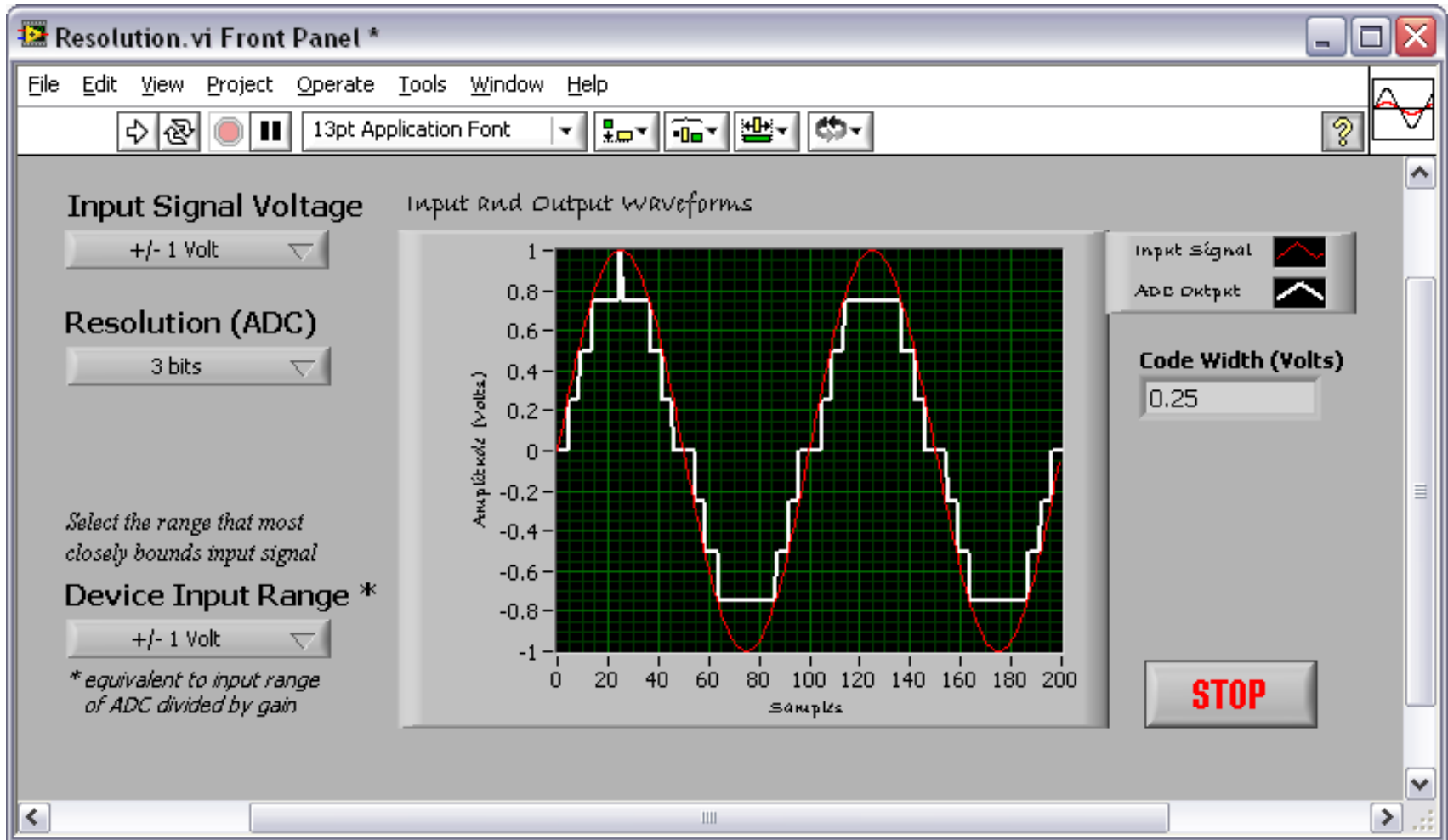
А. Проектирование лицевой панели – промежутки (зазоры)

New VI	Ctrl+N
New...	
Open...	Ctrl+O
Close	Ctrl+W
Close All	
Save	Ctrl+S
Save As...	
Save All	Ctrl+Shift+S
Save for Previous Version...	
Revert...	
New Project	
Open Project...	
Save Project	
Close Project	
Page Setup...	
Print...	
Print Window...	Ctrl+P
VI Properties	Ctrl+I
Recent Projects	▶
Recent Files	▶
Exit	Ctrl+Q

New VI	Ctrl+N
New...	
Open...	Ctrl+O
Close	Ctrl+W
Close All	
Save	Ctrl+S
Save As...	
Save All	Ctrl+Shift+S
Save for Previous Version...	
Revert...	
New Project	
Open Project...	
Save Project	
Close Project	
Page Setup...	
Print...	
Print Window...	Ctrl+P
VI Properties	Ctrl+I
Recent Projects	▶
Recent Files	▶
Exit	Ctrl+Q

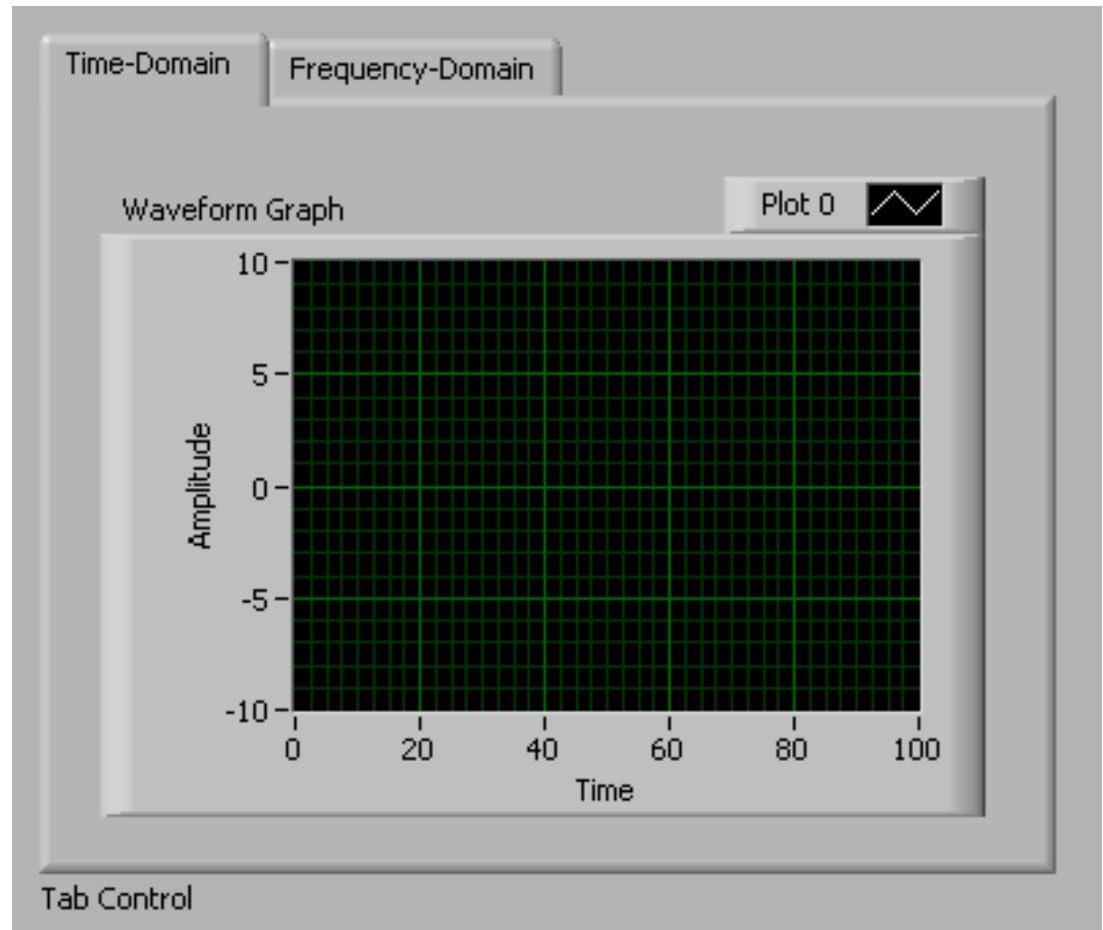
A. Проектирование лицевой панели – Текст и шрифты

Пример плохого оформления



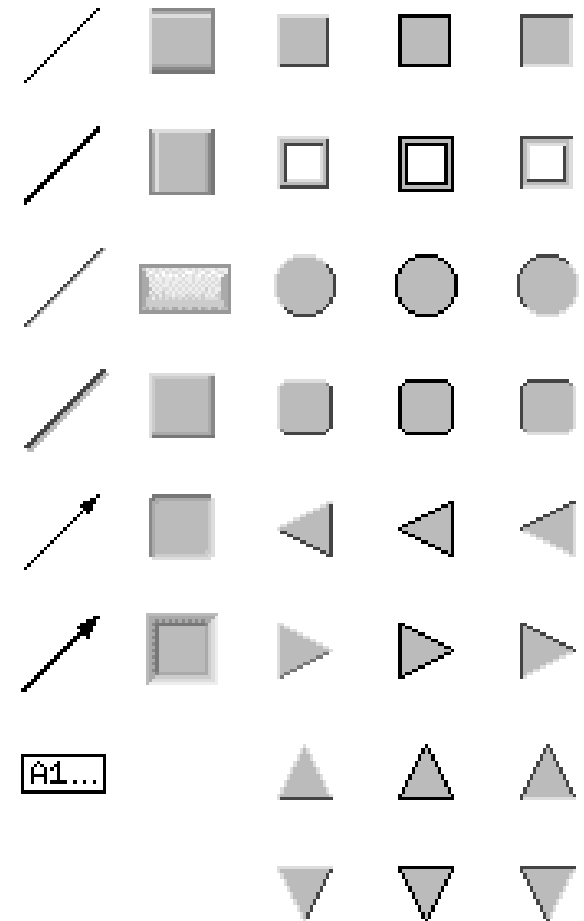
А. Проектирование лицевой панели – Tab Controls (табуляторный элемент управления)

Используйте tab controls для того, чтобы элементы управления и индикации можно было накладывать друг на друга, и они занимали меньше площади



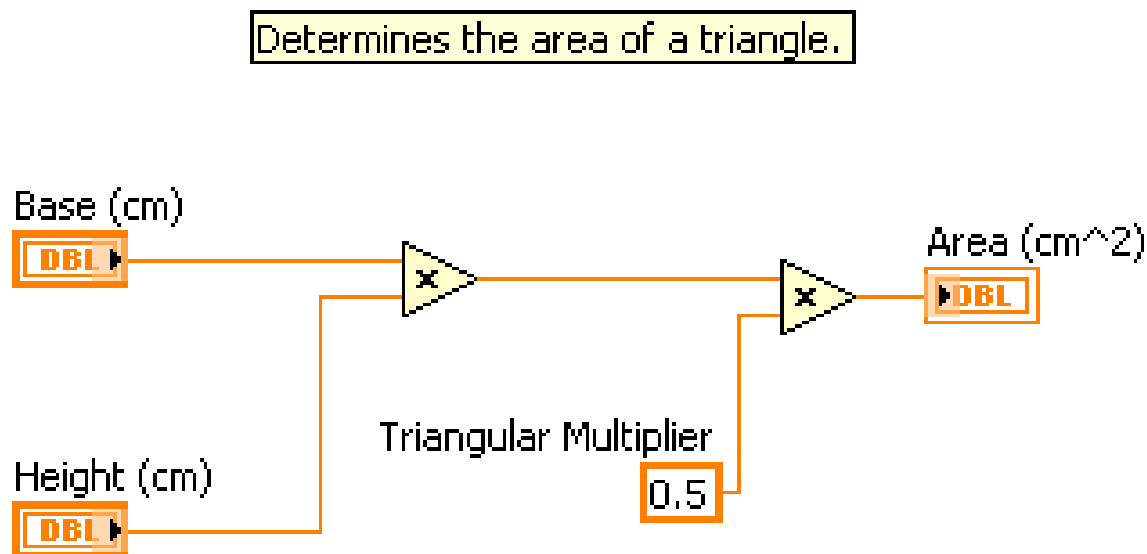
А. Проектирование лицевой панели – Элементы декорации

- Используйте элементы декорации – боксы, линии или стрелки для визуального группирования и разделения объектов лицевой панели
- Эти объекты служат только для декоративного оформления



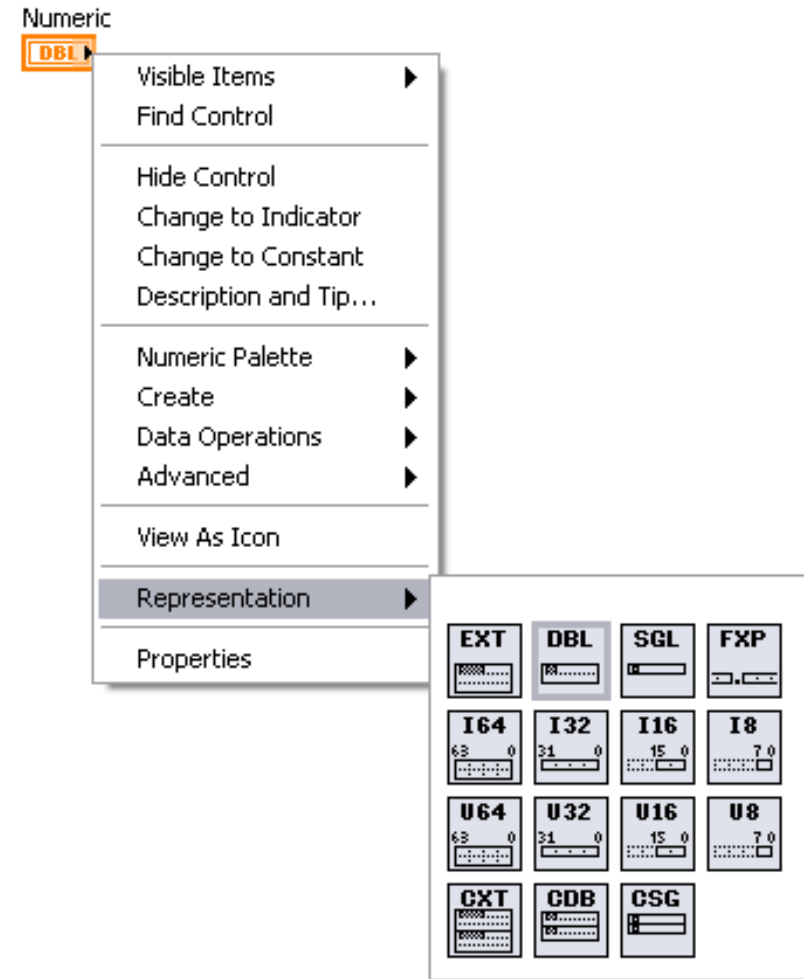
В. Типы данных LabVIEW – Terminals

Внешний вид терминалов визуально информируют о типе представляемых данных



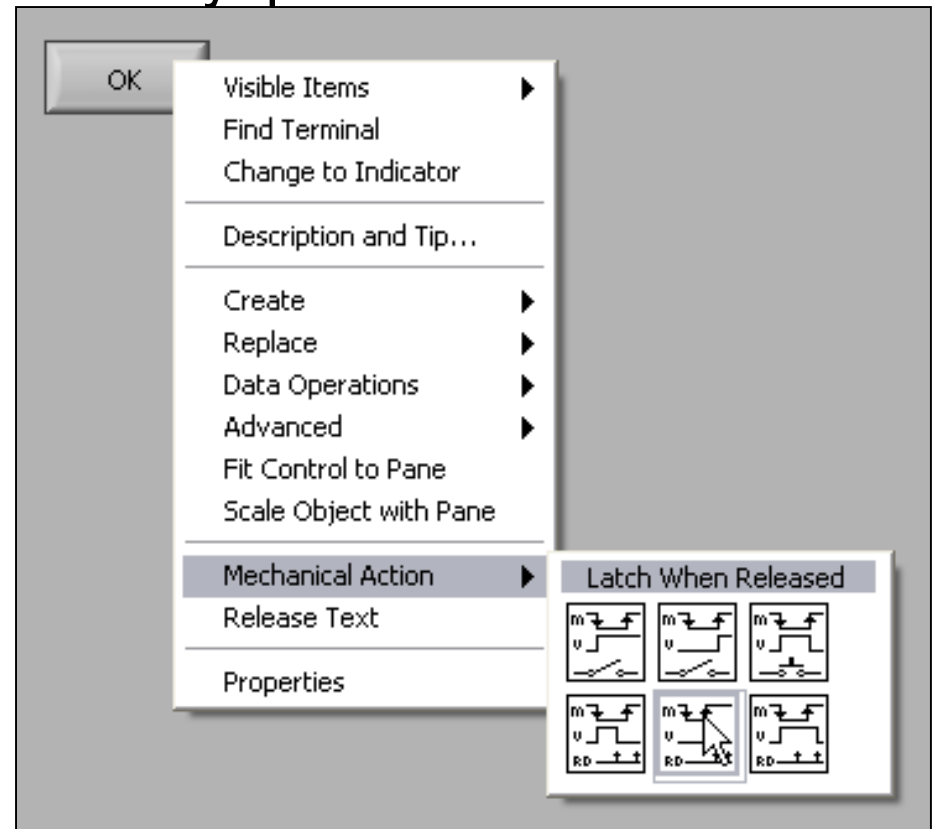
В. Типы данных LabVIEW – Numerics (числовые)

- Числа разных форматов представляются числовым типом данных
- Для изменения формата представления чисел щелкните правой кнопкой мыши по элементу управления или индикации или константе и выберите в контекстном меню пункт **Representation**



В. Типы данных LabVIEW – Boolean

- Поведение булевских элементов управления отличается механическим действием
- Булевский тип данных в LabVIEW представляется **зеленым** цветом



Механические функции булевских элементов управления

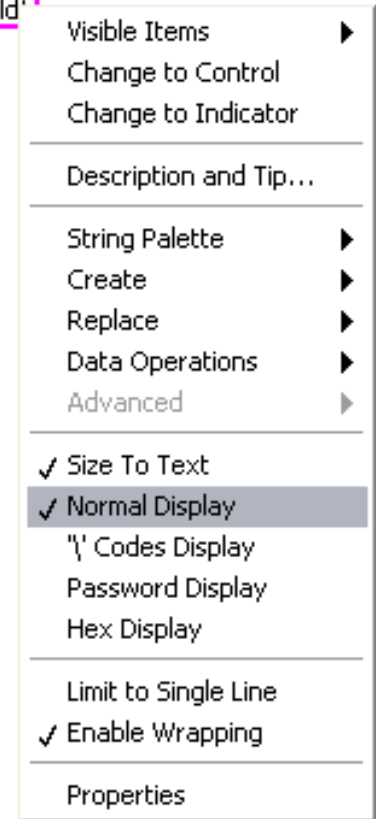
Используйте Mechanical Action of Booleans VI из поисковика примеров (NI Example Finder) для изучения различных свойств переключателей и защелок.

ДЕМОНСТРАЦИЯ

Hello World

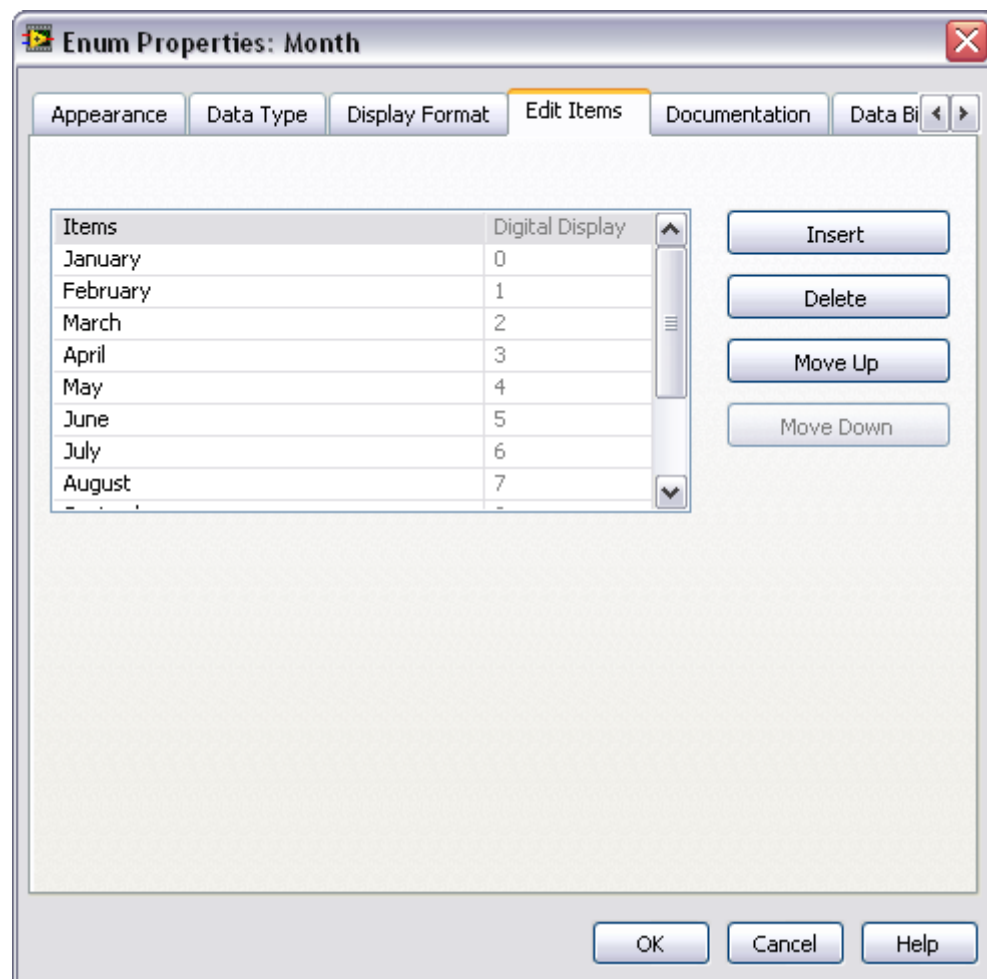
В. Типы данных – String (строки)

- Последовательность отображаемых и неотображаемых символов ASCII
- Строковые данные на лицевой панели используются в таблицах, полях ввода текста и метках
- Изменяйте форматы отображения из контекстного меню: Normal, '\ ' Codes, Password и Hex
- На блок-диаграмме редактировать строки и манипулировать с ними можно с помощью функций субпалитры String
- Данные строкового типа в LabVIEW отображаются **розовым** цветом (**pink**)



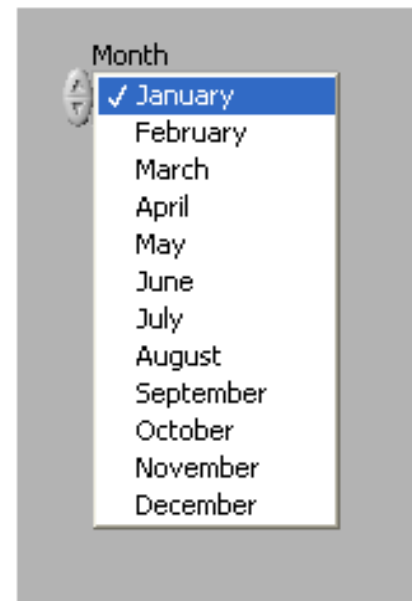
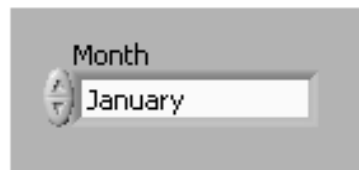
В. Типы данных – Enum (перечислительный)

Перечислительный тип данных представляет пару из строки и числа, где enum может принимать одно из определяемых списком значений



В. Типы данных – Enum (перечислительный)

- Enum: элементы управления, константы или элементы индикации
- Объекты с типом данных Enums полезны, т.к. на блок-диаграмме проще манипулировать с числами, чем со строками



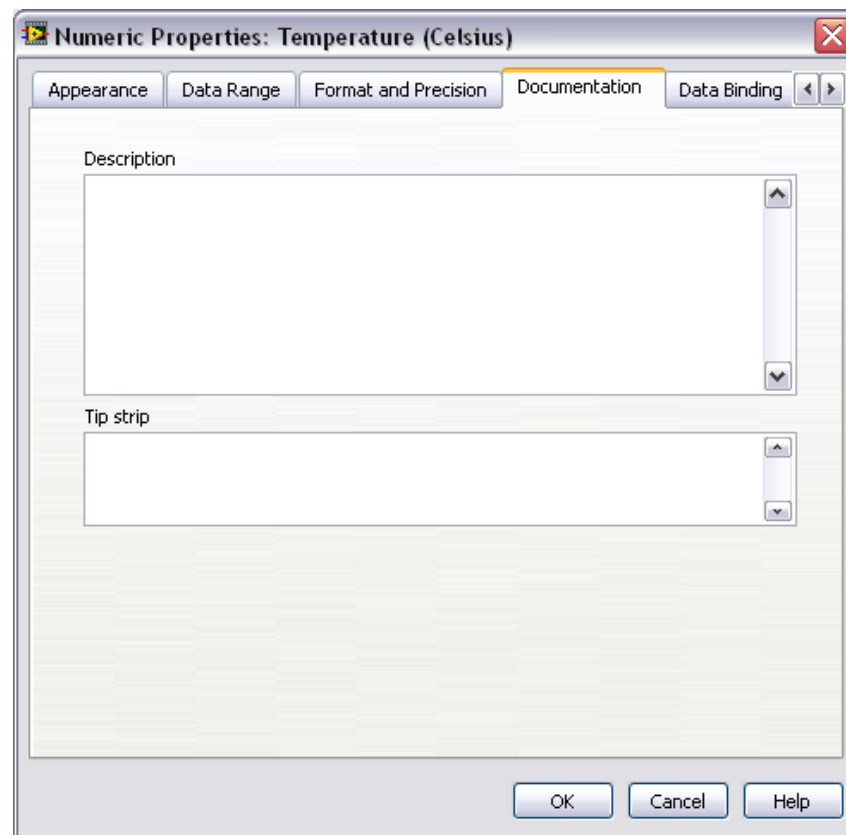
В. Типы данных – Dynamic (динамический)



- Хранит информацию, генерируемую или получаемую в Express VI
- VI не Express типа не принимают данные типа dynamic
 - Чтобы использовать встроенные VI или функции для анализа или обработки данных типа dynamic, необходимо выполнить преобразование типов данных
 - Индикаторы данных типа Numeric, waveform или Boolean, а также входы автоматически конвертируют данные типа при подключении
- Данные типа dynamic в LabVIEW отображаются **темно-синим** цветом (dark blue)

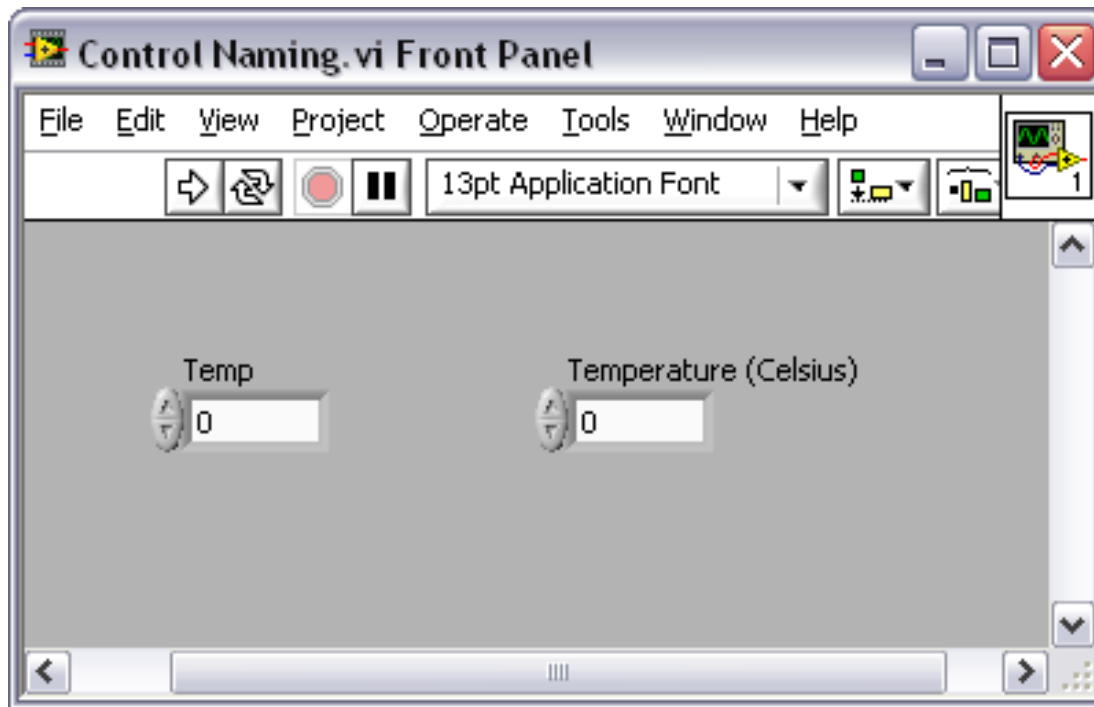
С. Документирование кода – Front Panels

- Tip Strips (строки подсказок)
- Descriptions (описания)
- VI Properties (свойства VI)
- Good Design (хороший стиль проектирования)



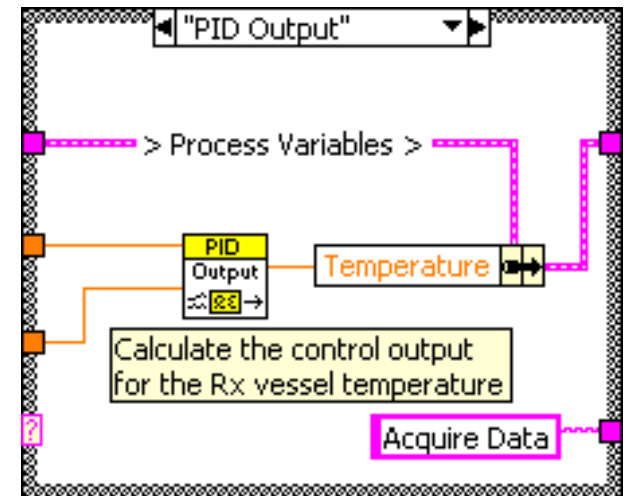
С. Документирование кода – Присвоение имен

Давайте элементам управления и индикации логичные и наглядные имена, увеличивая тем самым удобство работы с лицевой панелью



С. Документирование кода – Block Diagram

- Используйте на блок-диаграмме комментарии для:
 - Описания алгоритмов
 - Разъяснения содержания данных в проводниках
- Используйте инструмент Labeling или помещайте свободные метки из палитры **Functions**



Конфигурирование среды проектирования LabVIEW

- Диалоговое окно Options
 - Страница палитр Controls/Functions
 - Выберите **Load palettes during launch**, чтобы можно было использовать поиск в палитрах Search сразу после запуска
 - Установите формат палитр (Set Palette) в **Category (Icons and Text)**
 - Страница Block Diagram
 - Снимите флажок **Place front panel terminals as icons**, чтобы терминалы элементов управления и индикации отображались в компактном виде
 - Сконфигурируйте режим **Block Diagram Cleanup** для подгонки блок-диаграмм

Конфигурирование среды проектирования LabVIEW

- Палитра Functions
 - Прикрепите кнопкой палитру Functions и, выбрав **View»Change Visible Categories**, щелкните по пункту **Select All**
- Палитра Controls
 - Прикрепите кнопкой палитру Controls и, выбрав **View»Change Visible Categories**, щелкните по пункту **Select All**

Упражнение 4-1

Determine Warnings VI

ДОМАШНЕЕ ЗАДАНИЕ

Создать и задокументировать VI.

GOAL

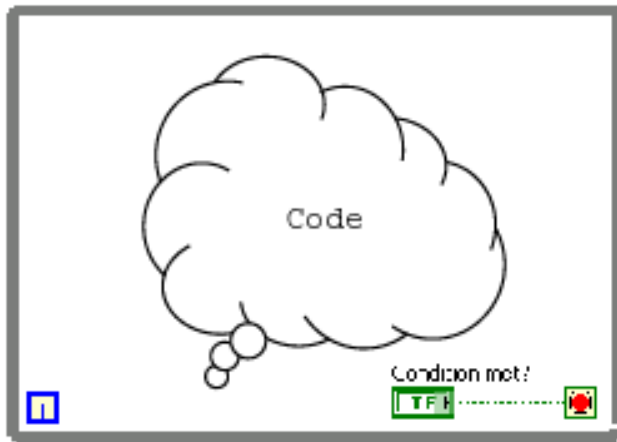
ЦЕЛЬ

Упражнение 4-1

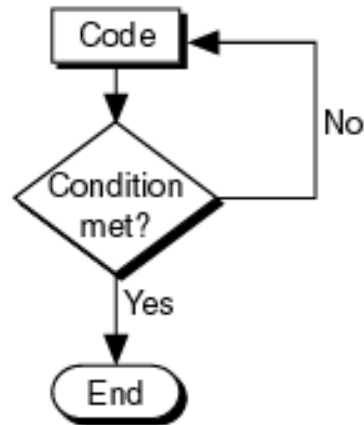
Determine Warnings VI

- Что будет, если значение Max. Temp ниже значения Min. Temp?

D. Циклы While



LabVIEW While Loop



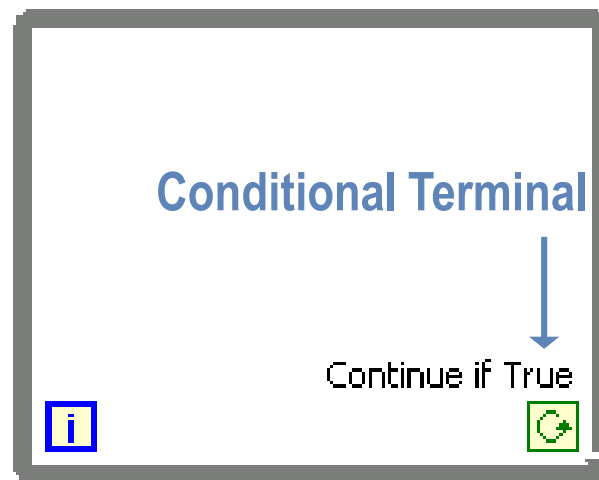
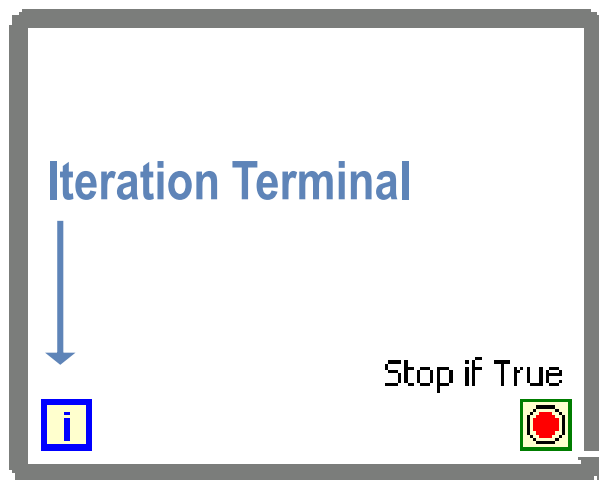
Flowchart

```
Repeat (code) ;  
Until Condition met ;  
End ;
```

Pseudo Code

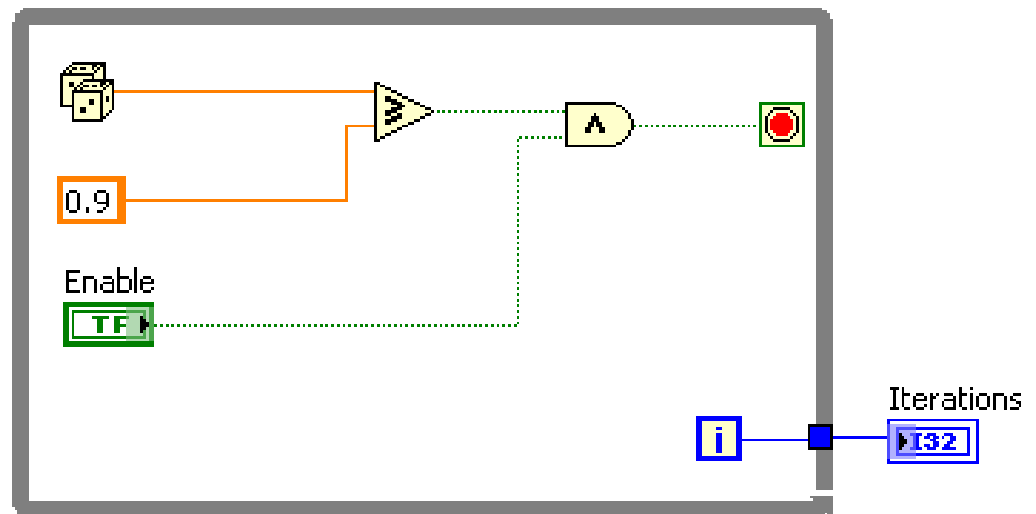
D. Циклы While

- Терминал итераций (Iteration terminal): возвращает количество выполнений цикла; индексируется с нуля
- Терминал условия (Conditional terminal): определяет условие останова выполнения цикла



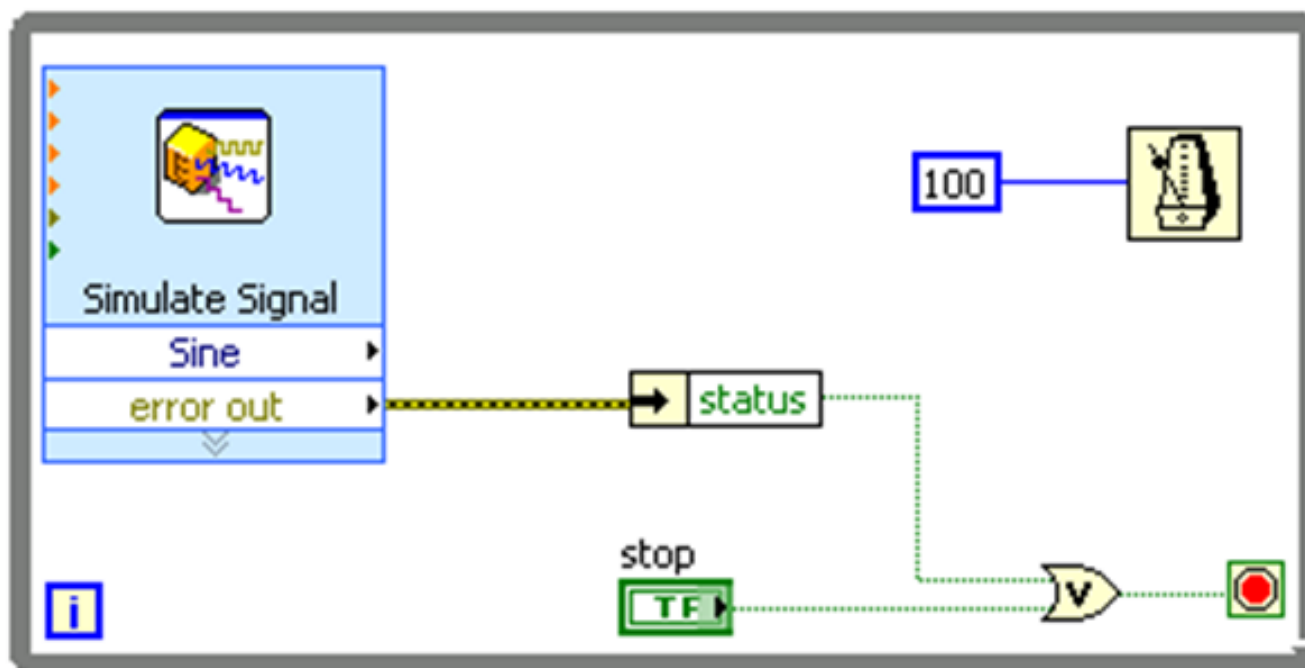
D. Циклы While – Tunnels (Туннели)

- Через туннели данные поступают в структуры и выводятся из структур
- Туннели окрашиваются в цвет типа данных, подключенных к туннелям
- Данные выводятся из цикла после завершения выполнения цикла
- Хотя данные проходят в цикл через туннели, цикл выполняется только после того, как данные поступили во все туннели



D. Циклы While – Контроль и обработка ошибок

Используйте кластер ошибок в цикле While для завершения выполнения цикла при обнаружении ошибки



Упражнение 4-2

Auto Match VI

Используйте цикл While и терминал итераций, пропускайте данные через туннель.

GOAL

ЦЕЛЬ

Упражнение 4-2

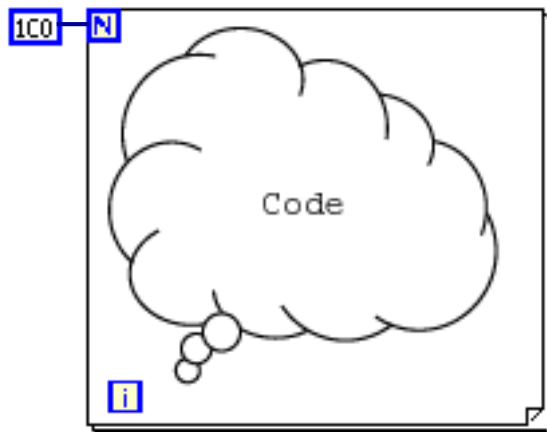
Auto Match VI

- Сколько раз обновляется индикатор итераций? Почему?

