МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
им. М.В. ЛОМОНОСОВА

Физический факультет
Кафедра физики колебаний

# Цифровая обработка сигналов с помощью платы АЦП и программного комплекса LabView

Задача создана
студентом V курса кафедры
физики колебаний
С.Д. Мещеряковым

Научный руководитель
проф. В.П. Митрофанов

Москва-2004

# Содержание

# I. Платы АЦП

## 1.1. Введение

Почти все сигналы, возникающие в макроскопических физических процессах, являются одновременно и непрерывными во времени и имеющими непрерывное множество значений. Такие непрерывные сигналы, изменяющиеся по мере того, как происходят изменения в (непрерывных) физических процессах, называют аналоговыми сигналами.

Существуют также дискретные во времени сигналы. Значение такого сигнала известно только в определенные дискретные моменты времени и им можно воспользоваться только в эти моменты времени. Дискретный во времени сигнал можно рассматривать как результат взятия выборок непрерывного во времени сигнала.

Точно так же и величина сигнала может принимать только некоторые дискретные значения. Тогда сигнал называют дискретным по величине сигналом. Такой сигнал может принимать только конечное число значений между заданным верхним и нижним пределами. Сигналы, дискретные по величине и во времени, мы будем называть цифровыми сигналами.

Часто требуется преобразовать аналоговый сигнал в число, пропорциональное амплитуде сигнала и наоборот. Это играет важную роль в тех случаях, когда компьютер или процессор регистрируют или контролируют ход эксперимента или технологического процесса. Аналого-цифровое преобразование используется в областях, где для обеспечения помехоустойчивой и шумо-защищенной передачи аналоговая информация преобразуется в промежуточную цифровую, а также в самых разнообразных измерительных средствах (включая обычные настольные приборы типа цифровых универсальных измерительных приборов и более экзотические приборы, такие, как усреднители переходных процессов, «ловушки для выбросов» и осциллографы с цифровой памятью), в устройствах генерации и обработки сигналов, таких, как цифровые синтезаторы колебаний и устройства шифрования данных.

Процесс преобразования сигнала с непрерывным множеством значений в сигнал с дискретными значениями называется квантованием и реализуется с помощью аналого-цифрового преобразователя. Тогда непрерывность во времени сохраняется даже для сигналов с дискретными значениями. Однако большинство аналого-цифровых преобразователей действуют не мгновенно, поскольку процедура преобразования требует некоторого времени. Следующее преобразование в последовательности преобразований возможно только тогда, когда выполнено предыдущее. В таких преобразователях должно производиться взятие выборки сигнала, поэтому свойство непрерывности во времени теряется.

## 1.2. Виды АЦП

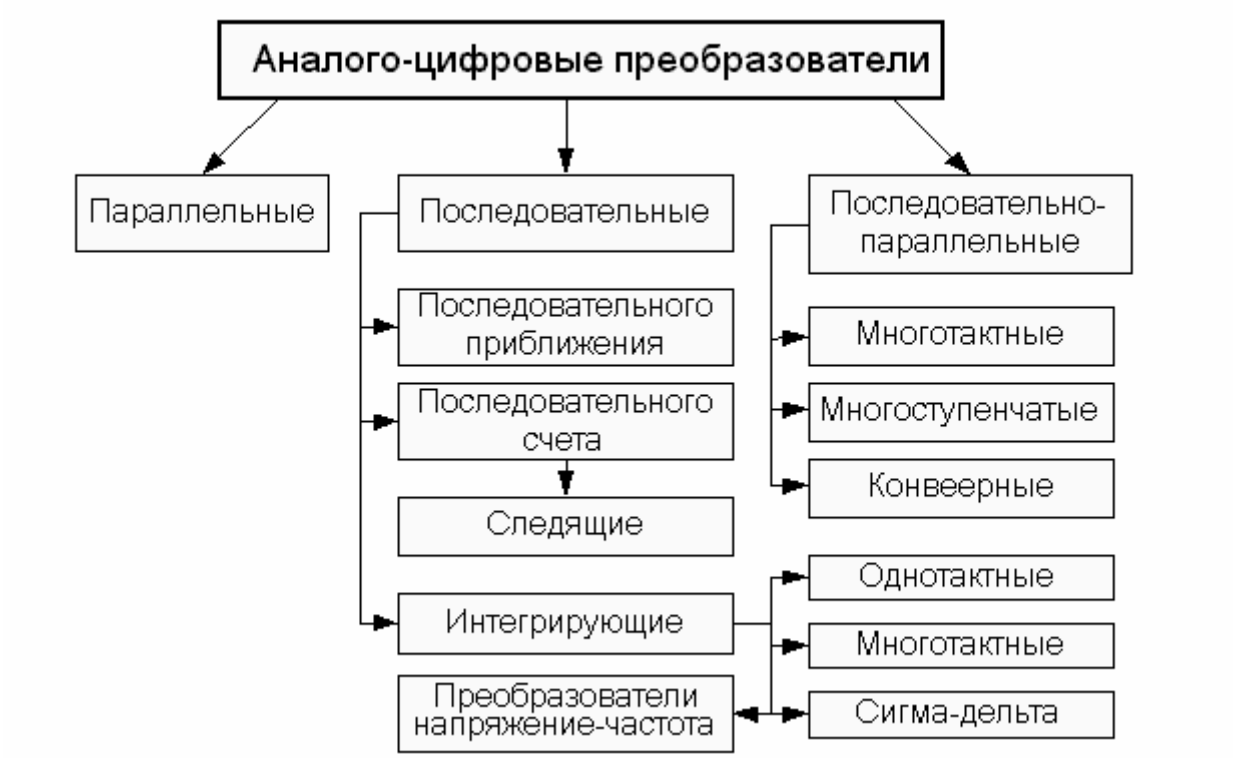



Рис.1 Классификация АЦП

В настоящее время известно большое число методов преобразования напряжение-код. Эти методы существенно отличаются друг от друга потенциальной точностью, скоростью преобразования и сложностью аппаратной реализации. На Рис. 1 представлена классификация АЦП по методам преобразования.

В основу классификации АЦП положен признак, указывающий на то, как во времени разворачивается процесс преобразования аналоговой величины в цифровую. В основе преобразования выборочных значений сигнала в цифровые эквиваленты лежат операции квантования и кодирования. Они могут осуществляться с помощью либо последовательной, либо параллельной, либо последовательно-параллельной процедур приближения цифрового эквивалента к преобразуемой величине.

### 1.2.1. Параллельные АЦП

АЦП этого типа осуществляют квантование сигнала одновременно с помощью набора компараторов, включенных параллельно источнику входного сигнала. На Рис. 2 показана реализация параллельного метода АЦ-преобразования для 3-разрядного числа.

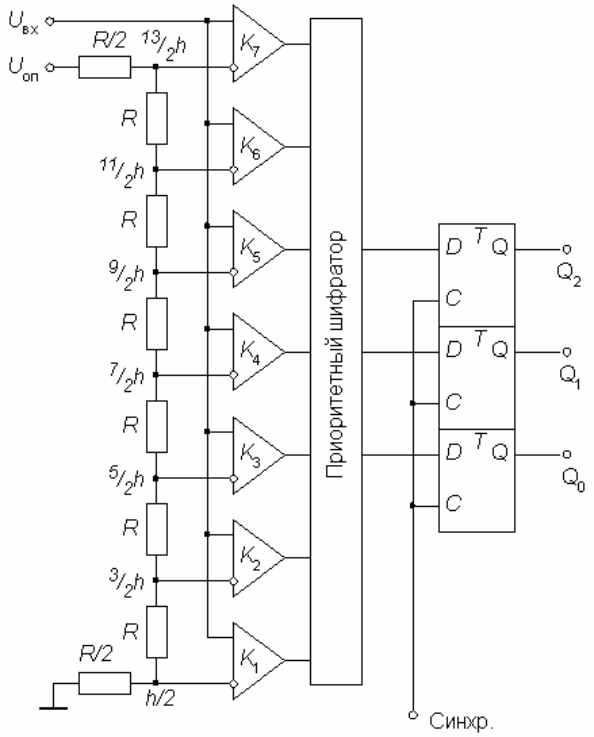



Рис.2 Параллельный метод АЦ-преобразования

С помощью трех двоичных разрядов можно представить восемь различных чисел, включая нуль. Необходимо, следовательно, семь компараторов. Семь соответствующих эквидистантных опорных напряжений образуются с помощью резистивного делителя.

Благодаря одновременной работе компараторов параллельный АЦП является самым быстрым. Например, восьмиразрядный преобразователь типа MAX104 позволяет получить 1 млрд отсчетов в секунду при времени задержки прохождения сигнала не более 1,2 нс. Недостатком этой схемы является высокая сложность. Действительно, N-разрядный параллельный АЦП сдержит $2^{N-1}$ компараторов и 2N согласованных резисторов. Следствием этого является высокая стоимость (сотни долларов США) и значительная потребляемая мощность. Тот же MAX104, например, потребляет около 4 Вт.

### 1.2.2. АЦП последовательного счета

Этот преобразователь является типичным примером последовательных АЦП с единичными приближениями и состоит из компаратора, счетчика и ЦАП (Рис. 3). На один вход компаратора поступает входной сигнал, а на другой - сигнал обратной связи с ЦАП.

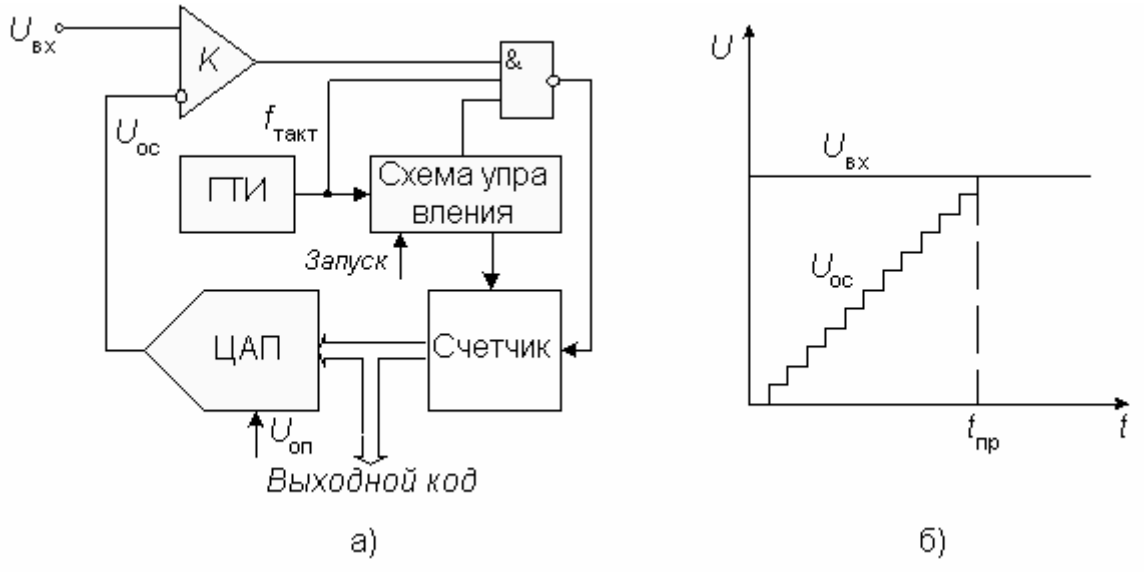



Рис.3 АЦП последовательного счета

Работа преобразователя начинается с прихода импульса запуска, который включает счетчик, суммирующий число импульсов, поступающих от генератора тактовых импульсов ГТИ. Выходной код счетчика подается на ЦАП, осуществляющий его преобразование в напряжение обратной связи $U_{ос}$. Процесс преобразования продолжается до тех пор, пока напряжение обратной связи сравняется со входным напряжением и переключится компаратор, который своим выходным сигналом прекратит поступление тактовых импульсов на счетчик. Переход выхода компаратора из 1 в 0 означает завершение процесса преобразования.

### 1.2.3. АЦП последовательного приближения

Преобразователь этого типа, называемый в литературе также АЦП с *поразрядным уравновешиванием*, является наиболее распространенным вариантом последовательных АЦП.

В основе работы этого класса преобразователей лежит принцип *дихотомии*, т.е последовательного сравнения измеряемой величины с $^1/_2$, $^1/_4$, $^1/_8$ и т.д. от возможного максимального значения ее. Это позволяет для N-разрядного АЦП последовательного приближения выполнить весь процесс преобразования за N последовательных шагов (итераций) вместо $2^N$-1 при использовании последовательного счета и получить существенный выигрыш в быстродействии. Так, уже при N=10 этот выигрыш достигает 100 раз и позволяет получить с помощью таких АЦП до $10^5$...$10^6$ преобразований в секунду. В то же время статическая погрешность этого типа преобразователей, определяемая в основном используемым в нем ЦАП, может быть очень малой, что позволяет реализовать разрешающую способность до 18 двоичных разрядов при частоте выборок до 200 кГц.

## 1.2.4. Последовательно-параллельные АЦП

Последовательно-параллельные АЦП являются компромиссом между стремлением получить высокое быстродействие и желанием сделать это по возможности меньшей ценой. Последовательно-параллельные АЦП занимают промежуточное положение по разрешающей способности и быстродействию между параллельными АЦП и АЦП последовательного приближения. Последовательно-параллельные АЦП подразделяют на многоступенчатые, многотактные и конвеерные.

## 1.2.5. Многоступенчатые АЦП

В многоступенчатом АЦП процесс преобразования входного сигнала разделен в пространстве. В качестве примера на Рис. 4 представлена схема двухступенчатого 8-разрядного АЦП.

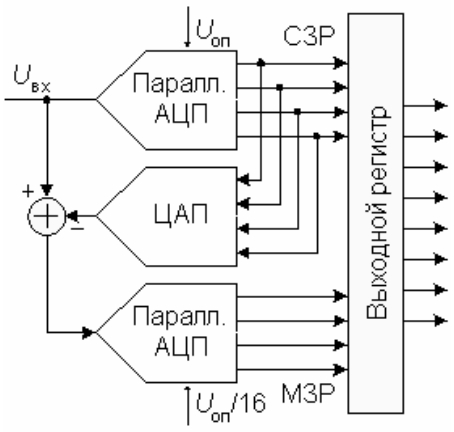



Рис.4 Двухступенчатый 8-разрядный АЦП

Верхний по схеме АЦП осуществляет грубое преобразование сигнала в четыре старших разряда выходного кода. Цифровые сигналы с выхода АЦП поступают на выходной регистр и одновременно на вход 4-разрядного быстродействующего ЦАП. Во многих ИМС многоступенчатых АЦП (AD9042, AD9070 и др.) этот ЦАП выполнен по схеме суммирования токов на дифференциальных переключателях, но некоторые (AD775, AD9040A и др.) содержат ЦАП с суммированием напряжений. Остаток от вычитания выходного напряжения ЦАП из входного напряжения схемы поступает на вход АЦП2, опорное напряжение которого в 16 раз меньше, чем у АЦП1. Как следствие, квант АЦП2 в 16 раз меньше кванта АЦП1. Этот остаток, преобразованный АЦП2 в цифровую форму представляет собой четыре младших разряда выходного кода. Различие между АЦП1 и АЦП2 заключается прежде всего в требовании к точности: у АЦП1 точность должна быть такой же как у 8-разрядного преобразователя, в то время как АЦП2 может иметь точность 4-разрядного.

