МИНОБРНАУКИ РОССИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ
«НОВОСИБИРСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ» (НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ, НГУ)

Кафедра компьютерных систем

Чуркин Вадим Сергеевич

Программа микроконтроллера для системы распознавания речевых сигналов.

МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ
по направлению высшего профессионального образования

230100.68 ИНФОРМАТИКА И ВЫЧИСЛИТЕЛЬНАЯ ТЕХНИКА

ФАКУЛЬТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

Тема диссертации утверждена распоряжением по НГУ № 4 от «11» января 2012 г.

Руководитель

Н. Г. Загоруйко
д. т. н., проф.

Новосибирск, 2013г.

Содержание

ВВЕДЕНИЕ

Эффект маскировки это особенность человеческого восприятия, согласно которой, для распознавания речевых сигналов используется не полное спектральное описание сигнала, а значения частот формантных максимумов. Данные полученные в целом ряде экспериментов, проведенных различными исследователями, подтверждают тот факт, что для принятия фонемного решения, человек использует не весь спектр, а лишь его локальные признаки (особенности)[7].

В психоакустике известны два типа маскировок: в частотной и временной области. При маскировке в частотной области: от каждой гармоники в спектре в сторону высоких частот (прямая маскировка), и низких (обратная маскировка), строятся кривые маскировки – затухающие и пропорциональные амплитуде маскирующей гармоники. Если амплитуда какой-либо гармоники в спектре оказывается ниже кривой маскировки, то тестируемая гармоника маскируется, т.е. полностью удаляется из спектрального описания. Аналогичные рассуждения и во временной области: маскерами становятся опять все гармоники в спектре, только кривые прямой и обратной маскировки строятся уже во временной области на постоянной частоте. Таким образом, мы получаем форматные максимумы спектрального описания сигнала.

На данный момент точные параметры кривых маскировок неизвестны, поэтому нахождение их представляет определенный интерес. Так же недостаточно изучено то, как использование этого эффекта повлияет на конечный результат анализа. Исходя из того, что человеком игнорируется замаскированная информация, можно предполагать, что качество распознавания не снизится, при снижении времени распознавания. Хотя может оказаться и так, что применение эффекта маскировки улучшит распознавание в целом, за счёт подавления “паразитной” информации.

Использование данное эффекта в сжатии звуковых файлов, позволяет снизить их объем в10 – 12 раз, без ощутимой потери качества[9].

1. ПОСТАНОВКА ЗАДАЧИ

В данной работе реализованы процедуры имитирующие эффекты маскировок. Разработаны механизмы сбора, модификации и визуализации речевых сигналов. Для обоих эффектов возможно изменение самой формы кривых маскирующих сигналов, а также изменение параметров этих кривых, что позволяет производить настройку максимально гибко. Предусмотрен вывод спектрограмм в хранилище (словарь), а так же разработаны методы сравнения спектрограмм со спектрограммами расположенными в словаре.

Так как теоретическая часть этого вопроса уже в определенной степени изучена, моей задачей было также реализация системы в аппаратном исполнении, для экспериментального подтверждения полученных результатов. Критические по времени процессы преобразования сигнала решено было реализовать на отдельном устройстве, а на персональный компьютер возложить задачи по анализу сигналов.

В качестве дополнительного устройства решено было взять, современный микроконтроллер для цифровой обработки сигналов семейства dsPic33F, архитектура которого специально разрабатывалась для работы с аналоговыми сигналами.

В целях повышения технологичности разработки новых приложений существуют т.н. стартовые наборы (Started kits) которые содержат встроенную аудио периферию и аппаратные средства для решения широкого круга задач.

Набор представляет собой полнофункциональную отладочную платформу, которая в своем составе имеет все необходимые аппаратные и программные средства для разработки и отладки приложений. Благодаря возможностям дополнительного микроконтроллера пользователи получают возможность разрабатывать и отлаживать аудио приложения, без дополнительных технических трудностей.

Для своей задачи я приобрел STARTER KIT FOR dsPIC® DIGITAL SIGNAL CONTROLLERS.

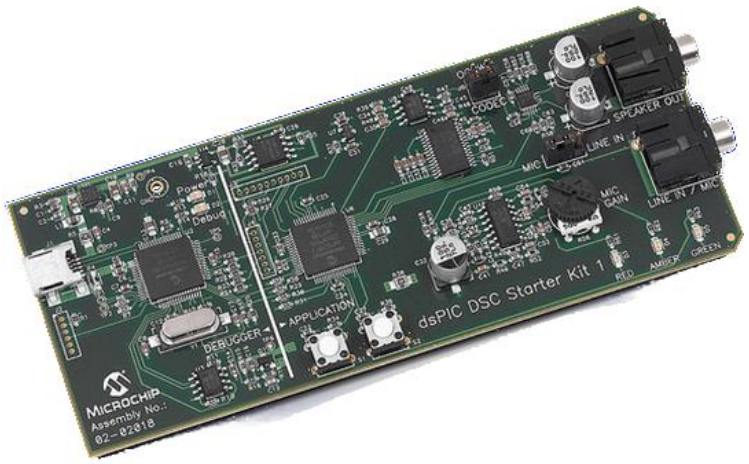


Рисунок 1. Внешний вид стартового набора

Как уже было сказано выше, часть функций системы реализовано на ПК, в большинстве своем, это функции которые требуют настройки и прямого контакта с пользователем, а обеспечение полноценного пользовательского интерфейса возможно лишь на персональном компьютере.

Список функций реализованных на ПК:

а). Маскировки (временная и частотная)

б). Поддержка работы с базой данных (Словарь)

в). Визуализация спектрограмм

г). Алгоритмы сравнения.

2. ОБРАБОТКА АНАЛОГОВОГО СИГНАЛА

2.1 Первоначальная обработка сигнала

Изначально акустический сигнал поступает на микрофон с граничной частотой не менее 10 КГц, и отношением сигнал/шум порядка 60 дБ (судя по внешнему виду и представленным на рынке образцам). Так как вся система, в текущем варианте, рассчитана на обработку речевого сигнала в пределах 8 КГц, то влияние характеристик микрофона на общее качество распознавания можно не учитывать. Микрофон подключен к Started kit'у, в котором с помощью джемпера(J7), выбрана схема смещения напряжения для микрофона, а не линейного входа – где смещения нет[2]. Коэффициент усилителя можно изменять переменным резистором в пределах от 3 до 23 дБ. Далее усиленный сигнал поступает на микросхему кодека, после которого преобразованный сигнал поступает по протоколу DCI на вход микроконтроллера для последующей обработки. Общая схема включения на рис 2.