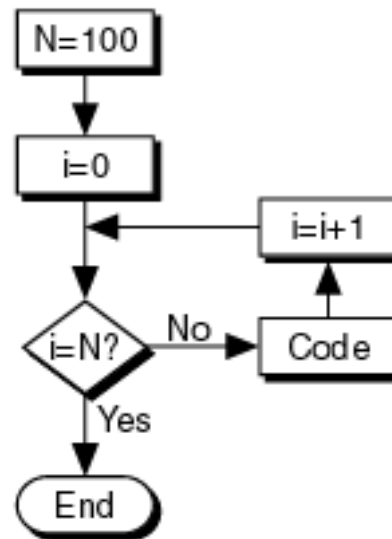
ДИСКУССИЯ

DISCUSSION

Е. Циклы For



LabVIEW For Loop




Flowchart

```
N=100;  
i=0;  
Until i=N:  
    Repeat (code;i=i+1);  
End;
```

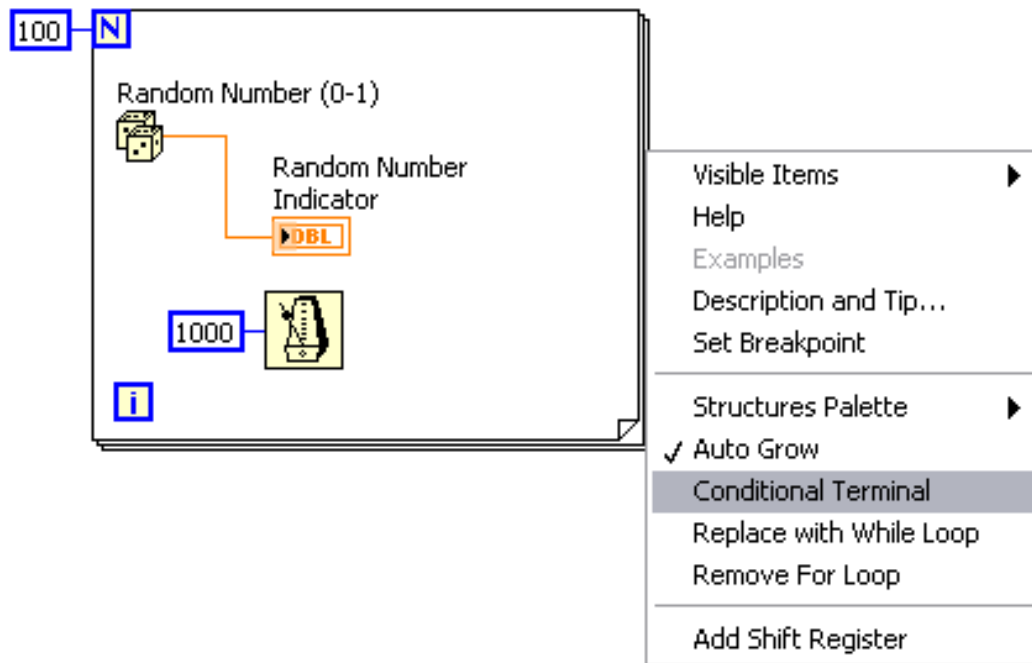
Pseudo Code

Е. Циклы For

- Создайте цикл For таким же способом, как и цикл While
- Если нужно заменить уже существующий цикл While циклом For, щелкните правой кнопкой мыши по границе цикла While и выберите в контекстном меню **Replace with For Loop**
- Значение терминала Count (входной терминал)  указывает, сколько раз будет повторяться фрагмент блок-диаграммы, заключенный в цикл

Е. Циклы For – Conditional Terminal (терминал условия)

Вы можете добавить в цикл For терминал условия для конфигурирования останова выполнения цикла по некоторому булевскому условию или при возникновении ошибки

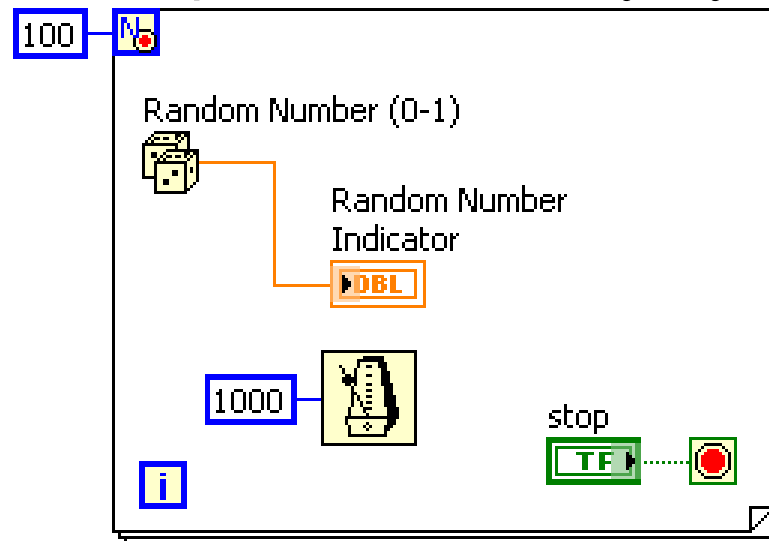


Е. Циклы For – Conditional Terminal (терминал условия)

Цикл For, сконфигурированный для выхода по условию содержит:

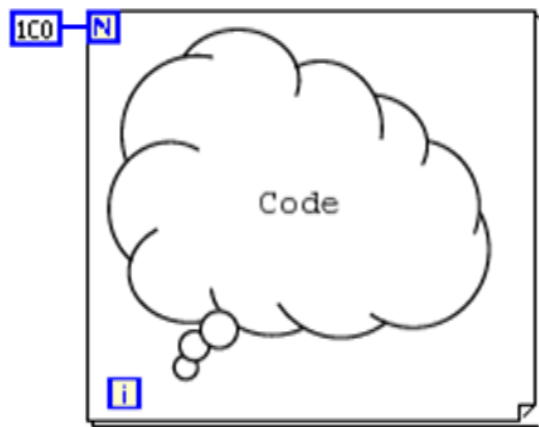
Красный значок рядом с терминалом Count

Терминал условия в правом нижнем углу



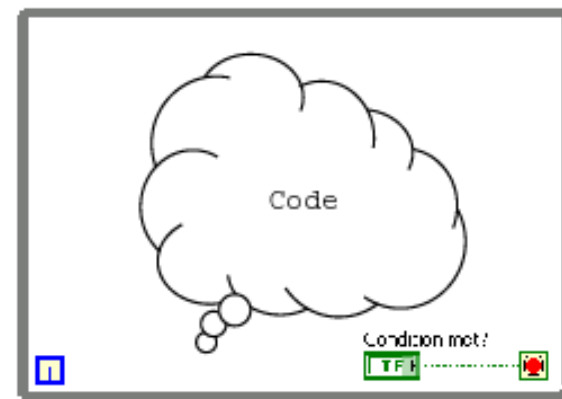
Е. Сравнение циклов For и While

For Loop



- Выполняется заданное число раз, если не добавлен терминал выхода по условию
- Может ни разу не выполняться
- По умолчанию через туннели выводятся массивы данных

While Loop



- Завешается выполнение, только если значение на терминале условия соответствует выбранному условию
- Выполняется не менее одного раза
- По умолчанию через туннели выводятся последние значения

Е. Циклы For – Преобразование чисел

- Количество итераций цикла For должно быть задано неотрицательным целым числом (integer)
- Если вы подключите к терминалу Count число с плавающей точкой двойной точности, LabVIEW преобразует его в 32-битное целое ближайшее большее число со знаком

Double-Precision
Floating Point

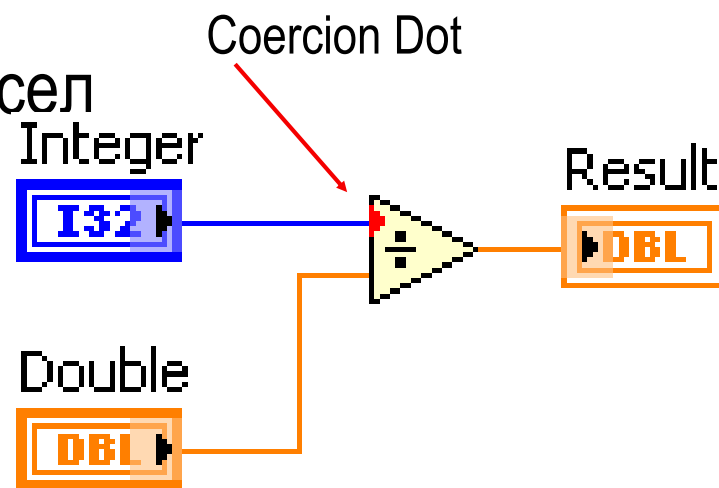


32-Bit Signed Integer



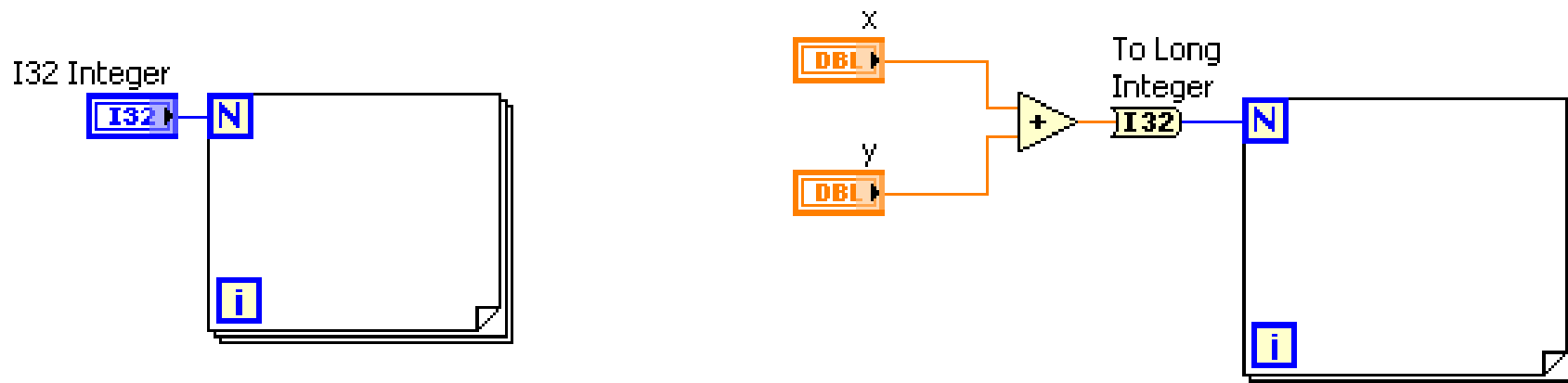
Е. Циклы For – Преобразование чисел

- Обычно при подключении ко входам функции чисел разных форматов, функция возвращает число большего или более широкого формата
- LabVIEW выбирает формат чисел с большим числом бит
- Однако терминал Count цикла Loop всегда приводит подключенное к терминалу число к 32-битному целому со знаком



Е. Циклы For – Преобразование чисел

- Избегайте приведения типов, чтобы улучшить производительность
 - Выбирайте подходящий тип данных
 - Выполняйте преобразование к соответствующему типу данных программным способом



Упражнение 4-3

Тема: Цикл While или цикл For

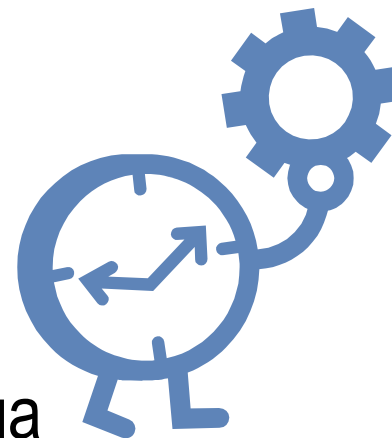
ДОМАШНЕЕ ЗАДАНИЕ

Понять, когда нужно использовать цикл While, а когда цикл For.

GOAL

ЦЕЛЬ

Е. Тактирование VI



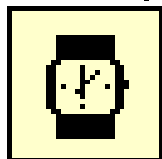
Почему нужно тактировать VI?

- Чтобы управлять частотой выполнения цикла
- Чтобы предоставить процессору время для выполнения других задач, например, обслуживание интерфейса пользователя

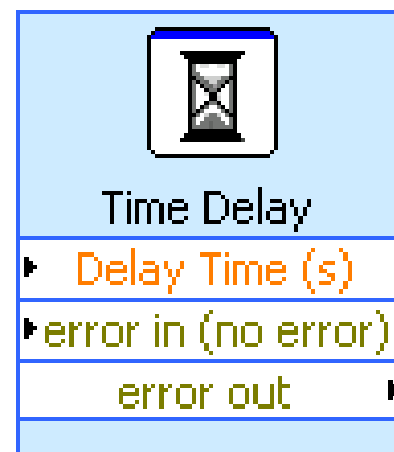
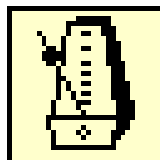
Г. Задание времени VI – функции ожидания

- Функции ожидания внутри цикла позволяют VI «засыпать» на заданное время
- Позволяют процессору обращаться к другим задачам во время ожидания
- Используют миллисекундный таймер операционной системы

Wait (ms)



Wait Until
Next ms Multiple



Г. Задание времени VI – Elapsed Time Express VI

- Определяет, сколько времени прошло после некоторой точки в VI
- Отслеживает время, пока VI продолжает выполняться
- Не предоставляет процессору времени для выполнения других задач



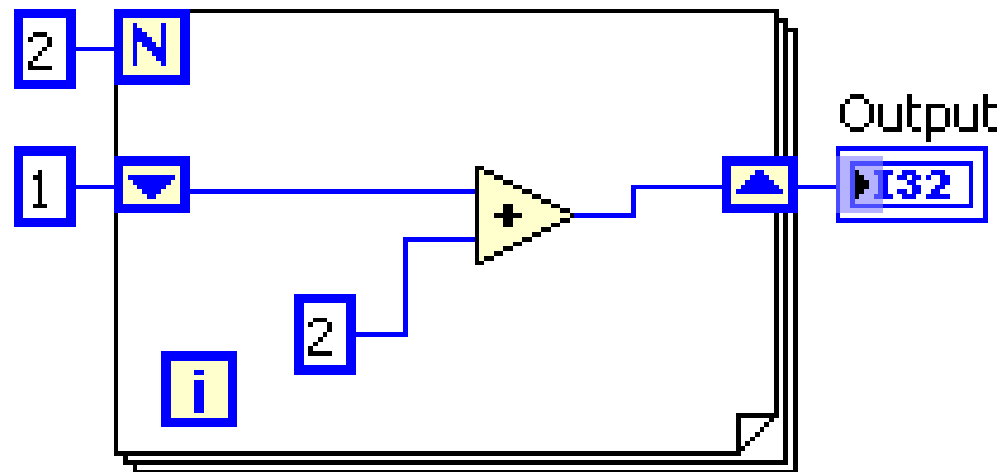
Wait Chart VI

Определите общие признаки и различия использования функций Wait и Elapsed Time Express VI для тактирования программы.

ДЕМОНСТРАЦИЯ

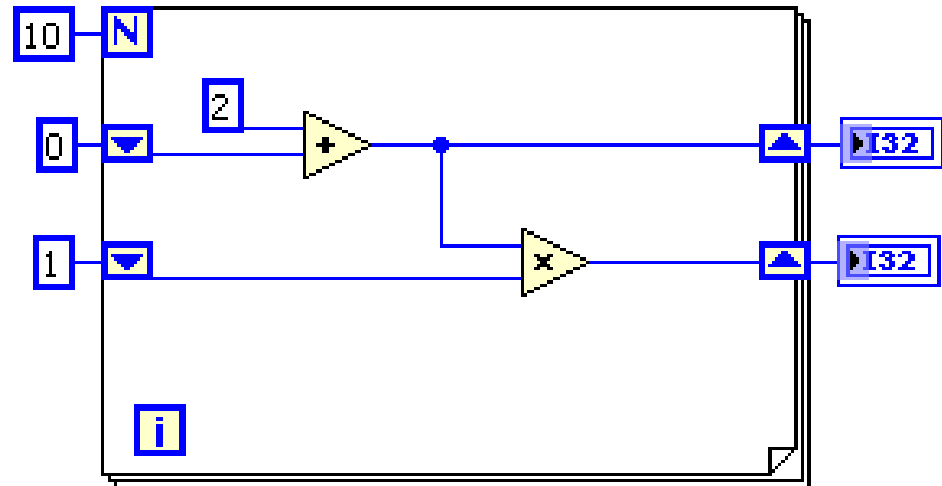
G. Передача данных между итерациями

- При использовании циклов в программе часто требуется знать значения данных, полученных в предыдущей итерации цикла
- Значения данных передаются из текущей итерации цикла в следующую с помощью сдвиговых регистров (Shift Registers)



G. Передача данных между итерациями – Shift Registers

- Щелкните правой кнопкой по границе и выберите в контекстном меню **Add Shift Register**
- Правый сдвиговый регистр запоминает данные, полученные в текущей итерации
- Левый сдвиговый регистр предоставляет данные в начале следующей итерации



G. Передача данных между итерациями – Инициализация

Run once

VI finishes

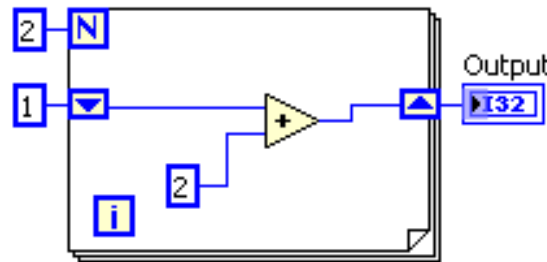
Run again

Block Diagram

1-й запуск

2-й запуск

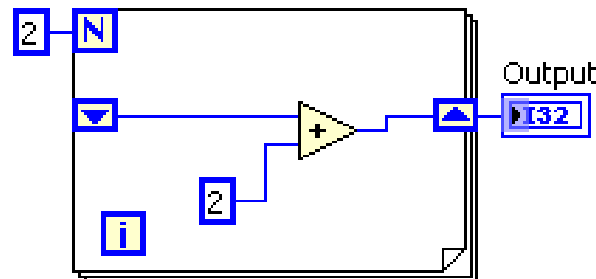
Инициализированный
сдвиговый
регистр



Output = 5

Output = 5

Неинициализированный
сдвиговый
регистр

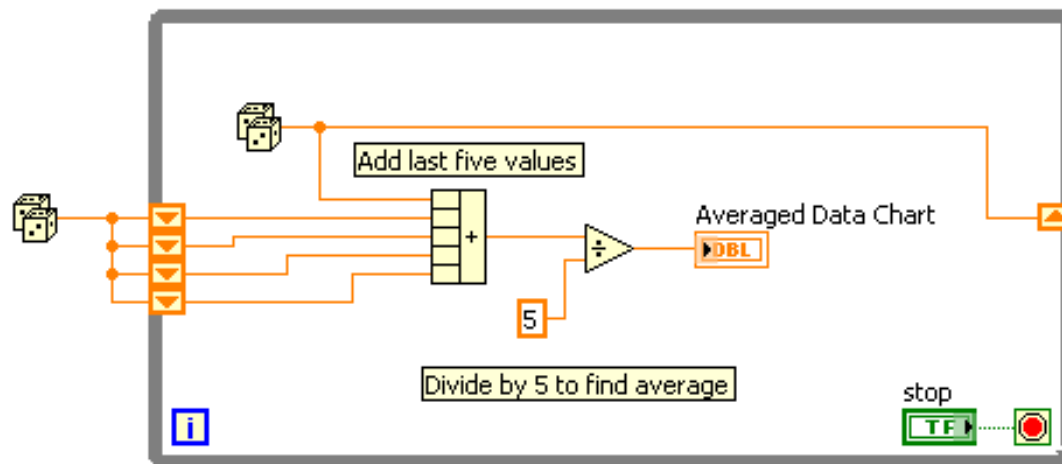


Output = 4

Output = 8

G. Передача данных между итерациями – Stacked Shift Registers

- В стеке сдвиговых регистров запоминаются данные, полученные в нескольких предыдущих итерациях и передаваемые в следующие итерации
- Щелкните правой кнопкой по левому сдвиговому регистру и выберите в контекстном меню **Add Element**



Упражнение 4-4

Average Temperature VI

ДОМАШНЕЕ ЗАДАНИЕ

Используйте цикл For и сдвиговые регистры для усреднения данных.

GOAL

ЦЕЛЬ

Упражнение 4-4

Average Temperature VI

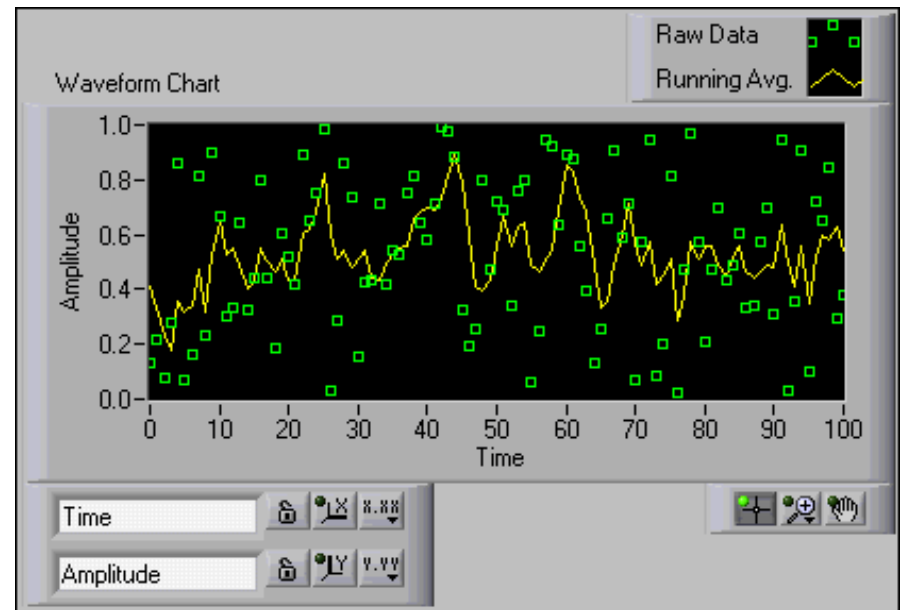
- Вы рассчитали среднее значение 3-х последних отсчетов температуры. Как модифицировать VI для вычисления среднего значения 5-ти последних отсчетов температуры?

ДИСКУССИЯ

DISCUSSION

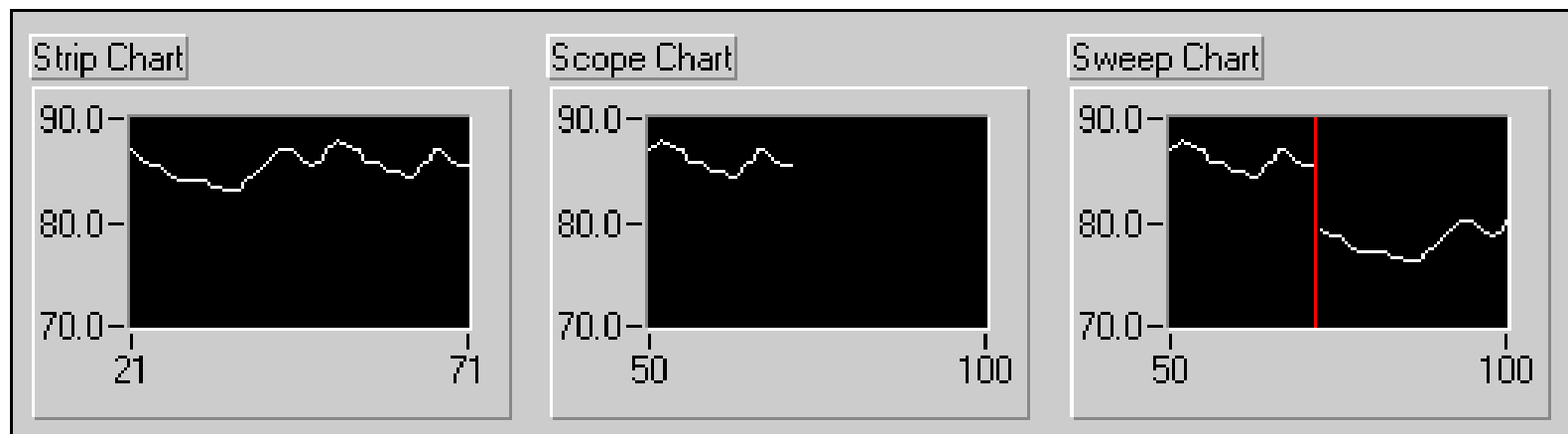
Н. Отображение графиков данных – Waveform Chart (Диаграммы сигналов)

- Специальный тип числовых индикаторов для отображения одного или более графиков данных, обычно собираемых с постоянной частотой
- Визуализируют один или несколько графиков

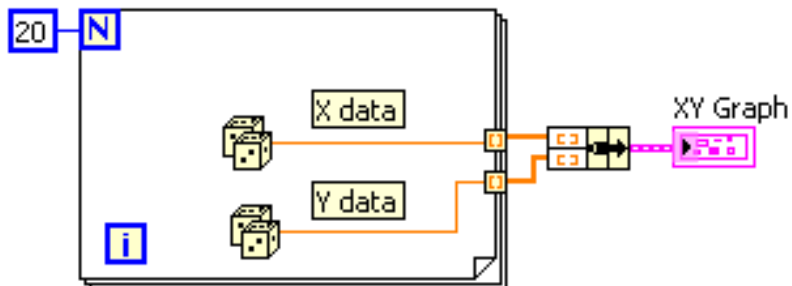
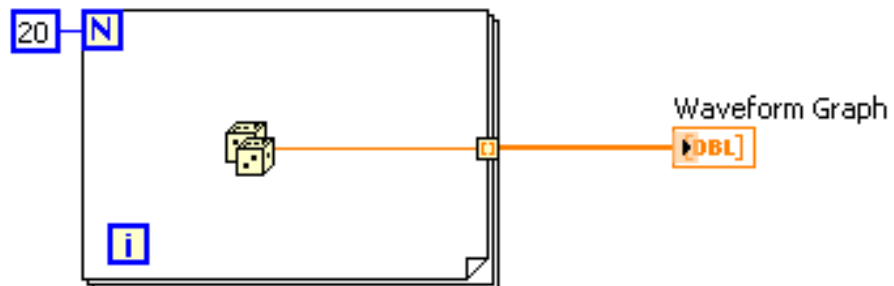
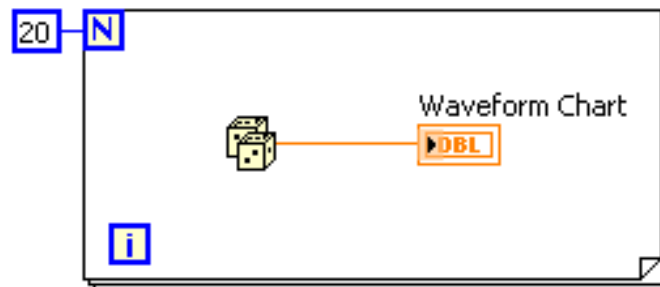
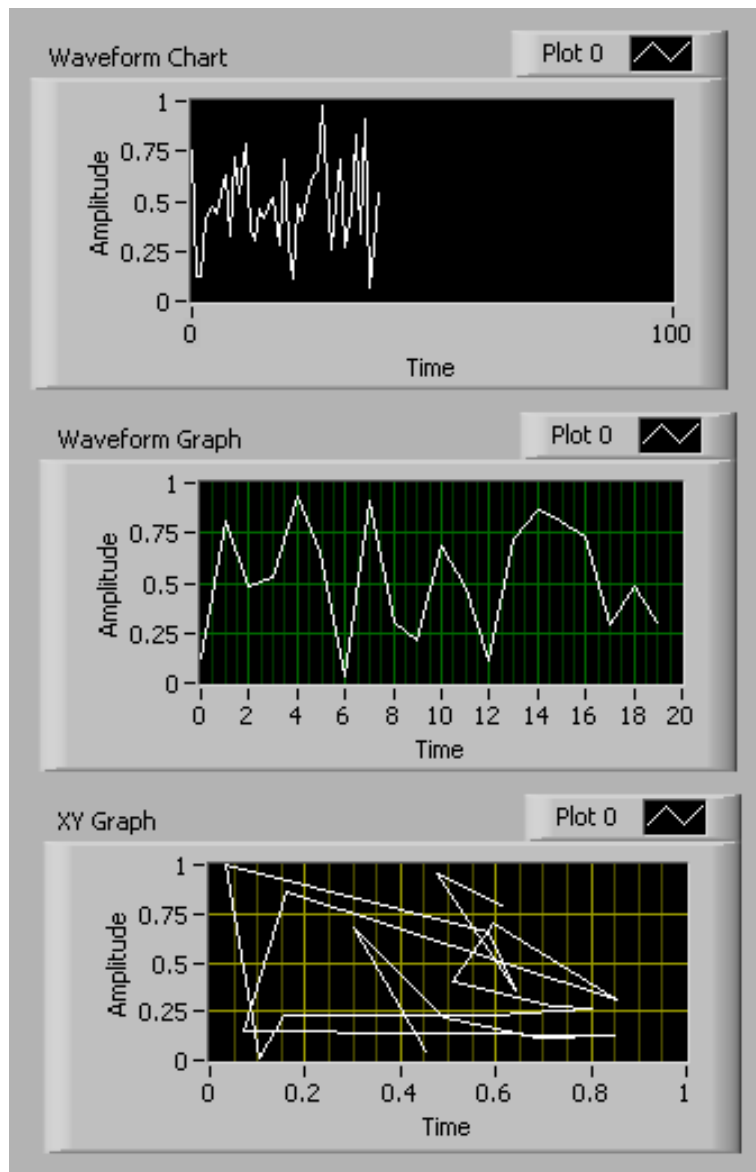


Н. Отображение графиков данных – Режимы обновления диаграмм (Chart Update Modes)

- Щелкните правой кнопкой по графику диаграмм и выберите в контекстном меню **Advanced»Update Mode**
- Режим обновления по умолчанию Strip chart (ленточная диаграмма)
- Режимы Scope chart (осциллографический) and Sweep chart (отображение с замещением) отображают графики значительно быстрее, чем режим Strip chart

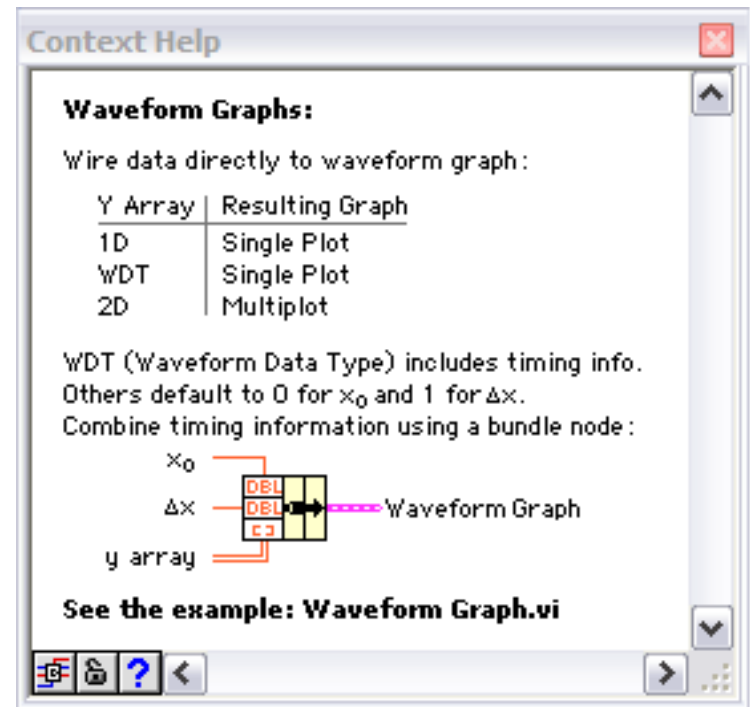


Н. Отображение графиков данных



Н. Отображение графиков данных – Waveform Graphs (Осциллограммы сигналов)

Используйте окно **Context Help** для того, чтобы объяснить, как подключать несколько источников данных к Waveform Graphs и XY Graphs



Упражнение 4-5

Temperature Multiplot VI

Выведите на один график Waveform Chart несколько наборов данных и настройте вид отображения.

GOAL

ЦЕЛЬ

Упражнение 4-5

Temperature Multiplot VI

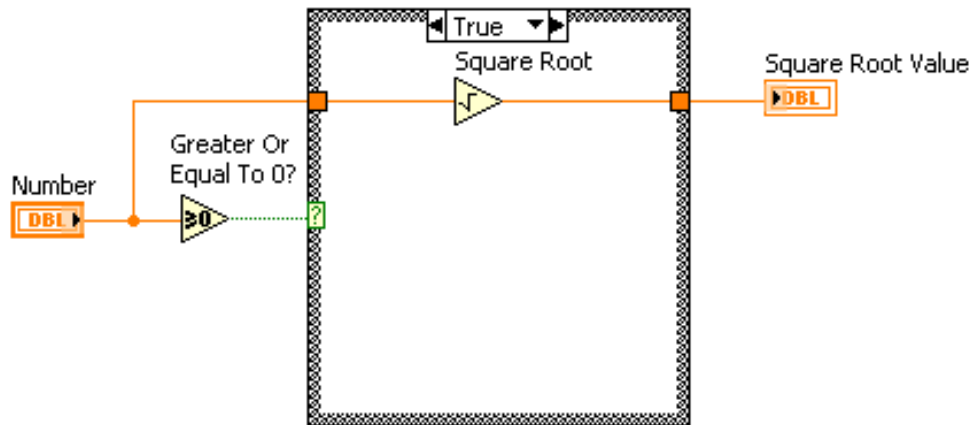
- В этом эксперименте какой тип графика лучше использовать - Chart или Graph?

ДИСКУССИЯ

DISCUSSION

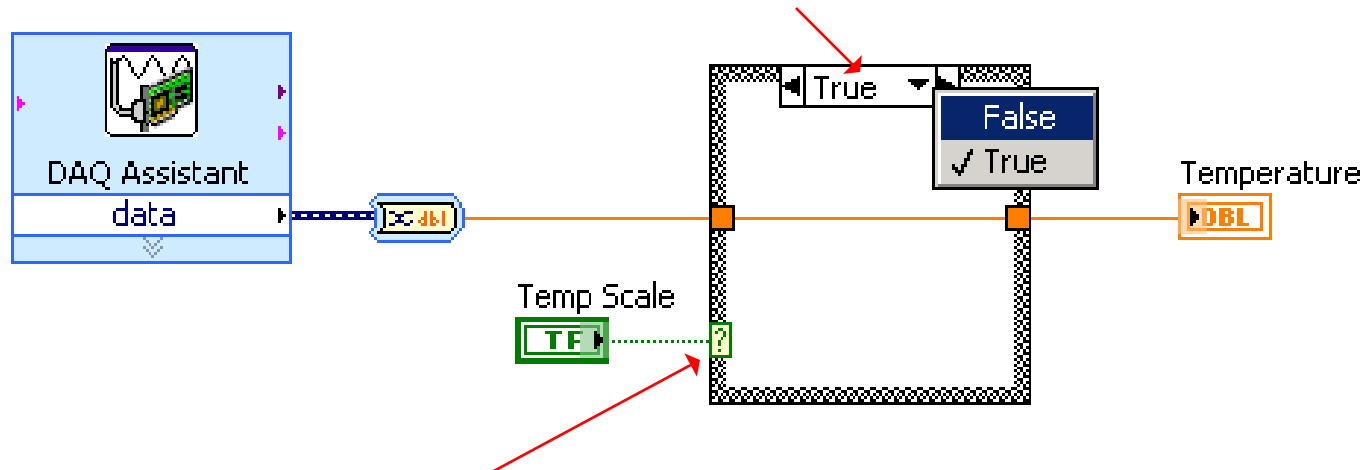
I. Структуры Case

- Состоит из двух или более субдиаграмм или вариантов
- В любой момент времени отображается и выполняется только одна субдиаграмма (вариант)
- Значение входа определяет, какая субдиаграмма будет выполняться
- Структура аналогична оператору `case` или конструкции `if...then...else` в текстовых языках программирования



I. Структуры Case

- Метка селектора выбора (Case Selector): содержит имя текущего варианта и кнопки декремента и инкремента по бокам метки



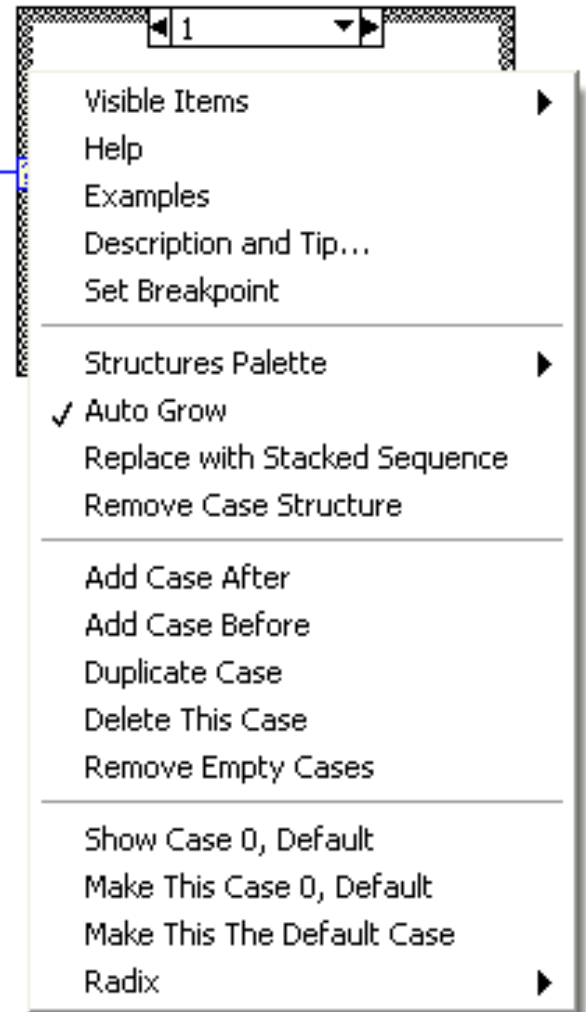
- Терминал селектора: подключите вход или селектор для того, чтобы определить, какой вариант будет выполняться

I. Структуры Case – Default Case (вариант по умолчанию)

- В структуре Case можно задать вариант по умолчанию
 - Если вы определили варианты 1, 2 и 3, а поступило значение 4 структура Case выполняет вариант по умолчанию
- Щелкните правой кнопкой по границе структуры Case, для того , чтобы добавить, удалить, продублировать или переупорядочить варианты и выбрать вариант по умолчанию

Numeric

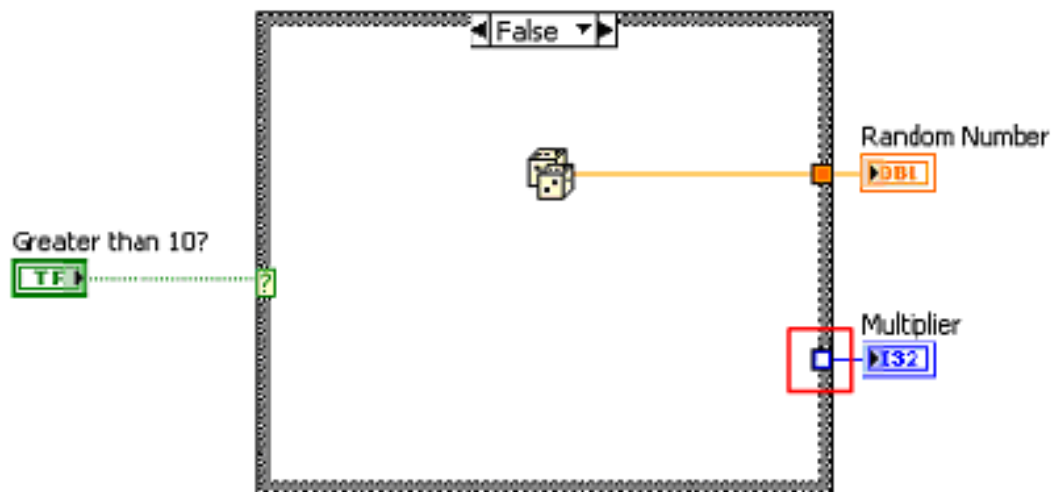
I32



I. Структуры Case – входные и выходные туннели

Вы можете создавать множество входных и выходных туннелей

- Входы доступны для всех необходимых вариантов
- Каждый выходной туннель должен быть определен в каждом варианте



I. Структуры Case – Use Default if Unwired

(использовать значение по умолчанию, если не подключен)

Значения по умолчанию:

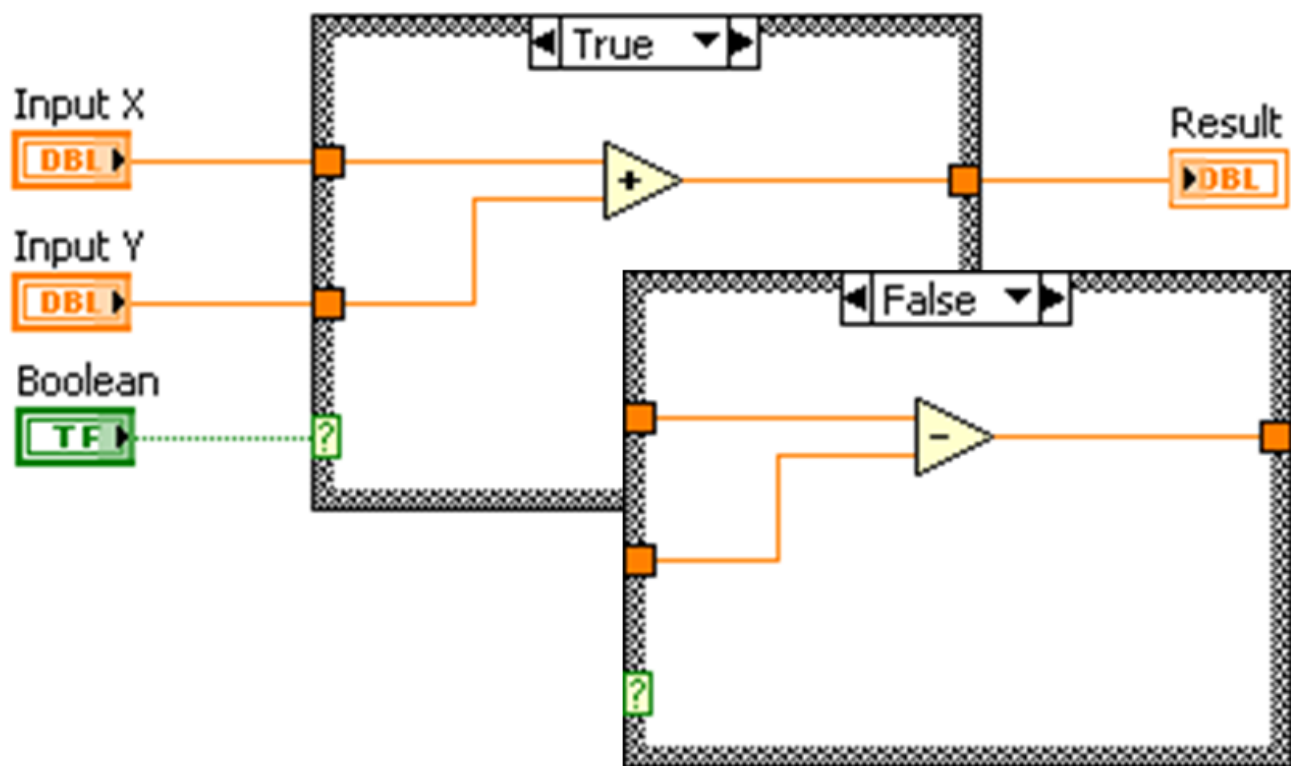
Data Type	Default Value
Numeric	0
Boolean	FALSE
String	Empty

Избегайте использование **Use Default if Unwired** для туннелей структуры Case

- Усложняет код
- Затрудняет отладку кода

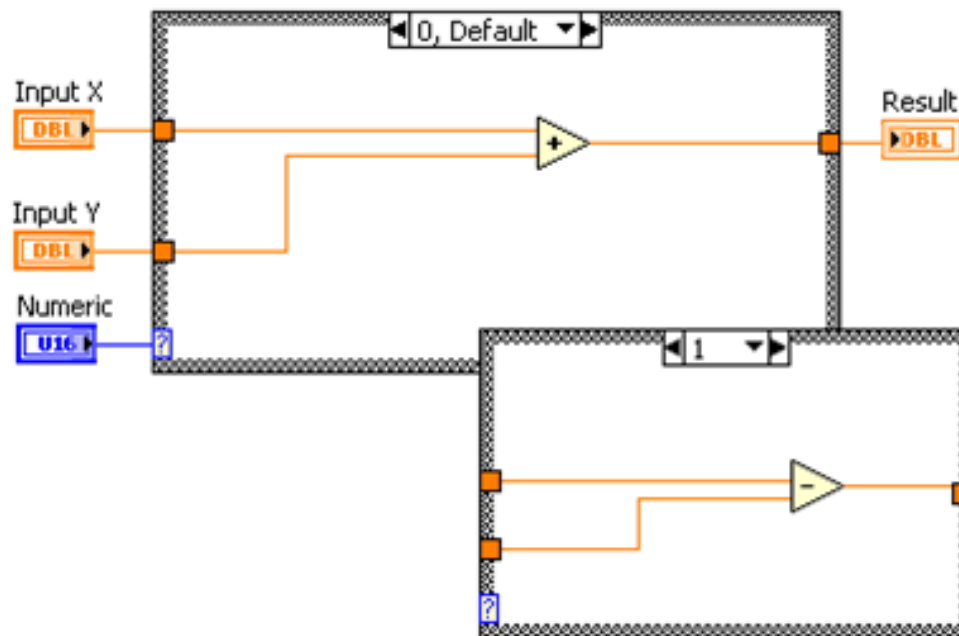
I. Структуры Case – Boolean

Булевский вход создает два варианта: True and False



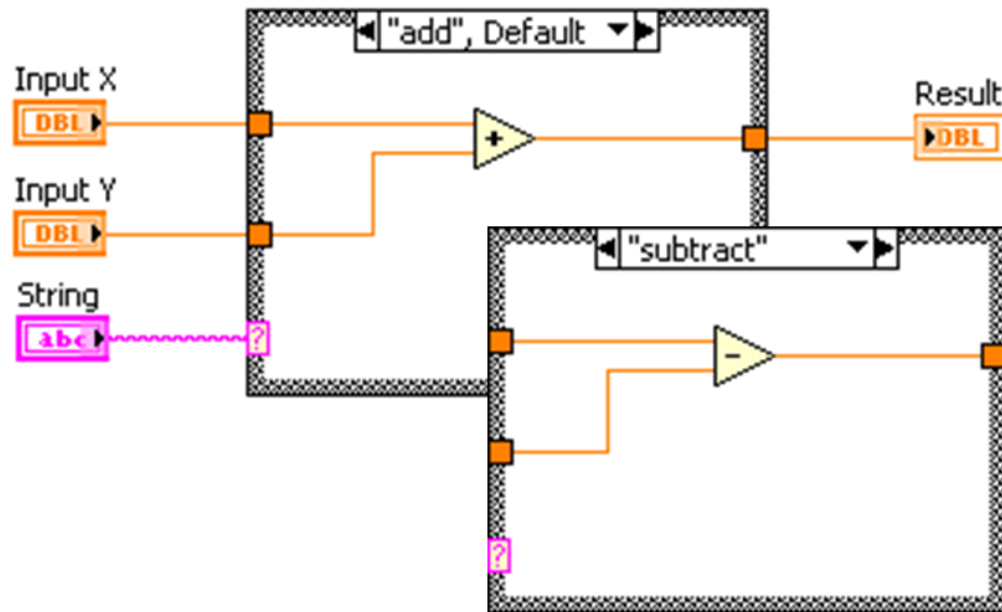
I. Структуры Case – Integer

- По мере необходимости добавляйте вариант для каждого значения integer
- Целые значения, для которых не определен вариант, используют вариант по умолчанию



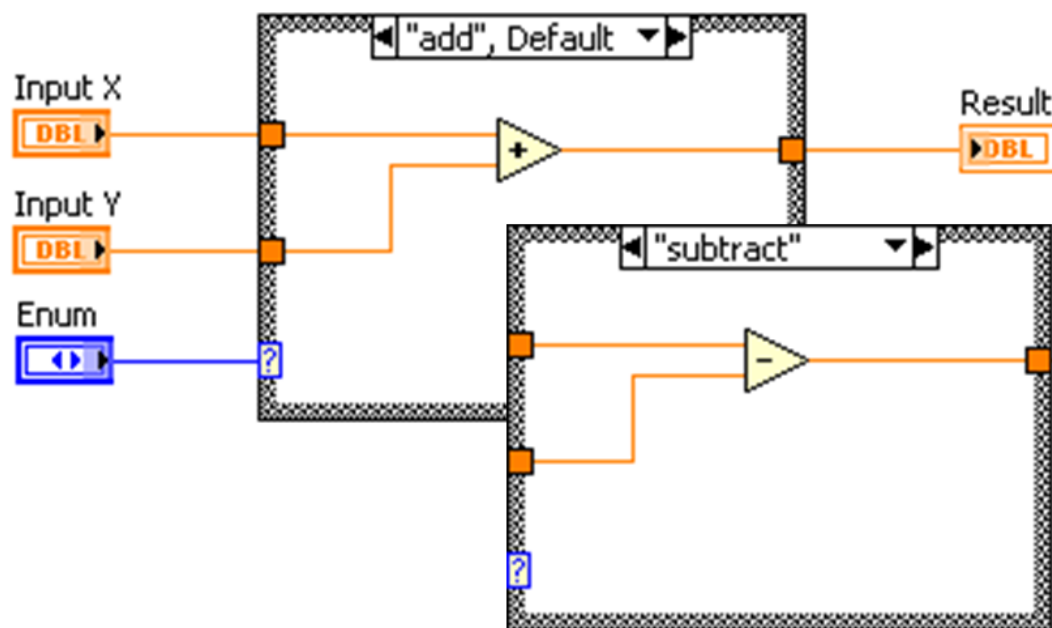
I. Структуры Case – String

- По мере необходимости добавляйте вариант для каждого значения string
- Значения строк, для которых не определен вариант, используют вариант по умолчанию



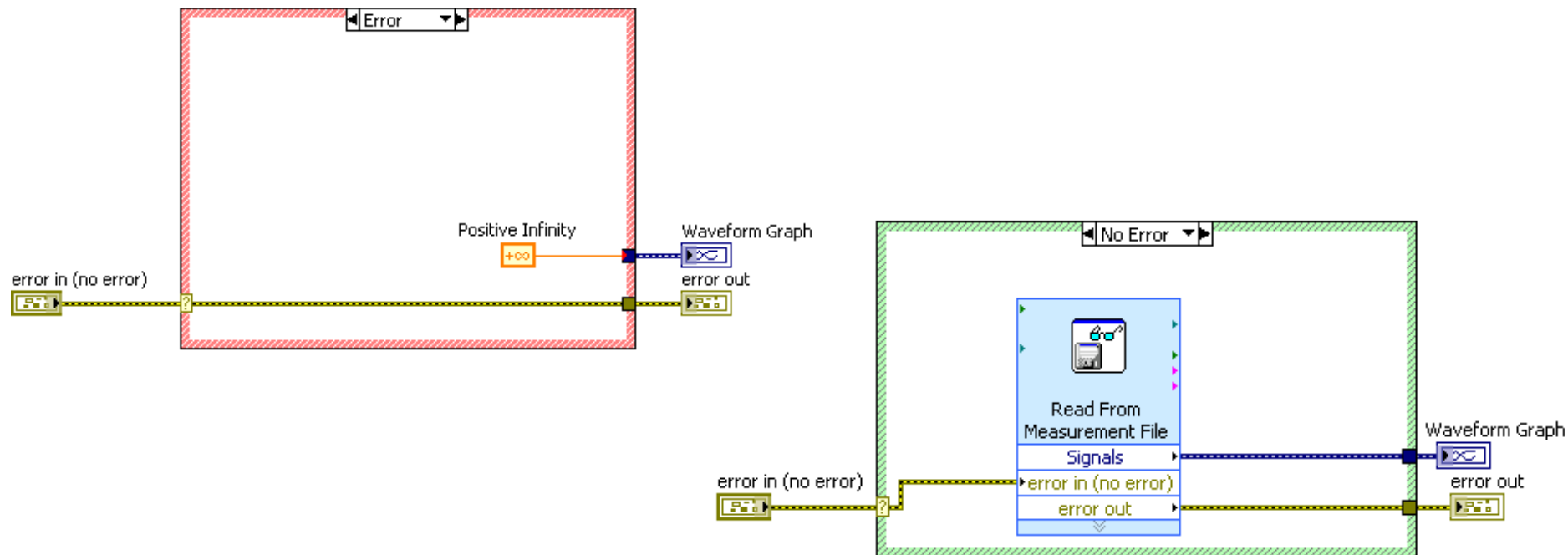
I. Структуры Case – Enum

- Дают пользователю список элементов, из которого выбираются варианты
- В переключателе селектора выбора отображаются варианты для каждого значения элемента управления типа Enum



I. Структуры Case – Контроль и обработка ошибок

Используйте в VI структуры Case для того, чтобы код выполнялся, если ошибок нет, и не выполнялся – при обнаружении ошибки



Упражнение 4-6

Determine Warnings VI

Модифицируйте VI, чтобы использовать структуру Case для принятия решения в программе

GOAL

ЦЕЛЬ

Упражнение 4-6

Determine Warnings VI

- Что будет, если все значения равны 10? Как можно это разрешить?