## 1.3. Параметры АЦП

При последовательном возрастании значений входного аналогового сигнала $U_{вх}(t)$ от 0 до величины, соответствующей полной шкале АЦП $U_{пш}$ выходной цифровой сигнал D(t) образует ступенчатую кусочно-постоянную линию. Такую зависимость по аналогии с ЦАП называют обычно характеристикой преобразования АЦП. В отсутствие аппаратных погрешностей средние точки ступенек расположены на *идеальной прямой* 1 (Рис. 5), которой соответствует идеальная характеристика преобразования. Реальная характеристика преобразования может существенно отличаться от идеальной размерами и формой ступенек, а также расположением на плоскости координат. Для количественного описания этих различий существует целый ряд параметров.

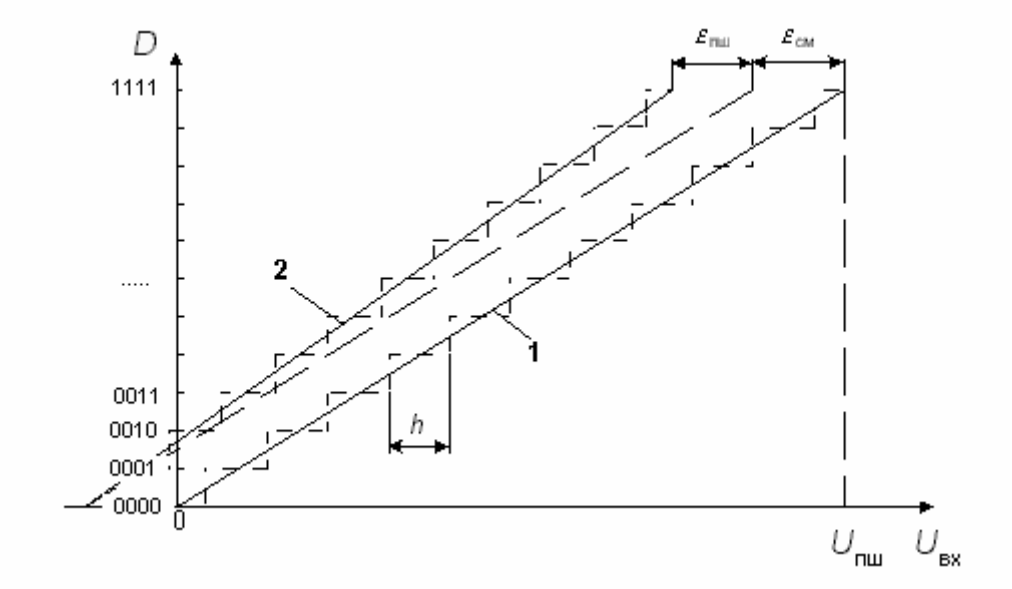



Рис.5 Статистическая характеристика преобразователя АЦП

### 1.3.1. Статические параметры

*Разрешающая способность* - величина, обратная максимальному числу кодовых комбинаций на выходе АЦП. Разрешающая способность выражается в процентах, разрядах или децибелах и характеризует потенциальные возможности АЦП с точки зрения достижимой точности. Например, 12-разрядный АЦП имеет разрешающую способность 1/4096, или 0,0245% от полной шкалы, или -72,2 дБ.

Разрешающей способности соответствует приращение входного напряжения АЦП $U_{вх}$ при изменении $D_j$ на единицу младшего разряда (EMP). Это приращение является шагом квантования. Для двоичных кодов преобразования номинальное значение шага квантования $h=U_{пш}/(2^N-1)$, где $U_{пш}$ - номинальное максимальное входное напряжение АЦП (напряжение полной шкалы), соответствующее максимальному значению выходного кода, N - разрядность АЦП. Чем больше разрядность преобразователя, тем выше его разрешающая способность.

Погрешность полной шкалы - относительная разность между реальным и идеальным значениями предела шкалы преобразования при отсутствии смещения нуля.

$$\delta_{пш} = \frac{\varepsilon_{пш}}{U_{пш}} \cdot 100\%$$

Эта погрешность является мультипликативной составляющей полной погрешности. Иногда указывается соответствующим числом ЕМР.

*Погрешность смещения нуля* - значение Uвх, когда входной код ЦАП равен нулю. Является аддитивной составляющей полной погрешности. Обычно определяется по формуле

$$\varepsilon_{см} = U_{вх.01} - h/2$$

где $U_{вх.01}$ - значение входного напряжения, при котором происходит переход выходного кода из О в 1. Часто указывается в милливольтах или в процентах от полной шкалы:

$$\delta_{см} = \frac{\varepsilon_{см}}{U_{пш}} \cdot 100\%$$

Погрешности полной шкалы и смещения нуля АЦП могут быть уменьшены либо подстройкой аналоговой части схемы, либо коррекцией вычислительного алгоритма цифровой части устройства.

*Погрешности линейности* характеристики преобразования не могут быть устранены такими простыми средствами, поэтому они являются важнейшими метрологическими характеристиками АЦП.

*Нелинейность* - максимальное отклонение реальной характеристики преобразования $D(U_{вх})$ от *оптимальной* (линия 2 на Рис. 5). Оптимальная характеристика находится эмпирически так, чтобы минимизировать значение погрешности нелинейности. Нелинейность обычно определяется в относительных единицах, но в справочных данных приводится также и в ЕМР. Для характеристики, приведенной на Рис. 5

$$\delta_{л} = \frac{\varepsilon_{j}}{U_{пш}} \cdot 100\%$$

*Дифференциальной нелинейностью* АЦП в данной точке k характеристики преобразования называется разность между значением кванта преобразования $h_k$ и средним значением кванта преобразования h. В спецификациях на конкретные АЦП значения дифференциальной нелинейности выражаются в долях ЕМР или процентах от полной шкалы. Для характеристики, приведенной на Рис. 6,

$$\delta_{дл} = \frac{h_k - h}{U_{пш}} \cdot 100\%$$

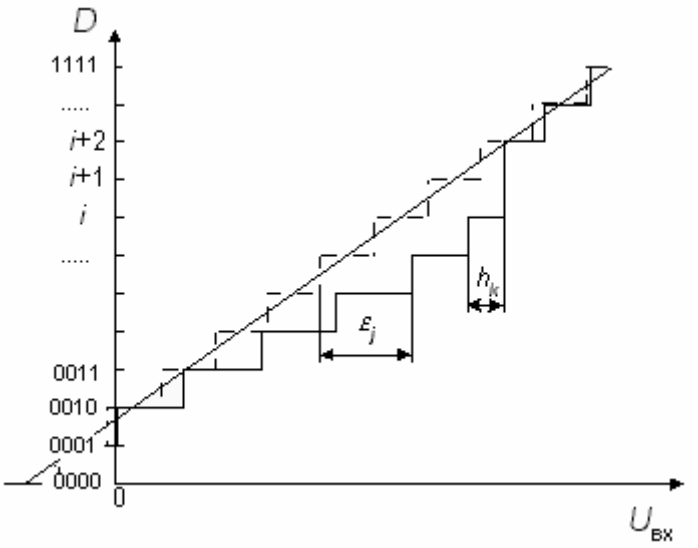



Рис.6 Погрешности линейности характеристики преобразования АЦП

*Погрешность дифференциальной линейности* определяет два важных свойства АЦП: непропадание кодов и монотонность характеристики преобразования. Непропадание кодов - свойство АЦП выдавать все возможные выходные коды при изменении входного напряжения от начальной до конечной точки диапазона преобразования. Пример пропадания кода i+1 приведен на Рис. 6. При нормировании непропадания кодов указывается эквивалентная разрядность АЦП - максимальное количество разрядов АЦП, для которых не пропадают соответствующие им кодовые комбинации.

*Монотонность характеристики преобразования* - это неизменность знака приращения выходного кода D при монотонном изменении входного преобразуемого сигнала. Монотонность не гарантирует малых значений дифференциальной нелинейности и непропадания кодов.

*Температурная нестабильность* АЦ-преобразователя характеризуется *температурными коэффициентами погрешности* полной шкалы и погрешности смещения нуля.

### 1.3.2 Динамические параметры

Возникновение динамических погрешностей связано с дискретизацией сигналов, изменяющихся во времени. Можно выделить следующие параметры АЦП, определяющие его динамическую точность.

*Максимальная частота дискретизации (преобразования)* - это наибольшая частота, с которой происходит образование выборочных значений сигнала, при которой выбранный параметр АЦП не выходит за заданные пределы. Измеряется числом выборок в секунду. Выбранным параметром может быть, например, монотонность характеристики преобразования или погрешность линейности.

*Время преобразования ($t_{пр}$)* - это время, отсчитываемое от начала импульса дискретизации или начала преобразования до появления на выходе устойчивого кода, соответствующего данной выборке. Для одних АЦП, например, последовательного счета или многотактного интегрирования, эта величина является переменной, зависящей от значения входного сигнала, для других, таких как параллельные или последовательно-параллельные АЦП, а также АЦП последовательного приближения, примерно постоянной. При работе АЦП без УВХ (устройство выборки-хранения) время преобразования является апертурным временем.

*Время выборки (стробирования)* - время, в течение которого происходит образование одного выборочного значения. При работе без УВХ равно времени преобразования АЦП.

## II. Программный комплекс LabView

### 2.1. Введение

В последнее время произошла революция в создании и разработке измерительных средств. Это в первую очередь связано с активным развитием компьютерных технологий применительно к технологиям измерений. Основными достижениями революции в измерительных технологиях стали:

Так называемые DAQ – boards ( Data Acquisition Boards – Платы сбор данных) – измерительные модули, встраиваемые непосредственно в компьютер.

Специализированные измерительные интегрированные программные оболочки для сбора, обработки и визуального представления измерительной информации (например – LabView).

Под Виртуальными Измерительными Системами понимается средство измерений, построенные на базе персональных компьютеров (ПК), встраиваемых в компьютер многофункциональных и многоканальных АЦП – плат, внешних программно-управляемых модулей предварительной обработки сигналов и приборов и специализированных измерительных интегрированных программных оболочек для сбора, обработки и визуального представления измерительной информации.

В отличие от традиционных средств, их функции, пользовательский интерфейс, алгоритмы сбора и обработки информации определяются пользователем, а не производителем.

Уже более 10 лет назад фирма National Instruments представила ученым и инженерам графическую среду программирования LabView как средство разработки для быстрого проектирования и модификации инструментальных систем. Цель LabView всегда состояла в упрощении задачи программирования, чтобы ученые и инженеры могли использовать все возможности ПК и в то же самое время выполнять свою работу быстро и легко. В последнее время LabView стала ведущей промышленной средой разработки для систем сбора данных, контроля и измерения, и для исследовательских приложений. LabView несет мощь новейших технологий программного обеспечения вместе с простотой графической среды разработки.

LabView — интегрированная среда разработчика для создания интерактивных программ сбора, обработки данных и управления периферийными устройствами. Программирование осуществляется на уровне функциональных блок-диаграмм. Сочетание графического языка программирования и современного компилятора позволяет значительно сократить время разработки сложных систем при сохранении высокой скорости выполнения программ. Библиотеки современных алгоритмов обработки и анализа данных превращают LabView в универсальный инструмент создания интегрированных систем на базе PC.

LabVIEW, подобно C или БЕЙСИКУ, является универсальной системой программирования с мощными библиотеками функций для различных задач программирования. LabView включает в себя библиотеки инструментов для:

- сбора данных,
- анализа данных,
- представления данных,
- хранения обработанных данных

LabView также включает стандартные средства автоматического проектирования приложений, такие, что Вы можете устанавливать контрольные точки, представлять в виде стендовой модели выполнение Вашей программы, так, чтобы видеть, как данные проходят через программу шаг за шагом, чтобы упростить понимание происходящих процессов.

LabView - универсальная система программирования, но также включает библиотеки функций и средств проектирования, разработанных определенно для сбора данных и инструментов управления и обработки данных. Программы LabView названы виртуальными приборами (VIs), потому что их действия и внешний вид может имитировать реальные приборы. В тоже время, VIs подобны функциям стандартных языков программирования. Однако, VIs имеют ряд преимуществ перед функциям стандартных языков программирования:

VIs более наглядны, просты для конструирования измерительных модулей и взаимодействия с оператором,

Внутренняя структура VIs является для пользователя "чёрным ящиком" с известными входами и выходами, что упрощает применение VIs и обеспечивает автоматическую совместимость различных VIs.

## 2.2. Преимущества LabView

LabView — графическая система программирования на уровне функциональных блок-диаграмм, позволяющая графически объединять программные модули в виртуальные приборы (Virtual Instruments — VI).Таким образом, LabView дает возможность избежать сложностей обычного "текстового" программирования.

***•Разработка законченной системы***

Как правило программный пакет покрывает только один аспект поставленной задачи, но не решает все проблемы — сбор данных, их анализ, представление и управление. LabView предоставляет вам все необходимые средства, объединенные единой методологией, поэтому вам вряд ли понадобится покидать среду LabVIEW .Вы имеете доступ к библиотекам виртуальных инструментов (VI) для управления и получения данных через встраиваемые платы сбора данных. LabView предлагает более 600 драйверов для приборов от более чем 50 мировых производителей таким образом исключается необходимость низкоуровнего программирования приборов. После сбора данных вы можете использовать библиотеку виртуальных инструментов (VI) анализа для получения из потока данных необходимого результата. Вы можете воспользоваться цифровой обработкой сигналов (DSP), цифровой фильтрацией, статистикой и численным анализом. Наконец, вы можете управлять системой с помощью вашей программы и визуализировать результаты, используя интерактивные лицевые панели. С помощью этих панелей создается стандартный, легко узнаваемый интерфейс независимо от аппаратного обеспечения системы. Кроме того, вы имеете широкие возможности по манипулированию данными — запись/чтение с диска, передача по сети и печать на принтере.

***•Построение собственного виртуального инструмента***

В LabView вместо написания программы вы строите виртуальные инструменты (VI). Легко создаваемая лицевая панель пользовательского

интерфейса дает вам возможность интерактивного управления вашей программной системой. Для описания функционирования системы вы строите блок-диаграмму привычный элемент для любой технической разработки. Но в LabView блок-диаграмма является кроме всего исходным кодом вашей программы. Таким образом, решается требующая немало времени и усилий при обычном подходе задача трансформация идеи разработчика в код программы. LabView — идеальное средство для построения вашей программной системы. Виртуальные инструменты, с их графическим представлением, очень легко модифицируются, отлаживаются и полностью самодокументированны. Не менее важно, что созданные блоки вы можете встраивать как пиктограммы в диаграммы верхнего уровня для построения сложных программных комплексов. Для построения виртуального инструмента, в первую очередь, вы создаете лицевую панель с необходимым набором кнопок, переключателей, регуляторов, экранов и т. п. Лицевая панель работает как интерактивный интерфейс ввода и вывода для вашей измерительной системы или системы управления. В LabView конструирование лицевой панели сводится к рисованию картинки. Для начала вам предоставляются различные индикаторы и управляющие элементы. Остается только выбрать их из меню и расставить на панели. Кроме того, вы можете изменить цвет, размер, метку каждого элемента, его тип данных и диапазон значений. Есть возможность импортировать любое изображение для создания специфического элемента для вашей задачи. Когда виртуальный инструмент будет закончен, вы можете использовать элементы лицевой панели для управления системой даже во время выполнения программы, меняя положение переключателей и регуляторов, поворачивая ручки управления и вводя значения с клавиатуры. Таким образом панель "оживает". обеспечивая обратную связь с вашей системой.