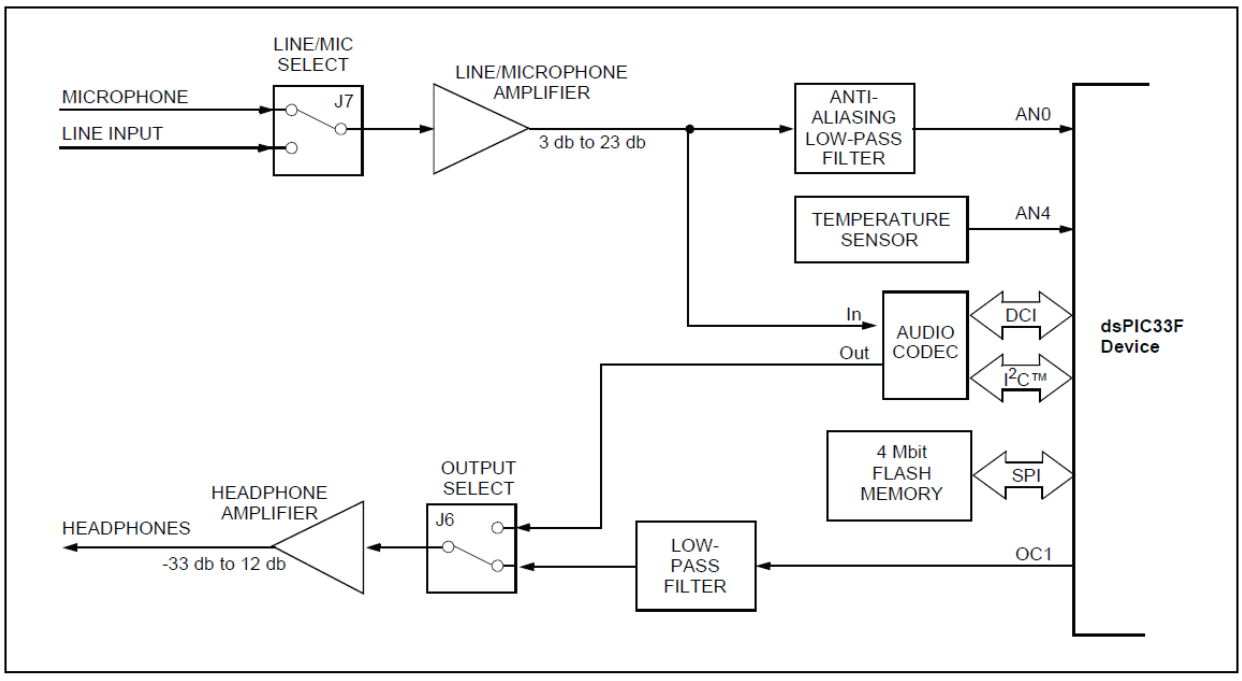


Рисунок 2. Схема обработки аналогового сигнала.

2.2 Кодек

Ранее был реализован механизм получения цифрового сигнала для микроконтроллера, посредством аналого-цифрового преобразователя (АЦП), но от такого варианта пришлось отказаться по двум причинам: на входе АЦП стоит сглаживающий низко-частотный фильтр с граничной частотой 3,3 КГц – что ограничивало рабочую полосу распознавания, а так же из-за частоты дискретизации при поступлении мощного сигнала, появлялись значительные нелинейные искажения [3] (на примере синусоиды – гармоника "обрастала" дополнительными "паразитными" составляющими), что негативно сказывалось на распознавании. Поэтому было решено сделать на кодеке. Кодек представляет собой микросхему (Wolfson WM8510), по заверению производителя это удачное решение для HI-End аудио приложений, VoIP (Voice over Internet Protocol) и цифровых телефонов[4]. Внутреннее устройство микросхемы кодека на рисунке 3.

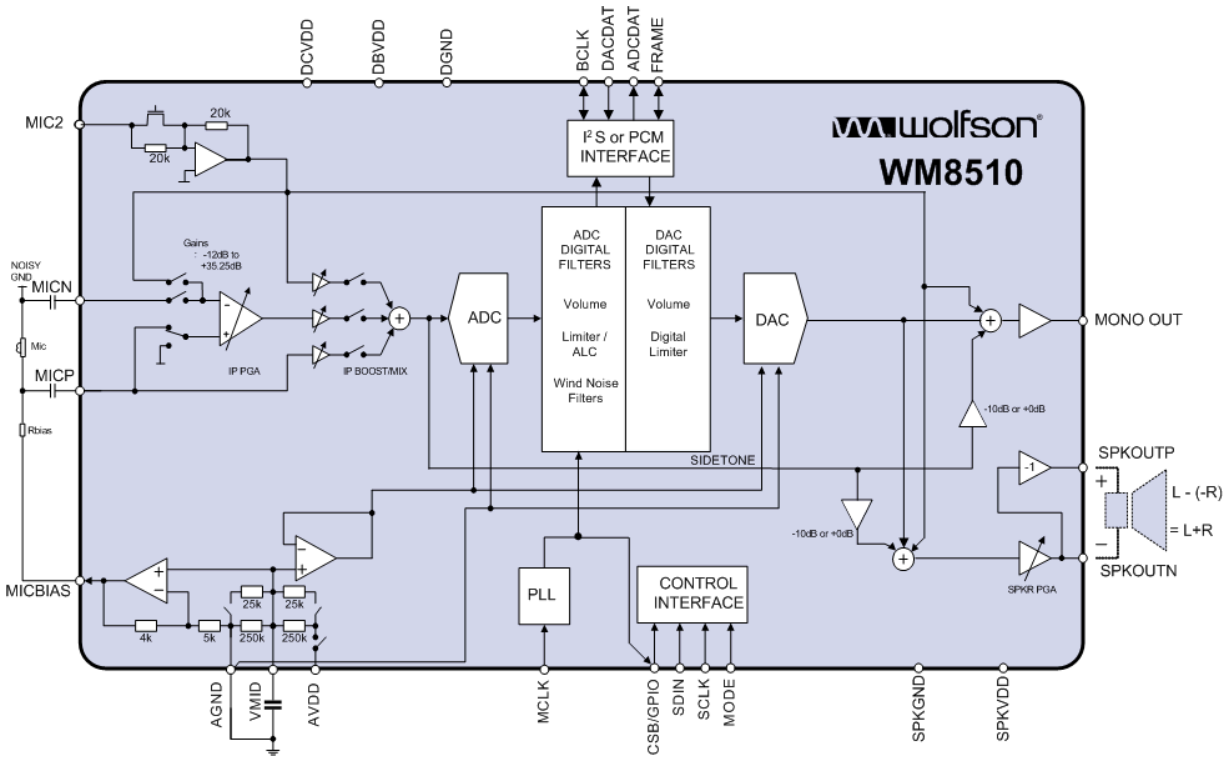


Рисунок 3. Микросхема кодека.