ДИСКУССИЯ

DISCUSSION

Заключение – Контрольный вопрос

1. Что позволяет отличить на блок-диаграмме элемент управления от элемента индикации?
 - a) Caption (Название)
 - b) Location (Положение)
 - c) Label (Метка)
 - d) Value (Значение)

Заключение – Ответ на контрольный вопрос

1. Что позволяет отличить на блок-диаграмме элемент управления от элемента индикации?
 - a) Caption (Название)
 - b) Location (Положение)
 - c) **Label (Метка)**
 - d) Value (Значение)

Заключение – Контрольный вопрос

2. Какая структура должна выполняться по крайней мере один раз?
- a) While Loop
 - b) For Loop

Заключение – Ответ на контрольный вопрос

2. Какая структура должна выполняться по крайней мере один раз?
- a) **While Loop**
 - b) For Loop

Заключение – Контрольный вопрос

3. Какой объект доступен только на блок-диаграмме?
- a) Control (Элемент управления)
 - b) Constant (Константа)
 - c) Indicator (Элемент индикации)
 - d) Connector Pane (Панель подключения)

Заключение – Ответ на контрольный вопрос

3. Какой объект доступен только на блок-диаграмме?
- a) Control (Элемент управления)
 - b) Constant (Константа)**
 - c) Indicator (Элемент индикации)
 - d) Connector Pane (Панель подключения)

Заключение – Контрольный вопрос

4. Если вы щелкнули по булевскому элементу управления, какое его механическое действие является причиной изменения булевского значения из состояния False в состояние True и сохранения в состоянии True, пока вы не отпустите элемент управления и LabVIEW не прочтет это значение?
- a) Switch Until Released
 - b) Switch When Released
 - c) Latch Until Released
 - d) Latch When Released

Заключение – Ответ на контрольный вопрос

4. Если вы щелкнули по булевскому элементу управления, какое его механическое действие является причиной изменения булевского значения из состояния False в состояние True и сохранения в состоянии True, пока вы не отпустите элемент управления и LabVIEW не прочтет это значение?
- a) Switch Until Released
 - b) Switch When Released
 - c) **Latch Until Released**
 - d) Latch When Released

Лекция 5

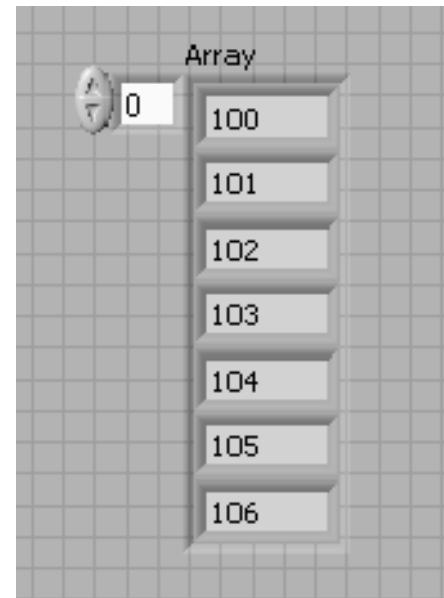
Связываемые данные

ТЕМЫ

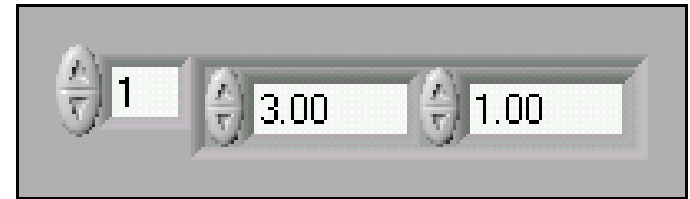
- A. Arrays (массивы)
- B. Clusters (кластеры)
- C. Type Definitions (определители типа)

A. Arrays (Массивы)

- Массивы состоят из элементов и характеризуются размерностью
 - Элементы: данные, из которых состоит массив
 - Размерность: длина, высота или глубина массива
 - Массив может иметь одну или более размерностей и до $(2^{31})-1$ элементов на размерность, если позволяет память
- Рассмотрим использование массивов при работе с наборами однотипных данных и при выполнении повторяющихся вычислений



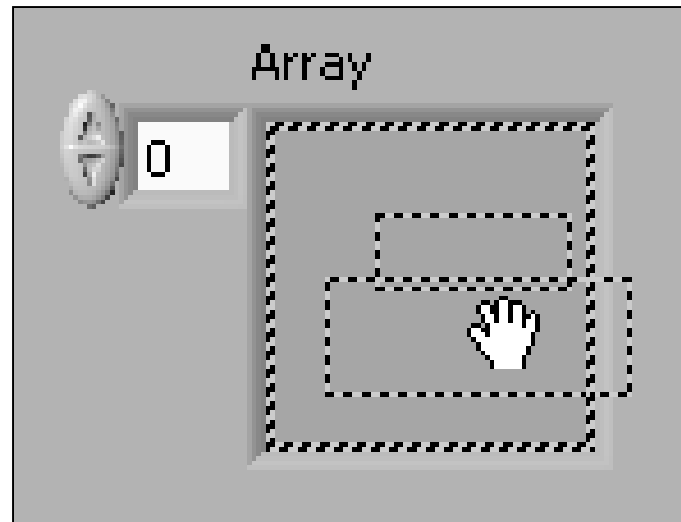
A. Arrays



- В показанном массиве первый элемент (3.00) имеет индекс 1, а второй элемент (1.00) имеет индекс 2
- Элемент с индексом 0 не показан на этом рисунке, поскольку переключателем индекса выбран элемент 1
- Элемент, на который указывает переключатель индекса всегда отображается в верхнем левом углу индикатора элементов

A. Arrays – Создание массивов

1. Поместите контейнер массива на лицевую панель
2. Перетащите объект данных или элемент в контейнер массива



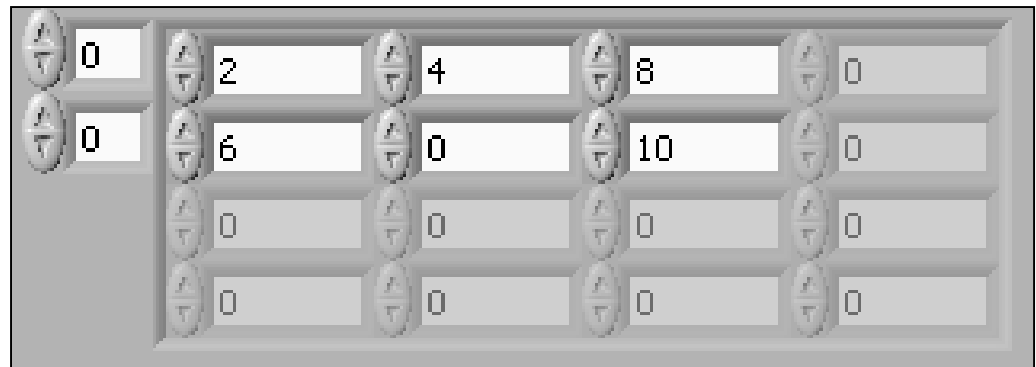
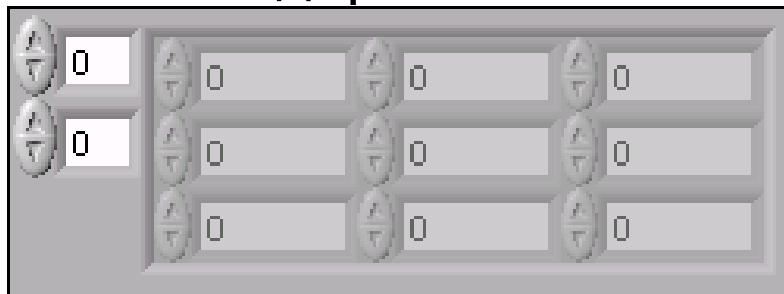
A. Arrays – 2D Array (2-мерный массив)



- Хранит элементы в сетке
- Для нахождения элемента массива необходимы индекс столбца и индекс строки, отсчитываемые с 0
- Для создания на лицевой панели многомерного массива, щелкните правой кнопкой по переключателю индексов и выберите в контекстном меню **Add Dimension**
- Вы можете также растягивать переключатель индексов, пока не получите требуемую размерность

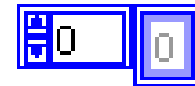
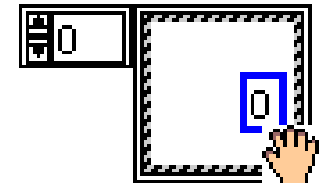
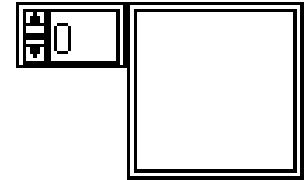
A. Arrays – Initializing (Инициализация)

- Массив можно проинициализировать или оставить не инициализированным
- Для инициализации массивов определите количество элементов в каждой размерности и содержимое каждого элемента
- Неинициализированный массив имеет размерности, но не содержит элементов



A. Arrays – Creating Constants (Создание массива констант)

- Для создания массива констант:
 - Выберите массив констант в палитре **Functions**
 - Поместите контейнер массива на Place блок-диаграмму
 - Поместите константу в контейнер массива
- Вы можете использовать массив констант для хранения неизменных данных или в качестве базы для сравнения с другим массивом
- Массив констант полезен также для ввода данных в subVI

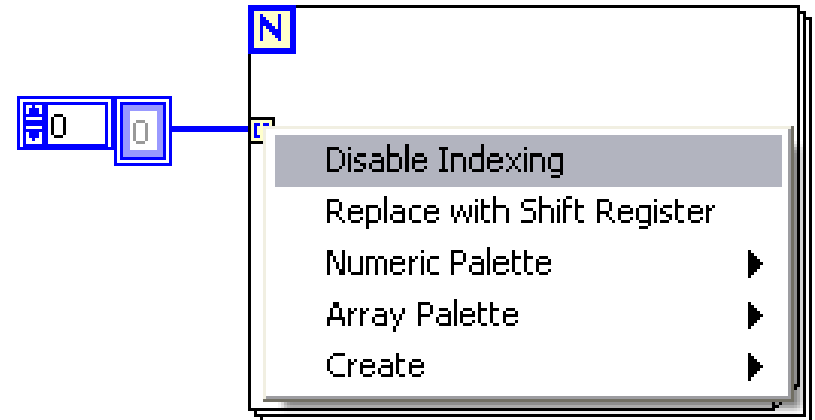


A. Arrays – Auto-indexing (Автоиндексация)

- Если подключить массив ко входу или выходу цикла For или While, то можно связать каждую итерацию цикла с элементом массива путем разрешения автоиндексации туннеля

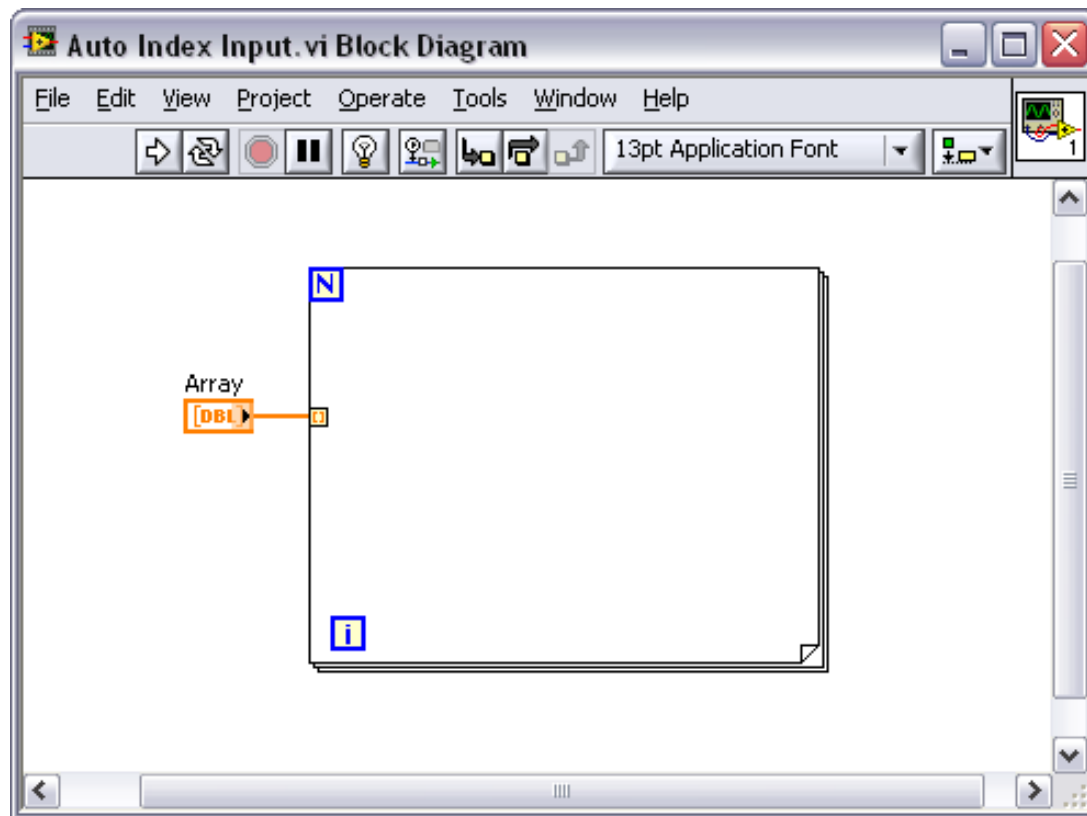


- Внешний вид туннеля при этом изменится с закрашенного квадрата на изображение, помещенное выше, это свидетельствует об автоиндексации



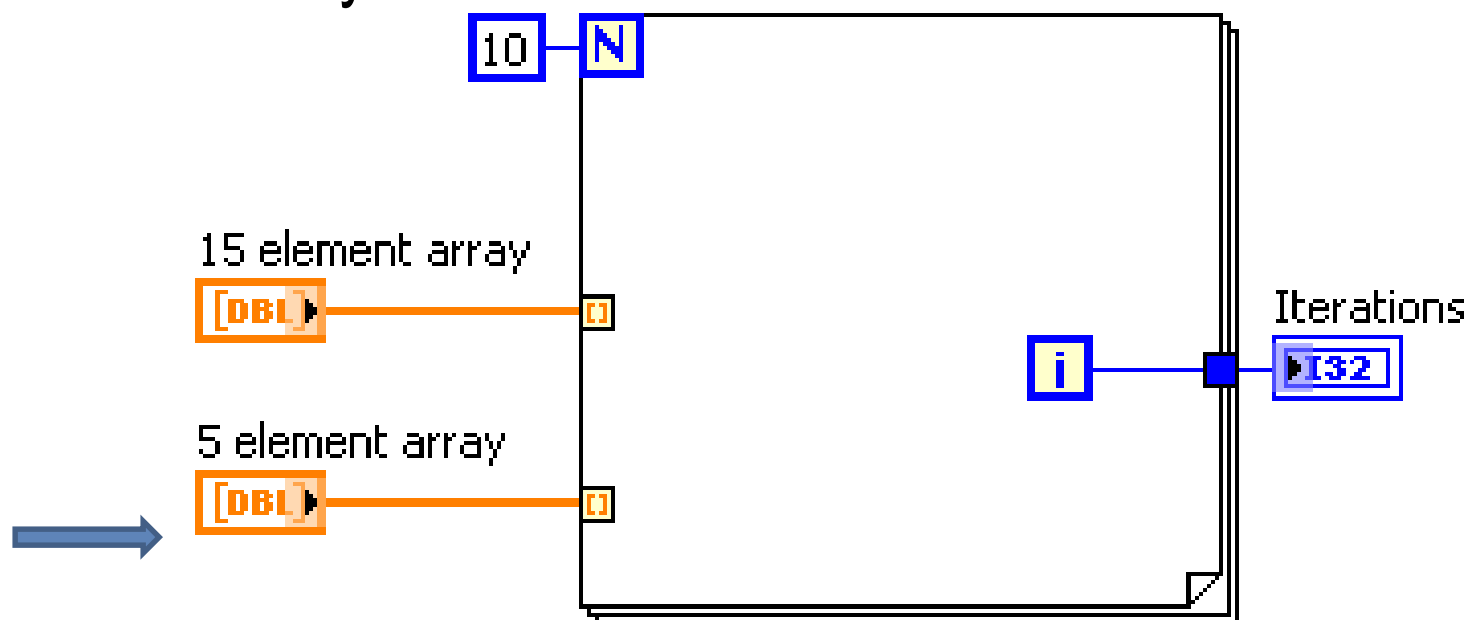
A. Arrays – Автоиндексируемый вход

Количество выполнений
цикла For равно
количеству элементов
в массиве



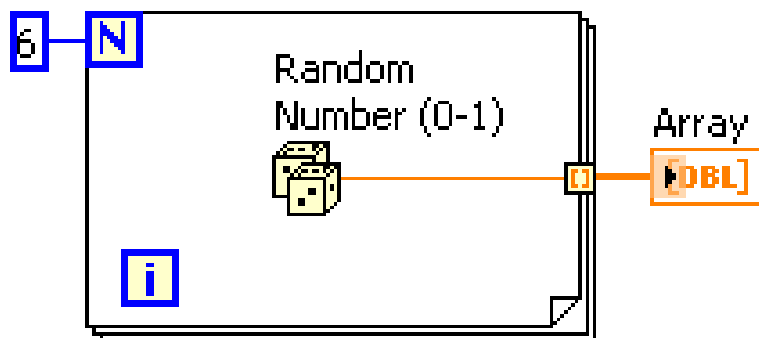
A. Arrays – Автоиндексируемый вход

Если подключен терминал количества итераций Count, а к автоиндексируемым туннелям подключены массивы различного размера, реально количество итераций равно наименьшему из чисел.



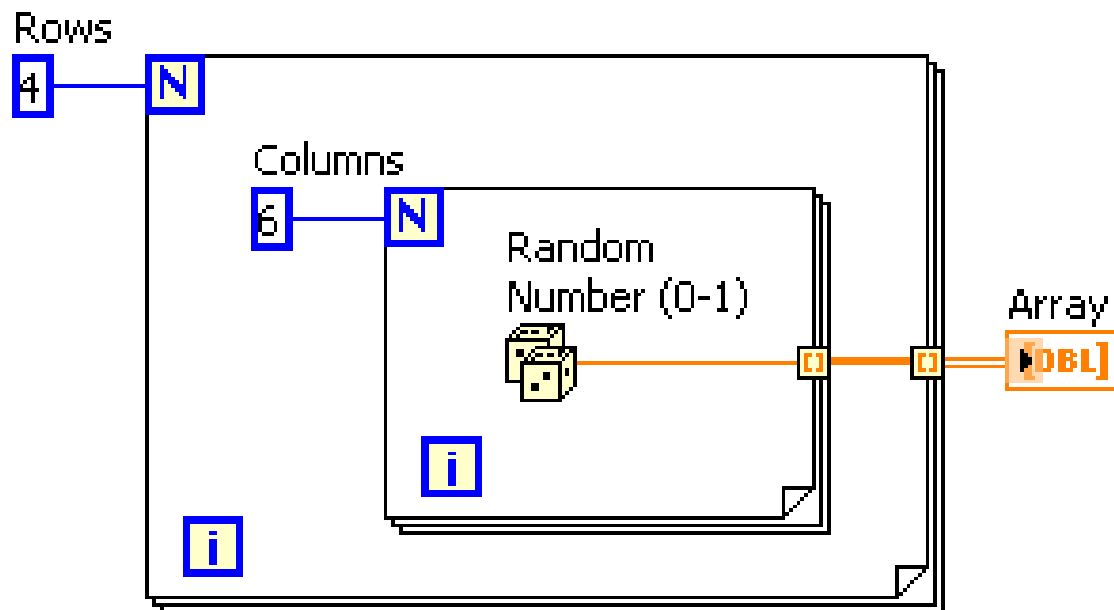
A. Arrays – Автоиндексируемый выход

- Если вы разрешите автоиндексацию массива в выходном туннеле, в выходной массив помещается новый элемент на каждой итерации цикла
- Размер автоиндексируемого выходного массива всегда равен количеству итераций



A. Arrays – Создание 2-мерных массивов (2D Arrays)

Вы можете использовать два цикла For, один внутри другого, для того, чтобы создать 2-мерный массив



Упражнение 5-1

Тема: Манипуляции с массивами

Манипулируйте с массивами, используя различные функции LabVIEW.

GOAL

ЦЕЛЬ

Упражнение 5-1

Тема: Манипуляции с массивами

- Как можно программным способом создать 3-мерный массив?

ДИСКУССИЯ

DISCUSSION

B. Clusters (Кластеры)

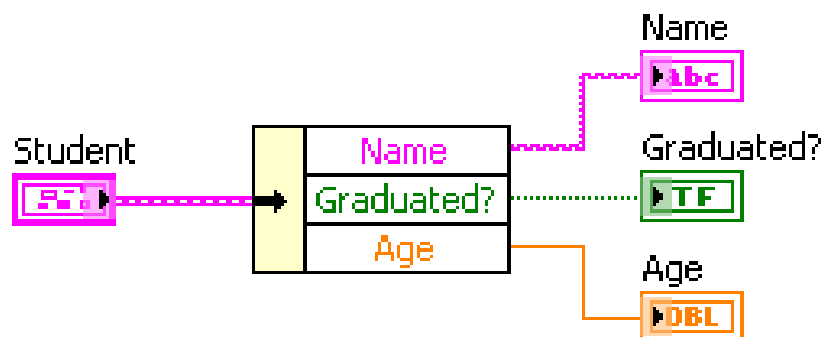
- В кластерах группируют элементы разных типов
- Аналогичны конструкциям record или struct в текстовых языках программирования

Student

Name

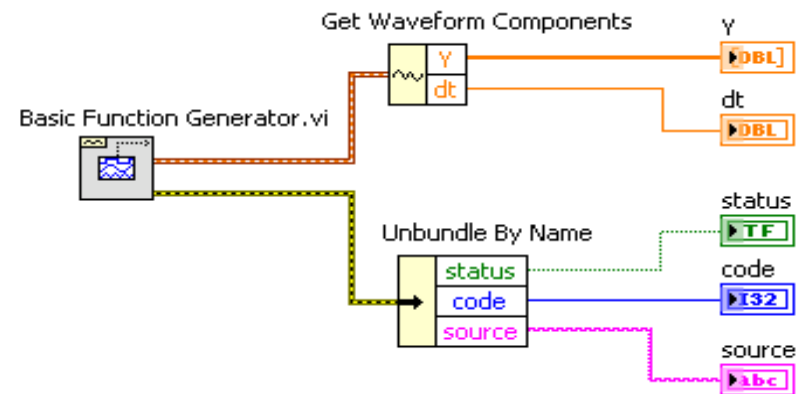
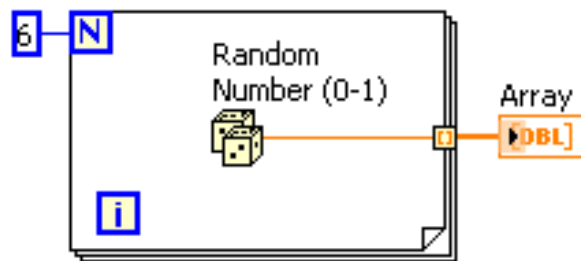
Graduated? Y ☐ N

Age 0



B. Clusters – Сравнение массива и кластера

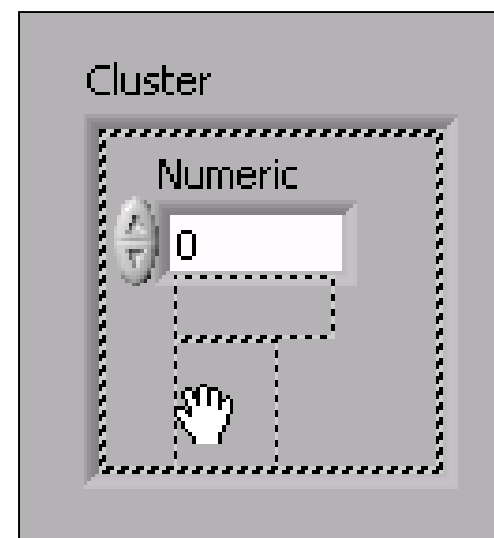
- В отличие от массивов, размер кластера фиксирован
- Кластеры могут содержать компоненты разных типов данных; массивы состоят из компонентов только одного типа данных
- Как и массив, кластер состоит только из элементов управления или только из элементов индикации, но не из тех и других одновременно



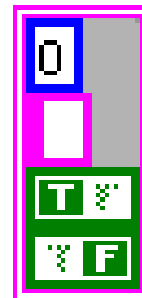
B. Clusters – Создание кластера

На лицевой панели кластер создают следующим образом:

- Поместите на лицевую панель контейнер кластера
- Перетащите объекты данных или элементы, которые могут представлять данные разных типов - числового, булевского, строкового, пути, ссылок, массивы или кластеры элементов управления или индикации в контейнер кластера



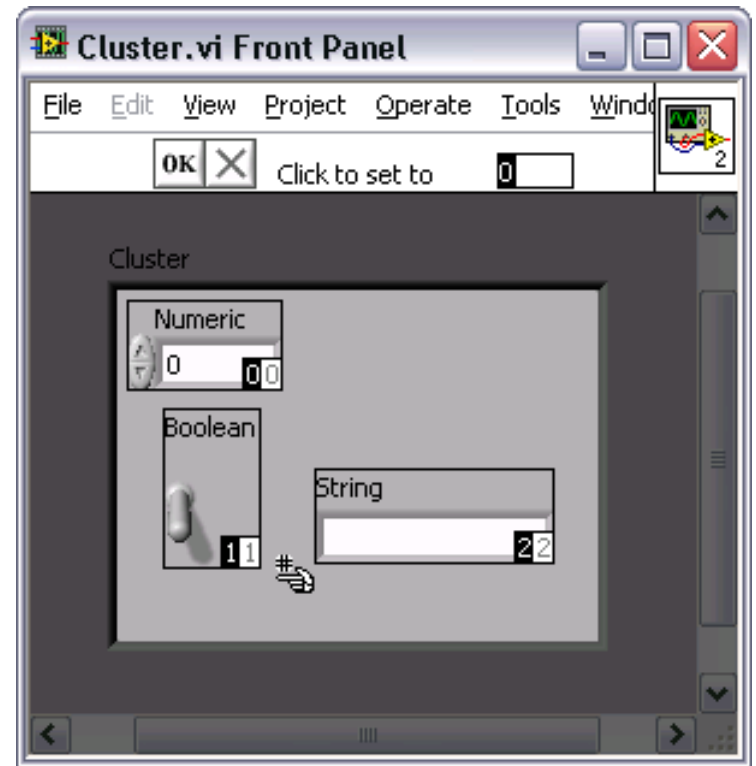
B. Clusters – Константы



- Для создания кластера констант:
 1. Выберите контейнер кластера констант из палитры Functions
 2. Поместите контейнер кластера на блок-диаграмму
 3. Поместите константу в контейнер кластера
- Если у вас есть кластер элементов управления или индикации, щелкните правой кнопкой мыши по кластеру на блок-диаграмме и выберите из контекстного меню **Create»Constant**

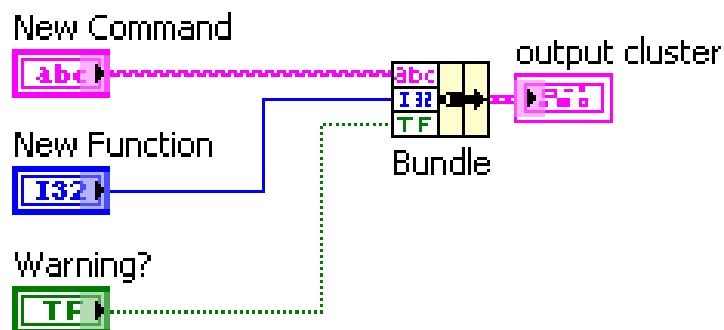
B. Clusters – Порядок элементов

- Элементы кластера упорядочены логически независимо от их места в контейнере
- Вы можете узнавать и изменять порядок элементов в кластере, щелкнув правой кнопкой по границе кластера и выбрав в контекстном меню **Reorder Controls In Cluster**



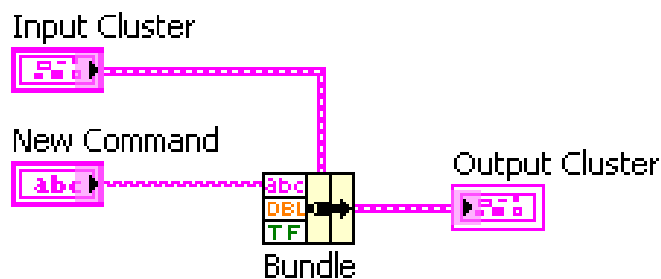
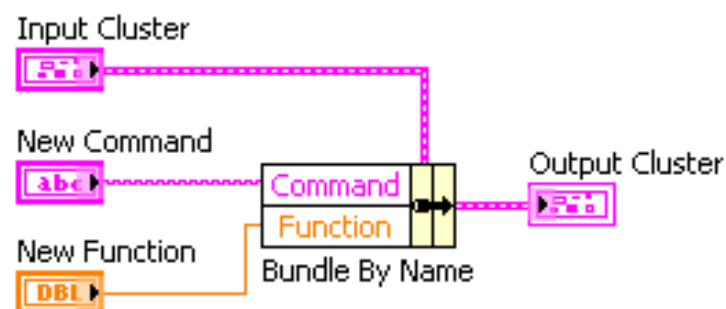
B. Clusters – Сборка в кластер

Для сборки элементов в новый кластер используйте функцию Bundle



B. Clusters – Модификация кластера

Для модификации существующего кластера используйте функции Bundle By Name или Bundle



B. Clusters – Разборка кластера

Для получения отдельных элементов кластера используйте функции Unbundle By Name или Unbundle

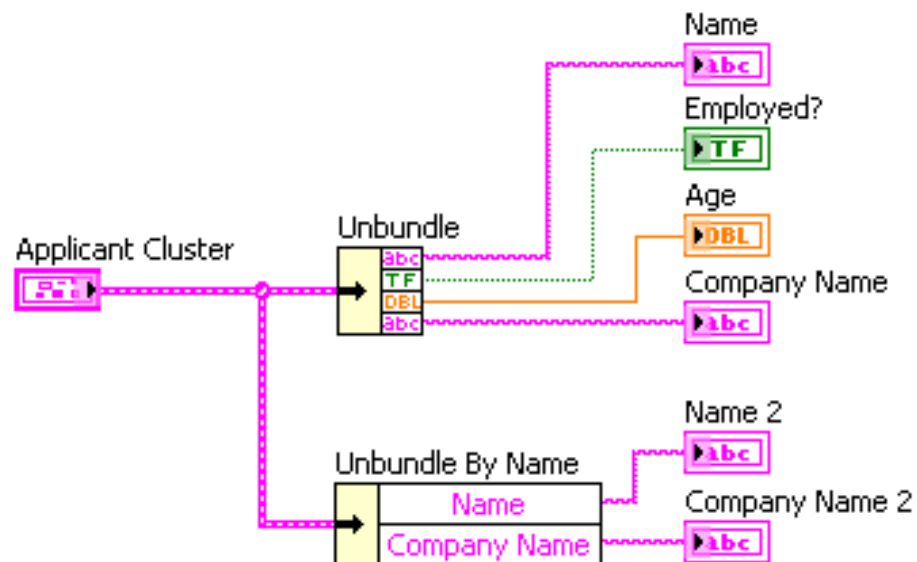
Applicant Cluster

Name

Age

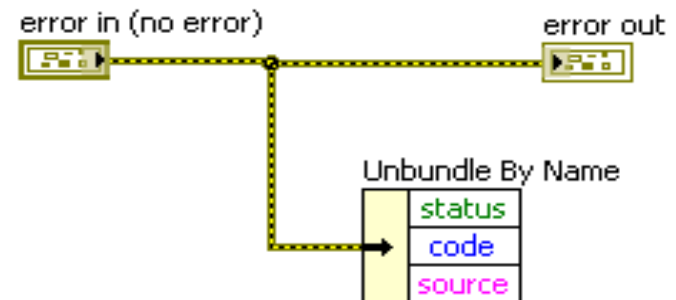
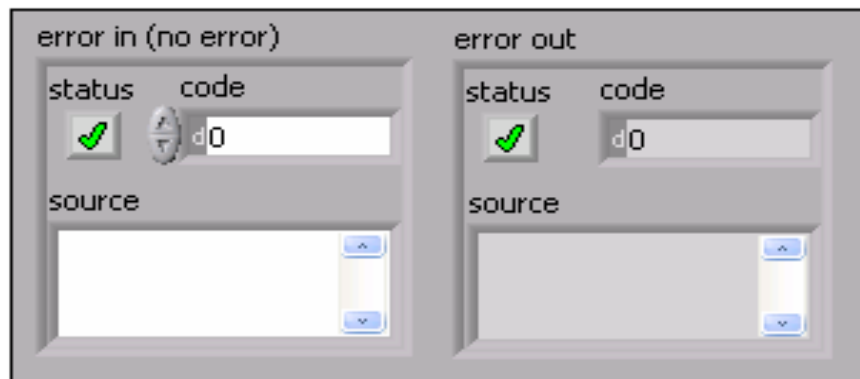
Employed?
Yes ☐
No ☐

Company Name



B. Clusters – Кластеры ошибок

- LabVIEW использует кластеры ошибок для распространения информации об ошибках
- Кластер ошибок состоит из следующих элементов:
 - **status**: булевский элемент, значением которого True сообщается об обнаружении ошибки
 - **code**: 32-битное целое число со знаком служит для идентификации ошибки
 - **source**: строка – идентифицирует место возникновения ошибки



Упражнение 5-2

Тема: Clusters

На лицевой панели создайте кластеры, измените порядок элементов в кластерах, используйте функции для сборки и разборки кластеров.

GOAL

Упражнение 5-2

Тема: Clusters

- Что произойдет, если кластер Small Cluster, в котором первым элементом был числовой элемент, а вторым – булевский, переупорядочить?