### •Блок-диаграммы

Программируя виртуальные инструменты вы освобождаете себя от многих синтаксических деталей обычного программирования. Выбирая функциональные блоки из меню вы соединяете их с помощью проводников для обеспечения передачи данных от одного блока другому. Это могут быть как блоки элементарных алгебраических операций, так и сложные функции сбора и анализа данных, сетевые операции и файловый ввод/вывод, обмен данными с жестким диском в ASCII, бинарном формате и в формате табличного процессора. LabView имеет обширный набор средств для разработки, тестирования и отладки вашей системы. Окно подсказки (Help Window) описывает каждый блок и его соединения. LabView немедленно проинформирует вас о неправильных соединениях и списке ошибок в окне Error Window. В ассортимент отладочных средств входят подсветка выполнения блок-диаграммы, пошаговый режим, прерывания и индикация значений. Таким образом, вы можете производить трассировку и исследование выполнения программы непосредственно на блок-диаграмме.

### •Структурное программирование

В то время как потоки данных предпочтительны для параллельных операций, вы можете задавать и специальный порядок выполнения. LabView, законченная система программирования, предлагает такие программные структуры, как итерактивный

цикл (FOR), последовательный цикл (WHILE) и оператор выбора (CASE), для последовательных, повторяющихся или разделяющих операций. Эти структуры представлены как графические рамки, окаймляющие управляемые блоки на блок-диаграмме.

***•Модульность и иерархия***

LabView является модульной средой по своей структуре. Любой VI может использоваться в блок-диаграмме другого виртуального инструмента как subVI. Разбив свою программную систему на subVI, вы можете независимо разработать и интерактивно протестировать эти subVI, и тут же использовать их как узлы для построения виртуального инструмента более сложного уровня. Использование модульной иерархии позволяет эффективно разрабатывать, модифицировать, заменять и комбинировать виртуальные инструменты для удовлетворения изменяющихся требований конкретного приложения. Ваши возможности значительно расширяет иерархия VI. Создавая пиктограмму для собственного VI и используя ее в диаграмме другого виртуального инструмента, вы скрываете сложность низкоуровневой диаграммы, однако сохраняете доступ к общим переменным через панели нижнего уровня. Вы можете даже конфигурировать эти панели для автоматического открытия, создания анимаций и контекстозависимого интерфейса пользователя .

***•Графический компилятор***

Во многих приложениях, скорость выполнения является критичной. LabView — единственная графическая среда программирования с компилятором, который генерирует оптимизированный код. Скорость выполнения LabView близка к скорости выполнения компилированных Си программ. Поэтому, используя данный графический язык, вы можете увеличить свою производительность при создании программ без снижения скорости их выполнения.

## 2.3. Первое знакомство с LabView

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

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

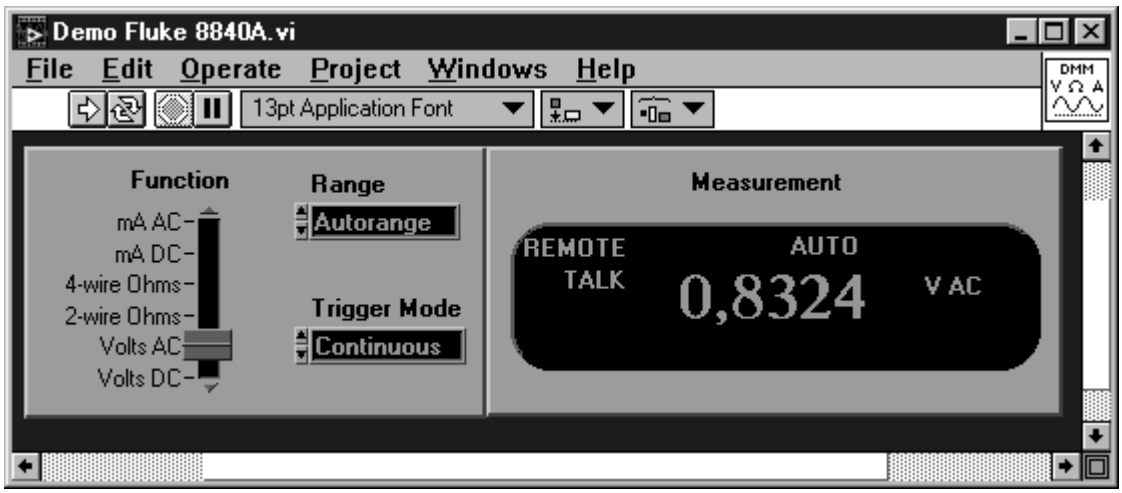



Рис.7 Лицевая панель ВП - аналога цифрового тестера Fluke 8840A

На Рис.7 к индикаторам относится, например, табло "Measurement", отображающее разряды числового значения измеряемой величины, а к средствам управления - ползунок "Function", переключатель диапазона измерений "Range" и переключатель режима измерений "Trigger Mode".

2. Вторая часть, *блок-схема* (или *блок-диаграмма*) описывает алгоритм работы ВП.

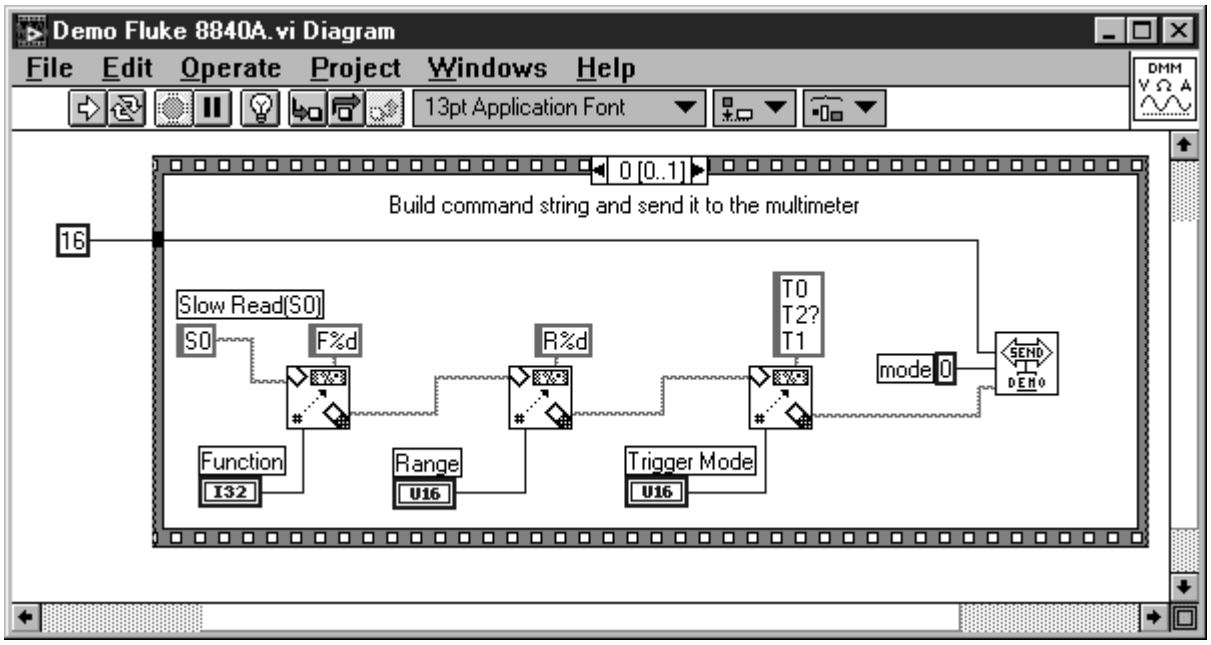



Рис.8 Блок-схема ВП - аналога цифрового тестера Fluke 8840A

Каждый ВП, в свою очередь, может использовать в качестве составных частей другие ВП, подобно как любая программа, написанная на языке высокого уровня, использует свои подпрограммы. Такие ВП нижнего уровня обычно называются субВП. На Рис.8 к субВП относится элемент "Send DEMO" - это ВП, непосредственно реализующий операции по переключению диапазонов, преобразованию сигналов, генерации поразрядного представления результата и т.п.

Также на рисунке можно отметить многочисленные функциональные блоки, играющие роль "задних контактов" для объектов лицевой панели, - это так

называемые терминалы. Каждому терминалу обязательно соответствует какой-либо индикатор или средство управления, расположенные на лицевой панели.

Важными элементами блок-схемы являются функциональные узлы - встроенные субВП, являющиеся частью LabView и выполняющие предопределенные операции над данными.

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

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

## 2.4 Палитры LabVIEW.

В LabView есть три графические палитры Tools (инструментальная палитра), Controls (палитра управления), Functions (палитра функций), которые можно свободно перемещать по экрану. Они служат для создания и реализации виртуальных приборов (ВП).

### 2.4.1 Палитра Tools (Инструментов)

Эта палитра содержит инструменты, которые Вам понадобятся для построения и использования ВП. Вы можете создавать, изменять, и отлаживать ВП, используя эти инструменты. Если палитра Tools - не видна, выберите Show Tools Palette в меню Windows, чтобы палитра появилась. После того, как Вы выбираете инструмент из этого меню, курсор мыши обретет его форму. Вы можете использовать любой из инструментов, найденных в палитре Tools для работы с подпрограммами и функциями. Для получения информации о подпрограммах и функциях необходимо поместить любой из инструментов палитры Tools на нужный объект.

| | | | |
|---|---|---|---|
| | Инструмент управления. Используйте его, чтобы работать с передней панелью управления и индикаторами. | | Инструмент Расположения. Используйте этот инструмент, чтобы выбирать, перемещать, или изменять размеры объектов. Инструмент изменяется на при изменении размеров объекта. |

| | | | |
|---|---|---|---|
|  | Маркер. Используйте этот инструмент, который выглядит так  , для ввода текста в ярлыки (метки) или создания свободных ярлыков (меток). Инструмент изменяется на  , когда Вы создаете свободные ярлыки (метки). |  | Соединительный кабель. Используйте этот инструмент, чтобы соединять объекты проводами на блок-схеме. |
|  | Инструмент объектного всплывающего меню. Используйте этот инструмент, чтобы вызвать всплывающее меню объекта с помощью левой кнопкой мыши |  | Инструмент прокрутки. Используйте этот инструмент для просмотра окна без использования полос прокрутки. |
|  | Инструмент Контрольной точки. Используйте этот инструмент, чтобы установить контрольные точки на ВП, функциях, и структурах. |  | Пробник. Используйте этот инструмент для того, чтобы снимать пробы на проводах на схеме. |
|  | Инструмент копирования цвета. Используйте этот инструмент для копирования цвета и вставки его с помощью инструмента цвета. |  | Инструмент цвета. Используйте этот инструмент для окраски объекта. С его помощью можно также отобразить передний план и фон объекта. |

**Палитра Controls (управления) и палитра Functions (функций).**
Палитра управления и палитра функций составлены из значков верхнего уровня, представляющих подпалитры, дающие доступ к полному диапазону доступных объектов, которые могут использоваться в создании ВП. К подпалитрам можно обращаться, нажимая на значок верхнего уровня.

### 2.4.2 Палитра Controls (управления).

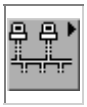
Вы добавляете средство управления и индикаторы к передней панели через палитру Controls. Каждая опция в палитре отображает подпалитру доступного средства управления и индикаторов для выбора. Если палитра Controls - не видна, Вы можете открыть палитру, выбирав Show Controls Palette в меню Windows. Вы можете также вызвать палитру Controls, открыв всплывающее меню на пустой области на передней панели. Вы можете открыть всплывающее меню щелкнув по пустой области передней панели, далее можно использовать правую кнопку мыши. Палитра Controls может быть "пришпилена" к рабочему столу с помощью

кнопки в левом углу палитры, либо убрана кнопкой "крестик".


| | | | |
|---|---|---|---|
|  | Подпалитра Numeric (числовых значений). Состоит из средств управления и индикаторов для числовых данных. |  | Подпалитра Boolean (Булевых значений). Состоит из средств управления и индикаторов для Булевых величин (либо 0, либо 1). |
|  | Подпалитра String (строковых значений). Состоит из средств управления и индикаторов для строк и таблиц |  | Подпалитра List & Ring (списков закольцованных списков). Состоит из средств управления и индикаторов для меню, выполненных в форме списков и закольцованных списков. |
|  | Подпалитра Array & Cluster (массивов и кластеров). Состоит из средств управления и индикаторов для группировки наборов типов данных. |  | Подпалитра Graph. Состоит из индикаторов, чтобы построить график данных в графах или диаграммах в реальном масштабе времени (осциллограф). |
|  | Подпалитра Path & Refnum (путей и ссылок). Состоит из средств управления и индикаторов для путей и ссылок. |  | Подпалитра Decorations (оформления). Состоит из графических объектов для настройки дисплеев передней панели. |
|  | Подпалитра Control . Отображает диалоговое окно, чтобы загрузить самодельные элементы управления |  | Подпалитра User Controls (средства управления пользователя). Состоит из специальных средств управления, которые формирует сам пользователь. |
|  | Подпалитра ActiveX (объектов ActiveX). Состоит из средств управления, позволяющих внедрить объекты ActiveX на переднюю панель. | | |

### 2.4.3 Палитра Functions (функций)

С помощью палитры Functions Вы формируете блок-схему. Каждая опция в палитре отображает подпалитру значков верхнего уровня. Если палитра Functions - не видна, Вы можете вызвать палитру, выбрав Show Functions Palette в меню Windows. Вы можете также открыть палитру Функций, вызвав всплывающее меню на пустой области в окне Diagram. Палитра Функций может быть "пришпилена" к рабочему столу с помощью кнопки в левом углу палитры, либо убрана кнопкой "крестик".

| | | | |
|---|---|---|---|
|  | Подпалитра Structures (структур). Состоит из управляющих структур программы, таких как цикл For . |  | Подпалитра Numeric (числовых функций). Состоит из тригонометрических, логарифмических и числовых функций |
|  | Подпалитра Boolean (Булевых функций). Состоит из логических и Булевых функций. |  | Подпалитра String (строковых функций). Состоит из функций для работы со строковыми величинами. |
|  | Подпалитра Array (массивов). Состоит из функций для обработки массивов. |  | Подпалитра Cluster (кластеров). Состоит из функций для обработки кластеров. |
|  | Подпалитра Comparison (сравнения). Состоит из функций для сравнения числовых, строковых значений, Булевых переменных. |  | Подпалитра Time & Dialog. Состоит из функций для диалоговых окон, синхронизации, и обработки ошибок. |
|  | Подпалитра File I/O (ввода/вывода файла). Состоит из функций и ВП |  | Подпалитра Communication (связи). |

| | | | |
|---|---|---|---|
| | для ввода/вывода в файл. | | Состоит из ВП для работы с сетями TCP, DDE, Apple Events, и OLE. |
|  | Подпалитра Instrument I/O ( инструментов ввода/вывода). Состоит из ВП для связи и управления приборами по шине GPIB, VISA(программная архитектура виртуальных приборов). |  | Подпалитра Data Acquisition (сбора данных). Состоит из ВП для внедрения плат сбора данных. |
|  | Подпалитра Analysis (анализа). Состоит из ВП для анализа данных. |  | Подпалитра Tutorial (обучающей программы). Состоит из ВП, используемых в обучающей программе LabVIEW. |
|  | Подпалитра Advanced (расширенная). Состоит из разных функций типа функции библиотечного запроса, манипуляции данных, и т.д. |  | Подпалитра VI…. Состоит из диалогового окна для внедрения подпрограмм в текущий ВП. |
|  | Подпалитра Instrument Drivers (драйверы приборов ). Состоит из ВП, способных управлять внешними приборами, осциллоскопами, генераторами, и т.д., через последовательный порт или интерфейс GPIB |  | Подпалитра User Libraries. С помощью нее организуется быстрый доступ к нужному ВП. |
|  | Подпалитра Application Control (управления приложением). Состоит из ВП, управляющих виртуальными приборами (ВП), а также ВП VI серверов, позволяющих запускать ВП на других компьютерах через сеть. | | |

## 2.5 Как создается программа

ВП имеют три основных части: передняя панель, блок-схема, и значок.

