#### ***Министерство образования и науки Российской Федерации***

#### *Федеральное государственное бюджетное образовательное учреждение высшего образования*

#### ***«Владимирский государственный университет***

#### ***имени Александра Григорьевича и Николая Григорьевича Столетовых»***

Институт информационных технологий и радиоэлектроники

Кафедра радиотехники и радиосистем

УТВЕРЖДАЮ

Заведующий кафедрой

\_\_\_\_\_\_\_\_\_\_\_\_\_\_ О.Р. Никитин

«\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 20\_\_\_

Основание:

решение кафедры

от «\_\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 20\_\_\_\_

**МЕТОДИЧЕСКИЕ УКАЗАНИЯ К ВЫПОЛНЕНИЮ ЛАБОРАТОРНЫХ РАБОТ ПО ДИСЦИПЛИНЕ «СИСТЕМЫ МОБИЛЬНОЙ, ТРАНКИНГОВОЙ И СОТОВОЙ СВЯЗИ» И «СИСТЕМЫ МОБИЛЬНОЙ РАДИОСВЯЗИ» СПЕЦИАЛЬНОСТЕЙ 11.03.01 «РАДИОТЕХНИКА» и 11.03.02 «ИНФОКОММУНИКАЦИОННЫЕ ТЕХНОЛОГИИ И СИСТЕМЫ СВЯЗИ»**

Составитель

Е.А. Архипов,

Владимир 2018

УДК 621.396

ББК 32.84

Рецензент

к.т.н., профессор

Колесник Г.П

Методические указания к выполнению лабораторных работ по дисциплине «Системы мобильной, транкинговой и сотовой связи» и «Системы мобильной радиосвязи» специальностей 11.03.01 «радиотехника» и 11.03.02 «инфокоммуникационные сети и системы связи»; составитель Архипов Е.А., 2018.- 75с.

Даются описание и методические указания по выполнению лабораторных работ по дисциплинам «системы мобильной, транкинговой и сотовой связи» и «системы мобильной радиосвязи» специальностей 11.03.01 «радиотехника» и 11.03.02 «инфокоммуникационные сети и системы связи». Приводятся рекомендации по оформлению и защите работ. Библиогр.: 9 назв.

УДК 621.396

ББК 32.84

Оглавление

[ВВЕДЕНИЕ 3](#_Toc514769697)

[1. ОБЩИЕ ПРИНЦИПЫ РЕАЛИЗАЦИИ МОДЕЛИ ПРИЕМОПЕРЕДАТЧИКА 5](#_Toc514769698)

[1.1 Состав программы 5](#_Toc514769699)

[1.2 Принципы оверлейной организации программы 7](#_Toc514769700)

[2.1 Инициализация приложения 8](#_Toc514769701)

[2.2 Обработка прерывания 48534D20-INT 2F 9](#_Toc514769702)

[2.3 Функции установки/получения адресов переменных и структур данных и сброса системы 11](#_Toc514769703)

[2.4 Функция запуска оверлейного модуля 12](#_Toc514769704)

[2.5. Выделение и освобождение памяти 13](#_Toc514769705)

[2.6 Функции получения/установки флагов и параметров 14](#_Toc514769706)

[3 МОДУЛЬ ИНИЦИАЛИЗАЦИИ СИСТЕМЫ ПОДДЕРЖКИ GSM 16](#_Toc514769707)

[3.1 Модуль gsm\_set 16](#_Toc514769708)

[3.3 Ввод данных глобального назначения 17](#_Toc514769709)

[3.4.1 Функция Make\_symbols 18](#_Toc514769710)

[3.4.2 Функция Make\_Previous 19](#_Toc514769711)

[3.4.3 Функция Make\_next 20](#_Toc514769712)

[3.4.4. Функция Make\_Start 21](#_Toc514769713)

[3.4.5 Функция Make\_stops 21](#_Toc514769714)

[4 ПРИЕМ И ПЕРЕДАЧА БЕЗ КОДИРОВАНИЯ 22](#_Toc514769715)

[4.1.1 Функция Gendata 23](#_Toc514769716)

[4.3 Приемник 35](#_Toc514769717)

[4.3.1 Фунция MaFi 35](#_Toc514769718)

[4.3.4 Функция DEMux 40](#_Toc514769719)

[5.1.1 Общие принципы кодирования 41](#_Toc514769720)

[5.1.2 Функция Channel\_enc 43](#_Toc514769721)

[5.2.1 Функция DeInterleave 45](#_Toc514769722)

[6.1 Модули демонстрации образования сигналов 49](#_Toc514769723)

[6.2 Модули вывода переменных 59](#_Toc514769724)

[6.3 Модули тестирования и пошагового режима 60](#_Toc514769725)

[Список используемой литературы 61](#_Toc514769726)

## 

## ВВЕДЕНИЕ

Выполнение и защита лабораторных работ является важным этапом изучения студентами соответствующих дисциплин, позволяющим закрепить и проверить полученные в лекционном курсе знания.

Спецификой дисциплин «Системы мобильной, транкинговой и сотовой связи» и «Системы мобильной радиосвязи» является системный характер изучаемых курсов и быстрый прогресс развития этой области.

Большое разнообразие образцов систем мобильной радиосвязи, их малые габариты и отсутствие защиты блоков от выхода из строя при неправильном включении делают проблематичным аппаратную реализацию лабораторных работ. Дополнительным ограничением является частотный диапазон функционирования устройств, требующий применения очень дорогой измерительной техники.

Это определяет проведение курса лабораторных работ на компьютерах, заменяя исследование реальной аппаратуры исследованием компьютерной модели, позволяющей студентам изучить основные приемы обработки сигналов в приемо-передающей аппаратуре сотовой связи.

При выборе объекта моделирования рассматривались варианты приемо-передающей аппаратуры стандартов второго поколения GSM-900 и IS-95. В силу ограниченности времени на проведения работ реализован лишь первый вариант. Однако он позволяет студентам изучить основные процедуры обработки сигналов используемые в системах сотовой связи третьего и четвертого поколений. К ним можно отнести процедуры цифровой модуляции, адаптивной согласованной обработки, кодирования и декодирования сигналов, перемежения, деперемежения. Модель позволяет изучит проблемы функционирование системы в условиях действия аддитивных шумов и замираний сигналов вызванных эффектом многолучевого распространения.

В методические указания входят четыре лабораторных работы:

1) Изучение модели GSM по документации без участия исполняемых файлов. Целью этой работы являетсяподготовка студента к последующим лабораторным работам;

2) Изучение процесса модуляции и передачи без кодирования. Целью данной работы является изучение свойств сигналов с гауссовской частотной манипуляцией с минимальным сдвигом частоты и адаптивной согласованной обработки. Получение характерных осциллограмм квадратурных составляющих сигнала иосциллограмм оценки импульсной характеристики канала связи без шумов по тестовой последовательности.

3)Изучение процедур каскадного блочно- сверточного кодирования/декодированиядекодирования сигналов. Целью данной работы является изучение методов блочного и сверточного кодирования. Изучение алгоритмаВитерби при декодировании сверточных кодов и при демодуляции сигналов.

4)Исследование модели передачи сигналов по каналу связи с шумами и многолучевым распространением. Целью этой работы является изучение осциллограмм сигналов при различных свойствах канала связи и исследование зависимости количества ошибок передачи от свойств и вида канала связи. При этом рассматривается влияние на вероятность ошибки отношения сигнал/шум в канале связи, исследуется аналогичная зависимость в условиях многолучевого распространения и межсимвольной интерференции.

В программе предусмотрен модуль тестирования, который служит для автоматической оценки результатов выполнения лабораторных работ. Вход в него защищен паролем, который задаваемым при установке программы преподавателем.

Вопросы для этой программы хранятся в кодированном виде в файлах questx.que, где x=номеру лабораторной работы. Каждый файл содержит 80 вопросов. Принцип построения текстового файла вопросов прост и становится понятен при просмотре файла. В пакете есть специальная утилита quest.exe, преобразующая файл из текстового формата в формат que. Эта утилита вместе с исходными файлами вопросов находится в установочном пакете questtxt. Преподаватель может изменять вопросы по своему рассмотрению или вводить новые, после чего обязательно создать соответствующий файл que заново.

**Внимание:**

1. при создании текстового файла вопросов обязательно пользоваться кодировкой DOS, для чего использовать соответствующий редактор. Если это условие не будет выполнено, программа тестирования будет выдавать вопросы неправильно.
2. Ни один вопрос или ответ не должен быть длиннее 48 символов, в противном случае вывод на экран будет неудобочитаемым.
3. Не рекомендуется использовать в строках стрелки, градусы и другие символы, отображение которых целиком зависит от операционной системы; их отображение может отличаться от отображения в редакторе.

При запуске модуля определяется состояние таймера. По младшему байту выбирается один из 16 вариантов 5-вопросной последовательности. Предугадать этот процесс невозможно.

На пять вопросов студенту дается 180 секунд. Если ответ на вопрос правильный, надпись об оставшемся времени приобретает зеленый цвет, иначе – красный. Более того, при оставшемся времени менее минуты надпись начинает мигать.

Если студент дал 4 и более правильных ответов, защита принимается, в противном случае нет. Если время, отведенное на защиту, вышло, это расценивается как 0 правильных ответов. Время контролируется по часам реального времени, поэтому попытка приостановить процесс приведет только к замиранию показаний времени, но не к его остановке.

Студент имеет две попытки защиты лабораторной работы с помощью тестера на одном занятии. Если работа не защищена с четырех попыток, вступает в силу традиционный метод защиты преподавателю.

В распоряжении преподавателя имеется файл install.exe. При установке преподаватель вводит лицензионный ключ (этот ключ задается при генерации установочного пакета при помощи утилиты Create). Если ключ неверен, установка прекращается.

Далее происходит распаковка файлов пакета. Это файлы пакетов исходных кодов, исполняемых файлов, текстовых вопросов и документации. Эти пакеты распаковываются аналогичным образом, причем распаковка исполняемых кодов включает запись идентификаторов процессора и установку пароля для GSMTest.

Попытка скопировать установленную систему GSM на другой компьютер приведет к выдаче соответствующего сообщения при инициализации.

Для выполнения любых операций прежде всего необходимо инициализировать систему поддержки GSM, то есть выбрать соответствующий пункт меню. Далее следует руководствоваться меню и запросами программы.

## 1 ОБЩИЕ ПРИНЦИПЫ РЕАЛИЗАЦИИ МОДЕЛИ ПРИЕМОПЕРЕДАТЧИКА

## 1.1 Состав программы

За основу разработки модели в идеологическом аспекте взята модель в системе MATLAB. Эта модель была тщательно проанализирована.

Все принципы кодирования / декодирования и модуляции / демодуляции реализованы без существенных изменений. Зато генерация блока данных, эмулятор канала, функции контроля передачи значительно усовершенствованы. В программный комплекс добавлены также модули пошаговой работы, учебного контроля и демонстрации процесса передачи. Значительно усовершенствованы также головные модули передачи.

Итак, комплекс программ состоит из следующих модулей:

1. основного, содержащего интерфейс пользователя, процедуры управления оверлейными модулями, главную процедуру обмена адресами блоков данных через вектор прерывания, функции элементарной обработки данных;
2. модуля инициализации, производящего проверку окружения программы, ввод исходных данных и запуск генератора переменных и таблиц;
3. модуля учебного контроля, производящего опрос студента, защищающего лабораторную работу, и выдачу оценки за защиту работы;
4. модуля отладки, позволяющего вводить команды в интерактивном режиме;
5. 9 модулей демонстрации;
6. 2 головных модулей, осуществляющих запуск процесса приема/ передачи
7. восемнадцати функций работы с GSM, в которые входит приемник, передатчик, канал связи и процедуры начальной конфигурации;
8. модули создания и распаковки установочных пакетов программы;
9. модуля подготовки файлов вопросов для тестирования;
10. файлов вопросов для тестирования.
11. документации на программу.

Общая структурная схема модели приведена на рисунке 1.1.

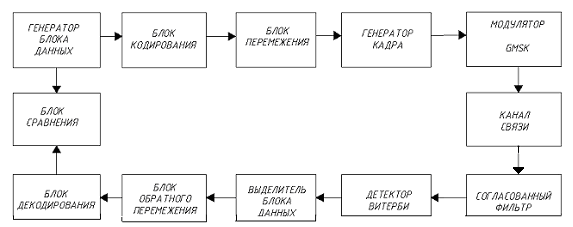


Рисунок 1.1- Общая структурная схема модели GSM

Программа написана по принципу структурного программирования. Внешние функции являются фактически самостоятельными модулями, которые могут быть использованы независимо от других.

Обмен данными между функциями производится через функцию 48534D20h прерывания INT 2Fh. Это введено с целью упростить передачу входных параметров функциям и возвращение ими выходных параметров. Однако такая организация обладает недостатком, коим является сложность согласования занятых и незанятых блоков памяти между основным модулем и функциями поддержки. Эта проблема не является определяющей и сравнительно легко разрешима, чего нельзя сказать о выделении блока памяти размером более 64 кБайт за один раз.

Язык Ассемблер не обладает никакими заранее реализованными функциями, особенно это касается комплексных чисел. Для написания функций в подобных случаях требуется значительное время, но оно не является определяющим фактором. С другой стороны, общие принципы реализации не позволяют переводить каждую строку из Matlab в Ассемблер, что приводит к необходимости создания набора основных функций поддержки, например вывод матрицы на экран.

Наиболее сложным в этом отношении является алгоритм Витерби. Его модель на Matlab занимает 186 строк, а на Ассемблере в конечном варианте он занимает еще больше. Однако выполняемый код будет, безусловно, значительно меньше, и это окупит трудности создания данной функции.

Определенные трудности также вызвало создание оптимального фильтра и процедур кодирования. В последних, к сожалению, требуется использование большого количества табличных данных, что увеличивает объем кода.

Из интерфейса пользователя наибольшую трудность вызвал вывод на экран матрицы комплексных чисел из-за различия в количестве их знаков, и как следствие, проблем выравнивания столбцов. Здесь точного выравнивания достичь очень трудно.

## 1.2 Принципы оверлейной организации программы

Главная причина выбора такой организации – удобство отладки. Если каждая функция находится в отдельном файле, при отладке можно перекомпилировать только измененную функцию. Вторая причина – ускорение и упрощение процедуры запуска внешних модулей. С этой целью формат функций или их заглушек несколько изменен.

Если в режиме отладки при работе основного или вспомогательного модулей введена команда, не значащаяся в перечне внутренних команд, производится попытка найти файл с именем, соответствующим этой команде и расширением .OVL. Если такой файл не найден, выводится сообщение об ошибке.

