|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Содержание  Введение …………………………………………………………………………….5  1. Общая часть …………………………………………………………………………6  1.1. Обзор элементов языка программирования…………………………………..6  2. Специальная часть ………………………………………………………………….8  2.1. Общая постановка задачи ……………………………………………………..8  2.2. Описание программ комплекса ……………………………………………...10  2.2. Макро блок- схема комплекса………………………………………………..12  2.4. Таблица идентификаторов комплекса ………………………………............13  2.5. Описание наборов данных ……………………………………………….......14  2.6. Структура записей файлов …………………………………………………...14  2.7. Постановка проблемной программы (процедуры) …………………………16  2.7.1. Описание проблемной программы ……………………………………16  2.7.2. Блок – схема проблемной программы ………………………………...16  3. Организация производства ………………………………………………………..17  3.1. Комплекс технических средств, необходимый для решения задачи……....17  3.2. Инструкция пользователю по работе с программой ……………………….18  Заключение …………………………………………………………………………20  Приложения …………………………………………………………………………21  Список используемых источников ………………………………………………..35 | | | | | | |
| *Выполнил* | | *Муханов А.* |  | *05.12.09* | *КР 3706002 000009 ПЗ* | лист |
| *Проверила* | | *Енсепова Г.К.* |  | *07.12.09* | 4 |
| *изм* | *лист* | *№ докум* | *Подпись* | *Дата* |
| **Введение**  Цель выполнения курсовой работы состоит в том, чтобы научиться описывать предметную область реального мира – объект и его атрибуты, закрепить навыки использования основных процедур, способов их описания и основных операции над ними.  Действительно, при помощи Turbo Pascal достаточно не просто создать программу, которая бы демонстрировала работу с ПК, создать всё-таки основу и саму сущность формирования вывода звука для встроенного динамика, расположенного в системном блоке компьютера.  В этой курсовой работе рассмотрим программирование звука и видео. В начале рассмотрим, как программируется вывод звука для встроенного динамика и видео, для вывода изображения на монитор. Затем познакомимся с основными принципами программирования для звуковой и видео карты.  Невозможно представить себе современный персональный компьютер без устройств вывода звука и видео. Нормой стало оснащение компьютера специальным устройством – звуковой картой, позволяющей творить со звуком самые настоящие чудеса. «Озвученный» компьютер – это полноценный музыкальный центр. Он даёт пользователю возможность превратиться в исполнителя, играющего на различных музыкальных инструментах, он может даже стать неплохим рассказчиком, «проговаривая» набранный в электронном формате текст.  Персональные компьютеры за последние годы значительно подтянулись в качестве. Вместе с центральными процессорами эволюционировали и остальные компоненты компьютера. Соответственно, требования пользователей к компьютеру постоянно повышается. И если 20-25 лет назад был хрипящий звук встроенного в корпус маленького динамика (РС – спикер), то сегодня непременным атрибутом персонального компьютера является специализированная звуковая подсистема.  Для того, чтобы уверенно ориентироваться в современном разнообразии различных устройств, служащих цели «озвучить» персональный компьютер, необходимо четко и на достаточном уровне представить себе все процессы, происходящие внутри компьютера, которые были связаны звуком. | | | | | | |
| *Выполнил* | | *Муханов А.* |  | *05.12.09.* | *КР 3706002 000009 ПЗ* | лист |
| *Проверила* | | *Енсепова Г. К.* |  | *07.12.09* | 5 |
| *изм* | *лист* | *№ докум* | *Подпись* | *Дата* |
| **1 Общая часть**   * 1. **Обзор элементов языка программирования**   Что представляет собой звук?  Свое знакомство с программированием звука на Турбо Паскале мы начнем с использования встроенного динамика. Прежде всего замечу, что модуль Crt содержит две процедуры, предназначенные для работы с динамиком. Первая из них- Sound(Hz) - включает динамик на звуковой частоте, задаваемой значением параметра Hz (типа Word) в герцах. Динамик генерирует звук до тех пор, пока он не будет отключен вызовом процедуры NoSound, не имеющей параметров Процедура Sound обеспечивает довольно бедную интонацию, потому что она использует только целочисленные частоты. Да и встроенный динамик, к сожалению, дает звук самого низкого качества Он не допускает управления громкостью, очень по-разному резонирует на разных частотах. Не ждите от него многого' Тем не менее, мы постараемся хотя бы частично исправить положение и разработаем свои собственные процедуры для модуля Speaker.  Прежде всего, познакомимся поближе с работой встроенного динамика. Источником звуковых акустических колебаний является его диффузор. Он может находиться в одном из двух возможных положений. Для управления работой динамика ему посылаются прямоугольные импульсы определенной частоты. В течение первой половины каждого цикла диффузор перемещается в одно положение, а затем, в течение второй половины цикла, в другое. Это вызывает движение воздуха. В процессе генерации акустических колебаний прямоугольный импульс, который порождает неприятно звучащую ноту, несколько сглаживается. Так возникает звуковой сигнал.  Теперь несколько слов о технических деталях. Компьютер содержит микросхему 8255, которую называют «программируемым периферийным интерфейсом» (PPI — Programmable Peripheral Interface). PPI управляет работой динамика, используя микросхему таймера 8254 (PIT — Programming Interval Timer). Микросхема 8255 имеет различные порты, включая порт ввода\вывода $61 размером («шириной») в один байт с битами b7…b0. Запись в порт $61 нулевого либо единичного значения бита b0 позволяет установить режим управления динамиком. Если b0 = 1, то динамиком управляет второй канал микросхемы таймера. Микросхема таймера имеет 3 канала, каждый из которых предназначен для решения своих задач. Нулевой канал обслуживает системные часы, первый канал обслуживает микросхему прямого доступа к памяти (DMA). Второй канал связан со встроенным динамиком. Обычно он запрограммирован на генерацию последовательности прямоугольных импульсов, что дает при включении динамика непрерывный тон определенной частоты. Управление встроенным динамиком связано прежде всего с программированием второго канала таймера.  Для работы с таймером используются порты (всего 4), на которых для нас интерес представляют порт $42 (порт второго канала таймера) и порт $43 (порт управляющего слова — открыт только для записи). Каждый канал имеет свой | | | | | | |
