Министерство образования и науки Российской Федерации
Федеральное агентство по образованию
Государственное образовательное учреждение высшего профессионального образования
МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ИМ. Н.Э. БАУМАНА (МГТУ им. Н.Э. Баумана)
Факультет «Информатика и системы управления»
Факультет «Аэрокосмический»
Кафедра «Компьютерные системы и сети»

«Утверждаю»

Зав.каф. ИУ6
_____Сюзев В.В.
«___»_____ 2008 г.

Изучение принципов работы аналогово-цифровой преобразователя

Методические указания
по выполнению лабораторных работ
по курсу
«Интерфейсы периферийных устройств»

Автор: к.т.н., доцент Галямова Е.В.

Москва, 2008

Цель работы:

1. Знакомство с теорией, терминологией и основными параметрами аналого-цифрового преобразователя
2. Изучение устройства АЦП(на примере ICP DAS PCI-1002L)..

Введение

Цифровая вычислительная и управляющая техника получает все большее распространение в современной промышленности. Это связано с рядом известных преимуществ цифровых методов обработки информации (точность, помехоустойчивость, надежность, длительность хранения…), а так же с непрерывным прогрессом в технологии производства интегральных схем. Однако широкое распространение ЭВМ выдвигает такую проблему, как их связь с внешним миром, миром аналоговых величин, т.е. процессов, описываемых непрерывными функциями времени и измеряемых параметров.

Непосредственная подача таких функций в ЭВМ невозможна, так как аналоговые и цифровые сигналы имеют разную математическую и физическую форму представления, и для их совместимости необходима процедура, известная как аналого-цифровое преобразование.

Понятно, что преимущества цифровых методов обработки информации могут быть реализованы лишь в том случае, когда в процессе аналого-цифрового преобразования не вносится ограничений по точности и быстродействию.

1. Общие сведения об аналогово-цифровом преобразовании.

1.1 Математическое обоснование

Математически операция преобразования аналог-цифра представляет собой преобразование непрерывной функции S(t) в последовательность чисел $\{S(t_n)\}$, n = 0,1,2... отнесенных к фиксированным моментам времени, и может быть разделена на две вспомогательные операции:

-дискретизацию, т.е. преобразование непрерывной функции S(t) в последовательность отсчётов $\{S(t_n)\}$.

-квантование, т.е. преобразование непрерывной последовательности $\{S(t_n)\}$ в

дискретную - $\{\hat{S}(t_n)\}$.

Обе эти операции могут быть выполнены с принципиальными ограничениями по точности, рассмотрение которых приведено ниже. Влияние факторов, которые мы условно назовем технологическими, такие как, например, изменение температуры окружающей среды, параметров питающих напряжений, отклонение параметров отдельных элементов от номиналов, в данной работе не рассматриваются.

1.2 Дискретизация непрерывных сигналов.

В основе дискретизации лежит принципиальная возможность представления их в виде сумм

$$S(t) = \sum_{n=-\infty}^{+\infty} a_n f_n, \qquad (1)$$

где a_n - некоторые коэффициенты, которые характеризуют исходный сигнал в дискретные моменты времени, а f_n(t) - набор элементарных функций, с помощью которых происходит восстановление сигнала по его коэффициентам.

Наиболее распространенной формой дискретизации является равномерная, основанная на теореме отсчетов Котельникова, согласно которой в качестве коэффициентов a_n необходимо использовать мгновенные значения сигнала $S(t_n)$ в точках отсчета $t_n = n\Delta t$, а переод дискретизации Δt выбрать из условия:

$$\Delta t \le \frac{\pi}{\omega_m}, \qquad (2)$$

Предполагается, что ω_m - верхняя граничная частота спектра непрерывного сигнала. Это дает известное выражение теоремы отсчетов:

$$S(t) = \sum_{n=-\infty}^{+\infty} S(n\Delta t)\frac{\sin[\omega_m(t - n\Delta t)]}{\omega_m(t - n\Delta t)}, \qquad (3)$$

Графики функций $\varphi(t) = \frac{\sin[\omega_m(t - n\Delta t)]}{\omega_m(t - n\Delta t)}$ показаны на рис. 1. Рассматривая семейство этих функций, соответствующих различным значениям $n\Delta t$, видим, что в каждый момент времени $n\Delta t$ только одна функция равна 1, а все остальные равны нулю.

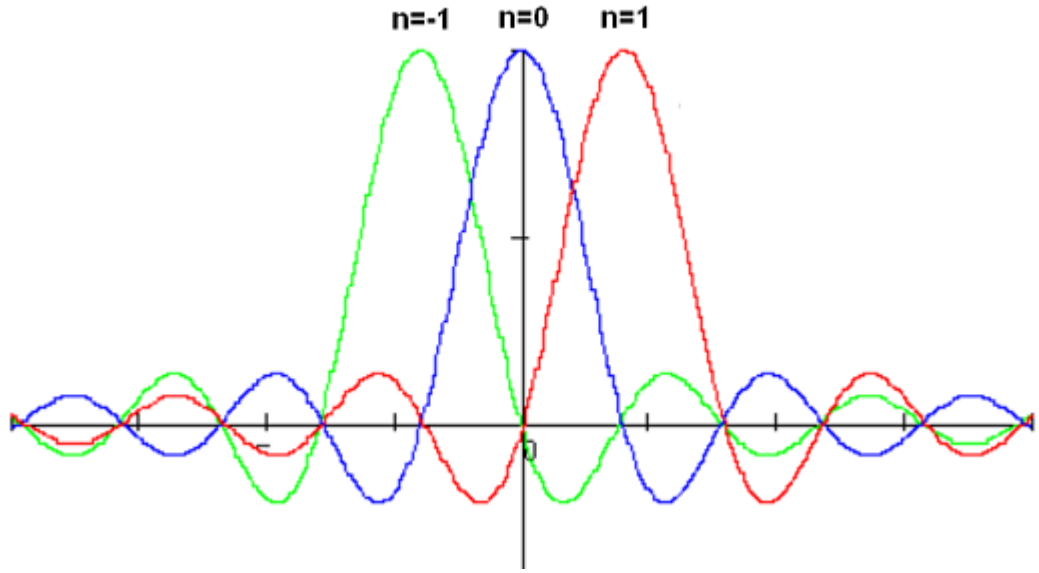
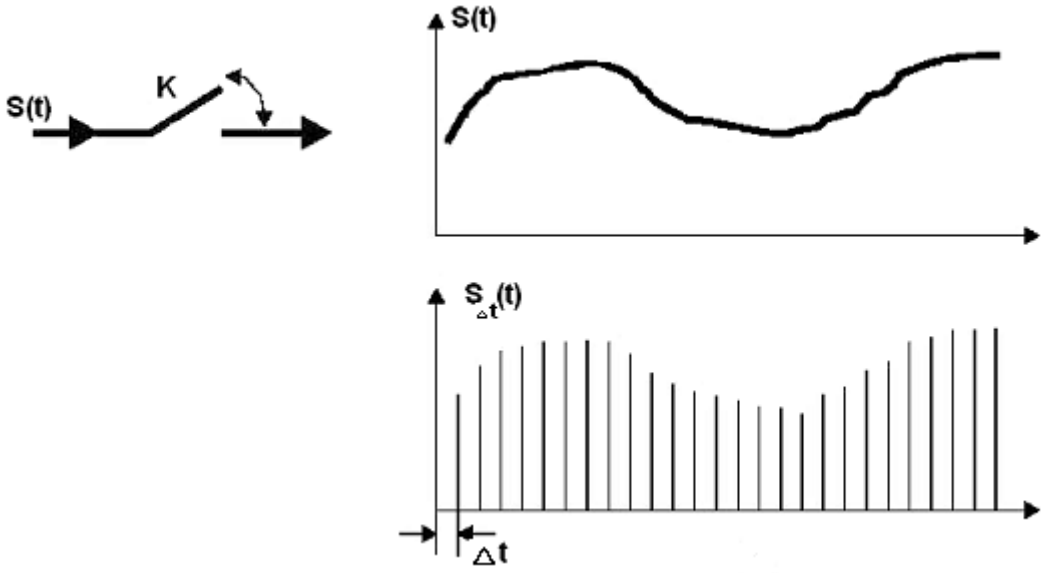


Рис.1 Графики функций $\varphi(t) = \frac{\sin[\ \omega_m (t - n\Delta t)]}{\omega_m (t - n\Delta t)}$.

Полученные «мгновенные» значения сигнала $S(n\Delta t)$ фактически реализуются кратковременными замыканиями, повторяющимися через интервалы времени Δt, механического или электрического ключа К (рис. 2а). Работа ключа поясняется с помощью рис. 2б, где $S(t)$ и $S_{\Delta t}(t)$ его входной и выходной сигналы соответственно.



а) б)

Рис.2.

Следует отметить, что спектры $S(i<\omega)$ большинства реальных сигналов стремятся к нулю асимптотически (рис. 2а), и применение к ним равномерной дискретизации приводит к

возникновению погрешности, т.е. равенство (3) осуществляется с определенной степенью точности.

Дело в том, что спектр $S_{\Delta t}(i\omega)$ дискретизированного сигнала $S_{\Delta t}(t)$ согласно теории представляет сдвинутые на $n \cdot \frac{2\pi}{T}, (n = \pm 1, \pm 2 \ldots)$ копии спектра *S(iω)* непрерывного сигнала (рис. 2а).