ДИСКУССИЯ

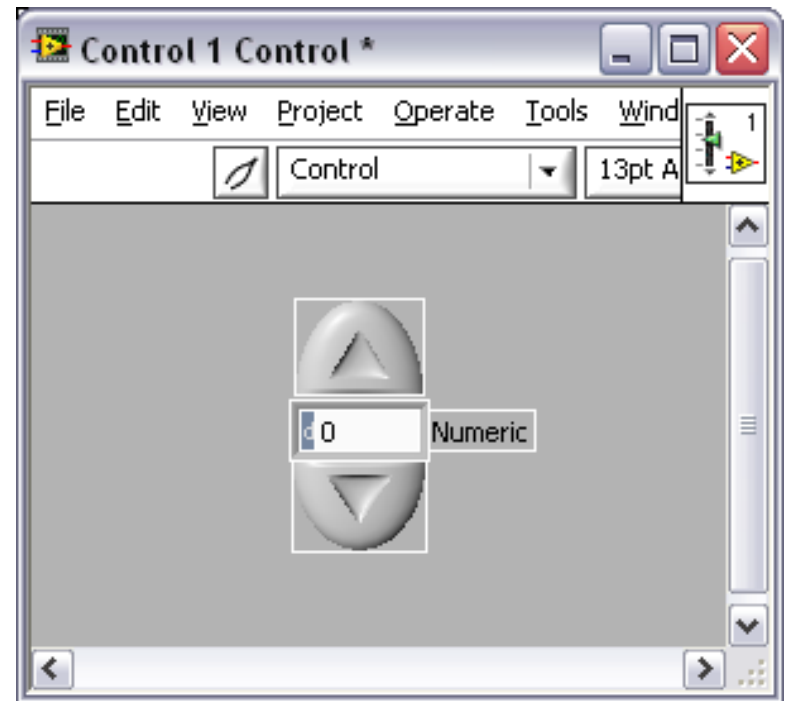
DISCUSSION

C. Type Definitions – Custom Controls

(Определители типа – пользовательские элементы)

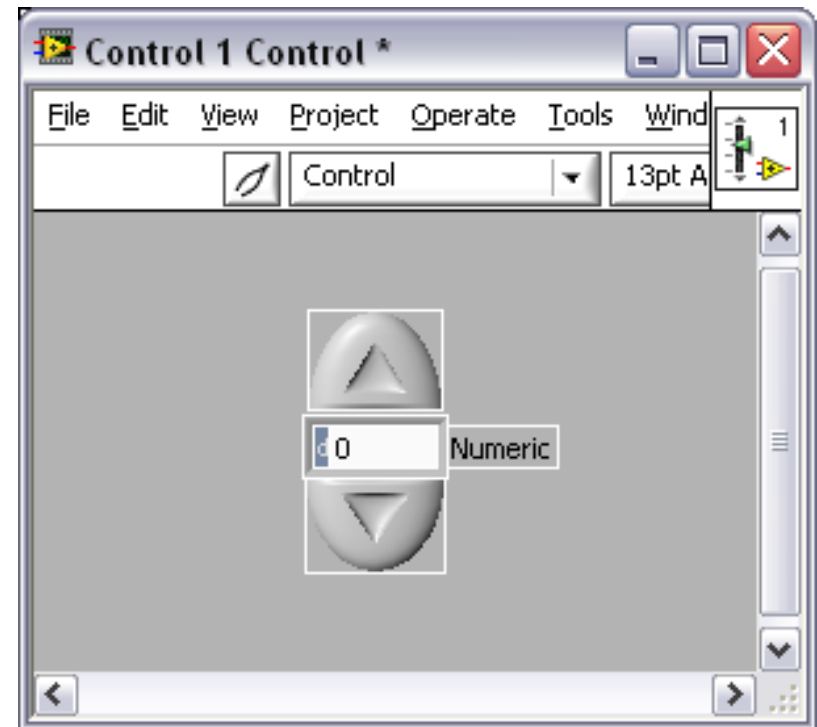
Применяйте пользовательские элементы управления и индикации для расширения доступного набора объектов лицевой панели

1. Создайте и сохраните пользовательский элемент управления или индикации
2. Применяйте пользовательские элементы управления или индикации на других лицевых панелях



C. Type Definitions – Control Editor (Редактор элементов)

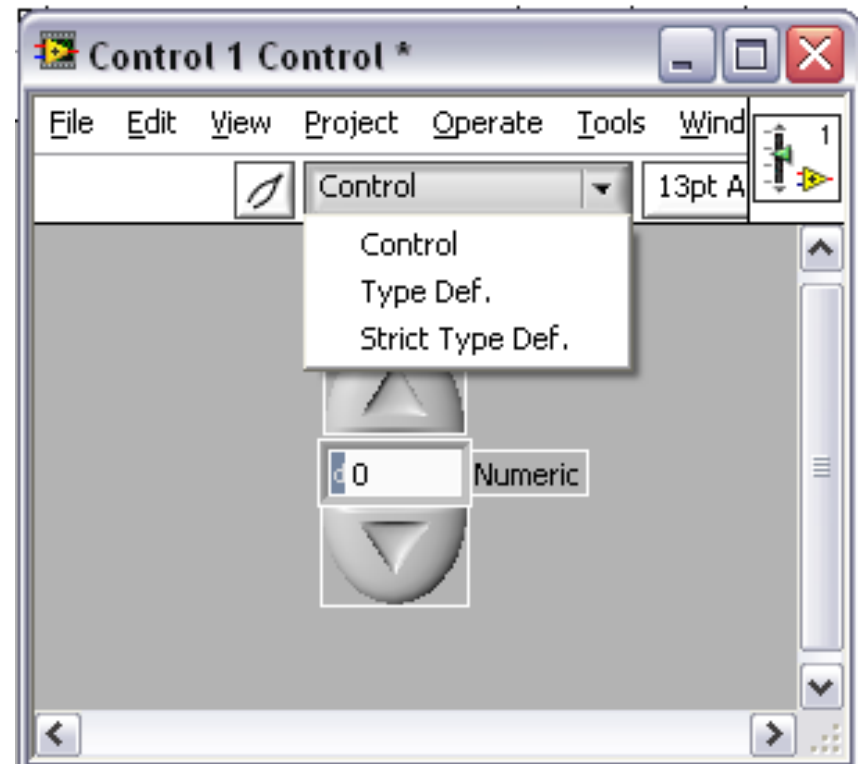
- Используйте редактор Control Editor для адаптации элементов управления или индикации
- Для открытия окна Control Editor, щелкните правой кнопкой мыши по элементу управления или индикации и выберите **Advanced» Customize**



C. Type Definitions

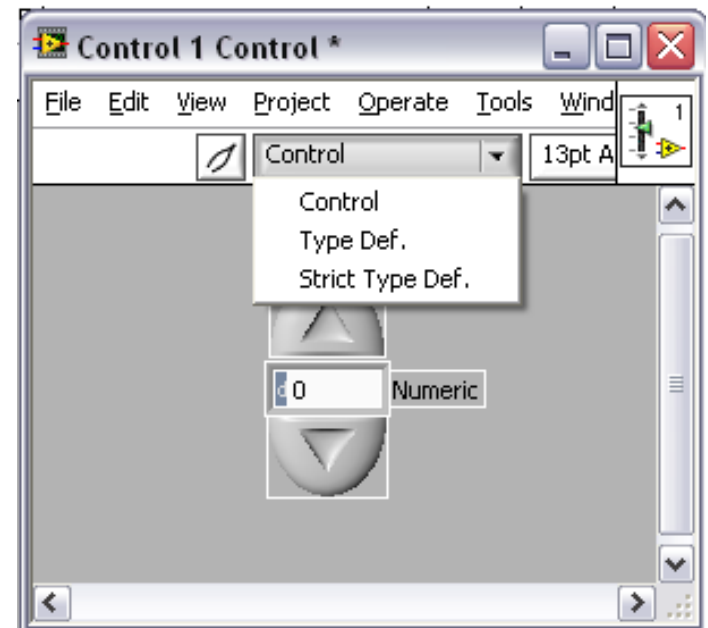
Вы можете сохранить пользовательский элемент управления и индикации как:

- Control (элемент управления и индикации)
- Type Definition (определитель типа)
- Strict Type Definition (строгое определение типа)



C. Type Definitions – Control Type (тип элемента управления и индикации)

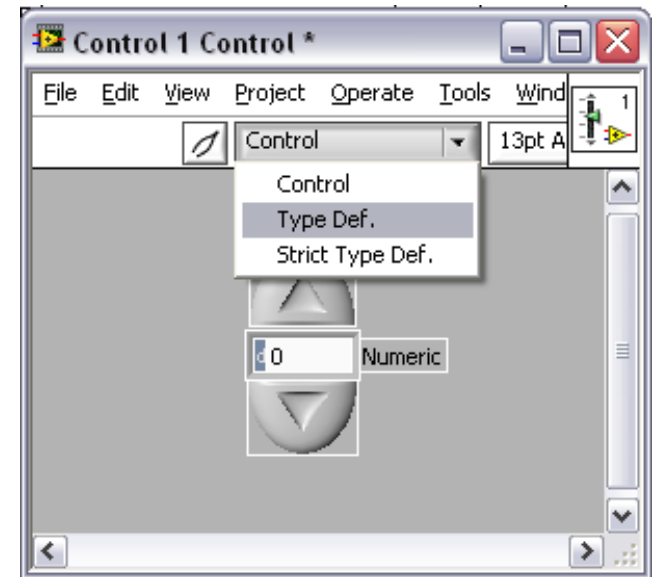
- Control (элемент управления и индикации)
- Нет связи между сохраненным вами пользовательским элементом управления и индикации и экземпляром пользовательского элемента в VI
- Файл обновлен, но экземпляры не обновляются



C. Type Definitions – Type Definition (определители типа)

Type Definition (type def):

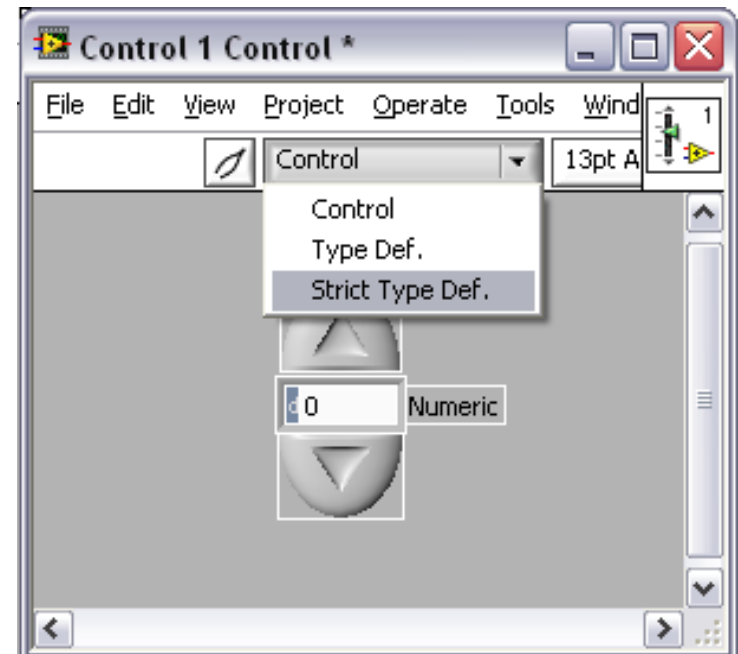
- Изменяет сохраняемый файл, все экземпляры обновляются в соответствии с внесенными изменениями
- Закрепляет тип данных в каждом экземпляре, делая их идентичными
- Примеры:
 - Добавьте элемент в определитель типа enum, экземпляры enum обновятся с добавленным элементом
 - Если вы измените размеры элемента enum control type definition на лицевой панели, размеры экземпляров enum не изменятся



C. Type Definitions – Strict Type Definition (строгое определение типа)

Strict Type Definition:

- Подобен type definition, кроме того, что strict type definition закрепляет в экземпляре все свойства идентично strict type definition, за исключением:
 - label (метки)
 - description (описания)
 - default value (значения по умолчанию)



Упражнение 5-3

Тема: Type Definition

Создайте элемент управления type defined enumerated и исследуйте различия между type definition и strict type definition.

GOAL

ЦЕЛЬ

Упражнение 5-3

Тема: Type Definition

- Если в редакторе Control Editor вы изменили только цвет элемента Strict Type Def Numeric.ctl, должен ли измениться цвет всех экземпляров элемента?
- Если в редакторе Control Editor вы изменили только цвет элемента Type Def Numeric.ctl, должен ли измениться цвет всех экземпляров элемента?

Заключение – Контрольный вопрос

1. Можно ли создать массив массивов?
 - a) True (да)
 - b) False (нет)

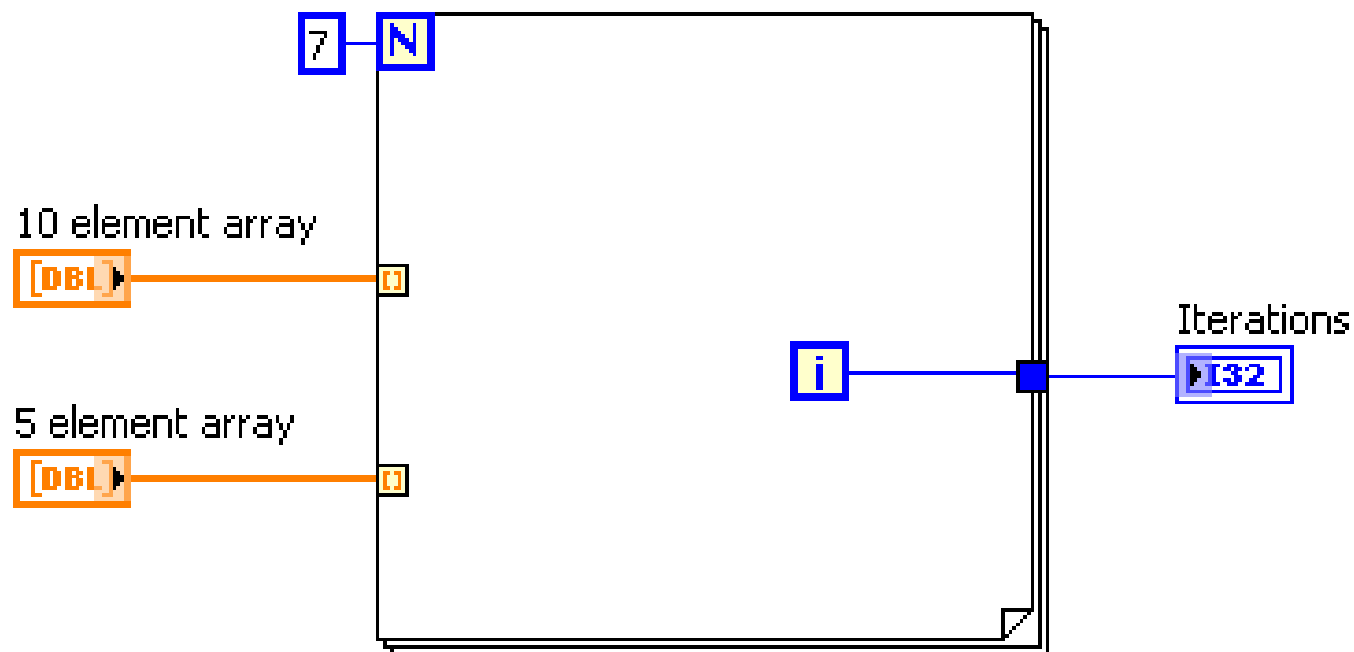
Заключение – Ответ на контрольный вопрос

1. Можно ли создать массив массивов? .
 - a) True (да)
 - b) False (нет)**

Вы не можете перетащить объект типа массив в контейнер массива. Однако вы можете создать двумерный массив.

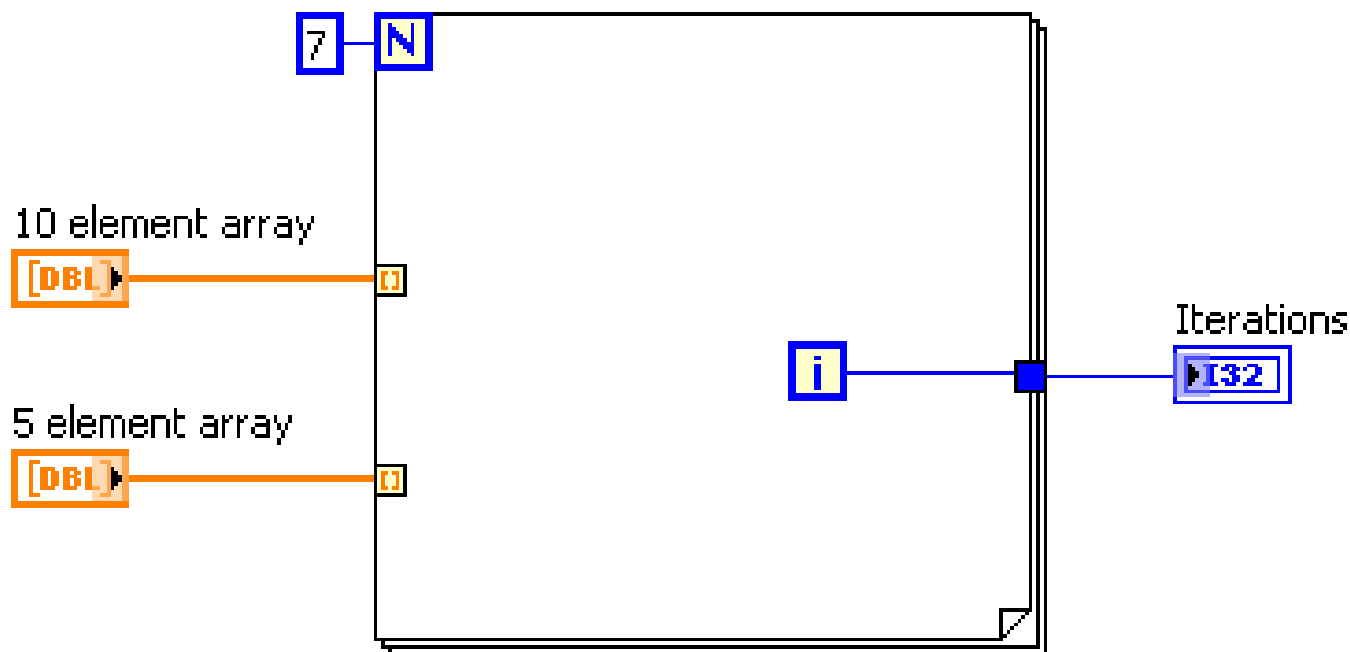
Заключение – Контрольный вопрос

2. Что покажет индикатор итераций после выполнения этого VI?



Заключение – Ответ на контрольный вопрос

2. Что покажет индикатор итераций после выполнения этого VI? **Число итераций = 4**



Заключение – Контрольный вопрос

3. Вы переработали элемент управления, выбрав из выпадающего меню **Type Def. Status** вариант **Control**, и сохранив элемент, как файл `.ctl`. Затем вы используете экземпляры переработанного элемента на лицевой панели. Если вы откроете файл `.ctl` и модифицируете элемент управления, отразятся ли изменения на лицевой панели?
- a) Yes (Да)
 - b) No (Нет)

Заключение – Ответ на контрольный вопрос

3. Вы переработали элемент управления, выбрав из выпадающего меню **Type Def. Status** вариант **Control**, и сохранив элемент, как файл `.ctl`. Затем вы используете экземпляры переработанного элемента на лицевой панели. Если вы откроете файл `.ctl` и модифицируете элемент управления, отразятся ли изменения на лицевой панели?
- a) Yes (Да)
 - b) No (Нет)

Заключение – Контрольный вопрос

4. Вы вводите данные, представляющие окружность. Эти данные состоят из трех чисел двойной точности: координаты x , координаты y и радиуса. В будущем вам может потребоваться расширить все экземпляры данных окружности для дополнения цветом окружности, задаваемого целым числом. Как вы должны изменить представление окружности на лицевой панели?
- a) 3 отдельных элемента для двух координат и радиуса
 - b) Кластер, содержащий все данные
 - c) Пользовательский элемент, содержащий кластер
 - d) Элемент type definition, содержащий кластер
 - e) Массив из 3-х элементов

Заключение – Ответ на контрольный вопрос

4. Вы вводите данные, представляющие окружность. Эти данные состоят из трех чисел двойной точности: координаты x , координаты y и радиуса. В будущем вам может потребоваться расширить все экземпляры данных окружности для дополнения цвета окружности, задаваемого целым числом. Как вы должны изменить представление окружности на лицевой панели?
- a) 3 отдельных элемента для двух координат и радиуса
 - b) Кластер, содержащий все данные
 - c) Пользовательский элемент, содержащий кластер
 - d) Элемент `type definition`, содержащий кластер**
 - e) Массив из 3-х элементов

Лекция 6

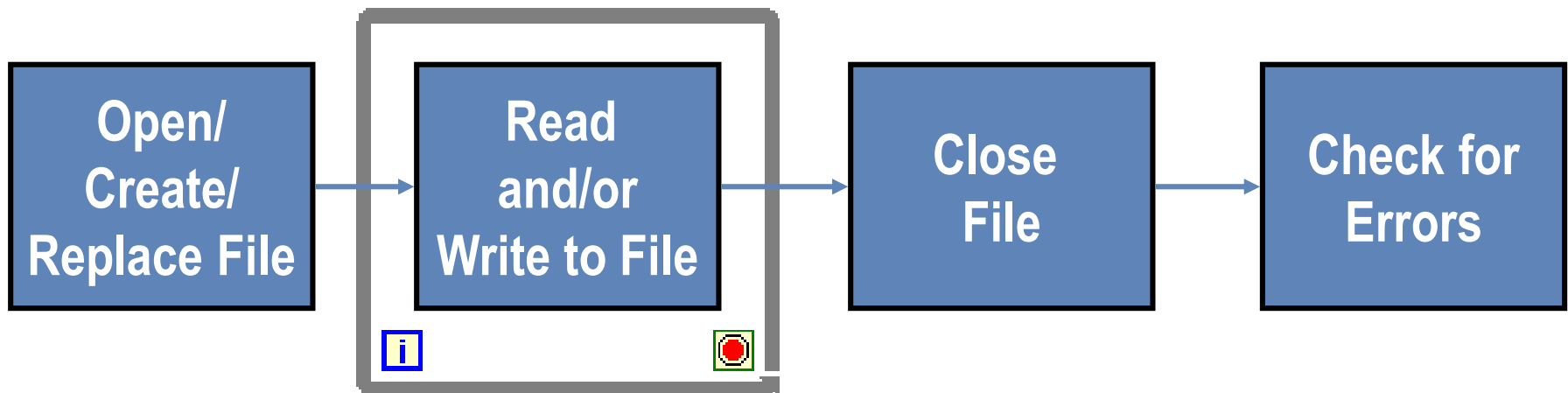
Управление ресурсами

ТЕМЫ

- A. Файловый ввод-вывод
- B. Высокоуровневый
файловый ввод-вывод
- C. Низкоуровневый
файловый ввод-вывод
- D. Программирование
оборудования DAQ
- E. Программное управление
измерительными
приборами
- F. Использование драйверов
измерительных приборов

А. Файловый ввод-вывод

- Функции файлового ввода-вывода пишут в файл или читают из файла
- Типовые операции файлового ввода-вывода включают следующие процессы:



А. Файловый ввод-вывод – форматы файлов

В LabVIEW можно использовать или создавать файлы следующих форматов:

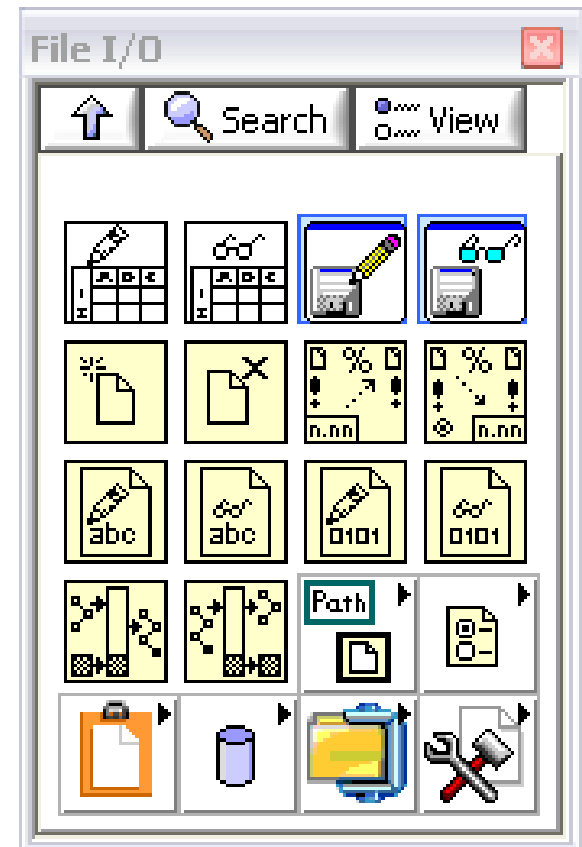
- Binary—двоичный – базовый формат для всех других форматов файлов
- ASCII—специальный тип двоичного файла – стандарт, используемый большинством программ
- LVM— файл данных результатов измерений в LabVIEW (.lvm) – текстовый файл с разделителями табуляцией; подобные файлы можно открывать в приложениях, работающих с текстами или электронными таблицами
- TDMS—разновидность двоичного файла, созданная для продуктов NI; состоит из 2-отдельных файлов: двоичного файла данных и двоичного файла индексов

А. Файловый ввод-вывод – форматы файлов

- В этом курсе вы научитесь создавать текстовые (ASCII) файлы
- Текстовые файлы используют в следующих ситуациях:
 - Нужен доступ к файлу из других приложений
 - Размер дискового пространства и скорость файлового ввода-вывода не критичны
 - Нет необходимости в произвольном обращении при выполнении операции чтения и записи
 - Точность представления чисел не является важной

В. Высокоуровневый файловый ввод-вывод

- VI высокого уровня
 - В распространенных операциях файлового ввода-вывода выполняются все три операции (открытие, чтение/запись, закрытие)
 - Могут быть не так эффективны, как функции, спроектированные или сконфигурированные для отдельных операций
- VI низкого уровня
 - Индивидуальные VI для каждой операции файлового ввода-вывода
 - Если запись в файл производится в цикле, используйте функции низкоуровневого файлового ввода-вывода



В. Высокоуровневый файловый ввод-вывод

Запись в файл электронных таблиц (Write to Spreadsheet File)

- Массивы чисел двойной точности преобразуются в строки текста, которые записываются в файл ASCII

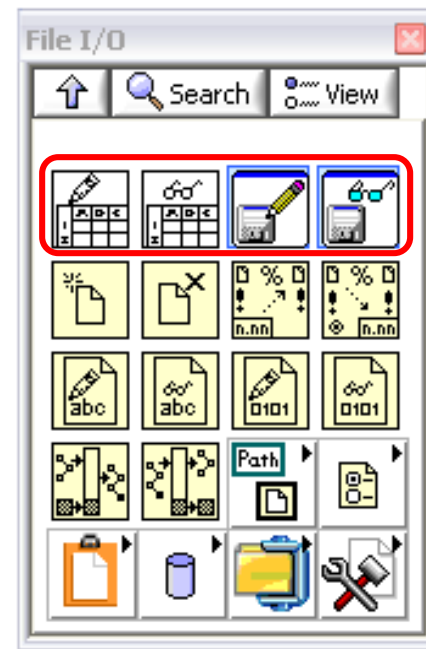
Чтение из файла электронных таблиц (Read From Spreadsheet File)

- Читается заданное количество линий или строк из текстового файла, представляющих числа, и выводятся в 2-мерный массив чисел двойной точности

Запись/чтение в/из файла результатов измерений

(Write to/Read from Measurement File)

- Express VI для записи/чтения данных в/из файлов форматов LVM или TDMS



Упражнение 6-1

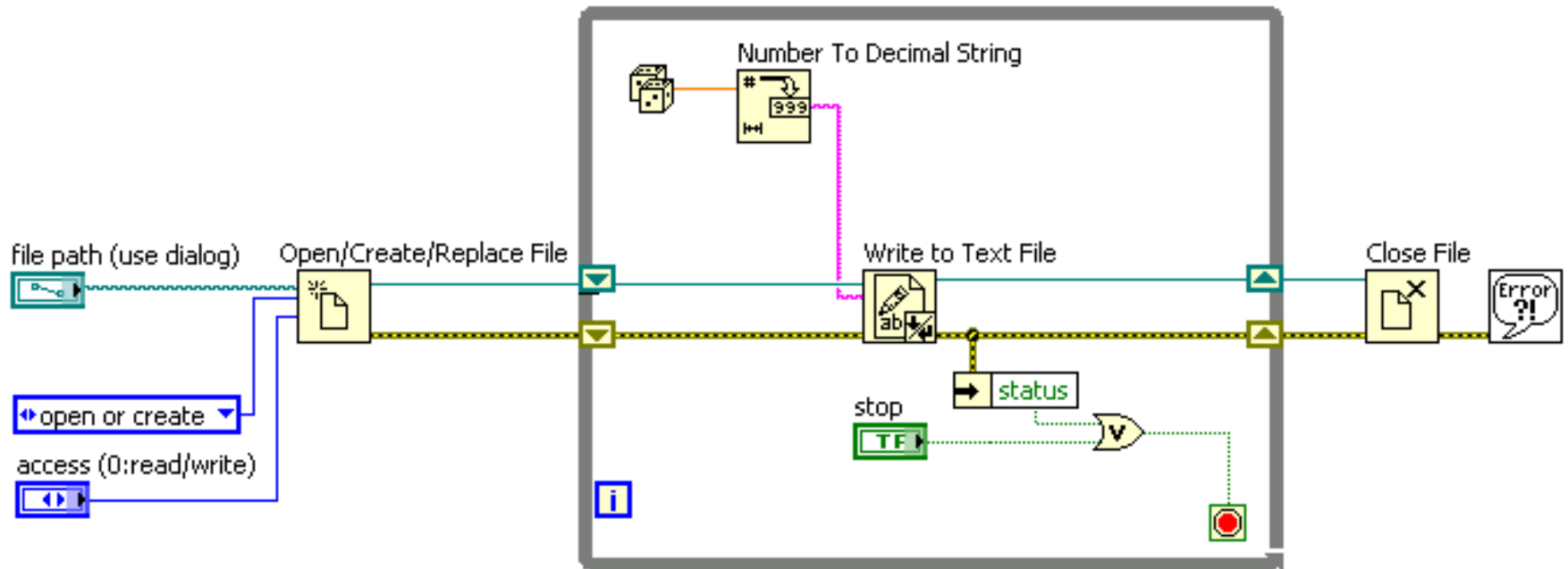
Spreadsheet Example VI

Используйте VI файлового ввода-вывода высокого уровня для записи в файл, который можно читать из приложений, работающих с файлами электронных таблиц.

GOAL

ЦЕЛЬ

С. VI низкоуровневого файлового ввода-вывода



Упражнение 6-2

Temperature Log VI

Модифицируйте VI для потоковой записи в ASCII файл.

GOAL

ЦЕЛЬ

Упражнение 6-2

Temperature Log VI

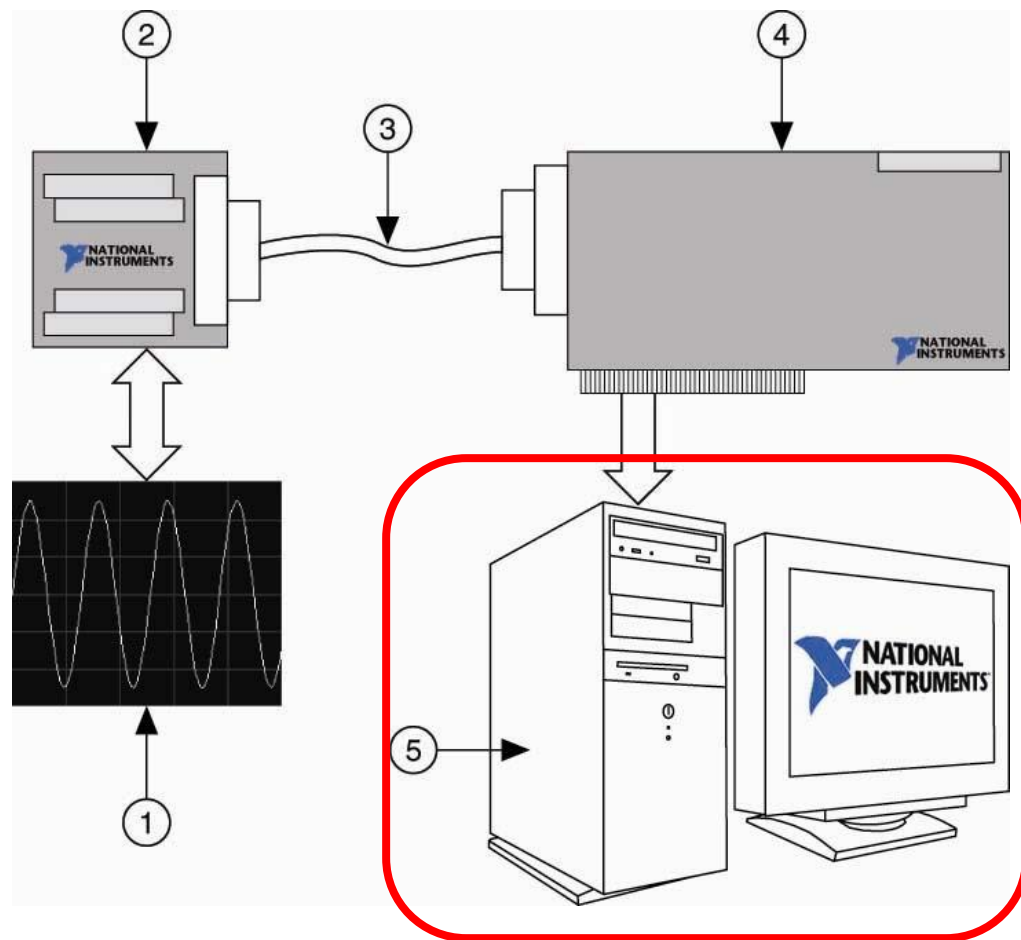
- Что произойдет, если вы используете Express VI Write to Measurement File внутри цикла While?

ДИСКУССИЯ

DISCUSSION

D. Программирование оборудования DAQ

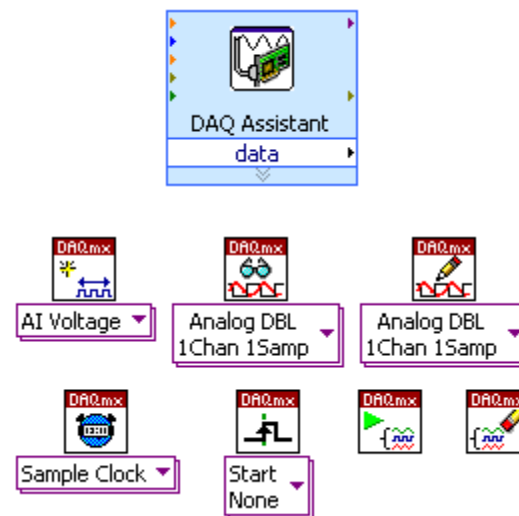
1. Signal (сигнал)
2. Terminal Block
(коннекторный блок)
3. Cable (кабель)
4. DAQ Device
(устройство сбора данных)
5. Computer (компьютер)



D. Программирование оборудования DAQ

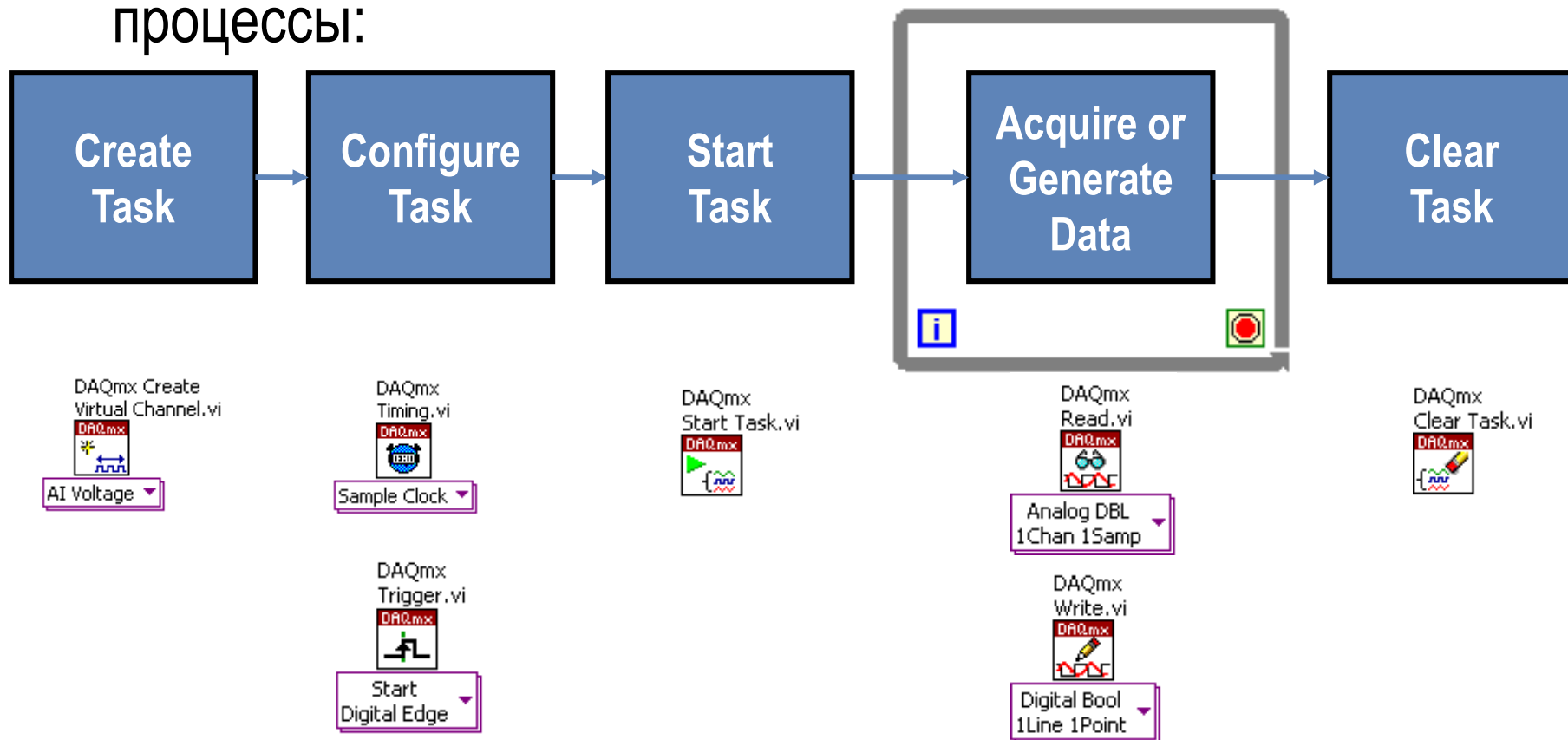
– Обзор программного обеспечения

- NI-DAQmx
 - Программные драйверы
 - Распознавание DAQ устройств
 - Инсталляция NI-DAQmx функций в LabVIEW
- Measurement & Automation Explorer
 - Конфигурирование и тестирование DAQ устройств
- DAQ Assistant
 - Конфигурируемые Express VI для разработки приложений DAQ
- DAQmx API
 - Предоставляют набор VI для программирования приложений DAQ)



D. Программирование оборудования DAQ – Основной алгоритм

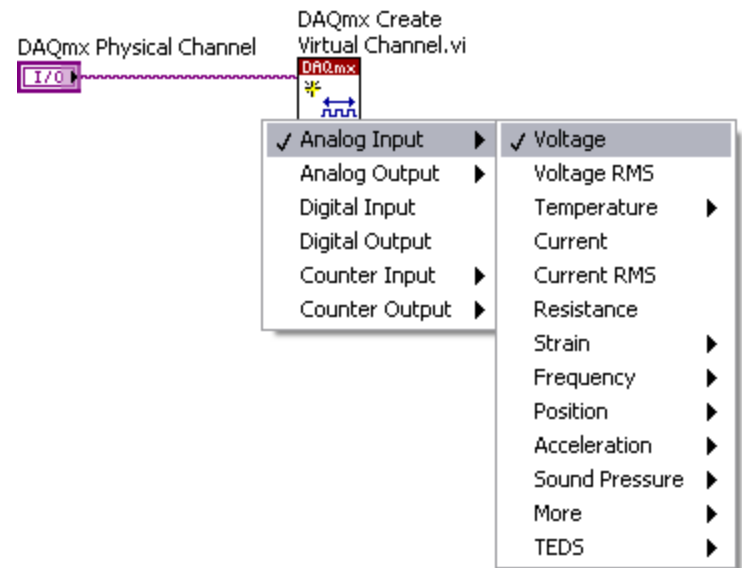
- Типовые приложения DAQmx включают следующие процессы:



D. Программирование оборудования DAQ

– Create Task (создание задачи)

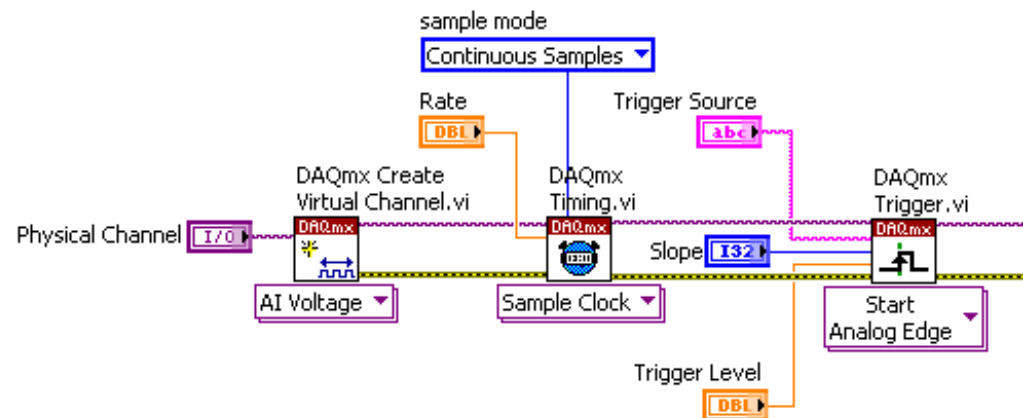
- Создайте Virtual Channel VI
 - Создайте виртуальный канал и добавьте его в задачу
 - Используйте выпадающее меню, чтобы выбрать подходящий вариант для этого VI



D. Программирование оборудования DAQ

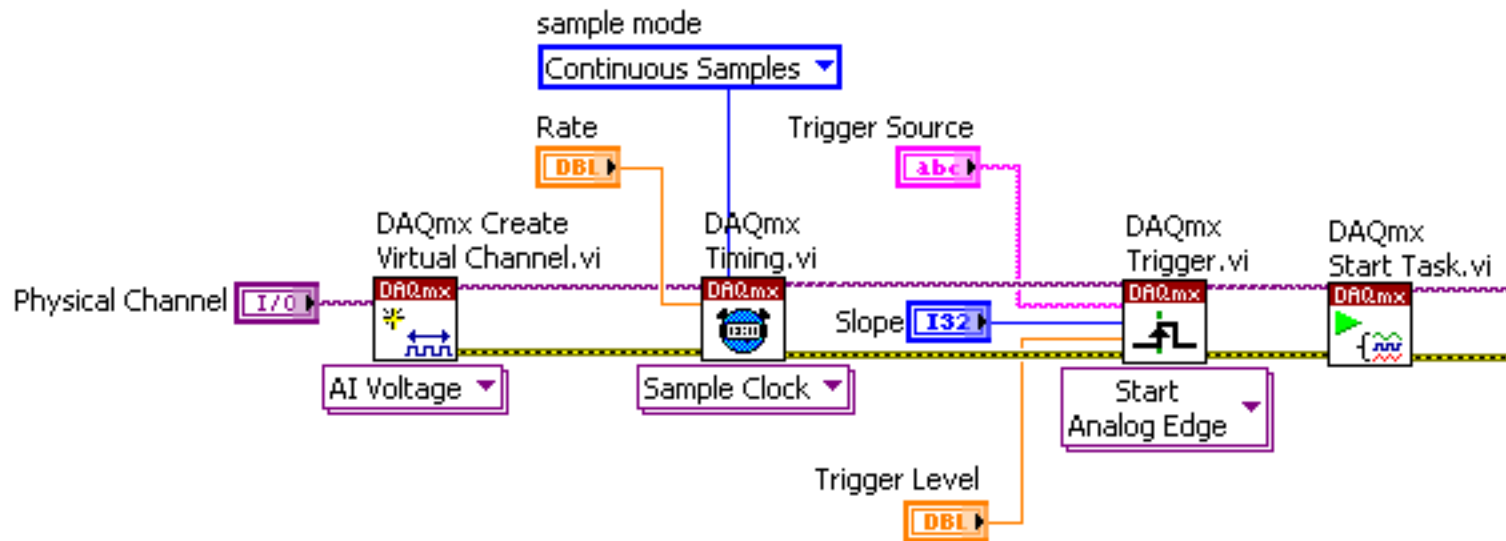
– Configure Task (Конфигурирование задачи)

- Сконфигурируйте временную диаграмму считывания последовательности отсчетов
 - Частоту выборки, источник синхронизации и т.п.
- Сконфигурируйте режим запуска, если это необходимо в разрабатываемом приложении
 - Сконфигурируйте параметры запуска или останова по нарастающему или ниспадающему фронту цифрового или аналогового сигнала, или в зоне аналогового сигнала



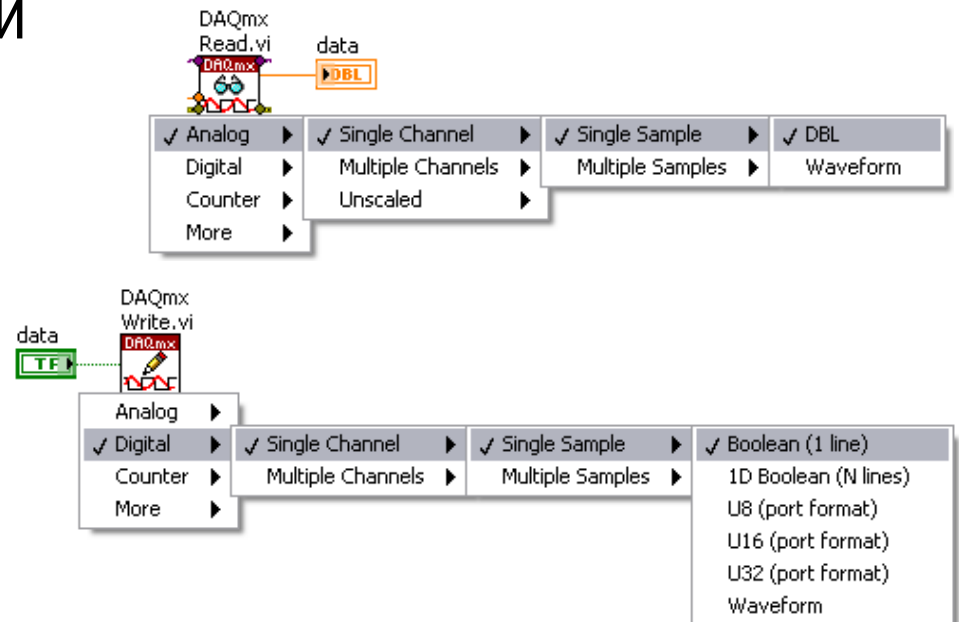
D. Программирование оборудования DAQ – Start Task (Запуск задачи)

- Запускайте задачу после того, как сконфигурируете задачу



D. Программирование оборудования DAQ – Acquire or Generate Data (Сбор или генерация данных)

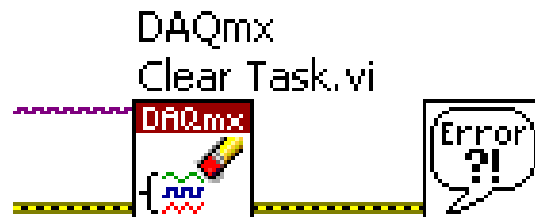
- Собирайте или генерируйте данные с помощью DAQ устройства
- Согласуйте совместимость выбора в выпадающем меню с конфигурацией задачи