| *Выполнил* | | *Муханов А.* |  | *05.12.09.* | *КР 3706002 000009 ПЗ* | лист |
| *Проверила* | | *Енсепова Г.К.* |  | *07.12.09.* | 6 |
| *изм* | *лист* | *№ докум* | *Подпись* | *Дата* |
| счетчик, содержимое которого уменьшается от некоторого максимального значения. Для второго канала определена константа $1234DD. Это значение представляет собой число, которое нужно разделить на частоту в герцах, чтобы получить значение счётчика для таймера.  Итак, мы выяснили, что единичное значение управляющего бита b0 означает, что динамик подключён к каналу, а нулевое значение – что он отключён от канала. Динамик включён, если бит b1равен единице. Таким образом, прежде всего надо научиться «включать» биты b0 и b1, записывая в них единичные значение, не изменяя значений всех прочих битов. Чтобы сообщить таймеру о том, что следующие два байта являются значением типа Word, обратно пропорциональным частоте, и что на выходе должен быть прямоугольный импульс, в порт микросхемы таймера $43 должно быть записано значение $86. После этого таймер будет включать и выключать динамик с определённой частотой. | | | | | | |
| *Выполнил* | | *Муханов А.* |  | *05.12.09.* | *КР 3706002 000009 ПЗ* | лист |
| *Проверила* | | *Енсепова Г. К.* |  | *07.12.09.* | 7 |
| *изм* | *лист* | *№ докум* | *Подпись* | *Дата* |
| **2 Специальная часть**  **2.1 Общая постановка задачи**  Темой данной курсовой работы, который был выполнен в ходе изучения курса “Основы алгоритмизации и программирования” является, - «Обработка потока, поступающего с видео и аудио устройства».  Далее мы обсудим: что такое DirectShow, для чего он предназначен, как он работает и рассмотрим технологию программирования звука. Потом нам предстоит написать программу, которая генерирует последовательности прямоугольных звуковых импульсов (стр. 21-22)  DirectShow – это архитектура для воспроизведения, перехвата и обработки потоков мультимедиа. С помощью этого API можно:   * проигрывать мультимедийные файлы различного формата, такие как MPEG (Motion Picture Experts Group), AVI (Audio-Video Interleaved), MP2 (MPEG Audio Layer-2), DVD и конечно WAV; * перехватывать видео-поток с различного рода TV-карт, видеокамер и т.п.; * создавать нестандартные обработчики мультимедиа-потоков и свои собственные форматы файлов; * обращаться непосредственно к видео и аудио потокам, чтобы выводить их на Surface DirectDraw.   DirectShow интегрирован с DirectX так, что использует DirectDraw и DirectSound для вывода изображения и звука, и, при наличии аппаратного ускорения, автоматически им воспользуется.  В концепции DirectShow мультимедийные данные – это поток, который проходит через несколько обрабатывающих блоков. Блоки, обрабатывающие поток данных, передают данные по цепочке друг другу, таким образом можно представить себе несколько “устройств”, каждое из которых выполняет какую-то обработку данных и передает их соседнему “устройству”. Эти “устройства” или “блоки обработки” данных называют фильтрами. Цепочка, по которой передаются данные, содержит несколько фильтров, связанных определенным образом.  В DirectShow имеются готовые фильтры, из которых, словно из детских кубиков, программист может выстроить ту или иную цепочку обработку данных, кроме того, конечно, можно создать свои, нестандартные фильтры.  Для создания такой “цепочки обработки” (которая, кстати, официально называются Filter Graph – “граф фильтров” или, в несколько вольном переводе - “схема соединения фильтров”), так вот для создания схемы соединения фильтров, предназначен самый базовый и лежащий в основе всех основ компонент DirectShow, под названием Filter Graph Manager – Менеджер Графа Фильтров.  Например, программа показывающая видео из AVI-файла может построить такой граф фильтров: | | | | | | |
| *Выполнил* | | *Муханов А.* |  | *05.12.09.* | *КР 3706002 000009 ПЗ* | лист |
| *Проверила* | | *Енсепова Г.К.* |  | *07.12.09* | 8 |
| *изм* | *лист* | *№ докум* | *Подпись* | *Дата* |
| В этом примере пять фильтров, первый (File Source) просто читает данные с диска, второй фильтр (AVI Splitter) разделяет данные на кадры и передает упакованные видео данные фильтру AVI Decompressor, который их распаковывает и передает фильтру Default DirectSound Device, выводящему звук. AVI Decompressor передает распакованные данные фильтру Video Renderer, который выводит кадры видео на экран.  Фильтры делятся на три типа:  Фильтры-источники (Source filters) - эти фильтры просто получают данные из какого-то источника, с диска (как фильтр File Source (Async) на рисунке), с CD или DVD дисковода или с TV – карты или карты, к которой подключена цифровая видеокамера.  Фильтры-преобразователи (Transform filters) – эти фильтры как видно из названия преобразуют поток данных, проходящий через них каким-либо образом, например – разделяет поток данных на кадры, производят декомпрессию и т.п. На нашем рисунке к таким фильтрам относятся AVI Splitter и AVI Decompressor.  Фильтры вывода (Renderer filters) – фильтры, которые получают полностью обработанные данные и выводят их на монитор, звуковую карту, пишут на диск или выводят на еще какое-нибудь устройство.  Итак из фильтров-кубиков можно выстраивать граф. Делается это с помощью интерфейса IGraphBuilder. Создать объект типа IGraphBuilder можно так:  *CoCreateInstance(CLSID\_FilterGraph,****nil****,CLSCTX\_INPROC\_SERVER,IID\_IG raphBuilder,MyGraphBuilder);*  Здесь переменная MyGraphBuilder имеет тип IGraphBuilder; идентификатор класса CLSID\_FilterGraph и IID\_IGraphBuilder обьявлены в файле DShow.pas, поэтому не забудьте добавить:  ***uses*** *DShow.pas*  Итак, интерфейс IGraphBuilder получен. Можно построить граф фильтров, такой, какой нам нужно. Впрочем, все не так сложно, IGraphBuilder достаточно интеллектуален, он может сам, автоматически, построить граф, в зависимости от  того какие файлы мы собираемся воспроизводить. Интерфейс IGraphBuilder имеет метод RenderFile, который получает имя файла в качестве параметра и, в зависимости от типа файла (которое определяется по расширению и по  специальным сигнатурам в файле), сканирует реестр, в поисках необходимой для построения графа информации, создает необходимые фильтры и строит | | | | | | |