Если файл найден, он открывается. Первые два байта его указывают смещение точки входа находящейся в файле функции. Далее выделяется память объемом, равным размеру файла, и производится его чтение. Однако чтение в данном случае производится в режиме Read, а не Load, что означает необходимость дальнейшей передачи управления. Передача управления производится как дальний вызов подпрограммы путем возврата на точку входа. Возвращение из функции производится как возврат из подпрограммы. После этой операции файл модуля закрывается и память, занятая им, освобождается.

При всех достоинствах такой организации она обладает одним недостатком. При получении ошибки от функции Malloc (выделение памяти), или, если это возможно, от функции InputString (ввод строки с клавиатуры) возврат должен быть произведен прямо на вызвавший адрес. Однако это невозможно, поскольку эти ошибки индицируются сообщениями, расположенными в кодовом сегменте основного модуля, а не в кодовом сегменте вызывающего модуля. Поэтому при возникновении такого рода ошибок может произойти остановка системы из-за разрушения блока управления памятью.

2 ОСНОВНОЙ МОДУЛЬ ПРОГРАММЫ GSM.EXE

Этот модуль выполняет следующие функции:

1. Инициализацию приложения;
2. Управление загрузкой оверлейных модулей;
3. Управление интерфейсом приложения – вывод строки на экран, ввод строки с клавиатуры, вывод различного вида сообщений и предупреждений, вывод меню;
4. Выбор режима работы системы GSM меню;
5. Запуск необходимого оверлейного модуля в соответствии с выбранным пунктом;
6. Выдача входных данных внешним модулям и прием от них выходных данных;
7. Управление памятью;
8. Управление обработкой исключений операционной системы.

2.1 Инициализация приложения

При запуске программы GSM.EXE в первую очередь проверяется, не запущена ли уже еще одна копия программы. Оверлейная организация обладает тем недостатком, что если повторно занять тот же вектор прерывания, то это вызовет ошибку при выделении памяти при завершении работы копии, которая была запущена позднее. Поэтому вызывается подфункция 00 прерывания 2F/48534D20 для проверки. Если возвращено значение 80h (как принято), то работа программы останавливается и выводится сообщение (в режиме TTY).

Далее необходимо обеспечить поддержку русскоязычного интерфейса, для чего в видеоконтроллер загружается шрифт (при помощи прерывания 10/1100). С целью исключить сбои при каждой очистке экрана шрифт загружается заново, что обеспечивается переназначением прерывания 10/0003 на процедуру загрузки шрифта.

Для обеспечения возможности выделения динамической памяти необходимо выделить блок памяти под выполняемую программу. К сожалению, организация памяти DOS такова, что за один раз нельзя выделить блок размером более 65520 байт (4095 параграфов), а это приводит к тому, что выделение памяти может начаться с точки адресного пространства, лежащей ниже точки входа программы. Это, в свою очередь, приведет к остановке системы после выхода из программы из-за того, что программа, занимавшая эту память, окажется поврежденной. Для исключения этого в начале выделяется блок размером 16 байт (1 параграф), который попадает на эту область. Он является недоступным далее.

Крупнейшим недостатком современных операционных систем является то, что они неспособны идентифицировать попытку перехода в графический режим из текстового оконного режима. Если программа произведет таковую попытку на компьютере, имеющем видеоконтроллер некоторых редких типов GeForce, операционная система будет остановлена и придется перегружать компьютер. Для решения этой проблемы при запуске программы производится переход в полноэкранный режим при помощи функции проверки поддержки VESABIOS. Видеокарты, не обладающие указанным недостатком, при вызове этой функции не производят автоматический переход в полноэкранный режим. Поэтому настоятельно рекомендуется перед первым запуском программы создать ярлык с установкой на полноэкранный режим.

Далее устанавливается вектор прерывания INT 2Fh. В этом прерывании занимается функция 48534D20h (GSM). Как показала проверка, ни один стандартный драйвер устройства не использует эту функцию, и, как следствие, она может быть использована без каких-либо ограничений.

Наконец, выводится главное меню. В зависимости от выбранного пользователем пункта запускается тот или иной оверлейный модуль. Все оверлейные модули будут рассмотрены далее.

2.2 Обработка прерывания 48534D20-INT 2F

Этот вектор прерывания служит для обмена данными между основным модулем и внешними функциями или между внешними функциями, а также для вызова интерфейсных функций приложения. Имена подфункций находятся в файле GSMExec.asm, включаемом в каждый модуль приложения. В состав процедуры обработки прерывания входят следующие подфункции:

Таблица 1.1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Номер | Назначение | Входные параметры | Выходные параметры при успешном завершении | Выходные параметры при аварийном завершении |
| 00 | Проверка поддержки GSM | – | AL=80h | – |
| 01 | Получить структуру переменных GSM (см. таблицу 2.1) | – | ES:BX=адрес структуры | CY=1 при отсутствии структуры |
| 02 | Получить структуру переменных алгоритма Витерби (см. таблицу 2.2) | – | ES:BX=адрес структуры | CY=1 при отсутствии структуры |
| 03 | Получить размер генерируемого блока данных | – | AX=размер | – |
| 04÷19h | Получить адрес блока данных (см. таблицу 1.2) | – | ES:BX=адрес запрошенного блока | CY=1 при отсутствии блока |
| 2Bh | Получить тип данных | – | AL=тип данных (см таблицу 1.4) | – |
| 40h | Сгенерировать псевдослучайную последовательность | CX=длина, ES:DI=адрес | AX=0 при успешном завершении | – |
| 41h | Вывод целого десятичного числа на экран | ES:CX=адрес памяти с числом, DX=позиция экрана | – | – |
| 42h | Вывод действительного числа на экран | ES:CX=адрес памяти с числом, DX=позиция экрана | – | – |
| 45h | Загрузка и запуск оверлейного модуля | BH=номер модуля (см. таблицу 1.3) | AX=0 | Остановка программы |
| 55h | Получить адрес шрифта | – | ES:BX=адрес шрифта | – |
| 60h | Сохранить экран в файл | DS:DX=указатель на имя файла | – | – |
| 70h | Выделение памяти | CX=размер блока в байтах | ES:BX=адрес блока | Остановка программы |
| 71h | Освобождение блока памяти | ES=адрес блока для освобождения | – | Остановка программы |
| 7Bh | Вывод меню и выбор пункта | DS:DX=указатель на первый пункт меню в формате OutputString, CX=количество пунктов | AX=номер выбранного пункта | AX=0FFFFh, если пользователь нажал клавишу ESC. |
| 7Ch | Вывод строк на экран | DS:DX=указатель на строку в формате OutputString, CX=количество строк | – | – |
| 7Dh | Вывод рамки с сообщением на экран | CX=цвет, DS:DX=указатель на строку в формате OutputString | – | – |
| 7Eh | Ввод числа с клавиатуры | DS:DX=адрес строки приглашения, ES:CX=адрес для введенного числа, ESI=минимальное значение, EDI=максимальное значение | – | CY=1, если пользователь нажал клавишу ESC |
| 80h | Вывод рамки на экран | – | – | – |
| 81h | Установить адрес структуры переменных GSM (см. таблицу 2.1) | ES:BX=адрес структуры | – | – |
| 82h | Установить адрес структуры алгоритма Витерби (см. таблицу 2.2) | ES:BX=адрес структуры | – | – |
| 83h | Установить размер генерируемого блока данных | CX=размер | – | – |
| 84h÷99h | Установить адрес блока данных (см. таблицу 1.2) | ES:BX=адрес блока данных | – | – |
| 0AAh | Сбросить систему GSM | – | AX=0 | Остановкапрограммы |
| 0ABh | Установить тип генерируемого блока данных | CL=тип (см. таблицу 1.4) | – | – |
| 0ССh | Установить тип канала связи | CL=тип (см. таблицу 1.5) | – | – |
| 0CDh | Получить тип канала связи | – | AL=тип (см. таблицу 1.5) | AL=0, если тип еще не установлен |
| 0FCh | Установить параметр канала связи | ES:BX=адрес параметра | – | – |
| 0FDh | Получить параметр канала связи | – | ES:BX=адрес параметра | ES=0, если адрес еще не установлен |
| 0FEh | Установить флаг вывода результатов | CL=0 – разрешить вывод  CL=1 – запретить вывод | – | – |
| 0FFh | Получить флаг вывода результатов | – | CY=0, если вывод разрешен, CY=1, если запрещен | – |

Некоторые функции требуют комментариев.

Функция выделения памяти при аварийном завершении выдает на экран предупреждение в красной рамке. Текст сообщения зависит от возвращенного кода ошибки. В случае разрушения блока памяти (код 07) корректный выход из программы невозможен, поскольку системе не удастся загрузить командный процессор и это приведет к остановке операционной системы. В расчете на то, что модель будет использоваться в режиме виртуального процессора, вместо выхода из процесса в этом случае используется функция NTVDM (BOP 0x19), именуемая VDDTerminateVDM, осуществляющая автоматическое закрытие программы-монитора виртуального процессора. То же самое касается функции освобождения памяти.

Функция генерации псевдослучайной последовательности генерирует последовательность действительных чисел от -1 до 1. Далее эта последовательность может быть преобразована в двоичную по тому или иному критерию единицы.

Флаг вывода результатов указывает функции, следует ли выводить на экран результат ее выполнения. Вывод результатов может быть назначен в пошаговом режиме для контроля выполнения функций. Во всех остальных случаях он должен быть отключен.

Процедура сохранения экрана в файл производит копирование в файл символьного образа экрана (если текстовый режим) или в графический файл формата BMP (если графический режим).

2.3 Функции установки/получения адресов переменных и структур данных и сброса системы

Функции получения и установки адресов структур переменных GSM и алгоритма Витерби просто производят запись в соответствующие ячейки памяти или считывание из них указанных адресов. Если при считывании оказывается, что адрес не установлен, устанавливается флаг переноса.

Функции получения и установки адресов блоков данных производят аналогичную операцию, но с одним дополнением. Если при установке адреса обнаружено, что блок уже установлен, производится сканирование таблицы адресов на предмет совпадения адреса какого-либо блока с тем, что уже установлен. Если такого соответствия нет, сначала освобождается старый блок. Если этого не учесть, то постепенное накопление неиспользуемых блоков приведет к нехватке памяти и остановке программы.

Соответствие номера функции и блока данных приведено в таблице 1.2.

Таблица 1.2

|  |  |
| --- | --- |
| **Номер** | **Назначение** |
| 04 | Передаваемый блок данных |
| 05 | Передаваемый кадр |
| 06 | Дифф. представление передаваемого кадра |
| 07 | Элементарная частотная функция модулятора |
| 08 | Элементарная фазовая функция модулятора |
| 09 | Синфазная компонента передаваемого сигнала |
| 0Ah | Квадратурная компонента передаваемого сингала |
| 0Bh | Синфазная компонента принятого сигнала |
| 0Ch | Квадратурная компонента принятого сигнала |
| 0Dh | MSK-представление принятого кадра |
| 0Eh | Автокорреляционная функция ИХ канала связи |
| 0Fh | Принятый кадр |
| 10h | Принятый блок данных |
| 11h | Кодированныйблокпередачи №1 |
| 12h | Кодированныйблокпередачи №2 |
| 13h | Матрица передачи №1 |
| 14h | Матрица передачи №2 |
| 15h | Матрица приема №1 |
| 16h | Матрица приема №2 |
| 17h | Декодированный блок |
| 18h | Биты паритета |
| 19h | Промежуточный блок |

Функция сброса системы очищает адреса всех установленных блоков данных и таким образом подготавливает систему к новой передаче.

2.4 Функция запуска оверлейного модуля

Этой функции в качестве единственного параметра передается номер запускаемого модуля. Имена модулей и их номера, а также необходимые данные для их запуска приведены в таблице 1.3.

Таблица 1.3

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Номер | Наименование | Назначение | Необходимые данные | Примечание |
| 0 | Gsminit | Инициализация системы | – | \* |
| 1 | Gsmset | Создание структуры GSM | – | Однократный запуск |
| 2 | Vitinit | СозданиетаблицалгоритмаВитерби | ЧислоотсчетовИХ | Однократный запуск |
| 3 | Gsmtest | Учебный контроль | – |  |
| 4 | OutVar | Вывод переменных структуры GSM | – |  |
| 5 | VitOut | ВыводтаблицалгоритмаВитерби | – |  |
| 6 | Gsmdebug | Отладочный модуль | – |  |
| 7 | Gendata | Генератор блока данных | Тип блока данных | По умолчанию – случайные данные |
| 8 | Genburst | Генератор кадра | Блок данных |  |
| 9 | Bdenc | Дифференциальное кодирование кадра | Передаваемый кадр |  |
| 10 | Phg | Генератор элементарных функций модулятора | – |  |
| 11 | Gmskmod | Модулятор | Дифф. кодированный кадр |  |
| 12 | gsmmod | Заглушка для запуска последних 3 функций | Передаваемый кадр |  |
| 13 | Mafi | Согласованный фильтр | Принятый сигнал |  |
| 14 | Vitdet | Детектор Витерби | MSK-представление принятого сигнала |  |
| 15 | Demux | Выделитель блока данных | Принятый кадр |  |
| 16 | Chanenc | Кодирующий модуль | Передаваемый блок данных (260 бит) |  |
| 17 | Chandec | Декодирующий модуль | Матрица приема |  |
| 18 | Interlv | Перемежитель | Два кодированных блока |  |
| 19 | Deinterlv | Обратный перемежитель | Две матрицы приема |  |
| 20 | Chsim | Эмулятор канала | Передаваемый сигнал, тип канала и его параметр |  |
| 21 | Chselect | Выбор типа канала связи | – |  |
| 22 | Phgdemo | Демонстрация phg | – | \* |
| 23 | Gmskdemo | Демонстрация GMSK | – | \* |
| 24 | Mafidemo | Демонстрация mafi | – | \* |
| 25 | Digtrans | Демонстрация логических операций передачи | – | \* |
| 26 | Vitdemo | Демонстрация алгоритма Витерби | – | \* |
| 27 | Encdemo | Демонстрация кодирования | – | \* |
| 28 | Decdemo | Демонстрация декодирования | – | \* |
| 29 | Msk | Демонстрация формирования MSK-представления | – | \* |
| 30 | Nocod | Головной модуль передачи без кодирования | – | \* |
| 31 | Cod | Головной модуль передачи с кодированием | – | \* |

"Звездочка" в графе "Примечание" означает, что данный модуль *не рекомендуется запускать в пошаговом режиме*.

Номер модуля определяет смещение в таблице имен файлов. Система открывает указанный файл (при этом считается, что файл существует – проверка наличия файлов производится при инициализации поддержки GSM). Далее производится определение его размера и подсчет контрольной суммы. Выделяется память необходимого размера, файл читается и управление передается на его точку входа, адрес которой указан в первом 16-разрядном слове файла. После возврата из модуля память, занятая им, освобождается. Такая организация избавляет от необходимости слежения за использованием памяти, поскольку при применении стандартных функций загрузки оверлеев DOS память, занятая с точки зрения одного модуля, может оказаться свободной с точки зрения другого.