***Передняя панель.***
Вы формируете переднюю панель из VI с комбинацией средств управления и индикаторов. Средства управления - ваши средства поставки данных к вашему ВП. Индикаторы отображают данные, который ваш ВП генерирует. Есть много типов средств управления и индикаторов. Вы добавляете средства управления и индикаторы к передней панели от различных подпалитр палитры Controls. Два наиболее часто используемых элемента - цифровой элемент управления и

цифровой индикатор. Чтобы вводить или изменять значения в цифровом элементе управления, Вы можете нажать на кнопки приращения при помощи инструмента управления или двойным щелчком по числу при помощи маркера  или инструмента управления  .

**Ноды**

Нод - программный элемент. Ноды аналогичны операторам, функциям, и подпрограммам, используемым в традиционных текстовых языках программирования. Есть четыре типа нодов - функции, ноды-подпрограммы, структуры, и Code Interface Node (CINs) (элемент, содержащий фрагмент кода, написанного на традиционном языке программирования). Функции - встроенные ноды для выполнения элементарных операций типа добавления чисел, ввода - вывода в файл, или форматирования строковых значений. Ноды-подпрограммы - ВП, с помощью которых Вы создаете и позднее вызываете из основной программы другой ВП. Структуры типа цикла For и, цикла While управляют выполнением программы. CINs - интерфейсы основной программы и кода, написанного на C. Терминалы - порты, через которые данные проходят между блок-схемой и передней панелью и между нодами блок-схемы.

**Терминалы**

Терминалы аналогичны параметрам и константам. Есть два типа терминалов - терминал управления или индикатор и терминал-нод. Терминалы управления и терминалы-индикаторы принадлежат к средствам управления и индикаторам передней панели. Значения, что оператор или вызывающий ВП вносит в эти средства управления, идут к блок-схеме через эти терминалы, при выполнении этих ВП . Когда ВП заканчивает работу, выходные данные идут от блок-схемы к передней панели через терминалы индикатора. Управление и терминалы индикатора автоматически создаются или удаляются, когда Вы создаете или удаляете средство управления или индикатор передней панели. Блок-схема из ВП на предыдущей странице показывает терминалы, принадлежащие четырем средствам управления и индикаторам передней панели. Подобно ВП, функции Add и Subtract также имеют терминалы-ноды, которые лежат в основе значка этих функций.

**Проводка**

Провода - пути данных между терминалами. Они аналогичны переменным на обычных языках. Данные идут в только одном направлении, с исходного терминала на один или более терминалов адресата. Различные образцы провода представляют собой различные типы данных. На цветном мониторе каждый тип данных появляется в различном цвете для акцента. Примеры основных типов проводов показаны ниже.

| | Scalar | 1D Array | 2D Array | Color |
|---|---|---|---|---|
| Number | | | | Orange (floating point), Blue (integer) |
| Boolean | | | | Green |
| String | | | | Purple |

Проводка осуществляется с помощью левой кнопки мыши.

При поднесении этого инструмента к терминалу появляется "размотанный конец кабеля". Чтобы соединить проводами два терминала, нажмите Соединительный кабель на первом терминале, переместите инструмент ко второму терминалу, и нажмите на него. Не имеет значения, с какого терминала начинать.

Когда Соединительный кабель находится над терминалом, область терминала мигает, указывая, что щелчок подключит провод к этому терминалом. Вы не должны держать кнопку мыши при перемещении Соединительного кабеля от одного терминала до другого. Вы можете изгибать провод, щелкая кнопку мыши, чтобы прикрепить провод и перемещать мышь в перпендикулярном направлении. При нажатии на клавишу "пробел" изменяется трасса проводника.

**Всплывающие подсказки**

Всплывающие подсказки облегчают идентификацию терминалов функций и нодов для соединения. При перемещении Соединительного кабеля поверх терминала, всплывает полоса советов. Всплывающие подсказки состоят из маленьких, желтых текстовых заголовков, которые отображают название терминала.


**Создание объектов передней панели с помощью блок-схемы.**

С любым инструментом LabView, Вы можете вызвать всплывающее меню на любой функции или подпрограмме LabView и выбирать Create Constant, Create Control или Create Indicator. Если Вы используете Соединительный кабель, созданная константа, управление, или индикатор будут соединены соответственно автоматически.

**Использование помощи**

Все встроенные функции LabView и ВП имеют полный интерактивный справочник. Если Вы нашли незнакомую функцию или ВП, размещаете их в вашу блок-схему, открываете всплывающее меню, и выбираете Online help для полного описания функций и параметров объекта.

Если Вы нуждаетесь в простой справке, чтобы напомнить Вам о ВП или функции, и ее параметрах ввода и вывода, выберите Show Help из Help menu, и появится окно справки.

Окно справки контекстно-зависимо, так всякий раз, когда Вы нуждаетесь в быстрой справке, поместите указатель мыши поверх ВП или функции. Переключите кнопку блокировки внизу окна на Locked Help , чтобы

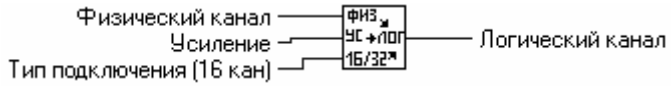
блокировать окно справки для текста, появившегося в окне последним. Переключите эту кнопку обратно, чтобы сделать окно справки вновь контекстно-зависимым.

## 2.6 Как связана плата АЦП и LabView

Для того, чтобы в программах на LabView можно было непосредственно работать с платами АЦП, существуют специальные библиотеки виртуальных инструментов, которые предоставляют нам методы для взаимодействия с платой. Эти виртуальные инструменты представляют собой соответствующим образом оформленный программный интерфейс для вызова функций из LabView.

Приведем ниже основные функции.

### *Формирование лог. номера канала АЦП (AI Create Channel - CREATE_CHANNEL.vi)*




ВИ рассчитывает логический номер канала (***логический канал***). Все функции аналогового ввода требуют на входе логический номер канала, формат которого описан в документации на конкретную плату. Для того, чтобы по физическому номеру канала (т.е. по номеру входной линии АЦП 0-15 или 0-31) определить логический номер канала, необходимо использовать данную функцию

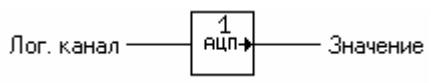
Входные параметры:

I. Физический номер канала АЦП (***физический канал***) от 0 до 31

II. ***усиление*** - код входного диапазона (по умолчанию принимается ± 2В)
   - A. 0 ± 1В
   - B. 1 ± 2В
   - C. 2 ± 5В

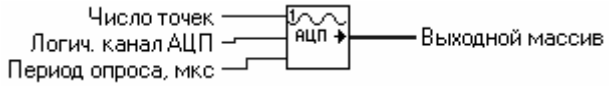
III. ***тип подключения*** каналов (по умолчанию 16-канальный)
   - A. 0 - 16 каналов
   - B. 1 - 32 канала

### *Однократный ввод с канала АЦП (AI Sample Channel - ADCHAN.vi)*



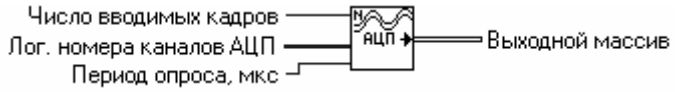

ВИ устанавливает заданный ***логический канал*** АЦП и осуществляет аналого-цифровое преобразование. Данная функция удобна для осуществления асинхронного ввода с разных каналов АЦП. Возвращает результат преобразования по каналу.

### *Аналоговый одноканальный ввод (AI Acquire Waveform - STREAM.vi)*



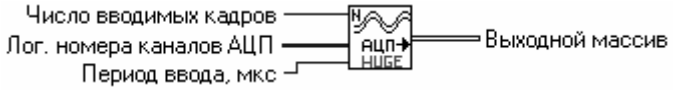

ВИ считывает последовательность отсчетов с заданного ***логич. канала АЦП*** с интервалом ***период опроса*** между отсчетами. Результат ввода возвращается в одномерном массиве ***выходной массив***.

### *Аналоговый многоканальный ввод (AI Acquire Waveforms - SOFT.vi)*



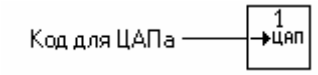

ВИ осуществляет ввод ***числа вводимых кадров*** по аналоговым каналам АЦП, логические номера которых передаются в массиве ***лог. номера каналов АЦП***. Под кадром подразумевается результат опроса всех заданных каналов АЦП. Результат ввода возвращается в двумерном массиве, в котором первый индекс - номер кадра (отсчета), второй индекс - канал АЦП.
Размер вводимого массива не должен превышать 64Кб. Данное ограничение снято в ВИ "AI Huge Waveforms - SOFT_HUGE.vi", который присутствует только в коммерческой версии библиотеки.

### *Аналоговый многоканальный ввод больших массивов (AI Huge Waveforms - SOFT_HUGE.vi)*



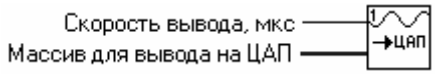

ВИ осуществляет ввод ***числа вводимых кадров*** по аналоговым каналам АЦП, логические номера которых передаются в массиве ***лог. номера каналов АЦП***. Под кадром подразумевается результат опроса всех заданных каналов АЦП. Результат ввода возвращается в двумерном массиве, в котором первый индекс - номер кадра (отсчета), второй индекс - канал АЦП.
В отличие от ВИ "AI Acquire Waveform - STREAM.vi" в данном ВИ можно вводить массивы больше 64Кб.

### *Асинхронный вывод на ЦАП (AO Set DA Code - OUTDA.vi)*




Устанавливает выходное напряжение на ЦАПе в соответствии с ***кодом***.

### *Синхронный одноканальный вывод на ЦАП (AO Generate Waveform - DASTREAM())*




ВИ выводит массив отсчетов с указанным периодом на ЦАП.

Размер массива ограничен 64 Кб (т.е. можно вывести максимум 32767 отсчетов). Данное ограничение снято в ВИ "AO Generate Huge Waveform - DASTREAM.vi", который присутствует только в коммерческой версии библиотеки.

Все эти функции находятся в подпалитре User Libraries палитры Functions (функций).

## III. Упражнения

### 3.1 Простейшая программа, строящая график sin(t)

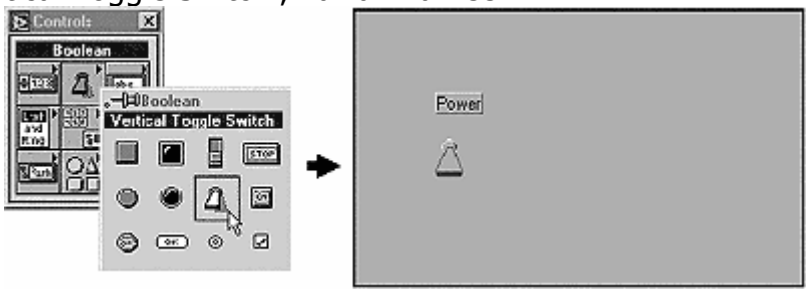
Задача

Создадим ВП, строящий график sin(t).

Реализация

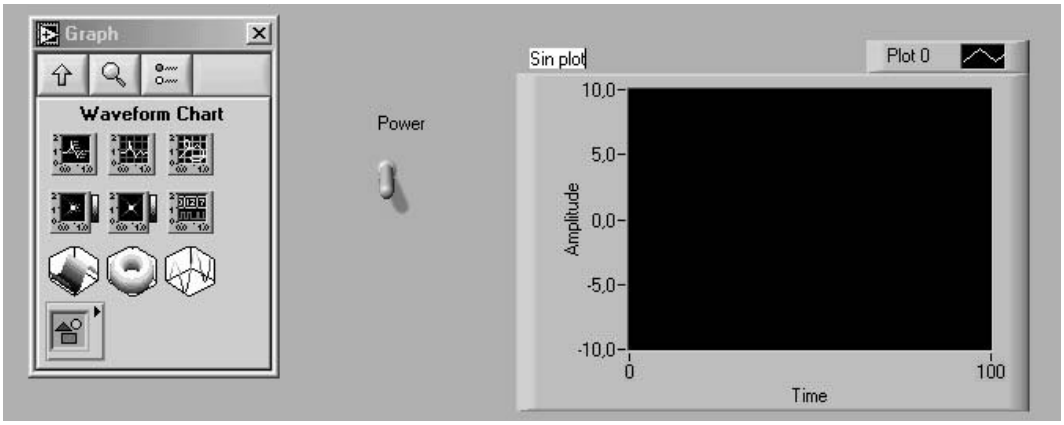
1. Запустите LabView и создайте новый ВП, выбрав New VI в диалоговом окне LabView.

3. Выберите и разместите Vertical Toggle Switch (Boolean (Булевы величины)) на передней панели. Нажмите на подпалитру Булевых значений (Boolean) в палитре Controls, и переместите указатель мыши, чтобы выбрать Vertical Toggle Switch . Как только Вы выбираете его переместите указатель мыши на переднюю панель, и разместите Vertical Toggle Switch , нажав на нее.

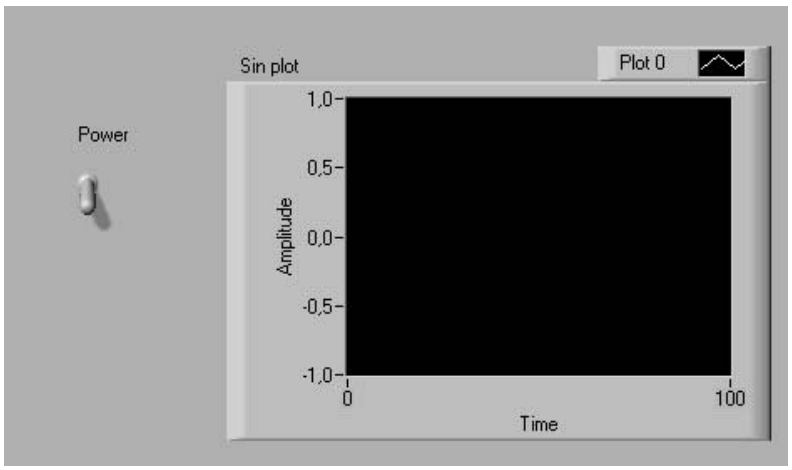



4. Напечатайте Power в метке для выключателя. Если метки нет, выберите Show Label из всплывающего меню выключателя. Используйте инструмент Маркер , чтобы подсветить и изменить текст любой метки.

5. Выберите и разместите Waveform Chart (Graphs (графы)) на передней панели, и назовите его Sin Plot.