| *Выполнил* | | *Муханов А.* |  | *05.12.09.* | *КР 3706002 000009 ПЗ* | лист |
| *Проверила* | | *Енсепова Г. К.* |  | *07.12.09.* | 9 |
| *изм* | *лист* | *№ докум* | *Подпись* | *Дата* |
| граф, предназначенный для воспроизведения файлов этого типа (WAV, AVI, MP2, MPG и т.д.).  **2.2 Описание программ комплекса**  После построения графа, DirectShow готов к воспроизведению. Для управления потоком данных через граф обработки предназначен интерфейс IMediaControl – он имеет методы Run, Pause и Stop.  Пример:  ***uses***  *... DShow, ActiveX,ComObj;*  ***var***  *MyGraphBuilder : IGraphBuilder;*  *MyMediaControl : IMediaControl;*  ***begin***  *CoInitialize(nil);*  *{получаем интерфейс IGraphBuilder}*  *CoCreateInstance(CLSID\_FilterGraph,nil,CLSCTX\_INPROC\_SERVER,IID\_I GraphBuilder,MyGraphBuilder);*  *{вызываем RenderFile - граф фильтров строится автоматически}*  *MyGraphBuilder.RenderFile(cool.avi',nil);*  *{получаем интерфейс ImediaControl}*  *MyGraphBuilder.QueryInterface(IID\_IMediaControl,MyMediaControl);*  *{Примечание - MyMediaControl - переменная типа IMediaControl}*  *{проигрываем видео}*  *MyMediaControl.Run;*  *{ждем пока пользователь не нажмет ОК (видео воспроизводится в отдельном (thread) потоке)}*  *ShowMessage('Нажмите OК');*  *CoUninitialize;*  ***end;***  Интерфейс IVideoWindow содержит методы для управления заголовком, стилем, местоположением и размерами окошка в котором проигрывается видео.  Давайте попробуем переделать наш пример так, чтобы видео выводилось не в отдельном окошке, а, скажем на компоненте TPanel, расположенном в нашей форме. Добавьте на форму компонент TPanel, пусть он называется Panel1. | | | | | | |
| *Выполнил* | | *Муханов А.* |  | *05.12.09.* | *КР 3706002 000009 ПЗ* | лист |
| *Проверила* | | *ЕнсеповаГ.К.* |  | *07.12.09* | 10 |
| *изм* | *лист* | *№ докум* | *Подпись* | *Дата* |
| ***uses***  *... DShow, ActiveX,ComObj;*  ***procedure*** *TForm1.Button1Click(Sender: TObject);*  ***var***  *MyGraphBuilder : IGraphBuilder;*  *MyMediaControl : IMediaControl;*  *VideoWindow : IVideoWindow;*  ***begin***  *CoInitialize(nil);*  *{получаем интерфейс IGraphBuilder}*  *CoCreateInstance(CLSID\_FilterGraph,nil,CLSCTX\_INPROC\_SERVER,IID\_I GraphBuilder,MyGraphBuilder);*  *{вызываем RenderFile - граф фильтров строится автоматически}*  *MyGraphBuilder.RenderFile('C:\Program Files\Borland\Pascal\Demos\Coolstuf\cool.avi',nil);*  *{получаем интерфейс ImediaControl}*  *MyGraphBuilder.QueryInterface(IID\_IMediaControl,MyMediaControl);*  *{Примечание - MyMediaControl - переменная типа IMediaControl}*  *{получаем интерфейс IVideoWindow}*  *MyGraphBuilder.QueryInterface(IID\_IVideoWindow,VideoWindow);*  *{Примечание - VideoWindow - переменная типа IVideoWindow}*  *{располагаем окошко с видео на панель}*  *VideoWindow.Set\_Owner(Self.Panel1.Handle);*  *VideoWindow.Set\_WindowStyle(WS\_CHILD OR WS\_CLIPSIBLINGS);*  *VideoWindow.SetWindowPosition(0,0,Panel1.ClientRect.Right,Panel1.ClientR ect.Bottom);*  *{проигрываем видео}*  *MyMediaControl.Run;*  *ShowMessage('Нажмите OК');*  *CoUninitialize;*  ***end****;*  Программа Speaker1 предназначена для генерации последовательности прямоугольных звуковых импульсов:  Двоичное представление шестнадцатеричного значения #FC имеет вид 11111100. Вначале определяются значение, находящееся в порте динамика, и два младших бита обнуляются (благодаря использованию операции and и маски $FC). В цикле динамик включается (в бит b1 записывается единица), затем, после  небольшой задержки, отключается и так повторяется до нажатия произвольной | | | | | | |
| *Выполнил* | | *Муханов А.* |  | *05.12.09.* | *КР 3706002 000009 ПЗ* | лист |
| *Проверила* | | *Енсепова Г.К.* |  | *07.12.09.* | 11 |
| *изм* | *лист* | *№ докум* | *Подпись* | *Дата* |
| клавиши. После нажатия клавиши выполнение цикла прекращается. Важно то, что оба младших бита будут содержать нулевые значения, и при всех последующих обращениях к динамику других программ он будет работать правильно. Программа генерирует тон частотой примерно 100Гц.  **2.3 Макро блок - схема комплекса**  Существует много способов описания, представления алгоритмов. Мы остановимся на четырех: естественном языке, графическом языке, алгоритмическом языке и языке программирования.  Каждый вид представления имеет средства описания алгоритмов, ориентированные на своего исполнителя:   * естественный язык - слова на естественном языке общения; * графический язык - графические схемы (блок-схемы) с описанием действий;   - алгоритмический язык - определенные слова естественного языка, которым придается значение действия или команды для исполнителя;  - язык программирования - команды языка программирования.  Во всех предложенных способах представления алгоритма, кроме последнего исполнителем является человек, а в последнем четвертом случае исполнитель-компьютер.  Алгоритмический язык - это система обозначений и правил для единообразной и точной записи алгоритмов и их исполнения.  Алгоритмический язык, с одной стороны, близок к обычному языку, поэтому читается и записывается как обыкновенный текст. С другой стороны, по своей форме алгоритмический язык приближается к языкам программирования, что дает возможность быстрее освоить языки программирования для ЭВМ.  Алгоритмический язык имеет свой словарь. Основу его составляют слова, представляющие команды этого языка. В алгоритмическом языке также используются слова, смысл и способ употребления которых не изменяется. Они называются служебными словами.  Служебные слова алгоритмического языка подчеркиваются и записываются, как правило, в сокращенной форме и предназначены для оформления алгоритма.  Синтаксические конструкции языка подразделяются на два типа: описания данных (величин) и операторов (команд).  Представление алгоритмов в виде блок-схем. Наиболее наглядным способом представления алгоритма является его графическое представление в виде блок-схем.  Блок-схема - это графическое представление алгоритма, каждое действие алгоритма записывается соответствующей геометрической фигурой. | | | | | | |