D. Программирование оборудования DAQ

– Clear Task (Очистка задачи)

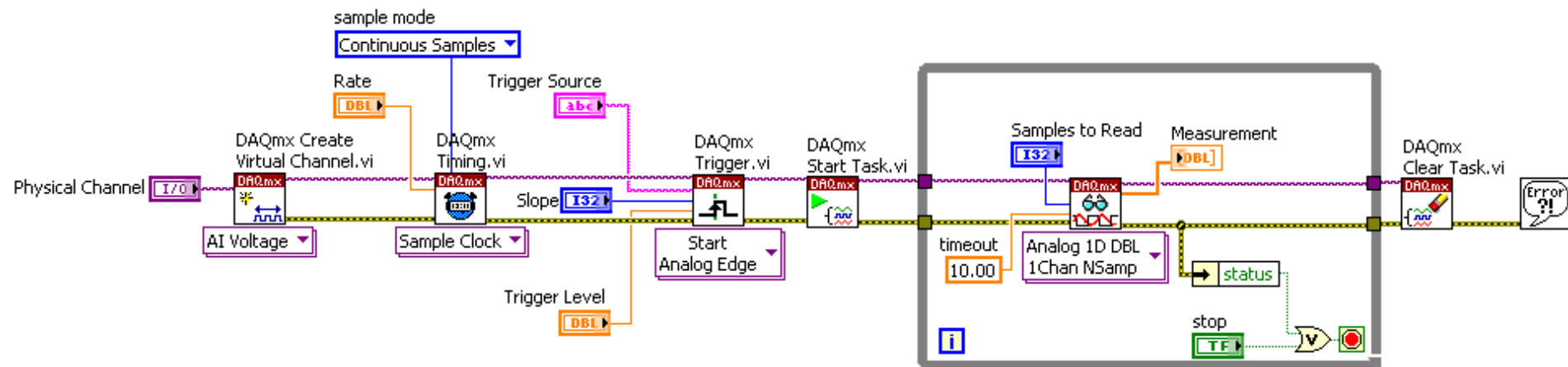
- DAQmx Clear Task VI
 - Останавливает выполнение задачи
 - Освобождает все ресурсы, зарезервированные в задаче
 - Очищает задачу



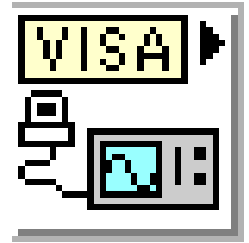
D. Программирование оборудования DAQ

– Примеры

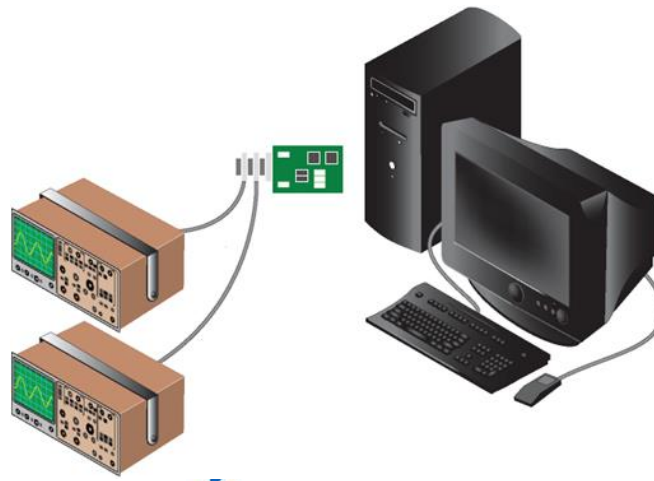
- Пример сбора данных с запуском



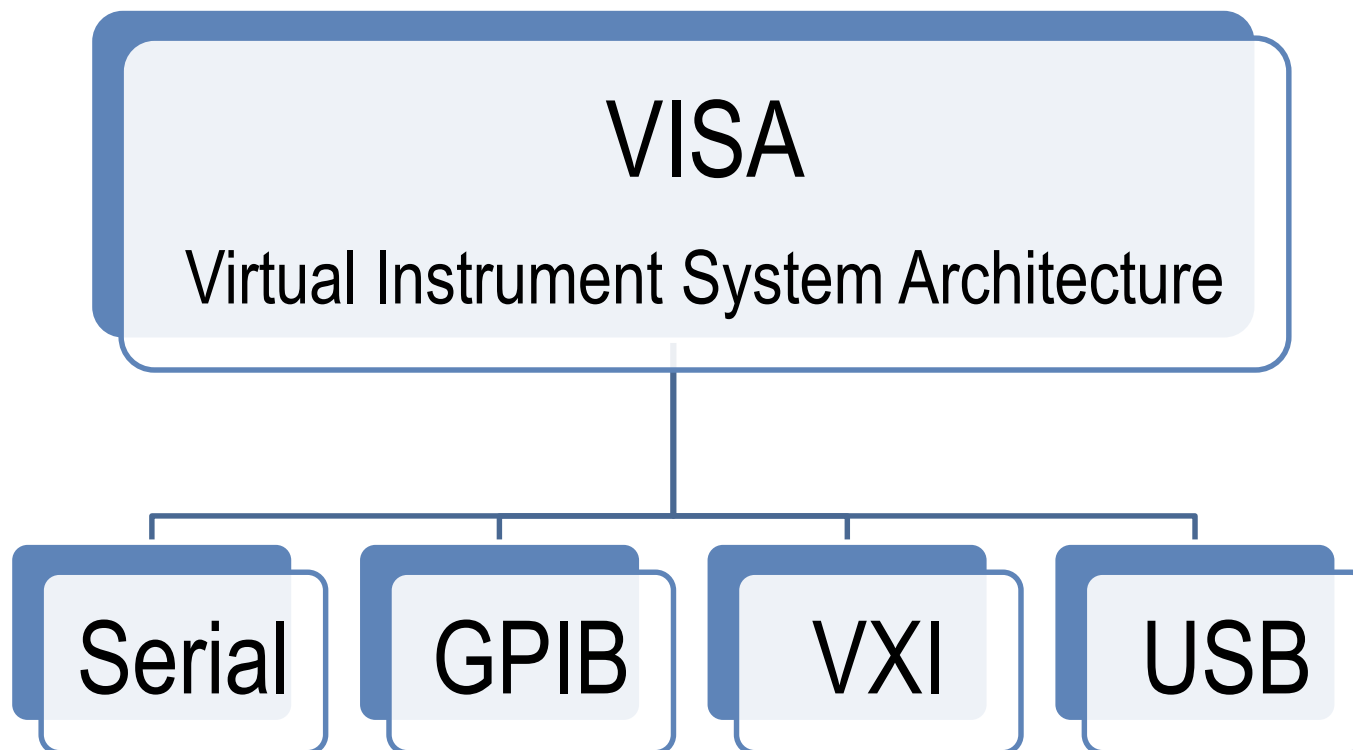
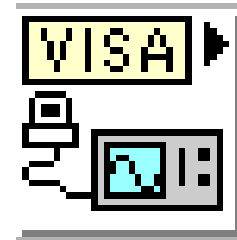
Е. Программное управление измерительными приборами



- Virtual Instrument Software Architecture (VISA) – архитектура ПО виртуальных измерительных приборов:
 - Высокоуровневые функции API для вызова низкоуровневых драйверов
 - Могут управлять оборудованием VXI, GPIB, с последовательным портом; в зависимости от типа используемого прибора вызывают соответствующие драйверы



VISA



VISA – Терминология программирования

- Resource (Ресурсы)

Любой прибор в системе, включая приборы с последовательным и параллельным портом

- Session (Сессия)

Когда вы открываете сессию работы с прибором, LabVIEW возвращает номер VISA сессии, который является уникальной ссылкой на этот прибор

- Instrument Descriptor (Дескриптор прибора)

Определяет тип интерфейса (GPIB, VXI, ASRL), адрес устройства и тип сессии VISA (INSTR или Event)

VISA Alias (Псевдонимы VISA)

Присвойте устройству или ресурсу определенное пользователем имя вместо дескриптора прибора



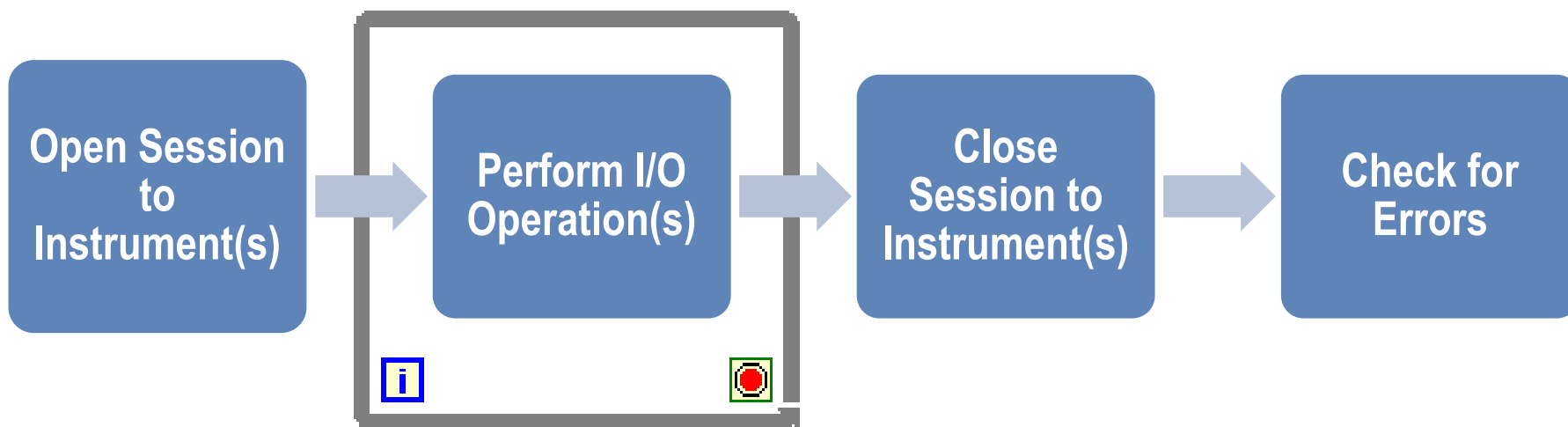
 GPIB0::1::INSTR

Device Type: GPIB Instrument

VISA Alias on My System:

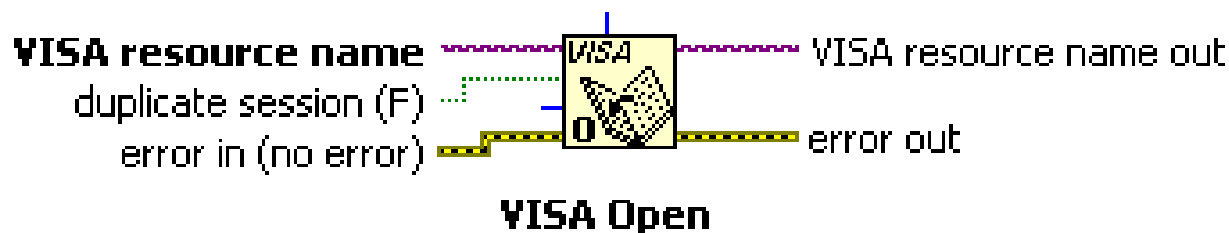
Программирование VISA

Функции VISA действуют аналогично функциям файлового ввода-вывода



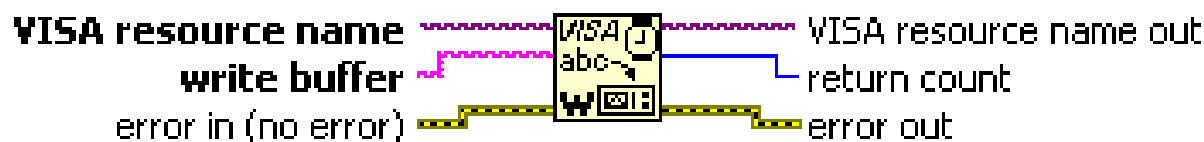
Функция VISA Open

- Предоставляет линию связи с ресурсом
- Обычно используется однократно для каждого ресурса
- Возвращает имя ресурса VISA

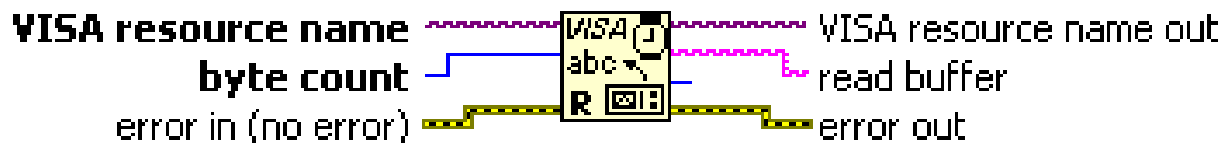


Функции ввода-вывода VISA

VISA Write и VISA Read Functions



VISA Write



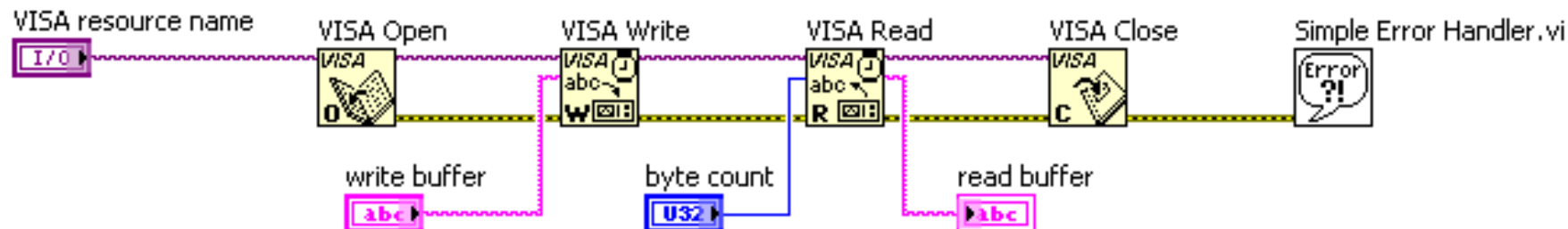
VISA Read

Функция VISA Close

- Сессии захватывают ресурсы систем
- Закрывайте сессии прежде, чем завершите программу

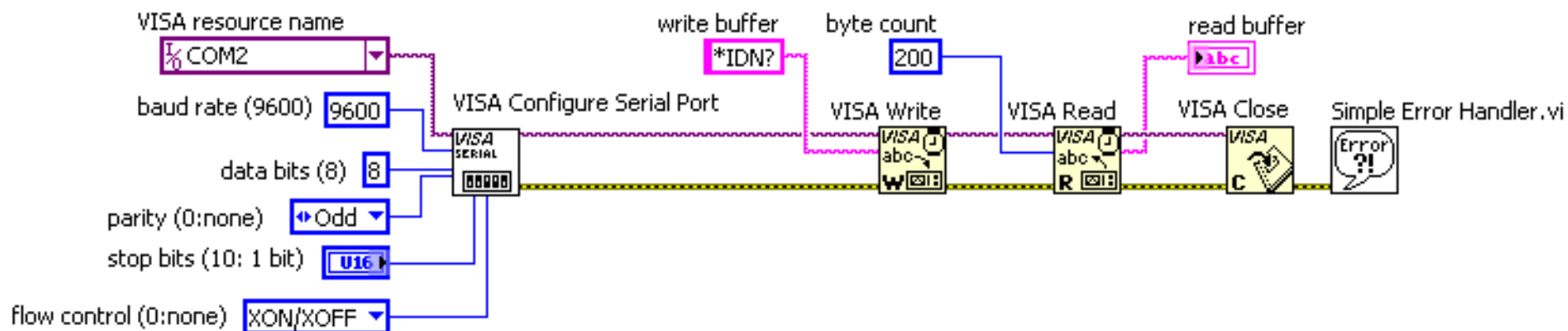


Примеры VISA Write и Read



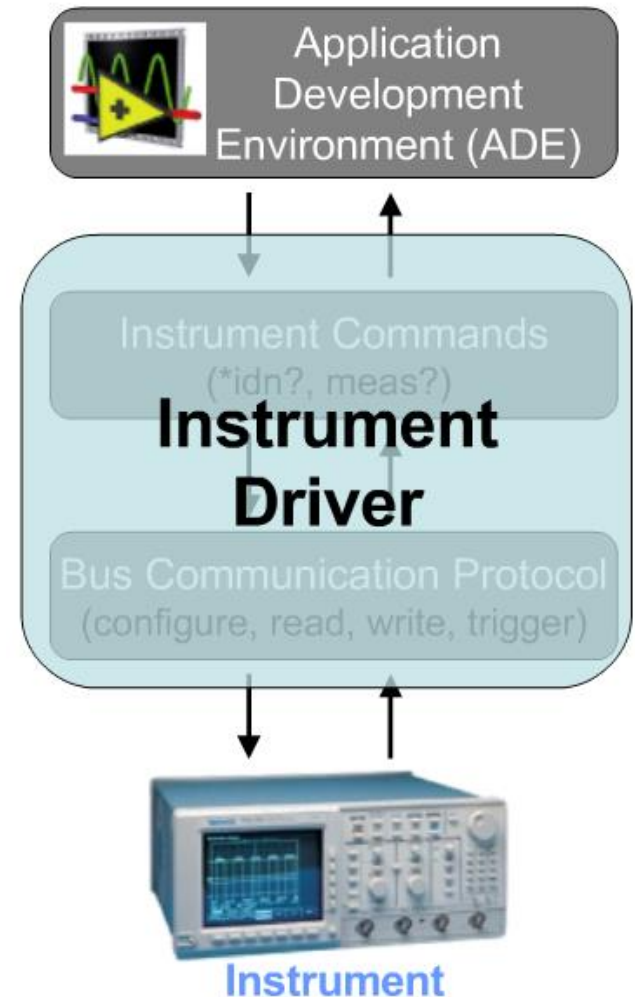
VISA – Последовательный порт

VISA Configure Serial Port VI инициализирует порт, определенный именем ресурса VISA, в соответствии с заданными настройками



Г. Использование драйверов измерительных приборов

- Организованный набор VI для управления программируемым измерительным прибором
 - Каждый VI выполняет несколько инструкций
 - Сгруппирован по типу операций (конфигурирование, данные и т.д.)
- Сокращает время проектирования
 - Упрощает управление прибором
 - Возможно повторное использование
 - Унифицированные архитектура и интерфейс



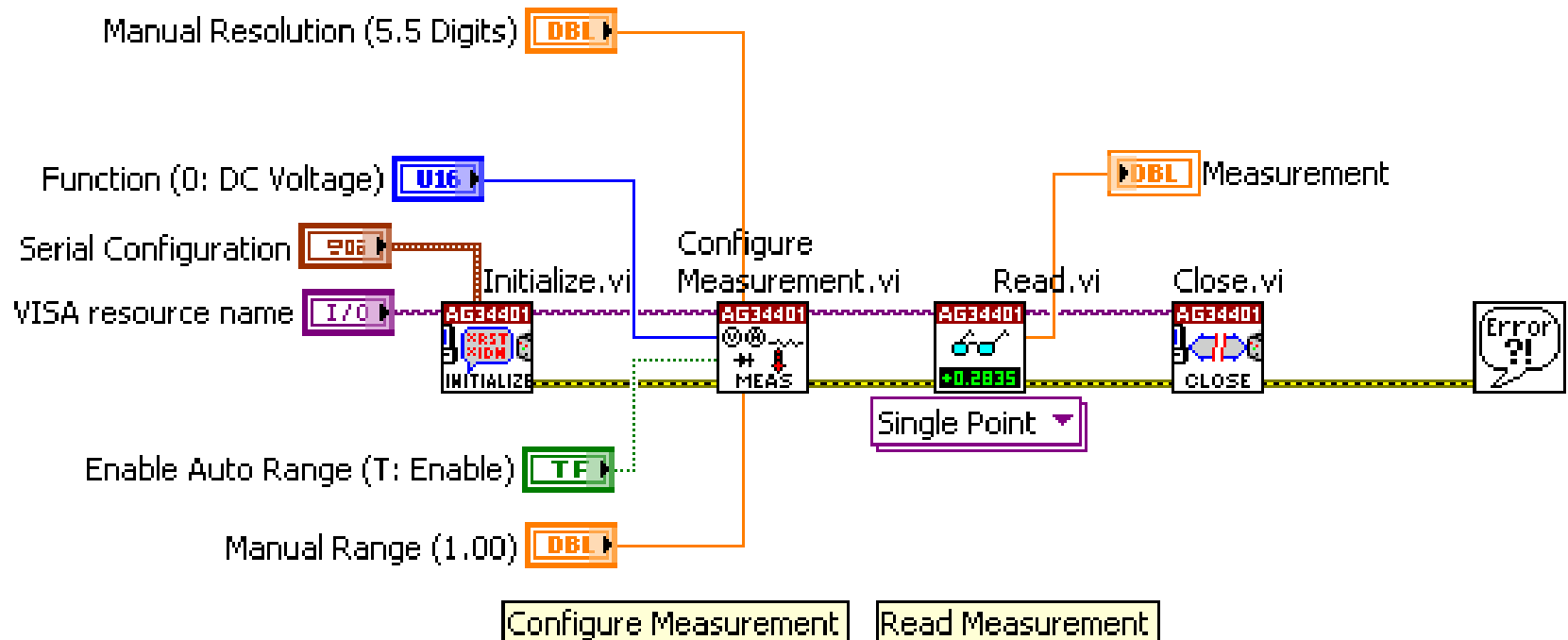
Использование драйверов измерительных приборов

- Используемый вами драйвер измерительного прибора содержит программный код, специфичный для данного прибора
- Если вы заменяете прибор, замените и VI драйвера прибора на VI драйвера нового прибора, это существенно сократит время переработки проекта

Использование драйверов измерительных приборов – где их найти?

- Вы можете найти большинство драйверов приборов LabVIEW Plug and Play в поисковике Instrument Driver Finder
 - Доступ к поисковику в LabVIEW осуществляется выбором **Tools» Instrumentation»Find Instrument Drivers** или **Help»Find Instrument Drivers**
 - Подключает вас к `ni.com` для поиска драйверов
- Когда вы устанавливаете драйвер прибора
 - Пример программы, использующей драйвер, добавляется в поисковик примеров NI Example Finder
 - VI драйверов приборов добавляются в субпалитру **Instrument I/O»Instrument Drivers** палитры Functions

Использование драйверов измерительных приборов – Примеры



Упражнение 6-3

Использование DAQmx (DAQ)

или

Упражнение 6-4

NI Devsim VI(GPIB/serial)

ДОМАШНЕЕ ЗАДАНИЕ

6-3: Изучите пример программы DAQmx для непрерывного сбора данных и модифицируйте ее для ожидания цифрового запуска

6-4: Инсталлируйте драйвер прибора и изучите примеры программ, поставляемых с драйвером этого прибора

GOAL

ЦЕЛЬ

Упражнение 6-3

Использование DAQmx (DAQ)

или

Упражнение 6-4

NI Devsim VI(GPIB/serial)

- В Упражнении 6-3 VI каких типов выполняются вне цикла While?
- В Упражнении 6-4, как нужно модифицировать примеры программ, если вы хотите непрерывно собирать данные?

Заключение – Контрольный вопрос

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

Заключение – Контрольный вопрос

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

Заключение – Контрольный вопрос

2. Если вы хотите видеть данные в текстовом редакторе, подобном Notepad, какой формат файла нужно использовать при сохранении данных?
- a) ASCII
 - b) TDMS

Заключение – Контрольный вопрос

2. Если вы хотите видеть данные в текстовом редакторе, подобном Notepad, какой формат файла нужно использовать при сохранении данных?
- a) **ASCII**
 - b) TDMS

Заключение – Контрольный вопрос

3. Какая из следующих цепочек соответствует основному алгоритму программирования DAQmx?
- a) Create Task»Configure Task»Acquire/Generate Data»Start Task
 - b) Acquire/Generate Data»Start Task»Clear Task
 - c) Start Task»Create Task»Configure Task»Acquire/Generate Data»Clear Task
 - d) Create Task»Configure Task»Start Task»Acquire/Generate Data»Clear Task

Заключение – Контрольный вопрос

3. Какая из следующих цепочек соответствует основному алгоритму программирования DAQmx?
- a) Create Task»Configure Task»Acquire/Generate Data»Start Task
 - b) Acquire/Generate Data»Start Task»Clear Task
 - c) Start Task»Create Task»Configure Task»Acquire/Generate Data»Clear Task
 - d) **Create Task»Configure Task»Start Task»Acquire/Generate Data»Clear Task**

Заключение – Контрольный вопрос

4. VISA – это высокоуровневые API, вызывающие драйверы низкого уровня.
- a) True (Да)
 - b) False (Нет)

Заключение – Контрольный вопрос

4. VISA – это высокоуровневые API, вызывающие драйверы низкого уровня.
- a) **True (Да)**
 - b) **False (Нет)**

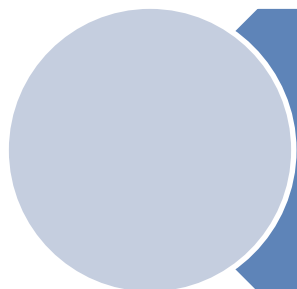
Лекция 7

Разработка модульных приложений

ТЕМЫ

- A. Понятие модульности
- B. Иконка и панель подключения
- C. Использование SubVI

А. Понятие модульности



Модульность – уровень компоновки программы из отдельных модулей, при котором изменения в одном модуле минимально влияют на другие модули

Модули LabVIEW называются subVI

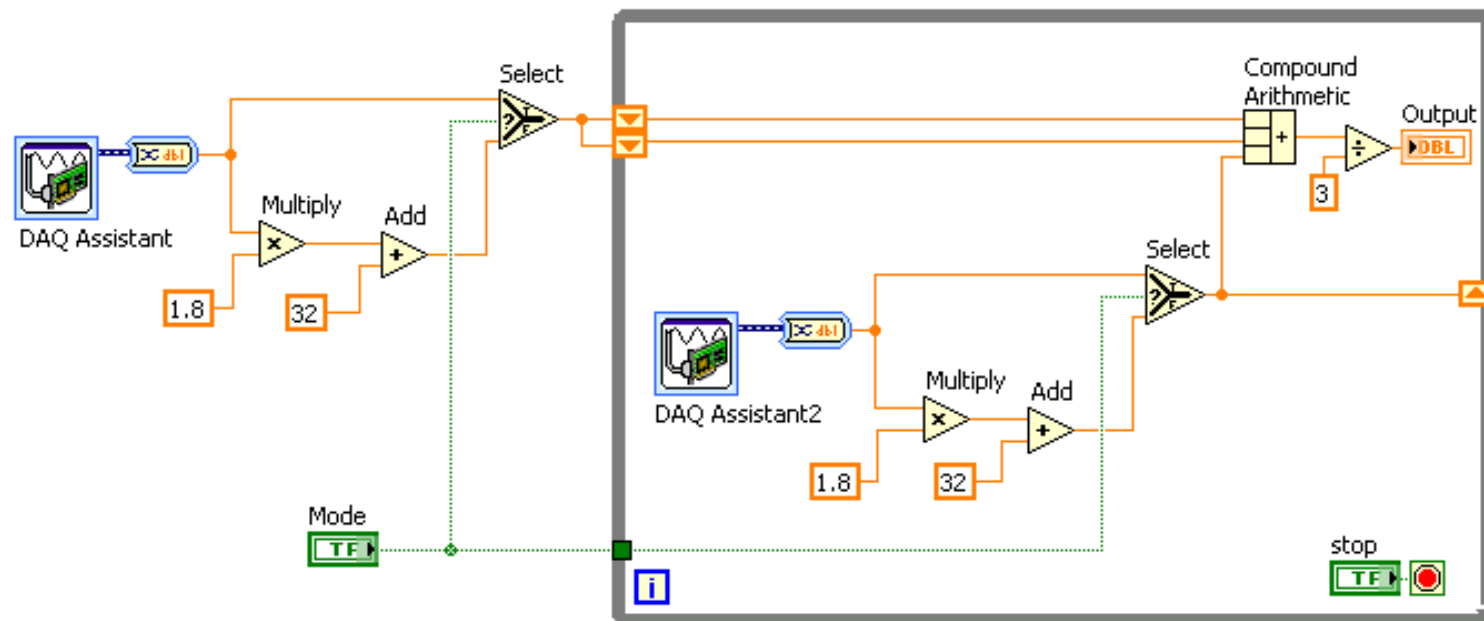
А. Понятие модульности – SubVI



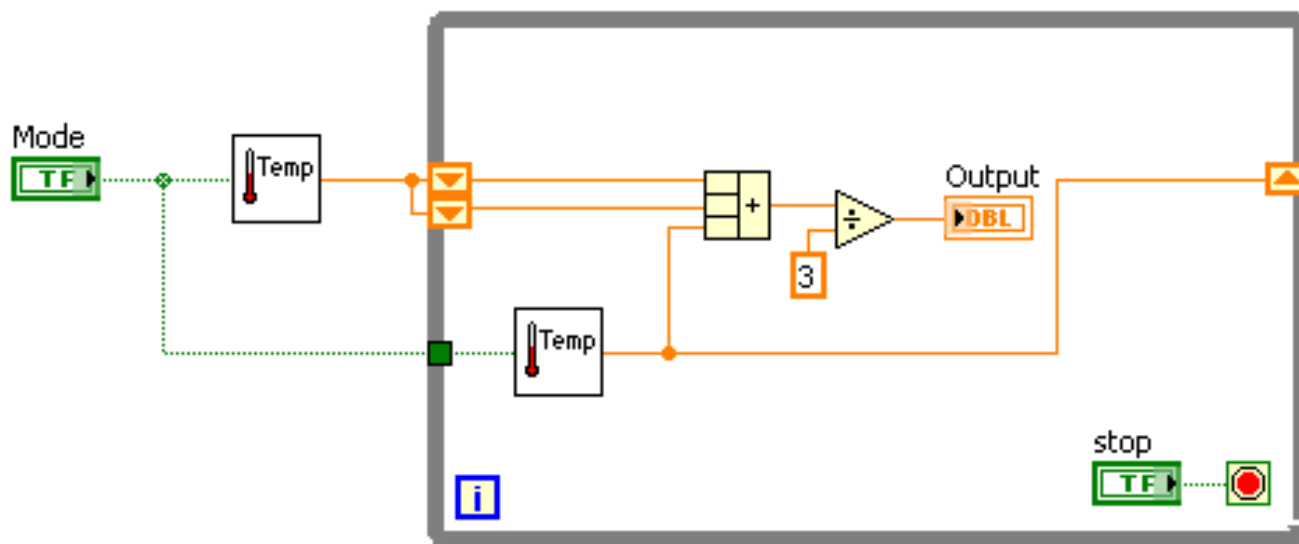
SubVI- это VI внутри других VI

- SubVI соответствуют подпрограммам в текстовых языках программирования
- В верхнем правом углу лицевой панели и блок-диаграммы находится иконка VI
- Эта иконка идентифицирует VI, когда VI помещается на блок-диаграмму

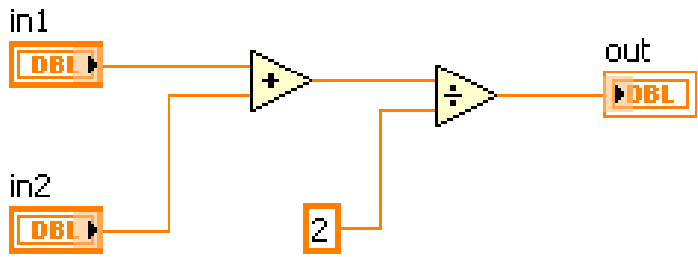
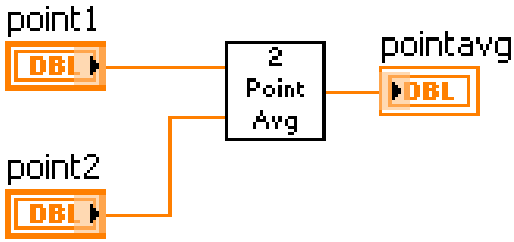
A. Понятие модульности – SubVI



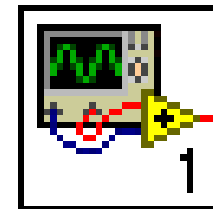
А. Понятие модульности – SubVI



А. Понятие модульности – SubVI

Код функции	Код вызывающей программы
<pre>function average (in1, in2, out) { out = (in1 + in2)/2.0; }</pre>	<pre>main { average (point1, point2, pointavg) }</pre>
Блок-диаграмма SubVI	Блок-диаграмма вызывающего VI
	

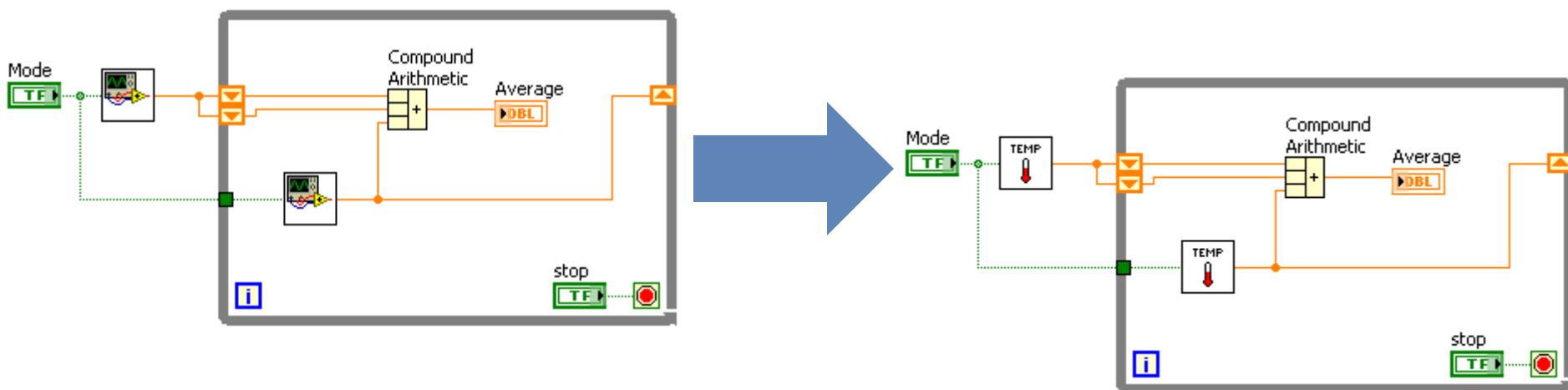
В. Иконка и панель подключения



- После разработки VI создайте иконку и панель подключения, чтобы VI можно было использовать в качестве subVI
- Иконка и панель подключения соответствуют прототипу функции в текстовых языках программирования
- В верхнем правом углу окон лицевой панели и блок-диаграммы каждого VI находится иконка VI
- Иконка – это графическое представление VI
- Если вы используете VI в качестве subVI, иконка идентифицирует subVI на блок-диаграмме VI

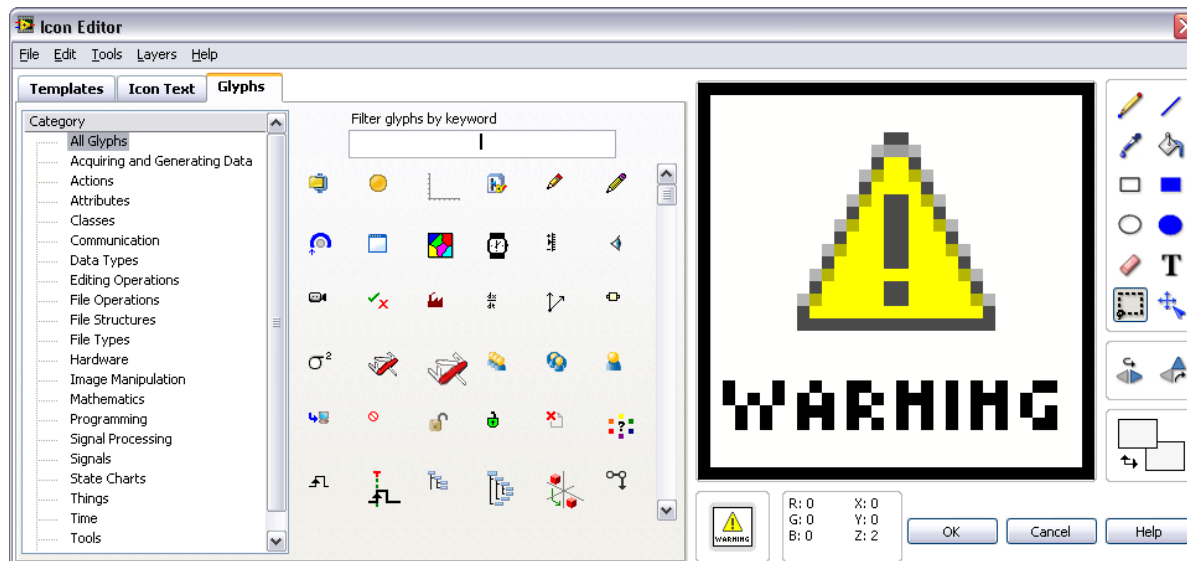
В. Иконка и панель подключения – Хорошая иконка VI

- Хорошая иконка VI характеризуется
 - Отражением функциональности VI путем использования:
 - Соответствующего графического изображения
 - Содержательного текста



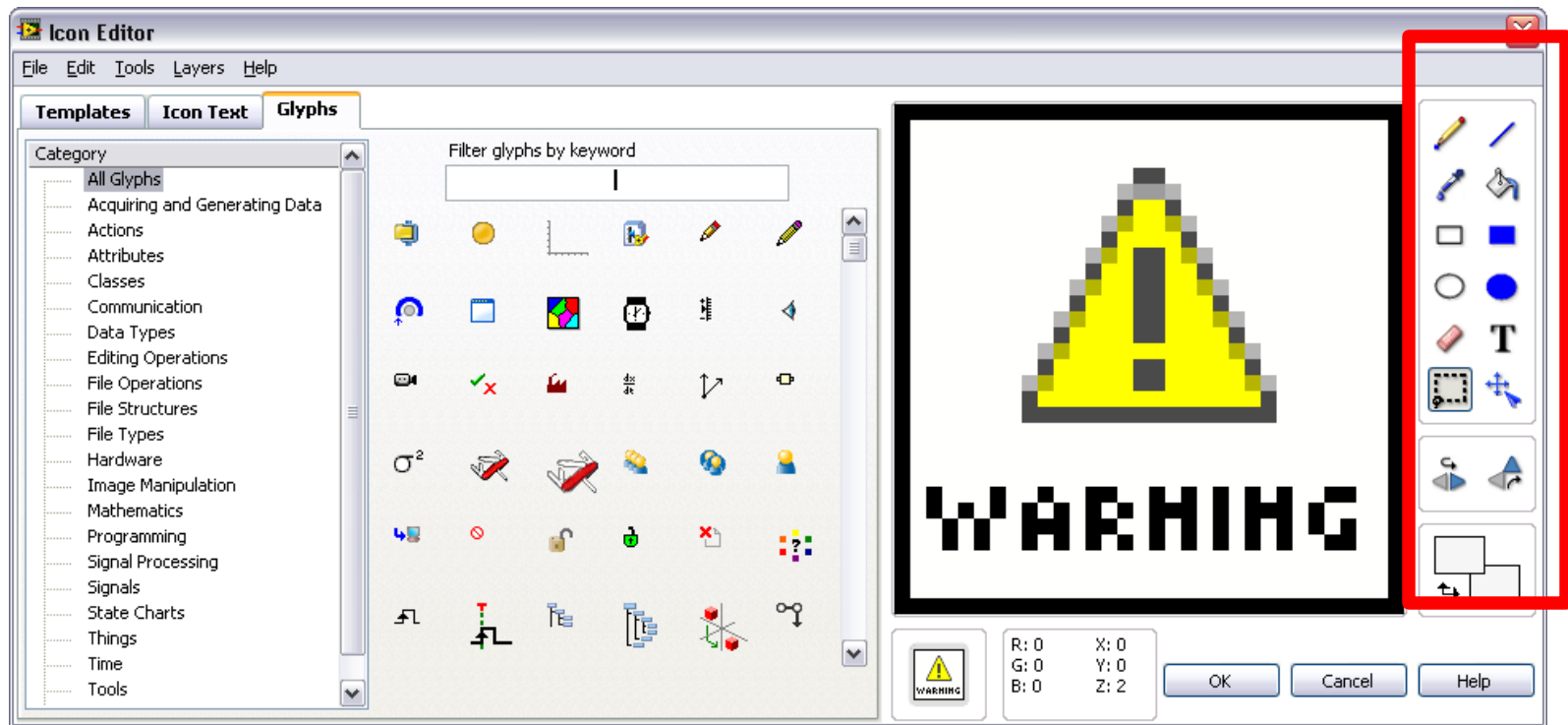
В. Иконка и панель подключения – Создание иконки

- Пользовательская иконка создается щелчком правой кнопки мыши по иконке в верхнем правом углу лицевой панели или блок-диаграммы и выбором **Edit Icon** или двойным щелчком по иконке
- Вы можете также перетащить рисунок откуда-нибудь из вашей файловой системы и оставить его в иконке



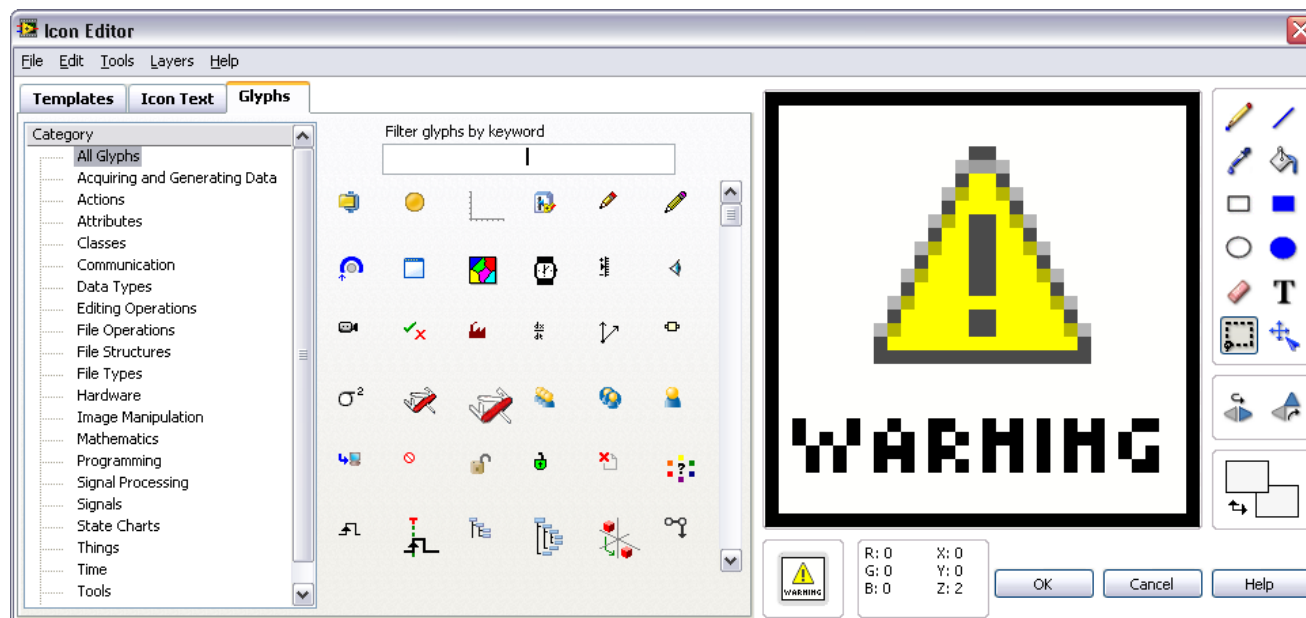
В. Иконка и панель подключения – Создание иконки