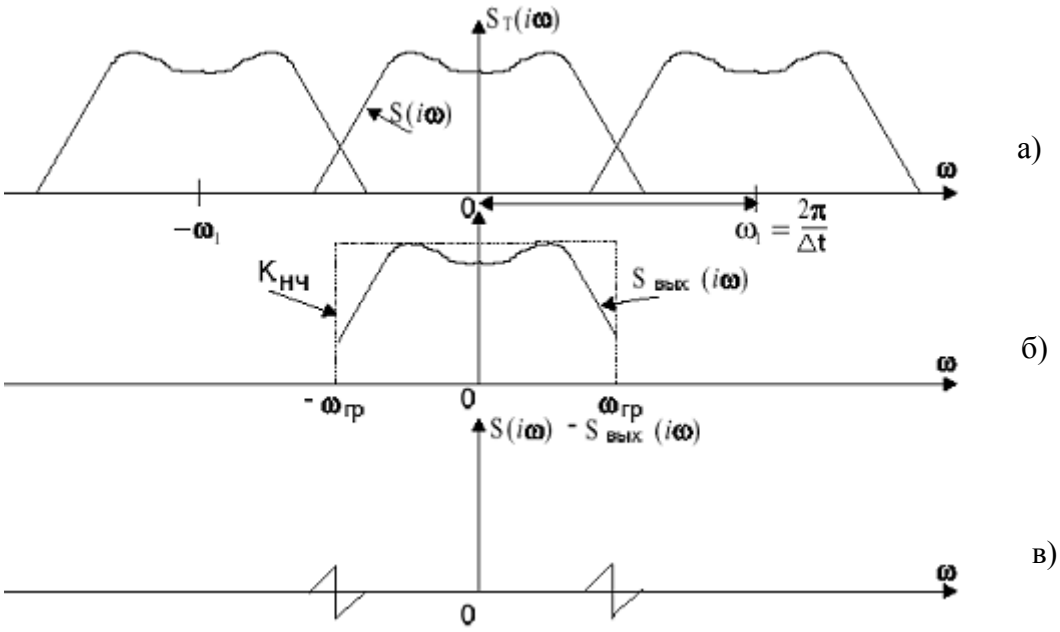


Рис. 3

При восстановлении непрерывного сигнала из отсчетов, которое осуществляется с помощью фильтра НЧ с граничной частотой $\omega_{гр} = \frac{\pi}{\Delta t}$ - его АЧХ показана на рис.3б штрихпунктирной линией, погрешность возникает по двум причинам. Во-первых, в полосе прозрачности фильтра НЧ вблизи его частоты среза $(\omega < \omega_{гр})$ из-за наложения соседнего спектра спектр выходного сигнала $S_{вых}(i\omega)$ не совпадает со спектром $S(i\omega)$ исходного сигнала. Во-вторых, в полосу прозрачности фильтра НЧ не попадает более или менее значительная часть спектра исходного сигнала с частотами $(\omega > \omega_{гр})$. В результате погрешность $S(t) - S_{вых}(t)$ имеет спектр, показанный на рис. 3в

Существует два способа уменьшения погрешности дискретизации: увеличение частоты дискретизации с тем, чтобы уменьшить область перекрывания соседних спектров, и использование перед АЦП низкочастотных фильтров с целью более четкого ограничения спектра исходного сигнала перед дискретизацией.

1.3 Погрешности, возникающие при квантовании.

Квантование по уровню заключается в замене несчетного множества возможных значений сигнала в дискретные моменты времени $n\Delta t$ (отсчетов) конечным множеством. Предполагается, что функция *S(t)* ограничена, т.е. значения ее лежат в конечном интервале. Этот интервал разбивается на конечное число дискретных уровней S_i , при чем обычно уровни располагаются равномерно по шкале функции, так что $S_i(i \cdot \Delta)$, Δ носит название **шага квантования.** Если в момент $n\Delta t$ значение отсчета лежит между уровнями $i \cdot \Delta$ и $(i+1) \cdot \Delta$, ему приписывается дискретное значение $i \cdot \Delta$. С математической точки зрения квантование есть не что иное, как округление чисел. Номер уровня *i* однозначно определяет мгновенное значение сигнала и может быть представлен двоичным числом, имеющим *r* разрядов. При этом :

$L = 2^r$ (6)

- определяет число уровней в диапазоне изменений непрерывного сигнала. При квантовании возникает специфическая погрешность, для выявления характера которой рассмотрим совместную работу АЦП и обратного преобразователя цифра-аналог (ЦАП) рис.4а

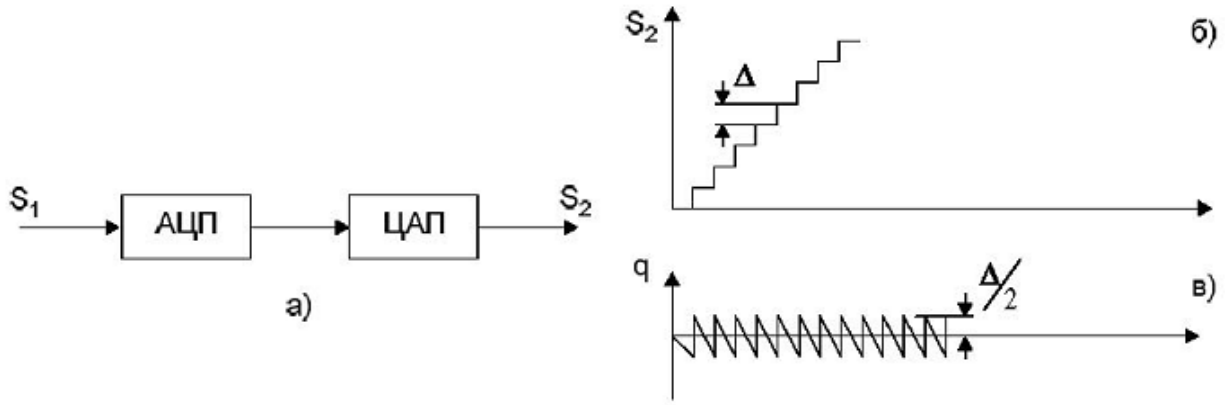


Рис.4

В ЦАП каждой комбинации нулей и единиц, т.е. двоичному числу, поступающему на вход, соответствует определенный дискретный уровень выходного напряжения. В результате при равномерном шаге квантования Δ зависимость S_2 от S_1 приобретает вид ломанной линии (рис. 4б) Разность S_2—S_1 = q следует рассматривать как **погрешность квантования.** Видно, что небольшая ошибка, по абсолютной величине не превышающая $\Delta / 2$, с возрастанием S_1 остается неизменной (рис. 4в)

Продолжим это рассмотрение для гармонического входного колебания S_1 (t)(рис. 5)

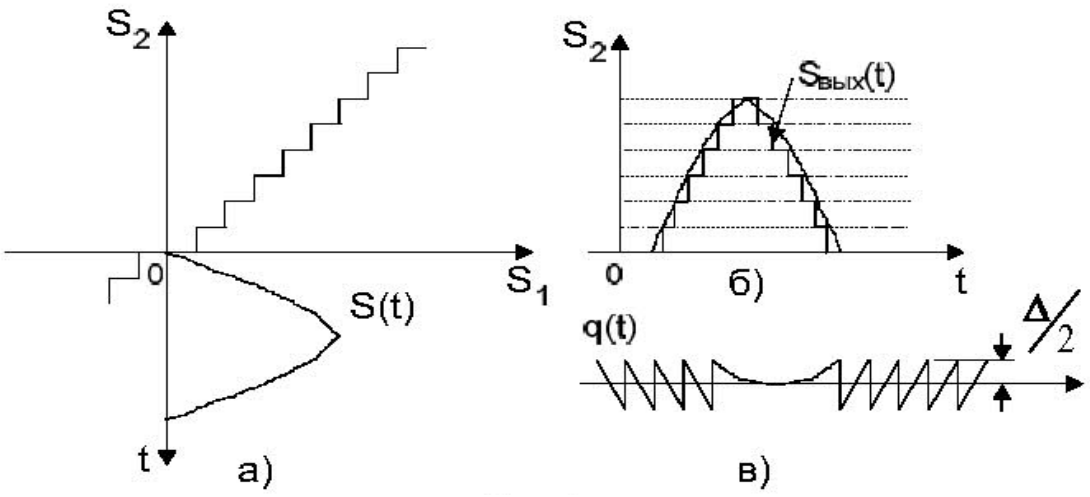


Рис.5

Выходное колебание $S_2(t)$ приобретает ступенчатую форму, отличающуюся от входного колебания, а ошибка квантования принимает вид функции

$q(t) = S_2(t)-S_1(t)$ (7)

При изменении в широких пределах амплитуды и частоты гармонического колебания изменяется только частота следования зубцов; форма их остается близкой к треугольной при неизменной амплитуде $\Delta / 2$. Функцию $q(t)$ называют **шумом квантования.** Можно вычислить среднюю мощность шума квантования. При допущении треугольной формы зубцов с амплитудой $\Delta / 2$ средняя за длительность одного зубца мощность равна $\frac{1}{3}(\Delta / 2)^2 = \Delta^2 / 12$. Так как эта величина не зависит от длительности зубца, можно считать, что средняя мощность шума квантования

$P_q = \Delta^2 / 12$ (8)

1.4 Различные виды АЦП и их характеристики

Широкое применение АЦП в различных областях науки и техники явилось предпосылкой создания разных структур АЦП, каждая из которых позволяет решить определенные задачи, предъявляемые к АЦП в каждом конкретном случае. Из всего многообразия существующих методов аналого-цифрового преобразования в интегральной технологии нашли применение в основном три:

1) метод прямого (параллельного) преобразования;

2) метод последовательного приближения (поразрядного уравновешивания);

3) метод интегрирования.

Потребность в АЦП с оптимальными параметрами или с отдельными экстремальными параметрами обусловила появление структур преобразователей, использующих комбинацию перечисленных методов. Рассмотрим структурные схемы АЦП, нашедших наибольшее распространение в интегральной технологии

В АЦП с **параллельным преобразованием** входной сигнал прикладывается одновременно ко входам всех компараторов. В каждом компараторе он сравнивается с опорным сигналом, значение которого эквивалентно определенной кодовой комбинации. Опорный сигнал снимается с узлов резистивного делителя, питаемого от источника опорного напряжения. Число возможных кодовых комбинаций (а следовательно, число компараторов) равно 2^m—1, где *m*—число разрядов АЦП. АЦП прямого преобразования обладают *самым высоким быстродействием* среди других типов АЦП, определяемым быстродействием компараторов и задержками в логическом дешифраторе. Недостатком их является необходимость в большом количестве компараторов. Так, для 8-разрядного АЦП требуется 255 компараторов. Это затрудняет реализацию многоразрядных (свыше 6—8-го разрядов) АЦП в интегральном исполнении. Кроме того, точность преобразования ограничивается точностью и стабильностью каждого компаратора и резистивного делителя. Тем не менее на основе данного принципа строят наиболее быстродействующие АЦП со временем преобразования в пределах десятков и даже единиц наносекунд, но ограниченной разрядности (не более шести разрядов).