| *Выполнил* | | *Муханов А.* |  | *05.12.09.* | *КР 3706002 000009 ПЗ* | лист |
| *Проверила* | | *Енсепова Г. К.* |  | *07.12.09* | 12 |
| *изм* | *лист* | *№ докум* | *Подпись* | *Дата* |
| |  |  | | --- | --- | | Условное обозначение | Пояснение | |  | Начало алгоритма;  Конец алгоритма;  Процесс или группа действий;  Выбор действия в зависимости от истинности или ложности;  Повторение действии заданное количество раз  Ввод (вывод) значений переменных;  Направление вычисления. |   **2.4 Таблица идентификаторов комплекса**  Для написания программы Speaker1, мы использовали следующие идентификаторы:   |  |  | | --- | --- | | **Speaker1** | Этот идентификатор является  названием программы | | **Crt** | В модуле CRT реализованы специальные процедуры и функции для работы с текстовой информацией на дисплее, позволяющие: управлять текстовыми режимами, организовывать окна вывода на экран, настраивать цвета символов на экране, управлять курсором. Кроме того, в модуль включены функции опроса клавиатуры и процедуры управления встроенным в ПЭВМ динамиком. | | **Speaker\_port** | Имя константы, которому присвоено значение %61. | | **Portval** | Название порядкового типа данных, которое имеет значение Byte. | | **Byte** | Порядковый тип данных. Длина, байт=1;  Диапазон значений 0…255 | | **Delay** | Эта процедура приостанавливает выполнение программы на заданное число миллисекунд. | | **KeyPressed** | Возвращает значение типа Boolean, указывающее состояние буфера клавиатуры: False означает, что буфер пуст, a True - что в буфере есть хотя бы один символ, еще не прочитанный программой. | | | | | | | |
| *Выполнил* | | *Муханов А.* |  | *05.12.09.* | *КР 3706002 000009 ПЗ* | лист |
| *Проверила* | | *Енсепова Г.К.* |  | *07.12.09.* | 13 |
| *изм* | *лист* | *№ докум* | *Подпись* | *Дата* |
| **2.5 Описание наборов данных**  В основе иерархии классов, обеспечивающих функционирование наборов данных в приложениях баз данных, лежит класс TDataSet. Хотя он почти не содержит методов, реально обеспечивающих работоспособность основных механизмов набора данных, тем не менее его значение трудно переоценить.  Этот класс задает структурную основу функционирования набора данных. Другими словами, это скелет набора данных, к методам которого необходимо лишь добавить требуемые вызовы соответствующих функций реальных технологий.  При решении наиболее распространенных задач программирования в процессе создания приложений баз данных класс TDataSet не нужен.  Тем не менее знание основных принципов работы набора данных всегда полезно. Кроме этого, класс TDataSet может использоваться разработчиками в качестве основы для создания собственных компонентов. Поэтому рассмотрим основные механизмы, реализованные в наборе данных.  **2.6 Структура записей файлов**  Multimedia Streaming – это архитектура, используемая в DirectShow для облегчения жизни программиста. Эта архитектура позволяет работать с мультимедиа данными, как с абстрактным потоком, не вдаваясь в подробности форматов хранения мультимедиа-файлов или специфику устройств-источников мультимедиа. Используя эту архитектуру, программист концентрируется не на расшифровке и преобразовании данных, а на управлении потоком данных, кадрами видео или аудио семплами.  На вершине иерархии находится базовый объект Multimedia Stream, который является контейнером для объектов Media Stream. Объект Multimedia Stream может содержать один или несколько объектов Media Stream. В то время как каждый объект типа Media Stream предназначен для работы с данными какого-то одного типа (видео, аудио и т.п.) – Multimedia Stream – просто содержит методы для обращения к содержащимся в нем объектам Media Stream и не зависит от типа данных.  Сейчас мы создадим объект типа IAMMultiMediaStream. Этот интерфейс унаследован от IMultimediaStream и содержит, кроме прочего, функцию OpenFile, которая автоматически строит граф фильтров для воспроизведения файла:  *CoCreateInstance(CLSID\_AMMultiMediaStream,nil,CLSCTX\_INPROC\_SE RVER, IID\_IAMMultiMediaStream, AMStream);*  Здесь переменная AMStream имеет тип IAMMultiMediaStream.  Мы создали контейнер для мультимедийных потоков. Сверяемся с рисунком - мы на верхнем уровне иерархии. У нас есть объект типа IMultimediaStream – теперь в этот контейнер нужно проинициализировать и добавить один или несколько мультимедиа потоков, нужного нам типа. Сначала инициализация:  *AMStream.Initialize(STREAMTYPE\_READ,*  *AMMSF\_NOGRAPHTHREAD, nil);* | | | | | | |
| *Выполнил* | | *Муханов А.* |  | *05.12.09.* | *КР 3706002 000009 ПЗ* | лист |
| *Проверила* | | *Енсепова Г.К.* |  | *07.12.09.* | 14 |
| *изм* | *лист* | *№ докум* | *Подпись* | *Дата* |
| *При инициализации указываем, что будут создаваться мультимедиа потоки для чтения, передав значение STREAMTYPE\_READ (другие варианты STREAMTYPE\_WRITE, STREAMTYPE\_TRANSFORM).*  *Создадим теперь мультимедиа потоки для видео и звука:*  *AMStream.AddMediaStream(DDraw, MSPID\_PrimaryVideo, 0, NewMediaStremVideo);*  *AMStream.AddMediaStream(nil, MSPID\_PrimaryAudio, AMMSF\_ADDDEFAULTRENDERER, NewMediaStremAudio);*  *Вызываем метод OpenFile – файл загружается, и автоматически строится граф фильтов:*  *AMStream.OpenFile('cool.avi', 0);*  *Осталось направить видео поток мультимедиа поток на Surface. Вот процедура, которая делает это:*  ***procedure*** *TForm1.RenderStreamToSurface(Surface : IDirectDrawSurface; MMStream : IMultiMediaStream);*  ***var***  *PrimaryVidStream : IMediaStream;*  *DDStream : IDirectDrawMediaStream;*  *Sample : IDirectDrawStreamSample;*  *RECT : TRect;*  *ddsd : TDDSURFACEDESC;*  *Z : DWORD;*  ***begin***  *MMStream.GetMediaStream(MSPID\_PrimaryVideo, PrimaryVidStream);*  *PrimaryVidStream.QueryInterface(IID\_IDirectDrawMediaStream, DDStream);*  *ddsd.dwSize := sizeof(ddsd);*  *DDStream.GetFormat(ddsd, Palitra, ddsd, Z);*  *rect.top:=(480-ddsd.dwHeight)div 2; rect.left:=(640 - ddsd.dwWidth) div 2;*  *rect.bottom := rect.top+ddsd.dwHeight; rect.right := rect.left+ddsd.dwWidth;*  *DDStream.CreateSample(Surface, Rect, 0, Sample);*  *MMStream.SetState(STREAMSTATE\_RUN);*  ***end****;*  Метод IDirectDrawStreamSample.Update выводит очередной кадр на Surface. При достижении конца потока он вернет ошибку с кодом $40002 (MS\_S\_ENDOFSTREAM), я в этом случае просто перематываю поток к началу, методом Seek.  В этой программе инициализируется DirectDraw, создается Surface , а затем на него выводится видео из avi-файла. | | | | | | |