2.5. Выделение и освобождение памяти

Эти две функции используют стандартные прерывания DOS для управления памятью. Комментариев они не требуют, за исключением одной тонкости.

Как уже говорилось, в начале программы выделяется блок размером 16 байт для исключения возможности конфликта по памяти материнских процессов с дочерними. В процессе разработки программы организация все более и более усложнялась, что в конце концов привело к необходимости проверки пересечения границ памяти приложения. Поэтому при каждом выделении памяти обязательно, чтобы выделенный блок лежал выше точки входа программы. Если это условие не выполнено, выделение повторяется.

2.6 Интерфейсные функции

Интерфейс пользователя оперирует прежде всего со строкой сообщения. Формат строки следующий. Первый байт указывает номер строки экрана для вывода, второй – номер позиции для вывода, третий – количество символов в сообщении, четвертый – цвет. Далее идет текст сообщения. Если количество символов указано нулевое, то считается, что необходим вывод повторяющегося символа количеством, указанным в пятом байте, а сам символ указывается в шестом байте.

Функция вывода рамки на экран строит рамку строго определенного размера из символов псевдографики (координаты (2;4) – (20;76), цвет желтый).

Функция вывода рамки с сообщением производит вывод рамки с двойной границей и сообщением указанным цветом. При этом если указан цвет 4Fh (белый на красном фоне), дополнительно выводится сообщение "Нажмите любую клавишу для аварийного выхода".

Функция вывода меню производит вывод последовательности строк на экран. Выбранной считается строка, у которой цвет 4Fh. Далее производится выбор пункта меню путем изменения цветов строк и их повторного вывода. В случае нажатия Enter возвращается номер выбранного пункта, в случае нажатия ESC возвращается значение 0FFFFh.

Функция ввода числа с клавиатуры производит ввод целого десятичного числа с проверкой допустимых пределов, а затем помещает его в указанную ячейку памяти. Если нажимается клавиша ESC, производится установка флага переноса.

Вывод целых или действительных чисел на экран производится путем их преобразования к формату упакованных десятичных чисел. Точность представления действительного числа не регламентируется, поскольку точность вычисления сопроцессора компьютера не ограничена, а входными данными являются только биты. Обычно точность составляет 6 знаков после запятой.

2.7 Функции получения/установки флагов и параметров

Указанные функции используются для контроля окружения передачи. Они просто записывают/считывают идентификаторы типов из ячеек памяти.

Функция генерации блока данных использует установленный идентификатор типа блока данных, возможные значения которого приведены в таблице 1.4.

Таблица 1.4

|  |  |
| --- | --- |
| Значение | Тип блока данных |
| 0 | Случайные |
| 1 | Только нули |
| 2 | Только единицы |
| 3 | Попеременно нули и единицы (0,1,0,1…) |
| 4 | Попеременно нули и единицы (0,0,1,1,0,0,1,1…) |

Функция эмулятора канала оперирует с указанным идентификатором типа канала, возможные значения которого приведены в таблице 1.5.

Таблица 1.5

|  |  |  |
| --- | --- | --- |
| Значение | Тип канала | Параметр |
| 0 | Не определен | – |
| 1 | Идеальный (без шумов и переотражений) | – |
| 2 | Канал с аддитивными шумами, задаваемый фактором | Фактор |
| 3 | Канал с аддитивными шумами, задаваемый отношением сигнал/шум | Отношение сигнал/шум |
| 4 | Канал связи с переотражениями | Интервал запаздывания |
| 5 | Канал связи с межсимвольной интерференцией | Интервал запаздывания |

Функция получения/установки адреса параметра канала запоминает или выдает адрес параметра канала. Если адрес уже был установлен, он остается занятым.

3 МОДУЛЬ ИНИЦИАЛИЗАЦИИ СИСТЕМЫ ПОДДЕРЖКИ GSM

Этот модуль предназначен для проверки окружения приложения и инициализации параметров, необходимых для любой работы в модели.

В первую очередь проверяется, не произведена ли уже инициализация. Повторная инициализация может быть необходима для изменения глобальных параметров. Если инициализация произведена, выдается предупреждения. Для подтверждения повтора необходимо нажать Y. При повторной инициализации создание структуры GSM пропускается, а алгоритм Витерби инициализируется только в случае задания другого числа отсчетов ИХ (Lh). Сделано это с целью исключить сбои из-за разного размера блоков памяти, необходимых для матриц Витерби для разного числа Lh.

Далее проверяется тип процессора. При установке программы после ввода лицензионного ключа происходит запись текущих идентификаторов процессора в файл для защиты от несанкционированного копирования программы. Теперь эти идентификаторы сверяются с текущим процессором. При их несовпадении выводится соответствующее сообщение.

Следующим этапом является проверка наличия оверлейных файлов. Если какой-либо из 32 файлов модулей, либо из 4 файлов вопросов к лабораторным работам отсутствует, выводится сообщение и имя отсутствующего файла.

Далее запускается модуль GSM\_SET.

3.1 Модуль gsm\_set

Эта функция выполняет следующие действия:

1. Выделяет память под структуру параметров GSM, которая потом заполняется адресами переменных.
2. Создает переменные. В их число входят:

Init\_l – длина области данных в кадре GSM. По стандарту она равна 114 битам;

Tb – длительность одного бита кадра в секундах. Она равна 3,692e-6.

Bt – нормированная полоса пропускания. Она определяется как полоса частот, деленная на длительность бита, и равна 0,3.

Для каждой переменной выделяется память, адрес которой записывается в структуру GSM.Создает тренировочную последовательность. Эта последовательность располагается в середине кадра между блоками данных. Для правильного создания кадра GSM и правильного его декодирования эта последовательность преобразуется из двоичного вида в MSK-вид при помощи функции T\_seq\_gen.

3.2 Функция t\_seq\_gen

Эта функция производит преобразование 26-битной тренировочной последовательности в GMSK-вид.

Сначала производится проверка входной последовательности на наличие посторонних символов и соответствие длины. Каждый бит для простоты занимает один байт. Если последовательность некорректна, то выводится сообщение и функция завершается аварийно.

Далее выделяется 26 байт для полярного представления последовательности и 26 байт для дифференциального представления. С каждым битом a(n) производится операция

,

что приводит к полярному представлению (от -1 до 1). Далее производится перемножение каждого последующего бита полярного представления на предыдущий:

y(n)=x(n)\*x(n-1).

Конечным результатом работы данной функции является последовательность битов, связанных законом

z(n)=j\*y(n)\*z(n-1).

Здесь и далее комплексные числа представляются последовательностью либо двух байт, либо двух квадрослов (в данном случае байт). Первый бит выходной последовательности для определенности выбран равным 1. Функция завершается освобождением блоков дифференциального и полярного представления последовательности.

3.3 Ввод данных глобального назначения

К таковым данным относится начальный указатель псевдослучайной последовательности, число шагов модуляции на один бит и число отсчетов импульсной характеристики канала.

Первый из этих параметров определяет значение, загружаемое в сдвиговый регистр для получения псевдослучайной последовательности. Выбран 32-разрядный указатель, так что диапазон допустимых значений до 4294967295. Если задавать один и тот же указатель, то будет одна и та же последовательность, что удобно для отладки. В принципе, можно поле для указателя оставить пустым.

Число шагов модуляции на один бит (Oversamplingratio, OSR) определяет число отсчетов сигнала, используемых для передачи одного бита, и может иметь значение от 2 до 16.

Число отсчетов импульсной характеристики канала необходимо для функционирования алгоритма Витерби. Оно определяет матрицу Витерби, и, как следствие, все его дальнейшие действия. В данной модели оно может составлять от 2 до 4. Столь маленький диапазон допустимых значений обусловлен тем, что при поиске стартовых и стоповых символов поисковые матрицы являются эталонными, а хранить большое их количество в константах не слишком выгодно.

Ввод параметров контролируется. Для ввода параметров с экрана создана специальная процедура Input. Ее рассмотрение в рамках данной работы не требуется.

На этом общая процедура инициализации заканчивается. Следующим этапом является подготовка таблиц для алгоритма Витерби.

3.4 Инициализация алгоритма Витерби

Единственным входным параметром для процедуры Viterbi\_Init является число отсчетов импульсной характеристики.

Сначала выделяется память под структуру адресов таблиц алгоритма Витерби. Алгоритм процедуры состоит из пяти этапов, каждый из которых для удобства выделен в отдельную функцию.

3.4.1 Функция Make\_symbols

Эта функция генерирует матрицу Витерби для указанного числа отсчетов импульсной характеристики Lh.

Для Lh=1 матрица имеет вид 1, i, -1, -i . Значение lh=1 не может быть воспринято функцией по причинам, которые станут понятны позднее.

Каждый последующий декремент Lh производит следующую операцию с матрицей Витерби. Первый столбец умножается на i и записывается на место первой половины первого столбца, затем тот же самый столбец умножается на –i и записывается в вторую половину первого столбца. Остальные столбцы есть просто двойная копия исходной до операции матрицы, сдвинутая на один столбец. Например, для Lh=2 матрица имеет следующий вид:

i 1

-1 i

-i -1

1 -i

-i 1

1 i

i -1

-1 -i

Таким образом, каждая последующая операция вызывает увеличение количества строк матрицы в два раза, а количества столбцов на 1.

Приведем матрицы для Lh=3 и Lh=4:

Lh=3

-1 i 1

-i -1 i

1 -i -1

i 1 -i

1 -i 1

i 1 i

-1 i -1

-i -1 -i

1 i 1

i -1 i

-1 -i -1

-i 1 -i

-1 -i 1

-i 1 i

1 i -1

i -1 -i

Lh=4

-i -1 i 1

1 -i -1 i

i 1 -i -1

-1 i 1 -i

i 1 -i 1

-1 i 1 i

-i -1 i -1

1 -i -1 -i

i 1 i 1

-1 i -1 i

-i -1 -i -1

1 -i 1 -i

-i -1 -i 1

1 -i 1 i

i 1 i -1

-1 i -1 -i

i -1 i 1

-1 -i -1 i

-i 1 -i -1

1 i 1 -i

-i 1 -i 1

1 i 1 i

i -1 i -1

-1 -i -1 -i

-i 1 i 1

1 i -1 i

i -1 -i -1

-1 -i 1 -i

i -1 -i 1

-1 -i 1 i

-i 1 i -1

1 i -1 -i

Теперь очевидно, что при Lh=1 функция выдаст неверную матрицу из-за того, что первый же декремент вызовет зацикливание.

Следует отметить, что элемент (0,0) матрицы Витерби обязательно должен быть мнимым. С этой целью при обнаружении его нулевой мнимой части требуется перевернуть матрицу таким образом, чтобы позиции столбцов сохранились, а номера строк поменялись на противоположные (0 на 31, 1 на 30 и т. д.)

В принципе, данная функция способна сгенерировать матрицу при Lh>4, однако такое значение недопустимо для последующих функций.

3.4.2 Функция Make\_Previous

Эта функция генерирует матрицу предыдущих состояний на основе заданной матрицы Витерби.

Матрица Витерби обладает тем свойством, что каждой строке соответствует **не более двух предыдущих состояний** (поскольку это либо 0, либо 1). Их поиск производится следующим образом.

Из матрицы Витерби выделяются все столбцы, кроме последнего, и помещаются в поисковую матрицу. Из каждой строки матрицы Витерби выделяется вектор из элементов со 2 по последний столбец, после чего он сравнивается последовательно с каждой строкой поисковой матрицы. Как только будет обнаружено совпадение, соответствующий номер строки заносится в выходную матрицу.

Весьма существенно, что нумерация элементов начинается с нуля, поэтому по сравнению с моделью в пакете Matlab все элементы имеют на единицу меньшие значения.

Полученная матрица для Lh=4 имеет следующий вид:

3 5

0 6

1 7

2 4

11 13

8 14

9 15

10 12

19 21

16 22

17 23

18 20

27 29

24 30

25 31

26 28

3 5

0 6

1 7

2 4

11 13

8 14

9 15

10 12

19 21

16 22

17 23

18 20

27 29

24 30

25 31

26 28

3.4.3 Функция Make\_next

Эта функция генерирует матрицу последующих состояний на основе заданной матрицы Витерби.

Матрица Витерби обладает тем свойством, что каждой строке соответствует **не более двух последующих состояний** (поскольку это либо 0, либо 1).

Поиск аналогичен, за исключением того, что поисковая матрица формируется из столбцов 2:Lh исходной матрицы, а поисковый вектор из столбцов 1:Lh-1.

Полученная матрица для Lh=1 имеет следующий вид:

1 17

2 18

3 19

0 16

3 19

0 16

1 17

2 18

5 21

6 22

7 23

4 20

7 23

4 20

5 21

6 22

9 25

10 26

11 27

8 24

11 27

8 24

9 25

10 26

13 29

14 30

15 31

12 28

15 31

12 28

13 29

14 30

3.4.4. Функция Make\_Start

Эта функция производит поиск символа матрица Витерби, с которого следует начинать цикл поиска метрики.

Для каждого значения Lh здесь существует свой поисковый вектор, поэтому значение Lh ограничено сверху 4, а снизу 1. Этот вектор при Lh=4 представляет собой комплексно сопряженную матрицу от исходной первичной матрицы Витерби (то есть 1,-i,-1,i).

Производится сканирование матрицы Витерби до момента совпадения строки с поисковым вектором. Матрица Витерби обладает тем свойством, что **при любом значении Lh существует единственный номер стартового символа.** Для Lh=4 этот номер 1.

3.4.5 Функция Make\_stops

Эта функция производит поиск номеров символов матрицы Витерби, которыми может заканчиваться процесс поиска минимальной метрики. Для Lh<=3 существует только один такой номер. Для Lh=4 существуют два номера.

Процесс поиска стопового символа несколько сложнее, чем стартового. Поисковая матрица здесь имеет вид исходной символьной при Lh=1 с измененным знаком действительной части. Для Lh=4 она состоит из двух идентичных строк. Поиск производится построчно, а в выходную матрицу заносится номер совпадающей строки-1.

Для Lh=4 стоповые символы имеют номера 3 и 5.

4 ПРИЕМ И ПЕРЕДАЧА БЕЗ КОДИРОВАНИЯ

Именно такое название имеет первый вариант выполнения программы, который обеспечивается модулем nocod.ovl.

После сброса системы GSM в начале модуля вводится количество циклов приема-передачи, которое будет использоваться при каждой попытке передачи. В каждом цикле передается 4 кадра для совместимости с вариантом с кодированием.

Далее запускается модуль выбора канала для передачи. На этом этапе следует ввести необходимый параметр канала.

После этого выводится рамка и сообщение "Передача…". В течение передачи на экране отображается информация о каждом ее цикле. После отображения 15 циклов программа останавливается и ждет нажатия клавиши. Так продолжается до исчерпания указанного количества циклов.