Вход кодека это выход предусилителя через разделительный конденсатор. Работу кодека можно настраивать в программе микроконтроллера посредством модуля Inter-

Integrated Circuit (I^2C). Настройка работы кодека осуществляется микроконтроллером по интерфейсу I^2C - производится запись во внутренние регистры кодека конфигурационной информации.

Список возможных параметров настройки включает в себя: протокол коммуникации, частота квантования, контроль громкости, контроль уровня, настройки фильтра и др. Описание информационных битов приводится в Datasheet WM8510 [4] от производителя. Большинство оптимальных значений конфигурации устанавливаются кодеком по умолчанию, при подачи питания на микросхему. Для своей задачи я отключил часть фильтров, установил частоту дискретизации внутреннего АЦП 16 КГц и отключил аттенюацию.

Кодек конвертирует входящий аудио сигнал в цифровой и отправляет его на модуль Digital Converted Interface (DCI) микроконтроллера.

Снятая частотная характеристика, показывает очень высокое качество обработки сигнала: после дискретного преобразования Фурье получается чистый спектр с низким уровнем паразитных составляющих, и спектрограмма самого речевого сигнала получается более отчетлива. Частота квантования 16 КГц, и после быстрого преобразования Фурье (БПФ), мы получаем спектр с максимальной частотой 8 КГц, что соответствует верхней граничной частоте нашего рабочего диапазона.

3. ОБРАБОТКА СИГНАЛА В МИКРОКОНТРОЛЛЕРЕ

3.1 Общее описание

Работа с сигналом начинается с того, что оцифрованный сигнал считываем с модуля DCI, с периодом 16 мсек в размере 256 отсчетов, получая для одного отсчета время квантования 62,5 мксек, при переводе в частоту 16 КГц, и заносим в буфер. В начало буфера заносим, значения с предыдущего отсчета, тем самым обеспечивая перекрытие в 7 мсек, необходимое для сохранения информации утерянной в связи с окном Хэмминга. Затем каждый отсчет перемножаем с коэффициентом окна Хэмминга и производим БПФ. После БПФ находим энергетический спектр, преобразуем отсчеты в однобайтовый формат и отправляем через модуль UART, на микросхему преобразования уровней COM порта и оттуда уже на COM порт компьютера.

На данный момент большинство программ микроконтроллера пишутся на языке Си, и отдельные критические по времени части на ассемблере. В моем случае программа реализована таким же способом.

Первый этап в работе программы микроконтроллера это настройка его составных модулей, инициализация констант и глобальных переменных. Далее идет бесконечный цикл (while(1)), внутри которого происходит проверка состояния, от состояния зависит то, какие операции будет выполнять микроконтроллер. Также в виде циклов реализованы задержки ожидания готовности того или иного модуля, к примеру готовность ни прием или получение данных (while(устройство не готово?)).

3.2 Быстрое преобразование Фурье

Ранее я использовал стандартный алгоритм БПФ[1] со своими модификациями, однако меня не удовлетворило время затрачиваемое при его работе, но как оказалось производитель микроконтроллеров уже предоставил библиотеку для ЦОС (цифровой обработки сигнала).

Библиотека DTS (digital signal processing) для семейств микроконтроллеров dsPic30F/33F. В ней реализованы наиболее используемые механизмы для цифровой обработки сигналов. Библиотека спроектирована с учетом конструкции микроконтроллеров данных семейств, такие как: разделенное адресное пространство, для максимальной эффективности. Функции в библиотеке написаны на Ассемблере. Для своей задачи я использовал быстрое преобразование Фурье, генератор коэффициентов для БПФ, генератор коэффициентов окна Хэмминга, функцию получения энергетического спектра из спектра с мнимыми и реальными составляющими.

3.3 Отправка данных с микроконтроллера

В связи с отсутствием поддержки протокола USB в данном микроконтроллере, для совместной работы с ПК пришлось вносить в плату дополнительные модификации и разрабатывать дополнительное устройство согласования. В качестве модуля передачи использовался Universal Asynchronous Receiver Transmitter (UART), данный модуль можно настроить для эффективной работы с ком портом [5]. Формат передачи основан на формате RS-232C и представлен на рисунке 4. Линия связи состоит из двух проводов. Каждый сегмент передается последовательно по одному элементу, после передачи сегмента идут три байта с выбранным мною значением 0x5a – характеризующие переход на следующий сегмент.

Настройки RS интерфейса в моей работе следующие:

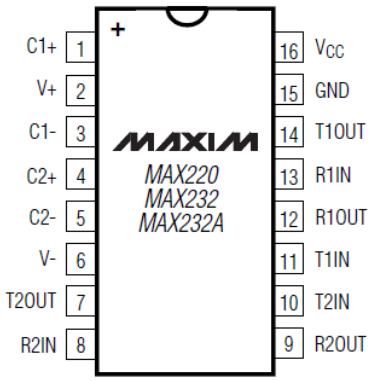
а). Скорость передачи может быть выбрана во время работы и имеет одно из трех значений: 38400 бод, 19200 бод, 9600 бод.
б). Восемь бит в пакете.
в). Контроль четности (паритет) – положительный.
г). Два стоп бита.

Рисунок 4. Формат асинхронной передачи RS-232C

4. УСТРОЙСТВО СОГЛАСОВАНИЯ С ПЕРСОНАЛЬНЫМ КОМПЬЮТЕРОМ

4.3 Предназначение

Данное устройство необходимо чтобы согласовать уровни напряжений микросхемы (0 – 3 В) с уровнями используемые в интерфейсе ком порта (минус 17 – 17 В), такие высокие уровни обусловлены целями помехозащищенности линий связи. Для согласования была применена стандартная микросхема MAX232, принципиальная схема включения приведена на рисунке 5. Принципиальная схема и прочие характеристики взяты из Datasheet на эту микросхему [6].