| *Выполнил* | | *Муханов А.* |  | *05.12.09.* | *КР 3706002 000009 ПЗ* | лист |
| *Проверила* | | *Енсепова Г.К.* |  | *07.12.09.* | 15 |
| *изм* | *лист* | *№ докум* | *Подпись* | *Дата* |
| **2.7 Постановка проблемной программы (процедуры)**  **2.7.1 Описание проблемной программы**  У программы Speaker1 (стр. 21) есть существенный недостаток. Обработка центральным процессором прерываний во время её выполнения влияет на точность соблюдения временных задержек. Попробуем подвигать мышью во время работы программы, и мы услышим перерывы в звучании динамика. По этой причине использование процедуры задержки DELAY является нежелательным. Процедура вывода звука, кроме того, «захватывает» центральный процессор, и одновременно выполнение других процедур оказывается невозможным. Улучшенный вариант программы использует второй канал таймера (программа speaker 2, стр. 22 ).  Надо помнить, что для использования таймера необходимо сначала подключить динамик ко второму каналу PPI, а затем записать единичные значения двух младших битов в порт динамика.  После манипуляции с мышью уже не будут приводить к прерывистому звучанию динамика.  **2.7.2 Блок - схема проблемной программы**  Блок-схема программы speaker2 (стр. 34) отличается от блок-схемы speaker1 (стр. 33) тем, что он более совершенен и не имеет такого недочёта как проблемы с мышью при его движении. Более наглядно можно посмотреть на | | | | | | |
| *Выполнил* | | *Муханов А.* |  | *05.12.09.* | *КР 3706002 000009 ПЗ* | лист |
| *Проверила* | | *Енсепова Г.К.* |  | *07.12.09.* | 16 |
| *изм* | *лист* | *№ докум* | *Подпись* | *Дата* |
| **3 Организация производства**  **3.1 Комплекс технических средств, необходимый для решения задач**  Для полноценной работы наших программы speaker1 и speaker2 необходим звук, который будет описан ниже.  Интегрированный AC-link контроллер в Intel 82801ВА I/O Controller Hub (ICH2), имеющий поддержку AMR и CNR;  Стерео АС'97-аудиокодек Analog Devices AD 1885;  На плате расположены пластмассовые разъёмы с цветной РС99 маркировкой (микрофонный вход, линейный выход, линейный вход), а также два внутренних разъёма - AUX in, CD in.  Программная часть:  Комплект П/О SoundMAX with SPX от Analog Devices с весом дистрибутива драйверов в 54 Мбайт. Заявлена оптимизация под ММХ и SSE2 SIMD инструкции для процессоров Intel Pentium III и Pentium 4.  Драйвера:  Win98 SE, WinME и Win2K.  Совместимость с игровыми 2D Sound API:   * DirectSound/DirectX 8.0; * Sound Production extensions (SPX); * EAX2.0; * A2D 1.0; * Sensaura Macro FX / ZoomFX; * IA-SIGI2DL2. Характеристики аудиокодека: * полный дуплекс вплоть до 16 бит 48 кГц; * SNR = 94дБА; * THD = 0,01 %; * IMD = 90дБ;   - FR = 20 Гц - 20 кГц +/- 0.1 дБ.   * MIDI: * wavetable синтезатор 4 Мбайт DLS2, 128 GM-инструментов; * 1.2 Мбайт, 260 XGlite-инструментов; * 1024 MAX voice программных голоса;   - SPX - Sound Production eXtention - технология аудиорендеринга нового поколения.  Синтезом звука разработчики управляют напрямую из игр и мультимедиа-приложений. ЗD-звук от Sensaura:  - EnvironmentFX, звуковой рендер 2D с 26 пресетами и полной поддержкой ЕАХ1.0/2.0, I2DL2 реверберации;   * эффекты ближнего поля Sensaura MacroFX; * эффекты дальнего поля Sensaura ZoomFX; | | | | | | |
| *Выполнил* | | *Муханов А.* |  | *05.12.09.* | *КР 3706002 000009 ПЗ* | лист |
| *Проверила* | | *Енсепова Г.К.* |  | *07.12.09.* | 17 |
| *изм* | *лист* | *№ докум* | *Подпись* | *Дата* |
| * технология, использующая отдельные HRTF-фильтры для наушников, 2/4/6 колонок Digital Ear (возможен апгрейд до технологии индивидуальной подстройки HRTF-фильтров Virtual Ear); * технология Virtual Theater Surround для прослушивания 4 и 5.1-канальных саундтреков в фильмах (возможен апгрейд до аналогичной технологии для наушников Headphone Virtual Theater Surround).   Установка  В комплекте с материнской платой идут драйвера под Win98 SE, WinME и Win2K. Драйвера звуковых устройств выполнены в виде WDM-драйверов.  Конфигурация тестового компьютера:   * материнская плата Intel D815EFV; * процессор Intel РЗ-550Е; * видеокарта GeForce2 MX; * звуковая плата EgoSys WaveTerminal 2496; * сетевая карта Realtek 8029. * Операционная система Win98SE rus 4.10.2222А, DirectX 8.0а rus.   По умолчанию автоинсталлятор от Intel устанавливает все необходимые патчи и сам автоматически перезагружает компьютер в процессе по мере необходимости, что очень приятно (не нужно сидеть и всё время тупо нажимать кнопку Далее). Однако после всех установок в системе стоит SoundMAX версии 2, несмотря на то, что на компакте имеется дистрибутив последней версии драйверов - SoundMAX2. После корректного удаления второй версии ставим третью:  После перезагрузки в системном трее появляется синенькая эмблемка Analog Devices.  После выбора верхнего пункта появляется панель управления многочисленными настройками SoundMAX 2.0.  Настройка динамиков позволяет выбрать три варианта: наушники, рядом расположенные стереоколонки, разнесённые на некоторое расстояние стереоколонки. Звуковые среды - это выбор типа среды окружения. Доступно свыше десятка пресетов. Виртуальное пространственное звучание- функция создания виртуальных источников в наушниках или колонках. Клик на картинку переносит нас на сайт Sensaura в виртуальный магазин, где можно купить этот апгрейд (предназначенный только для SonudMAX2.0) или загрузить 21-дневную trial-версию.  **3.2 Инструкция пользователю по работе с программой**  Эта инструкция предназначена для программы zvuko\_zapis, который наглядно показан в приложении В (стр. 23). Для начала, чтобы получить работоспособную программу надо её текст вставить в Турбо Паскаль и откомпилировать её и она (наша программа) сохранится там, где сохранён исходник программы. Далее у нас появится екзешник (т.е. файл с расширением .EXE) – это и есть наша программа. Теперь осталось её запустить. После запуска наша программа будет иметь такой вид: | | | | | | |