- Используйте средства редактирования для изменения иконки вручную



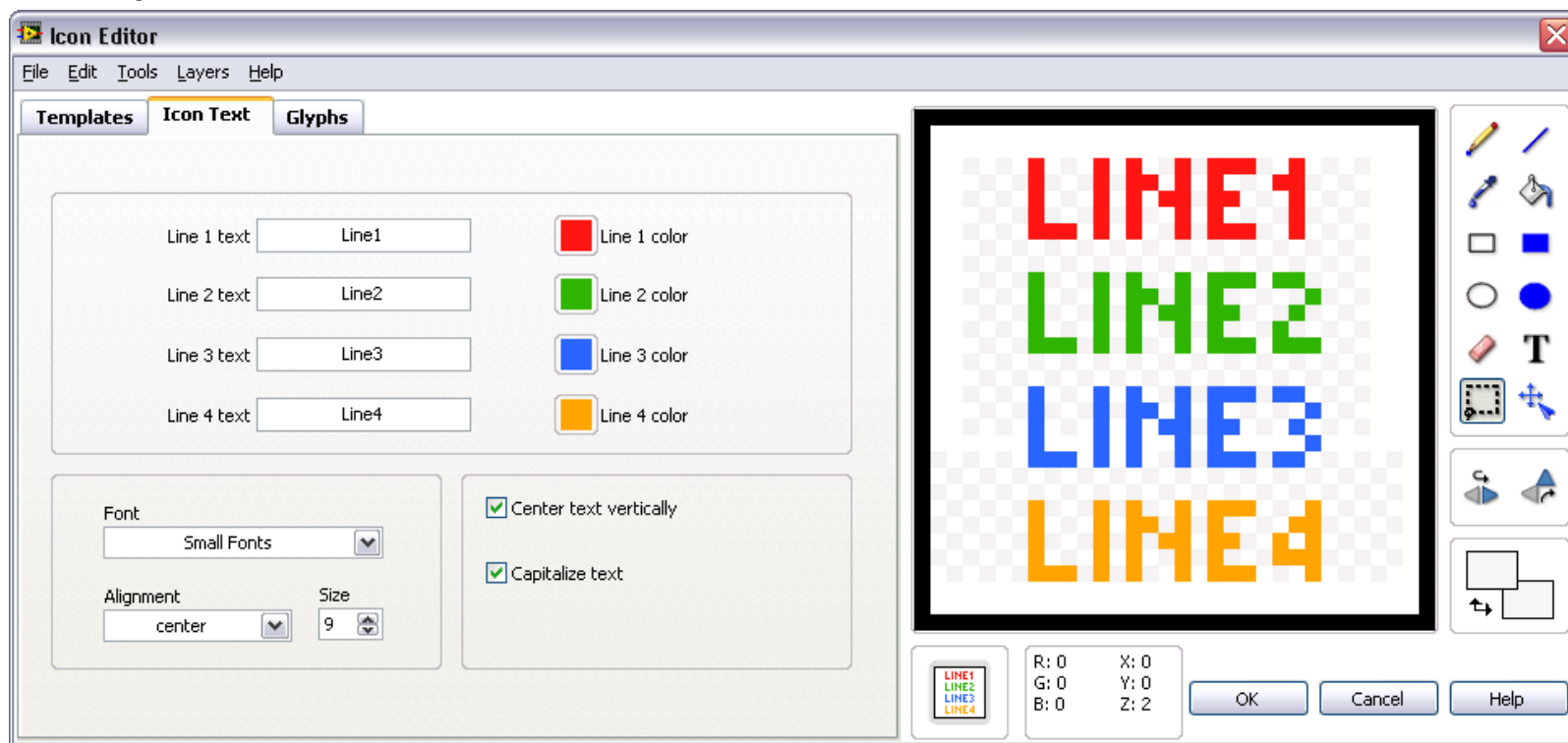
В. Иконка и панель подключения – Создание иконки

- Используйте закладку Glyphs для просмотра глифов, которые можно вставить в иконку
- Для обновления коллекции глифов выберите **Tools»Synchronize with ni.com Icon Library**



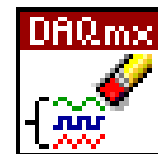
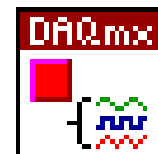
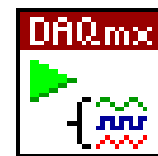
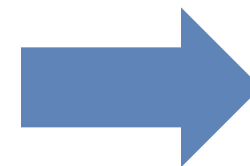
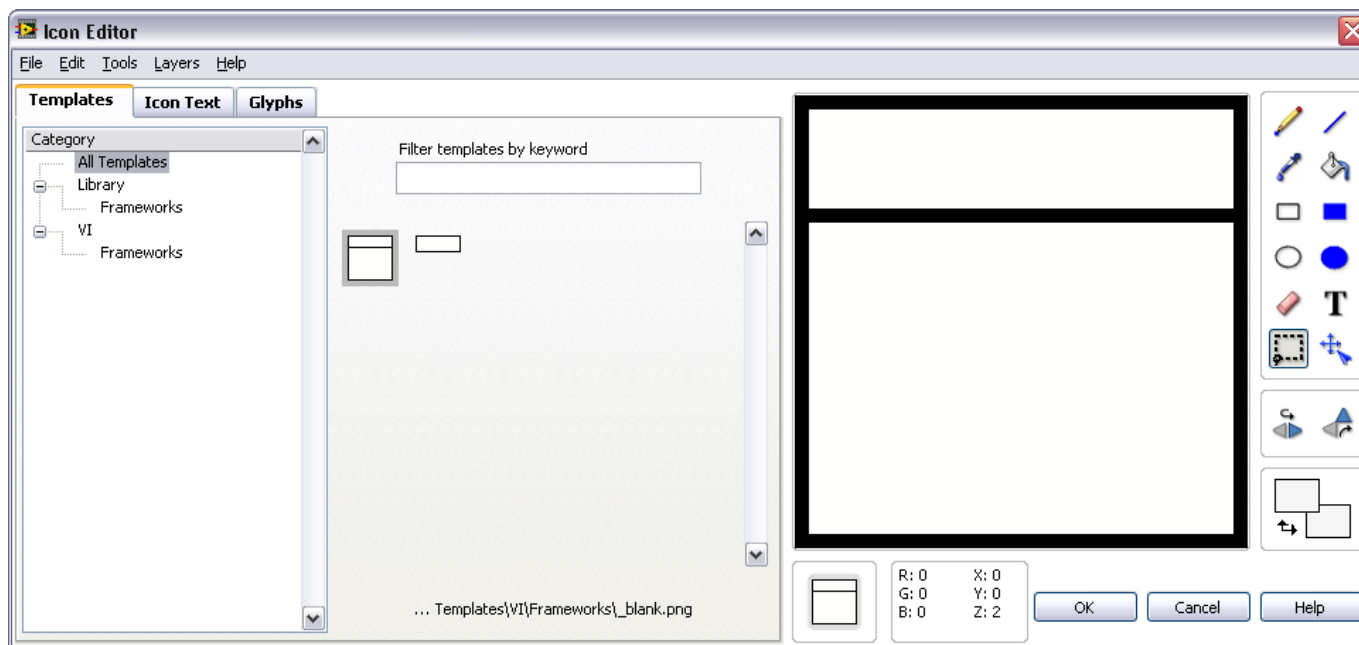
В. Иконка и панель подключения – Создание иконки

- Используйте закладку Icon Text для настройки текста, отображаемого в иконке



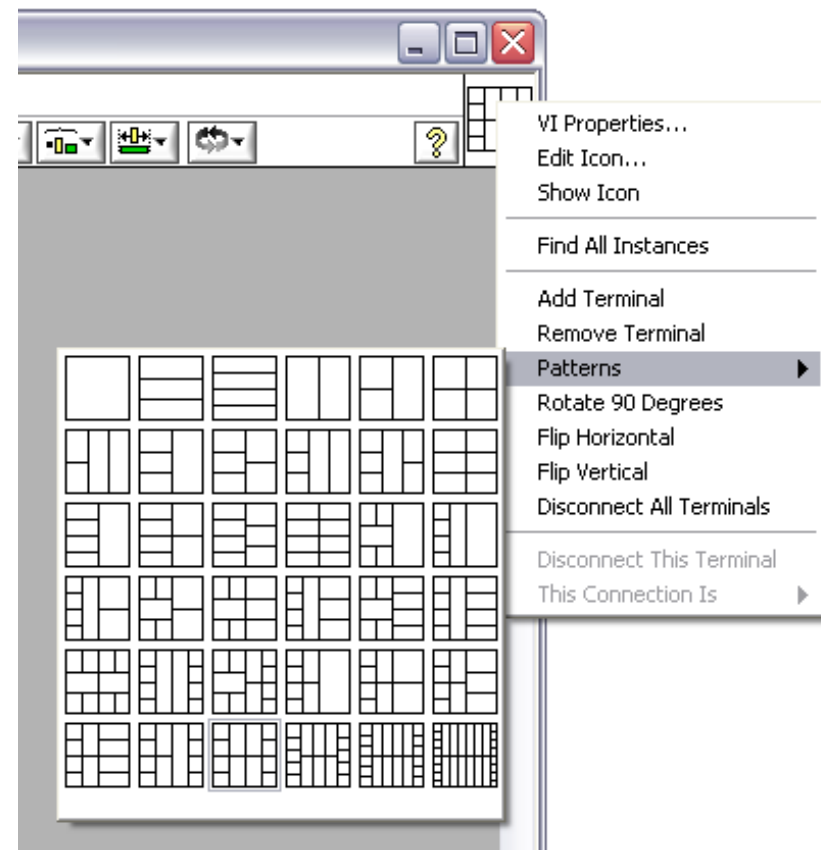
В. Иконка и панель подключения – Создание иконки

- Используйте закладку Templates для отображения шаблонов иконок, которые можно использовать в качестве основы создаваемой иконки



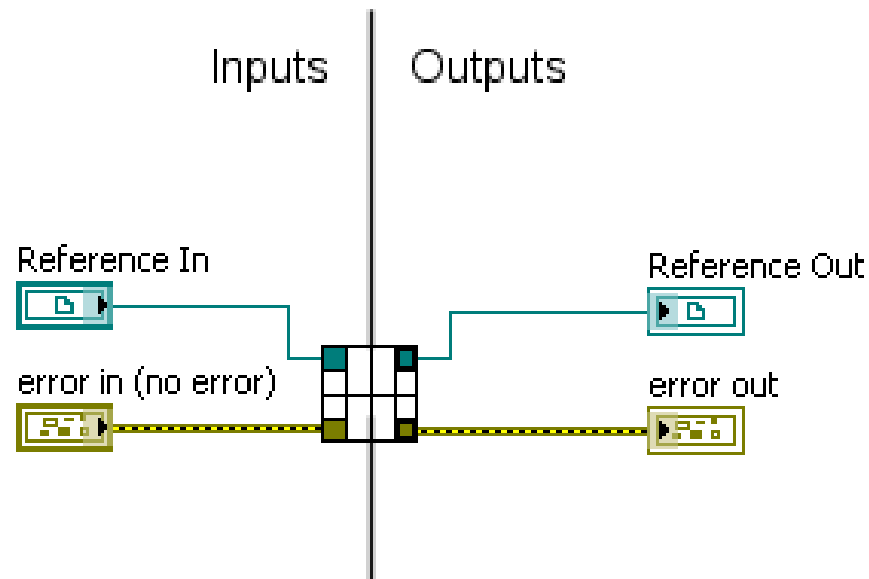
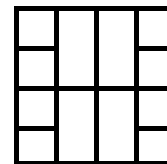
В. Иконка и панель подключения – Настройка панели подключения

- Щелкните правой кнопкой в верхнем правом углу лицевой панели и выберите **Show Connector**
 - Каждый прямоугольник панели подключения представляет терминал
 - Используйте терминалы для присвоения входам и выходам
- Выбирайте различные шаблоны, щелкнув правой кнопкой по панели подключения и выбрав в контекстном меню **Patterns**



В. Иконка и панель подключения – Стандарты

- Используйте эту схему панели подключения в качестве стандарта
- Верхние терминалы обычно резервируют для ссылок, например, ссылок на файлы
- Нижние терминалы обычно резервируют для кластеров ошибок

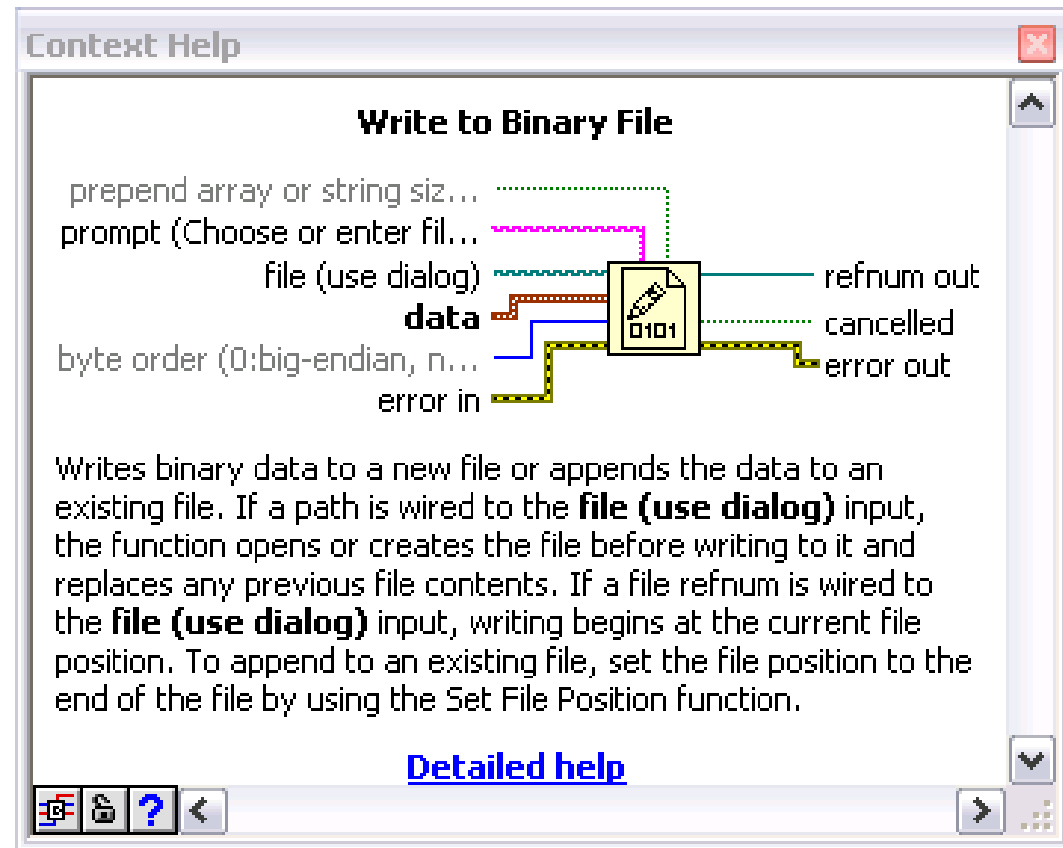


С. Использование SubVI

- Для размещения subVI на блок-диаграмме
 - Щелкните по **Select a VI** в палитре **Functions**
 - Отыщите VI, который вы хотите использовать, как subVI
 - Щелкните дважды, чтобы поместить subVI на блок-диаграмму
- Чтобы поместить открытый VI на блок-диаграмму другого открытого VI
 - Щелкните по иконке VI, который вы хотите использовать, как subVI
 - Перетащите иконку на блок-диаграмму другого VI

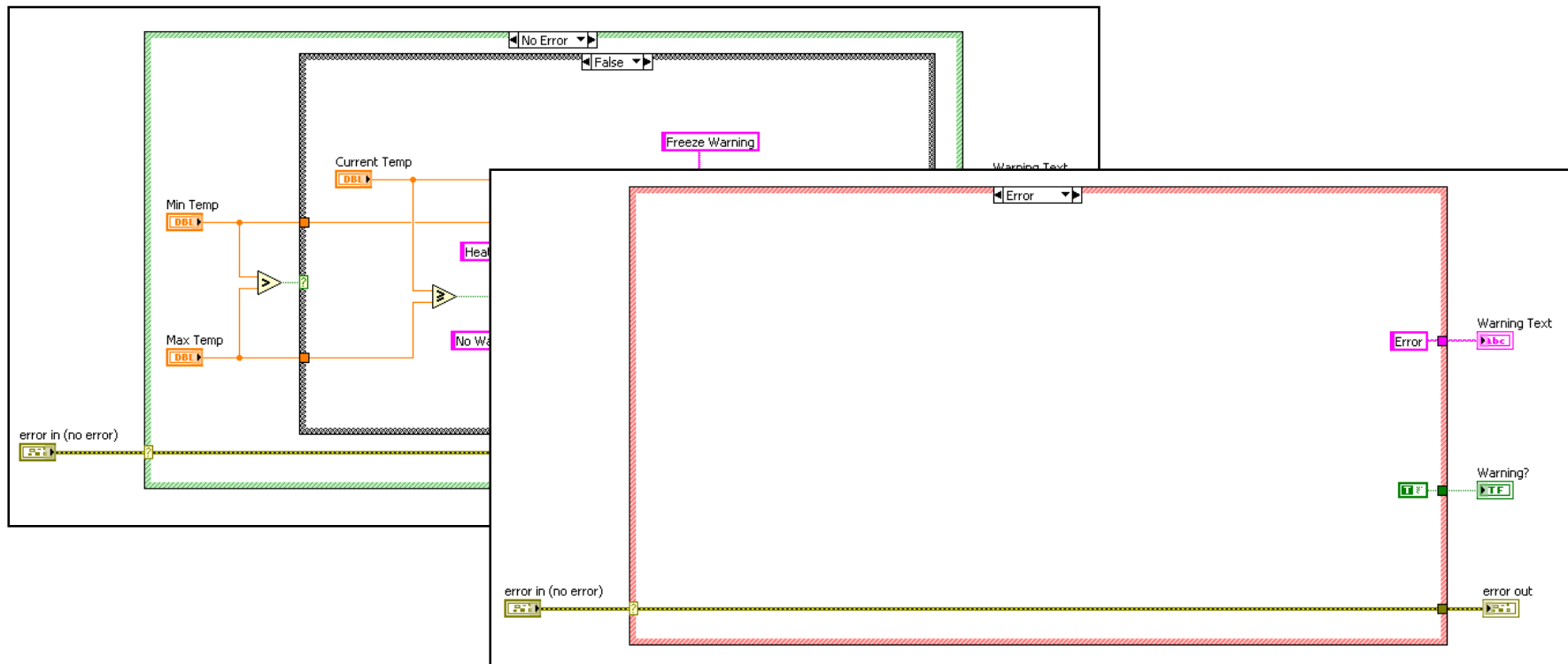
С. Использование SubVI – Настройка терминалов

- **Bold:** Обязательные терминалы
- Plain: Рекомендованные терминалы
- Dimmed: Необязательные терминалы



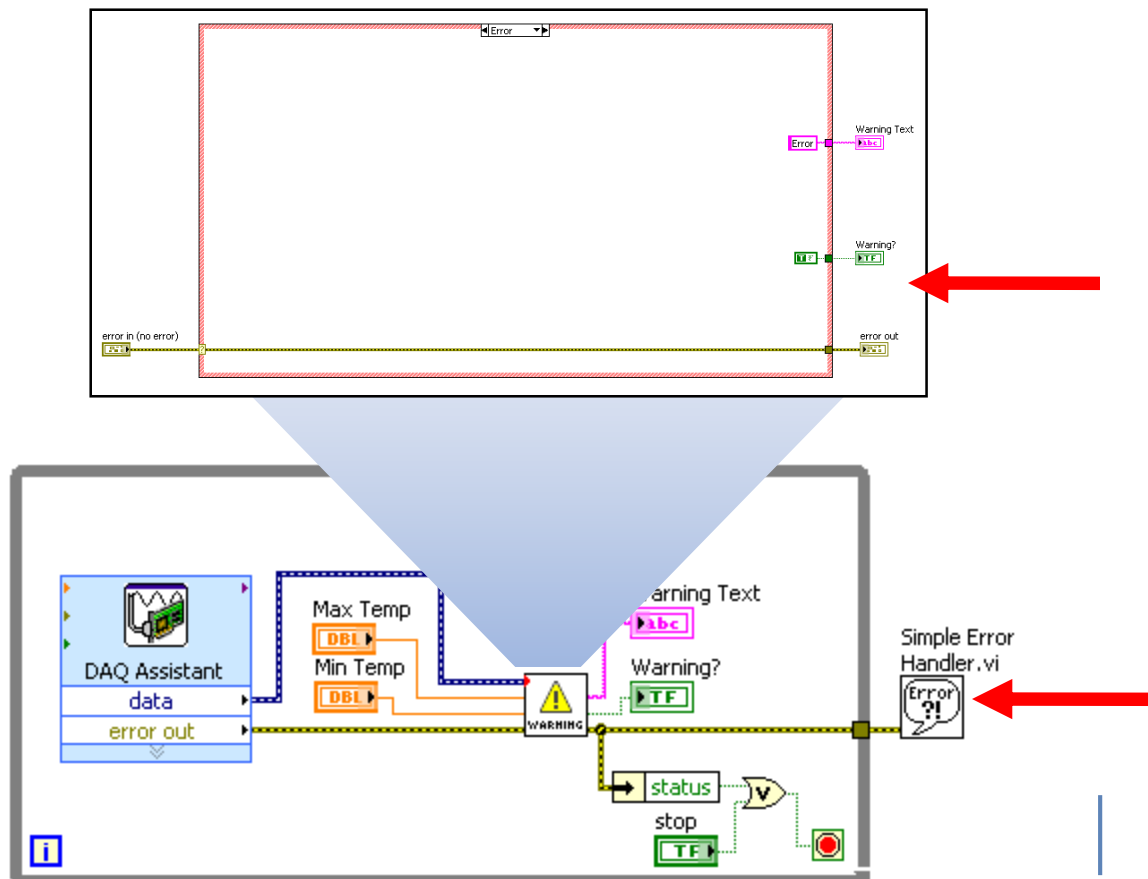
С. Использование SubVI – Обработка ошибок

- Используйте структуру Case для обработки ошибок, проходящих в subVI



С. Использование SubVI – Обработка ошибок

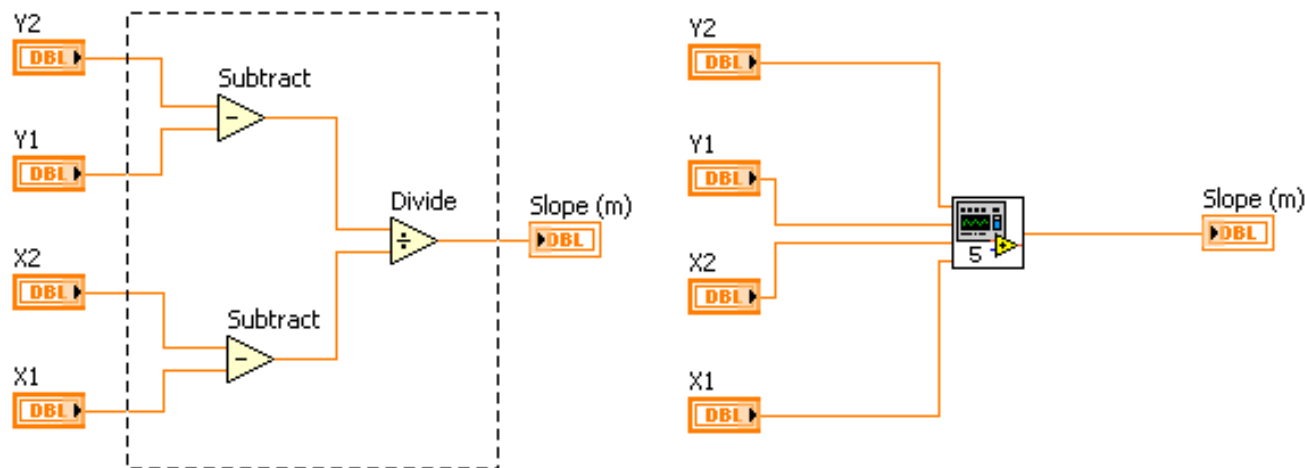
- Избегайте использования VI обработчика ошибок LabVIEW внутри subVI



С. Использование SubVI – Преобразование фрагментов VI в SubVI

Для преобразования фрагмента VI в subVI:

- Инструментом Positioning выделите фрагмент блок-диаграммы, который вы хотите использовать повторно
- Выберите в меню **Edit»Create SubVI**



Упражнение 7-1

Determine Warnings VI

Создайте иконку и панель подключения VI, чтобы его можно было использовать в качестве subVI.

GOAL

ЦЕЛЬ

Упражнение 7-1

Determine Warnings VI

- Что нужно сделать, если для этого subVI необходимо 20 входов и выходов?

ДИСКУССИЯ

DISCUSSION

Заключение – Контрольный вопрос

1. Какая настройка терминала subVI может послужить причиной ошибки, если этот терминал не подключен?
 - a) Required
 - b) Recommended
 - c) Optional

Заключение – Контрольный вопрос

1. Какая настройка терминала subVI может послужить причиной ошибки, если этот терминал не подключен?
 - a) **Required**
 - b) Recommended
 - c) Optional

Заключение – Контрольный вопрос

2. Вы должны создать специальную иконку для использования VI в качестве subVI.
 - a) True (Да)
 - b) False (Нет)

Заключение – Контрольный вопрос

2. Вы должны создать специальную иконку для использования VI в качестве subVI.
 - a) True (Да)
 - b) False (Нет)

Нет необходимости создавать специальную иконку для использования VI в качестве subVI, но это настоятельно рекомендуется для того, чтобы ваш код был более читабельным.

Лекция 8

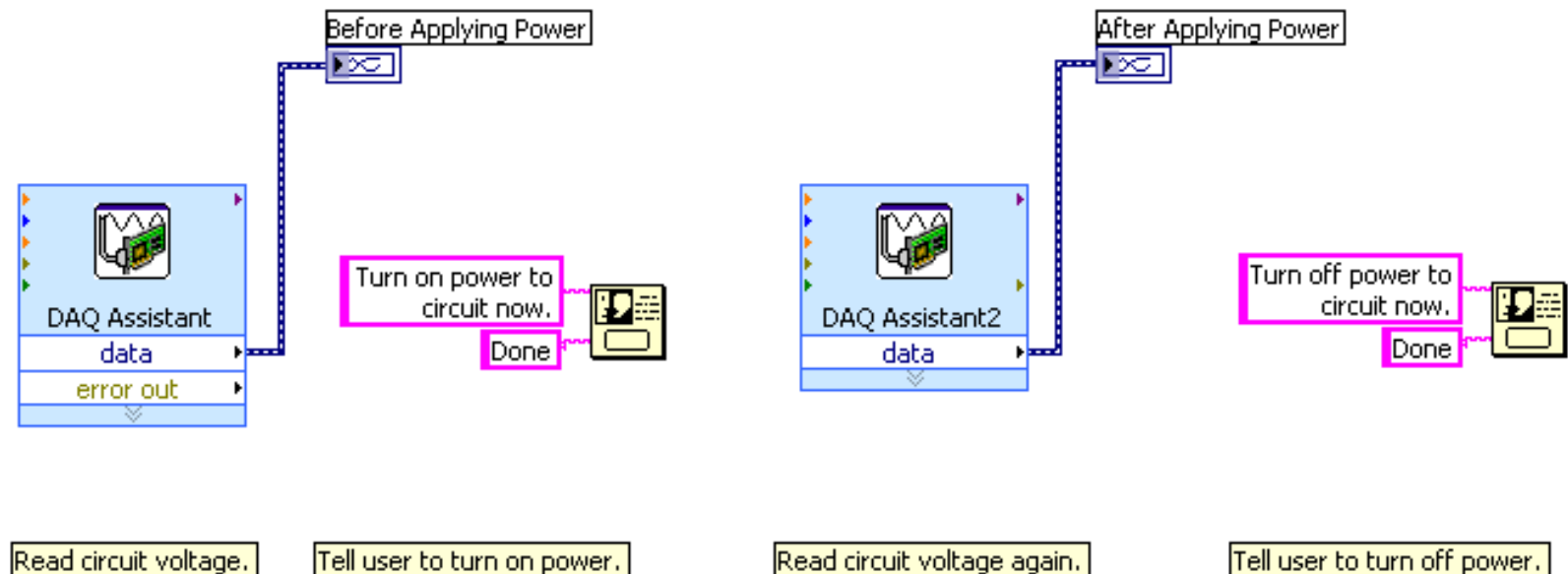
Общепринятая методика проектирования и шаблоны

ТЕМЫ

- А. Программирование последовательностей
- В. Программирование состояний
- С. Конечные автоматы

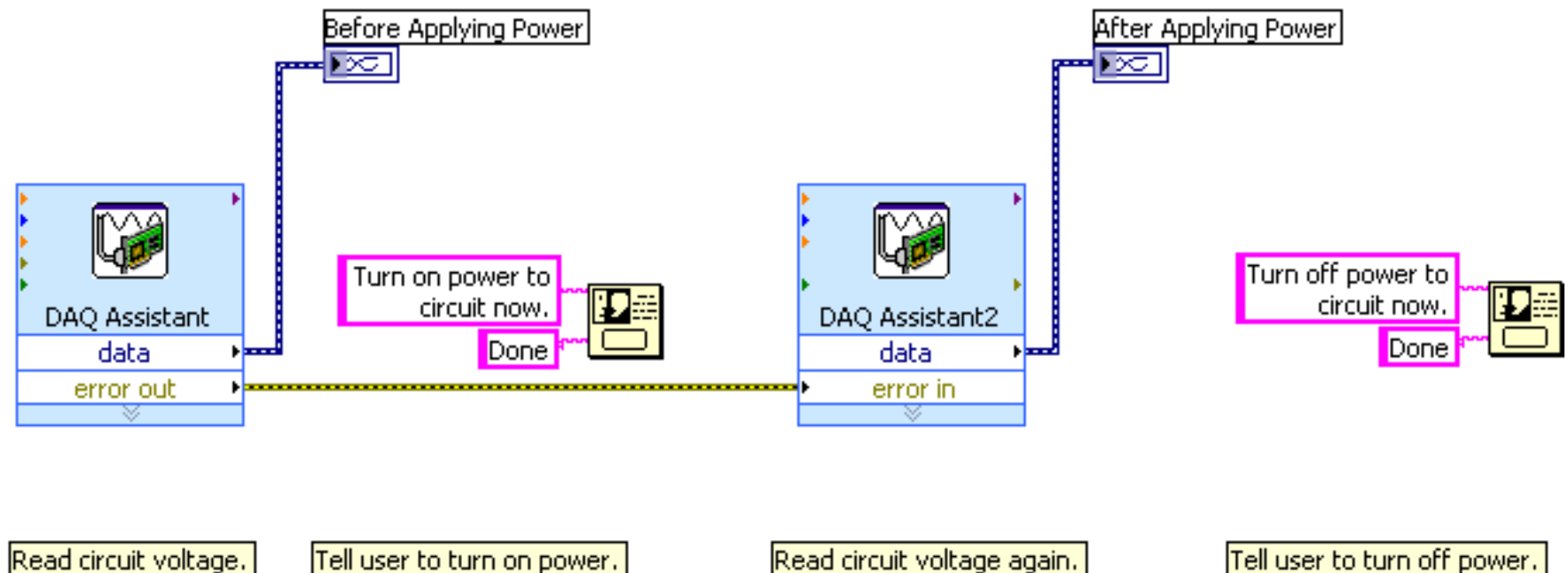
А. Программирование последовательностей

- Многие VI должны выполнять задачи последовательно
- Это необходимо в таких блок-диаграммах, где важен определенный порядок выполнения задач – какая-то одна из этих задач должна выполняться первой



А. Программирование последовательностей

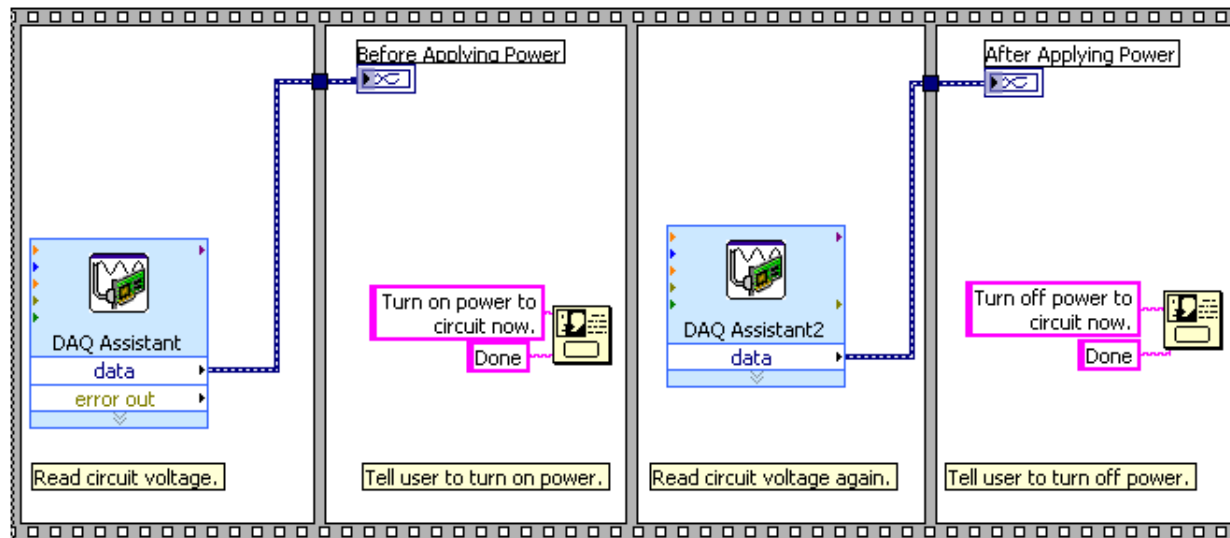
Используйте кластеры ошибок, чтобы задать порядок выполнения



А. Программирование последовательностей

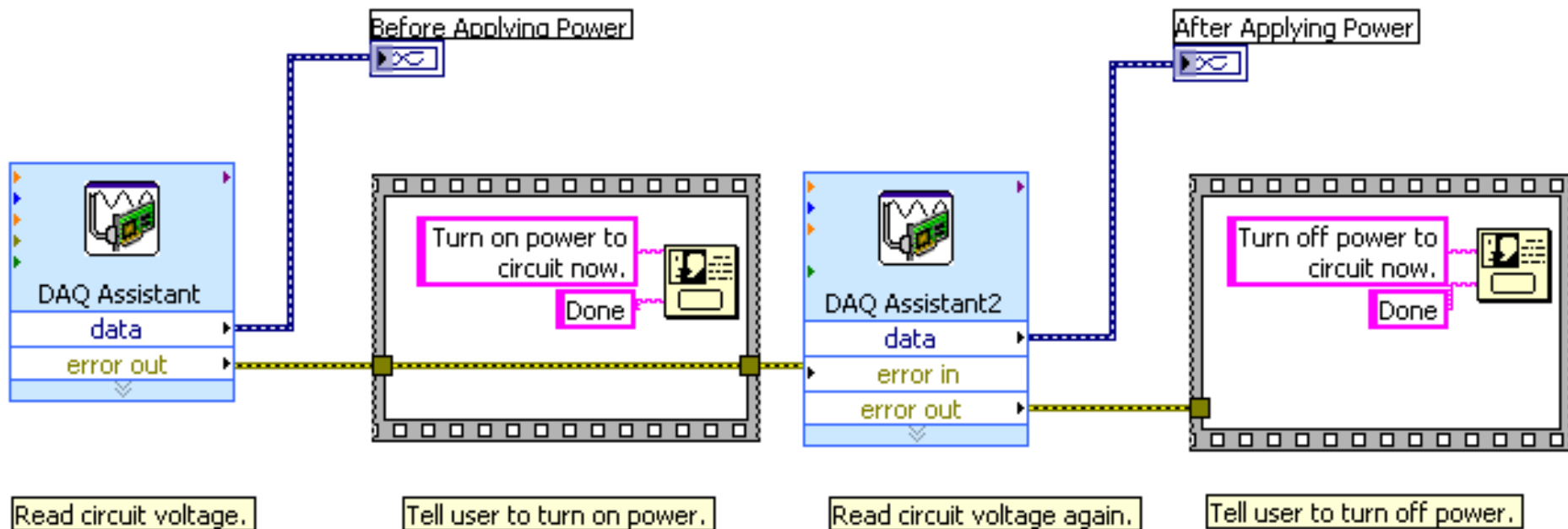
Для задания последовательности выполнения используйте структуру Sequence

- Структура состоит из фреймов, каждый из которых выполняется по порядку следования
- Второй фрейм не начнет выполняться, пока не будет выполнено все, что требуется в первом фрейме



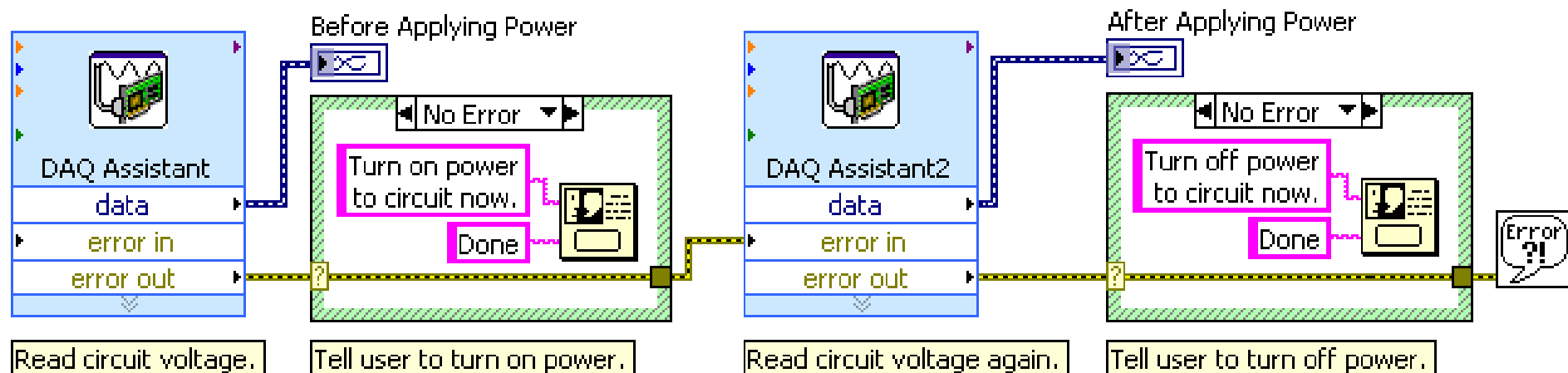
А. Программирование последовательностей

- Не злоупотребляйте использованием структур Sequence
- Остановить выполнение последовательности на каком либо фрейме нельзя



A. Программирование последовательностей

Лучший способ разработки такого VI – заключить диалоговые окна в структуры Case, соединяя кластеры ошибок селектором выбора структуры Case



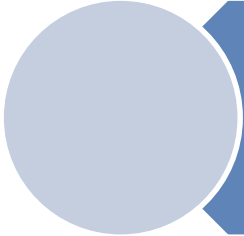
В. Программирование состояний

Хотя структуры Sequence или последовательно соединенные subVI соответствуют назначению программирования последовательности, не всегда их выбор является лучшим:

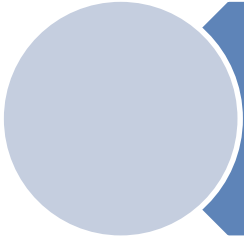
- Что делать, если вам необходимо изменить порядок выполнения последовательности?
- Что делать, если вам необходимо повторять один из элементов последовательности более часто, чем другие?
- Что делать, если некоторые элементы последовательности нужно выполнять только при определенных условиях?
- Что делать, если вам необходимо остановить программу немедленно, до завершения выполнения всей последовательности?