DIP/SO

CAPACITANCE (μF)					
DEVICE	C1	C2	C3	C4	C5
MAX220	0.047	0.33	0.33	0.33	0.33
MAX232	1.0	1.0	1.0	1.0	1.0
MAX232A	0.1	0.1	0.1	0.1	0.1

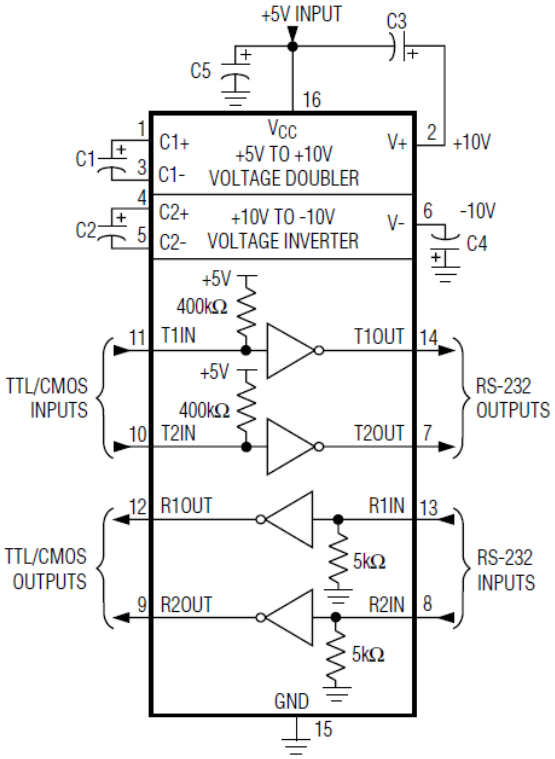


Рисунок 5. Схема включения микросхемы MAX232.

5. ПРОГРАММА НА ПЕРСОНАЛЬНОМ КОМПЬЮТЕРЕ

5.1 Общее описание

После оценки примерных требований к данной программе, было решено сделать ее на языке C#. Так как не предвиделось: ни очень высокого быстродействия, ни специфических механизмов работы.

Удобство работы, кроссплатформенность, широкая, поддержка делегатов (функция как объект), удобная генерация модульных тестов, высокая скорость разработки.

Подавляющие число задач системы распознавания, были реализованы в данной программе.

5.2 Получение данных.

После запуска программы каждую секунду происходит сканирование системы на присутствие COM портов, список доступных COM портов отображается в отдельном окне - в котором можно выбрать нужный и открыть для чтения. Интерфейс представлен на рисунке 9. Для правильной работы интерфейса приложения получение данных с COM порта производится в отдельном потоке.

Как уже ранее было сказано, данные с микроконтроллера мы получаем посегментно, определяя конец каждого сегмента по трем последовательным байтам со значением 0x5A. Через делегат (специфика многопоточности C#) сегмент – представляющий собой массив из 128 объектов типа double, заносится в общий массив спектрограммы.

Настройка COM порта клиента по умолчанию:

а). Положительный контроль четности.
б). Два стоп-бита.
в). Восемь бит в пакете.
г). По умолчанию – 38400 пакетов в секунду.

5.3 Реализация маскировок

На текущий момент кривые маскировок описаны следующими выражениями взятыми из[7]:

$$J_j = J_m \cdot \exp\left(-1{,}44 * Q^2 \left(\frac{\omega_m}{\omega_j} - 1\right)^2\right), \quad (1)$$

$$J_j = \beta \cdot J_m \left(1 - \alpha \cdot \left(1 - \exp\left(-\frac{k \cdot \Delta t}{\tau}\right)\right)\right), \quad (2)$$

$$J_j = \beta \cdot J_m \cdot \exp\left(-\frac{k \cdot \Delta t}{\tau}\right) \quad (3)$$

где

J_m – интенсивность маскера,

ω_m – частота маскера,

J_j- пороговая интенсивность тона,

ω_j – частота тона,

$Q = 1 \div 3$,

$k = j - m$,

$\alpha = 0{,}7 \div 0{,}8$,

$\tau = 50$ мсек ,

$\beta = 1$.

.

На рисунке 6 изображено трехмерное представление маскировок, а на рисунке 8 их влияние на гистограмму сигнала, по разработанному мною алгоритму. Функция маскировок приведена в приложении А.

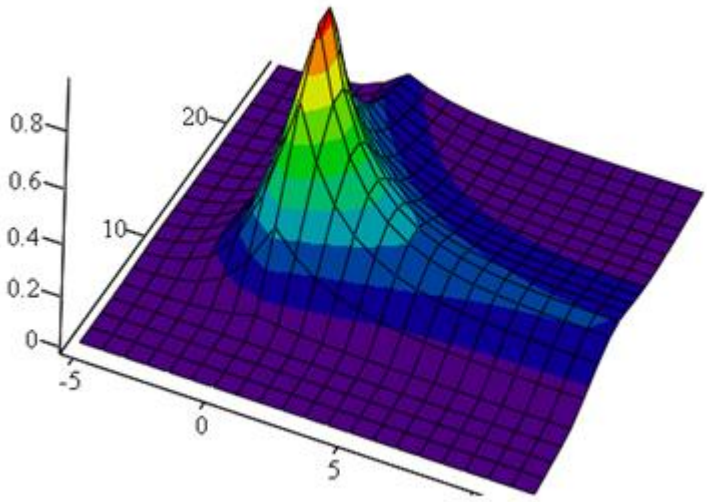


Рисунок 6. Трехмерное представление маскировок.

Настроить параметры маскировок в формулах, можно с использованием дополнительной формы, вызвать которую можно нажав на кнопку “ Настройки маскировок”. После нажатия на кнопку, появляется модальное одноименное окно, интерфейс которого представлен на рисунке 7. В этом окне помимо типа маскировок и установки параметров, так же присутствуют окна визуализации в которых отображается текущие формы маскировок с учетом выбранных параметров. Цифры на горизонтальной шкале представляют с тобой, для частотной маскировки отклонение от текущей частоты на количество октав согласно номеру от маскера в сторону высоких и низких частот, для временной количество стандартных временных интервалов (25 мсек) в вперед и назад по времени.

После того как выбраны нужные параметры, текущее окно закрывается и далее эти маскирующие параметры используются при проведении маскировок. В любой момент, если требуется, можно опять открыть это окно и произвести модификации.