АЦП **последовательного приближения** имеет несколько меньшее быстродействие, но существенно большую разрядность (разрешающую способность). В нем используется только один компаратор, максимальное число срабатываний которого за один цикл измерения не превышает числа разрядов преобразователя. Суть такого метода преобразования заключается в последовательном сравнении входного преобразуемого напряжения *Us* с выходным напряжением образцового ЦАП, изменяющимся по закону последовательного приближения до момента наступления их равенства (с погрешностью дискретности). Входной сигнал *Us* с помощью аналогового компаратора *К* сравнивается с выходным сигналом образцового ЦАП, который управляется в свою очередь регистром последовательного приближения *РгПП*. При запуске схемы *РгПП* устанавливается генератором *Г* в исходное состояние. При этом на выходе ЦАП формируется напряжение, соответствующее половине диапазона преобразования, что обеспечивается включением его старшего разряда 100 ... 0. Если *Us* меньше выходного напряжения ЦАП, то старший разряд выключается, включается второй по старшинству разряд (на входе ЦАП код 0100...0), что соответствует формированию на выходе ЦАП напряжения, равного половине предыдущего. В случае если *Us* превышает это напряжение, то дополнительно включается третий разряд (на входе ЦАП код 0110...0), что приводит к увеличению выходного напряжения ЦАП в 1,5

раза. При этом выходное напряжение ЦАП вновь сравнивается с напряжением *Us* и т. д. Описанная процедура повторяется *m* раз (где m—число разрядов АЦП). В итоге на выходе ЦАП формируется напряжение, отличающееся от входного преобразуемого напряжения *Us* не более чем на единицу младшего разряда ЦАП. Результат преобразования напряжения *Us* в его цифровой эквивалент — параллельный двоичный код снимается с выхода *РгПП*. Очевидно, погрешность преобразования и быстродействие такого устройства определяются в основном параметрами ЦАП (разрешающей способностью, линейностью, быстродействием) и компаратора (порогом чувствительности, быстродействием). Преимуществом рассмотренной схемы является возможность построения многоразрядных (до 12 разрядов и выше) преобразователей сравнительно высокого быстродействия (время преобразования порядка нескольких сот наносекунд). На основе метода последовательного приближения реализована и серийно выпускается ИМС 12-разрядного АЦП К572ПВ1 с временем преобразования 100 мкс.

Наиболее простыми по структуре среди **интегрирующих преобразователей** являются АЦП с преобразованием напряжения в частоту, построенные на базе интегрирующего усилителя и аналогового компаратора. Погрешность их преобразования определяется нестабильностью порога срабатывания компаратора и постоянной времени интегратора. Более высокими метрологическими характеристиками обладают АЦП, реализованные по принципу двойного интегрирования (например, ИМС, 11-разрядного АЦП К572ПВ2), поскольку при этом практически удается исключить влияние на погрешность преобразования нестабильности порога срабатывания компаратора и постоянной времени интегратора.

Анализ описанных методов преобразования и структурных схем АЦП позволяет сделать вывод, что наибольшим *быстродействием* обладают АЦП **прямого преобразования**, однако их разрядность невысока. АЦП **поразрядного уравновешивания**, обладая *средним быстродействием*, дают возможность получить достаточно *высокую разрешающую способность*. Но помехозащищенность тех и других преобразователей невысока. АЦП **интегрирующего типа**, обладая *наименьшим быстродействием*, обеспечивают *наибольшую помехозащищенность* и *точность преобразования*.

2. Описание платы АЦП ICP DAS PCI-1002L.

2.1 Общие сведения

В данной лабораторной работе предлагается ознакомиться с принципом работы АЦП на примере платы ICP DAS PCI-1002L.

ICP DAS PCI-1002L(в дальнейшем просто ICP) – это высокопроизводительная многофункциональная PCI-плата с аналогово-цифровым преобразователем и цифровыми входами и выходами. В состав IPC входят: 12-ти разрядный аналогово-цифровой преобразователь с частотой дискретизации 110кГц, 16 канальные цифровые входы и выходы, программируемый источник прерываний. Плата поддерживает механизм “Plug and Play”. Подача аналогового сигнала может осуществляется по 32 обычным аналоговым входам или по 16 дифференциальным, что можно выбрать с помощью перемычек (Более подробное описание платы, в частности положения перемычек, дано на английском языке в Приложениях 1 и 2). Плата поддерживает три программируемых метода синхронизации: программная синхронизация, внутренняя синхронизация и внешняя синхронизация. Внешняя синхронизация может быть также 3-х видов: до-импульсной, после-импульсной, и обычной внешней. Плата в непрерывном режиме позволяет передавать 2.7M слов в секунду и может даже работать на полной скорости шины PCI (33МГц).

Общая схема функционирования платы

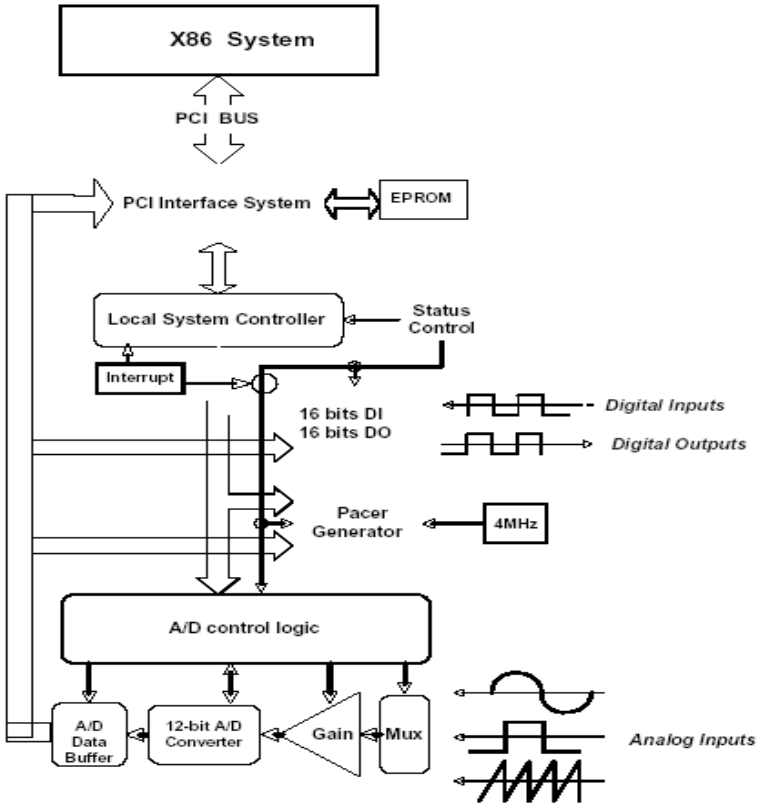


Рис.4. Общая схема функционирования платы

Методы синхронизации

- программный синхронизатор(software trigger)
- внутренний синхронизатор(pacer): 16-битный программируемый таймер/счетчик
- внешний синхронизатор(external trigger): до-импульсный синхронизатор, после-импульсный, внешний синхронизатор.

Графическая иллюстрация перечисленных выше методов представлена на рис.5

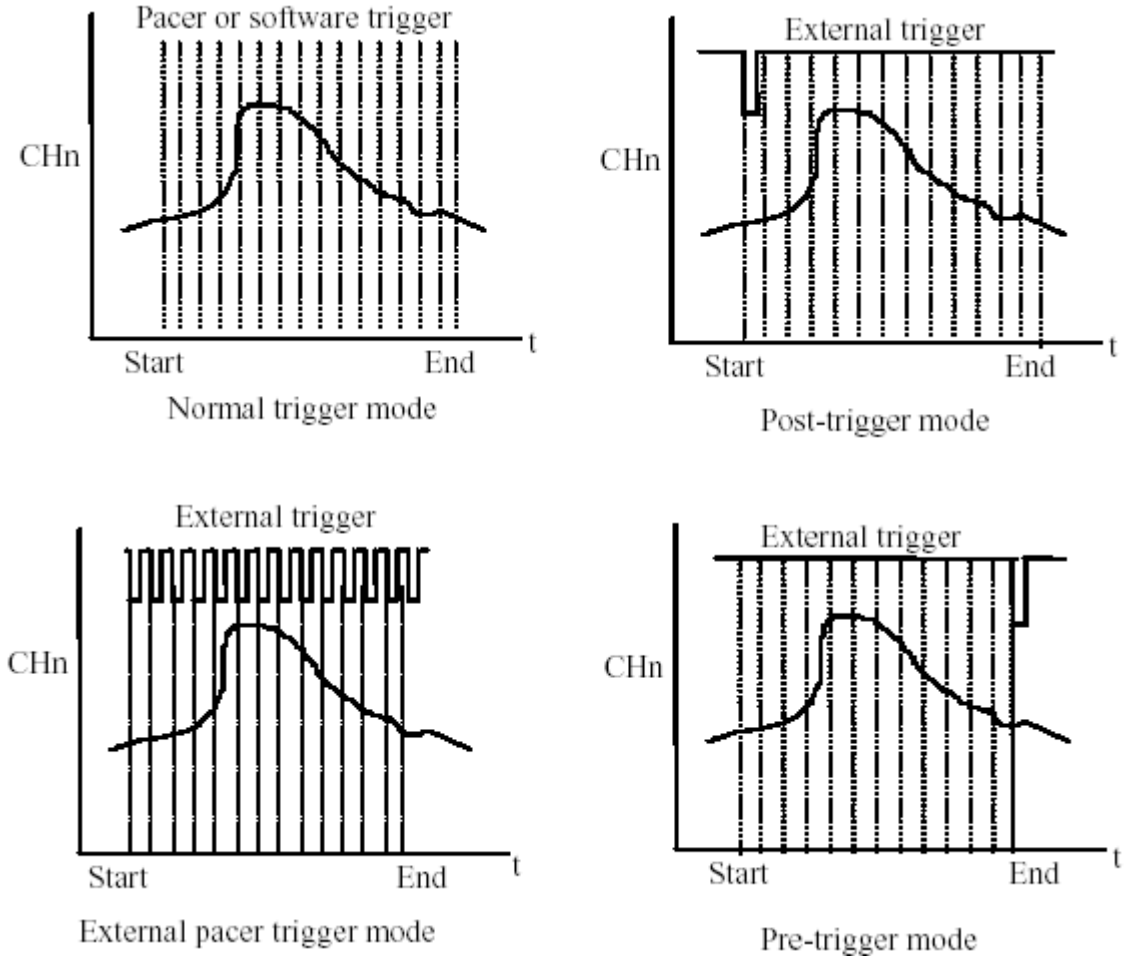


Рис.5. Методы синхронизации.

Источник прерывания

Прерывание INTA автоматически приходит от PCI-инициатора. Можно разрешить/запретить обработку прерывания с помощью регистра управления PCI или дополнительного(add-on) регистра. Источник прерывания выбирается встроенным регистром управления.

