

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ  
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение  
высшего образования

«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Политехнический институт  
Факультет информационных  
технологий и электроники

Кафедра  
«Информационная безопасность  
систем и технологий»

Специальность 10.05.03 –

Информационная безопасность  
автоматизированных систем  
Защищённые автоматизированные  
системы управления

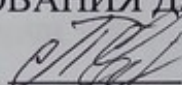
Специализация -

ДИПЛОМНЫЙ ПРОЕКТ

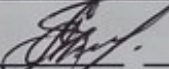
на тему

ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ МОДУЛЯ ПОМЕХОУСТОЙЧИВОГО  
КОДИРОВАНИЯ ДАННЫХ

Студент

 Стенькин Павел Александрович

Руководитель начальник НИС – 251, АО «ПНИЭИ»

 Павликов Е. А.

Нормоконтролёр

 Иванов А. П.

Рецензент зам. ген. директора АО «ПНИЭИ»

 Сомкин С. А.

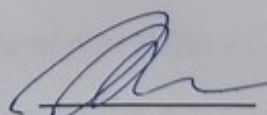
Работа допущена к защите (протокол заседания кафедры от 10.06.22 № 9 )

Заведующий кафедрой

 С. Л. Зефирова

Работа защищена с отметкой хорошо (протокол заседания ГЭК  
от 23.06.22 № 9 )

Секретарь ГЭК

 А. П. Иванов

Пенза 2022

## Оглавление

|                                                                                                                            |    |
|----------------------------------------------------------------------------------------------------------------------------|----|
| Введение .....                                                                                                             | 5  |
| 1 Анализ состояния рассматриваемой технической проблемы .....                                                              | 7  |
| 1.1 Выбор метода построения помехоустойчивой защиты .....                                                                  | 7  |
| 1.2 Выбор и обоснование помехоустойчивого кода .....                                                                       | 13 |
| 1.3 Выбор и обоснование каскадного кода .....                                                                              | 16 |
| 1.4 Расчёт длины кодовой комбинации внутреннего кода.....                                                                  | 19 |
| 1.5 Расчет корректирующих способностей внутреннего кода.....                                                               | 22 |
| 1.6 Описание выбранного внутреннего кода .....                                                                             | 26 |
| 1.7 Описание внешнего кода .....                                                                                           | 30 |
| 1.8 Система циклового фазирования .....                                                                                    | 31 |
| 1.9 Выбор способа реализации УЗО .....                                                                                     | 33 |
| 2 Разработка программного обеспечения модуля помехоустойчивого кодирования малогабаритной аппаратуры передачи данных ..... | 35 |
| 2.1 Программная модель малогабаритной аппаратуры передачи данных.....                                                      | 35 |
| 2.2 Описание межмодульного взаимодействия.....                                                                             | 35 |
| 2.3 Обоснование выбора языка программирования.....                                                                         | 37 |
| 2.4 Описание программного обеспечения модуля помехоустойчивого кодирования.....                                            | 38 |
| 2.5 Описание алгоритма кодирования кодом Рида - Соломона.....                                                              | 39 |
| 2.6 Описание алгоритма приема информации.....                                                                              | 41 |
| 2.7 Описание алгоритма декодирования кодом Рида–Соломона .....                                                             | 45 |

|                                                                                                                                                                                           |    |
|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----|
| 3 Разработка организационных и технических мероприятий для снижения воздействия психофизических факторов, влияющих на психоэмоциональное состояние специалиста по защите информации ..... | 57 |
| 3.1 Вредные психофизиологические факторы, влияющие на психоэмоциональное состояние специалиста по защите информации.....                                                                  | 57 |
| 3.2 Методы и средства физической культуры, снижающие воздействие психофизиологических факторов, влияющих на психоэмоциональное состояние специалиста по защите информации .....           | 59 |
| 4 Технология разработки программного обеспечения малогабаритных устройств передачи данных .....                                                                                           | 61 |
| Заключение .....                                                                                                                                                                          | 69 |
| Список используемых источников .....                                                                                                                                                      | 70 |
| Приложение А Текст программного обеспечения модуля помехоустойчивого кодирования.....                                                                                                     | 71 |
| Приложение Б Результат тестирования.....                                                                                                                                                  | 81 |

## Обозначения и сокращения

В дипломном проекте используются следующие обозначения и сокращения:

АПД – аппаратура передачи данных;

МАПД – малогабаритная аппаратура передачи данных;

ИС – источник сообщения;

ПО – программное обеспечение;

ТЗ – техническое задание;

КС – канал связи;

АКП – автоматический коммутатор пакетов;

УПС – устройство приёма/передачи сообщения;

ПС – получатель сообщения;

ВВХ – вероятностно-временные характеристики;

РС-код – код Рида – Соломона;

ФП – фазирующая посылка;

ЗУ – запоминающее устройство;

ООД – окончное оборудование данных.

## Введение

В настоящее время существует огромная потребность в передаче больших объёмов информации. Данная потребность заключается в значимости самой информации, поэтому, при передаче по каналам связи необходимо соблюдать три свойства безопасности информации: конфиденциальность, целостность, доступность.

Защита информации производится с применением специальных методов: многократное повторение, параллельная передача, помехоустойчивое кодирование информации, комбинированный метод, каналы с обратной связью.

Среди перечисленных методов защиты информации от ошибок особое место занимает метод помехоустойчивого кодирования информации.

Помехоустойчивое кодирование используется для защиты данных в памяти вычислительных устройств, а также для защиты от неправильного функционирования или шумов в цифровых логических цепях.

Технические средства, позволяющие передавать информацию по каналам связи, строятся на основе аппаратуры передачи данных (АПД) с обязательной функцией помехоустойчивой защиты информации, основанной на математических алгоритмах защиты информации и реализуемой в устройстве защиты от ошибок (УЗО), являющемся составной частью АПД.

Модуль помехоустойчивого кодирования, к которому разрабатывается программное обеспечение, является составной частью малогабаритной аппаратуры передачи данных, использующейся в низовых звеньях управления войсками. Данный модуль призван защищать передаваемую информацию от помех, которые в низовых звеньях управления являются явлением постоянным. И хотя информация, передаваемая на данном уровне, не имеет большой актуальности защищать ее все же необходимо, так как от нее зависят человеческие жизни.

Данный модуль целесообразно применять во всех звеньях управления войсками, начиная с самых нижних, поскольку именно там вероятность

появления помех как искусственных, так и естественных (горы, овраги, влага) очень велика.

Поэтому целью ВКР является разработка программного обеспечения модуля помехоустойчивого кодирования в МАПД.

Целью работы является создание ПО, реализующего функции помехоустойчивого кодирования.

## 1 Анализ состояния рассматриваемой технической проблемы

### 1.1 Выбор метода построения помехоустойчивой защиты

Помехоустойчивое кодирование используется для защиты данных в памяти вычислительных устройств, а также для защиты от неправильного функционирования или шумов в цифровых логических цепях. Коды используются также для сжатия данных.

Поскольку электромагнитный спектр всё больше и больше заполняется создаваемым человеком сигналом, помехоустойчивые коды позволяют устройствам передачи данных надёжно работать при наличии интерференции. В вооружённых силах помехоустойчивые коды используются для защиты против намеренно организованной противником интерференции.

Одной из особенностей помехоустойчивой защиты является её восстанавливающая способность. При выборе подходящего кода, контролирующего ошибки, возникает возможность в восстановлении утерянных пакетов по известным пакетам.

Система связи соединяет источник данных с получателем данных посредством канала. На рисунке 1 представлена структурная схема системы передачи сообщений и на плакате “Форматы информационных кадров кода Рида-Соломона, кода Хемминга, каскадного кода”, лист 1.