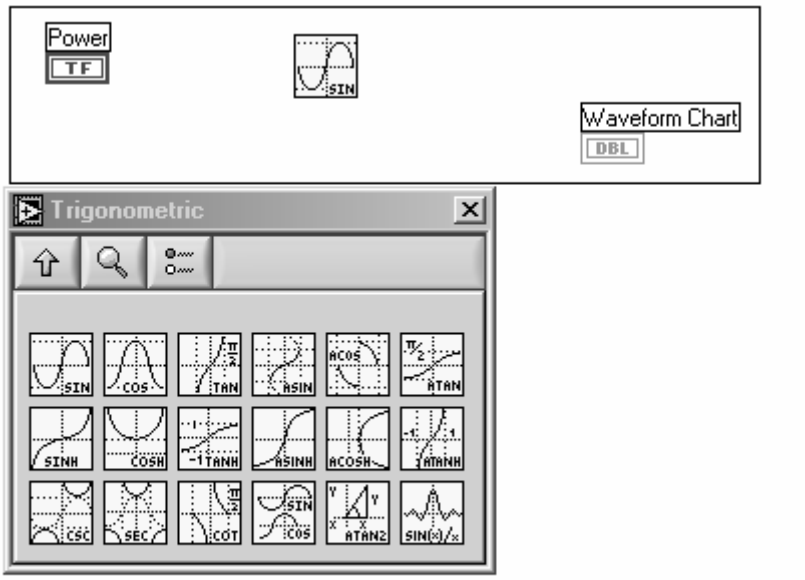



6. Используя инструмент управления , дважды щелкните на 10.0 по Оси Y диаграммы, и введите 1.0 вместо него, так как мы будем составлять график чисел между -1 и 1. Или Вы можете открывать всплывающее меню на диаграмме и выбрать Y Scale (масштаб) > > AutoScale (автомасштаб) Y.



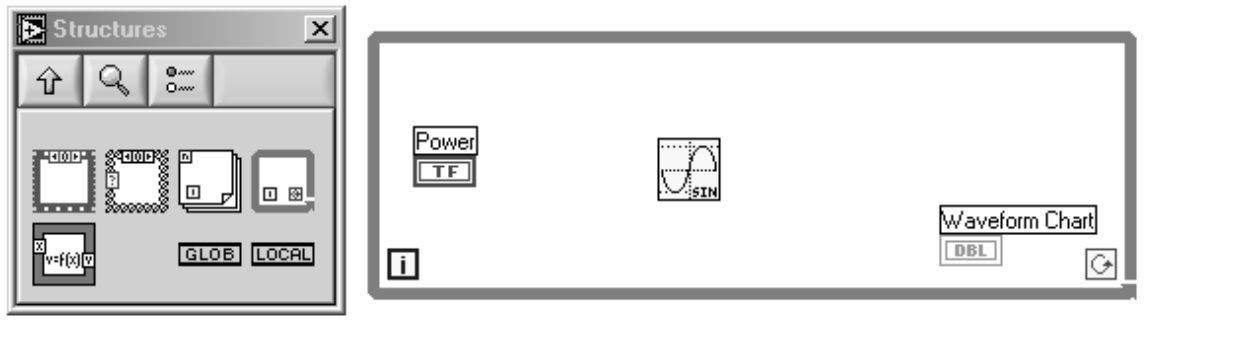

7. Откройте блок-схему, выбрав Windows>>Show Diagram.
8. Обратите внимание, что терминал Sin Plot и терминал Power уже на блок-схеме. Эти терминалы соответствуют элементам, которые Вы создавали на передней панели.
9. Выберите и разместите на блок-схеме sin из палитры Functins (Numeric>>Trigonometric).

**Примечание** Можно посмотреть описание любой функции, созданной в программе, нажав по ней правой кнопкой и выбрав пункт меню Help. Там будет полностью описана функция и все ее параметры.




10. Выберите While Loop (Functions>>Structures (подпалитра Structures)), нажав один раз левой кнопкой мышки по квадратику. Переместите указатель мыши, который приобрел вид специального квадратного значка, к месту, где Вы хотите прикрепить верхний левый угол цикла while. Нажмите левую кнопку мыши и, не отпуская левую кнопку мышшки, переместите указатель так, чтобы в область цикла (прямоугольник на экране) были включены функция sin, а также терминалы Power и Sin Plot .

**Примечание** Цикл while – программа работает следующим образом: функции, включенные в область цикла (которые находятся в квадрате цикла) будут последовательно повторяться до тех пор, пока переменная цикла, обозначенная картинкой  не станет равна нулю. Переменная I, обозначенная картинкой  будет увеличиваться при каждом новом повторении на единицу.

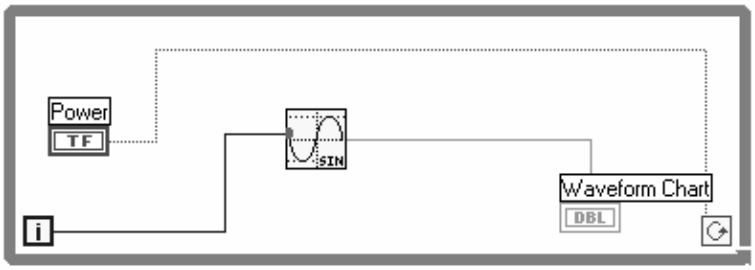



**Возможные ошибки**

Если Вам не удалось успешно применить цикл While, как показано на блок-схеме, откройте всплывающее меню цикла While и выберите Remove Loop. Выберите цикл снова из палитры Functions. Нажмите, чтобы прикрепить верхний левый угол цикла While, и, не используя мышь, перемещайтесь по диагонали поперек кодов (модулей), которые собираетесь включить в цикл. Используйте мышь, когда Вы успешно включили терминалы и функцию sin в цикл.

11. Соедините объекты блок-схемы, используя Соединительный кабель  . Чтобы соединить функцию sin с терминалом Sin Plot , переместите мышь к функции sin (к правой части квадратика, где написано sin). Подождите, пока функция не начнет мигать, затем щелкните левой кнопкой мыши по этой функции. Отпустите кнопку, и двигайтесь к терминалу Sin Plot . Подождите до тех пор, пока терминал Sin Plot не начнет мигать, затем щелкните левой кнопкой мыши, чтобы подсоединить второй конец кабеля. Далее соедините остальные элементы программы, как показано на рисунке.

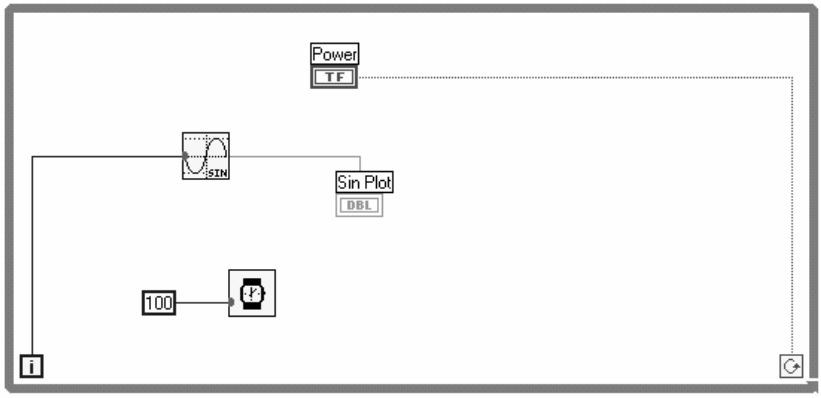
**Примечание** Если у функции несколько входных параметров то когда курсор мыши находится на ее квадратике, то подсвечивается тот параметр, к которому выполнится соединение при нажатии левой кнопки мыши. Обычно входные параметры располагаются слева, а выходные – справа. Для просмотра структуры функции нажмите на ней правой кнопкой и выберите пункт меню Help.

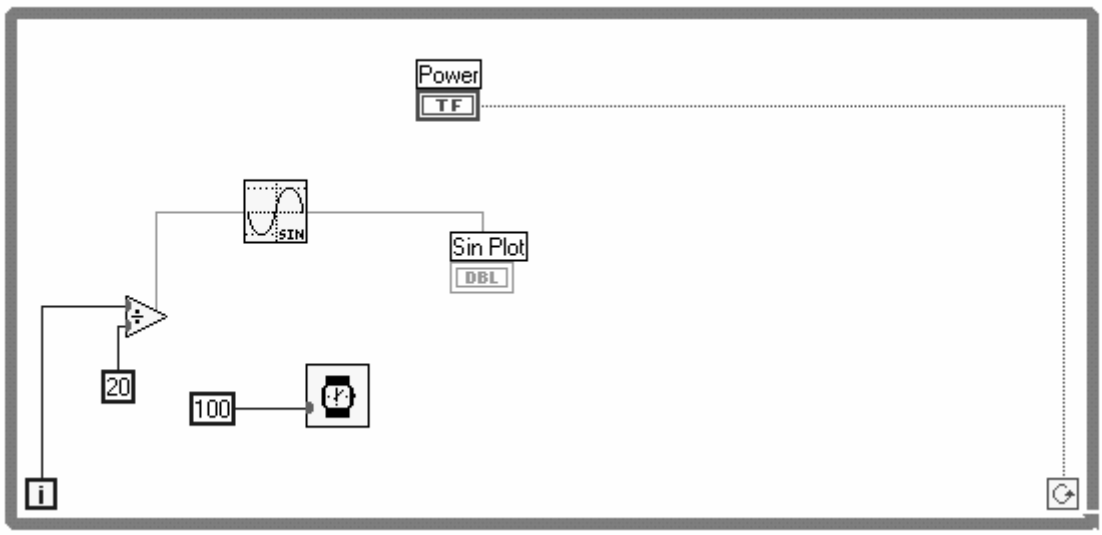

**Возможные ошибки**

Если ваш провод появляется как черная пунктирная линия, выберите Remove Bad Wires из меню Edit. Это значит, что элементы соединены неправильно

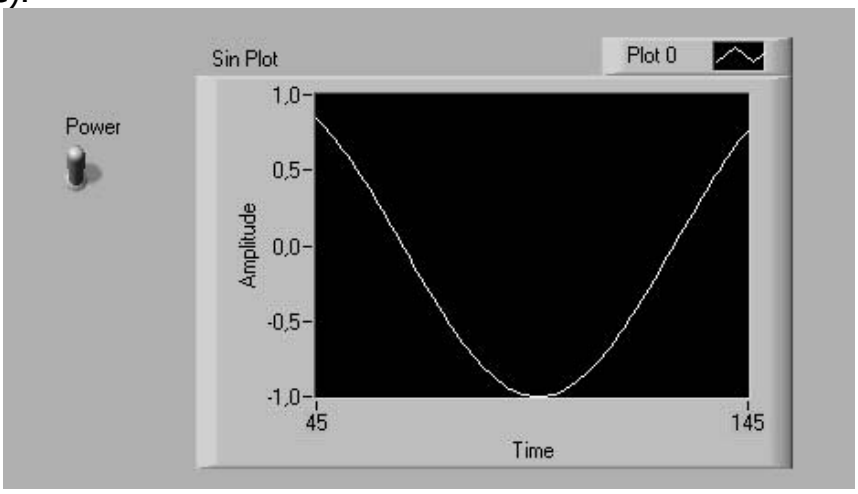
12. Добавьте задержку по времени в цикл как показано на рисунке. (задержку дает функция Wait (Functions>>Time & dialog (подпалитра Time & dialog))).
Далее задайте интервал задержки, создав константу и передав ее в функцию Wait. (Numeric Constant в подпалитре Numeric)




12. Изменим частоту sin, поделив i на число 20. Для этого воспользуемся функцией Divide из подпалитры Numeric и введем константу Constant также из подпалитры Numeric.




13. Возвратитесь на переднюю панель, выбрав Windows > > Show Front Panel, или щелкнув в окне передней панели.
14. С помощью инструмента управления, нажмите на переключатель Power, чтобы включить его (при запуске программы он должен находиться в верхнем положении), и запустите ВП (Запустить программу можно, выбрав пункт Run из меню Operate).

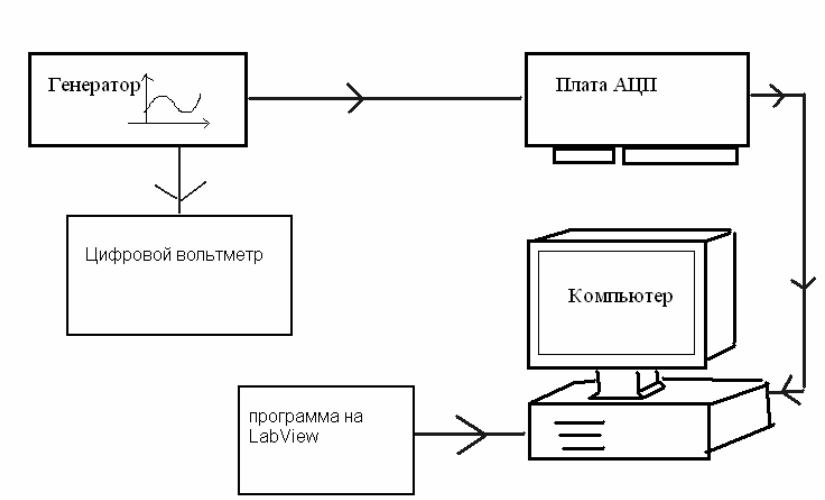



15. Остановите выполнение ВП, нажав на переключатель Power , чтобы выключить его.
Отчетность Показать преподавателю программу, строящую график синуса.

## 3.2 Построение осциллографа. Определение разрядности и динамического диапазона платы.

Задача

Используя плату АЦП и LabView создать осциллограф для визуализации сигнала, подаваемого с внешнего генератора. (Генератор сигналов EFG-3210 подключен к входу платы АЦП, вставленной в компьютер. К выходу генератора также подключен цифровой вольтметр В7-20.)



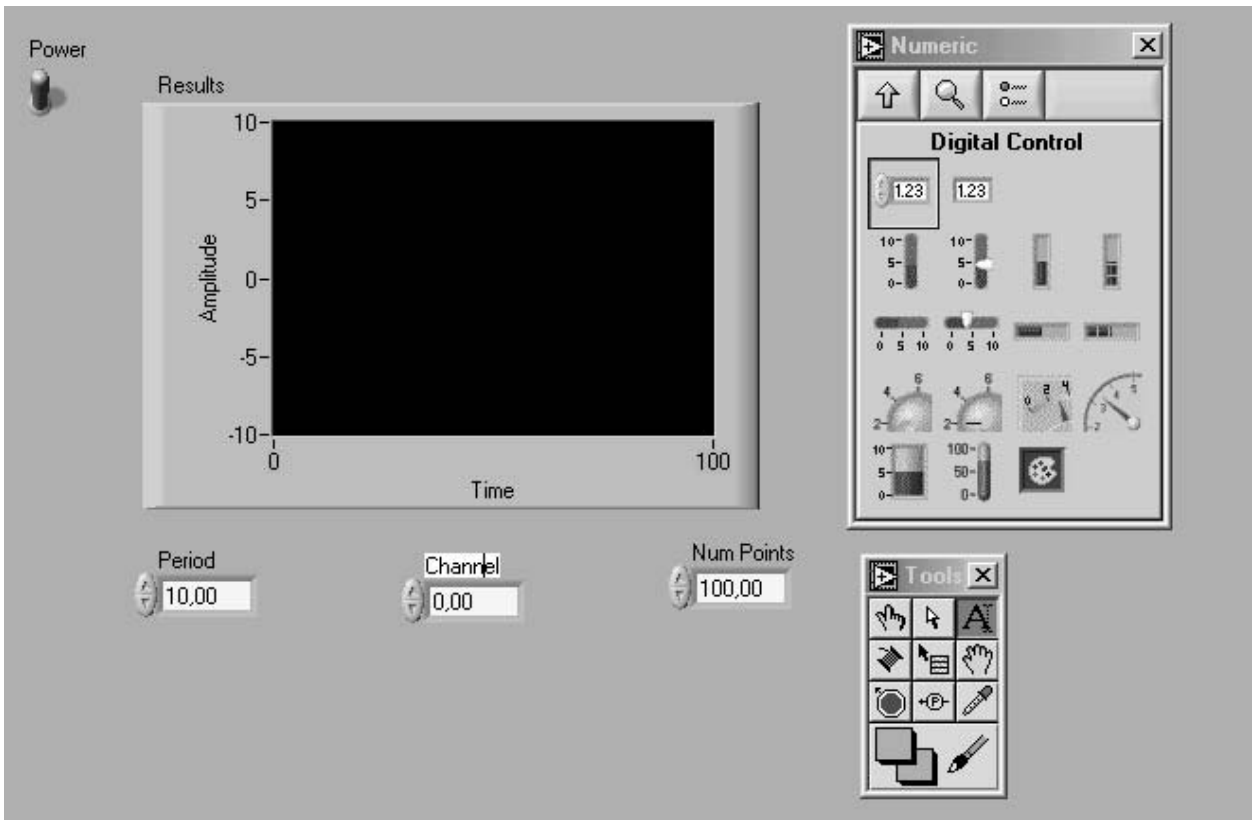

Для этого нам понадобятся функции : Формирование логического номера канала АЦП, Аналоговый одноканальный ввод. Функции для работы с платой находятся в подпалитре
User Libraries>>DLLDRV Vis.