Варианты прерывания:

1. Прерывание А/Ц преобразования
2. Прерывание синхронизатора 0 (таймер 0)
3. Прерывание синхронизатора 1 (таймер 1)
4. Внешнее прерывание

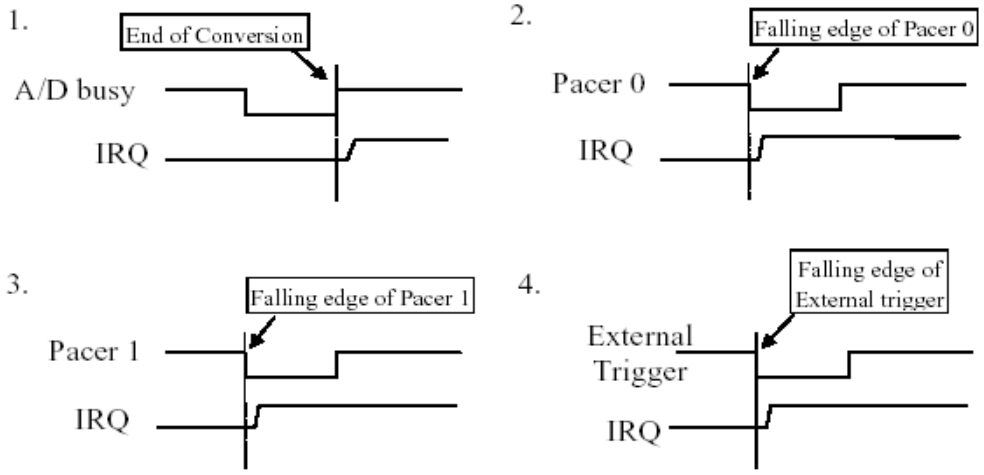


Рис. 6. Программируемый источник прерывания.

Программируемый таймер/счетчик

На плате есть три таймера 82C54-8:

Таймер 0 - предназначен для внутреннего синхронизатора и прерывания

Таймер 1 - предназначен для внешнего синхронизатора и прерывания

Таймер 2 - машинно-независимый таймер для программирования

Частота генератора 4 МГц.

Возможные способы применения платы

1) Анализ сигналов
2) Быстрое преобразование Фурье и частотный анализ
3) Анализ переходных процессов
4) Анализ речи
5) Монитор температуры
6) Анализ вибрации
7) Измерение энергии
8) Другие промышленные, лабораторные измерения и контроль

2.2 Аппаратная реализация платы.

Функциональная диаграмма

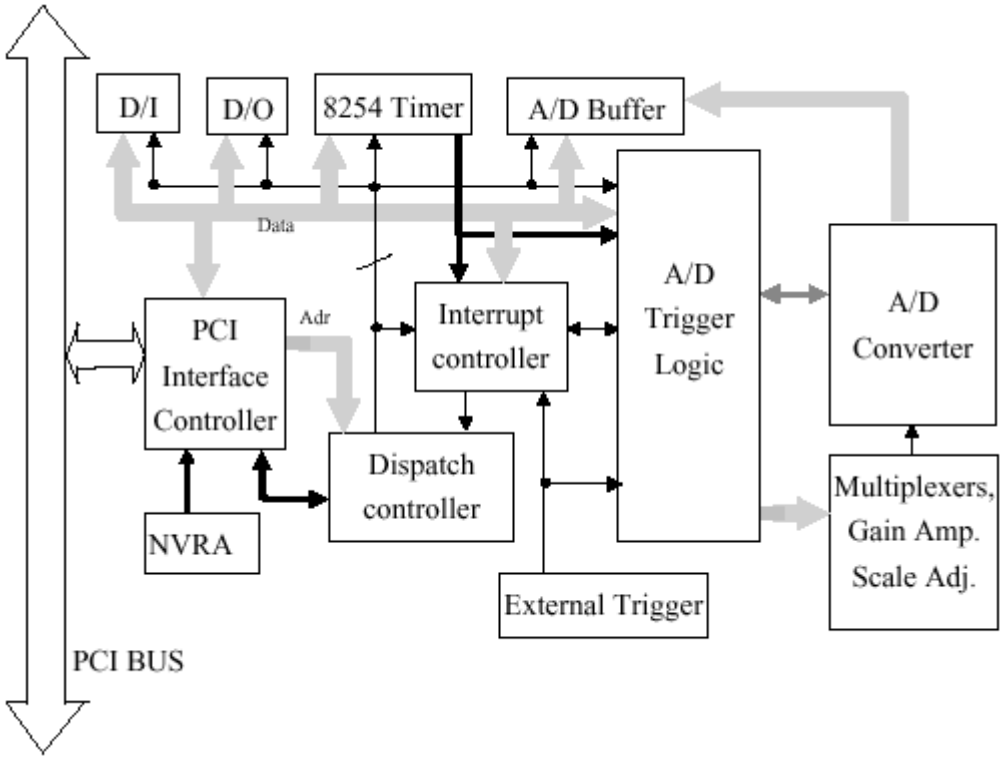


Рис.7 Функциональная диаграмма

Калибровка АЦП

1. Подать +10В на канал 0
2. Подать 0В на канал 1
3. Подать -10В на канал 2
4. Запустить DEMO6.EXE
5. Настроить регулятор VR1 пока на канале 0 не будет fff или ffe
6. Настроить регулятор VR2 пока на канале 1 не будет 800 или 801
7. Настроить регулятор VR3 пока на канале 2 не будет 000 или 001

2.3 Адреса регистров ввода-вывода

Как определить адрес порта ввода-вывода

Plug&Play Bios назначает подходящий адрес порта ввода-вывода плате на этапе начальной загрузки. Разработчик предоставил все необходимые функции для работы с платой, как например:

1. P1002_DriverInit(&wBoard) – эта функция позволяет определить сколько плат в системе.

   wBoard=1 – одна

   wBoard=2 – две

2. P1002_GetConfigAddressSpace(wBoardNo,*wBase,*wIrq,*wPLX) –эта функция позволяет пользователю сохранить состояние всех плат PCI-1002 установленных в системе, что позволяет программе впоследствии контролировать все функции PCI-1002 напрямую.

wBoardNo=0 до N – номер платы

wBase – базовый адрес управляющего слова

wIrq – номер IRQ платы

wPLX – базовый адрес PCI-интерфейса.

Пример использования этих функций см. в Приложении

Карта адресов ввода вывода

Список регистров PCI-1002 представлен в Таблице 1. Адрес каждого регистра получается просто путем добавления смещения к базовому адресу соответствующего сегмента. Более подробное описание регистров приведено ниже.

Таблица 1.

Сегмент	Смещение	Название	Доступ	Длина, бит
1	4ch	Регистр управления PCI	R/W	8/16/32
2	00h	8254 таймер1	R/W	8/16/32
	04h	8254 таймер2	R/W	8/16/32
	08h	8254 таймер3	R/W	8/16/32
	0Ch	8254 регистр управления	W	8/16/32
	10h	Регистр управления аналоговым каналом	W	8/16/32
	10h	Регистр состояния	R	8/16/32
	14h	Регистр управления	W	8/16/32
	18h	Общий регистр управления	W	8/16/32
	1Ch	А/Ц программный синхронизатор	W	8/16/32
	1Ch	Снятие прерывания	R	8/16/32
	20h	Регистр цифрового выхода	W	8/16/32
	20h	Регистр цифрового входа	R	8/16/32
	30h	Регистр дынных А/Ц преобразования	R	8/16/32

Сегмент 1

Несмотря на то, что 128 портов ввода-вывода используются встроенным контроллером PCI интерфейса, только один регистр используется в реальных приложениях. Пользователи не в коем случае не должны модифицировать другие регистры. Регистр управления прерываем PCI (4Ch) контролирует прерывание, генерируемое вашей системе. Регистр устанавливается в режим “запрета прерывания” после включения компьютера или аппаратного сигнала сброса, так что никакое прерывание не может быть сгенерировано пока этот регистр не будет активирован. Чтобы разрешить PCI-прерывание нужно записать 43h в этот регистр, чтобы запретить PCI-прерывание нужно записать 03h в этот регистр. Формат регистра управления PCI представлен в таблице 2.

Таблица 2.

Биты1-7	Бит 6	Биты 5-3	Бит2	Биты1-0
Не используется	Разрешить прерывание	Не используется	Флаг прерывания	Выбор прерывания

Биты 0 – 1 всегда должны быть установлены в 1

Бит 2 – только для чтения. “1” -показывает, что дополнительное устройство сгенерировало прерывание. “0”- что дополнительное устройство не сгенерировало прерывание.

Бит 6 –“1” разрешить PCI-прерывание, а “0”-запретить.

Замечание:

1. Так как PCI-1002 поддерживает механизм “Plug and Play”, номер прерывания будет автоматически присвоен вашей системой.
2. Если система поддерживает “Разделяемые IRQ”, то несколько периферийных устройств будут разделять один номер IRQ. Чтобы определить, что прерывание было сгенерировано именно платой нужно проверить бит 2.

Сегмент 2

Этот сегмент используется для дополнительного управления логикой. Здесь используются 64 байта портов ввода-вывода.

Регистры 8254

Микросхема 8245, программируемый таймер/счетчик, предназначен для генерации периодического А/Ц сигнала, периодического сигнала прерывания и в качестве машинно-независимого таймера. Адреса 00h, 04h, 08h и 0Ch предназначены для управления 8254.

Таймер 0 используется как синхронизатор 0. Таймер 2 используется как машинно-независимый таймер, в частности для функции задержки P1002_Delay().

Регистр цифрового ввода-вывода

Адрес 20h предназначен для портов цифрового ввода-вывода. Запись в этот порт запишет данные в регистр DO. Чтение из этого порта прочитает данные из регистра DI.

А/Ц буфер

Адрес 30h используется для А/Ц буфера. Для этого адреса доступна только операция чтения. Формат А/Ц буфера следующий:

Таблица 3.

Биты 15-12	Биты 11-0
Канал аналогового ввода	Данные А/Ц преобразования

Регистр состояния

Адрес 10h используется под регистр состояния. Чтение по этому адресу выдаст данные из регистра состояния. Формат регистра состояния отображен в таблице 4:

Таблица 4.

Биты7-6	Бит 5	Бит 4	Бит 3	Бит2	Бит1	Бит0
Контроль коэффициента усиления	8245 таймер1	8245 таймер0	8245 таймер2	Зарезервировано	Тип аналогового входа	Занятость АЦП

Бит 7-6 – Текущий коэффициент А/Ц усиления.

Бит 5 – Выход таймера 1

Бит 4 - Выход таймера 0

Бит 3 – Выход таймера 2

Бит 2 - Зарезервирован. Используется для аппаратного тестирования.

Бит 1 - Тип аналогового входа. «1»- обычный, «2»-дифференциальный.

Бит 0 – Занятость АЦП. «0»-занят, «1» -не занят.

Регистр программного синхронизатора

Запись в порт 1Ch сгенерирует сигнал А/Ц преобразования.