В. Программирование состояний – граф состояний-переходов

Разновидность блок-схемы, в которой отражаются состояния программы и переходы между состояниями



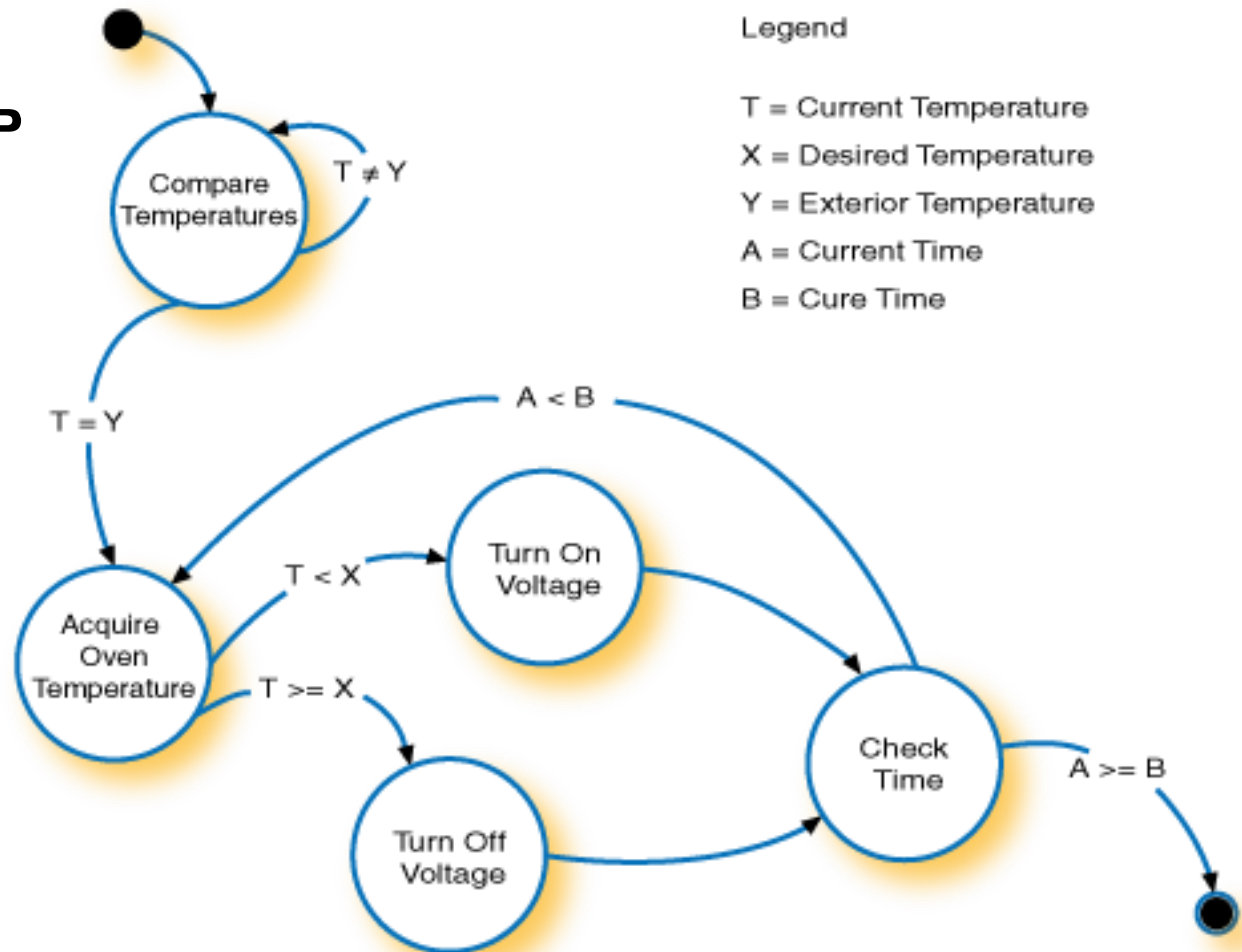
State – состояние – часть программы, которая удовлетворяет условию выполнения действия или ожиданию события



Transition – переход – условие, действие или событие которое вызывает переход в следующее состояние

В. Программирование состояний – граф состояний-переходов

Пример: печь

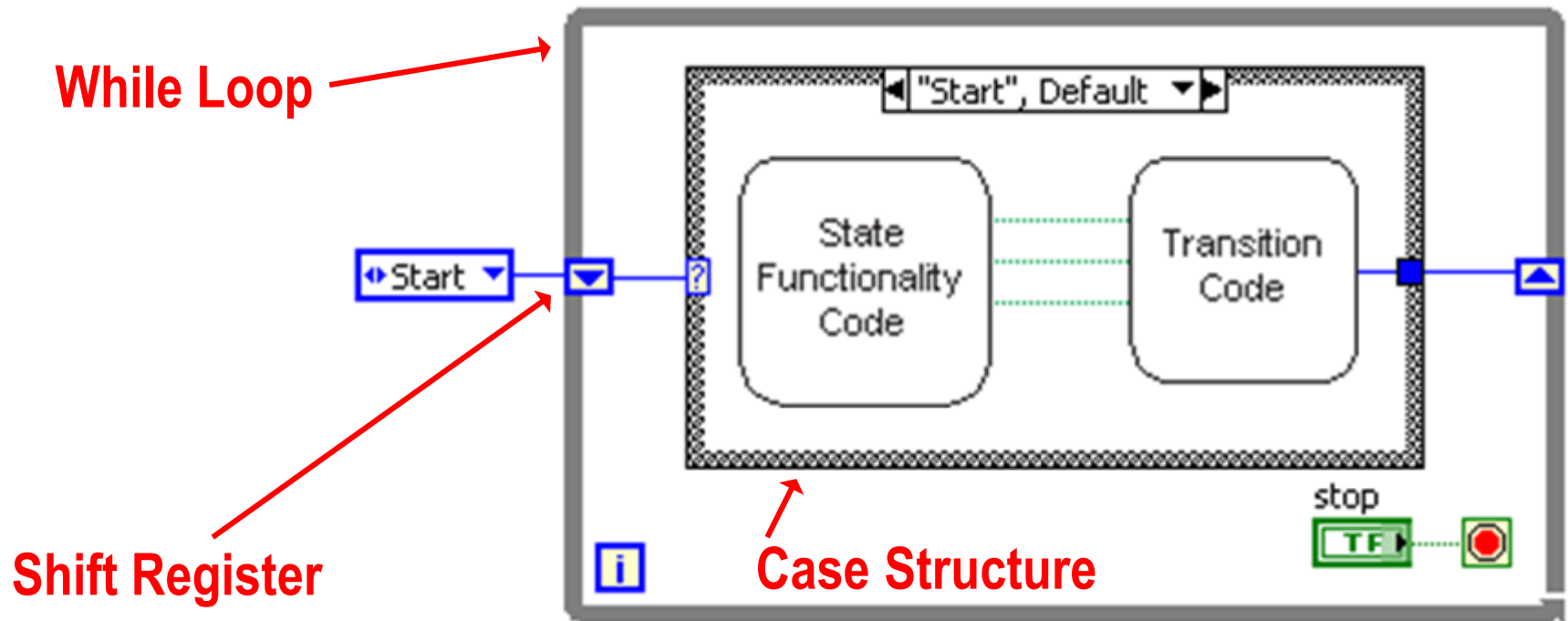


С. Конечные автоматы

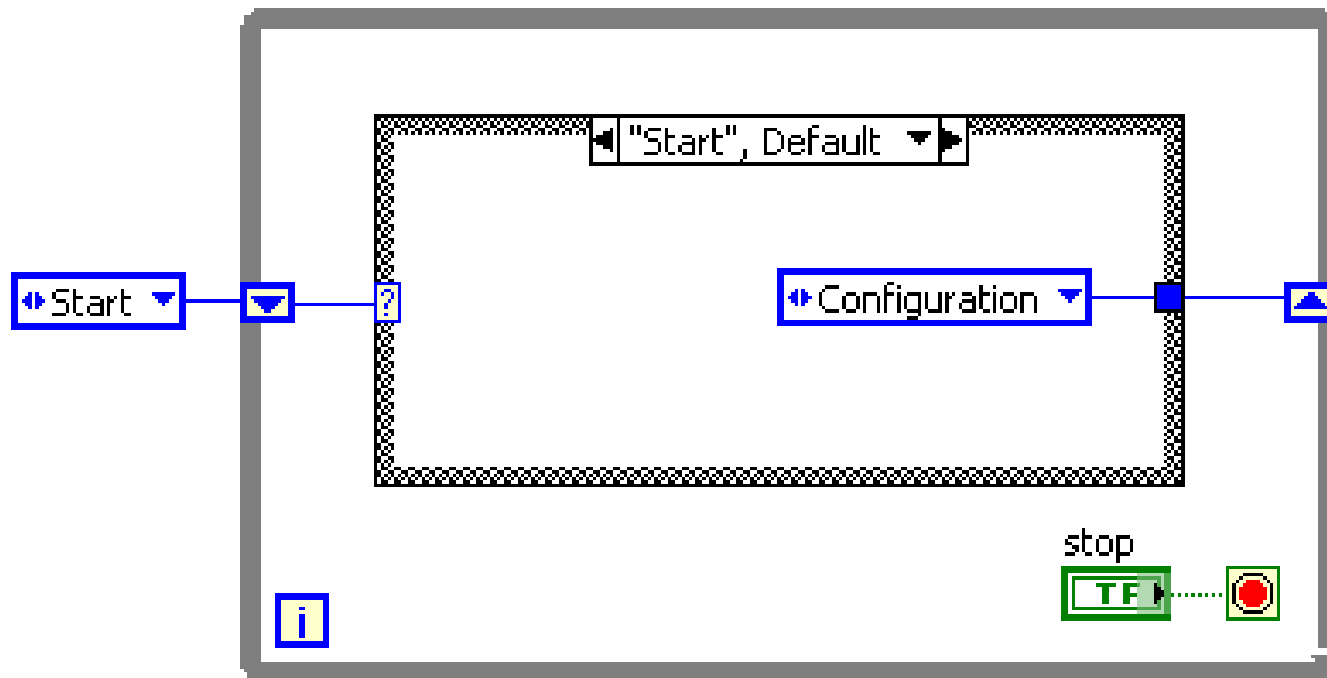
- Шаблон проекта конечного автомата реализует диаграмму состояний или блок-схему последовательности действий
- Когда используют конечный автомат?
 - Обычно используют при создании интерфейса пользователя, где различные действия оператора переводят пользовательский интерфейс в различные состояния
 - Обычно используют в процедурах тестирования, в которых каждый этап процесса представляет собой состояние

С. Конечные автоматы – Инфраструктура

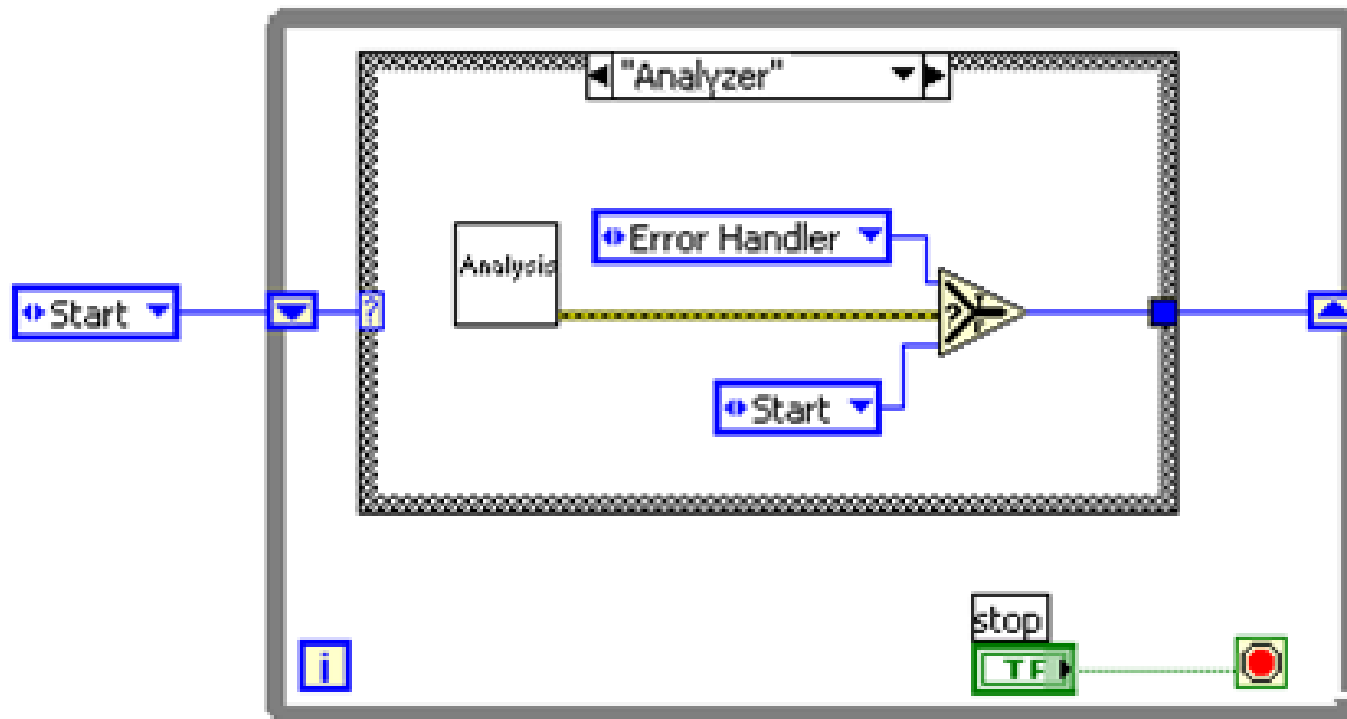
- Конечный автомат состоит из набора состояний и функций перехода, которые определяют следующее состояние
- Из каждого состояния можно перейти в одно или несколько состояний или выйти из процесса



С. Конечные автоматы – Переход по умолчанию

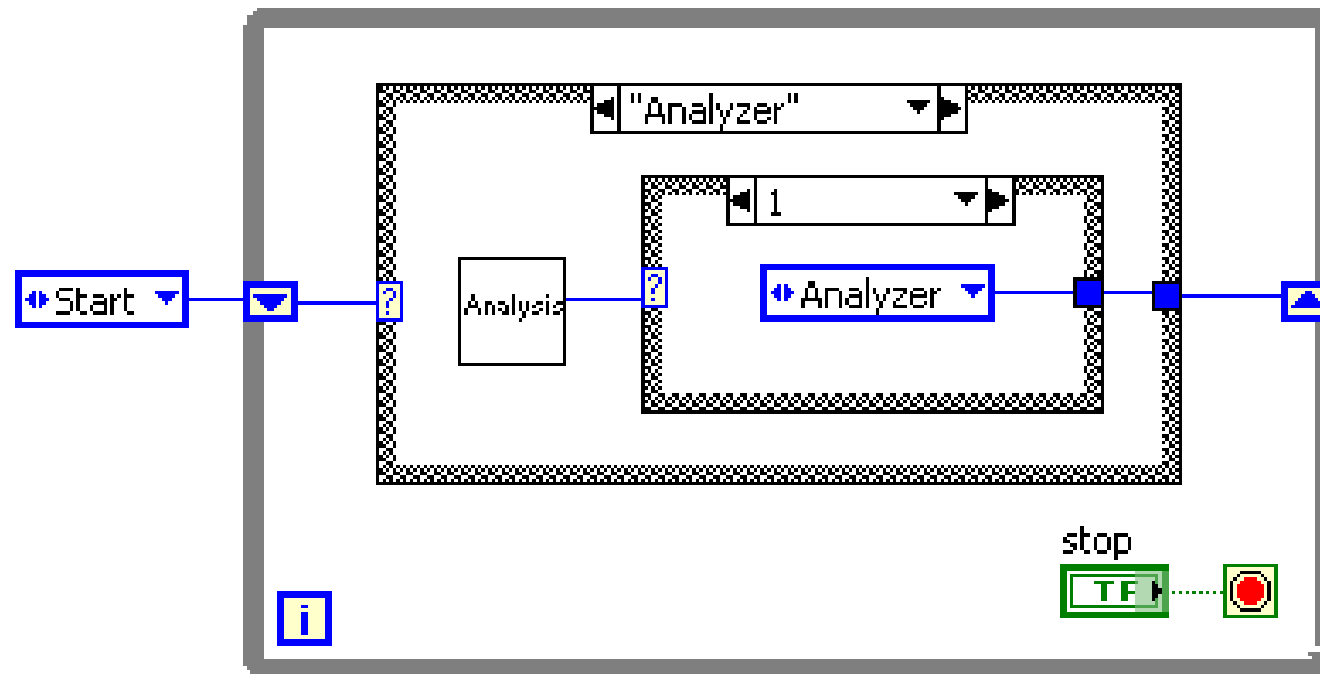


С. Конечные автоматы – Переход между двумя состояниями

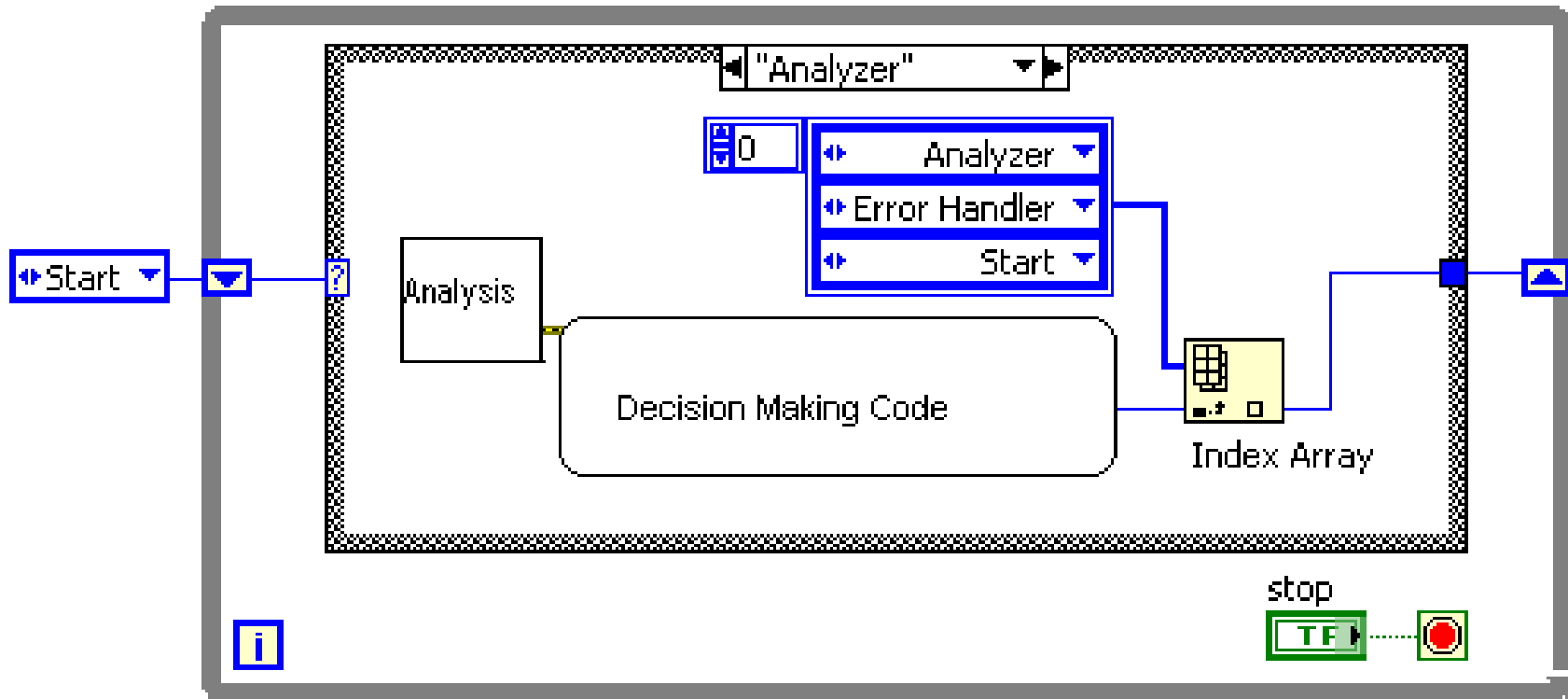


С. Конечные автоматы – Переходы и структура

Case



С. Конечные автоматы – Переход с использованием массива переходов



Курсовой проект

Демонстрирует реализацию конечного автомата.

<Exercises>\LabVIEW Core 1\Demonstrations\Course Project

DEMONSTRATION

Упражнение 8-1

State Machine VI

Создайте простой конечный автомат, используя элемент управления `type-defined enumerated` и практического опыта, полученного при изучении всего курса.

GOAL

ЦЕЛЬ

Упражнение 8-1

State Machine VI

- Если вы хотите добавить Process 3 в программу, как нужно модифицировать VI?

ДИСКУССИЯ

DISCUSSION

Заключение – Контрольный вопрос

1. При использовании структуры Sequence вы можете остановить выполнение на любой стадии последовательности.
 - a) True (Да)
 - b) False (Нет)

Заключение – Контрольный вопрос

1. При использовании структуры Sequence вы можете остановить выполнение на любой стадии последовательности.
 - a) True (Да)
 - b) False (Нет)**

Вы не можете остановить выполнение на любой стадии последовательности.

Заключение – Контрольный вопрос

2. Какие из следующих преимуществ дает применение конечного автомата взамен последовательной структуры?
- a) Вы можете изменять порядок выполнения в последовательности
 - b) Вы можете повторять отдельные элементы последовательности
 - c) Вы можете задать условия, которые определяют, когда должен выполняться элемент последовательности
 - d) Вы можете остановить выполнение программы в любом месте последовательности

Заключение – Контрольный вопрос

2. Какие из следующих преимуществ дает применение конечного автомата взамен последовательной структуры?
- a) **Вы можете изменять порядок выполнения в последовательности**
 - b) **Вы можете повторять отдельные элементы последовательности**
 - c) **Вы можете задать условия, которые определяют, когда должен выполняться элемент последовательности**
 - d) **Вы можете остановить выполнение программы в любом месте последовательности**

Лекция 9

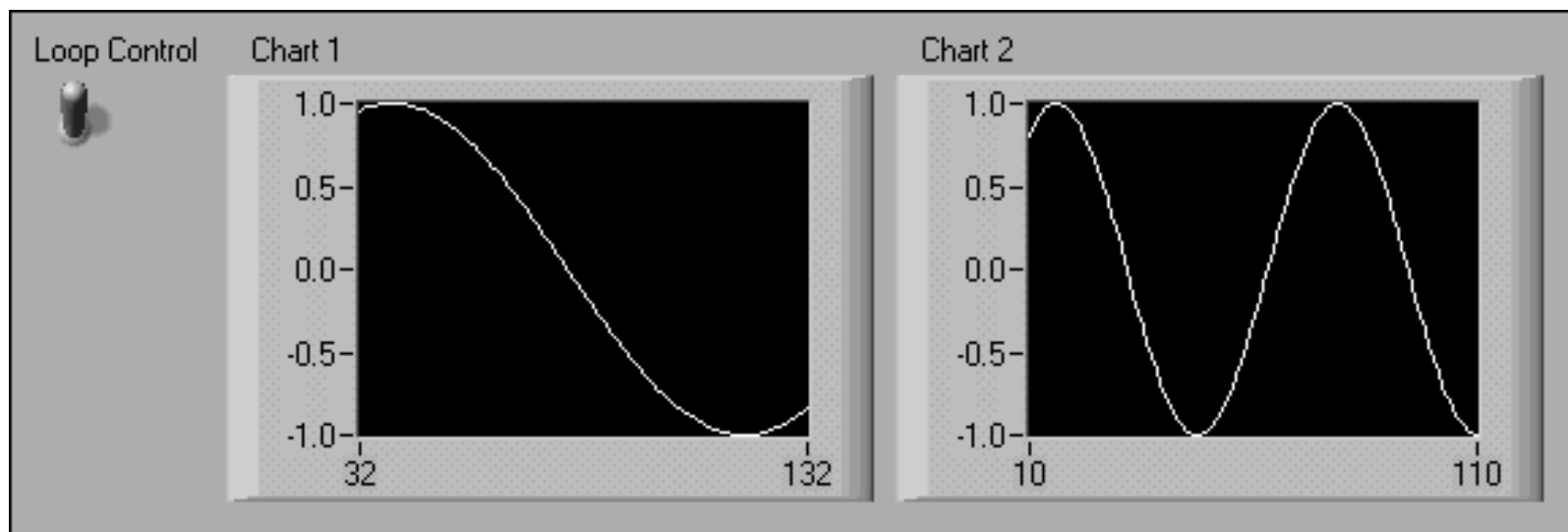
Использование переменных

ТЕМЫ

- A. Parallelism (Параллелизм)
- B. Variables (Переменные)
- C. Functional Global Variables (Функциональные глобальные переменные)
- D. Race Conditions (Состязания)

A. Parallelism

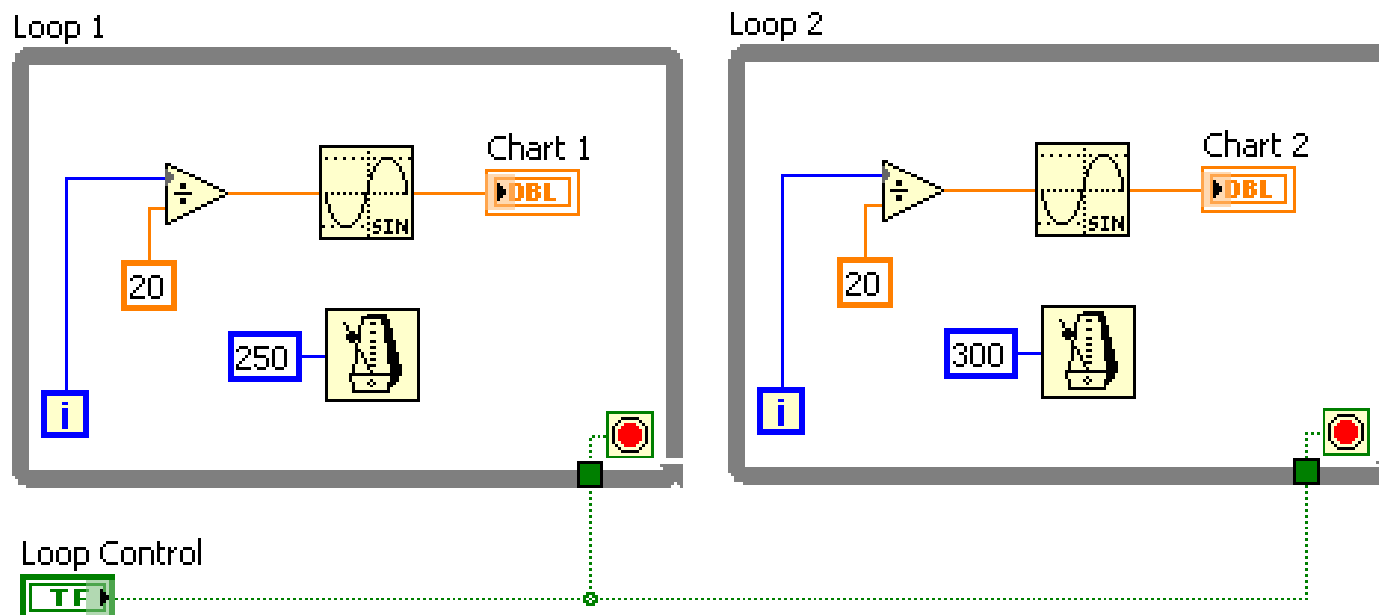
Выполнение нескольких задач одновременно



A. Parallelism

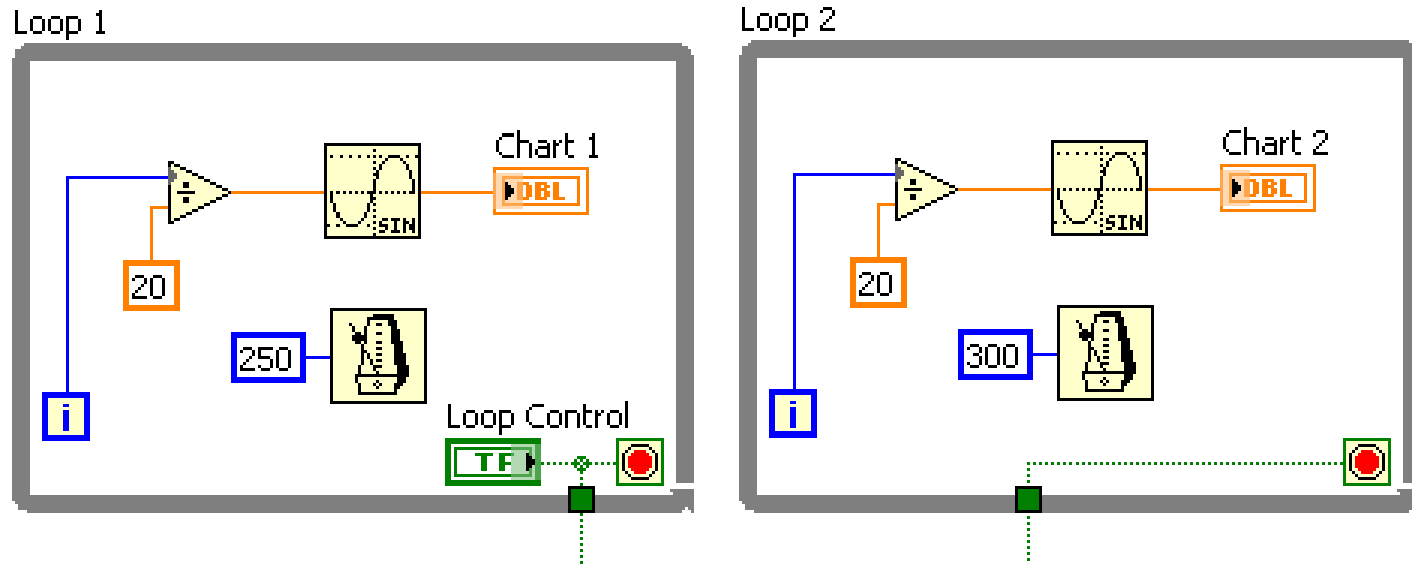
Передача данных между параллельными циклами является проблемой

Как выполняется останов циклов в этом примере?



A. Parallelism

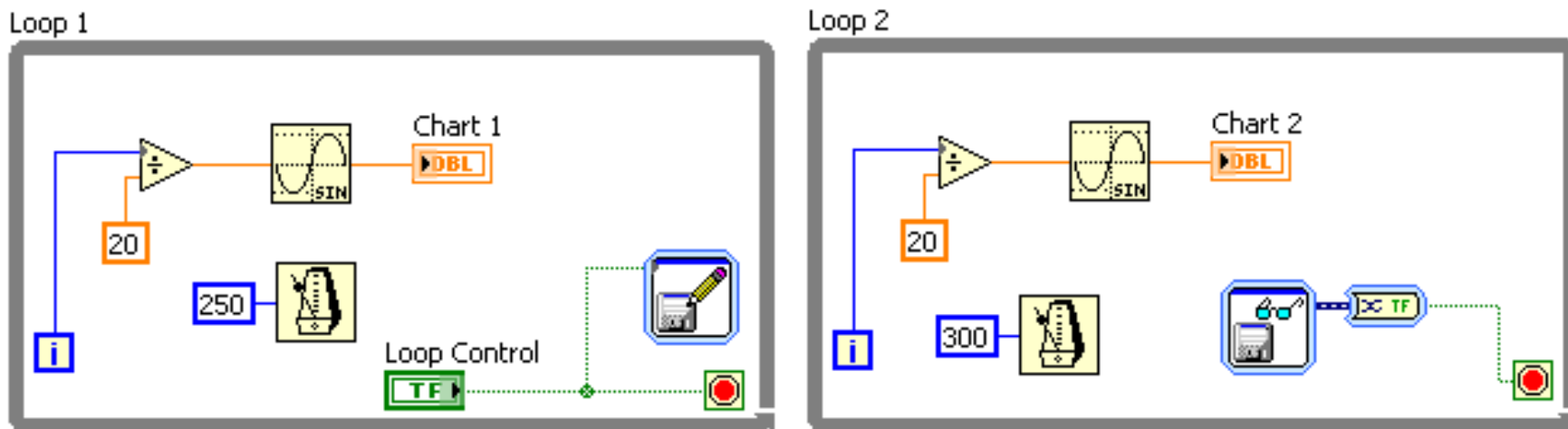
Как выполняется останов циклов в этом примере?



A. Parallelism

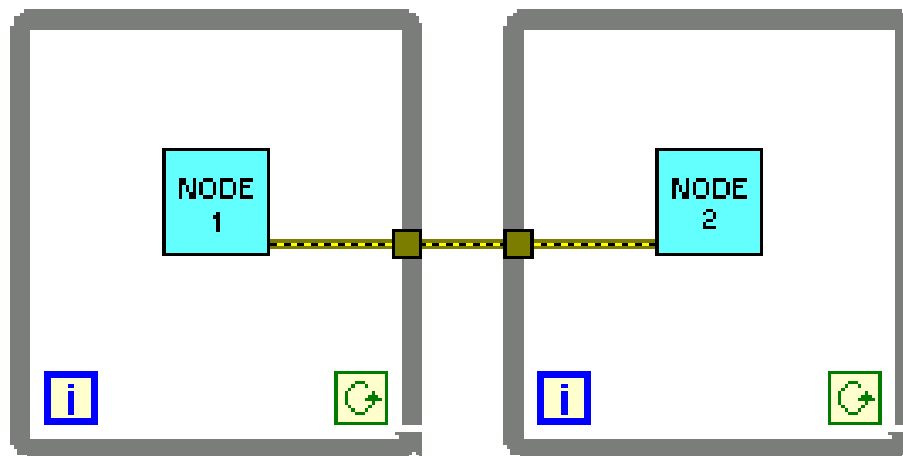
Состояние кнопки Stop читается из файла

- Каждый цикл независимо обращается к файлу
- Однако для чтения из файла и записи в файл требуется много времени

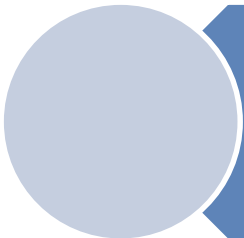


B. Variables (Переменные)

- Данные между параллельными циклами нельзя передавать по проводникам
- Переменные позволяют обойти проблему нормального потока данных путем передачи данных из одного места в другое без проводников



B. Variables (Переменные)



Variables – элементы блок-диаграммы, которые позволяют извлекать или сохранять данные из других мест

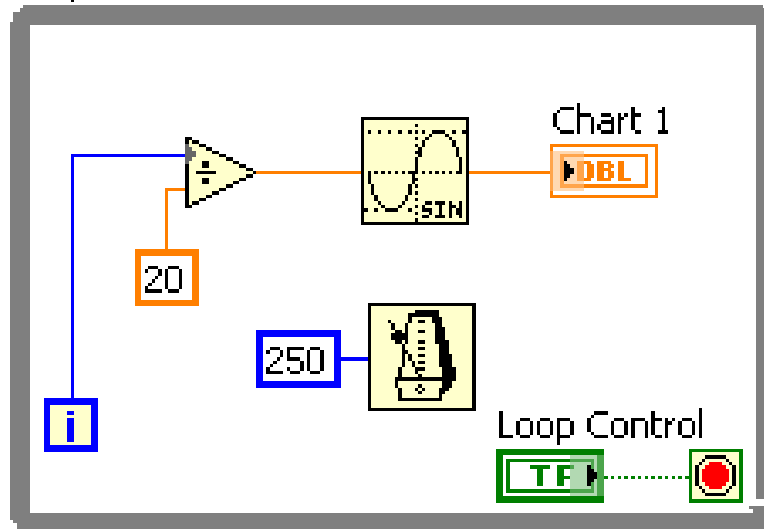
Переменные могут быть следующих типов:

- **Local** – локальные, хранят данные в элементах управления и индикации лицевой панели
- **Global** – глобальные, хранят данные в специальных репозиториях, которые могут быть доступны из разных VI
- **Functional Global** – функциональные глобальные, хранят данные в сдвиговых регистрах цикла While
- **Shared** – коллективно используемые (разделяемые), передают данные между различными распределенными целевыми устройствами, соединенными между собой сетью

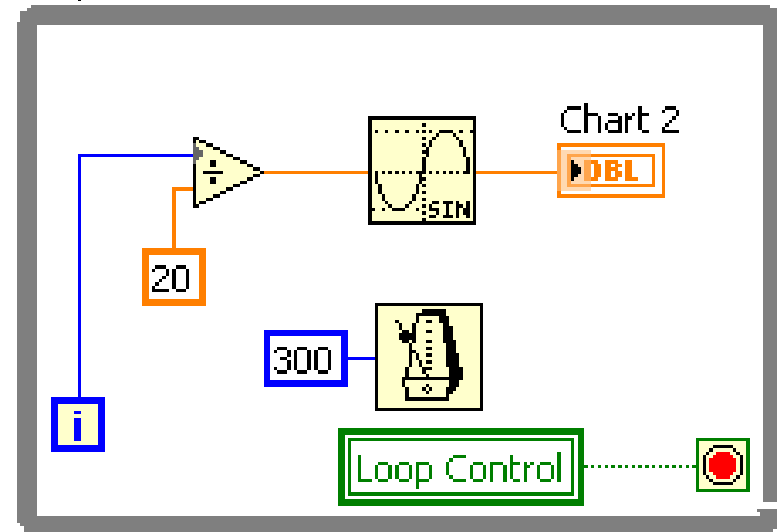
B. Variables – Переменные, использование в одном VI

Для обмена данными в одном VI используют local variables (локальные переменные)

Loop 1



Loop 2



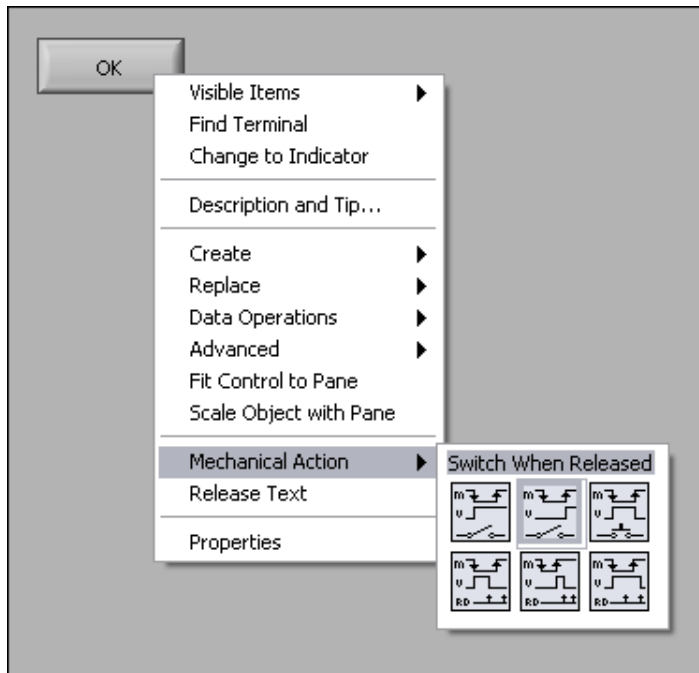
Создание Local Variables

Создание и использование локальных переменных.

ДЕМОНСТРАЦИЯ

V. Variables - Переменные

- Булевские элементы управления с ассоциированными с ними локальными переменными должны быть настроены на определенные механические действия
- Действие булевского элемента latch action (защелкнуть) несовместимо с локальными переменными



Упражнение 9-1

Local Variable VI

Используйте локальную переменную для записи и чтения в/из элемента управления.

GOAL

ЦЕЛЬ

Упражнение 9-1

Local Variable VI

- Какую функцию реализует локальная переменная в этом приложении?

ДИСКУССИЯ

DISCUSSION

B. Variables – Переменные для обмена данными между VI

Глобальные переменные (global variable) или разделяемые переменные одного процесса (single process shared variable) используются для совместного использования данных несколькими VI

- Global variable применяют для обмена данными между VI, выполняющимися на одном компьютере, особенно, если не используется файл проекта
- Single process shared variable применяют, если в будущем потребуется разделять информацию переменных между VI, выполняющимися на нескольких компьютерах

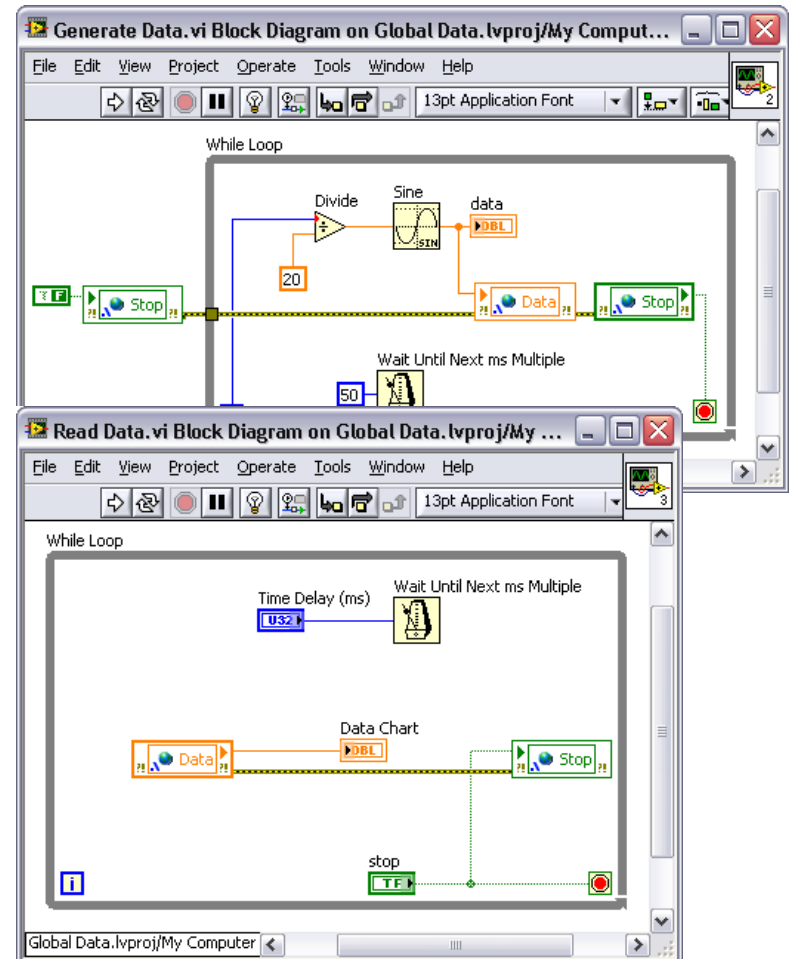
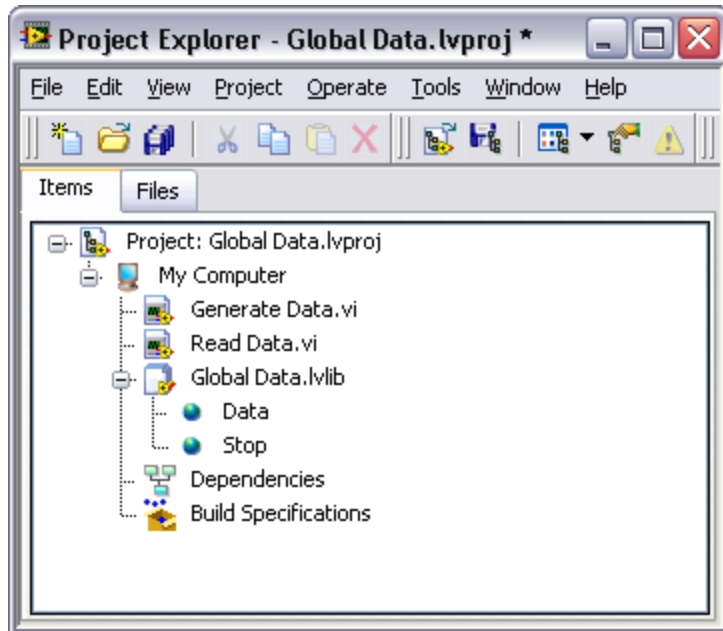
Создание Global Variables

Создайте и используйте глобальные переменные.