Реализация

Будем использовать функцию Аналоговый одноканальный ввод (AI Acquire Waveform - STREAM.vi) для получения сигнала. Она принимает следующие параметры:

- Число точек - это размер массива, возвращаемого функцией. Каждая точка это значение входного сигнала в определенный момент времени. Расстояние между точками характеризуется периодом опроса.
- Период Опроса
- Номер логического канала – его получим, вызвав функцию Формирование лог. номера канала АЦП (AI Create Channel - CREATE_CHANNEL.vi). Мы говорим LabView, с какого канала поступает входной сигнал.

Надо иметь возможность изменять параметры и выводить результат. Самый простой способ ввода параметров – использовать Digital Control из подпалитры Numeric. Для этого создадим переднюю панель виртуального прибора следующим образом:

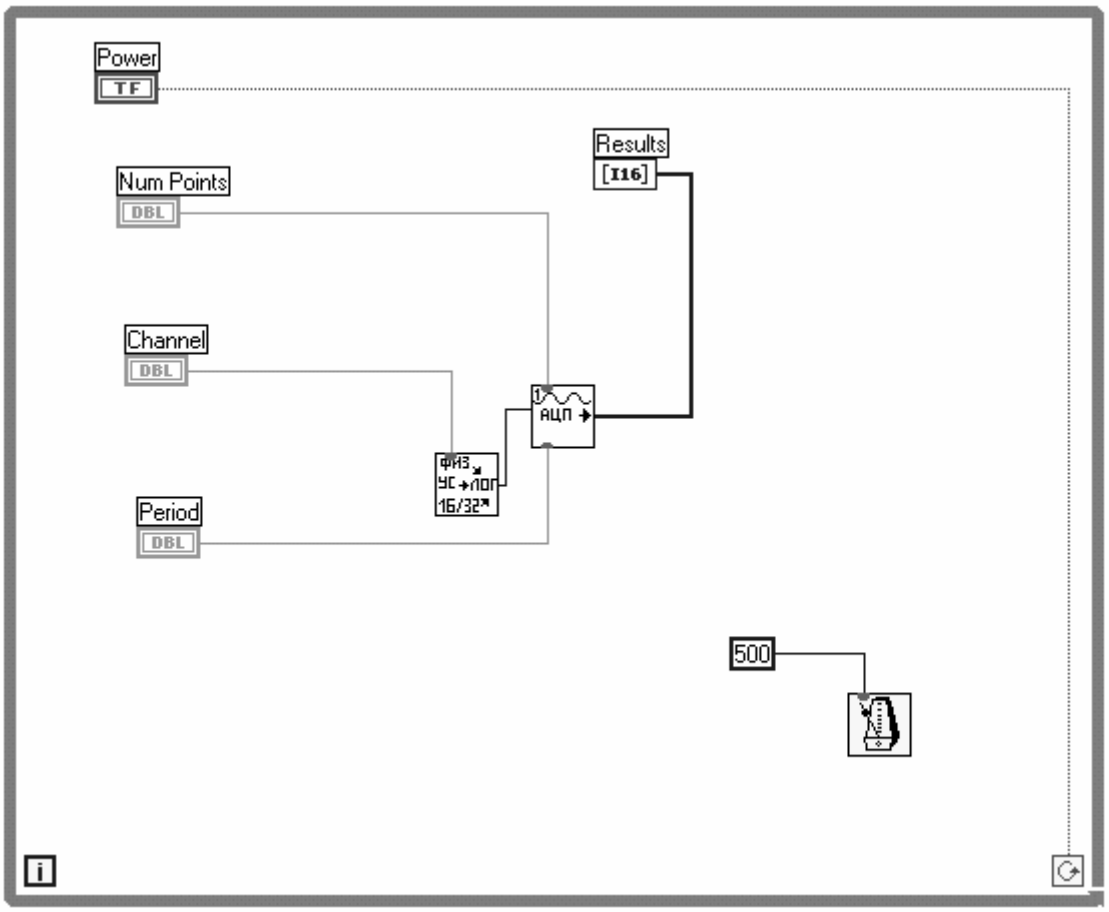



Также необходимо поставить автоматический масштаб у графика, щелкнув по нему правой кнопкой мыши и выбрав пункт меню ScaleY>>AutoScaleY.

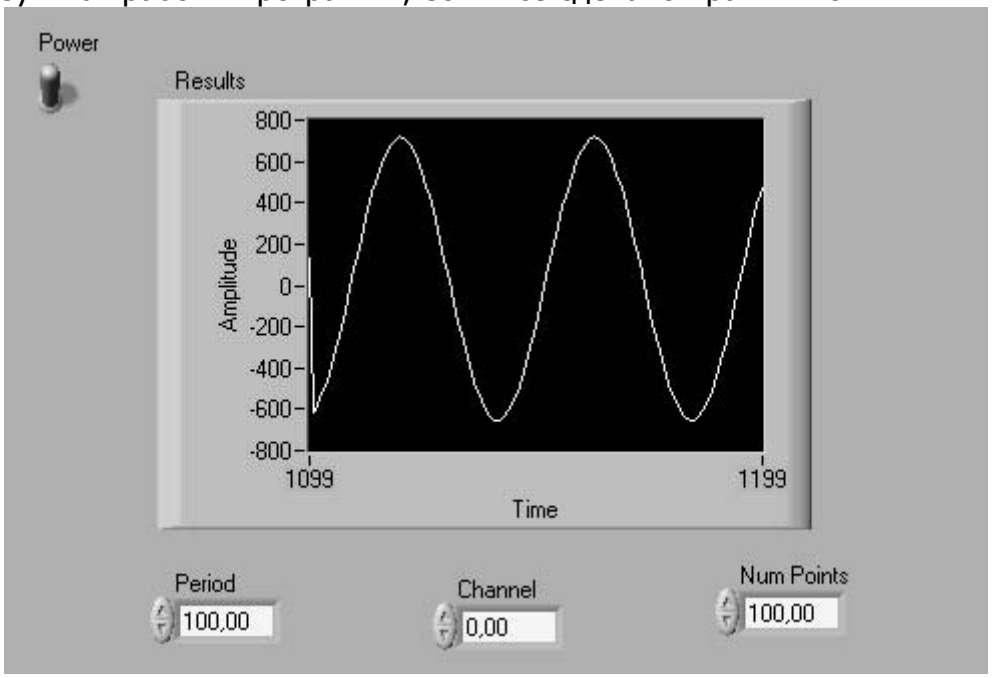
**Примечание** можно использовать любые другие элементы управления для задания необходимых параметров.

Установим начальные значения параметров следующим образом:
Period = 100; Channel = 0; NumPoints = 100;

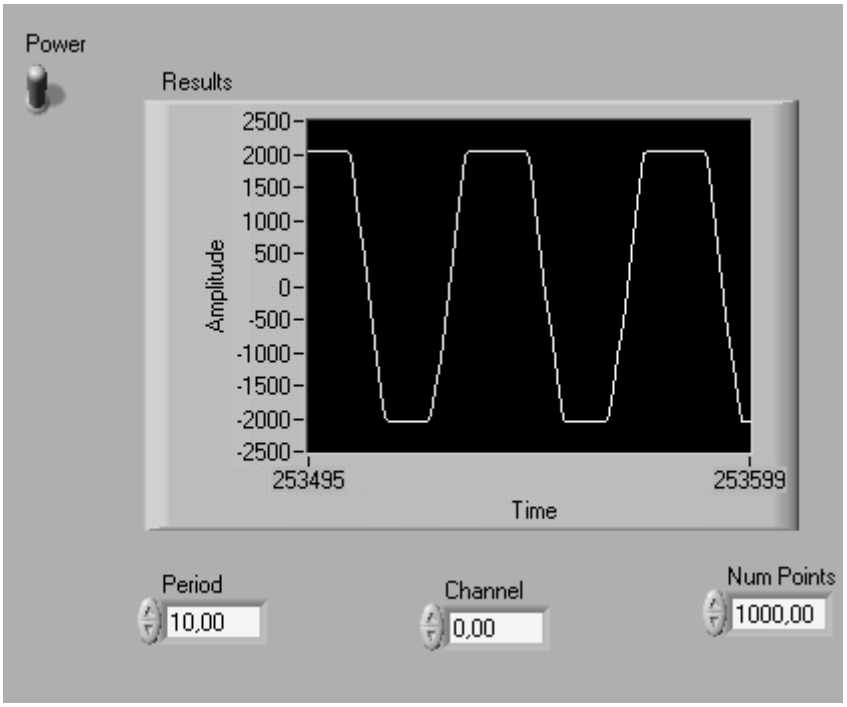
Откройте блок-схему, выбрав Windows>>Show Diagram. Выберите и разместите на блок-схеме функции Аналоговый одноканальный ввод (AI Acquire Waveform - STREAM.vi) и Формирование лог. номера канала АЦП (AI Create Channel - CREATE_CHANNEL.vi) так, как показано на рисунке (функции находятся в подпалитре User Libraries>>DLLDRV Vis). Теперь соедините провода как показано на рисунке. Не забудьте про цикл while.
**Примечание** задержка по времени служит для того, чтобы можно было увидеть сигнал на графике. Иначе картинка меняется слишком быстро.




Результат работы программы, если все сделано правильно:

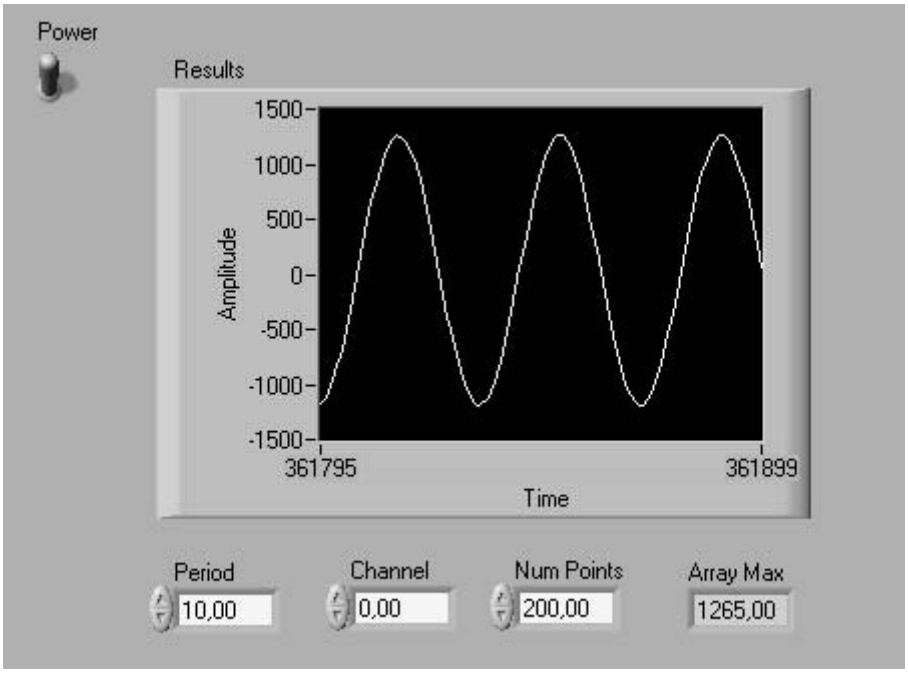


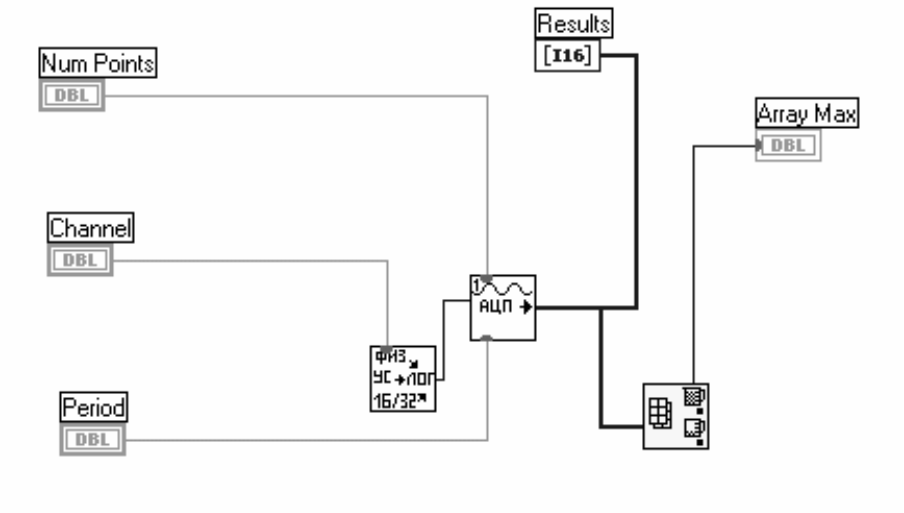

Сначала нам необходимо определить динамический диапазон платы – то есть максимальное напряжение, сигнал с которым можно подавать на плату. Подайте на генератор сигнал синусоидальной формы и изменяйте его амплитуду на генераторе. Когда напряжение станет слишком большим, сигнал начнет обрезаться. Пример обрезанного сигнала:




По вольтметру определите максимальное напряжение, которое можно подавать на плату. Это будет эффективное напряжение $V_{eff}$ . Для получения амплитудного значения $V_{\max}$ разделите эффективное на 0.7 .

Для дальнейших исследований платы нам необходимо знать ее разрядность. Для этого выведем значение оцифрованного сигнала в цифровой форме на передней панели. Функция Аналоговый одноканальный ввод (AI Acquire Waveform - STREAM.vi) возвращает массив чисел. Найдем максимум из массива с помощью функции Array Max & Min (Functions>>Array) и выведем его в текстовое поле на передней панели.