Замечание: Несмотря на то, что пользователь может осуществлять подачу синхроимпульса очень быстро, быстрее, чем скорость контроллера, между двумя синхроимпульсами необходимо добавить задержку как показано на рис.8.

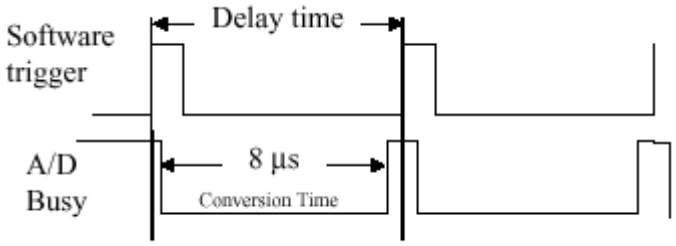


Рис. 8.

Снятие прерывания

Чтение по адресу 1Ch снимет прерывание дополнительного устройства.

Регистр выбора аналогового входа

Адрес 10h предназначен для выбора аналогового канала, а адрес 24h предназначен для доступа к регистру выбора аналогового усиления. Для выбора канала нужно в порт 20h записать значение 0-31(первые 5 бит) для обычного входа(0-15 для дифференциального-первые 4 бита). Для выбора усиления нужно в порт 24h записать значение от 0 до 3.(Рис.9)

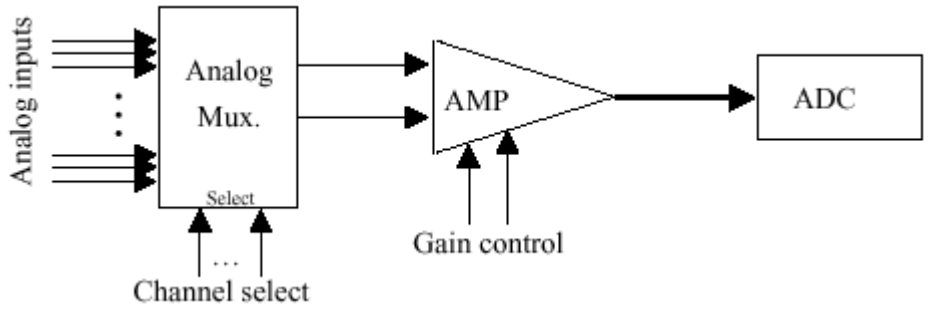


Рис.9. Управление аналоговыми входами.

Замечание: Только первые 2 бита рассматриваются при задании кода усиления. Код усиления задается в соответствие с Таблицей 5.

Таблица 5.

Код усиления	[0 0]	[0 1]	[1 0]	[1 1]
Усиление	1	2	4	8

Общий регистр управления

Общий регистр управления предназначен для контролирования сигнала прерывания от дополнительного устройства и выбора метода синхронизации А/Ц преобразования.

Таблица 6.

Биты 4-2	Бит 1-0
Регистр выбора источника прерывания	Регистр выбора режима синхронизации

Выбор источника прерывания:

Таблица 7.

Биты 4,3,2	Назначение
[0,0,0]	Нет источника прерывания, запретить все прерывания
[0,0,1]	Прерывание после А/Ц преобразования
[0,1,0]	Прерывание после обратного перехода по таймеру 0
[0,1,1]	Прерывание после обратного перехода по внешнему синхроимпульсу
[1,0,0]	Прерывание после обратного перехода по таймеру 1
Другие	Нет источника прерывания, запретить все прерывания

Замечание: Бит 3 общего регистра управления устанавливается в ноль поле аппаратного сброса.

Выбор метода синхронизации:

Таблица 8.

Биты 1,0	Назначение
[0,0]	Обычный режим синхронизации. Таймер 0(внутренний синхронизатор) или программный синхронизатор
[0,1]	Внешний синхронизатор
[1,0]	До-импульсный
[1,1]	После-импульсный

Замечание:

1. В обычном режиме, как таймер 0, так и программный синхронизатор принимаются как синхроимпульсы. В этом режиме, таймер 0 и программный синхронизатор не должны работать одновременно. То есть нельзя генерировать программный синхроимпульс, когда таймер 0 активирован.
2. В режиме внешнего синхронизатора один обратный переход внешнего синхронизатора сгенерирует одно А/Ц преобразование
3. До-импульсный режим работает под управлением таймера 1. пользователь должен сначала правильно настроить таймер 1, а затем установить до-импульсный режим. Как только до-импульсный режим был активирован, автоматически включится таймер 1 и начнется А/ Ц преобразование. Оно будет продолжаться до тех пор, пока синхронизатор не сгенерирует обратный переход. Любое изменение режима синхронизации выключит до-импульсный режим.

4. После-импульсный режим. После активации таймер 1 отключается, пока не будет получен синхроимпульс(обратный переход) от внешнего синхронизатора. Любое изменение режима синхронизации выключит после-импульсный режим.
5. Метод синхронизации А/Ц устанавливается в) после включения или аппаратного сброса.

2.4 Функциональные диаграммы работы отдельных блоков

Цифровой ввод-вывод

Плата предоставляет 16 цифровых входных каналов и 16 цифровых выходных каналов. Все уровни TTL-совместимы. (Рис 10)

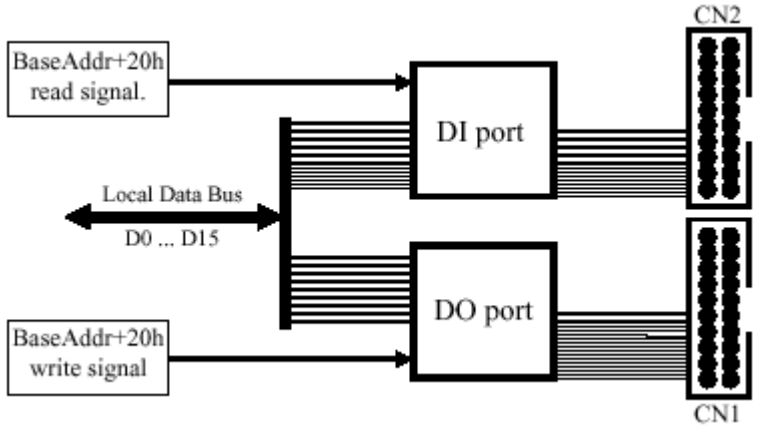


Рис.10. Функциональная диаграмма цифрового ввода-вывода

Таймер 8254

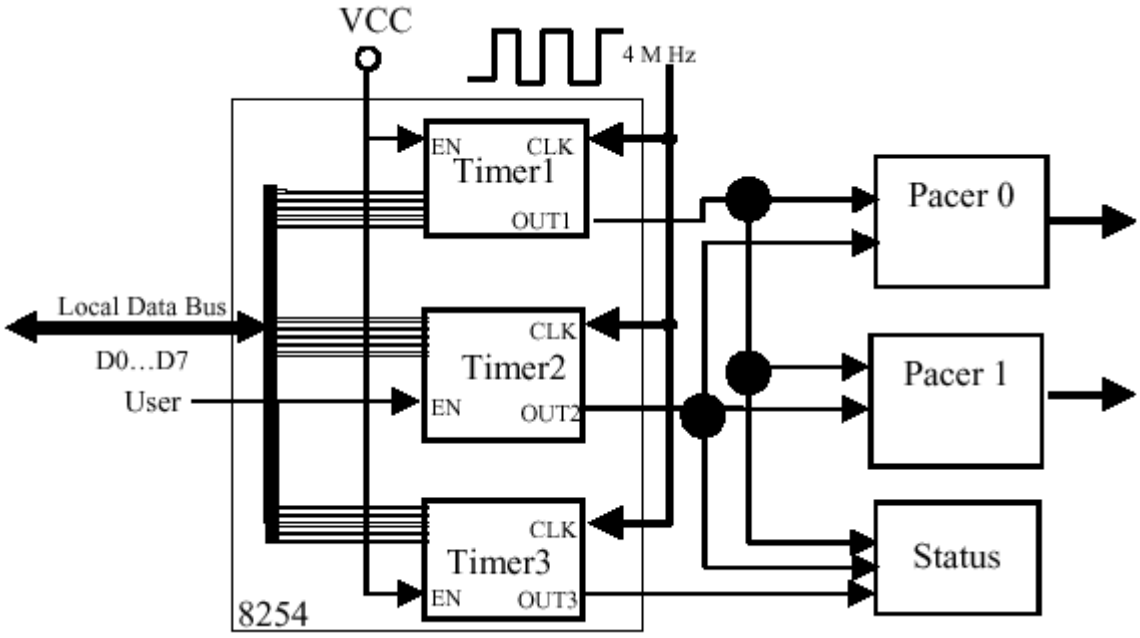


Рис.11 Функциональная диаграмма таймеров.

Синхронизатор А/Ц преобразования

Синхронизатор А/Ц преобразования управляется контроллером синхронизатора.

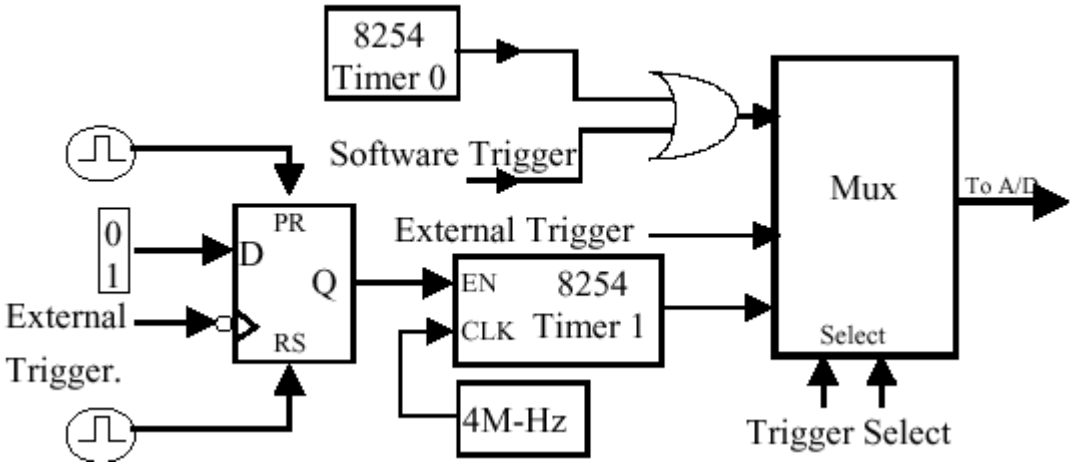


Рис.12 Контроллер синхронизатора А/Ц преобразования

Контроллер синхронизатора получает внешний синхроимпульс и осуществляет А/Ц преобразование. Чтобы контроллер распознал сигнал от внешнего синхронизатора, внешний синхроимпульс должен быть TTL-совместимым и должен иметь минимальную длительность t_{du}, чтобы избежать шума. Он должен удовлетворять требованиям, указанным в таблице 9.