В качестве источника, вырабатывающего для передачи сообщения (ИС), могут использоваться телефонные аппараты, автоматизированные датчики, ЭВМ и т. п.

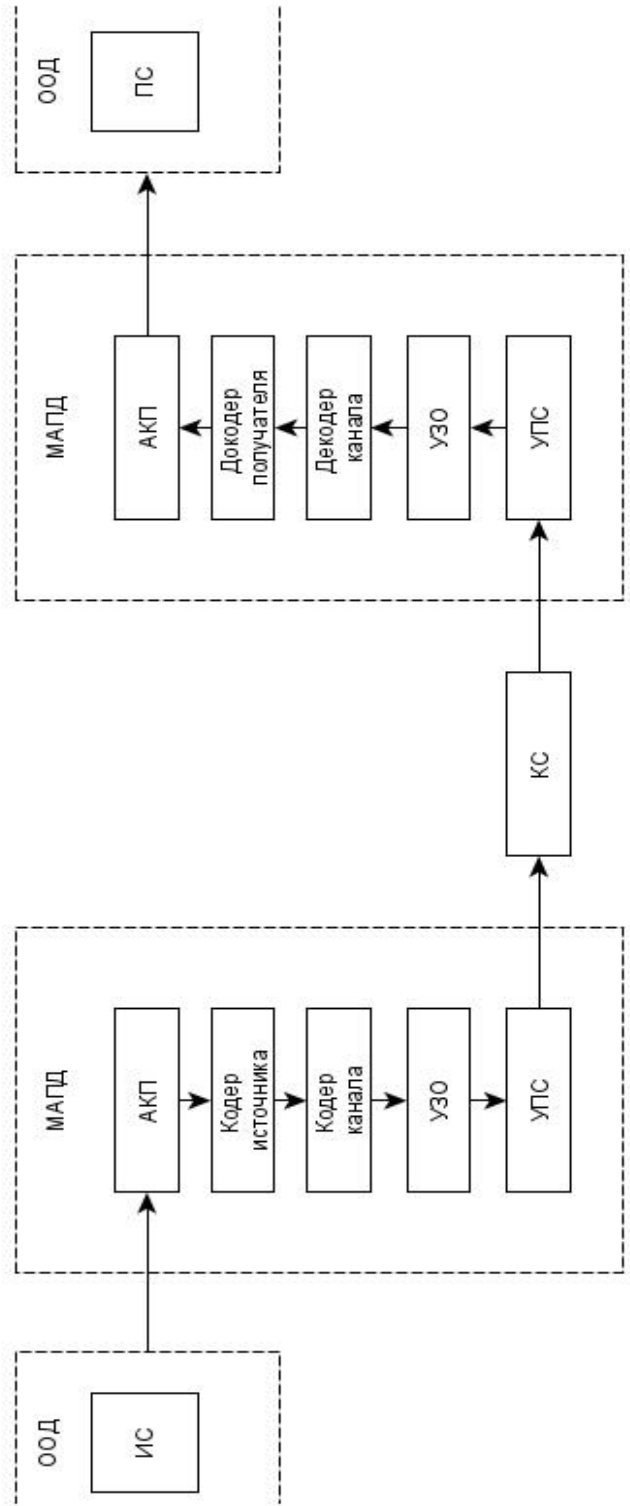


Рисунок 1- Структурная схема системы передачи дискретных сообщений

С целью обеспечения возможности передачи системой связи сообщений, формируемых различными источниками, устанавливается автоматический коммутатор пакетов (АКП). АКП реализует стык с ООД для ввода (вывода) пакетов информации и команд управления, а также осуществляющий адресование и маршрутизацию введенных пакетов информации по специальным протоколам взаимодействия.

Данные, поступающие от АКП, обрабатываются кодером источника, предназначенным для более компактного представления данных. Это промежуточное представление является последовательностью символов, которая называется кодовым словом источника. Затем данные обрабатываются кодером канала, преобразующим последовательность символов кодового слова источника в другую последовательность символов, называемую кодовым словом канала. Кодовое слово канала представляет более длинную последовательность с большей, чем у кодового слова источника, избыточностью. Каждый символ кодового слова канала может быть представлен битом или, возможно, группой битов.

УЗО, включаемые на входе дискретного канала, служат для обеспечения передачи дискретных сообщений от источника к получателю с определенной достоверностью в течение заданного промежутка времени.

Последовательность импульсов, составляющих кодовые слова, с выхода УЗО поступает на устройство приёма/передачи сигналов (УПС), обеспечивающее преобразование импульсов постоянного тока в модулированный сигнал и согласование выходных параметров передающей части аппаратуры с параметрами канала связи.

По способу ввода избыточности различают УЗО с постоянной избыточностью (симплексные) и УЗО с переменной избыточностью (дуплексные). В симплексных УЗО избыточность вводится заранее заданным способом независимо от состояния канала связи. В дуплексных УЗО передатчик по каналу обратной связи получает информацию о состоянии канала и при необходимости изменяет способ ввода избыточности (ее величину).

В дипломном проекте канал связи задан полудуплексным.

В симплексных УЗО повышение верности передачи может быть достигнуто тремя способами: путем многократного повторения символов, одновременной передачей одной и той же информации по нескольким каналам, применение кодов, исправляющих ошибки, применение комбинированного метода.

Метод многократного повторения основан на сравнении в месте приема всех принятых одноименных кодовых комбинаций между собой. Переданной считается та кодовая комбинация, которая принята одинаково большее число раз (критерий большинства). При работе простыми кодами исключение неопределенности достигается передачей каждой комбинации не менее трех раз. В этом случае  $n$ -элементарная кодовая комбинация считается принятой правильно, если две из трех комбинаций не будут искажены. В реальных каналах связи из-за пакетирования ошибок выигрыш, получаемый при методе повторений, снижается. Это объясняется тем, что при последовательной передаче повторяемых комбинаций один пакет ошибок с некоторой вероятностью может поразить несколько комбинаций. Для улучшения защиты от вредного влияния пакетирования ошибок необходимо использовать кодовые комбинации большей длины или разносить время повторения. В обоих случаях требуется усложнить аппаратуру (т. к. растут емкости накопителей) и увеличить время задержки при приеме информации. Таким образом, к недостаткам метода многократного повторения передаваемых комбинаций относятся:

- большая избыточность передаваемой информации, вследствие чего снижается пропускная способность УЗО;
- значительная задержка передаваемой информации;
- громоздкость решающих устройств из-за необходимости хранить большой объем принимаемой информации;
- необходимость применения специальных мер для декорреляции ошибок;

- сравнительно недостаточная защищенность метода от ошибок с малым коэффициентом группирования (неприем сообщений при искажении одних и тех же позиций в повторяемых комбинациях).

Метод передачи по нескольким каналам предусматривает передачу по трем параллельным каналам с последующим сравнением по критерию большинства. При использовании географически разнесенных каналов ошибки, возникающие в них, являются взаимно независимыми. Этот метод не имеет выше указанных недостатков первого метода, но он требует нескольких дорогих каналов связи, что в свою очередь является существенным недостатком. Так как в задании дипломного проекта предусмотрен один канал, то этот метод, очевидно, отпадает.

Применение мощных помехоустойчивых кодов, исправляющих пакеты ошибок, является наиболее перспективным направлением. К достоинству этого метода можно отнести уменьшение громоздкости устройств для хранения принимаемой информации, уменьшение времени передачи по сравнению с первым методом, а, следовательно, увеличение пропускной способности канала, более оптимальное использование избыточности. К недостаткам относятся: сложность кодирующих устройств, и, особенно декодирующих, а, следовательно, трудности в их настройке и ремонте.