Каждый цикл предполагает следующую последовательность запуска функций:

Gendata

Gsmmod

Chsim

Mafi

Vitdet

Demux

После этого производится сравнение переданного и принятого блоков данных и подсчитывается количество ошибок.

По окончании передачи на экран выводится суммарное количество переданных кадров, общее количество битов данных, общее количество ошибок, среднее количество ошибок за цикл, а также вероятность ошибки в долях единицы.

А теперь разберем процедуру передачи.

4.1Передатчик

В состав передатчика входят следующие функции:

Gendata – генерация блока данных

Genburst – генерация кадра GSM на основе блока данных

BDEnc – преобразование кадра GSM в дифференциальное представление

Phg – подсчет частотной и фазовой функций модулятора GMSK

GMSKMod – модулятор GMSK

GSMMod – заглушка для запуска последних четырех функций.



Рисунок 4.1- Структурная схема передатчика

4.1.1 Функция Gendata

Функция генерирует блок данных длиной, равной 114 для случая без кодирования. Тип данных предполагается установленным. Если он не установлен, выбираются случайные данные.

Если данные случайные, производится генерация случайных чисел от указателя ПСП путем сдвига этого указателя. Практика показала, что наименьшей периодичностью обладают сдвиги с интервалом снятия разрядов, равным 3. Результат лежит в пределах от -1 до 1, что достигается путем деления на 10-18. Далее в зависимости от соотношения модуля числа и 0,5 производится формирование выходного блока данных и фиксация его адреса.

При других типах данных формирование элементарно и пояснений не требует.

При наличии разрешения вывода результатов функция выводит на экран сгенерированный блок в виде последовательности битов.

4.1.2 Функция GenBurst

В соответствии со стандартом GSM каждый кадр имеет следующий формат.

|  |  |
| --- | --- |
| Смещение | Назначение |
| 00 | Хвост (000) |
| 03 | Данные (57 бит) |
| 60 | Управление (1) |
| 61 | Тренировочная последовательность (26 бит) |
| 87 | Управление (1) |
| 88 | Данные (57 бит) |
| 145 | Хвост (000) |

Для выполнения функции необходим блок данных. Если его нет, то выдается сообщение и функция завершается аварийно.

Размер кадра составляет 148 бит. Тренировочная последовательность служит для облегчения определения импульсной характеристики канала передачи.

Формирование кадра очевидно и не представляет сложности.

При разрешении вывода на экран результатов выводятся поля кадра – каждое на отдельной строке.

4.1.3 Функция BDEnc

Для выполнения функции необходим передаваемый кадр. Если его нет, то выдается сообщение и функция завершается аварийно.

Производит дифференциальное кодирование кадра GSM.

К кадру слева добавляется один единичный бит. Хотя кодирование и называется дифференциальным, на самом деле используется сложение по модулю 2 (исключающее ИЛИ):

y(n)= x(n) xorx(n-1) (4.1)

Далее производится полярное преобразование кадра по формуле:

Y(n)=1-2\*x(n) (4.2)

4.1.4 Моделирование процесса модуляции

Гауссовская частотная манипуляция с минимальным частотным сдвигом (GMSK) очень похожа на частотную манипуляцию с минимальным сдвигом (MSK). Оба вида модуляции относятся к модуляции с непрерывной фазой и сдвигом частоты (CPFSK – ContinuousPhaseFrequencyShiftKeying).

MSK сигнал можно описать следующим образом:

 (4.3)

где Ec – энергия одного бита, fc – частота несущей,  – фаза MSK сигнала, несущая информацию.

В данной модели радиосигнал не моделируется, а используется лишь метод комплексной огибающей. В связи с этим несущую частоту можно убрать:



, (4.4)

где А и представляет амплитудную и фазовую модуляцию несущей соответственно. Воспользуемся следующей формулой:



 (4.5)

Выражение (4.5) представляет собой комплексную огибающую модулированного сигнала. Очевидно, что синус и косинус  являются узкополосными сигналами, следовательно I и Q обозначают синфазную и квадратурную компоненты модулированного сигнала. Эти два узкополосных сигнала полностью описывают исходный сигнал по формуле (4.3).

Для записи квадратурных компонент удобно воспользоваться так называемой функцией сглаживания импульса p(t):

⎧cos(t / 2Tb) , -Tb≤ t ≤ Tb

p(t) = ⎨ (4.6)

⎩ 0, -Tb> t или Tb< t,

тогда иможно писать в следующей линейной форме:

 (4.7)

 (4.8)

где ac(t) и as(t) – взвешенные импульсные характеристики:

 (4.9)

 (4.10)

и δ - дельта-функция, а Θ[n] можно выразить следующим образом:

 (4.11)

С целью упрощения описания MSK видеосигнала, введем комплексный ряд I:

I[n] ≡ejΘ[n] = cos(Θ[n]) + j ⋅ sin(Θ[n]) = ac[n] + j ⋅ as[n] (4.12)

Используя комплексный ряд, можно представить MSK-видеосигнал в следующей форме:

 (4.13)

где  (4.14)

Как видно из (4.12), I[n] поочередно принимает вещественные и мнимые значения. Это можно объяснить тем, что Θ[n] ∈ {0, /2, , 3/2}. Это приводит к следующему рекурсивному выражению:

I[n] = j ⋅I[n-1] ⋅ a[a-1], (4.15)

где

I[n] ∈ {1, -1, j, -j},

a[n] ∈ {1, -1}.

В рекомендациях по стандарту GSM изложен алгоритм дифференциального кодирования. На рисунке 4.2 изображена общая блок - схема модулятора.





a[n]

d[n]

Дифф.

кодер

I[n] = j ⋅ I[n-1] ⋅

⋅ a[a-1]

p(t)

Отображатель

MSK сигнала

Рисунок 4.2- Общая блок-схема модулятора

В GMSK, фаза сигнала меняется медленнее, чем в MSK, поэтому спектр сигнала уже. Но при этом увеличивается глубина межсимвольной интерференции, которая приводит к увеличению вероятности ошибки на бит (BER – BitErrorRate). GMSK сигнал можно создать разными методами, один из методов изображен на рисунке 4.2, но здесь требуется выбрать соответствующую p(t). Метод, примененный в модели, немного отличается от вышеописанного метода, он изображен на рисунке 4.3.











a[n]

d[n]

Дифф.

кодер

g(t)

CosΘ

SinΘ

Рисунок 4.3- GMSK модулятор примененный в модели

Из рисунке 4.3 видно, что последовательность символов α, и частотная функция g(t), подвергаются операцией свертки, результат свертки умножается на и получается фазовая функция Θ(t, α).

Фазовую функцию Θ(t, α) можно переписать так:

 (4.16)

где h – индекс модуляции, который для GSM равен ½. Частотная функция g(t) является сверткой по времени прямоугольного импульса v(t), и гауссовской функции hg(t).

g’(t) = v(t) \* hg(t) (4.17)

где

⎧ 1/(2Tb) , 0 ≤ | t | ≤Tb/2

v(t) = ⎨ (4.18)

⎩0, 0 > t или Tb/2 < t

и

 где  (4.19)

Полоса гауссовской функции при ослаблении, равном 3дБ, определяется из значения нормированной полосы фильтра Вt, которая равна 0.3 для GSM. Идеальная гауссовская функция имеет период бесконечной длительности, t ⊂ [-∞,∞]. Так как сигнал конечной длительности легче обработать цифровым методом, то период гауссовской функции приравниватся к L. Значения OSR и L определяют количество выборок для представления колоколообразной гауссовской функции. Обычно, L принимает значения больше трех. Чтобы упростить обработку, гауссовская функция сдвигается на LTb/2, результирующую частотную функцию можно записать как,

, (4.20)

где

⎧ 1, 0 ≤ t ≤LTb,

ωL(t) = ⎨

⎩ 0, 0 > t или LTb< t.

4.1.4 Функция Phg

Функция служит для подсчета частотной и фазовой функций, необходимых для реализации GMSK-модуляции.

Для получения исходной функции на основе данных о длительности бита и числа шагов модуляции на бит предпринимаются следующие действия.

1) Создается массив из значений от (-2\*длительность бита) до (2\*длительность бита) с шагом

Ts=Tb/OSR,

размер которого таким образом определяется как 4\*OSR+1. Этот массив необходим для последующей генерации гауссовского распределения.

2) Создается массив из значений от (-2/длительность бита) до (2/длительность бита) с тем же шагом за вычетом этого шага. Размер его, таким образом, определяется OSR. Этот массив необходим для последующей генерации линейного распределения.

3) Генерируется гауссовский закон распределения. Для этой цели используется формула



,

где



PTV – массив, сгенерированный в пункте 1.

1. Генерируется линейный массив длиной, равной длине массива 2 пункта, каждый из элементов которого имеет значение

.

Результат выполнения пунктов 2 и 3 для OSR=16 иллюстрируется на рисунке 4.4

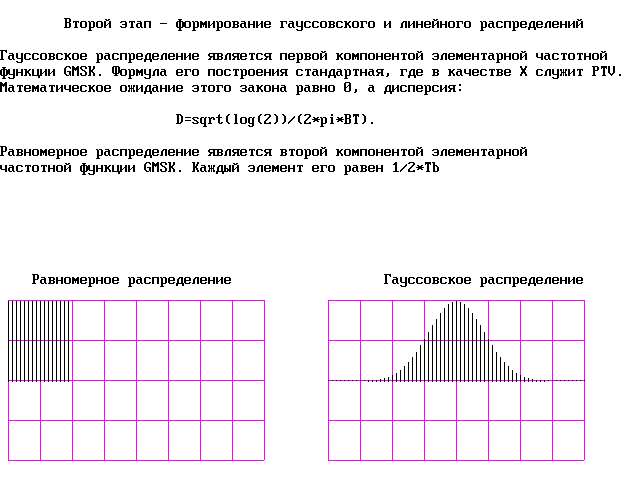


Рисунок 4.4

1. Вычисляется свертка гауссовского и линейного распределения. Для этой цели используется формула:

.

где массив Gauss дополнен нулями справа количеством (длина Rect - 1).

1. Далее из полученного массива выделяется подмассивG, начиная с элемента с номером OSR и до элемента 4\*OSR-1.

Результат выполнения представлен на рисунке 4.5.

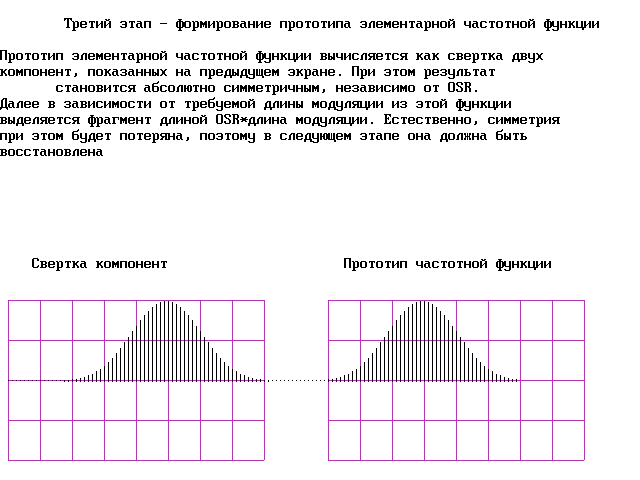


Рисунок 4.5

На основании полученного массива вычисляются частотные и фазовые функции по формулам

,

где k=3\*OSR,

.

В описанной процедуре есть один нюанс, а именно длина массива G. В связи с тем, что длина этого массива может быть от OSR до 5\*OSR, а его длина весьма сильно влияет на получаемый результат, было решено зафиксировать эту длину и установить ее равной 3\*OSR. Далее величина, равная количеству бит, перекрываемого одной элементарной частотной функцией, будет называться длиной модуляции.

Элементарная частотная и фазовая функция представлены на рисунке 4.6

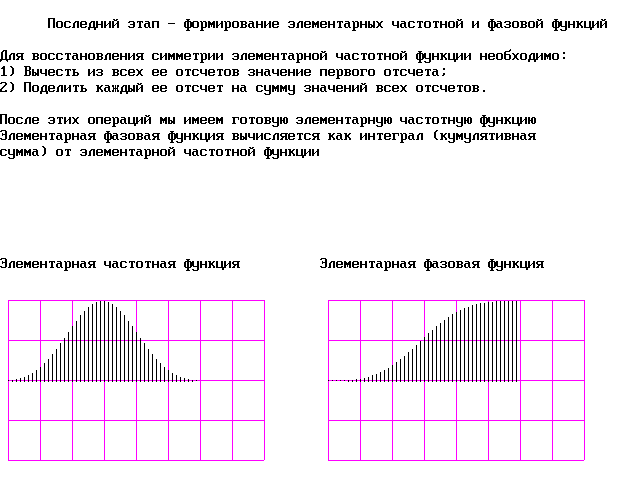


Рисунок 4.6

Если разрешена выдача результатов, на экран выводятся первые 12 отсчетов фазовой и первые 12 частотной функции.

4.1.5 Функция GMSK\_Mod

Функция предназначена для модуляции GSM-кадра.

Сначала производится запуск функции Phg. Далее создается массив результирующей частотной функции длиной

(Bits+2)\*OSR,

где Bits=148, и заполняется нулями.

Каждый последующий бит производит добавление сгенерированной с помощью функции Phgчастотной функции, умноженной на значение этого бита, к элементам результирующей функции с номерами от (n-1)\*OSR до (n+2)\*OSR. Таким образом, каждая позиция модулированного сигнала претерпевает добавление 3\*OSR раз, что совпадает с длиной частотной функции:







Вид результирующей частотной функции для OSR=4 и случайных данных представлен на рисунке 4.7.

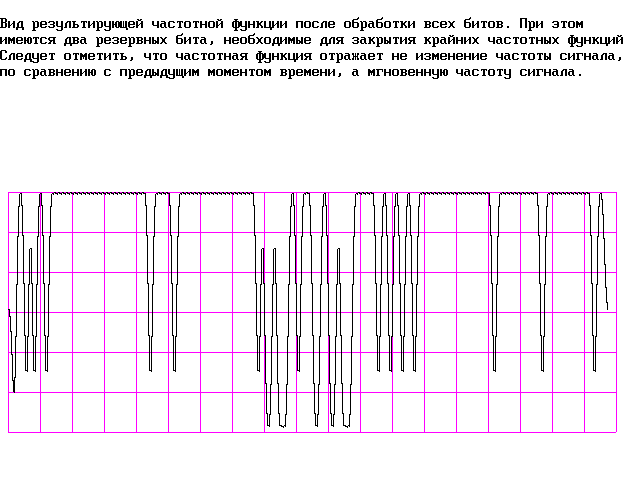


Рисунок 4.7

Результирующая фазовая функция вычисляется по формуле:

.