Таблица 9.

Символ	Название	Минимум	Максимум
T_{du}	Длительность	40нс	-
T_{re}	Время восставновления	100нс	-

Аналогово-цифровое преобразование

Аналогово-цифровое преобразование с помощью платы может осуществляться одним из трех способов: программно, с помощью внутреннего программируемого таймера или внешнего синхроимпульса, поданного на АЦП. После аналогово-цифрового преобразования, можно передавать данные одним из двух способов: опрашивая регистр состояния(status register) и читая данные, когда необходимо или генерируя аппаратное прерывание и осуществляя его обработку. Все режимы выбираются регистром управления. Перед выполнением аналогово-цифрового преобразования, вы должны иметь представление о следующих регистрах:

- регистр данных А/Ц, база+30h, хранит результат А/Ц- преобразования
- регистр готовности к А/Ц- преобразованию, база+10h, нужен для проверки готовности к А/Ц преобразованию.
- регистр контроля коэффициента амплитуды, база+14h
- регистр мультиплексора, база+10h,
- регистр управления режимом, база+0Ch, нужен для выбора типа синхронизации и передачи данных.
- регистр управления синхронизатором, BASE+1Ch

- JP1(перемычка 1) выбор обычного или дифференциального выхода.

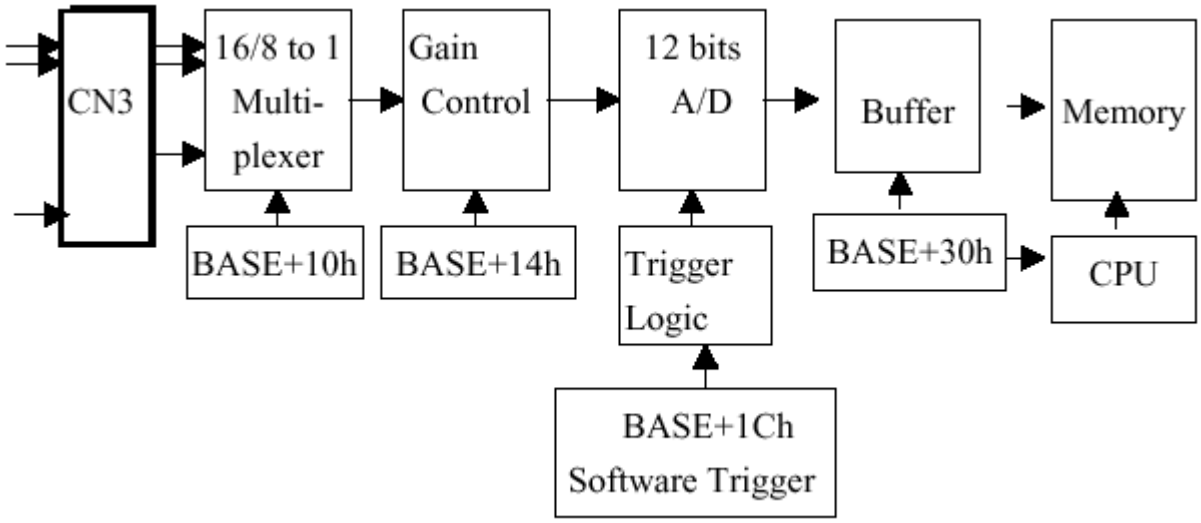


Рис. 13. Схема цифрового преобразования.

Драйвер платы поддерживает два режима передачи данных: опроса и прерывания. Пользователь может легко контролировать А/Ц преобразование в режиме опроса. Рекомендуется пользоваться драйвером при использовании прерывания.

Мультиплексор может выбирать 16 обычных или 8 дифференциальных сигналов, передавая их в модуль контроля усиления. Время установки мультиплексора зависит от импеданса источника сигнала. Пользователь должен обеспечить достаточное время задержки при переключении с одного канал на другой. Также задержка должна быть обеспечена при изменении режима усиления.

Режимы А/Ц преобразования:

1. Программный синхронизатор

   Занесение любого значения в регистр программного синхронизатора база+1Ch запустит цикл А/Ц преобразования. Этот режим очень простой, но в нем довольно сложно контролировать частоту отсчетов.
2. Режим внутренней синхронизации. Частота отсчетов внутреннего синхронизатора очень точна.
3. Режим внешней синхронизации. Когда нарастающий фронт внешнего синхросигнала поступит, произойдет А/Ц преобразование. Внешняя синхронизация поступает с 17 вывода разъема CON3.

Режимы передачи данных

1. передача по запросу

   Этот режим может использоваться при всех режимах синхронизации, только нужно отключить таймер 0 перед опросом. Чтение данных осуществляется из регистра по адресу база+30h, при READY_BIT=0 регистра база+10h
2. передача по прерыванию

Этот режим может использоваться при внутренней или внешней синхронизации.

Аппаратный сигнал посылается в систему, когда А/Ц преобразование закончилось.

При использовании этого режима рекомендуется пользоваться драйвером.

Пример. **Передача данных по опросу с программной синхронизацией*:***

1. Посылаем 00h в регистр управления режимом А/Ц (программная синхронизация + передача по запросу)
2. Посылаем номер канала в регистр управления мультиплексором
3. Посылаем коэффициент усиления в регистр управления усилением.
4. Посылаем произвольное значение в регистр управления программной синхронизации, чтобы сгенерировать программный синхросигнал
5. Сканируем бит готовности старшего байта данных А/Ц преобразования, пока он не станет равным 0
6. Читаем 12 бит А/Ц данных
7. Преобразуем эти 12 бит двоичных данных в число с плавающей точкой.

```
/* ------------------------------------------------------------ */
/* DEMO 3 : AdPolling                                           */
/* Compiler: Borland C++ 3.1, Mode Large                        */
/* Output Code: HEX code                                        */
/* ------------------------------------------------------------ */

#include "P1002.H"

WORD wBaseAddr,wIrq;
//-------------------------------------------------------
WORD P1002_Delay(WORD wDownCount)
{
  WORD h,l;
  int count;

  wDownCount &= 0x7fff;
  if (wDownCount<1) wDownCount=1;
  /* Clock in=4M --> count 4000 = 1 ms ,count 1 = 0.25 us */

  l=wDownCount&0xff;
  wDownCount=wDownCount / 256;
  h=wDownCount&0xff;

  outp(wBaseAddr+3*4,0xB0);           /* mode_0, counter_2 */
  outp(wBaseAddr+2*4,l);      /* counter_2 low byte first */
  outp(wBaseAddr+2*4,h);              /* counter_2 high byte ,0x07D0=2000 */

   outp(wBaseAddr+3*4,0x80);          /* latch counter_2 */
  l=inp(wBaseAddr+2*4);             /* delay starting two clks */
  h=inp(wBaseAddr+2*4);

  for (count=32767;count>0;count--){
            outp(wBaseAddr+12,0x80);          /* latch counter_2 */
            l=inp(wBaseAddr+8);
            h=inp(wBaseAddr+8);
            if (h>=0x80) return NoError;
  }
  return TimeOut;
```

```
}

//-------------------------------------------------------
void AdPolling(UCHAR channel, UCHAR gain, WORD delay)
    {

     outp(wBaseAddr+0x18,0);       // Select Mode 0
     outp(wBaseAddr+0x10,channel);
     outp(wBaseAddr+0x14,gain);
     P1002_Delay(delay);
     outp(wBaseAddr+0x1c,01);       // A/D software tirgger
    }

void SetupTimer(WORD wChannel, WORD wCoef)
{
  WORD   cmd;

  wChannel=wChannel&0x03;
  cmd=0x34+(wChannel<<6);
  outpw(wBaseAddr+3*4, cmd);
  outp(wBaseAddr+wChannel*4, (UCHAR)(wCoef&0xff));
  outp(wBaseAddr+wChannel*4, (UCHAR)(wCoef>>8));
}

//=========================================================
void main()
 {
 int  i,j;
 WORD wBoards,wRetVal,wPLX;
 WORD Drdy,wAdData=0;
 char c;

 clrscr();
 P1002_DriverInit(&wBoards);
 printf("\n(1) Threr are %d PCI-1002 Cards in this PC",wBoards);
 if ( wBoards==0 )
   {
   putch(0x07);  putch(0x07);  putch(0x07);
   printf("(1) There are no PCI-1002 card in this PC !!!\n"); exit(0);
   }

printf("\n(2) Show the Configuration Space of all PCI-1002:");
for(i=0; i<wBoards; i++)
   {
   P1002_GetConfigAddressSpace(i,&wBaseAddr,&wIrq,&wPLX);
   printf("\n    Card_%d: wBaseAddr=%x, wIrq=%x, wPLX=%x",i,wBaseAddr,wIrq,wPLX);
   }

 P1002_GetConfigAddressSpace(0,&wBaseAddr,&wIrq,&wPLX); /* select card_0 */
 printf("\n(3) *** Card_0, wBaseAddr=%x ***\n",wBaseAddr);

    SetupTimer(0,1);            // AdPolling have to disable timer 0
    AdPolling(0,0,23);          // channel=0, gain=+/-10, delay=23us

for(i=0;i<10;i++)
   {
    outp(wBaseAddr+0x1c,01);    // A/D software tirgger
    while(1)
              {
               if( ((inpw(wBaseAddr+0x10))&0x01)==1) // check A/D busy?
               break;
              }
     wAdData=((inpw(wBaseAddr+0x30))&0x0fff);
    printf("\nRang:+/-10V, Counter %d ,ADC channel 0 value: 0x%xH",i,wAdData);
  }
```

```
P1002_DriverClose();
}
```

3. Состав лабораторного стенда

1. Генератор специальных сигналов
2. Осциллограф
3. Персональный компьютер
4. Плата АЦП ICP DAS PCI-1002
5. Коммутатор входов-выходов

5. Порядок выполнения работы

1. Ознакомиться с описанием платы. Описание находится в Приложении 1 и в каталоге “C:\DAQPro\PCI1002 Win95\Manual”

   1002HW.pdf – описание платы, ее регистров и принципов функционирования.

   1002Soft.pdf – описание библиотечных функций
2. Подать сигнал с частотой 10 кГц с генератора на осциллограф, для снятия образцовой картинки.
3. Используя коммутатор входов-выходов подключить генератор к плате, как показано на рис. 4.