Комбинированный метод – сочетание любых рассмотренных методов. Сочетание первого и третьего методов позволяет осуществить защиту как от больших пакетов ошибок (с помощью первого метода, т. к. повторяемые комбинации имеют большую длину), так и от небольших пакетов (с помощью третьего метода). Недостатки комбинированного метода по сравнению с третьим заключаются в ухудшении ВВХ (увеличение времени доведения информации до потребителя), необходимости увеличения объема оперативной памяти для хранения обрабатываемой информации.

К дуплексной группе УЗО относятся устройства, в которых повышение верности передаваемой информации достигается за счет введения обратной

связи (ОС). Они в свою очередь делятся на системы с решающей (РОС), информационной (ИОС) и комбинированной (КОС) обратной связью.

В системах с РОС передаваемые данные кодируются избыточными кодами, позволяющими обнаруживать одиночные ошибки или группы ошибок. В данных системах приёмник на основании, поступившей последовательности, определяет решение, о необходимости повторения блока информации. При обнаружении ошибки в блоке, приёмник стирает принятое сообщение и отправляет запрос на повторное получение.

В УЗО с ИОС двоичная последовательность фиксируется приёмником, затем по каналу ОС передаётся вся или, как укороченная кодовая комбинация на передающую сторону. Передатчиком анализируется, полученная по каналу ОС, комбинация и по результатам анализа принимается решение о передаче следующего блока или о повторении предыдущего. Данное решение отправляется в приёмник и на его основании информация или стирается, или передаётся получателю.

УЗО с КОС решение о необходимости повторной передачи может приниматься, как приёмником, так и передатчиком.

Одной из главных задач проектирования УЗО является выбор способа защиты от ошибок, который при минимальных затратах обеспечит выполнение поставленных требований. Под затратами подразумевается не только стоимость аппаратуры, но и необходимые полоса частот (требуемое число каналов связи), время на передачу сообщения, а также стоимость обслуживания устройства в процессе эксплуатации.

В данном проекте выбирается метод построения УЗО, основанный на применении каскадного кода исправляющего пакет ошибок до шестидесяти символов. Это позволяет, в соответствии с ниже приведенными расчетами обеспечить заданную в ТЗ вероятность доведения информации.

## 1.2 Выбор и обоснование помехоустойчивого кода

История кодирования, обеспечивающего возможность сохранности информации при её передачи, началась с 1948 года после публикации статьи К. Шеннона. Шеннон показал, что с каждым каналом связано измеряемое в битах в секунду величина, которая называется пропускная способность канала. Если в системе связи скорость передачи информации меньше пропускной способности канала, то используя коды, исправляющие и обнаруживающие ошибки, то для данного канала можно построить такую систему связи, что вероятность ошибки на выходе будет мала. Поэтому построение слишком хороших каналов связи является расточительством и экономически выгоднее использовать кодирование.

Современное состояние проблемы помехоустойчивого кодирования характеризуется некоторым разрывом между теоретическими результатами и их практическим использованием. Если в теоретическом плане получен целый ряд весьма эффективных результатов, то на практике применяется лишь небольшое число помехоустойчивых кодов, да и то, как правило, наиболее простых и мало эффективных.

Обнаруживающие коды позволяют выявить ошибку, однако не представляют возможностей её исправления. Самый быстрый способ идентификации заключается в добавлении дополнительного бита для проверки на чётность или нечётность. Но такой способ позволяет лишь идентифицировать лишь одиночную ошибку. Так же внесение избыточности в код за счёт увеличения длины комбинации, приведёт к снижению скорости передачи.

Такое положение, прежде всего, объясняется тем, что основная масса работ в области помехоустойчивого кодирования направлена на поиск кодов, корректирующих независимые ошибки и здесь достигнуты существенные результаты. Однако в большинстве реальных каналов условие независимости ошибок не выполняется и искажения появляются в виде последовательности пакетов ошибок достаточно большой длины с высокой плотностью в пределах

пакета. В таких условиях коды, корректирующие независимые ошибки, малоэффективны, поскольку на интервалах между ошибками избыточность, содержащаяся в коде, используется мало, а в пределах пакета ошибок для коррекции требуется высокая избыточность. Положение может быть улучшено при использовании кодов достаточно большей длины, на много превышающей среднюю длину пакета. Однако, хотя теоретически такая возможность имеется, ее практическая реализация требует чрезмерного усложнения декодирующих устройств, так как сложность декодирования быстро возрастает с увеличением длины кодовой комбинации и числа корректируемых ошибок.

В настоящее время создано большое количество помехоустойчивых кодов, которые отличаются различными свойствами. Классификация помехоустойчивых кодов представлена на рисунке 2.



Рисунок 2 - Классификация помехоустойчивых кодов

Важный подкласс непрерывных кодов образуют свёрточные коды. Свёрточные коды относятся к классу непрерывных кодов. В основе построения этих кодов лежит рекуррентный принцип кодирования и декодирования, когда

на вход кодера поступает непрерывная последовательность информационных символов источника, а с выхода кодера снимается также непрерывная последовательность символов, являющихся функцией входных символов и структуры кодера. На вход декодера поступает непрерывная последовательность символов из канала связи (возможно, искаженная ошибками), а на выходе восстанавливается (возможно, с ошибками, но, как правило, меньшими чем канальные) последовательность информационных символов.

Из-за длины кода и увеличения значений расстояния и помехоустойчивости на практике свёрточные коды сложно реализуются. Составные коды в таких случаях дают некоторую компромиссность решения задач с кодированием. К наиболее распространённым относят каскадные коды.

Каскадные коды – это мощные коды, при построении которых было предложено использовать каскадное включение более простых кодовых конструкций, декодирование которых осуществляется гораздо проще, чем сложных кодов с большой исправляющей способностью. Каскадное кодирование подразумевает подачу с первого кодирующего устройства на вход следующего, таким образом комбинация простых кодеров является мощной кодирующей структурой. Последнее кодирующее устройство, стоящее перед каналом, называется внутренним кодирующим устройством. При декодировании можно использовать аналогичную структуру декодеров. Принцип каскадного кодирования представлен на рисунке 3.

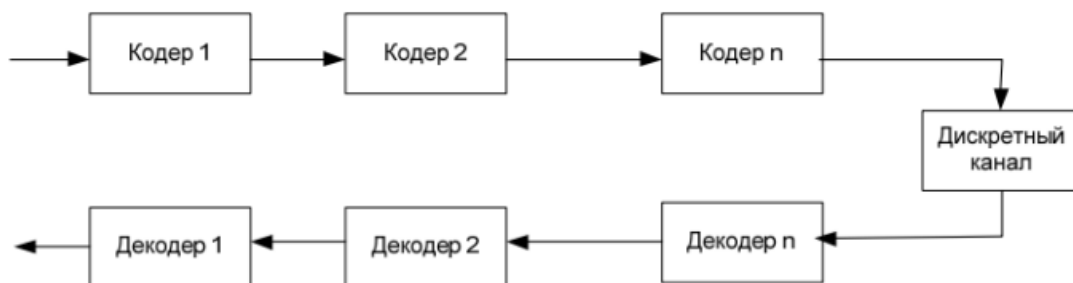


Рисунок 3 - Принцип каскадного кодирования и декодирования

Достоинством каскадных кодов является относительно низкая сложность кодирующих и декодирующих устройств. Каскадные коды позволяют реализовать достаточно большое кодовое расстояние, поэтому их применение в каналах с помехами эффективно.

К этому же направлению работ относится и метод построения каскадных кодов – кодов достаточно большой длины, обладающих двумя основными особенностями:

- вероятность ложного декодирования в каналах без памяти экспоненциально убывает с ростом количества проверочных символов в кодовой комбинации;

- сложность декодирования не возрастает по экспоненциальному закону с увеличением длины кодовой комбинации, т. к. процедура декодирования остается практически реализуемой при достаточно больших длинах кодовых комбинаций.