| *Выполнил* | | *Муханов А.* |  | *05.12.09.* | *КР 3706002 000009 ПЗ* | лист |
| *Проверила* | | *Енсепова Г.К.* |  | *07.12.09.* | 18 |
| *изм* | *лист* | *№ докум* | *Подпись* | *Дата* |
| Тут с её использованием всё просто. Если имеется микрофон, то можно сразу записывать звук нажав на кнопку «Запись», который выделен красным цветом. Кнопка «Start» предназначена для воспроизведения записанного звука. Кнопка «Stop» останавливает запись и подготавливает его для воспроизведения.  Записанный звук сохраняется под расширением .WAV. Кнопка «Входы» предназначен для приёма и показа звука с одного из входов. Звук показывается на рабочей области программы кривыми линиями. Кривые линии - это диапазон звука. | | | | | | |
| *Выполнил* | | *Муханов А.* |  | *05.12.09.* | *КР 3706002 000009 ПЗ* | лист |
| *Проверила* | | *Енсепова Г.К.* |  | *07.12.09.* | 19 |
| *изм* | *лист* | *№ докум* | *Подпись* | *Дата* |
| **Заключение**  Выполняя нашу курсовую работу, мы задались целью выяснить, какие есть способы программирование звука и видео и по возможности внести что-то свое. Выяснив, что на данный момент существует несколько способов программирование звука, мы решили написать программу звука на языке Турбо Паскаль.  В этой курсовой работе мы описали в основном 3 программы: speaker1, speaker2 и zvuko\_zapis.  В основу разработанной программы звука на языке Турбо Паскаль заложена программа speaker1 предназначенная для генерации последовательности прямоугольных звуковых импульсов и его улучшенный вариант - программа speaker2. Учтено, что манипуляции с мышью не должно приводить к прерывистому звучанию динамика, и использовано вещественное значение частоты, а также программа zvuko\_zapis, предназначенный для записи звука и приёма и показа звука с одного из входов.  Также рассмотрена работа DirectShow, для чего он предназначен и как он воспроизводит видео форматы.  Рассмотрев много языков программирования, которые позволяют осуществлять программирование звука и видео, мы остановили свой выбор на языке Турбо Паскаль.  Выполненное работа дает возможность использование программу, как преподавателям, так и студентам, для этого достаточно лишь установить данную программу на компьютер. | | | | | | |
| *Выполнил* | | *Муханов А.* |  | *05.12.09.* | *КР 3706002 000009 ПЗ* | лист |
| *Проверила* | | *Енсепова Г.К.* |  | *07.12.09.* | 20 |
| *изм* | *Лист* | *№ докум* | *Подпись* | *Дата* |
| **Приложения**  **Приложение А**  **Текст программы speaker1**  **Листинг:**  *Program speaker1;*  *Uses crt;*  *Const*  *Speaker\_port = %61;*  *Var*  *Portval : byte;*    *Begin*  *Portval := port[speaker\_port] and $FC;*  *While not KeyPressed do*  *begin*  *Port[speaker\_port] := portval or 2;*  *Delay(5);*  *Port[speaker\_port] := portval;*  *Delay(5);*  *End;*  *ReadKey;*  *End.* | | | | | | |
| *Выполнил* | | *Муханов А.* |  | *05.12.09.* | *КР 3706002 000009 ПЗ* | лист |
| *Проверила* | | *Енсепова Г. К.* |  | *07.12.09.* | 21 |
| *изм* | *лист* | *№ докум* | *Подпись* | *Дата* |
| **Приложение Б**  **Текст программы speaker2**  **Листинг :**  *Program speaker2;*  *Uses crt;*  *Const*  *Speaker\_port = $61;*  *Pit\_control = $42;*  *Pit\_channel\_2 = $42;*  *Pit\_freq = $1224dd;*  *Procedure sound (frequency : word);*  *Var*  *Counter : word;*  *Begin*  *Counter := pit\_freq div frequency;*  *Port[pit\_control] := $b6;*  *Port[pit\_channel\_2] := lo(counter);*  *Port[pit\_channel\_2] := hi(counter);*  *Port[speaker\_port] := port[speaker\_port] or 2;*  *End;*  *Procedure nosound;*  *Begin*  *Port[speaker\_port] := port[speaker\_port] and $fc;*  *End;*  *Begin*  *Sound(200);*  *Repeat until keypressed;*  *Nosound;*  *End.* | | | | | | |
| *Выполнил* | | *Муханов А.* |  | *05.12.09.* | *КР 3706002 000009 ПЗ* | лист |
| *Проверила* | | *Енсепова Г.К.* |  | *07.12.09.* | 22 |
| *изм* | *лист* | *№ докум* | *Подпись* | *Дата* |
| **Приложение В**  **Текст программы**  **Листинг:**  unit zvuko\_zapis;  interface  uses  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms,  Dialogs, StdCtrls, ExtCtrls, ComCtrls, MMSystem, Buttons;  type  TForm1 = class(TForm)  Button1: TButton;  Button2: TButton;  Label1: TLabel;  Image1: TImage;  Button2: TButton;  Label2: TLabel;  BitBtn1: TBitBtn;  CheckBox1: TCheckBox;  procedure Button1Click(Sender: TObject);  procedure Button2Click(Sender: TObject);  procedure FormCreate(Sender: TObject);  procedure FormCloseQuery(Sender: TObject; var CanClose: Boolean);  procedure Button2Click(Sender: TObject);  procedure BitBtn1Click(Sender: TObject);  private  { Private declarations }  public  procedure OnWaveIn(var Msg: TMessage); message MM\_WIM\_DATA;  { Public declarations }  end;  TData16 = array [0..127] of smallint;  PData16 = ^TData16;  tWaveFileHdr = packed record  riff: array[0..2] of Char;  len: DWord;  cWavFmt: array[0..7] of Char;  dwHdrLen: DWord;  wFormat: Word;  wNumChannels: Word;  dwSampleRate: DWord; | | | | | | |
| *Выполнил* | | *Муханов А.* |  | *05.12.09.* | *КР 3706002 000009 ПЗ* | лист |
| *Проверила* | | *Енсепова Г.К.* |  | *07.12.09.* | 23 |
| *изм* | *лист* | *№ докум* | *Подпись* | *Дата* |
| **Продолжение Приложения В**  dwBytesPerSec: DWord;  wBlockAlign: Word;  wBitsPerSample: Word;  cData: array[0..2] of Char;  dwDataLen: DWord;  end;  const BufSize=11000; { Размер буфера на 1 сек}  var  Form1: TForm1;  implementation  var  WaveIn: hWaveIn;  hBuf: THandle;  BufHead: TWaveHdr;  m:array[1..bufSize] of smallInt;  h,w,h2:integer;  zs:boolean=false;//запущен звук  //Для записи в wav  rec:boolean=false; //идет запись  mz :array of smallInt;  waveHdr:tWaveFileHdr;  qz:integer; //записано звука;  {$R \*.DFM}  PROCEDURE iniWav;  begin  WaveHdr.riff:='RIFF';  WaveHdr.cWavFmt:='WAVEfmt ';  WaveHdr.dwHdrLen:=16;  WaveHdr.wFormat:=1;  WaveHdr.wNumChannels:=1;  WaveHdr.dwSampleRate:=11000;  WaveHdr.wBlockAlign:=4;  WaveHdr.dwBytesPerSec:=22000;  WaveHdr.wBitsPerSample:=16;  WaveHdr.cData:='data';  WaveHdr.dwDataLen:=qz\*2;  end; | | | | | | |