Фазовая функция не имеет скачков, то есть частотный спектр используется эффективно. Вид ее для тех же условий, что и результирующей частотной, приведен на рисунке 4.8.

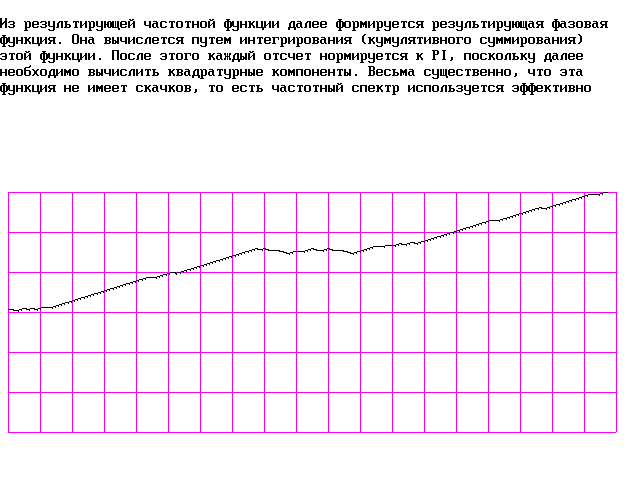


Рисунок 4.8

Синфазная компонента модулированного сигнала вычисляется как Cos от этой функции, а квадратурная – как Sin этой функции. Общее число позиций сигнала определяется как (148+2)\*OSR. Обе компоненты для тех же условий показаны на рисунке 4.9.

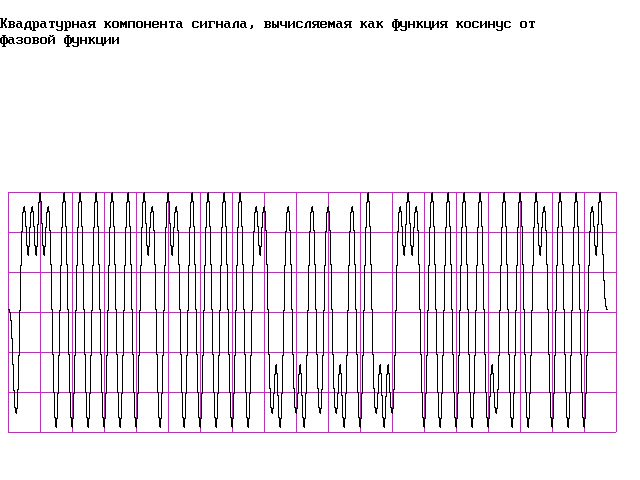
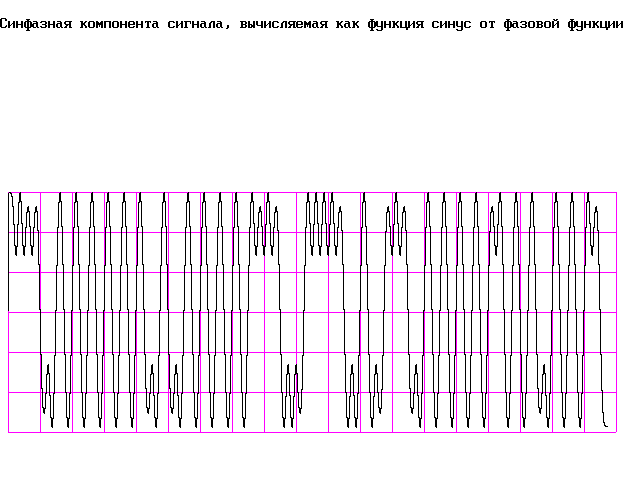


Рисунок 4.9

Наконец, необходимо показать модулированный сигнал. Он может быть вычислен по формуле:

 (4.21)

Для OSR<8 этот сигнал выглядит ненаглядно. На рисунке 4.10 представлен вид этого сигнала для случайных данных и OSR=16.

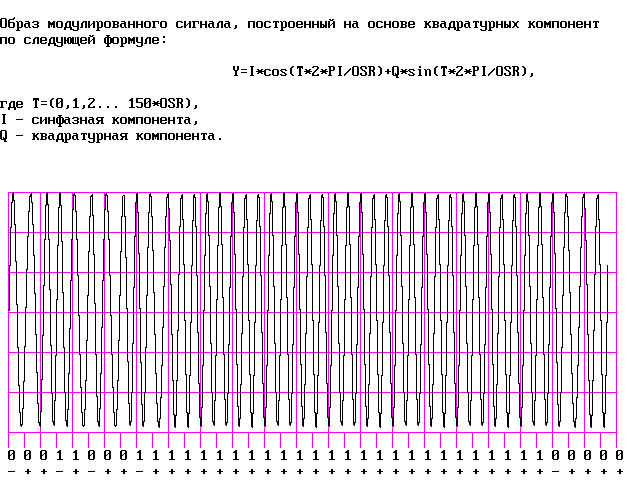


Рисунок 4.10

Если разрешена выдача результатов, на экран выдаются обе квадратурные компоненты. Возможна прокрутка информации. В каждый текущий момент на экране выводятся 12 отсчетов.

4.2Канал связи

На данный момент в программе реализовано по сути два вида канала связи – с добавлением шума и с переотражениями.

Если канал с добавлением шумов, то задание параметра возможно двояко – как фактор или как отношение сигнал/шум.

Фактором называется отношение сигнал/шум, приведенное к энергии сигнала. Вводится целое число, которое затем делится на 100.

Если вводится отношение сигнал/шум, то оно пересчитывается в фактор. Во-первых, из величины в дБ оно переводится в величину в разах. Далее соображения следующие.

1) Отношение сигнал/шум определяется как

 (4.2.1)

1. Сделаем допущение, что полоса частот канала и сигнала одинаковая. Тогда приведенную формулу можно представить в виде:

, (4.2.2)

где А – амплитуда сигнала, которая равна 1.

3) Окончательная формула имеет вид:

, (4.2.3)

Где–дисперсия белого шума с амплитудой отсчетов от -1 до 1, которая равна 1/3.

В обоих видах канала с шумом производится генерация случайного массива длиной 2\*число позиций сигнала. Затем каждый элемент этого массива умножается на фактор, после чего добавляется к сигналу (отдельно к синфазной и квадратурной компоненте).

Если выбран канал с переотражениями, то производится копирование отсчетов из передаваемого в принимаемый сигнал количеством, равным интервалу запаздывания, а далее производится сложение в противофазе двух одинаковых сигналов со смещением, равным интервалу запаздывания.

Интервал запаздывания не может иметь значение, большее величины Lh\*OSR, в противном случае все приведенные выше рассуждения относительно канала связи оказываются некорректными.

4.3Приемник

В состав приемника входят следующие функции:

MaFi – оптимальный фильтр

Make\_Increment – расчет таблицы связи метрики

Viterbi\_detector – алгоритм Витерби

Demux – выделение блока данных из принятого кадра.



Рисунок 4.3.1 Структурная схема приемника

4.3.1ФунцияMaFi

Эта функция высчитывает автокорреляционную функцию ИХ и возвращает отфильтрованный сигнал.

Именно в этой функции используется тренировочная последовательность. Первой операцией является выделение битов с номерами с 6 по 21 из MSK-представления тренировочной последовательности (массив T16).

Далее из принятого сигнала выделяется блок r\_sub с позициями от L/2-(h\*OSR) до L/2+(h\*OSR). Здесь L=половине длины сигнала, а h определяется экспериментальным путем и может принимать значения от 0 до 148. Оно определяет количество элементов подсигнала и, как следствие, количество позиций, на основе которых будет вычисляться автокорреляция.

Далее создается массив оценки канала, который определяется по следующей формуле:



Полученная функция для OSR=16 и отсутствия канала связи приведена на рисунке 4.11.

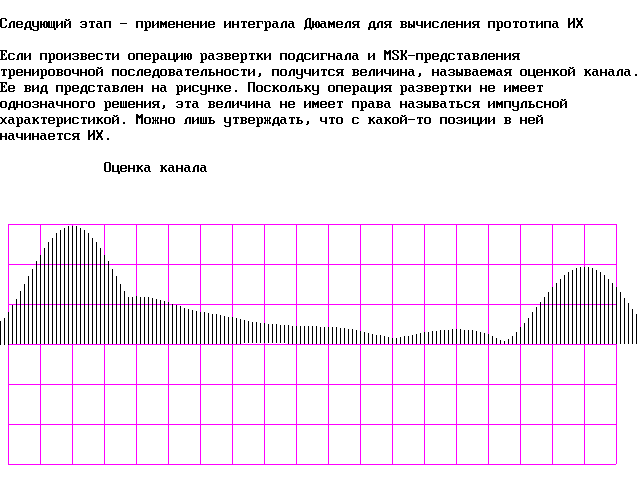


Рисунок 4.11

Можно утверждать, что с какой-то позиции в этой величине начинается импульсная характеристика. Модуль этого массива есть абсолютная величина корреляции. Из этого модуля высчитывается оценка мощности (power\_est), определяемая по формуле

,

где WL=OSR\*(Lh+1).

Полученная функция показана на рисунке 4.12.

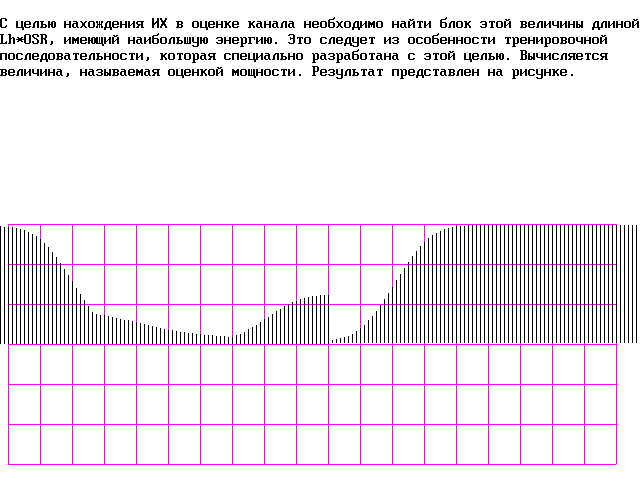


Рисунок 4.12

Если теперь найти максимум массива power\_est, то номер его элемента будет определять позицию оценки импульсной характеристики в массиве Chan\_est. Длина этой оценки составляет WL.

Вид ИХ для OSR=16 и отсутствия канала связи представлен на рисунке 4.13.

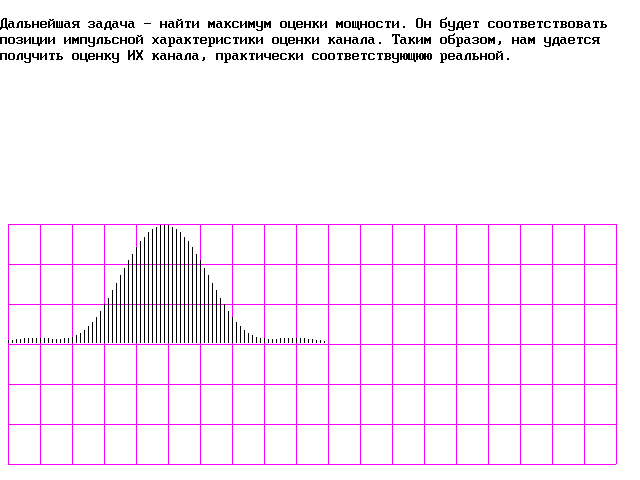


Рисунок 4.13

Далее ищется максимум оценки импульсной характеристики, позиция которого в сумме с позицией максимума оценки мощности будет определять начало принятого сигнала в массиве, который будет сформирован позже.

Эта позиция вычисляется как сумма позиции, с которой начиналось выделение подмассива сигнала + указанное число-OSR\*66. Здесь число 66 образовалось как число битов хвоста+число битов подблока данных + бит управления + пропущенные биты тренировочной последовательности-1.

Однако и этот номер не является окончательным. Теперь необходимо учесть, что в GSMK-модуляторе присутствует задержка на 2 длительности бита, которая и вычитается, получается значение burst\_start.

Теперь возможно высчитать автокорреляционную функцию канала на основе массива оценки импульсной характеристики h\_est. Для этой цели используется формула:

,

Нужная нам часть автокорреляционной функции начинается с середины этого массива, располагается через шаг OSR и имеет количество элементов, равное Lh+1.

Вспомним переменную h. Для того, чтобы сформировать массив принятой реализации, необходимо перед принятым сигналом вставить количество нулей, равное (h+1)\*OSR, далее сам сигнал, количество нулей, равное длине оценки импульсной характеристики и снова (h+1)\*OSR нулей.

Наконец, из полученного массива необходимо выделить принятый сигнал. Каждый отсчет n начинается с позиции

(h+1)\*OSR+burst\_start+n\*OSR,

занимает количество отсчетов, равное длине оценки импульсной характеристики и высчитывается путем умножения на комплексное сопряжение от этой оценки.

Данная функция является очень запутанной для понимания студентами, однако она играет весьма важную роль в построении приемника.

4.3.2 Функция Make\_Increment

Эта функция генерирует таблицу добавки метрики от одного состояния до некоторого другого состояния.

Эти добавки высчитываются следующим образом. Поскольку существуют только два возможных последующих состояния алгоритма, то здесь цикла не требуется. Для каждого из этих двух состояний берутся элементы матрицы Витерби с номером строки, равным текущему обрабатываемому, и домножаются на комплексное сопряжение от автокорреляционной функции. Результаты для всех столбцов суммируются, а затем полученное значение умножается на комплексное сопряжение от элемента матрицы Витерби первого столбца строки с номером, равным этому последующему состоянию. Результат записывается в элемент массива добавок в строку с номером, равным текущему обрабатываемому, столбец с номером, равным последующему состоянию. Эта операция повторяется для всех строк матрицы Витерби.

4.3.3 Функция Viterbi\_detector – алгоритм Витерби

Эта функция содержит необходимый код для реализации алгоритма Витерби.

Сначала инициализируются массивы метрики (Metric) и позиций метрики (SURVIVOR), а также с помощью функции Make\_Increment высчитывается таблица INCREMENT.

Стартовое состояние, очевидно, не имеет метрики, поэтому в качестве исходной метрики взят 0. В качестве базовой операции алгоритма Витерби вычисляется следующее выражение:



Здесь s = элемент массива последующих состояний с номером строки PS. В данном случае PS = стартовому состоянию.

Одновременно заполняется массив Survivor. Его элементы 1 столбца строки S устанавливаются равными номеру стартового символа.

Производится выделение строки с номером START из массива последующих состояний – это массив PREVIOUS\_STATES. Далее организуется цикл количеством итераций Lh-1.