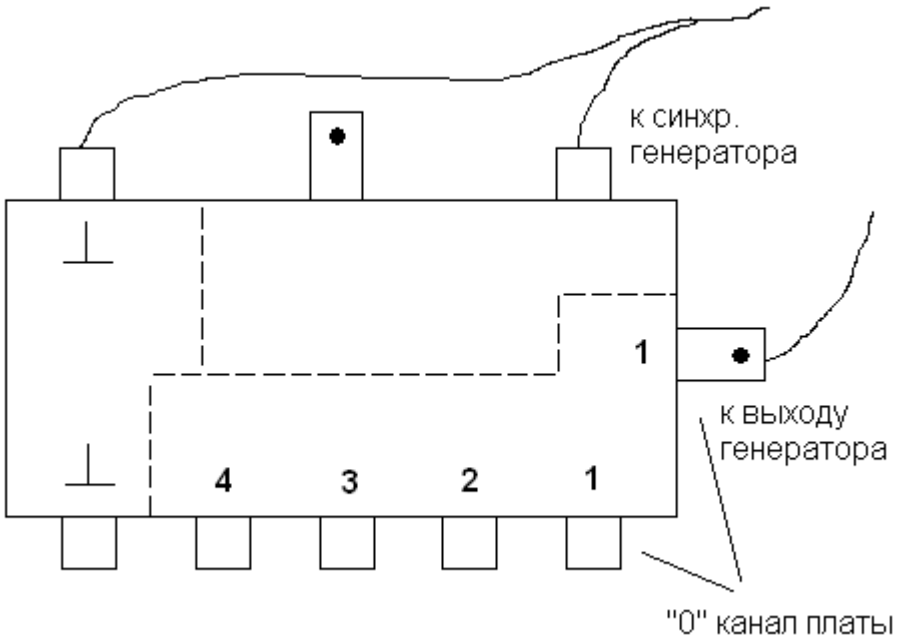


Рис. 14 Схема подключения генератора к коммутатору входов-выходов.

4. Меняя частоту входного сигнала от 100 Гц до 200 кГц (амплитуда сигнала остается постоянной), зарегистрировать (снять показания) мгновенных значений в точках отсчета – то есть, входной сигнал в цифровом виде, после преобразования в АЦП. Результаты

сохранить в отдельном файле. Количество отчетов и частот из диапазона(100 и 200кГц) может меняться (50, 100, 200, 300…) для разных вариантов заданий.

5. Для постоянного числа отчетов (например, 100) построить график выходного сигнала.

6. На том же графике построить идеальную кривую $y = A \cdot \sin(\omega t)$.

7. Составить таблицу (см. образец). Внести в таблицу цифру, соответствующую количеству периодов в зарегистрированном сигнале.

Таблица

№	f	Количество точек для построения	Кол-во зарегистрированных периодов	Кол-во точек на период
1				
2				
…				
N				

8. Вычислить среднеквадратическое отклонение зарегистрированной на выходе АЦП амплитуды сигнала от идеальной амплитуды, которую можно получить по формуле:

$\sigma = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(y_i - \bar{y}_i)^2}$, где y_i -полученное решение, $\bar{y}_i$ -модельное решение.

9. Сделать вывод о том, как среднеквадратическое отклонение соотносится с теоретически возможной ошибкой АЦП (1\2 шага квантования). Как меняется погрешность АЦП при изменении частоты входного сигнала?

6. Контрольные вопросы.

1. Сколько уровней квантования поддерживает данный АЦП?
2. Чем техника опроса отличается от техники передачи данных по прерыванию?
3. Какая максимальная погрешность у данного АЦП?
4. Сформулировать теорему Котельникова и пояснить ее смысл.
5. Какие виды АЦП вы знаете и в чем их преимущества, недостатки?

7. Список рекомендуемой литературы:

1. У. Титце. К. Шенк. «Полупроводниковая схемотехника», М. Мир, 1982

2. Авдеев В.А., Гузик В.Ф «Компьютеры: шины контроллеры, периферийные устройства», М. Радио и связь, 2001

3. И.С. Гоноровский «Радиотехнические цепи и сигналы»,М. Сов. Радио,1977

Приложение 1. Экспериментальные данные и результаты экспериментов

В соответствие с заданием отсчеты были получены для разных частот

100Гц,500Гц,1кГц,5кГц,10кГц

При проведении эксперимента для снятия отсчетов использовалась демонстрационная программа, исправленная в соответствии с заданием:

```
/*****************************************************************************
//*     DEMO1 program for one P100X card in the PC system.                  *
//*     Please turn the resolution of your monitor at least 1024*768.       *
//*****************************************************************************
//* First Card: some P100X function call demo.                              *
//* For the proper operation the P100X, the following functions must be used.*
//*    P100X_DriverInit();     <--   initial the driver                     *
//*    P100X_DriverClose();    <--   close the driver                       *
//*****************************************************************************

unit Unit1;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  StdCtrls, ComCtrls, ExtCtrls, math ;

Const gDataNo = 100;
type PBuf = Array [0.. gDataNo] of Single ;

type
  TForm1 = class(TForm)
    Button1: TButton;
    Button2: TButton;
    Timer1: TTimer;
    Label3: TLabel;
    Label4: TLabel;
    Edit3: TEdit;
    Edit4: TEdit;
    GroupBox1: TGroupBox;
    Image1: TImage;
    YEnd: TEdit;
    YStart: TEdit;
    GroupBox2: TGroupBox;
    Label1: TLabel;
    Label2: TLabel;
    cbCardType: TComboBox;
    cbSingDiff: TComboBox;
    GroupBox3: TGroupBox;
    Label5: TLabel;
    Label6: TLabel;
    Label7: TLabel;
    Label8: TLabel;
    Edit1: TEdit;
    Edit2: TEdit;
    UpDown1: TUpDown;
    InRange: TComboBox;
    cbChNo: TComboBox;
    lDataNo: TLabel;
    Label12: TLabel;
    procedure Button1Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
    procedure Timer1Timer(Sender: TObject);
```

```
    procedure ClearCanvas( Cv : TCanvas; cc:TColor );
    function  DrawWave( YStart , YEnd : Single ; Buf :PBuf ; DataNo : Integer ; im : TImage ):integer ;
    procedure InRangeChange(Sender: TObject);
    procedure YEndExit(Sender: TObject);
    procedure YStartExit(Sender: TObject);
    procedure cbCardTypeChange(Sender: TObject);
    procedure cbSingDiffChange(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form1: TForm1;
  myfile:textfile;
  flag:integer;
implementation

uses P100X, P100Xu;

{$R *.DFM}

Var
   wTotalBoard, wInitialCode : Word;
   wSet  : Word;
   Ad_Ch , CfgCode : Word;
   WaveData : PBuf ;

procedure TForm1.Button1Click(Sender: TObject);
begin
     if ( StrToInt(Edit2.Text) > StrToInt(Edit1.Text) -1)
     or ( StrToInt(Edit2.Text) < 0 ) then
     Begin
        ShowMessage('Invalid board number, Pls retry!!');
        exit;
     end;

    If Button1.Caption = 'Active' Then
    begin
        If P100X_ActiveBoard(StrToInt(Edit2.Text)) <> NoError Then
        begin
            ShowMessage('Can not Active the Board.');
            Exit;
        End;
        P100X_SetChannelConfig(AD_Ch, CfgCode);
        Button1.Caption   := 'Stop';
        Timer1.Enabled    := True;
        Button2.Enabled   := False;
        YStart.Enabled    := False;
        YEnd.Enabled      := False;
    end
    Else
    begin
        Button1.Caption   := 'Active';
        Timer1.Enabled    := False;
        Button2.Enabled   := True;
        YStart.Enabled    := True;
        YEnd.Enabled      := True;
    End;
end;

procedure TForm1.Button2Click(Sender: TObject);
begin
     Close;
```

```
end;

procedure TForm1.FormCreate(Sender: TObject);
begin
   //*************************************************************************
   //* NOTICE: call P100X_DriverInit() to initialize the driver.        *
   //* Initial the device driver, and return the board number in the PC *
   //*************************************************************************
    assignfile(myfile,'result.txt');
    rewrite(myfile);
    flag:=0;
   wInitialCode := P100X_DriverInit(wTotalBoard);
   Button1.Caption := 'Active';
   If wInitialCode <> NoError Then
   begin
      ShowMessage('No P100X card in this system !!!');
      Button1.Enabled := False;
   end
   Else
      Button1.Enabled := True;

   Edit2.Text := '0';
   Edit1.Text := IntToStr(wTotalBoard);
   Updown1.Max := wTotalBoard - 1;
   Updown1.Min := 0;
   wSet := 0;

   //RadioButton2.Check := True ;
   cbCardType.ItemIndex := 0;
   cbCardTypeChange( Sender );
   cbSingDiff.ItemIndex := 0;
   cbSingDiffChange( Sender );

   InRange.ItemIndex    := 0;
   YStart.Text          := '-10';
   YEnd.Text            := '10' ;
   AD_Ch                := 0;

   ClearCanvas( image1.Canvas , clYellow ) ;
end;

procedure TForm1.FormClose(Sender: TObject; var Action: TCloseAction);
begin
    Timer1.Enabled := False;
    P100X_DriverClose;
end;

procedure TForm1.Timer1Timer(Sender: TObject);
Var
  iVal    : word;
  fVal    : Single ;
  i:integer;
begin
   P100X_Polling(iVal);
   Edit3.Text := IntToHex( iVal , 4 );

   P100X_AdPolling(fVal);
   Edit4.Text := FloatToStrF(fVal, ffNumber , 8 , 4 );

   If P100X_AdsPolling(@WaveData[0], gDataNo) <> NoError Then
   begin
      ShowMessage('AD Polling Error!' );
      Button2Click(Sender);
      Exit;
   End;
```

```
   DrawWave(StrToFloat(YStart.Text),StrToFloat(YEnd.Text),WaveData, gDataNo , Image1);
end;

procedure TForm1.InRangeChange(Sender: TObject);
Var
  G : Single ;

begin

  CfgCode := InRange.ItemIndex ;  //While Input Range is changed, reset the Gain Code
  If cbCardType.ItemIndex = 1 Then   //Hi-Gain
  begin
    G := 10 / Power(10,InRange.ItemIndex);
    CfgCode := CfgCode + 16 ;
  end
  Else
    G := 10 / Power(2,InRange.ItemIndex);

  YEnd.Text   := FloatToStr(G) ;
  YStart.Text := FloatToStr(G * -1) ;

  YStartExit(Sender);
  YEndExit(Sender);
end;

procedure TForm1.YEndExit(Sender: TObject);
begin
    if StrToFloat(YStart.Text) >= StrToFloat(YEnd.Text) then
       YEnd.Text := FloatToStr(StrToFloat(YStart.Text) + 5) ;
end;

procedure TForm1.YStartExit(Sender: TObject);
begin
    if StrToFloat(YStart.Text) >= StrToFloat(YEnd.Text) then
       YStart.Text := FloatToStr(StrToFloat(YEnd.Text) - 5) ;
end;

//*************************************************************
//This function will Clear the specify Canvas.
procedure TForm1.ClearCanvas( Cv : TCanvas ; cc : TColor );
Begin
    with Cv do
    begin
        Brush.Style := bsSolid;
        Brush.Color := cc;
        FillRect(ClipRect);
    end;
end;

{*************************************************************
 Function : DrawWave
 Description : this function will compute the YStart , YEnd
             , DataNo , Image's Width , Image's Height
             to draw the appropriate wave graphicals.
 Parameter : YStart --> the Start Value (Base)
           YEnd   --> the End Value   (Max)
           Buf    --> Array [ 0 .. DataNo ] of Single
                    Store the AD value
           DataNo --> How many Data will be process
           im     --> What IMAGE control will be draw on
 Return : 0 --> No error occur
         ? --> Error
 *************************************************************}
function TForm1.DrawWave( YStart , YEnd : Single ; Buf : PBuf ; DataNo : Integer ; im : TImage ):integer;
Var
```

```
  DeltaX , DeltaY : Single ;
  px , py , i : integer ;

Begin
    ClearCanvas( im.Canvas , clYellow ) ;    //Clear Canvas before checking error
    lDataNo.Caption := IntToStr( DataNo );

    if YStart >= YEnd then  //if any error occur, user will see a white window
    begin
      result := 1;         //Error : YEnd must great then YStart
      exit;
    end;

    DeltaX := im.Width / (DataNo ) ;                //make some margin
    DeltaY := im.Height / ( YEnd - YStart ) ;
    With im.Canvas.Pen do
    begin
        Style := psSolid ;
        color := clRed   ;
        mode  := pmCopy  ;
    end ;
    px := 0;                    // close to 0 * DeltaX
    py := Round(( YEnd - Buf[0] ) * DeltaY );
    im.Canvas.MoveTo(px , py);
    if flag=0 then begin append(myfile); end;
    for i := 1 to DataNo -1 do
    begin
    //Showmessage()
    if (flag=0) then writeln(myfile,Buf[i]);
        px := Round(i * DeltaX) ;
        py := Round(( YEnd - Buf[i] ) * DeltaY );
        if px = 0 then
          im.Canvas.MoveTo( px,py )
        else
          im.Canvas.LineTo( px , py );
    end;
   if flag=0 then begin closefile(myfile); flag:=1; end;
    result := 0;
End;



procedure TForm1.cbCardTypeChange(Sender: TObject);
begin
   //While Hi/Lo Gain Selection is changed, reset contents of the Input Range (ComboBox).
   InRange.Clear();
   If cbCardType.ItemIndex = 0 Then   //Low Gain
   begin
      InRange.Items.Add('  -10 ~ 10  ');
      InRange.Items.Add('   -5 ~ 5   ');
      InRange.Items.Add(' -2.5 ~ 2.5 ');
      InRange.Items.Add('-1.25 ~ 1.25');
   end
   Else
   begin
      InRange.Items.Add('  -10 ~ 10  ');
      InRange.Items.Add('   -1 ~ 1   ');
      InRange.Items.Add(' -0.1 ~ 0.1 ');
      InRange.Items.Add('-0.01 ~ 0.01');
   End;

   InRange.ItemIndex := 0 ;        //Reset the Input Range

end;
```

```
procedure TForm1.cbSingDiffChange(Sender: TObject);
var
   i , ttl: integer;
begin
     cbChNo.Clear();

     // Single-End 32 Channels, Differential 16 Channels
     ttl := (2 - cbSingDiff.ItemIndex) * 16 ;

     for i := 0 to ttl -1 do
        cbChNo.Items.Add( IntToStr(i) );

     cbChNo.ItemIndex := 0;
end;

end.
```