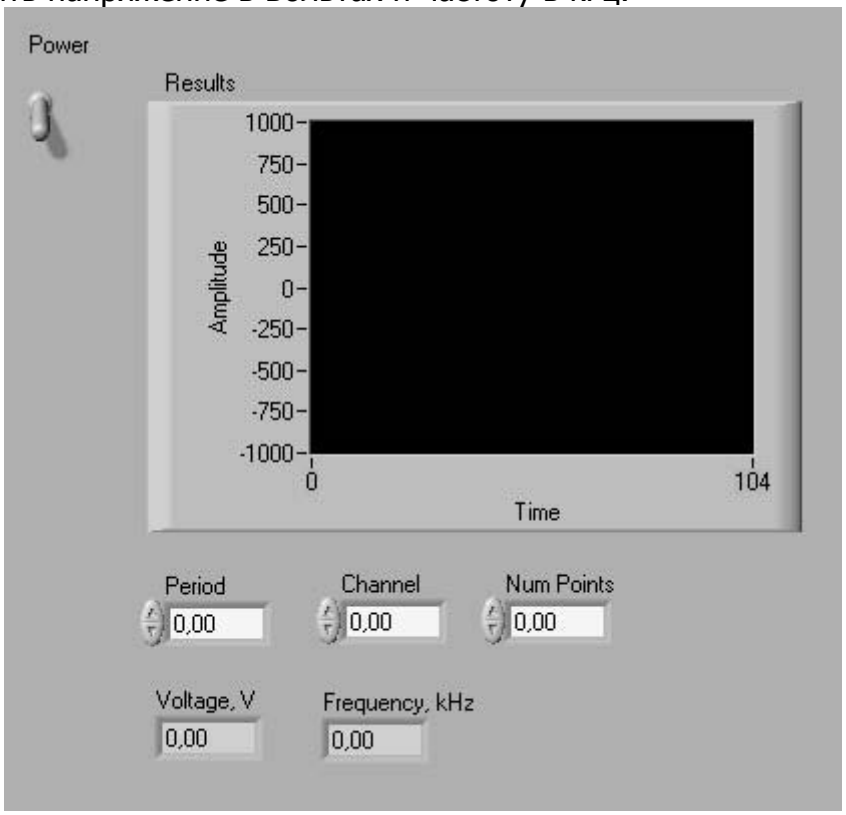





Теперь установим амплитуду на генераторе так, чтобы максимальное значение возвращаемого массива было около $D_1 = 200$. Запишем напряжение на вольтметре - $V_1$. Теперь установим напряжение на генераторе так, чтобы значение амплитуды в цифровой форме равнялось $D_2 = 1000$. Напряжение на вольтметре - $V_2$. Минимальный шаг определяется по формуле $V_0 = \frac{V_2 - V_1}{D_2 - D_1}$.

Теперь найдем разрядность по формуле $n = \log_2(\frac{V_{eff}}{V_0}) + 1$ где $V_0$ - шаг изменения напряжения, а $V_{eff}$ - максимальное напряжение, которое было определено ранее.

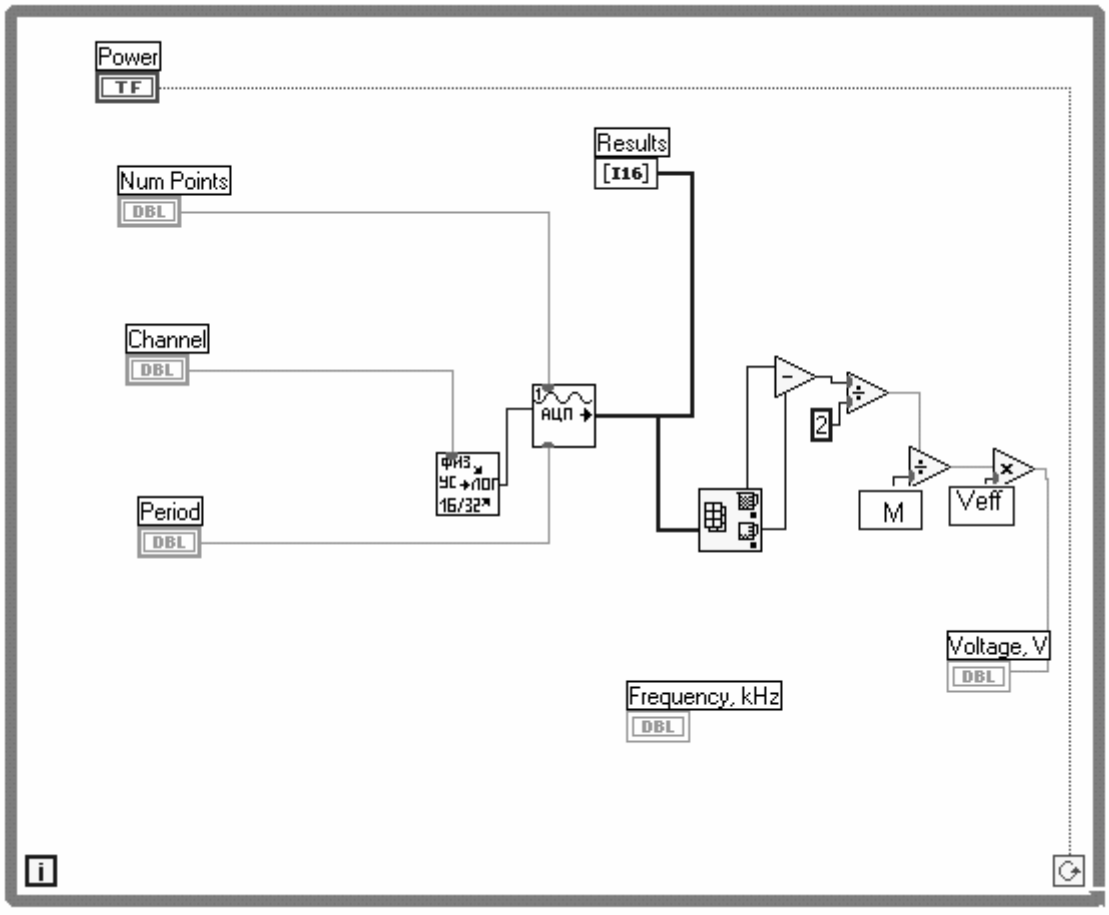
Теперь сделаем возможность определения эффективного напряжения сигнала и частоты. Для этого добавим на передней панели 2 элемента где мы будем выводить напряжение в Вольтах и частоту в кГц.



**Примечание** Вольтметры обычно измеряют среднеквадратичное значение напряжение (или эффективное в случае гармонического сигнала).

Функция Аналоговый одноканальный ввод (AI Acquire Waveform - STREAM.vi) возвращает массив чисел. Каждое число находится в диапазоне(-$M$ , $M$ ), где $M = 2^{n-1}$ (n – разрядность). При оцифровке сигнала происходит сдвиг нуля. Чтобы учесть этот факт будем находить амплитуду, используя максимальное и минимальное значения напряжения сигнала. Найдем минимум $A_{\min}$ и максимум $A_{\max}$ из массива с помощью функции Array Max & Min (Functions>>Array) и определим амплитуду по формуле $U = \frac{(A_{\max} - A_{\min})}{2 \cdot M} V_{eff}$ .

**Примечание** Если константы дробные и вы используете элемент Numeric Constant то для их задания умножьте константы на 100 и соответсвенно введите дополнительное деление на 100.

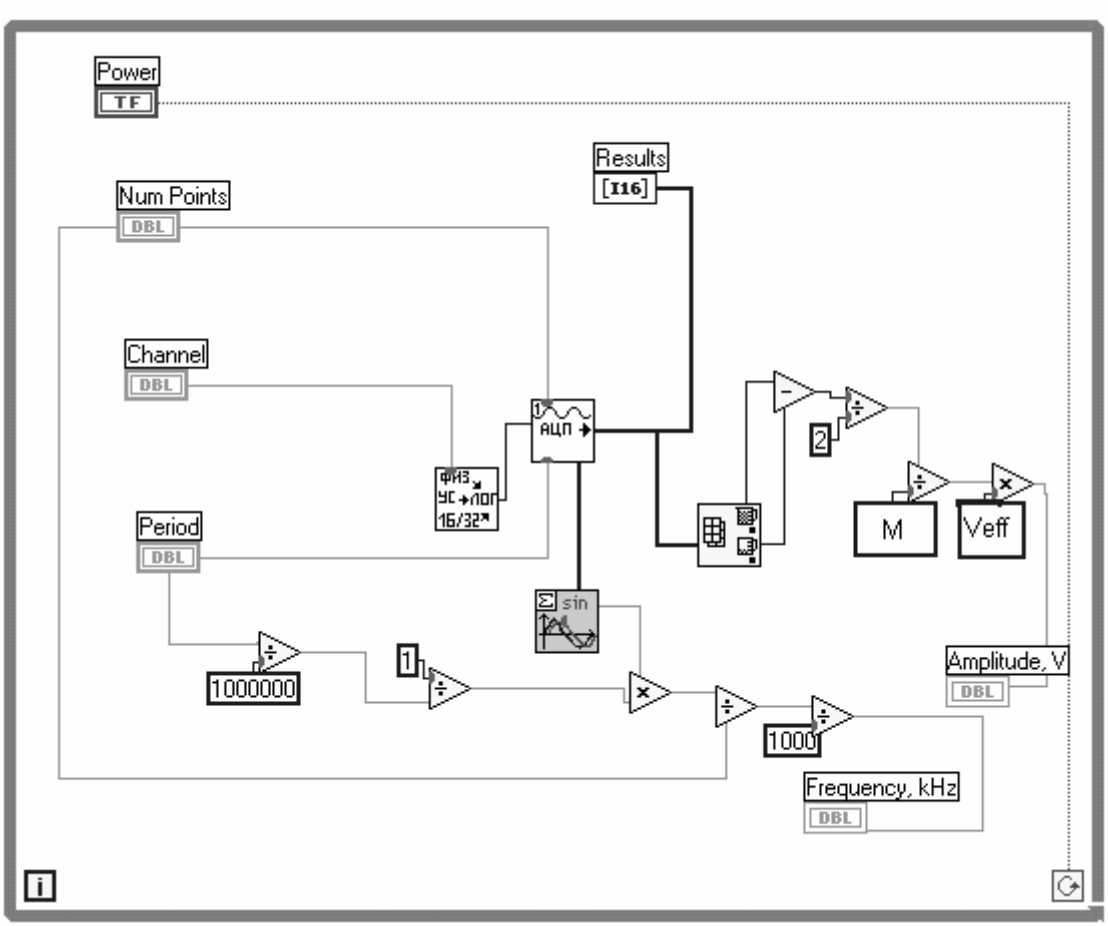



Для определения частоты гармонического сигнала нам понадобиться функция Buneman Frequency Estimator (Functions>> Analyse>> Signal Processing >>Frequency Domain) которая служит для определения частоты.

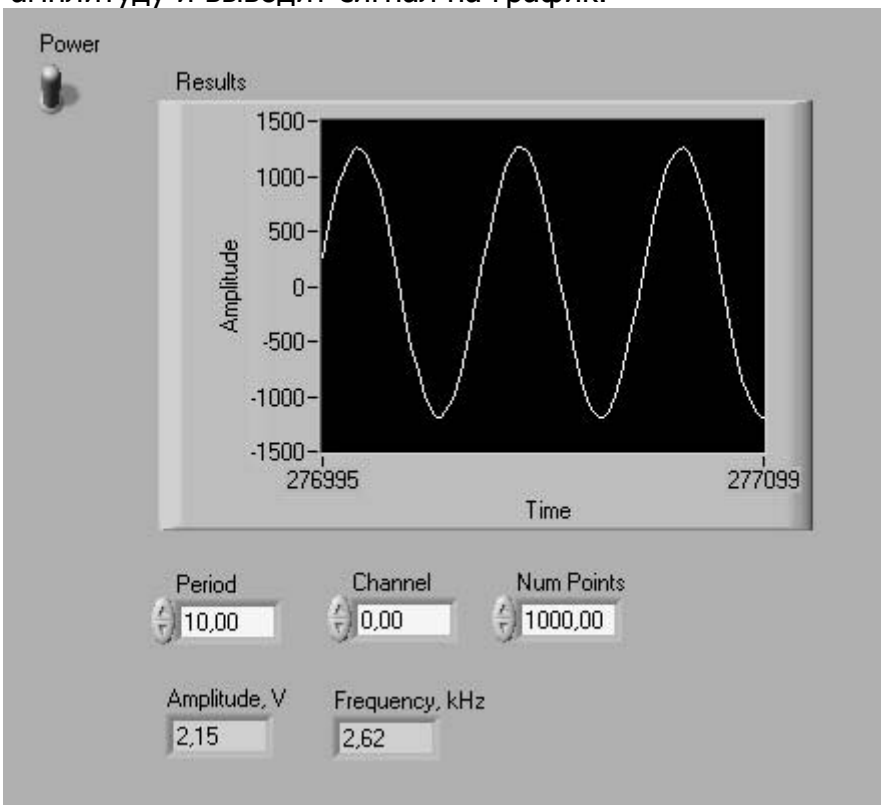

Она возвращает переменную $beta$ . Частота определяется по формуле

$$f = \frac{beta \cdot (f_{descr})}{количество\_точек}$$

Где $f_{descr} = \frac{1}{T}$ - частота дискретизации , где $T$ - период опроса.




Если все сделано правильно, то программа определяет частоту гармонического сигнала, его амплитуду и выводит сигнал на график.



Отчетность Показать преподавателю программу, выводящую оцифрованный сигнал и показывающую амплитуду и частоту сигнала. Измерить динамический диапазон платы. Также показать расчет и значение разрядности платы. Сравнить значение напряжения сигнала, измеренное стационарным вольтметром и созданным прибором.