В каждой из этих итераций делается сканирование массива PREVIOUS\_STATES. Прежде всего обнуляется счетчик состояний STATE\_CNTR (перед каждой итерацией). Для каждого элемента этого массива PS вычисляется базовая операция, где n=номеру текущей итерации+1. Здесь элемент массива позиций метрики с номером строки S и номером столбца, равным номеру итераций, устанавливается равным PS. Также создается массив USED, каждый элемент с номером STATE\_CNTR которого определяется номером последующего состояния. Счетчик STATE\_CNTR инкрементируется после каждой базовой операции. После окончания каждой итерации содержимое массива USED переписывается в PREVIOUS\_STATES, то есть количество возможных состояний каждый раз увеличивается в два раза. Конечная длина этого массива составляет 4\*Lh.

После проведения указанных операций мы уже располагаем Lh заполненными столбцами массива метрики. Если Lh нечетно, то соответствующий пройденному количеству метрик элемент матрицы Витерби содержит мнимую часть. Этого далее допустить нельзя.

Поэтому в этом случае требуется пройти еще одно состояние. Это делается аналогично последующим операциям.

Организуется цикл по n от последнего обработанного значения до 148. Для каждого элемента матрицы предыдущих состояний (PS) с номером строки S, который изменяется от 1 до размера матрицы Витерби -1 с шагом 2, выполняются по две базовые операции. В результате получаются две метрики. Из них в соответствии с алгоритмом Витерби нужно выбрать наибольшую[[1]](#footnote-1) метрику и сохранить ее в массиве метрик (s,n) и установить элемент позиций метрик (s,n), равный обрабатываемому предыдущему состоянию. То же самое производится для последующего номера элемента, однако здесь S изменяется от 2 до размера матрицы Витерби с шагом 2. Процесс повторяется для всех n до 148.

После этого мы имеем заполненный последний столбец массива метрики. Именно он нас интересует, а точнее его элементы с номерами, записанными в массиве конечных состояний. Если этот массив содержит два элемента, то выбирается тот, у которого метрика больше, а s устанавливается равным номеру этого конечного состояния.

Теперь возможно вычислить оценку выходной последовательности. Она вычисляется следующим образом. Последний элемент имеет ее имеет значение, равное элементу матрицы Витерби с номером столбца 1 и номером строки, равным s. Каждый предыдущий элемент вычисляется как элемент матрицы Витерби первого столбца и номера строки, равным элементу массива позиций метрики (s,n+1). Вычисление ведется от последнего элемента к первому.

Первый элемент массива принятого кадра вычиляется как

IEST(n)/j

Далее вычисляется полярное представление массива rx\_burst путем применения формулы



И, наконец, принятый кадр преобразуется в двоичное представление по закону rx\_burst=(rx\_burst+1)/2.

4.3.4 Функция DEMux

Эта функция производит выделение из принятого кадра полей данных. Она элементарна и пояснений не требует.

5 ПРИЕМ И ПЕРЕДАЧА С ПРИМЕНЕНИЕМ КОДИРОВАНИЯ

Кодирование и декодирование введено с целью оценить уменьшение количества ошибок при передаче данных с их использованием.

Головной модуль, реализующий данный процесс передачи, имеет имя cod.ovl.

После сброса системы GSM в начале модуля вводится количество циклов приема-передачи, которое будет использоваться при каждой попытке передачи. В каждом цикле передается 4, умещающих в себе один кодированный 260-битный блок данных.

Далее запускается модуль выбора канала для передачи. На этом этапе следует ввести необходимый параметр канала.

После этого выводится рамка и сообщение "Передача…". В течение передачи на экране отображается информация о каждом ее цикле. Количество ошибок отображается для каждого класса кодированного блока отдельно. После отображения 15 циклов программа останавливается и ждет нажатия клавиши. Так продолжается до исчерпания указанного количества циклов.

Первый блок данных представляет собой случайную последовательность, поэтому кодировать его нет смысла. Он передается без участия канала связи.

Каждый цикл предполагает следующую последовательность запуска функций:

Gendata

Chanenc

Interlv

Повторять два раза для передачи целого блока данных

Для каждой из четырех строк матрицы перемежения:

Gsmmod

Chsim

Mafi

Vitdet

Demux

Deinterlv

Chandec

После этого производится сравнение переданного и принятого блоков данных по классам и подсчитывается количество ошибок в каждом классе, а также количество ошибок в кодированном блоке.

По окончании передачи на экран выводится суммарное количество переданных кадров, общее количество битов данных, а также вероятность ошибки по каждому классу и в кодированном блоке в долях единицы.

А теперь разберем процедуру передачи.

5.1Передатчик

Для применения кодирования в состав передатчика введены две новых функции:

Channel\_enc – кодирование

Interleave – перемежение

5.1.1 Общие принципы кодирования

Кодирующий блок дает возможность приемнику GSM обнаружить ошибки при передаче и исправить их.

Функция channel\_enc разделяет входящую информацию, состоящую из 260 битов в три класса – класса Iа, класса Iб и класса II, в зависимости от важности бита. Ошибки битов высших классов влияют на качество передаваемой речи больше, чем биты нижних классов, поэтому разные биты кодируются по-разному в зависимости от их класса. На рисунке 6.1 изображена схема кодирующего блока.



Рисунок 5.1- Схема кодирующего блока

В кодирующем блоке биты класса Iа выбираются из последовательности и подвергаются кодированию четными битами. В GSM, кодер четных битов – систематический циклический кодер на основе трех битов проверки. Слово систематический означает, что четные биты прибавляются к исходной последовательности класса Iа. В результате получается последовательность со следующей структурой:

 ={[k битов данных][ n битов проверки]} (5.1)

Выражение на основе кодера имеет длину 4 бита и следующий вид:

 (5.2)

Остаток r(x) после деления является битами проверки:

, (5.3)

где r – число битов проверки, D(x) – число кодируемых битов и Q(x) – результат деления. Остаток r(x) используется для генерирования битов проверки и формирования . Умножение xr.D(x) эквивалентно сдвигу D(x) влево на r битов.

После кодирования четными битами, биты класса 1а рекомбинируются битами класса 1б, и к этой последовательности прибавляется 4 нуля. Результирующая последовательность, состоящая из 189 битов, передается к сверточному кодеру.

Сверточный кодер берет на входе k битов и возвращает n битов, скоростью сверточного кодера называется отношение k/n, которое в случае GSM равно ½. В сверточном кодере выходной бит зависит не только от текущего бита, а также от нескольких уже прошедших кодирование битов. Число необходимых входных битов для генерирования одного выходного бита называется длиной кодовых ограничений. В стандарте GSM используется кодер с длиной 4. Формулы кодирования даны ниже. Кодер можно изобразить как регистр сдвига (рисунок 6.2).



Рисунок 5.2-Сверточный кодер

5.1.2 Функция Channel\_enc

Функция производит кодирование блока данных длиной 260 бит.

Входной блок данных разделяется на три подблока длиной соответственно по 50, 132 и 78 бит.

К первому из этих блоков справа добавляются три нулевых бита и вычисляется остаток от обратной свертки полученного массива с вектором g=[1011]. Эта операция использует следующую формулу:

,

где q = результат свертки, r = остаток от свертки.

Из остатка выделяются три последних элемента. В соответствии с четностью или нечетностью этих элементов устанавливаются биты паритета, которые затем подсоединяются к первому блоку данных.

После этого к первому блоку данных подсоединяется второй, а также четыре нулевых бита.

Спереди к полученной последовательности подсоединяются также четыре нулевых бита, образуется последовательность Data\_seq. Далее каждый бит выходной последовательности вычисляется по следующей формуле:



Выходная последовательность размером 456 бит образуется путем соединения последовательности Encoded с третьим блоком исходных данных (78 бит).

5.1.3 Функция Intearleave

Блок перемежения смешивает биты выходного сигнала, идущие от кодирующего блока, и распределяет их по нескольким кадрам. После перемежения, ошибки в данных при приеме являются некоррелированными.

Блок перемежения работает по следующим формулам:

b = ((57⋅(T mod 4) + t⋅32 + 196 (t mod 2)) mod 456) (5.4)

B = ((T - (b mod 8)) div 4), (5.5)

где бит t в tx\_data, соответствующий кадру T, находится в кадре B в tx\_enc как бит b, (x mod y) – остаток после деления x/y, а (x div y) – результат деления x/y. Операции (5.4) и (5.5) изображены на рисунке 5.3.

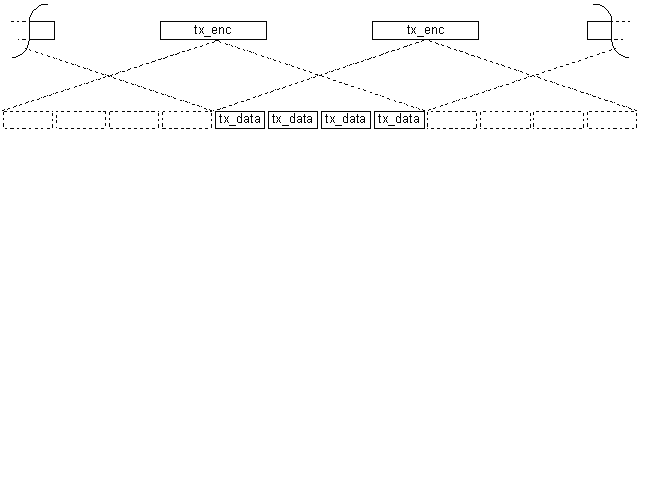
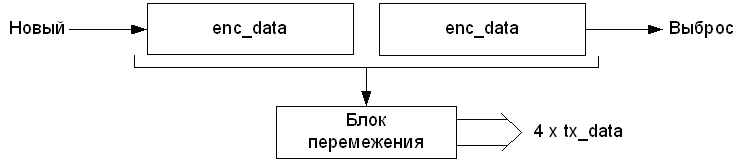


Рисунок 5.3 -Операция перемежения по формулам (5.4) и (5.5)

После анализа работы блока перемежения видно, что блок перемежения действует на две переменные типа tx\_enc одновременно, и после каждой операции перемежения возвращает четыре переменные типаtx\_data. Далее эти данные обрабатываются в генераторе кадра. Две переменные типа tx\_enc содержат по (456 х 2) битов, и четыре переменные типаtx\_data содержат только 456 битов. Очевидно, что не все биты со входа блока перемежения попадают на выход, поэтому каждая входная переменная типа tx\_enc подвергается перемежению два раза. В программе это делается путем создания очереди переменных tx\_enc, при каждой операции перемежения блок перемежения использует полное содержание очереди. Очередь имеет два слота, и при каждой операции перемежения второй блок выбрасывается, первый блок данных сдвигается на место второго и новый блок данных помещается вместо первого. Операция перемежения изображена на рисунке 5.4.



Рисуно. 5.4 Операция перемежения с помощью очереди

При такой организации передача одного 260-битного блока производится за 8 кадров – две матрицы передачи (в каждой матрице смешано по два одинаковых блока данных). В этих матрицах блоки данных меняются местами, поэтому в совокупности они дают полностью первый блок данных. Соответственно вторая и третья матрица дают полностью второй блок данных.

5.2 Приемник

В приемнике с кодированием все обстоит гораздо сложнее. В его состав входят две функции:

DeInterleave – обратное перемежение

Channel\_dec – декодирование.

5.2.1 Функция DeInterleave

Эта функция полностью противоположна функции Interleave.

Принятые матрицы соединяются по столбцам (то есть итоговая матрица имеет в два раза большее число строк). Для элементов выходных блоков данных справедливы следующие соотношения:

R = 4⋅B + (bmod 8) (5.6)

r = 2⋅((49⋅b) mod 57) + ((b mod 8) div 4), (5.7)

где b – номер бита в упорядоченном кадре, B – номер упорядоченного кадра rx\_enc, R – номер burst в rx\_data, r – порядковый номер в burst.

Анализируя (6.6) и (6.7), можно увидеть, что блок обратного перемежения может одновременно действовать на восьми наборах rx\_data. Каждый проход функции возвращает один набор rx\_enc. Очевидно, что не все биты, поступающие на вход блока, передаются на выход, так как 8 наборов rx\_data содержат 912 (2⋅456) битов, а rx\_enc содержит только 456 битов. Поэтому каждый набор rx\_data обрабатывается два раза. Практически это осуществляется путем создания очереди, как показано на рисунке 5.5.

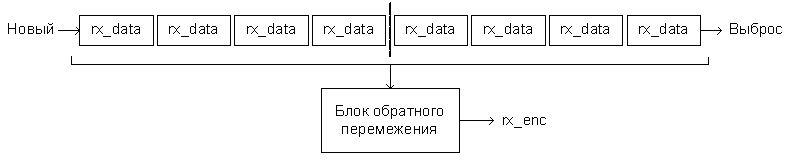


Рисунок 5.5- Блок обратного перемежения

5.2.2 Функция Channel\_Dec

Эта функция производит декодирование. Она является самой сложной в разрабатываемом пакете.

В соответствии с законом кодирования правые 78 бит являются некодированными, поэтому их можно сразу отсоединить.

Декодирование имеет в своей основе тот же алгоритм Витерби. Только здесь матрица Витерби заранее не определена, а метрика определяется наоборот. Далее будет понятно, что это означает.

Массив последующих состояний в целом определен заранее и строится следующим образом:

NEXT(n,:) = [zeroinonein] (5.8)

NEXT(n+1,:) = NEXT(n,:) (5.9)

Здесь n изменяется от 1 до 15 с шагом 2, переменные ZeroIn и OneIn имеют начальные значения 1 и 9 и инкрементируются с каждой последующей итерацией.