Учитывая все выше сказанное, в данном проекте выбирается каскадный код, исправляющий как одиночные ошибки, так и пакеты ошибок, и обеспечивающий относительно несложную схему декодирования.

### 1.3 Выбор и обоснование каскадного кода

Метод каскадного кода, разработанный Форни, - метод комбинирования двух кодов для образования большого кода, который в некоторой степени напоминает умножение кодов [1]. На рисунке 4 изображена общая схема каскадного декодирования/кодирования.

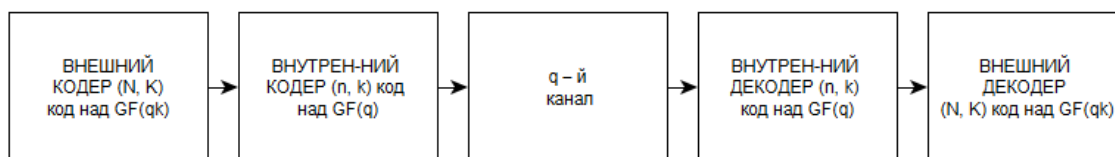


Рисунок 4- Канал связи с использованием каскадного кода

Первый код называется внутренним и представляет собой  $(n, k)$  код с символами из поля Галуа  $GF(g)$ . Второй код называется внешним и является  $(N, K)$  кодом с символами из поля Галуа  $GF(gk)$ . Кодирование осуществляется следующим образом:

- блок из  $l \cdot k$  информационных символов делится на  $k$  подблоков по  $l$  символов;
- каждый подблок из  $l$  символов рассматривается как символ над полем Галуа  $GF(gk)$ . Затем  $k$  подблоков кодируются в кодовые слова внешнего кода, составленные из  $N$  подблоков;
- каждый из  $N$  подблоков, содержащий  $l$  символов, кодируется как  $l$  информационных символов кодового слова длины  $n$  во внутреннем коде.

Таким образом, совокупность  $N$  кодовых слов внутреннего  $(n, k)$  кода является кодовым словом каскадного  $(Nk, Kk)$  кода.

Способ декодирования очевиден. Сначала происходит декодирование внутреннего кода, в результате которого  $N$  слов внутреннего кода сводятся к  $N$  подблокам по  $l$  символов. Затем эти  $N$  подблоков с помощью метода декодирования для внешнего кода декодируются в  $K$  подблоков.

Поэтапная реализация процедуры декодирования позволяет рационально распределить функции между внутренним и внешним декодером, реализуя исправление ошибок при минимальной сложности их построения, т.к. внутренний декодер обнаруживает и частично исправляет ошибки, а внешний декодер исправляет ошибки и стирания.

Но выше представленный метод реализации на схеме, состоящей из двух кодеров или декодеров, имеет один важный недостаток. Это – явление размножения ошибок в системе декодирования. Данное явление связано с ошибками, которые поступают на вход декодера. Данные ошибки в свою очередь на выходе, могут породить такую группу ошибок, что следующее декодирующее устройство с ней не справится, т.к. используются, как правило, довольно простые коды.

Возможны следующие режимы использования каскадных кодов:

- исправление ошибок внутренним и внешним кодами;
- обнаружение ошибок внутренним и исправление стираний внешним кодами;
- обнаружение ошибок внутренним и исправление стираний и ошибок внешним кодами;
- частичное исправление и обнаружение ошибок внутренним и исправление стираний внешним кодами;
- частичное исправление и обнаружение ошибок внутренним и исправление стираний и ошибок внешним кодами.

В рамках дипломного проекта для внешнего кодирования применяется код Рида-Соломона с заданной длиной  $N$  блоков в байтах. Применение данного кода позволяет декодеру исправлять долговременные ошибки. Каждый блок РС кода содержит  $K$  информационных байтов и  $r' = n' - k'$  проверочных. Скорость РС кода определяется как  $R1 = k'/n'$ .

На втором этапе применяется внешнее перемежение (интерливинг) для всех бит внутри РС блока.

На третьем этапе выполняется внутреннее кодирование. В качестве внутреннего кода был выбран – код Хемминга.

Формат блока каскадного кода представлен на рисунке 5 и на плакате “Форматы информационных кадров кода Рида-Соломона, кода Хемминга, каскадного кода”, лист 1, где, в качестве примера:

$n$  – общее количество бит комбинации внутреннего кода Хемминга;

$k$  – количество информационных бит комбинации внутреннего кода Хемминга;

$r$  – количество проверочных бит комбинации внутреннего кода Хемминга;

$n'$  – общее количество символов комбинации кода РС;

$k'$  – количество информационных символов комбинации кода РС;

$r'$  – количество проверочных символов комбинации кода РС.

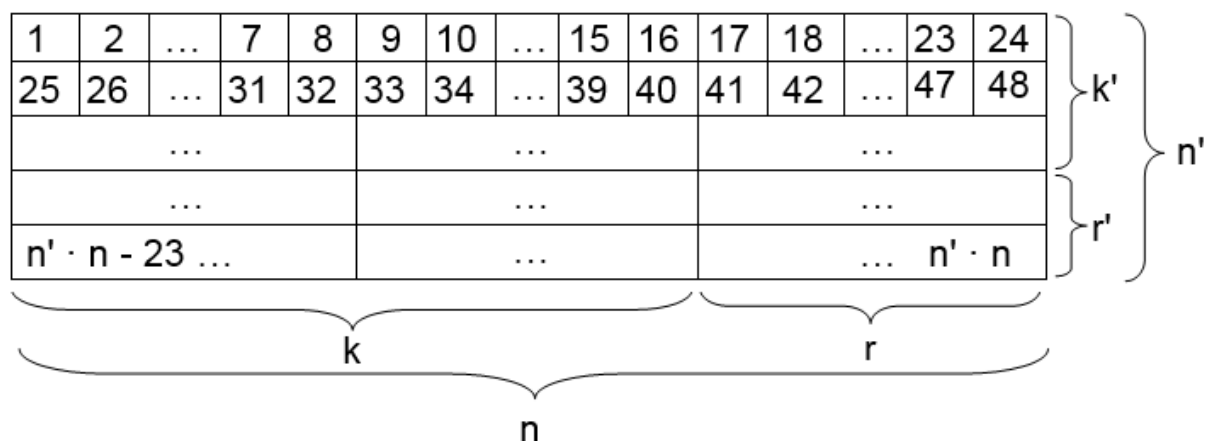


Рисунок 5 - Формат блока каскадного кода

#### 1.4 Расчёт длины кодовой комбинации внутреннего кода

Судя по многочисленным экспериментам, наиболее эффективным является передача сообщений комбинациями, состоящими из  $10 \div 60$  символов.

Конкретный выбор длины кодовой комбинации зависит от реальных условий, от характеристик канала связи.

Было показано [2], что поток ошибок в канале может быть описан с достаточной степенью точности для инженерной практики двумя параметрами:

- вероятностью ошибочного приема двоичного символа,  $P_0$ ;
- показателем группирования ошибок,  $\alpha_0$ .

На основе этих параметров разработаны инженерные методы расчета эффективности групповых кодов.

Заданный канал связи имеет следующие характеристики:

- вероятность ошибочного приема двоичного символа,  $P_0$ , равную  $5 \cdot 10^{-2}$ ;
- показатель группирования ошибок,  $\alpha_0$ , равный нулю, т. е. ошибки распределены равномерно, т. к. канал задан биномиальным.