Построения графиков по заданным точкам осуществлялось с помощью программы:

```
unit otu;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  ExtCtrls, StdCtrls, Math;

type
  TForm1 = class(TForm)
    Image1: TImage;
    Edit1: TEdit;
    Button1: TButton;
    Label1: TLabel;
    Edit2: TEdit;
    Button2: TButton;
    Label2: TLabel;
    Button3: TButton;
    Button4: TButton;
    Edit3: TEdit;
    Label3: TLabel;
    Edit4: TEdit;
    Label4: TLabel;
    procedure FormActivate(Sender: TObject);
    procedure Image1MouseMove(Sender: TObject; Shift: TShiftState; X,
      Y: Integer);
    procedure FormKeyDown(Sender: TObject; var Key: Word;
      Shift: TShiftState);
    procedure Button1Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
    procedure Draw;
    procedure Clear;
    procedure Button3Click(Sender: TObject);
    procedure Button4Click(Sender: TObject);

  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form1: TForm1;

implementation
```

```
{$R *.DFM}

var i,j:byte;
    y,x,ym,xm:array of extended;
    stepx,stepy:currency;
    ch,dx,razmm,razm:integer;

procedure TForm1.FormActivate(Sender: TObject);
var f,g,o,l:textfile;
    bufy,bufx:string;
begin
 clear();
 button1.Click();
 button2.Click();

end;

procedure TForm1.Image1MouseMove(Sender: TObject; Shift: TShiftState; X,
  Y: Integer);
begin
 //image1.canvas.textout(0,0,'x='+inttostr(x)+' '+'y='+inttostr(y));
end;

procedure TForm1.FormKeyDown(Sender: TObject; var Key: Word;
  Shift: TShiftState);
begin
 if key=27 then
 begin
   close;
 end;
end;

procedure TForm1.Button2Click(Sender: TObject);
var st:string;
begin
 clear();
end;

procedure TForm1.Button1Click(Sender: TObject);
var st:string;
begin
  clear();
end;

procedure Tform1.draw;
var xx,yy1,yy2,dx:longint;
begin

 dx:=0;
 xx:=dx+50+round(x[0]/stepx);
 yy1:=350-round(y[0]/stepy);
 image1.Canvas.moveto(xx,yy1);
 //Showmessage(inttostr(xx)+' '+inttostr(yy));
 image1.Canvas.Pen.Width:=2;
 for i:=1 to razm-1 do
  begin
   xx:=dx+50+round(x[i]/stepx);
   yy1:=350-round(y[i]/stepy);
   //yy2:=350-round(y[i-1]/stepy);
 //Showmessage(inttostr(xx)+' '+inttostr(yy));
 //  image1.Canvas.LineTo(xx,yy2);
   image1.Canvas.LineTo(xx,yy1);
  end;
```

```
 xx:=dx+50+round(xm[0]/stepx);
 yy1:=350-round(ym[0]/stepy);
 image1.Canvas.moveto(xx,yy1);
 image1.Canvas.Pen.Width:=1;
 image1.Canvas.Brush.Color:=clred;
 image1.Canvas.Pen.Color:=clred;
 //Showmessage(inttostr(xx)+' '+inttostr(yy));
 for i:=1 to razmm-1 do
  begin
   xx:=dx+50+round(xm[i]/stepx);
   yy1:=350-round(ym[i]/stepy);
 //Showmessage(inttostr(xx)+' '+inttostr(yy));

   image1.Canvas.LineTo(xx,yy1);
  end;


end;

procedure TForm1.Button3Click(Sender: TObject);
var f,g,o,l:textfile;
    st,st1,st2:string;
    j,ch,sign:integer;
    val,buf,delta,a:extended;
begin
clear();
  razmm:=strtoint(Edit3.text);
  razm:=strtoint(Edit4.text);
 Setlength(x,razm);
 Setlength(y,razm);
 Setlength(xm,razmm);
 Setlength(ym,razmm);
 assignfile(f,'yr.txt');
 reset(f);
 for i:=0 to razm-1 do
  begin
   readln(f,st);
   st1:='';  st2:='';
   for j:=1 to 17 do st1:=st1+st[j];
   for j:=20 to 23 do st2:=st2+st[j];
   ch:=ansipos('.',st1);
   if ch<>0 then
   st1[ch]:=',';
   if st[19]='+' then sign:=1 else sign:=-1;
   buf:=Power(10,sign*strtoint(st2));
   //Showmessage(currtostr(buf));
   val:=strtocurr(st1)*buf;
  // Showmessage(currtostr(val));
   y[i]:=val;
  end;
  closefile(f);
  A:=3.21;
  xm[0]:=0;
  delta:=0.00909;
   for i:=1 to razmm-1 do
   begin
   xm[i]:=xm[i-1]+delta;
   ym[i]:=A*sin(xm[i]*62.831853);
   end;
 x:=xm;
 draw;
end;

procedure TForm1.Button4Click(Sender: TObject);
```

```
begin
 clear;
end;

procedure Tform1.Clear();
var st:string;
begin
 image1.canvas.Pen.Color:=clwhite;
 image1.Canvas.Brush.color:=clwhite;
 image1.canvas.Brush.style:=bssolid;
 image1.Canvas.Rectangle(0,0,image1.width,image1.height);
 image1.canvas.Pen.Color:=clblack;
 dx:=0;
 with image1.canvas do
  begin
   moveto(50,350);
   lineto(750,350);
   moveto(dx+50,650);
   lineto(dx+50,50);
  end;
  for i:=0 to 29 do
   begin
    image1.canvas.moveto(dx+48,350-i*10);
    image1.canvas.lineto(dx+52,350-i*10);
   end;
  for i:=0 to 29 do
   begin
    image1.canvas.moveto(dx+48,350+i*10);
    image1.canvas.lineto(dx+52,350+i*10);
   end;
   for i:=0 to 70 do
   begin
    image1.canvas.moveto(50+i*10,348);
    image1.canvas.lineto(50+i*10,352);
   end;
  stepx:=strtocurr(edit2.text);
  stepy:=strtocurr(edit1.text);

   for i:=1 to 6 do
   begin
    st:=currtostr(i*stepy*50);
    image1.canvas.textout(dx+20,345-i*50,st);
   end;
  for i:=1 to 6 do
   begin
    st:=currtostr((-1)*i*stepy*50);
    image1.canvas.textout(dx+20,345+i*50,st);
   end;
  for i:=1 to 15 do
   begin
    st:=currtostr(i*stepx*50);
    image1.canvas.textout(dx+46+i*50,355,st);
   end;
  {  for i:=1 to 6 do
   begin
    st:=currtostr((-1)*i*stepx*50);
    image1.canvas.textout(dx+46-i*50,355,st);
   end;
   }

end;


end.
```