В результате имеем матрицу вида:

1 9

1 9

2 10

2 10

3 11

3 11

4 12

4 12

5 13

5 13

6 14

6 14

7 15

7 15

8 16

8 16

Матрица предыдущих состояний определяется как

PREVIOUS(n,:) = [n+offsetn+offset+1] (5.10)

для n от 1 до 8. Далее производится операция копирования содержимого верхней половины матрицы в нижнюю. Итоговая матрица:

1 2

3 4

5 6

7 8

9 10

11 12

13 14

15 16

1 2

3 4

5 6

7 8

9 10

11 12

13 14

15 16

Далее создается таблица парабитного декодирования. Она имеет следующий вид:

0 NaNNaNNaNNaNNaNNaNNaN 3 NaNNaNNaNNaNNaNNaNNaN

3 NaNNaNNaNNaNNaNNaNNaN 0 NaNNaNNaNNaNNaNNaNNaN

NaN 3 NaNNaNNaNNaNNaNNaNNaN 0 NaNNaNNaNNaNNaNNaN

NaN 0 NaNNaNNaNNaNNaNNaNNaN 3 NaNNaNNaNNaNNaNNaN

NaNNaN 0 NaNNaNNaNNaNNaNNaNNaN 3 NaNNaNNaNNaNNaN

NaNNaN 3 NaNNaNNaNNaNNaNNaNNaN 0 NaNNaNNaNNaNNaN

NaNNaNNaN 3 NaNNaNNaNNaNNaNNaNNaN 0 NaNNaNNaNNaN

NaNNaNNaN 0 NaNNaNNaNNaNNaNNaNNaN 3 NaNNaNNaNNaN

NaNNaNNaNNaN 1 NaNNaNNaNNaNNaNNaNNaN 2 NaNNaNNaN

NaNNaNNaNNaN 2 NaNNaNNaNNaNNaNNaNNaN 1 NaNNaNNaN

NaNNaNNaNNaNNaN 2 NaNNaNNaNNaNNaNNaNNaN 1 NaNNaN

NaNNaNNaNNaNNaN 1 NaNNaNNaNNaNNaNNaNNaN 2 NaNNaN

NaNNaNNaNNaNNaNNaN 1 NaNNaNNaNNaNNaNNaNNaN 2 NaN

NaNNaNNaNNaNNaNNaN 2 NaNNaNNaNNaNNaNNaNNaN 1 NaN

NaNNaNNaNNaNNaNNaNNaN 2 NaNNaNNaNNaNNaNNaNNaN 1

NaNNaNNaNNaNNaNNaNNaN 1 NaNNaNNaNNaNNaNNaNNaN 2

Аналогично создается таблица однобитного декодирования, которая имеет следующий вид:

0 NaNNaNNaNNaNNaNNaNNaN 1 NaNNaNNaNNaNNaNNaNNaN

0 NaNNaNNaNNaNNaNNaNNaN 1 NaNNaNNaNNaNNaNNaNNaN

NaN 0 NaNNaNNaNNaNNaNNaNNaN 1 NaNNaNNaNNaNNaNNaN

NaN 0 NaNNaNNaNNaNNaNNaNNaN 1 NaNNaNNaNNaNNaNNaN

NaNNaN 0 NaNNaNNaNNaNNaNNaNNaN 1 NaNNaNNaNNaNNaN

NaNNaN 0 NaNNaNNaNNaNNaNNaNNaN 1 NaNNaNNaNNaNNaN

NaNNaNNaN 0 NaNNaNNaNNaNNaNNaNNaN 1 NaNNaNNaNNaN

NaNNaNNaN 0 NaNNaNNaNNaNNaNNaNNaN 1 NaNNaNNaNNaN

NaNNaNNaNNaN 0 NaNNaNNaNNaNNaNNaNNaN 1 NaNNaNNaN

NaNNaNNaNNaN 0 NaNNaNNaNNaNNaNNaNNaN 1 NaNNaNNaN

NaNNaNNaNNaNNaN 0 NaNNaNNaNNaNNaNNaNNaN 1 NaNNaN

NaNNaNNaNNaNNaN 0 NaNNaNNaNNaNNaNNaNNaN 1 NaNNaN

NaNNaNNaNNaNNaNNaN 0 NaNNaNNaNNaNNaNNaNNaN 1 NaN

NaNNaNNaNNaNNaNNaN 0 NaNNaNNaNNaNNaNNaNNaN 1 NaN

NaNNaNNaNNaNNaNNaNNaN 0 NaNNaNNaNNaNNaNNaNNaN 1

NaNNaNNaNNaNNaNNaNNaN 0 NaNNaNNaNNaNNaNNaNNaN 1

Физические основы декодирования следующие. Если посмотреть на закон кодирования, то очевидно, что первая битовая пара кодированного блока определяется только одним информационным битом (т.к. первые 4 бита кодируемой последовательности заведомо равны 0). Поэтому если эти два бита различны, мы ничего не можем сказать о соответствующем информационном бите и обе метрики устанавливаем равными единице. Если биты пары одинаковы, то метрики устанавливаются 0и 2 соответственно для двух переходах по дереву состояний.

Здесь вместо Lh у нас выступает фиксированное число 4 (т.к. в начале кодируемой последовательности было четыре нулевых бита). Длина сканируемого массива VISITED\_STATES изменяется от 1 до 24. Далее все происходит аналогично алгоритму Витерби, причем в роли s выступает текущий элемент VISITED\_STATES, а в роли PS – текущий элемент массива NEXT. То есть предыдущие состояния как бы стали последующими, а последующие – предыдущими.

В каждом цикле вычисляется переменная LAMBDA, значение которой определяется соответствующим элементом таблицы парабитного кодирования и равно операции XOR над некоторыми битами кодированного блока и 0 или 1. Эта переменная играет роль INCREMENT-массива.

Массив STATE заполняется текущими элементами VISITED\_STATES. Массив VISITED\_STATES пополняется каждую итерацию элементами массива NEXT.

Следует отметить, что каждую итерацию производится сдвиг метрики на один столбец влево.

Далее операция производится совершенно аналогично алгоритму Витерби с указанными изменениями.

Необходимо найти минимальную метрику столбца 1 и запомнить ее номер как стоповое состояние. Если оно оказывается равным нулю, устанавливается флаг совпадения.

Далее создается массив выходной последовательности, который первоначально заполняется номером минимальной метрики, а затем каждый предыдущий его элемент n заменяется элементом STATE с номером столбца n+1 и строки с номером, равном последующему. Потом первый элемент этой последовательности устанавливается равным стартовому состоянию.

Теперь из таблицы однобитного декодирования выбираются элементы, номера строк которых равны элементам полученной последовательности, а номера столбцов – они же на единицу больше.

Наконец, формируется выходной блок – 50 первых бит из полученной последовательности, далее ее биты 54-185 и некодированный блок, отсоединенный в начале процедуры.

6 МОДУЛИ ДЕМОНСТРАЦИИ

Цель модулей демонстрации – наглядно показать процесс образования сигналов или последовательности логических операций.

Всего в программе девять модулей демонстрации. Большинство их повторяют модули ядра, то есть выполняют те же функции, однако комментируют выполнение соответствующими рисунками и сообщениями.

6.1 Модули демонстрации образования сигналов

Модули демонстрации образования сигналов запрашивают при необходимости тип блока данных и тип канала связи. Для модулей демонстрации образования элементарной частотной функции, модулятора и согласованного фильтра выводимая информация приведена на рисунках 6.1-6.19. Условия: OSR=16, блок данных случайный, канал связи отсутствует. **Соответствующие биты кадра выводятся только при OSR=16.**