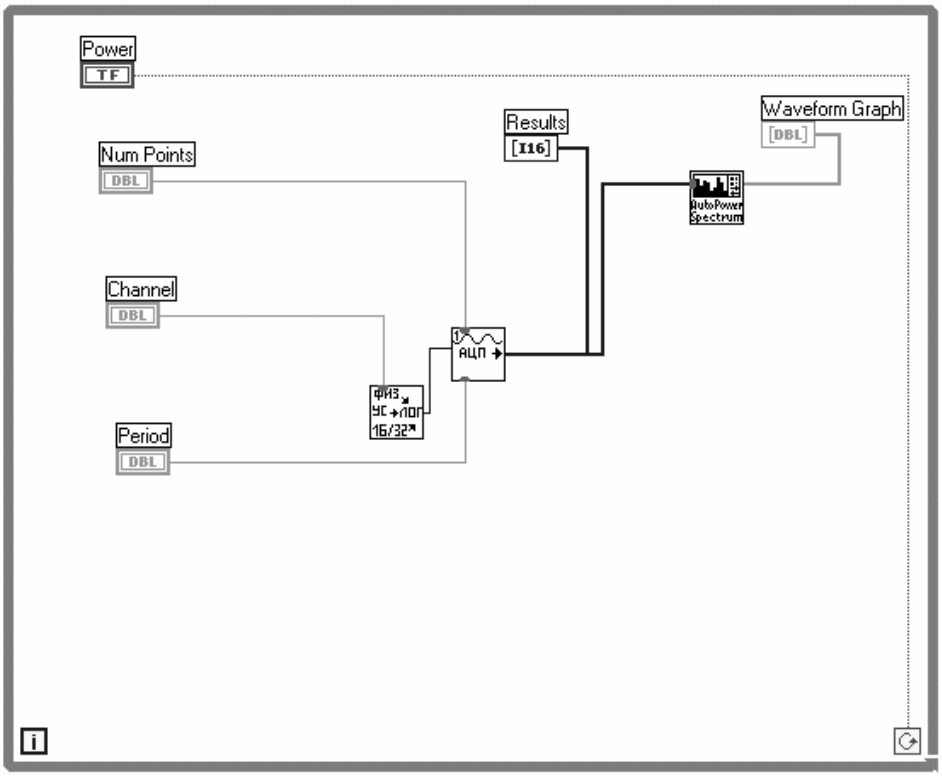
## 3.3 Измерение спектров различных сигналов

Задача

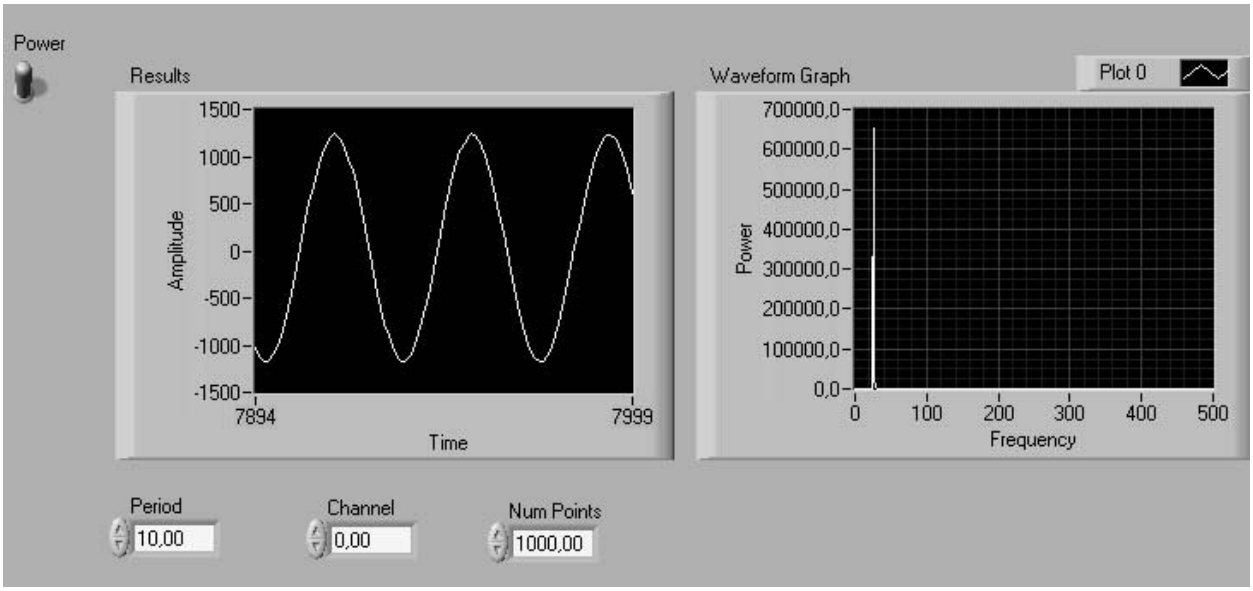
Будем подавать на плату сигналы различной формы и определим их спектр.

Реализация

Будем использовать программу, которую создали в предыдущем упражнении. Для вывода спектра на переднюю панель виртуального прибора воспользуйтесь элементом управления Waveform Graph (подпалитра Graph). Спектр находиться с помощью функции LabView Auto Power Spectrum  (Functions>> Analyse>> Signal Processing >>Frequency Domain). Создайте на передней панели еще одно графическое окно (Waveform Graph) для вывода спектра сигнала. Далее выходные данные с функции Аналоговый одноканальный ввод (AI Acquire Waveform - STREAM.vi) подайте на функцию Auto Power Spectrum и далее на график.




Результат работы программы должен выглядеть следующим образом:




Изменяйте вид сигнала и частоту и следите за изменениями спектра.

Отчетность Показать преподавателю программу, выводящую оцифрованный сигнал и его спектр. Зарисовать спектры синусоидального, треугольного и прямоугольного сигналов и показать преподавателю.

## 3.4 Определение частоты дискретизации

Задача

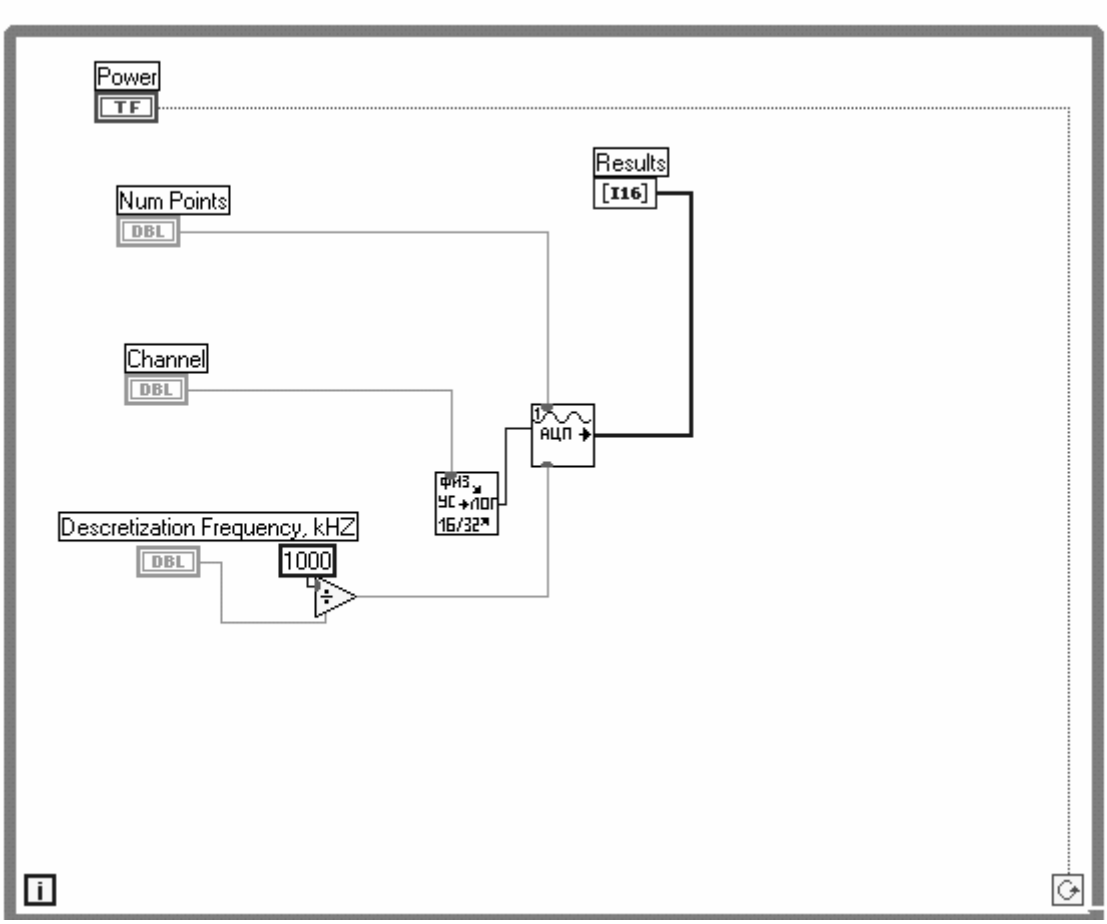
Какой частоты сигнал может быть подан на АЦП для получения хороших результатов?

Реализация

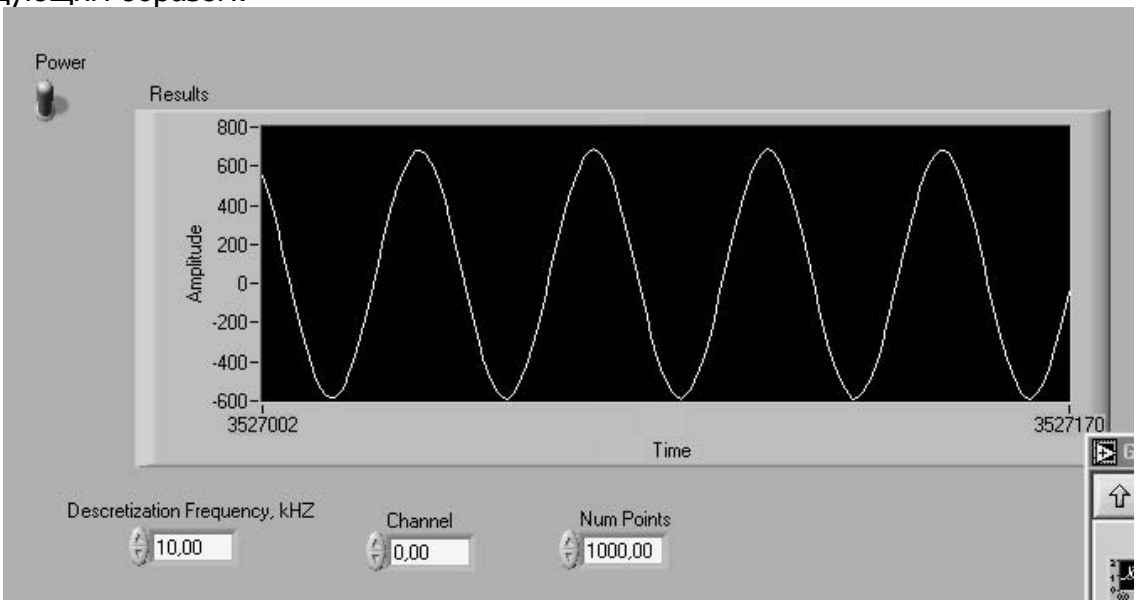
Важно решить вопрос о том, как много выборок необходимо брать в единицу времени, чтобы иметь возможность достаточно полно описать непрерывный по времени сигнал. Производя взятие выборок, мы не хотим потерять информацию, однако мы не хотим также брать выборки слишком часто. Ответ на этот вопрос дает теорема Котельникова о выборках. В этой теореме утверждается, что для восстановления (без ошибок) исходного сигнала по его выборочным значениям, взятым через равные промежутки времени, частота взятия выборок f должна более, чем вдвое, превосходить частоту $f_{signal}$ самой высокочастотной составляющей, имеющейся в непрерывном входном сигнале. Подадим на плату сигнал фиксированной частоты и будем изменять частоту взятия выборок (частота дискретизации).

**Примечание** Для изменения частоты дискретизации необходимо дать возможность ее задавать и через нее рассчитать период опроса по формуле $T = \frac{1}{f_d}$,

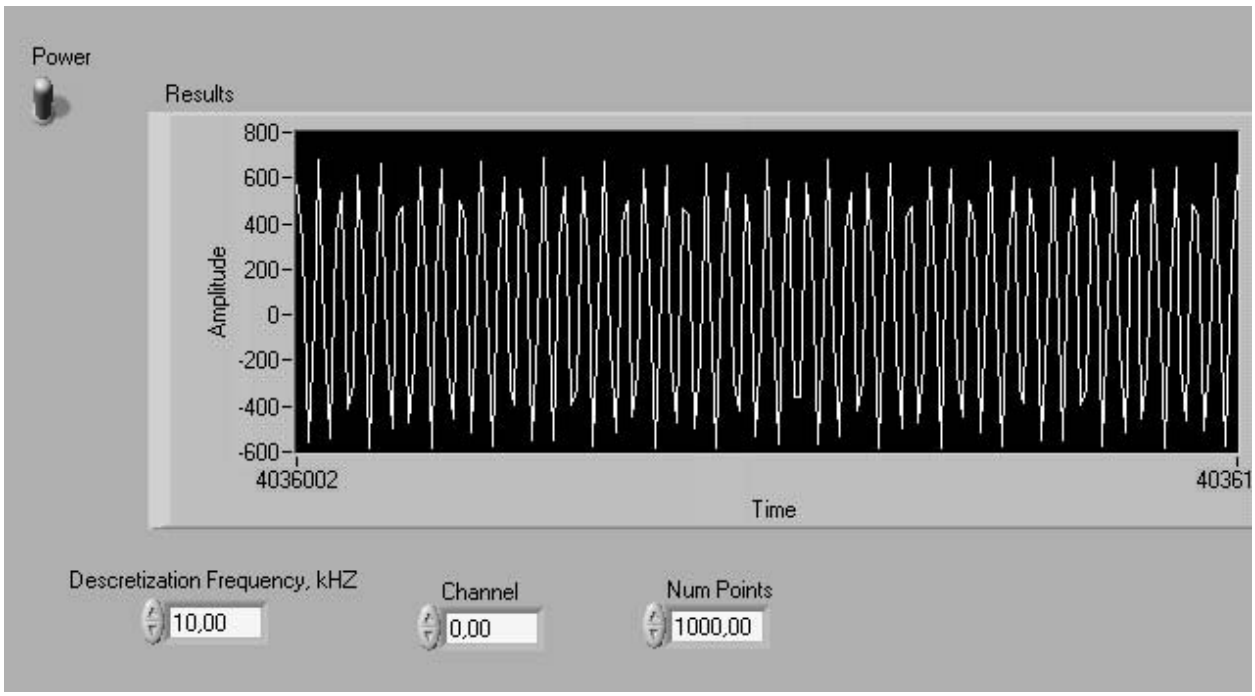
где $T$ - период опроса, а $f_d$ - частота дискретизации.




В данном случае множитель равен 1000 т.к. функция АЦП принимает период в микро секундах ($10^{-6}$), а частота задается в кГц ($10^{3}$). Установим значения параметров следующим образом:




$f_d$ =10 кГц, Chanel = 0, Num Points = 1000. Частота сигнала на генераторе – 0.2 кГц. Теперь будем увеличивать частоту на генераторе и следить за формой сигнала.




пример искаженного сигнала

**Замечание** Следует отметить что для узкополосных сигналов (полоса частот много меньше средней частоты спектра сигнала) можно выбрать частоту дискретизации значительно меньше частоты, определяемой теоремой Котельникова. Это режим – субдискретизации. ( см. [2] )

Отчетность Показать преподавателю искаженный сигнал и параметры платы и сигнала при его искажении. Убедиться в справедливости теоремы Котельникова.

## IV. Контрольные вопросы

1. Что такое АЦП
2. какие бывают платы АЦП
3. Расскажите про Параллельные  АЦП
4. Расскажите про последовательного счета
5. Что такое палитра и подпалитра в LabView
6. Как связаны плата АЦП и LabView
7. Что такое частота дискретизации?
8. Теорема Котельникова.
9. Когда можно выбрать частоту дискретизации меньше частоты Найквиста?

## V. Список литературы

1. К.Б. Клаассен , Основы измерений. Электронные методы и приборы в измерительной технике. Москва:Постмаркет,2000.
2. Ж. Макс Методы и техника обработки сигналов при физических измерениях. Мир, 1983.
3. П. Хоровиц, У. Хилл, Искусство схемотехники том2 Мир, 1993
4. Э. Оппенгейм, Применение цифровой обработки сигналов Мир, 1980
5. А.И. Азаров, В.А. Басик, И.Н. Мелешко, П.И. Монастырский, В.А. Радаева, Н.П. Феденко, В.С. Федосенко, А.С. Шибут, Т.С. Якименко Сборник задач по вычислительным методам. Москва:Физматлит,1994.
6. Бахвалов Н.С., Жидков Н. П. Практикум по программированию. Изд-во МГУ,1986.
7. Ф.А. Новиков Дискретная математика для программистов. Спб. Питер 2001.
8. Чарльз Калверт Программирование для Windows95, BINOM Publishers,1996/
9. Документация к плате АЦП Lcard-305.
10. Документация к программе LabView 6i