Для сравнения берутся три кодовые комбинации:

- (12,8) – кодовое расстояние  $d_1 = 3$ ;
- (24,16) – кодовое расстояние  $d_2 = 4$ ;

- (48,32) – кодовое расстояние  $d_3 = 4$ .

Для анализируемых кодовых комбинаций определяется наивысшая кратность гарантированно исправляемых ошибок:

$$t \leq \frac{d_{\min} - 1}{2}. \quad (1.1)$$

Согласно формуле (1.1) для выше указанных кодовых комбинаций наивысшая кратность гарантированно исправляемых ошибок будет соответственно равна:

$$t_1 \leq \frac{3-1}{2} = 1;$$

$$t_2 \leq \frac{4-1}{2} = 1,5, \text{ округляется до } t_2 = 1;$$

$$t_3 \leq \frac{4-1}{2} = 1,5, \text{ округляется до } t_3 = 1.$$

Далее подсчитывается вероятность правильного приема одной кодовой комбинации длины  $n$ :

$$P_{\text{прм}i} = C_n^i \cdot P_0^i \cdot (1 - P_0)^{n-i}, \quad (1.2)$$

где  $P_0 = 5 \cdot 10^{-2}$  – вероятность ошибочного приема двоичного символа;

$C_n^i$  – число комбинаций длины  $n$ , содержащих одну ошибку:

$$C_n^i = \frac{n!}{(n-i)! \cdot i!}. \quad (1.3)$$

Таким образом, согласно формуле (1.2), вероятности приема одной кодовой комбинации соответственно будут равны:

$$P_{\text{прм}(n=12)} = \frac{12!}{(12-1)! \cdot 1!} \cdot 5 \cdot 10^{-2} (1 - 5 \cdot 10^{-2})^{11} = 0,6 \cdot 0,95^{11} = 0,3412;$$

$$P_{\text{прм}(n=24)} = 0,95^{23} \cdot 1,2 = 0,3688;$$

$$P_{\text{прм}(n=48)} = 0,95^{47} \cdot 2,4 = 0,2153.$$

Но при кодировании информационный кадр также может быть принят и безошибочно, т. е.  $i$  равно нулю. Тогда, согласно формуле (1.2) вероятности приема одной кодовой комбинации соответственно будут равны:

$$P_{нрм (n=12)} = 0,5404;$$

$$P_{нрм (n=24)} = 0,2919;$$

$$P_{нрм (n=48)} = 0,0852.$$

Тогда вероятность правильного приема одной кодовой комбинации длины  $n$  будет соответственно равна:

$$P_{нрм.(n=12)} = 0,8816;$$

$$P_{нрм.(n=24)} = 0,6607;$$

$$P_{нрм.(n=48)} = 0,3005.$$

Для дальнейших расчетов выбирается код (24,16), т.к. код (12,8) хотя и имеет большую вероятность правильного приема всего информационного кадра, но не обладает обнаруживающей способностью, т. к. имеет минимальное кодовое расстояние,  $d_{\min}$ , равное трем.

Код (24,16) состоит из шестнадцати информационных и восьми проверочных символов и способен исправлять одну ошибку в двадцати четырех разрядной комбинации.

Получаемая при кодировании кодом (24,16) вероятность правильного приема всего информационного кадра ( $P_{н.нрм}$  равна 0,6607) не удовлетворяет условиям ТЗ ( $P_{н.нрм}$  равна 0,95).

Для повышения вероятности правильного приема применяется метод трехкратного переприема и кодирование внешним кодом. В качестве внешнего кода используется код Рида – Соломона [3].

Применяемый внешний код позволяет исправлять до шестнадцати стираний с помощью шестнадцати проверочных символов.

Далее определяется вероятность правильного приема информационного кадра при использовании кода Рида – Соломона (101,85).

$$P_{\text{пр.м.к.б.}} = \sum_{i=0}^{d_2} C_{n_2}^i \cdot (1 - P_{\text{пр.м.с.}})^i \cdot P_{\text{пр.м.с.}}^{n_2-i}, \quad (1.4)$$

где  $i$  – количество не принятых (стертых) блоков внутреннего кода;

$d_2$  – кодовое расстояние используемого внешнего кода (количество блоков, при которых пакет (кадр) будет принят);

$n_2$  – количество символов (блоков внутреннего кода) в передаваемом блоке внешнего кода;

$P_{\text{пр.м.с.}}$  – вероятность приема одного блока внутреннего кода.

Исходя из формулы (1.4) получаем, что  $P_{\text{пр.м.к.б.}}$  равна 0,98, т. е. полученная вероятность доведения информационного кадра не только удовлетворяет требованиям ТЗ ( $P_{\text{н.прм}}$  равна 0,95), но и превышает их (0,95 меньше 0,98).

### 1.5 Расчет корректирующих способностей внутреннего кода

В качестве внутреннего кода в разработанном УЗО применяется циклический код, исправляющий одну ошибку. Корректирующая способность циклического кода полностью определяется видом образующего многочлена  $P(x)$ , поэтому правильный выбор образующего многочлена является основной задачей, решаемой при построении циклического кода.

Для сравнения корректирующих способностей различных вариантов внутреннего кода и выбора наилучшего варианта сравниваются одиннадцать многочленов восьмой степени, приведенные в таблице 1.

Таблица 1 – Многочлены восьмой степени

| Многочлен восьмой степени                               | Обозначение |
|---------------------------------------------------------|-------------|
| $(x^7 + x + 1) \cdot (x + 1)$                           | 203         |
| $(x^7 + x^6 + x^5 + x^3 + x^2 + 1) \cdot (x + 1)$       | 345         |
| $(x^7 + x^6 + x^5 + x^3 + x^2 + x + 1) \cdot (x + 1)$   | 357         |
| $(x^7 + x^5 + x^3 + x + 1) \cdot (x + 1)$               | 253         |
| $(x^7 + x^6 + x^3 + 1) \cdot (x + 1)$                   | 217         |
| $(x^7 + x^6 + x^3 + x + 1) \cdot (x + 1)$               | 313         |
| $(x^7 + x^6 + x^5 + x^4 + x^3 + x^2 + x) \cdot (x + 1)$ | 375         |
| $(x^7 + x^6 + x^3 + 1) \cdot (x + 1)$                   | 271         |
| $(x^7 + x^3 + 1) \cdot (x + 1)$                         | 211         |
| $(x^8 + x^7 + x^6 + x^5 + x^2 + x + 1)$                 | 747         |
| $(x^8 + x^7 + x^6 + x + 1)$                             | 703         |

Все эти многочлены обеспечивают кодовое расстояние, равное четырем. Первые девять многочленов образуются умножением неприводимого многочлена седьмой степени на  $(x + 1)$ .

Два последних – неприводимые многочлены восьмой степени. Неприводимый многочлен – это многочлен, делящийся без остатка только на себя и на единицу [4].

Корректирующая способность внутреннего кода обуславливается числом обнаруженных ошибок различной степени, поэтому проводится расчет и анализ обнаруженных ошибок различной степени  $N$  ( $N = 3, 4, 5, 6, 7, 8$ ) для всех сравниваемых многочленов. В результате обработки на ЭВМ было определено число комбинаций, показанных в таблице 2, которые приводят к обнаруженным ошибкам степени  $N$  для всех многочленов.