ДЕМОНСТРАЦИЯ

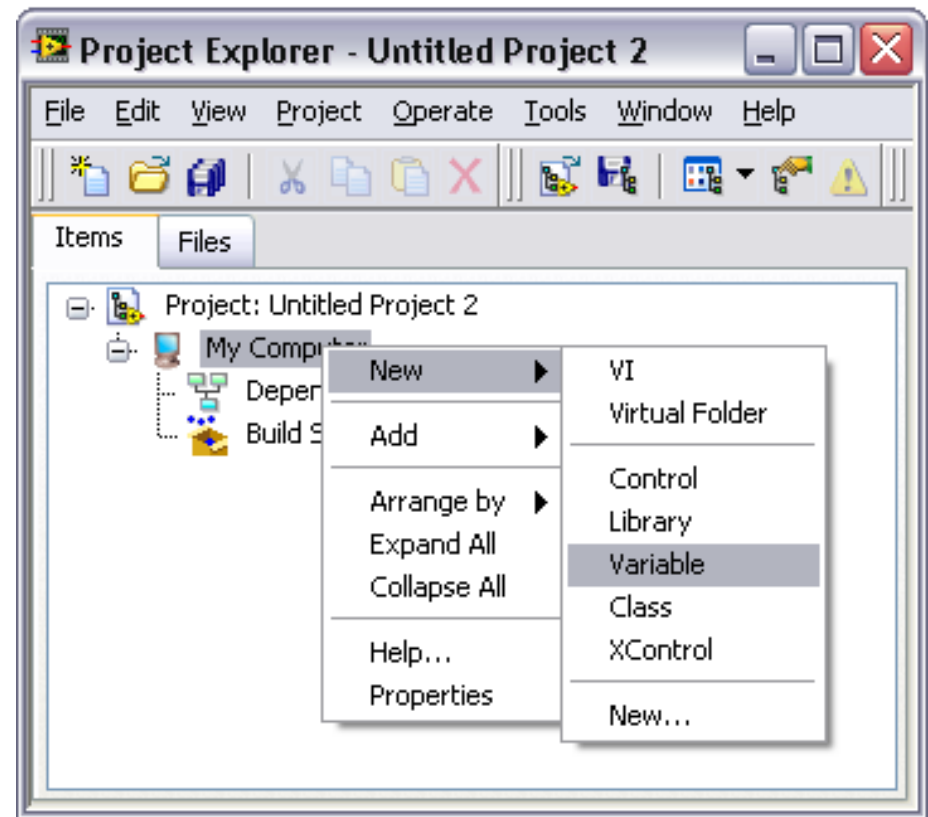
B. Variables – Переменные для обмена данными между VI

Single Process Shared Variables



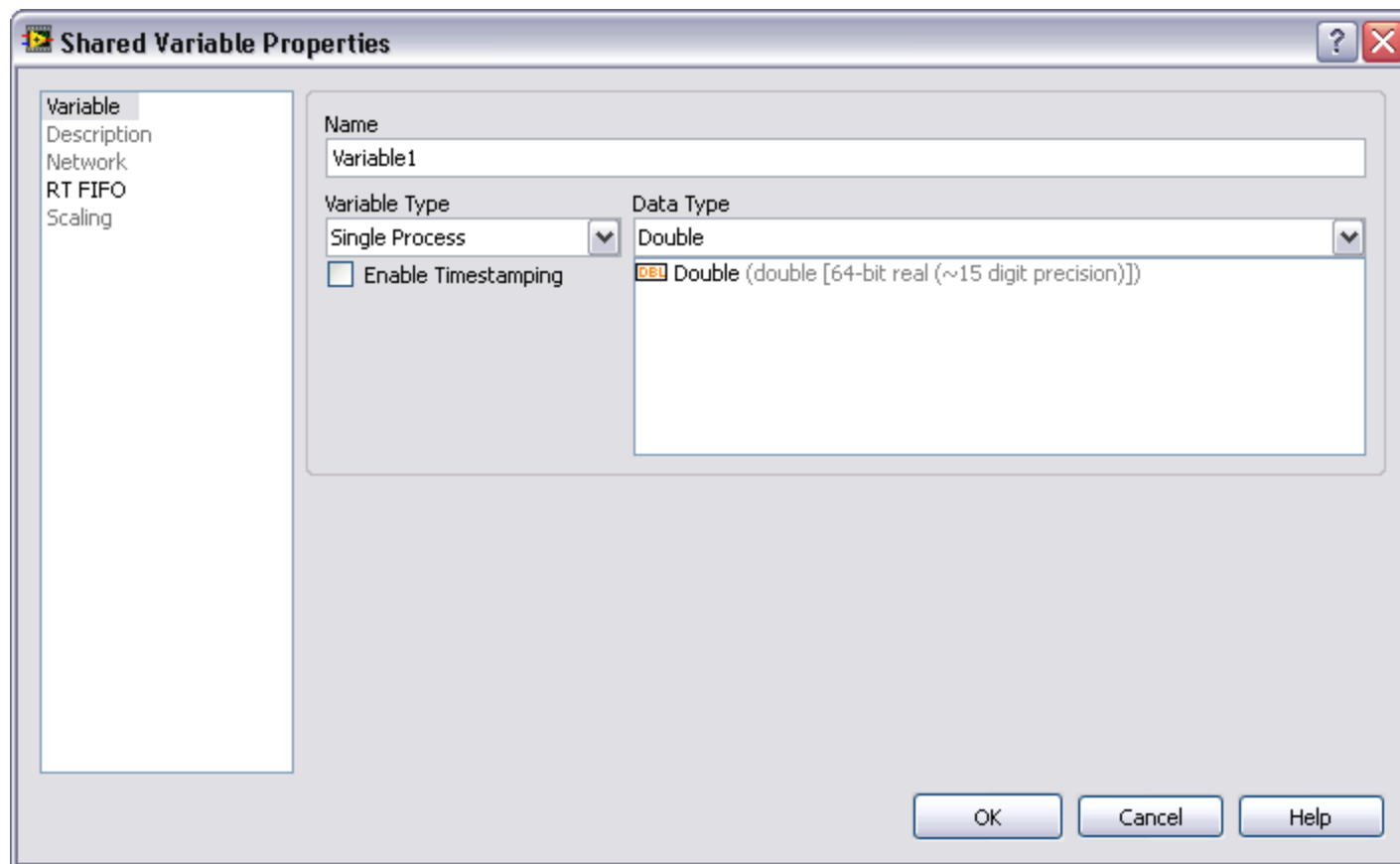
B. Variables – Создание разделяемых переменных Shared Variables

- Для создания shared variable нужно открыть проект
- Конфигурация переменных хранится в библиотеке Project Libraries
- LabVIEW автоматически создает библиотеку, если переменная создается не из существующей библиотеки



Shared Variables – Конфигурирование

Установка типа переменной **Single Process**

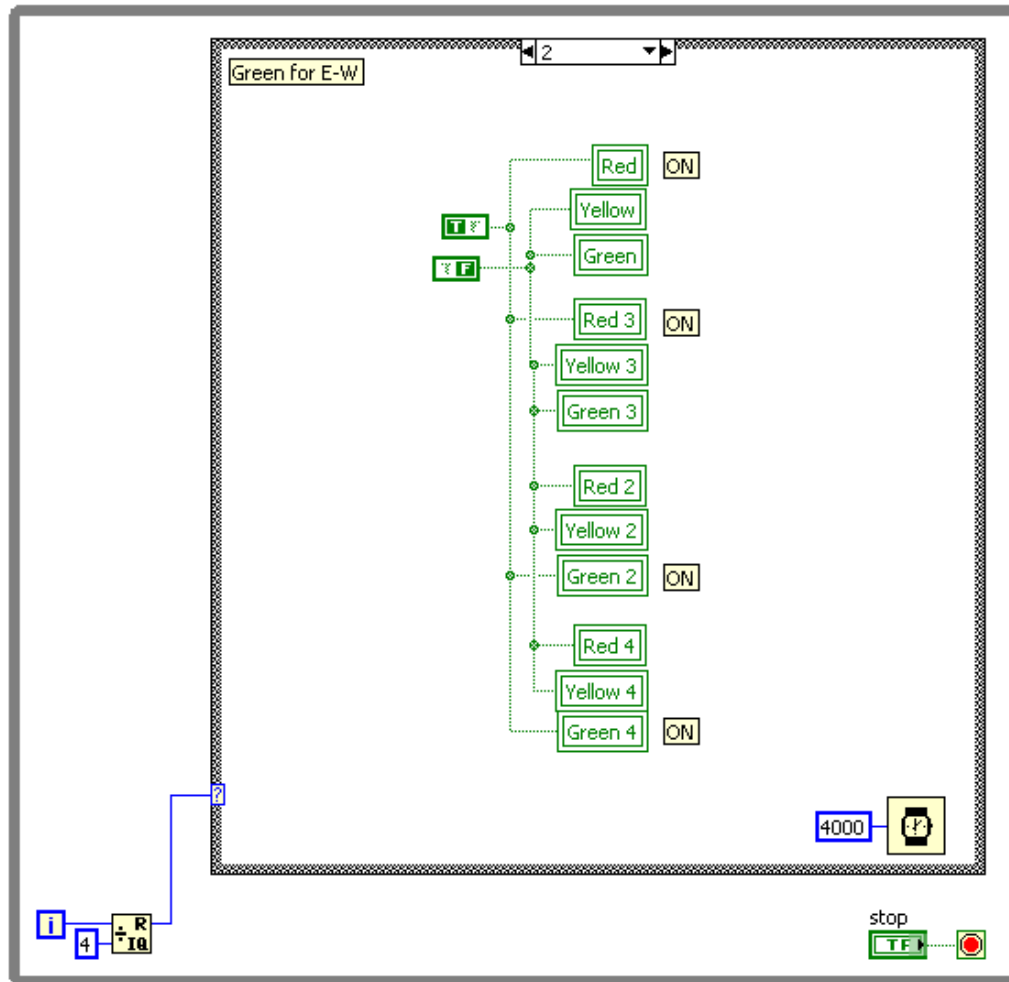


Создание Shared Variables

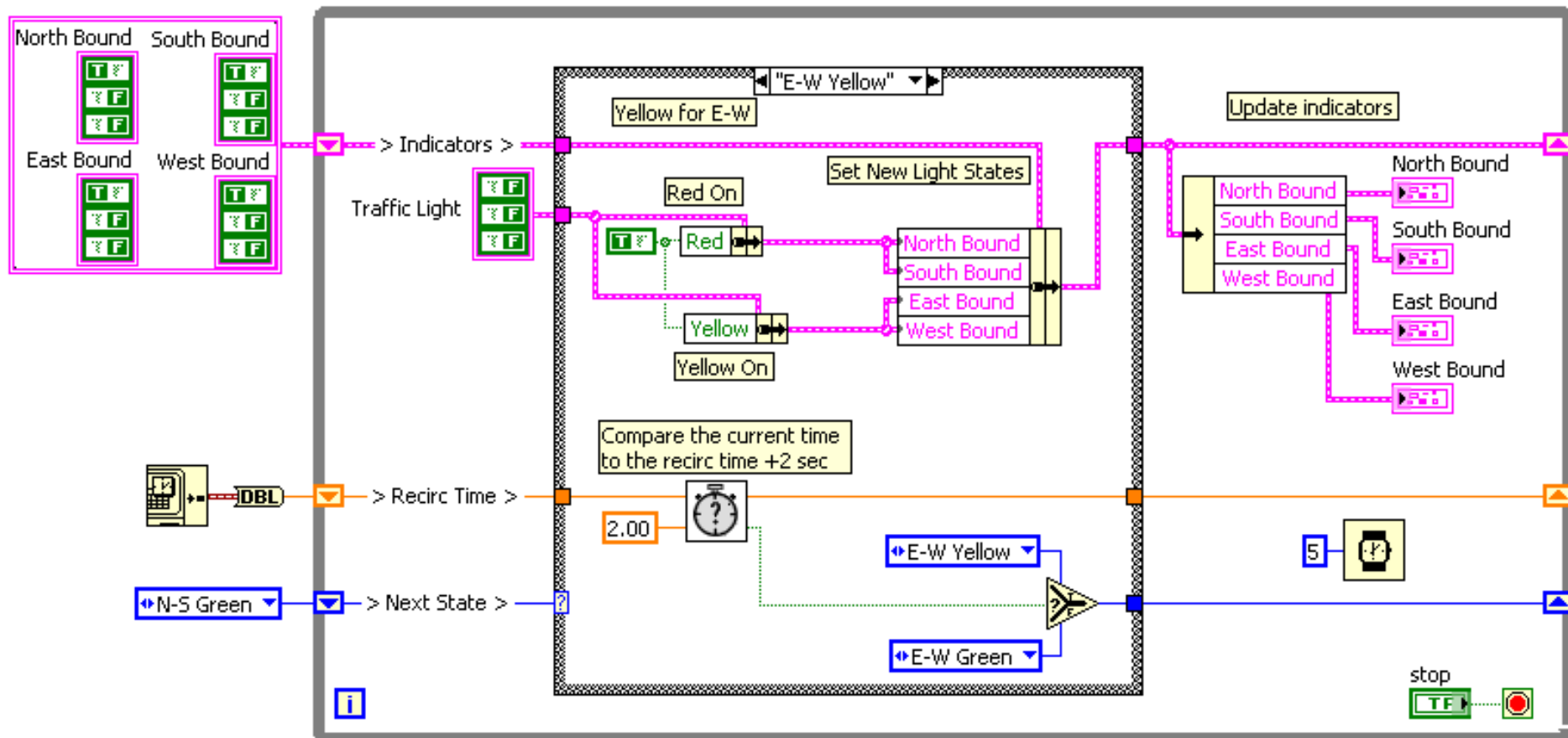
Создайте и используйте переменные типа Single process shared variables.

ДЕМОНСТРАЦИЯ

B. Variables – используйте осторожно

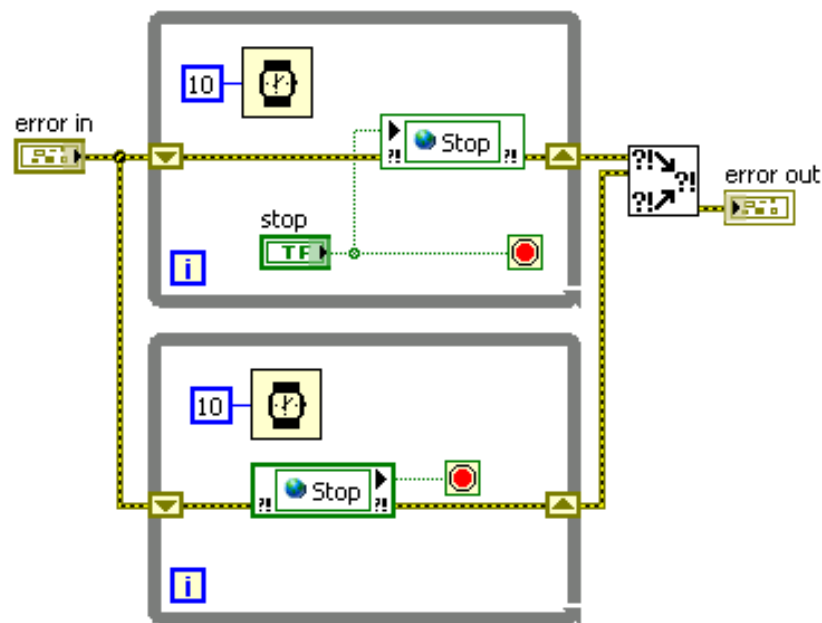


B. Variables – используйте осторожно

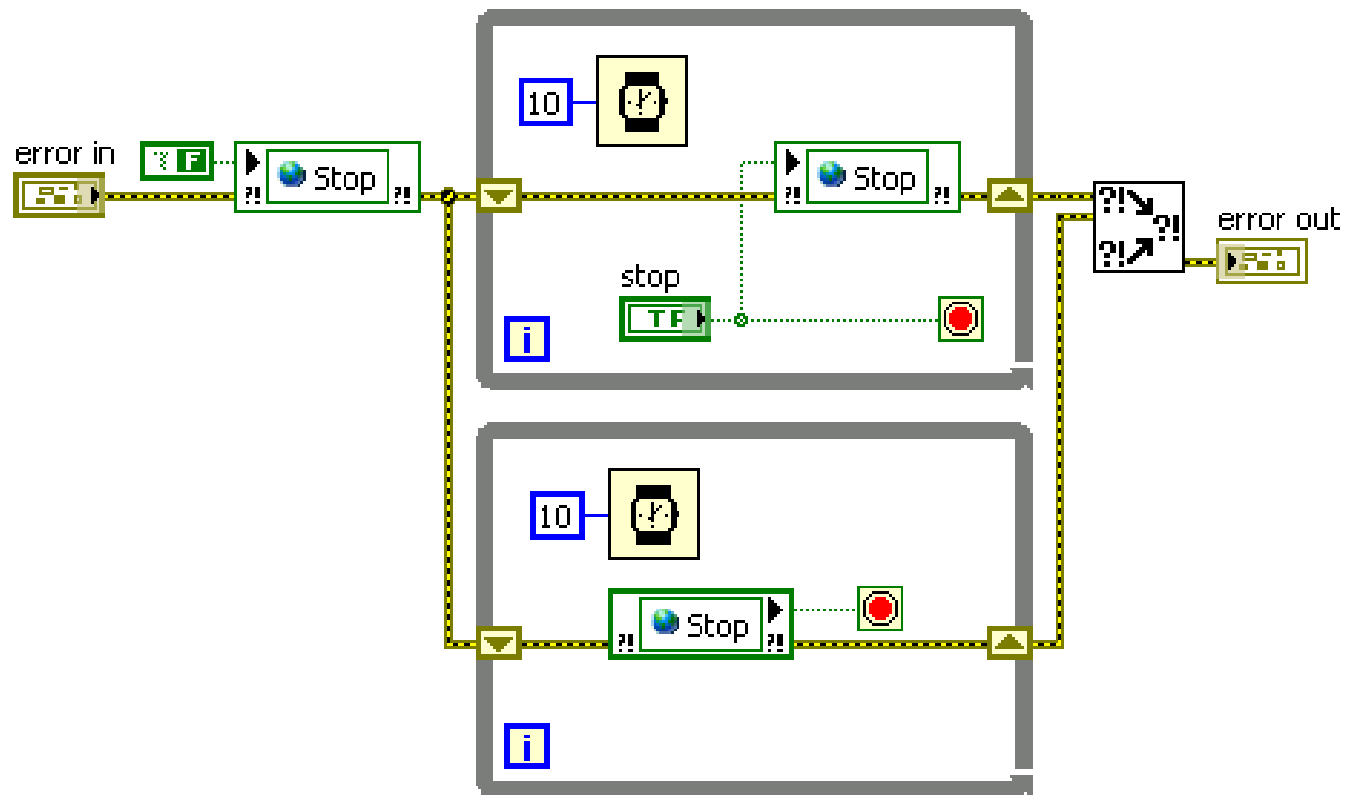


B. Variables – Инициализация

- Прежде, чем запускать VI, проверьте, содержат ли переменные известные данные
- Если не проинициализировать переменные, прежде чем VI прочитает переменные в первый раз, они будут содержать значения по умолчанию объектов лицевой панели, связанных с переменными



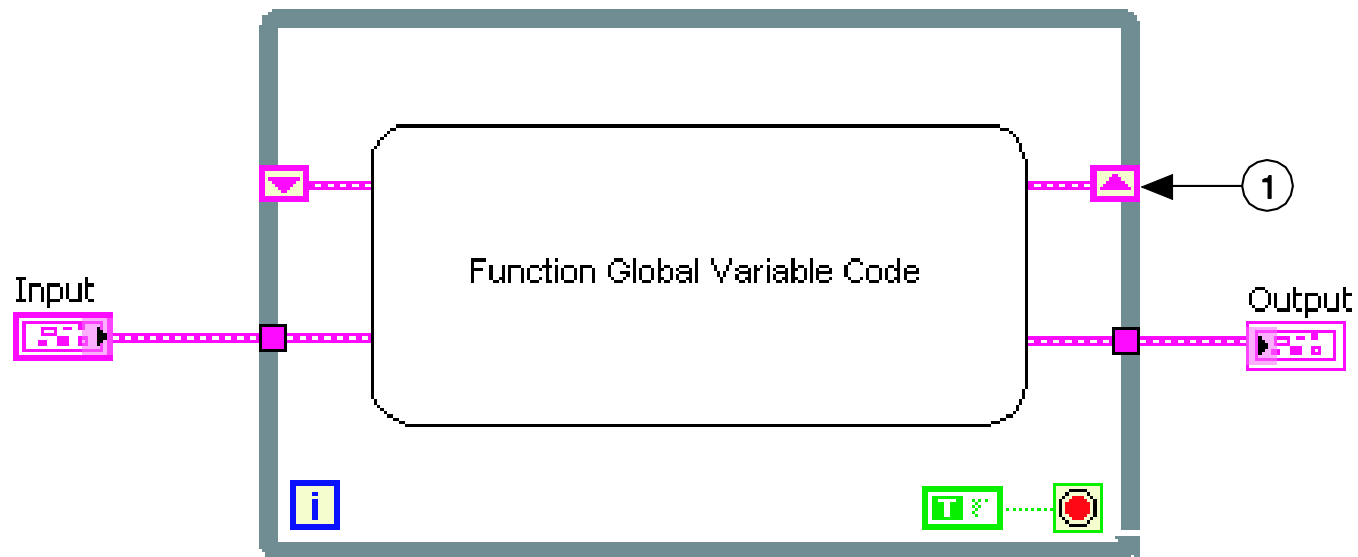
B. Variables – Инициализация



C. Functional Global Variables

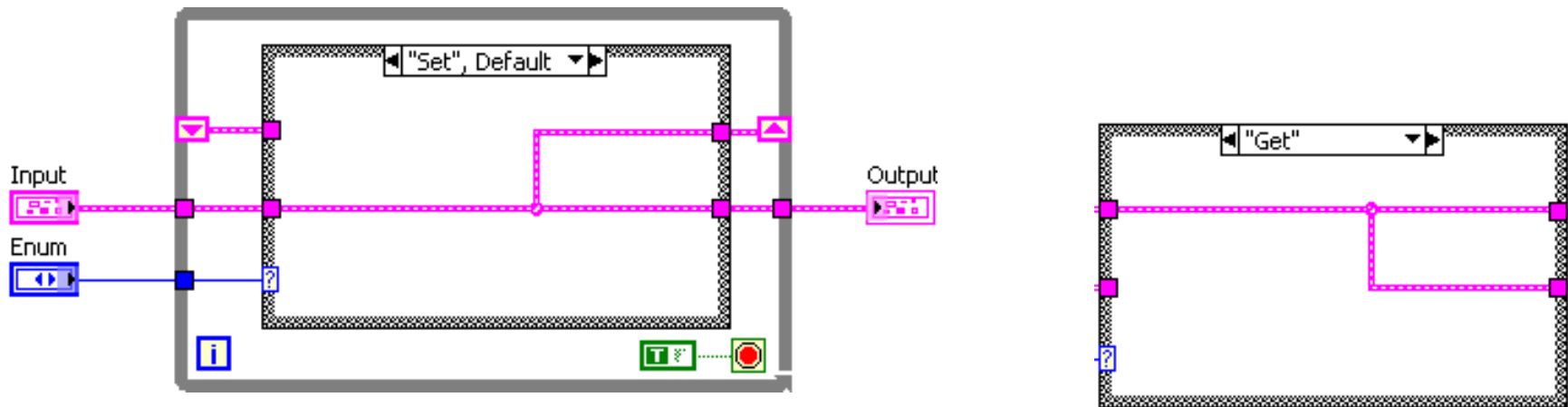
(Функциональные глобальные переменные)

В обобщенном виде функциональная глобальная переменная содержит неинициализированный сдвиговый регистр (1) в цикле For или While, которые выполняются один раз



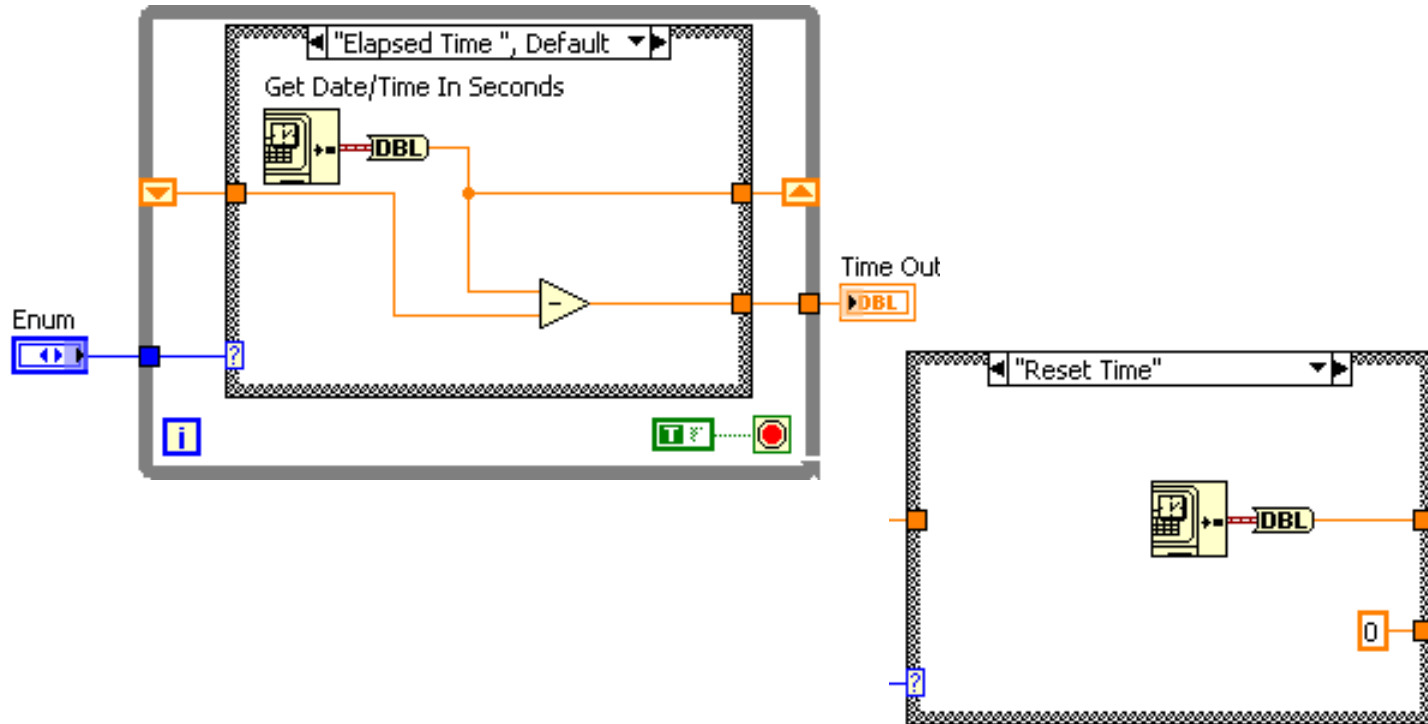
C. Functional Global Variables

- Функциональная глобальная переменная обычно имеет входной параметр **action** операции, которым определяется задача, выполняемая VI
- VI использует неинициализированный сдвиговый регистр в цикле While для хранения результата операции



C. Functional Global Variables – Временные параметры

Очень полезно применять для выполнения измерений по истечении заданного времени



Functional Global Variables

Изучите функциональную глобальную переменную Timer FGV.

Пронаблюдайте, как CallingVI.vi использует функциональную глобальную переменную.

ДЕМОНСТРАЦИЯ

Упражнение 9-2

Проект с глобальными переменными (Global Data Project)

Создайте проект, содержащий несколько VI, которые совместно используют данные, хранимые в Single process shared variable.

GOAL

ЦЕЛЬ

Упражнение 9-2

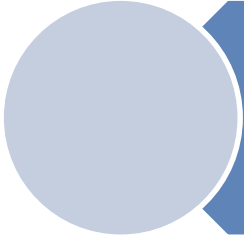
Проект с глобальными переменными (Global Data Project)

- Каково назначение первой переменной Stop в Generate Data VI ?

ДИСКУССИЯ

DISCUSSION

D. Состязания



Состязания - ситуация, когда время возникновения событий или график выполнения задач может непредумышленно воздействовать на значения выходных данных

Состязания являются общей проблемой программ, в которых параллельно выполняются несколько задач, обменивающиеся между собой данными

Состязания

Посмотрите демонстрацию условий появления состязаний.

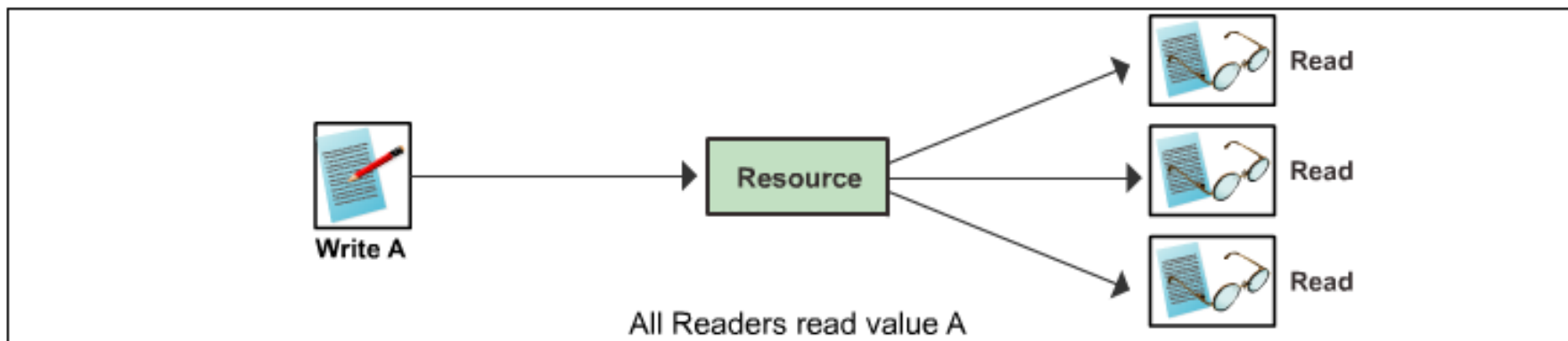
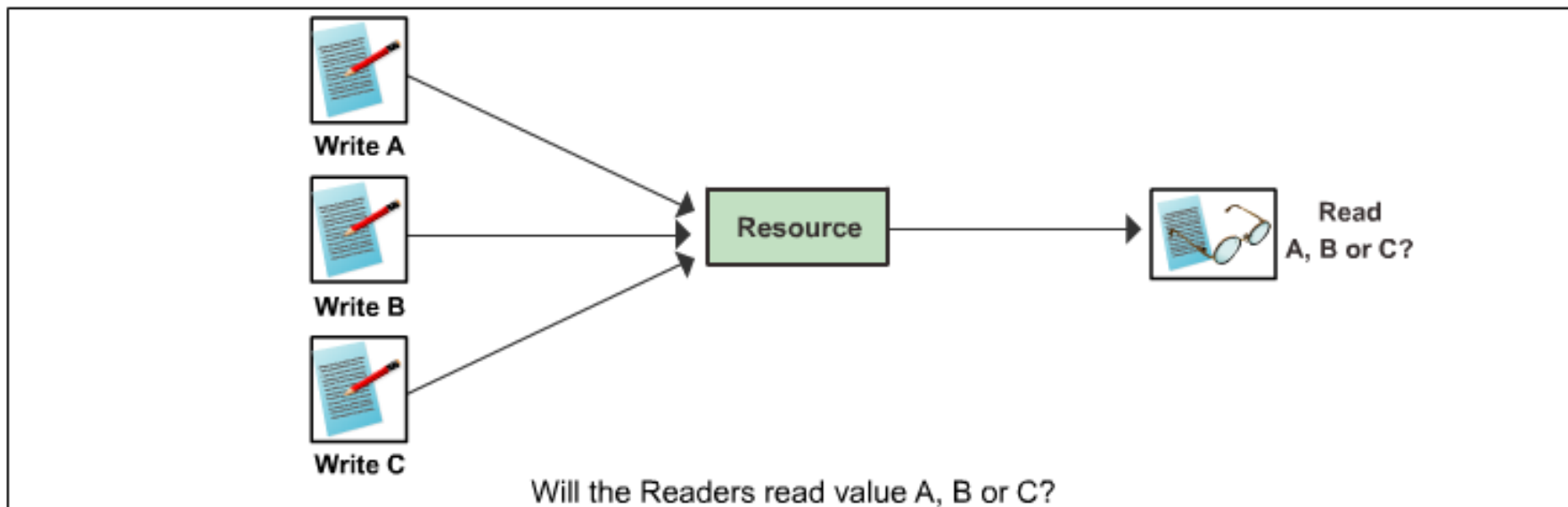
<Exercises>\LabVIEW Core 1\Demonstrations\Race Condition

ДЕМОНСТРАЦИЯ

D. Состязания

- Состязания очень сложно идентифицировать и отладить
- Часто код, в котором реализуются условия появления состязаний, может тысячи раз при тестировании возвращать одни и те же результаты, но существует возможность, что он выдаст результат, отличный от ожидаемого
- Избежать состязаний можно путем:
 - Управления разделяемыми ресурсами
 - Правильно упорядочивая инструкции
 - Идентифицируя и защищая критические участки кода
 - Уменьшая использование переменных

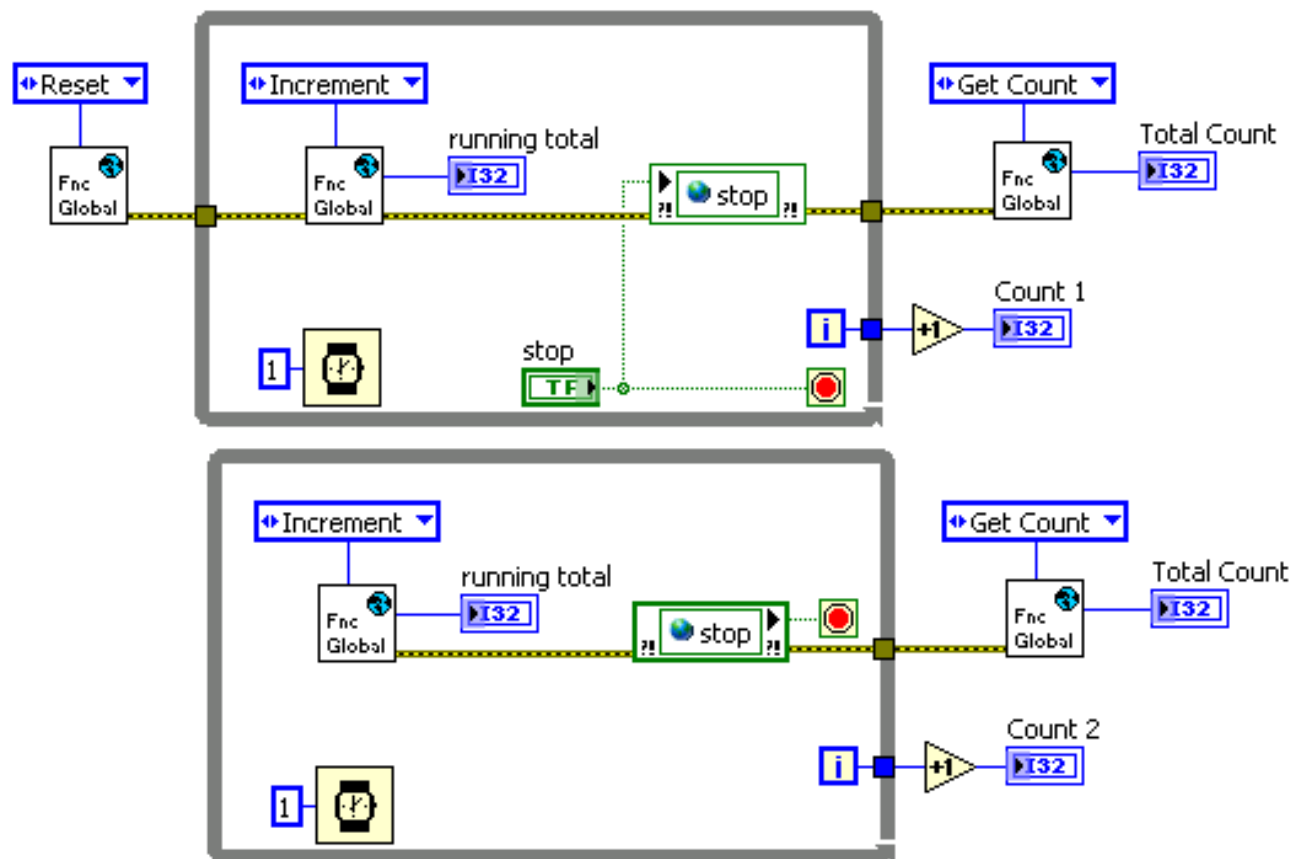
D. Состязания – Разделяемые ресурсы



D. Состязания – Критический код

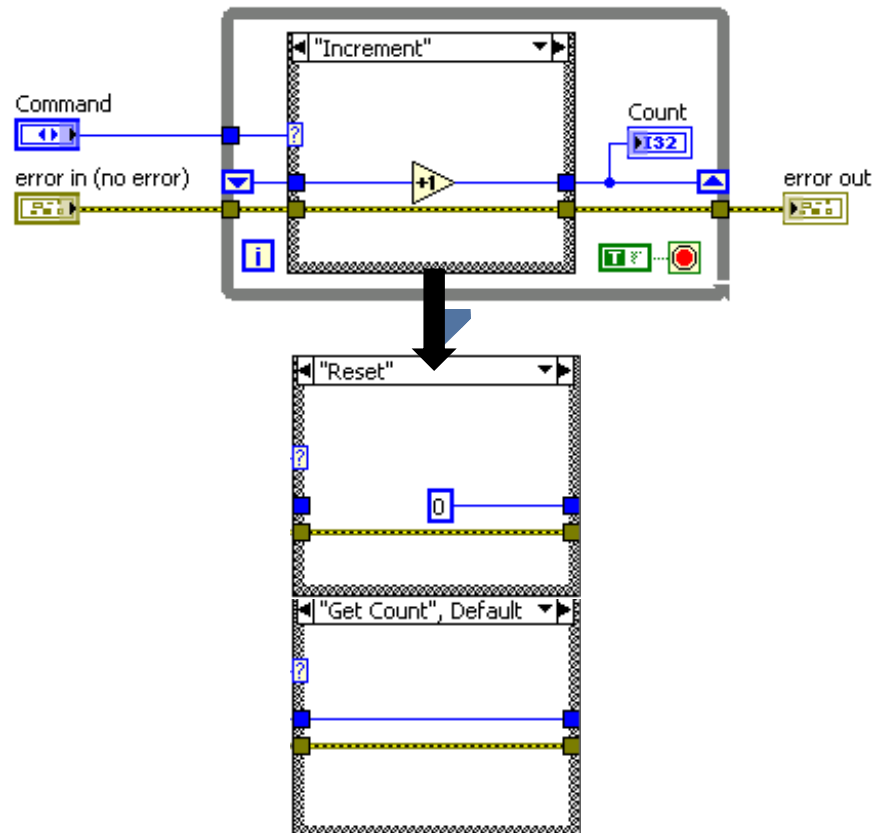
- Критический фрагмент кода – это код, который может вести себя неустойчиво, если некоторый разделяемый ресурс изменяется в процессе работы
- Если один цикл прерывается другим циклом, когда он выполняет критический фрагмент кода, возникают условия, при которых могут возникнуть состязания
- Исключайте условия появления состязаний путем их идентификации и защиты критических фрагментов кода с помощью:
 - Functional Global Variables (функциональных глобальных переменных)
 - Semaphores (семафоров)

D. Состояния – Критический код

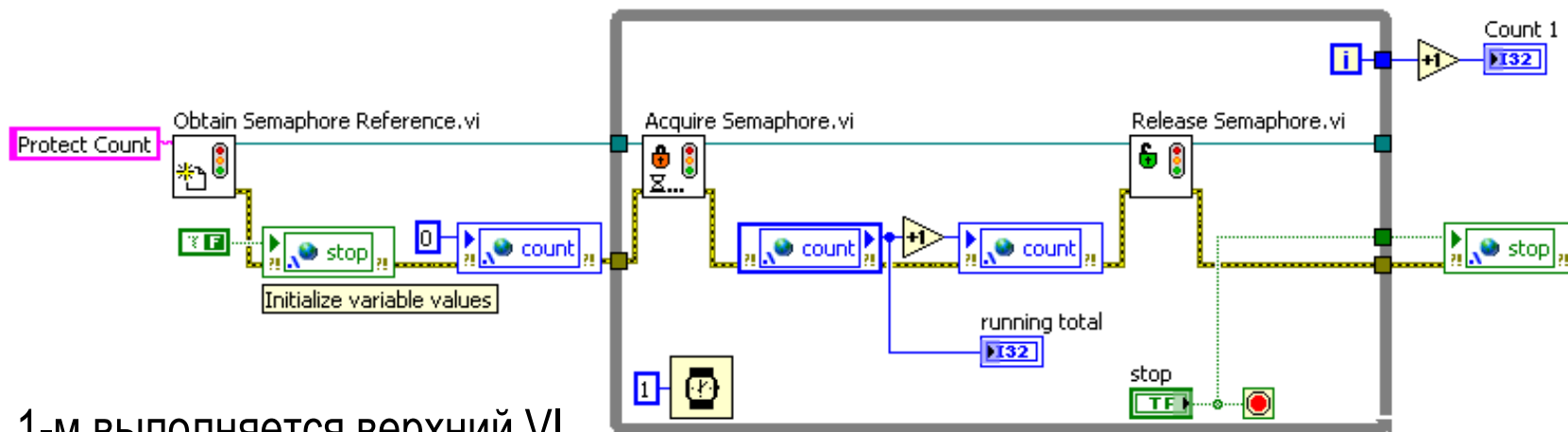


D. Состязания – Критический код

Функциональная глобальная переменная используется для защиты критического кода:

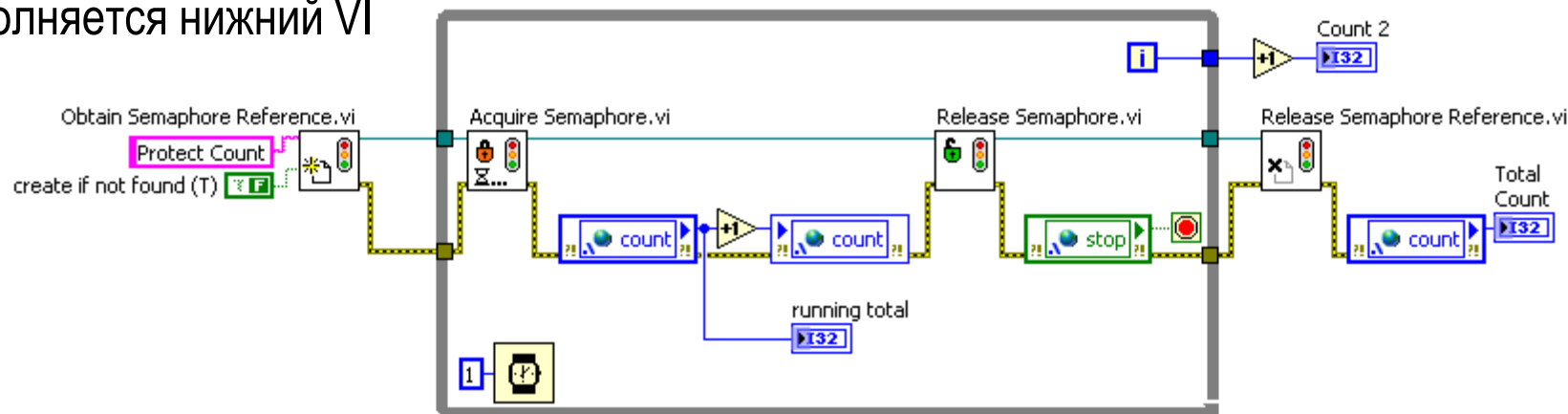


D. Состязания – Критический код



1-м выполняется верхний VI

2-м выполняется нижний VI

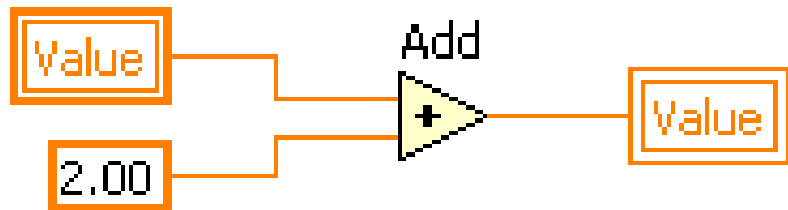
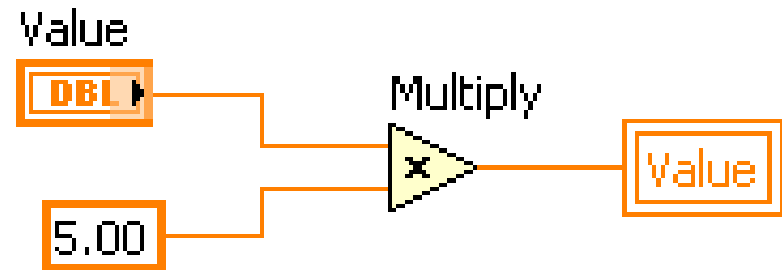


D. Состязания – Упорядочение

Чему равен результат?

Возможны 4 варианта:

- $\text{Value} = (\text{Value} * 5) + 2$
- $\text{Value} = (\text{Value} + 2) * 5$
- $\text{Value} = \text{Value} * 5$
- $\text{Value} = \text{Value} + 2$



Упражнение 9-3

Bank VI

ДОМАШНЕЕ ЗАДАНИЕ

Устраните условия появления состязаний, используя защиту критического фрагмента кода.

GOAL

ЦЕЛЬ

Упражнение 9-3

Bank VI

- Что можно использовать вместо семафора для защиты кода?

ДИСКУССИЯ

DISCUSSION

Заключение – Контрольный вопрос

1. Вы должны использовать переменные, где только возможно в вашем VI.
 - a) True (Да)
 - b) False (Нет)

Заключение – Контрольный вопрос

1. Вы должны использовать переменные где только возможно.
 - a) True (Да)
 - b) False (Нет)

Переменные нужно использовать только, когда это необходимо. Где только возможно, используйте проводники для передачи данных.

Заключение – Контрольный вопрос

2. Какие из объектов не могут передавать данные?
- a) Semaphores
 - b) Functional global variables
 - c) Local variables
 - d) Single process shared variables

Заключение – Контрольный вопрос

2. Какие из объектов не могут передавать данные?
- a) **Semaphores**
 - b) Functional global variables
 - c) Local variables
 - d) Single process shared variables

Заключение – Контрольный вопрос

3. Какие из объектов нужно использовать в проекте?
- a) Local variable
 - b) Global variable
 - c) Functional global variable
 - d) Single-process shared variable

Заключение – Контрольный вопрос

3. Какие из объектов нужно использовать в проекте?
- a) Local variable
 - b) Global variable
 - c) Functional global variable
 - d) **Single-process shared variable**

Заключение – Контрольный вопрос

4. Какие из объектов не могут быть использованы для обмена данными между несколькими VI?
- a) Local variable
 - b) Global variable
 - c) Functional global variable
 - d) Single-process shared variable

Заключение – Контрольный вопрос

4. Какие из объектов не могут быть использованы для обмена данными между несколькими VI?
- a) **Local variable**
 - b) Global variable
 - c) Functional global variable
 - d) Single-process shared variable

Продолжение обучения LabVIEW

- Обучение под руководством преподавателя
 - LabVIEW Core 2: Изучение шаблонов проектирования VI с несколькими циклами, узлы свойств и создание исполняемых приложений
 - Курсы по оборудованию, например, сбора данных и обработки сигналов
 - Курсы «Online», такие, как Machine Vision (машинное зрение) и LabVIEW Real-Time
- Самостоятельно: различные обучающие комплекты и инструментальные средства, спроектированные для изучения в подходящем для вас темпе

Продолжайте свое обучение

- ni.com/support
 - Доступ к руководствам по продукции, базе знаний, примерам кода, учебникам, описаниям приложений и форуму
 - Запрос технической поддержки
- Info-LabVIEW: www.info-labview.org
- Alliance Program: ni.com/alliance
- Publications: ni.com/reference/books/
- Practice! (Практика!)

Курсы

Начинающий
пользователь

LabVIEW Core 1

LabVIEW Core 2

Приобретаемые навыки:

- Ориентация в среде LabVIEW
- Создание базовых приложений в LabVIEW

Опытный пользователь

LabVIEW Core 3

Приобретаемые навыки:

- Модульная разработка приложений
- Методы структурного проектирования и опыт разработки
- Управление памятью и оптимизация производительности VI

Квалифицированный
пользователь

Managing Software Engineering
in LabVIEW

LabVIEW OOP System Design

Advanced Architectures in
LabVIEW

Приобретаемые навыки:

- Разработка приложений высокого качества
- Максимизация повторного использования кода
- Объектно-ориентированное программирование в LabVIEW
- Управление процессом проектирования в LabVIEW
- Продвинутое развитие шаблонов, используемых в архитектуре LabVIEW

Аттестация

Certified LV Associate
Developer Exam

Проверяемые навыки:

- Знание среды LabVIEW

Certified LabVIEW
Developer Exam

Проверяемые навыки:

- Эрудиция в области разработки приложений LabVIEW

Certified LabVIEW
Architect Exam

Проверяемые навыки:

- Искусство разработки приложений в LabVIEW



Пожалуйста, заполните [анкету курсов](#).

Спасибо!