| *Выполнил* | | *Муханов А.* |  | *05.12.09.* | *КР 3706002 000009 ПЗ* | лист |
| *Проверила* | | *Енсепова Г.К.* |  | *07.12.09.* | 24 |
| *изм* | *лист* | *№ докум* | *Подпись* | *Дата* |
| **Продолжение Приложения В**  procedure TForm1.Button1Click(Sender: TObject);  var  header: TWaveFormatEx;  BufLen: word;  buf: pointer;  begin  if zs then exit;  with header do begin  wFormatTag := WAVE\_FORMAT\_PCM;  nChannels := 1; // количество каналов  nSamplesPerSec := 11000;// частота  wBitsPerSample := 16; // бит на отсчет  nBlockAlign := nChannels \* (wBitsPerSample div 8);  nAvgBytesPerSec := nSamplesPerSec \* nBlockAlign;  cbSize := 0;  end;  WaveInOpen(Addr(WaveIn), WAVE\_MAPPER, addr(header),Form1.Handle, 0, CALLBACK\_WINDOW);  BufLen := header.nBlockAlign \* BufSize;  hBuf := GlobalAlloc(GMEM\_MOVEABLE and GMEM\_SHARE, BufLen);  Buf := GlobalLock(hBuf);  with BufHead do begin  lpData := Buf;  dwBufferLength := BufLen;  dwFlags := WHDR\_BEGINLOOP;  end;  WaveInPrepareHeader(WaveIn, Addr(BufHead), sizeof(BufHead));  WaveInAddBuffer(WaveIn, addr(BufHead), sizeof(BufHead));  zs:=true;  WaveInStart(WaveIn);  End;  procedure TForm1.Button2Click(Sender: TObject);  begin  if not zs then Exit;  WaveInReset(WaveIn);  WaveInUnPrepareHeader(WaveIn, addr(BufHead), sizeof(BufHead));  WaveInClose(WaveIn);  GlobalUnlock(hBuf);  GlobalFree(hBuf);  zs:=false;  end; | | | | | | |
| *Выполнил* | | *Муханов А.* |  | *05.12.09.* | *КР 3706002 000009 ПЗ* | лист |
| *Проверила* | | *Енсепова Г.К.* |  | *07.12.09.* | 25 |
| *изм* | *лист* | *№ докум* | *Подпись* | *Дата* |
| **Продолжение Приложения В**  procedure TForm1.OnWaveIn;  var  data16: PData16;  i,d,z,s,x,y,xx,max,s0: integer;  begin  //сразу пустим запись дальше чтоб не прерывалась  WaveInAddBuffer(WaveIn, PWaveHdr(Msg.lParam),SizeOf(TWaveHdr));  data16 := PData16(PWaveHdr(Msg.lParam)^.lpData);  //перепишем звук из массива в который пишется  //в массив который обрабатывается чтоб запись его не портила  move(data16^[0],m,BufSize\*2);  if data16^[0]<>m[1] then showMessage('Не успела');  //Обработка звука  s:=0;  s0:=0;  max:=0;  for i := 1 to BufSize do begin  z:=m[i];  inc(s0,z);  z:=abs(z);  inc(s,z);  if z>max then max:=z;  end;  //показ звука  s:=s div bufSize;  s0:=s0 div bufSize;  label1.caption:='Среднее: '+intToStr(s)+  ', Максимум: '+intToStr(max)+  ', Постоянный уровень: '+intToStr(s0);  with form1.image1.Picture.Bitmap.canvas do begin  fillRect(rect(0,0,w,h));  pen.color:=$CCCCCC; moveTo(w,h2); lineTo(0,h2);  pen.color:=0;  max:=abs(max-abs(s0));  if max<16 then max:=16;  if checkBox1.checked then d:=BufSize else d:=w;  for x:=1 to w do begin  xx:=x\*BufSize div d; | | | | | | |
| *Выполнил* | | *Муханов А.* |  | *05.12.09.* | *КР 3706002 000009 ПЗ* | лист |
| *Проверила* | | *Енсепова Г.К.* |  | *07.12.09.* | 26 |
| *изм* | *лист* | *№ докум* | *Подпись* | *Дата* |
| **Продолжение Приложения В**  y:=h2+(m[xx]-s0)\*h2 div max;  if x=1 then moveTo(0,y) else lineTo(x,y);  end;  end;  if rec then begin  //запись в массив для файла  setLength(mz,qz+bufSize+1);  move(m[1],mz[qz+1],BufSize\*2);  inc(qz,BufSize);  form1.label2.caption:='Записано '+formatFloat('0.00',qz\*2/1000000)+' мб';  end;  end;  procedure TForm1.FormCreate(Sender: TObject);  begin  w:=image1.width;  h:=image1.height;  h2:=h div 2;  image1.Picture.Bitmap.width:=w;  image1.Picture.Bitmap.height:=h;  Form1.Button1Click(Sender);  end;  procedure TForm1.FormCloseQuery(Sender: TObject; var CanClose: Boolean);  begin  Form1.Button2Click(Sender)  end;  procedure TForm1.Button2Click(Sender: TObject);  begin  winExec('SndVol22.exe /r',SW\_SHOW);  end;  procedure TForm1.BitBtn1Click(Sender: TObject);  var fw:file;  begin  if rec then begin  rec:=false;  if qz>0 then begin  //вывод в файл wav  iniWav;  assignFile(fw,'sound.wav');  rewrite(fw,1);  blockWrite(fw,waveHdr,sizeOf(waveHdr)); | | | | | | |
| *Выполнил* | | *Муханов А.* |  | *05.12.09.* | *КР 3706002 000009 ПЗ* | лист |
| *Проверила* | | *Енсепова Г.К.* |  | *07.12.09.* | 27 |
| *изм* | *лист* | *№ докум* | *Подпись* | *Дата* |
| **Продолжение Приложения В**  blockWrite(fw,mz[1],qz\*2);  closeFile(fw);  showMessage('Записано в "sound.wav"');  label2.visible:=false;  BitBtn1.caption:='Запись';  end;  end  else begin  qz:=0;  rec:=true;  label2.visible:=true;  BitBtn1.caption:='Стой';  end;  end;  end.  const BufSize=11000; { Размер буфера на 1 сек}  var  Form1: TForm1;  implementation  var  WaveIn: hWaveIn;  hBuf: THandle;  BufHead: TWaveHdr;  m:array[1..bufSize] of smallInt;  h,w,h2:integer;  zs:boolean=false;//запущен звук  //Для записи в wav  rec:boolean=false; //идет запись  mz :array of smallInt;  waveHdr:tWaveFileHdr;  qz:integer; //записано звука;  {$R \*.DFM}  PROCEDURE iniWav;  begin  WaveHdr.riff:='RIFF';  WaveHdr.cWavFmt:='WAVEfmt ';  WaveHdr.dwHdrLen:=16;  WaveHdr.wFormat:=1; | | | | | | |
| *Выполнил* | | *Муханов А.* |  | *05.12.09.* | *КР 3706002 000009 ПЗ* | лист |
| *Проверила* | | *Енсепова Г.К.* |  | *07.12.09.* | 28 |
| *изм* | *лист* | *№ докум* | *Подпись* | *Дата* |
| **Продолжение Приложения В**  WaveHdr.wNumChannels:=1;  WaveHdr.dwSampleRate:=11000;  WaveHdr.wBlockAlign:=4;  WaveHdr.dwBytesPerSec:=22000;  WaveHdr.wBitsPerSample:=16;  WaveHdr.cData:='data';  WaveHdr.dwDataLen:=qz\*2;  end;  procedure TForm1.Button1Click(Sender: TObject);  var  header: TWaveFormatEx;  BufLen: word;  buf: pointer;  begin  if zs then exit;  with header do begin  wFormatTag := WAVE\_FORMAT\_PCM;  nChannels := 1; // количество каналов  nSamplesPerSec := 11000;// частота  wBitsPerSample := 16; // бит на отсчет  nBlockAlign := nChannels \* (wBitsPerSample div 8);  nAvgBytesPerSec := nSamplesPerSec \* nBlockAlign;  cbSize := 0;  end;  WaveInOpen(Addr(WaveIn), WAVE\_MAPPER, addr(header),Form1.Handle, 0, CALLBACK\_WINDOW);  BufLen := header.nBlockAlign \* BufSize;  hBuf := GlobalAlloc(GMEM\_MOVEABLE and GMEM\_SHARE, BufLen);  Buf := GlobalLock(hBuf);  with BufHead do begin  lpData := Buf;  dwBufferLength := BufLen;  dwFlags := WHDR\_BEGINLOOP;  end;  WaveInPrepareHeader(WaveIn, Addr(BufHead), sizeof(BufHead));  WaveInAddBuffer(WaveIn, addr(BufHead), sizeof(BufHead));  zs:=true;  WaveInStart(WaveIn);  End;  procedure TForm1.Button2Click(Sender: TObject);  begin  if not zs then Exit; | | | | | | |