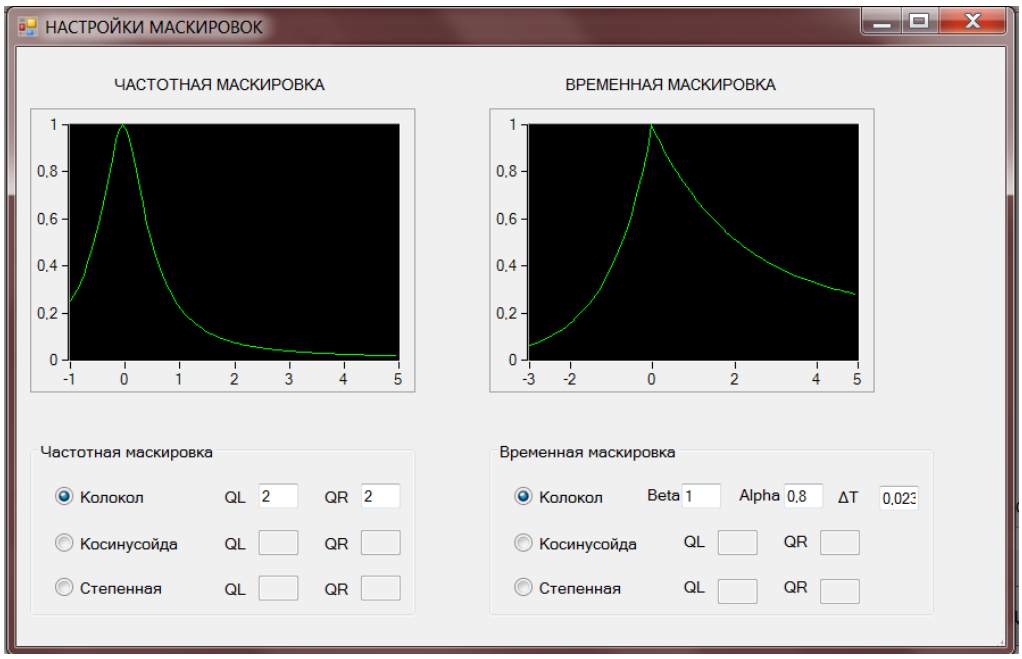


Рисунок 7. Спектрограмма слова “Помпа” до маскировки (слева) и после маскировки (справа).

5.4 Визуализация

Чтобы зрительно оценивать спектрограмму слова и влияния маскировок на нее, в программу требовалось добавить средство визуализации.

В текущей реализации спектрограмму можно наблюдать в специальном объекте от National Instruments – график интенсивности, в котором по вертикали (снизу вверх) расположены частоты, а по горизонтали время (справа - налево), энергия представлена в цветовом формате, шкала энергии разбита на две части, для поддержки двух типов режимов передачи:

а). Абсолютная – значения 0 - 128. Удобно производить анализ – стандартный тип шкалы.

б). Логарифмическая относительно полной шкалы (англ.- Decibels relative to full scale dBFS) – значения минус 100 - 0 дБ. Формат используемый в большинстве цифровых анализаторов – отсчет ведется относительно максимального значения прибора(0 дБ).

На этом графике можно так же видеть, как маскировка влияет на спектрограмму, на рисунке представлен пример этого влияния.

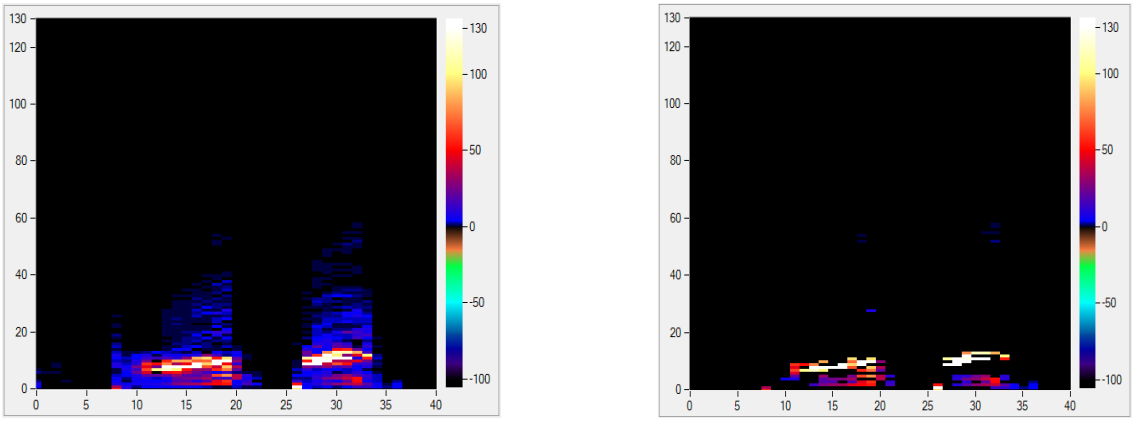


Рисунок 8. Спектрограмма слова “Помпа” до маскировки (слева) и после маскировки (справа).

После нажатия кнопки “СТАРТ”, начинает происходить прием данных. Новые данные заносятся в последний элемент массива, смещая все предыдущие, по шкале времени спектрограмма перемещается справа-налево. После нажатия на кнопку “СТАРТ” её название меняется на “СТОП”, и если нажать на неё еще раз, то прием данных прекращается, и можно производить различные виды анализов, маскировки и сохранять спектрограмму в базу данных. Название кнопки возвращается в исходное, и можно продолжать работу.

5.5 Работа со словарем

Словарь представляет собой файл базы данных MS Access. Реализация его в данном виде выгодна по ряду причин:

а). Удобство хранения: можно заменить, сохранить, удалить словарь как отдельный файл.

б). Удобство модификации и просмотра записей посредством MS Access.

в). Механизмы БД очень хорошо продуманны, что гарантирует быструю работу со словарем, даже очень большим.

г). Возможность сохранять в таблицу поля с дополнительной информацией, которая поможет в будущем сократить запросы поиска слова в очень большом словаре.

На данный момент таблица словаря состоит из следующих граф:

а). Индекс

б). Слово – в текстовом виде.

в). Двумерный сериализованный массив типа double. Спектрограмма слова, разбитая на частотные области.

OleDbConnection – класс поддержки работы с БД.