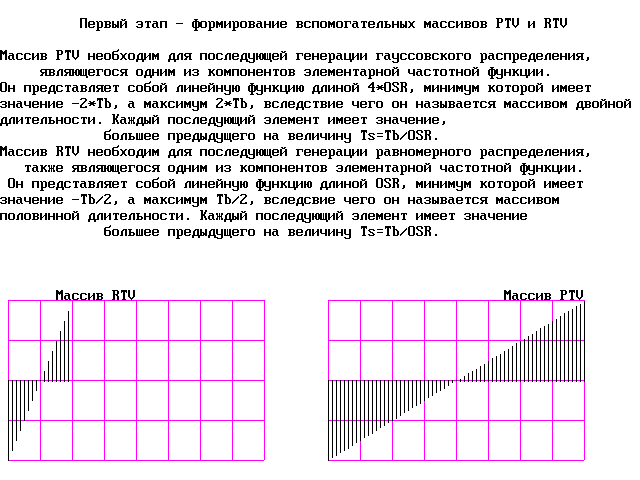


Рисунок 6.1

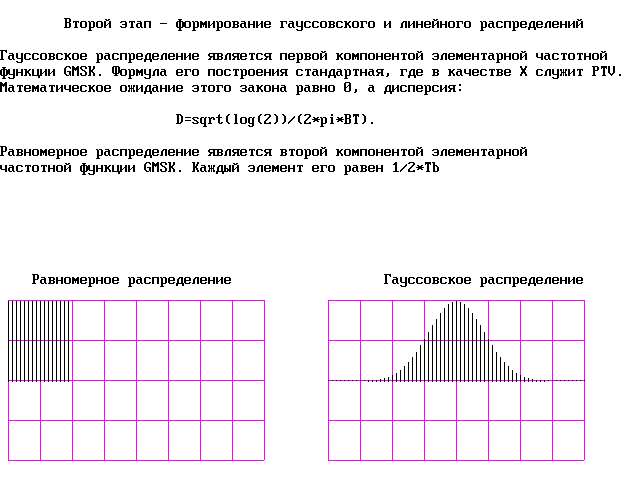


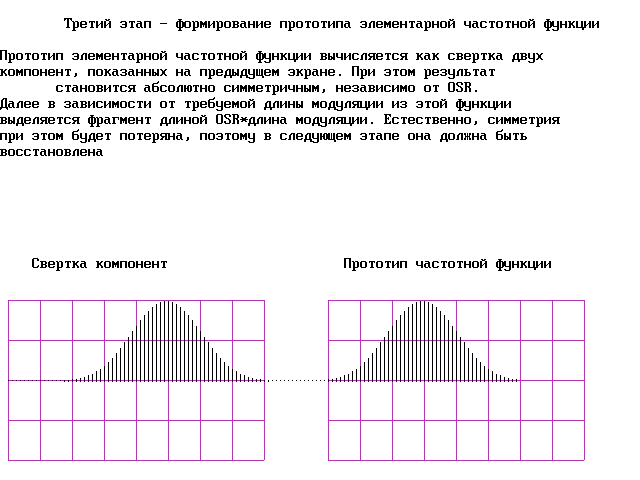
Рисунок 6.2

Рисунок 6.3

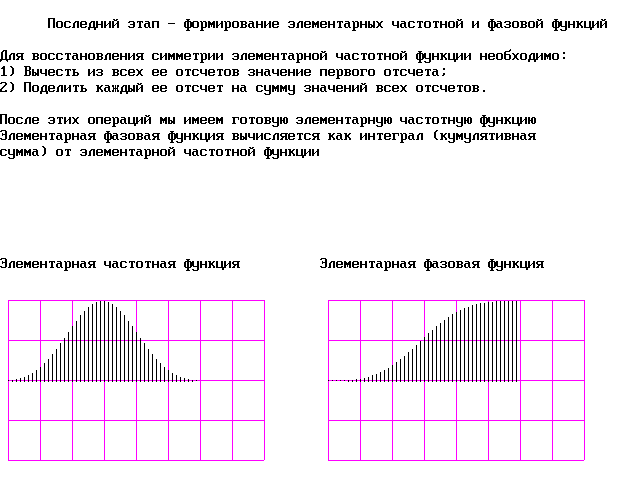


Рисунок 6.4

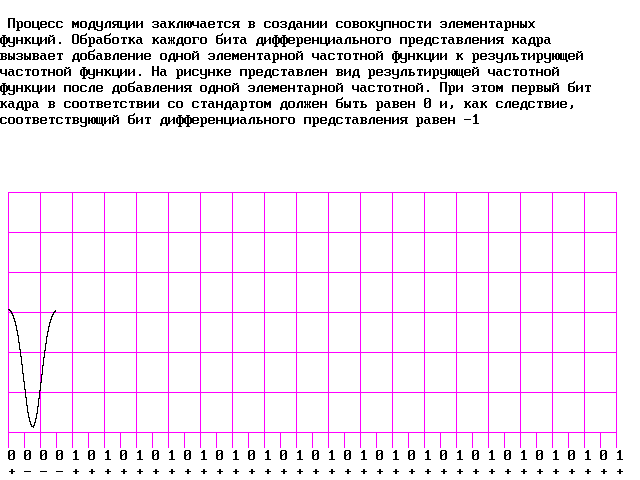


Рисунок 6.5

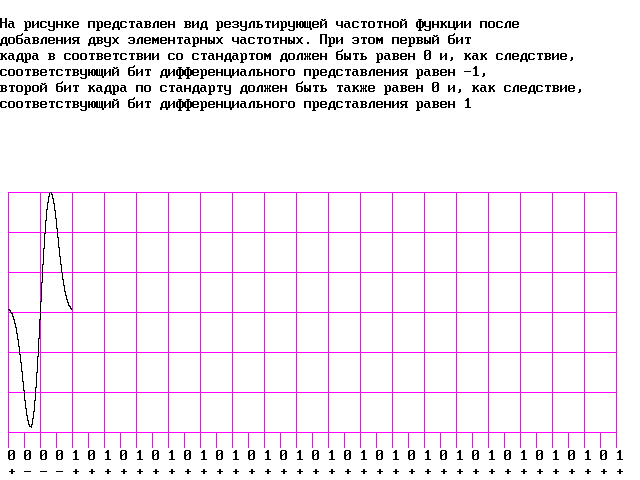


Рисунок 6.6

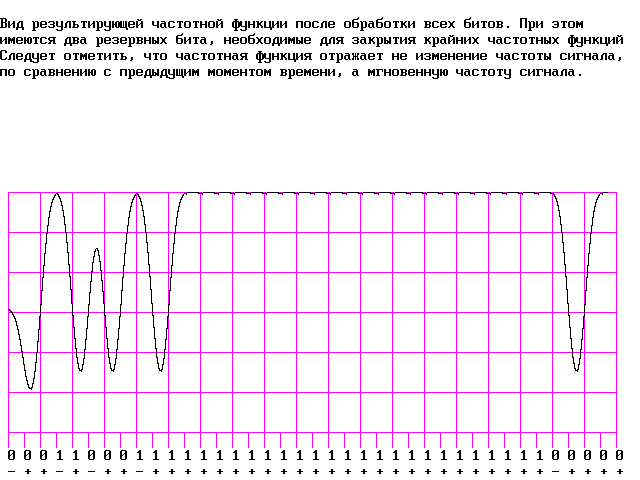


Рисунок 6.7

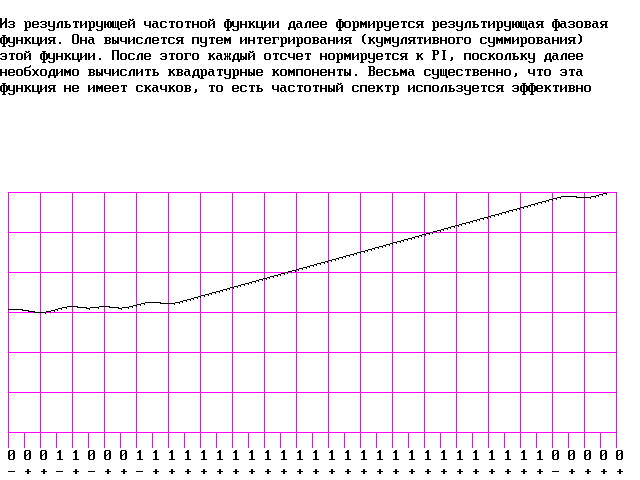


Рисунок 6.8

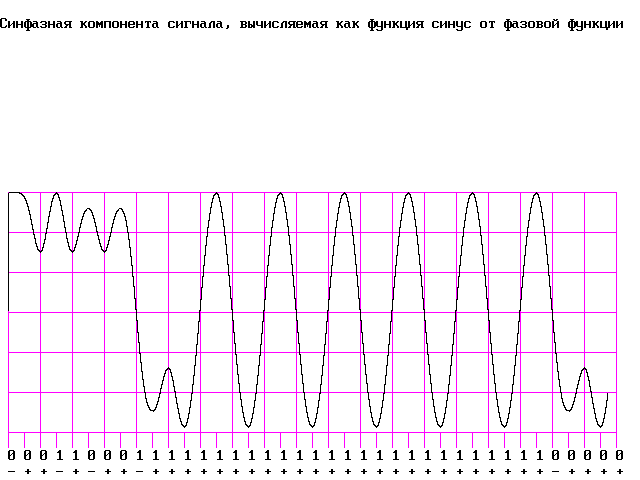


Рисунок 6.9

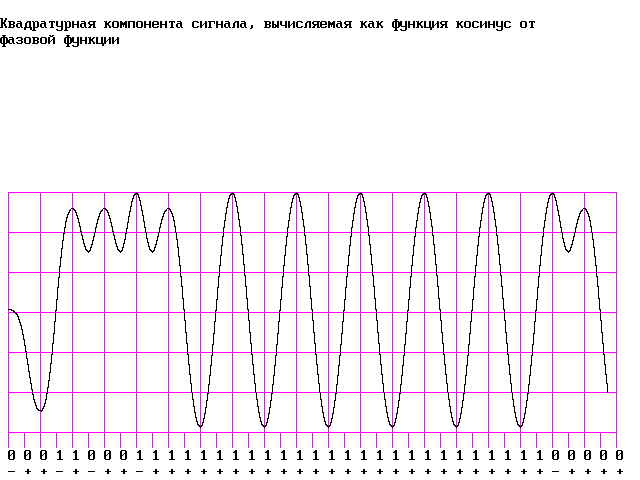


Рисунок 6.10

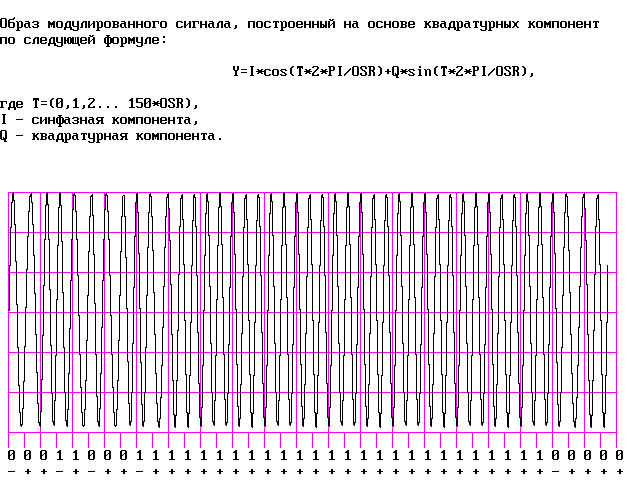


Рисунок 6.11

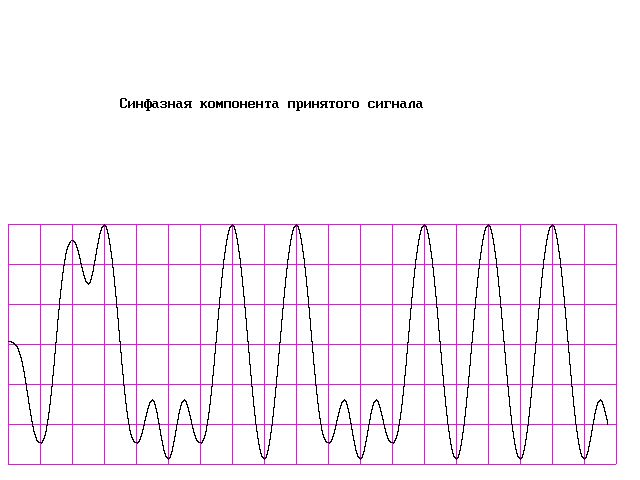


Рисунок 6.12

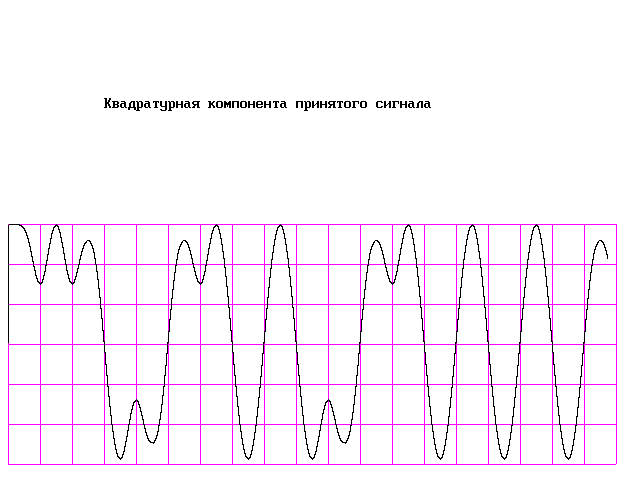


Рисунок 6.13

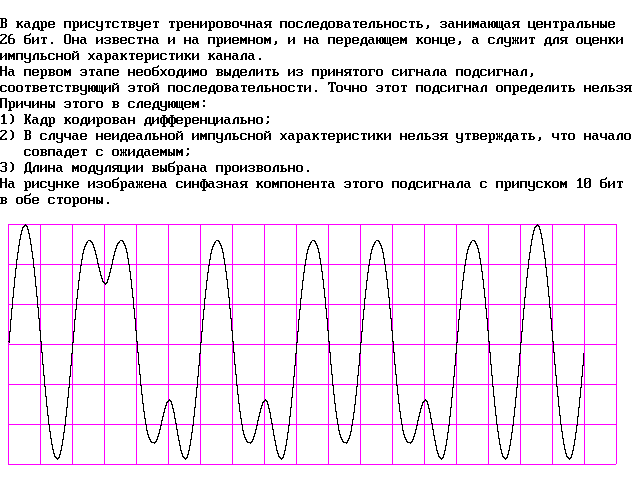


Рисунок 6.14

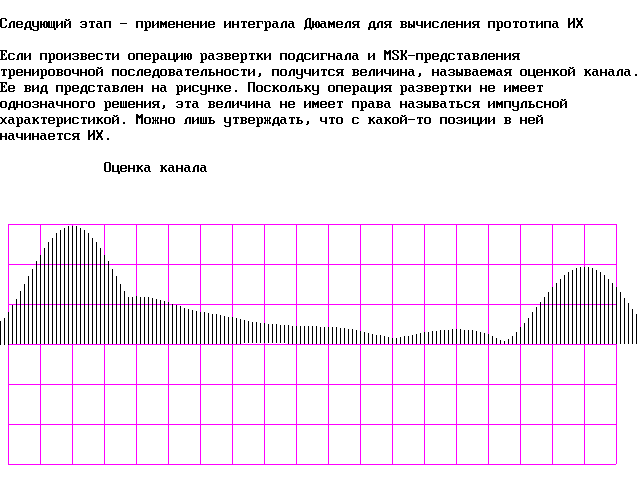


Рисунок 6.15

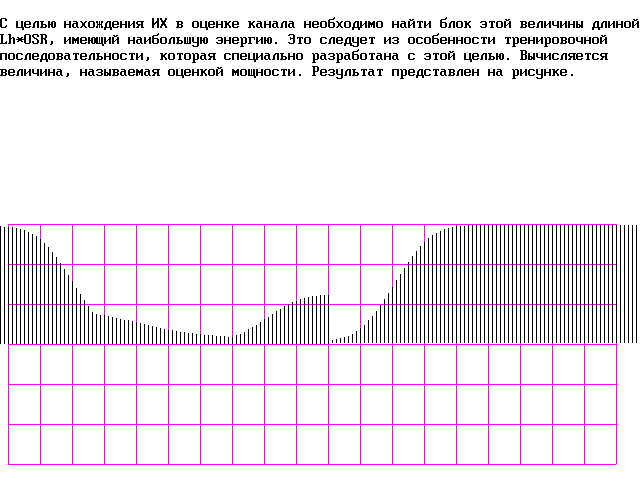


Рисунок 6.16

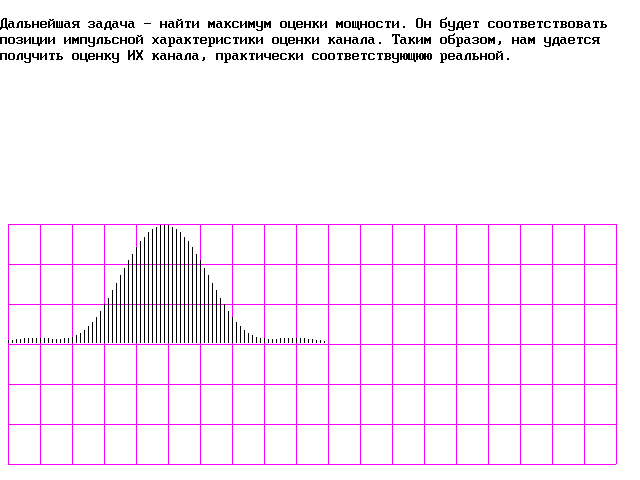


Рисунок 6.17

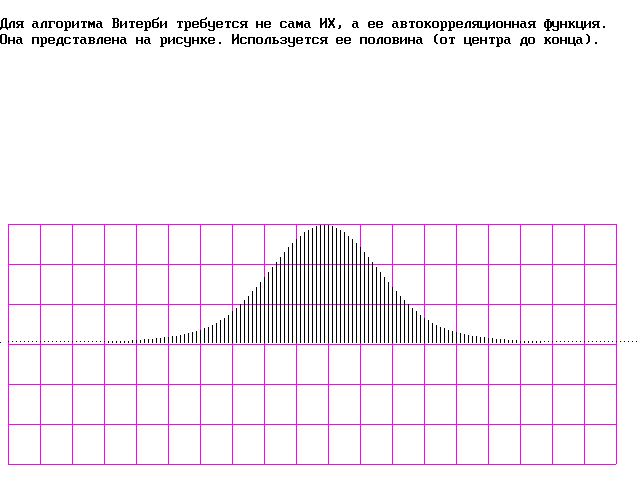
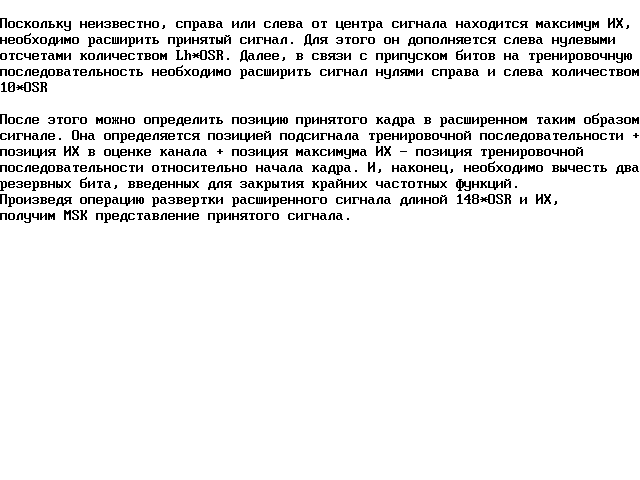


Рисунок 6.18



6.2 Модули вывода переменных

В эту категорию входят модуль вывода структуры GSM и модуль вывода таблиц алгоритма Витерби. Выводимая модулем OutVar информация:

Начальный указатель последовательности : 5

Длительность одного бита кадра : 3.692000e-6

База сигнала GMSK : 0.30

Число шагов модуляции на один бит : 4

Количество битов данных в одном кадре : 114

Число отсчетов имп. характеристики канала: 4

Тренировочная последовательность : 00100101110000100010010111

Занятый вектор прерывания : 2F-48534D20

Выводимая модулем VitOut информация полностью соответствует тем матрицам, которые были приведены при рассмотрении инициализации алгоритма Витерби.

6.3 Модули демонстрации логических операций

Модули демонстрации алгоритма Витерби и процесса декодирования очень трудно реализуемы, поэтому они заменены просто на пояснения соответствующих процессов и здесь не приводятся.

Процесс кодирования, образования кадра и образования MSK-представления двоичной последовательности реализованы полностью. Один из экранов модуля демонстрации кодирования показан ниже.

┌────────────────────────────────────────────────────────────────────────┐

│ │

│ │

│ Первый этап - разделение блока на классы. │

│ │

│ Класс Iа имеет длину 50 бит. В дальнейшем для него │

│ подсчитываются биты паритета и сверточный код. │

│ Класс Ib имеет длину 132 бита. Для него считается только код. │

│ Класс II имеет длину 78 бит. Это некодируемый класс. │

│ │

│ Класс Ia: 01100000011111111111111111100000100111000111111111 │

│ │

│ Класс Ib: 111111000110111000000001111111111 │

│ 110000011100001111100011111111100 │

│ 011000000100011000000111111000001 │

│ 001110010001001110001110001101110 │

│ Класс II: │

│ 001110001110000000000000111000000000000 │

│ 000011111111110000001111111111111000111 │

└────────────────────────────────────────────────────────────────────────┘

6.3 Модули тестирования и пошагового режима

Модуль пошагового режима предназначен главным образом для просмотра результатов выполнения функций. В нем все команды вводятся с клавиатуры, причем имя оверлейного модуля совпадает с командой. Есть также и другие команды – установка размера блока данных, установка/сброс флага выдачи результатов и др. Все команды выдаются на экран при вводе команды Help.

Этот модуль также способен выполнить команды, записанные в некотором текстовом файле, для чего существует команда Run<файл>.

# БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Громаков Ю.А. Сотовые системы подвижной радиосвязи.1994.
2. Громаков Ю.А. Стандарты и системы подвижной радиосвязи. М. Мобильные ТелеСистемы – Эко-Трендз. 1997.
3. Гольденберг Л.М., Матюшкин Б.Д., Поляк М.Н. Цифровая обработка сигналов. – М.: Радио и связь, 1990.
4. Кларк Дж., Кейн Дж. Кодирование с исправлением ошибок в системах цифровой связи. –М.: Радио и связь, 1987.
5. Ли У.К. Техника подвижных систем связи. –М.: Радио и связь, 1985.
6. Матвеенко И.П. Сотовая связь: сегодня и завтра. –М., 1994.
7. Рабинер Л.Р., Гоулд Б. Теория и применение цифровой обработки сигналов. –М.: Мир, 1978.
8. Ратынский М.В. Основы сотовой связи / Под ред. Д.Б. Зимина.-2-е изд., перераб. И доп. –М.: Радио и связь, 2000. –248 с.: ил.
9. Arne NorreEkstrøm, Jan H. MikkelsenGSMsim. A MATLAB Implementation of a GSM Simulation Platform / Institute of Electronic Systems, Division of Telecommunications, Aalborg University, 1997. – 136 стр.

1. На самом деле выбирается МЕНЬШАЯ метрика, поскольку в массиве метрик мы храним только возможные дальнейшие метрики. [↑](#footnote-ref-1)