Таблица 2 – Число комбинаций приводящих к необнаруженным ошибкам

| Число комбинаций с необнаруженными ошибками кратностью | Многочлен |      |      |      |      |      |      |      |      |      |      |
|--------------------------------------------------------|-----------|------|------|------|------|------|------|------|------|------|------|
|                                                        | 203       | 345  | 253  | 357  | 217  | 313  | 211  | 375  | 271  | 747  | 703  |
| 3                                                      | 0         | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    |
| 4                                                      | 90        | 70   | 92   | 80   | 78   | 91   | 87   | 80   | 70   | 37   | 37   |
| 5                                                      | 0         | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 192  | 202  |
| 6                                                      | 1046      | 1117 | 997  | 1061 | 1073 | 1009 | 1018 | 1054 | 1121 | 540  | 524  |
| 7                                                      | 0         | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 1360 | 1309 |
| 8                                                      | 5690      | 5622 | 5882 | 5704 | 5710 | 5786 | 5858 | 5786 | 5610 | 2822 | 2910 |

Общее число комбинаций с ошибками кратности  $N$  в  $n$ -элементной последовательности определяется формулой:

$$C_n^N = \frac{n!}{(n-N)! \cdot N!} \cdot \quad (1.5)$$

Общее число комбинаций с ошибками кратности  $N$  ( $N = 4, 5, 6, 7, 8$ ). Тогда по формуле (1.5) получаем:

$$C_{24}^4 = \frac{24!}{(24-4)! \cdot 4!} = \frac{21 \cdot 22 \cdot 23 \cdot 24}{1 \cdot 2 \cdot 3 \cdot 4} = 10626;$$

$$C_{24}^5 = \frac{24!}{(24-5)! \cdot 5!} = \frac{20 \cdot 21 \cdot 22 \cdot 23 \cdot 24}{1 \cdot 2 \cdot 3 \cdot 4 \cdot 5} = 42504;$$

$$C_{24}^6 = \frac{24!}{(24-6)! \cdot 6!} = 134595;$$

$$C_{24}^7 = \frac{24!}{(24-7)! \cdot 7!} = 346107;$$

$$C_{24}^8 = \frac{24!}{(24-8)! \cdot 8!} = 735471.$$

Вероятность появления необнаруженной ошибки определяется следующей формулой:

$$P_{\text{необн.ош.}}^N = \frac{C_{\text{необнар.общ.кр.}}}{C_n^N} = \frac{C_{\text{н.о.}}^N}{C_n^N} \cdot \quad (1.6)$$

Вероятность появления N-кратной ошибки в кодовой комбинации длины  $n$  равна:

$$P_n^N = C_n^N \cdot P_0^N \cdot (1 - P_0)^{n-N}, \quad (1.7)$$

где  $P_0 = 5 \cdot 10^{-2}$  – вероятность искажения одного бита в канале.

Вероятность ложного исправления кода определяется следующим образом:

$$P_{л.и.} = \sum_{N=4}^8 P_{необн.ош.}^N \cdot P_n^N. \quad (1.8)$$

$$P_{л.и.} = \sum_{N=4}^8 \frac{C_{н.о.}^N}{C_n^N} \cdot C_n^N \cdot P_0^N \cdot (1 - P_0)^{n-N} =$$

$$= \sum_{N=4}^8 C_{н.о.}^N \cdot P_0^N \cdot (1 - P_0)^{n-N}.$$

Таким образом, корректирующая способность кода определяется суммарной вероятностью ложного исправления для ошибок кратностью  $N = 4, 5, 6, 7, 8$ .

Для кода 203:

$$P_4 = 90 \cdot 5 \cdot 10^{-2 \cdot 4} \cdot (1 - 5 \cdot 10^{-2})^{24-4} = 90 \cdot 5 \cdot 10^{-8} \cdot 0,95^{20} = 1,613 \cdot 10^{-6};$$

$$P_6 = 1046 \cdot 5 \cdot 10^{-12} \cdot 0,95^{18} = 2,077 \cdot 10^{-9};$$

$$P_8 = 5690 \cdot 5 \cdot 10^{-16} \cdot 0,95^{16} = 1,252 \cdot 10^{-12};$$

$$P_{л. и.} = 1,613 \cdot 10^{-6} + 2,077 \cdot 10^{-9} + 1,252 \cdot 10^{-12} = 1,615 \cdot 10^{-6}.$$

Для кода 345:

$$P_{л. и.} = 1,254 \cdot 10^{-6} + 2,218 \cdot 10^{-9} + 1,237 \cdot 10^{-12} = 1,256 \cdot 10^{-6}.$$

Для кода 253:

$$P_{л.и.} = 1,649 \cdot 10^{-6} + 1,98 \cdot 10^{-9} + 1,294 \cdot 10^{-12} = 1,65 \cdot 10^{-6}.$$

Для кода 357:

$$P_{л.и.} = 1,434 \cdot 10^{-6} + 2,105 \cdot 10^{-9} + 1,255 \cdot 10^{-12} = 1,436 \cdot 10^{-6}.$$

Для кода 217:

$$P_{л.и.} = 1,398 \cdot 10^{-6} + 2,131 \cdot 10^{-9} + 1,256 \cdot 10^{-12} = 1,4 \cdot 10^{-6}.$$

Для кода 313:

$$P_{л.и.} = 1,631 \cdot 10^{-6} + 2,003 \cdot 10^{-9} + 1,273 \cdot 10^{-12} = 1,633 \cdot 10^{-6}.$$

Для кода 211:

$$P_{л.и.} = 1,559 \cdot 10^{-6} + 2,021 \cdot 10^{-9} + 1,289 \cdot 10^{-12} = 1,561 \cdot 10^{-6}.$$

Для кода 375:

$$P_{л.и.} = 1,273 \cdot 10^{-6} + 2,093 \cdot 10^{-9} + 1,273 \cdot 10^{-12} = 1,275 \cdot 10^{-6}.$$

Для кода 271:

$$P_{л.и.} = 1,254 \cdot 10^{-6} + 2,226 \cdot 10^{-9} + 1,234 \cdot 10^{-12} = 1,256 \cdot 10^{-6}.$$

Для кода 747:

$$P_{л.и.} = 6,631 \cdot 10^{-7} + 3,623 \cdot 10^{-8} + 1,072 \cdot 10^{-9} + 2,843 \cdot 10^{-11} + \\ + 6,21 \cdot 10^{-13} = 6,993 \cdot 10^{-7}.$$

Для кода 703:

$$P_{л.и.} = 6,631 \cdot 10^{-7} + 3,811 \cdot 10^{-8} + 1,041 \cdot 10^{-9} + 2,74 \cdot 10^{-11} + \\ + 6,404 \cdot 10^{-13} = 7,023 \cdot 10^{-7}.$$

Таким образом, результаты расчета показывают, что наилучшей корректирующей способностью обладает код 747, но для проектируемого устройства в соответствии со специальными протоколами взаимодействия, принятыми в ВС РФ выбирается многочлен  $217x^8 + x^7 + x^4 + 1$ .

## 1.6 Описание выбранного внутреннего кода

В качестве внутреннего кода выбран укороченный циклический код Хемминга. Множество кодовых слов указанного кода представляется совокупностью многочленов степени  $(n-1)$  и менее, делящихся на некоторый многочлен  $g(x)$  степени  $r = n-k$ , являющийся сомножителем двучлена  $x^n+1$ . Многочлен  $g(x)$  называется порождающим. В соответствии с протоколами,

принятыми в СОД ЕСУ ТЗ в качестве порождающего многочлена взят многочлен  $x^8 + x^7 + x^4 + 1$ .

Укороченный циклический код Хемминга удобно задавать с помощью, так называемой порождающей матрицы. Порождающая матрица представлена на рисунке

$$G = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1k} & b_{1k+1} & b_{1k+2} & \dots & b_{1n} \\ a_{21} & a_{22} & \dots & a_{2k} & b_{2k+1} & b_{2k+2} & \dots & b_{2n} \\ \cdot & \cdot & & \cdot & \cdot & \cdot & & \\ \cdot & \cdot & & \cdot & \cdot & \cdot & & \\ \cdot & \cdot & & \cdot & \cdot & \cdot & & \\ a_{k1} & a_{k2} & \dots & a_{kk} & b_{kk+1} & b_{kk+2} & \dots & b_{kn} \end{bmatrix}$$

**I**
**II**

Рисунок 6- Порождающая матрица

В нашем случае порождающая или образующая матрица имеет вид, показанный на рисунке 7.