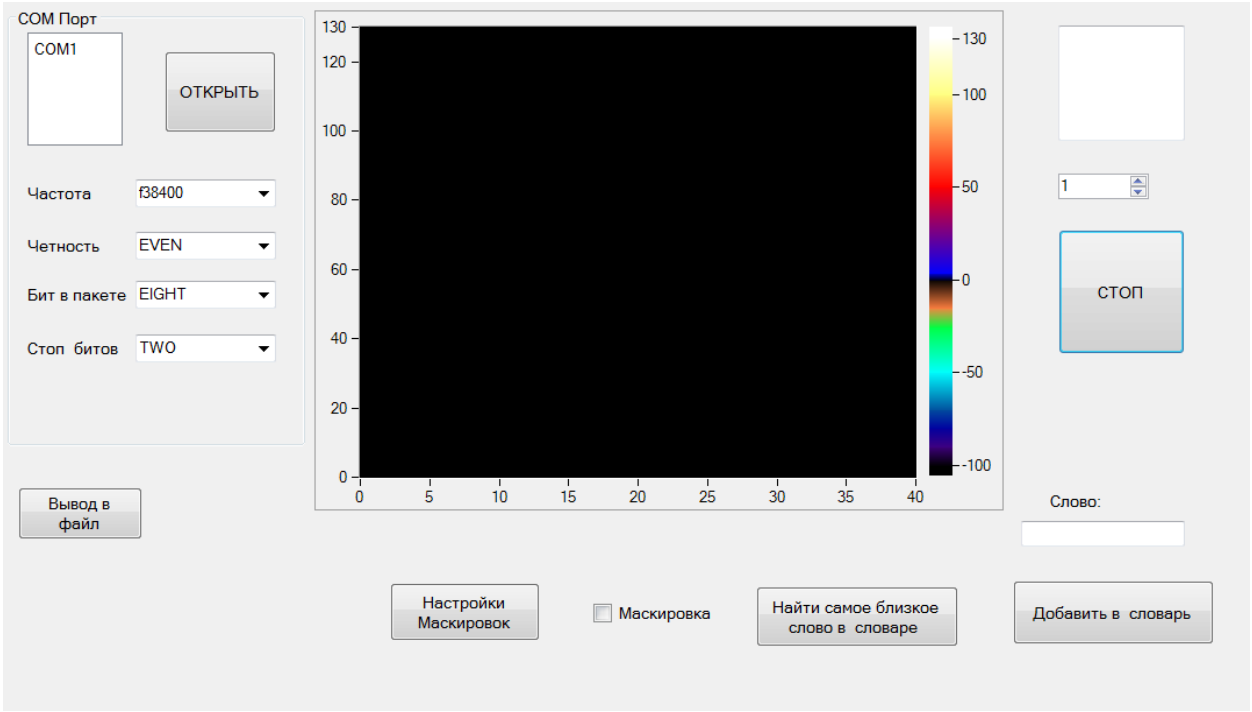


Рисунок 9. Интерфейс программы.

Изначально вся информация о слове хранится в оперативной памяти в виде спектрограммы, в 2-мерном массиве double. В объекте визуализации полностью представлены значения этого массива в каждый отдельный момент. Нажав кнопку “СТОП” мы останавливаем прием данных. Запись в БД можно осуществить, нажав на кнопку “Добавить в словарь”, предварительно ввести в поле выше текстовое наименование слова соответствующее текущей спектрограмме. После этого определяются границы слова, по разработанному мной алгоритму, чтобы сократить размер сохраняемого массива в БД. Какие либо еще преобразования к спектрограмме для сохранения, производить не желательно, так как мы можем потерять часть информации. После сокращения размеров слова, данный массив сериализуется – чтобы иметь возможность работать с ним как с данными (в нашей ситуации сохранять), и заносится в БД с помощью методов описанного выше класса. Процесс получения спектрограммы из БД следующий: методами класса OleDbConnection получаем сериализованный массив, десериализуем – получаем 2-мерный массив double, представляющий собой спектроограмму, и слово соответствующее этой спектрограмме.

5.5 Распознавание

В текущей реализации программы распознавание происходит после остановки приема данных с микроконтроллера, кнопка "СТОП". Предполагается, что слово для распознавания было произнесено почти перед остановкой. Спектрограмма распознаваемого слова отображается в окне визуализации. Представляет интерес то, как влияет маскировка на распознавание, включить маскировку можно, поставив галочку напротив метки “Маскировка”. После этого в окне визуализации отобразится спектрограмма с маскировкой, и к спектрограмме каждого слова взятого из словаря, предварительно, будет применяться маскировка.

Сам процесс распознавания состоит из следующих этапов:

а). Текущая спектрограмма разбивается по частотным полосам, рекомендованным специалистам по вокодерной технике, они имеют следующие значения (в герцах): 150 – 400 , 400 – 640, 640 – 1040, 1040 – 1520, 1520 – 2200, 2200 – 3040, 3040 – 4200, 4200 – 7000, 7000 – 10000.

б). Затем находится максимальный интеграл под кривой маскера для каждой частотной полосы, значение которого приписывают текущей частотной полосе.

в). Из словаря берем спектрограмму слова, которую аналогично разбиваем на частотные полосы. Как уже было отмечено выше, если задан режим с маскировками, то к спектрограмме слова применяем маскировку.

г). Опционально проводим дополнительные механизмы фильтрации по длине слов. Не прошедшие эти механизмы слова в общем анализе не учувствуют.

д). Находим меру сходства между сегментами двух спектрограмм. При расчёте меры сходства, каждый сегмент одного слова, сравнивается с каждым сегментом другого слова. После этого этапа, получаем массив размером m на n, где m – количество сегментов в первом слове, n- количество сегментов во втором слове. Мера сходства рассчитывается следующим образом:

$$d_{mn} = \frac{\alpha^2}{\alpha^2 + \rho_{mn}{}^2} \qquad (4)$$

где

d_{mn}- Мера близости между двумя сегментами,

α- экспериментально выбираемая константа,

ρ_{mn}- расстояние между сегментами m и n.

Расстояние между сегментами в текущем варианте определяется так:

$$\rho_{mn}{}^2 = \sum_{i=1}^{p}\left(ln\left(\frac{\tilde{E}_i^{(m)}}{E_0^{(m)}}\right) - ln\left(\frac{\tilde{E}_i^{(n)}}{E_0^{(n)}}\right)\right)^2 \quad (5)$$

где

p – количество частотных полос;

$$E_0^{(m)} = \sum_i E_i^{(m)}, \quad (6)$$

$$\tilde{E}_i^{(m)} = \sum_{j=li}^{ri} \bar{E}_i^{(m)}; \quad (7)$$

$$\bar{E}_i^{(m)} = \begin{cases} E_i^{(m)}, E_i^{(m)} \neq 0 \\ \hat{E}_i^{(m)}, E_i^{(m)} = 0 \end{cases}; \quad (8)$$

где

$\hat{E}_i^{(m)}$ – кривая маскировки на i - й частотной полосе,

$E_i^{(m)}$ – Амплитуда маскирующей гармоники,

ri – правая граница i - й частотной полосы,

li– левая граница i - й частотной полосы;

е). После применяем динамический алгоритм, который позволяет сравнивать спектрограммы слов с неоднородной структурой[8]. Рисунок 10 дает представление то, как происходит сравнения слов. В двумерную таблицу заносим меру сходства двух слов по принципу каждый с каждым, и нам требуется заполнить все пересечения внутри этой таблицы, чтоб найти максимальную величину сходства. Самые правые и нижние пересечения заполняются нулями. Затем по формуле (9), начинаем последовательно заполнять пересечения от нижнего правого угла. Моя реализации данного алгоритма приведена в приложении Б.