| *Выполнил* | | *Муханов А.* |  | *05.12.09.* | *КР 3706002 000009 ПЗ* | лист |
| *Проверила* | | *Енсепова Г.К.* |  | *07.12.09.* | 29 |
| *изм* | *лист* | *№ докум* | *Подпись* | *Дата* |
| **Продолжение приложения В**  WaveInReset(WaveIn);  WaveInUnPrepareHeader(WaveIn, addr(BufHead), sizeof(BufHead));  WaveInClose(WaveIn);  GlobalUnlock(hBuf);  GlobalFree(hBuf);  zs:=false;  end;  procedure TForm1.OnWaveIn;  var  data16: PData16;  i,d,z,s,x,y,xx,max,s0: integer;  begin  //сразу пустим запись дальше чтоб не прерывалась  WaveInAddBuffer(WaveIn, PWaveHdr(Msg.lParam),SizeOf(TWaveHdr));  data16 := PData16(PWaveHdr(Msg.lParam)^.lpData);  //перепишем звук из массива в который пишется  //в массив который обрабатывается чтоб запись его не портила  move(data16^[0],m,BufSize\*2);  if data16^[0]<>m[1] then showMessage('Не успела');  //Обработка звука  s:=0;  s0:=0;  max:=0;  for i := 1 to BufSize do begin  z:=m[i];  inc(s0,z);  z:=abs(z);  inc(s,z);  if z>max then max:=z;  end;  //показ звука  s:=s div bufSize;  s0:=s0 div bufSize;  label1.caption:='Среднее: '+intToStr(s)+  ', Максимум: '+intToStr(max)+  ', Постоянный уровень: '+intToStr(s0);  with form1.image1.Picture.Bitmap.canvas do begin  fillRect(rect(0,0,w,h));  pen.color:=$CCCCCC; moveTo(w,h2); lineTo(0,h2);  pen.color:=0; | | | | | | |
| *Выполнил* | | *Муханов А.* |  | *05.12.09.* | *КР 3706002 000009 ПЗ* | лист |
| *Проверила* | | *Енсепова Г.К.* |  | *07.12.09.* | 30 |
| *изм* | *лист* | *№ докум* | *Подпись* | *Дата* |
| **Продолжение приложения В**  max:=abs(max-abs(s0));  if max<16 then max:=16;  if checkBox1.checked then d:=BufSize else d:=w;  for x:=1 to w do begin  xx:=x\*BufSize div d;  y:=h2+(m[xx]-s0)\*h2 div max;  if x=1 then moveTo(0,y) else lineTo(x,y);  end;  end;  if rec then begin  //запись в массив для файла  setLength(mz,qz+bufSize+1);  move(m[1],mz[qz+1],BufSize\*2);  inc(qz,BufSize);  form1.label2.caption:='Записано '+formatFloat('0.00',qz\*2/1000000)+' мб';  end;  end;  procedure TForm1.FormCreate(Sender: TObject);  begin  w:=image1.width;  h:=image1.height;  h2:=h div 2;  image1.Picture.Bitmap.width:=w;  image1.Picture.Bitmap.height:=h;  Form1.Button1Click(Sender);  end;  procedure TForm1.FormCloseQuery(Sender: TObject; var CanClose: Boolean);  begin  Form1.Button2Click(Sender)  end;  procedure TForm1.Button2Click(Sender: TObject);  begin  winExec('SndVol22.exe /r',SW\_SHOW);  end;  procedure TForm1.BitBtn1Click(Sender: TObject);  var fw:file;  begin  if rec then begin  rec:=false; | | | | | | |
| *Выполнил* | | *Муханов А.* |  | *05.12.09.* | *КР 3706002 000009 ПЗ* | лист |
| *Проверила* | | *Енсепова Г.К.* |  | *07.12.09.* | 31 |
| *изм* | *лист* | *№ докум* | *Подпись* | *Дата* |
| **Продолжение приложения В**  if qz>0 then begin  //вывод в файл wav  iniWav;  assignFile(fw,'sound.wav');  rewrite(fw,1);  blockWrite(fw,waveHdr,sizeOf(waveHdr));  blockWrite(fw,mz[1],qz\*2);  closeFile(fw);  showMessage('Записано в "sound.wav"');  label2.visible:=false;  BitBtn1.caption:='Запись';  end;  end  else begin  qz:=0;  rec:=true;  label2.visible:=true;  BitBtn1.caption:='Стой';  end;  end;  end. | | | | | | |
| *Выполнил* | | *Муханов А.* |  | *05.12.09.* | *КР 3706002 000009 ПЗ* | лист |
| *Проверила* | | *Енсепова Г.К.* |  | *07.12.09.* | 32 |
| *изм* | *лист* | *№ докум* | *Подпись* | *Дата* |
| **Приложение Г**  **Блок- схема к программе speaker1** | | | | | | |
| *Выполнил* | | *Муханов А.* |  | *05.12.09.* | *КР 3706002 000009 ПЗ* | лист |
| *Проверила* | | *Енсепова Г.К.* |  | *07.12.09.* | 33 |
| *изм* | *лист* | *№ докум* | *Подпись* | *Дата* |
| **Приложение Д**  **Блок – схема к программе speaker2** | | | | | | |
| *Выполнил* | | *Муханов А.* |  | *05.12.09.* | *КР 3706002 000009 ПЗ* | лист |
| *Проверила* | | *Енсепова Г.К.* |  | *07.12.09.* | 34 |
| *изм* | *лист* | *№ докум* | *Подпись* | *Дата* |
| **Список использованных источников**   1. Ахо, Альфред, В., Хопкрофт, Джон, Ульман, Джеффри, Д. Структуры данных и алгоритмы. М.: Издательский дом «Вильямс», 2001. 2. Дискретная математика для программиста / Ф. А. Новиков. – СПб.: Питер, 2002. 3. Климова Л. М. Pascal 7.0. Практическое программирование. Решение типовых задач. – М.: КУДИЦ – ОБРАЗ, 2000. 4. Н. Культин Turbo Pascal 7.0 – СПб: БХВ – Санкт – Петербург, 1998.. 5. Немлюгин С. А. Turbo Pascal^ практикум. – СПб.: Питер, 2002. 6. Программирование на языке Паскаль: задачник / под ред. Усковой О.Ф. – СПб.: Питер, 2003. 7. Фаронов В. В. Турбо Паскаль 7.0 – Москва, Издательство «Нолидж», 2001 8. Тессел Д. Стиль, разработка, эффективность, отладка и испытание программ. – М.: Мир, 1981 9. Докунина Т. К. Программирование и алгоритмические языки. 10. Васикова Н. Д., Тюляева В. В. Практикум по основам   программирования. Язык Паскаль., 1991. | | | | | | |
| *Выполнил* | | *Муханов А.* |  | *05.12.09*. | *КР 3706002 000009 ПЗ* | лист |
| *Проверила* | | *Енсепова Г.К.* |  | *07.12.09.* | 35 |
| *изм* | *лист* | *№ докум* | *Подпись* | *Дата* |
|  | | | | | | |
| *Выполнила* | |  |  |  |  |  |
|  | |  |  |  |  |
|  |  |  |  |  |
|  | | | | | | |
|  | |  |  |  |  |  |
|  | |  |  |  |  |
|  |  |  |  |  |
|  | | | | | | |
|  | |  |  |  |  |  |
|  | |  |  |  |  |
|  |  |  |  |  |
|  | | | | | | |
|  | |  |  |  |  |  |
|  | |  |  |  |  |
|  |  |  |  |  |
|  | | | | | | |
|  | |  |  |  |  |  |
|  | |  |  |  |  |
|  |  |  |  |  |
|  | | | | | | |
|  | |  |  |  |  |  |
|  | |  |  |  |  |
|  |  |  |  |  |
|  | | | | | | |
|  | |  |  |  |  |  |
|  | |  |  |  |  |
|  |  |  |  |  |
|  | | | | | | |
|  | |  |  |  |  |  |
|  | |  |  |  |  |
|  |  |  |  |  |
|  | | | | | | |
| *Выполнила* | |  |  |  |  | лист |
| *Проверила* | |  |  |  |  |
| *изм* | *лист* | *№ докум* | *Подпись* | *Дата* |
|  | | | | | | |
| *Выполнила* | |  |  |  |  | лист |
| *Проверила* | |  |  |  |  |
| *изм* | *лист* | *№ докум* | *Подпись* | *Дата* |