Эту матрицу можно рассматривать как совокупность двух матриц: информационной I, составленной из элементов  $a_{ij}$  и проверочной II, составленной из элементов  $b_{ij}$ .

Матрица I – это единичная диагональная матрица, у которой  $k$  строк, и  $k$  столбцов.

| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 |
|---|---|---|---|---|---|---|---|---|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 1 |   |   |   |   |   |   |   |   |    |    |    |    |    |    |    | 0  | 0  | 0  | 0  | 1  | 0  | 1  | 1  |
|   | 1 |   |   |   |   |   |   |   |    |    |    |    |    |    |    | 1  | 1  | 0  | 0  | 1  | 1  | 0  | 1  |
|   |   | 1 |   |   |   |   |   |   |    |    |    |    |    |    |    | 1  | 0  | 1  | 0  | 1  | 1  | 1  | 0  |
|   |   |   | 1 |   |   |   |   |   |    |    |    |    |    |    |    | 0  | 1  | 0  | 1  | 0  | 1  | 1  | 1  |
|   |   |   |   | 1 |   |   |   |   |    |    |    |    |    |    |    | 1  | 1  | 1  | 0  | 0  | 0  | 1  | 1  |
|   |   |   |   |   | 1 |   |   |   |    |    |    |    |    |    |    | 1  | 0  | 1  | 1  | 1  | 0  | 0  | 1  |
|   |   |   |   |   |   | 1 |   |   |    |    |    |    |    |    |    | 1  | 0  | 0  | 1  | 0  | 1  | 0  | 0  |
|   |   |   |   |   |   |   | 1 |   |    |    |    |    |    |    |    | 0  | 1  | 0  | 0  | 1  | 0  | 1  | 0  |
|   |   |   |   |   |   |   |   | 1 |    |    |    |    |    |    |    | 0  | 0  | 1  | 0  | 0  | 1  | 0  | 1  |
|   |   |   |   |   |   |   |   |   | 1  |    |    |    |    |    |    | 1  | 1  | 0  | 1  | 1  | 0  | 1  | 0  |
|   |   |   |   |   |   |   |   |   |    | 1  |    |    |    |    |    | 0  | 1  | 1  | 0  | 1  | 1  | 0  | 1  |
|   |   |   |   |   |   |   |   |   |    |    | 1  |    |    |    |    | 0  | 1  | 1  | 1  | 1  | 1  | 1  | 0  |
|   |   |   |   |   |   |   |   |   |    |    |    | 1  |    |    |    | 0  | 1  | 1  | 1  | 1  | 1  | 1  | 1  |
|   |   |   |   |   |   |   |   |   |    |    |    |    | 1  |    |    | 1  | 1  | 1  | 1  | 0  | 1  | 1  | 1  |
|   |   |   |   |   |   |   |   |   |    |    |    |    |    | 1  |    | 1  | 0  | 1  | 1  | 0  | 0  | 1  | 1  |
|   |   |   |   |   |   |   |   |   |    |    |    |    |    |    | 1  | 1  | 0  | 0  | 1  | 0  | 0  | 0  | 1  |

Рисунок 7 - Порождающая матрица кода Хемминга (24,16)

Матрица  $I \ I$  – это проверочная матрица, у которой  $r$  ( $r=n-k$ ) столбцов и  $k$  строк. При выборе матрицы  $\Pi$  исходят из того, что число единиц в каждой её строке должно быть не менее двух. В нашем случае проверочная матрица имеет вид показанный на рисунке 8.

| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 |
|---|---|---|---|---|---|---|---|---|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 1  | 0  | 1  | 0  | 1  | 1  | 1  | 1  |    |    |    |    |    |    |    |
| 0 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 1  | 1  | 1  | 1  | 1  | 1  | 0  |    | 1  |    |    |    |    |    |    |
| 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 0  | 1  | 1  | 1  | 1  | 1  | 0  |    |    | 1  |    |    |    |    |    |
| 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 1  | 0  | 1  | 1  | 1  | 1  | 1  |    |    |    | 1  |    |    |    |    |
| 1 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0  | 1  | 1  | 1  | 1  | 0  | 0  |    |    |    |    | 1  |    |    |    |
| 0 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 1  | 0  | 1  | 1  | 1  | 1  | 0  |    |    |    |    |    | 1  |    |    |
| 1 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 1  | 0  | 1  | 1  | 1  | 1  | 0  |    |    |    |    |    |    | 1  |    |
| 1 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 0  | 1  | 0  | 1  | 1  | 1  | 1  |    |    |    |    |    |    |    | 1  |

Рисунок 8 - Проверочная матрица кода Хемминга (24,16)

В этом и заключается задание укороченного циклического кода порождающей и проверочной матрицами.

Комбинация информационной части укороченного циклического кода Хемминга, в соответствии со специальными протоколами взаимодействия, принятыми в ВС РФ, должна содержать символы двух кодов РС. Битовое представление комбинации внутреннего кода приведено на рисунке 9 и на плакате “Форматы информационных кадров кода Рида-Соломона, кода Хемминга, каскадного кода”, лист 1.



Рисунок 9 – Битовое представление комбинации внутреннего кода

Хемминга

Выдача комбинации укороченного циклического кода Хемминга в канал связи осуществляется последовательно с первого информационного бита. В качестве примера битовое представление и порядок выдачи в канал связи комбинации внутреннего кода приведено на рисунке 10 и на плакате “Форматы информационных кадров кода Рида-Соломона, кода Хемминга, каскадного кода”, лист 2.

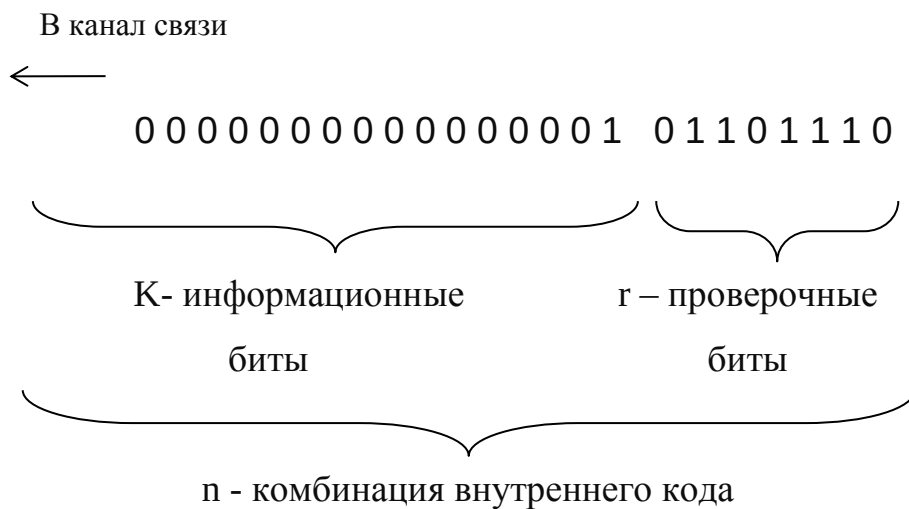


Рисунок 10 – Пример битового представления и порядок выдачи в канал связи комбинации внутреннего кода

## 1.7 Описание внешнего кода

Коды Рида-Соломона или РС-коды относятся к недвоичным циклическим кодам, т.е. кодам, символы которых взяты из конечного поля, содержащего  $q$  элементов и обозначаемого  $GF(q)$ , где  $q$  - степень некоторого простого числа.