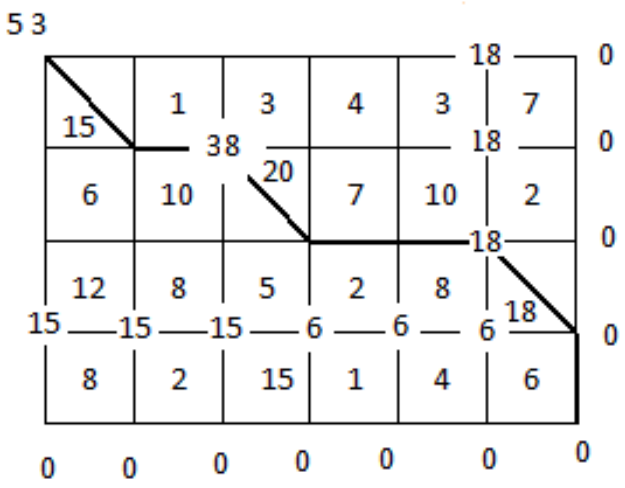


Рисунок 10. Пример работы алгоритма динамического алгоритма.

$$A(i,j) = max\begin{cases} A(i,j+1) \\ A(i+1,j+1) + d(i,j) \\ A(i+1,j) \end{cases} \quad i = I, I-1, \dots, 1;\ j = J, J-1, \dots, 1. \qquad (9)$$

где

$A(i,j)$ - Пересечение i- го столбца, с j- й строкой,

d- Мера близости между двумя сегментами.

ж). Проводя эти операции с каждым словом в словаре, находим такое, для которого мера близости окажется максимальной.

После того как найдется слово с максимальной мерой близости, отображается диалоговое окно в котором выводится величина меры близости и выбранное слово, вид диалогового окна представлен на рисунке 11.

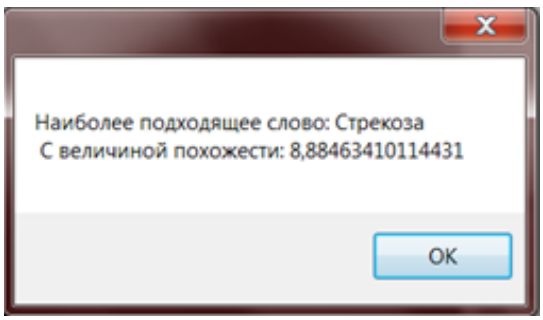


Рисунок 11. Диалоговое окно с результатом.

ЗАКЛЮЧЕНИЕ

На данный момент система может эффективно распознавать словарь, состоящий из порядка десяти непохожих друг на друга слов с зависимостью от диктора. Максимальные возможности алгоритмов, к сожалению, оценить не удалось, так как они ограниченны самым слабым звеном в системе, которым, как уже выше было указанно, является интерфейс микроконтроллер – персональный компьютер. Из-за невозможности передать большие объемы данных – спектрограмма получилась низкого разрешения, что не позволяет детали слов.

Но точно можно заключить то, что использование эффекта маскировки не понижает качества анализа, также возможно повышение анализа за счёт удаления паразитных составляющих сигнала. Стоит отметить, что повышение качества анализа маскированного сигнала, по сравнению с немаскированным, зависит от типа используемого анализа.

Помимо распознавания данный программный комплекс можно использовать для экспериментального вывода идеальных параметров для маскирующих кривых, с целью распознавания речевых сигналов.

Чтобы повысить качество работы, можно выделить следующие направления: прежде всего это конечно увеличить разрешение данных, внести в базу данных словаря новые характеристики слов, включения дополнительных алгоритмов сравнения.

Хотя такой размер словаря весьма мал, но он уже позволяет её использовать в системах голосового управления с низкими рисками: к примеру, включать свет в определённых помещениях в зависимости от речевых команд, или в условиях, где у оператора заняты руки.

Литература

1. Курячий М.И. Цифровая обработка сигналов: Учебное пособие для вузов с грифом УМО.–Томск: ТУСУР, 2009.–190 с.
2. Microchip Technology Inc. dsPIC33F MPLAB Starter Kit for dsPIC® Digital Signal Controllers – USA Chandler, 2005– 42c.
3. Яковлев А. Н. Радиотехнические цепи и сигналы. – Новосибирск НГТУ ИНФРА-М, 2003 -348с.
4. Wolfson Microelectronics plc WM8510 Mono CODEC with Speaker Driver Data Sheet – United Kingdom Edinburgh 2008 – 82c.
5. А.Ю. Кузьминов Интерфейс RS232. Связь между компьютером и микроконтроллером. – М, ДМК Пресс, 2004 – 320с.
6. Maxim Integrated Products +5V-Powered, Multichannel RS-232 Drivers/Receivers Data Sheet – San Jose, USA 2010 – 38c.
7. В.Г. Лебедев Эффект маскировки и автоматический анализ речевых сигналов. Статья. – Вычислительные системы, выпуск 61 - 1975 г.
8. Загоруйко Н.Г. Когнитивный анализ данных. Ин-т математики им. Соболева. – Новосибирск : Изд-во Гео, 2013. – 186 с.
9. Журнал «Звукорежиссер» 2000г Выпуск 3.
10. Предко М. Справочник по PIC-микроконтроллерам. – М, ДМК Пресс, 2002 – 416с.
11. Microchip Technology Inc. dsPIC33F Family Data Sheet – USA Chandler, 2005– 418c.
12. Марченко М.В. Ульяновск: Учебное пособие по дисциплине "Устройства на PIC-процессорах" УлГТУ, 2007. - 66 с
13. Л.В.Бондарко Фонетика современного русского языка, издательство С.-Петербургского Университета, 1998г. СПбГУ, 2006 (изд.2)

Приложение А
(справочное)

```
/// <summary>
/// Функция маскировки
/// </summary>
/// <param name="samples">Исходная спектрограмма,на выходе она замасикированая   </param>
/// <param name="freqDimSize">Размер спектрограммы в частотной области</param>
/// <param name="timeDimSize">Размер спектрограммы во временной области</param>
/// <param name="HiFreqSpectrum">Максимальная частота  в спектре</param>
/// <param name="FSettings">Настройки частотных маскировок</param>
/// <param name="TSettings">Настройки временных маскировок</param>
/// <param name="Scale">Тип исспользуемой шкалы</param>
static public void MaskiRTimeAndFreq(double[,] samples, int freqDimSize, int timeDimSize,
double HiFreqSpectrum, MaskFreqSettings FSettings, MaskTimeSettings TSettings, int Scale)
{
    int t, f, fj, tj;
    int ATTENUATION_LIMIT = 50;         //Лимит до каких пор еще идет рассмотрение
сигнала
    int VALUE_LIMIT = 1;                //если в спектрограме значение ниже - идем дальше

    double TestVal,SampleVal;
    double attF, attT;

    //Основной цикл перебора времени
    for (t = 0; t < timeDimSize; t++)
    {
        //Основной цикл перебора частот
        for (f = 0; f < freqDimSize; f++)
        {
            SampleVal = samples[t, f];

            switch (Scale) {
                case 0:
                    SampleVal = samples[t, f];
                    break;
                case 1:
                    SampleVal = samples[t, f] + 100;
                    break;
                }

            if (SampleVal <= VALUE_LIMIT) continue;  //Мал - пропускаем для экономии

            ////////////////////Назад по частотам/////////////////////////////////////
            fj = 0;
            while (f - fj >= 0)
            {//до тех пор пока не достигнем нуля
                //Стандартный колокол
                attF = AnalysisMask.KolokolF(FSettings.QL, ((double)f + 1.0) / ((double)f
- (double)fj + 1.0));       //ПОлучаем чнаячение после колокола чвстоты

                //Инвертируем
                if (1 / attF > ATTENUATION_LIMIT) break; //..если ослабление не станет
сильно  большим

                //Назад по времени//////////////////////////////
                tj = 0;
                while (t - tj >= 0)
                {
                    attT = AnalysisMask.KolokolTL(TSettings.Beta, -1.0 * (double)tj,
TSettings.DeltaT);
```

```
                    if (1 / (attT*attF) > ATTENUATION_LIMIT) break;              //..или
ослабление не станет сильно  большим
                    TestVal = SampleVal * attF * attT;
       //находим  уровень маскировки  на текущей частоте по отношению к  исследуемой

                    switch (Scale)
                    {
                        case 0:
                                if (samples[t - tj, f - fj] < TestVal) samples[t - tj, f
- fj] = 0; // если уровень маскировки выше выбранной гармоники маскируем ее(удаляем)

                            break;
                        case 1:
                            if (samples[t - tj, f - fj] + 100 < TestVal) samples[t - tj,
f - fj] = -100; // если уровень маскировки выше выбранной гармоники маскируем ее(удаляем)

                            break;
                    }

                        tj++;
                }
                //Вперёд по времени//////////////////////////////
                tj = 1;
                while (t + tj < timeDimSize) //Не выше последней
                {//Возможно подправить
                    //Стандартный колокол
                    attT = AnalysisMask.KolokolTR(TSettings.Beta, TSettings.Alpha,
(double)tj, TSettings.DeltaT);

                    if (1 / (attT * attF) > ATTENUATION_LIMIT) break;
                    TestVal = SampleVal * attF * attT;             //находим  уровень
маскировки  на текущей частоте

                    switch (Scale)
                    {
                        case 0:
                            if (samples[t + tj, f - fj] < TestVal) samples[t + tj, f -
fj] = 0; // если уровень маскировки выше выбранной гармоники маскируем ее(удаляем)

                            break;
                        case 1:
                            if (samples[t + tj, f - fj] + 100 < TestVal) samples[t + tj,
f - fj] = -100; // если уровень маскировки выше выбранной гармоники маскируем ее(удаляем)

                            break;
                    }

                    tj++;
                }

                fj++;   //След частота
            }


            ////////////////////Вперед по частотам//////////////////////////////////////
            fj = 1;
            while (f + fj < freqDimSize)
            {//до тех пор пока не достигнем нуля или... ( Возможно подправить)
                //Стандартный колокол
                attF = AnalysisMask.KolokolF(FSettings.QR, ((double)f + 1.0) / ((double)f
+ (double)fj + 1.0));       //Получаем значение после колокола частоты

                //Инвертируем
```

```
                if (1 / attF > ATTENUATION_LIMIT) break;                    //..если
ослабление не станет сильно  большим

                //Назад по времени//////////////////////////////
                tj = 0;
                while (t - tj >= 0)
                {
                    attT = AnalysisMask.KolokolTL(TSettings.Beta, -1.0 * (double)tj,
TSettings.DeltaT);

                    if (1 / (attT * attF) > ATTENUATION_LIMIT) break;
       //..или ослабление не станет сильно  большим
                    TestVal = SampleVal * attF * attT;
       //находим  уровень маскировки  на текущей частоте по отношению к  исследуемой

                    switch (Scale)
                    {
                        case 0:
                            if (samples[t - tj, f + fj] < TestVal) samples[t - tj, f +
fj] = 0; // если уровень маскировки выше выбранной гармоники маскируем ее(удаляем)
                            break;
                        case 1:
                            if (samples[t - tj, f + fj] + 100 < TestVal) samples[t - tj,
f + fj] = -100; // если уровень маскировки выше выбранной гармоники маскируем ее
(удаляем)
                            break;
                    }
             tj++;
                }

                //Вперёд по времени//////////////////////////////
                tj = 1;
                while (t + tj < timeDimSize) //Не выше последней
                {//Возможно подправить
                    //Стандартный колокол
                    attT = AnalysisMask.KolokolTR(TSettings.Beta, TSettings.Alpha,
(double)tj, TSettings.DeltaT);

                    if (1 / (attT * attF) > ATTENUATION_LIMIT) break;
                    TestVal = SampleVal * attF * attT;              //находим  уровень
маскировки  на текущей частоте

                    switch (Scale)
                    {
                        case 0:
                            if (samples[t + tj, f + fj] < TestVal) samples[t + tj, f +
fj] = 0; // если уровень маскировки выше выбранной гармоники маскируем ее(удаляем)
                            break;
                        case 1:
                            if (samples[t + tj, f + fj] + 100 < TestVal) samples[t + tj,
f + fj] = -100; // если уровень маскировки выше выбранной гармоники маскируем ее(удаляем)
                            break;
                    }

                    tj++;
                }

                fj++;   //След частота
            }
        } //Конец основного цикла перебора частот
    }//Конец основного цикла перебора времени
}
```

Приложение Б
(справочное)

```
/// <summary>
/// "Динамо" алгоритм  из книги           - проверено модульным тестом! работает
/// </summary>
/// <param name="SimilMass">Массив совместной похожести двух сигналов   </param>
/// <param name="MassXSize">размер по x (длинна 1го сигнала)     </param>
/// <param name="MassYSize">размер по y (длинна 2го сигнала)    </param>
/// <returns>Коэффициент похожести    </returns>
static public double DinamoAlgorithm(double[,] SimilMass, int MassXSize, int MassYSize)
{
    double[,] NetMass = new double[MassYSize + 1, MassXSize + 1]; //Массив для  хранения
значений функции сетки (должен быть заполнен нулями, надеюсь, так  по  умолчанию) - да
так  и есть

    double MaxSimilarity = 0,buff;


    for (int j = MassYSize - 1; j >= 0; j--)
    {
        for (int i = MassXSize - 1; i >= 0; i--)
        {
            buff = Math.Max(
                NetMass[j + 1, i + 1] + SimilMass[j,i],
                Math.Max(NetMass[j + 1, i], NetMass[j, i + 1])
                );

            NetMass[j, i] = buff;

            if (buff > MaxSimilarity)
            {
                MaxSimilarity = buff;
            }

        }
    }

        return MaxSimilarity;            //Возвращаем меру сходства
}
```