Код, образованный с помощью  $q$  - ичных кодовых комбинаций, называют  $q$  - ичным кодом. Такие коды, как и двоичные коды, могут быть простыми и корректирующими. Кодовые комбинации  $q$  - ичного кода могут быть представлены в виде полиномов с  $q$  - ичными коэффициентами. При этом сами коэффициенты являются полиномами с двоичными коэффициентами.

РС-коды - это важное и широко используемое на практике подмножество кодов БЧХ, у которых мультипликативный порядок алфавита символов кодовой комбинации делится на длину кода.

Интерес к этому классу линейных кодов обусловлен несколькими причинами. Прежде всего РС-коды являются максимальными, т.е. при заданной длине  $N$  и размерности  $K$  они обладают наибольшим кодовым расстоянием  $D$ , равным  $N$  минус  $K$  плюс один. Кроме того, эти коды имеют эффективный алгебраический метод декодирования.

Внешний код представляет собой два блока систематического укороченного циклического кода РС  $(n', k', 17)$ , где:

$n'$  - общее количество символов комбинации кода РС;

$k'$  - количество информационных символов комбинации кода РС;

число 17 - минимальное кодовое расстояние кода РС,  $d_{min}$ .

Величины  $n'$  и  $k'$  в соответствии со специальными протоколами взаимодействия, принятыми в ВС РФ, должны иметь четыре фиксированных значения, которые приведены в таблице 3 и на плакате "Форматы информационных кадров кода Рида-Соломона, кода Хемминга, каскадного кода", лист 2.

Таблица 3 – Фиксированные значения  $n'$  и  $k'$

| Наименование<br>кадра данных | Длина $n'$ ,<br>символ | Длина $k'$ ,<br>символ |
|------------------------------|------------------------|------------------------|
| Кадр 2                       | 49                     | 33                     |
| Кадр 3                       | 101                    | 85                     |
| Кадр 4                       | 157                    | 141                    |
| Кадр 5                       | 241                    | 225                    |

Комбинация блока внешнего кода формируется перемежением символов двух блоков кода РС.

Формат блока внешнего кода представлен на рисунке 11 и на плакате “Форматы информационных кадров кода Рида-Соломона, кода Хемминга, каскадного кода”, лист 2.

|                           |     |                          |                          |     |                      |   |    |
|---------------------------|-----|--------------------------|--------------------------|-----|----------------------|---|----|
| 1                         | ... | 8                        | 9                        | ... | 16                   | } | n' |
| 17                        | ... | 24                       | 25                       | ... | 32                   |   |    |
| ...                       |     |                          | ...                      |     |                      |   |    |
| $2 \cdot k' \cdot 8 - 15$ | ... | $2 \cdot k' \cdot 8 - 8$ | $2 \cdot k' \cdot 8 - 7$ | ... | $2 \cdot k' \cdot 8$ |   |    |
| 1                         |     | 8                        | 9                        |     | 16                   |   |    |
| ...                       |     |                          | ...                      |     |                      |   |    |
| $2 \cdot r' \cdot 8 - 15$ |     | $2 \cdot r' \cdot 8 - 8$ | $2 \cdot r' \cdot 8 - 7$ |     | $2 \cdot r' \cdot 8$ |   |    |
| 1-й блок кода РС          |     |                          | 2-й блок кода РС         |     |                      |   |    |

Рисунок 11 - Формат блока внешнего кода

### 1.8 Система циклового фазирования

В соответствии со специальными протоколами взаимодействия СОД ЕСУ ТЗ выбрана система циклового фазирования на основе рекуррентного способа формирования. С образующим многочленом  $x^8 + x^7 + x^4 + 1$  длиной 217 бит генерируется программным способом и хранится в ОЗУ процессора.

Структурная схема генератора фазирующей посылки представлена на рисунке 12.



Рисунок 12 – Структурная схема генератора ФП

В соответствии со специальными протоколами взаимодействия СОД ЕСУ ТЗ начальное заполнение генератора ФП определяются типом, форматом и длиной кадра данных. Анализируемая комбинация является последними 12 из 256 бит последовательности ФП. Комбинация начального заполнения и анализируемая комбинация приведены в таблице 4.

Таблица 4 - Комбинации начального заполнения и анализируемые комбинации

| Наименование кадра данных | Комбинация начального заполнения | Анализируемая комбинация |
|---------------------------|----------------------------------|--------------------------|
| Кадр 1                    | 100011100001                     | 101000010000             |
| Кадр 2                    | 011110001110                     | 100000101011             |
| Кадр 3                    | 011000001001                     | 110110000011             |
| Кадр 4                    | 011111001111                     | 011011100001             |
| Кадр 5                    | 010001101010                     | 011001100101             |
| Служебный кадр данных     | 010111001010                     | 010001001011             |

На рисунке 13 приведена ФП для кадра данных.

В канал связи

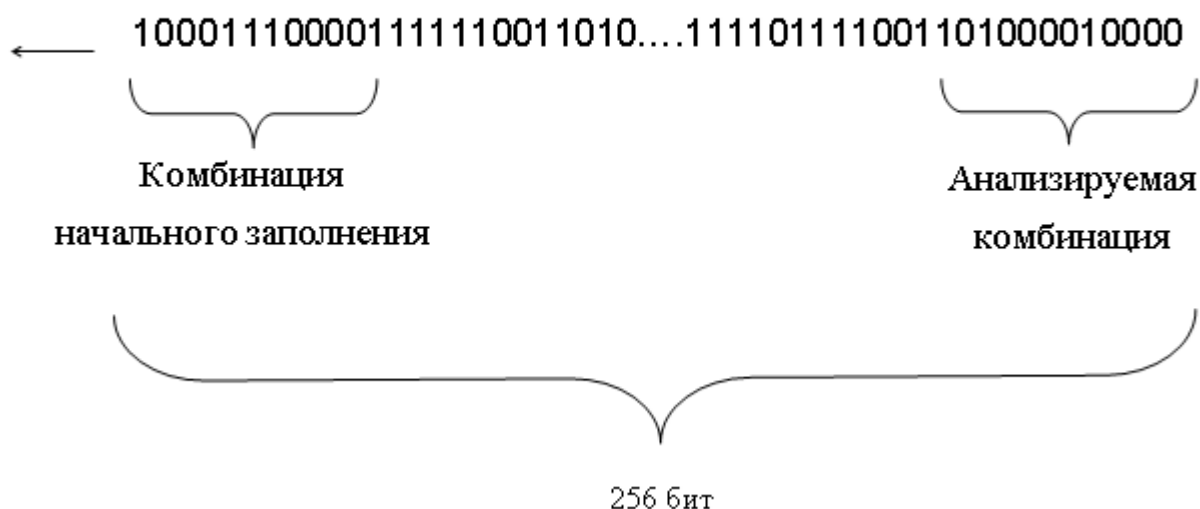


Рисунок 13 – Пример ФП для кадра данных «Кадр 1»

### 1.9 Выбор способа реализации УЗО

Для реализации цифровой обработки сигналов применяются современные цифровые процессоры, которые в свою очередь, позволяют реализовывать практически любые сложные алгоритмы обработки информации и управления. Но при аппаратной реализации основные действия реализуются одномоментно, без возможности добавления дополнительных алгоритмов. Поэтому, наиболее сложная основная часть реализуется программным способом.

Аппаратный способ реализации функций обработки информации гарантирует высокое быстродействие за счет создания устройств с “жесткой логикой”. Сложность таких устройств и аппаратные затраты быстро растут с повышением сложности реализуемых функций.

Программный способ реализации – подразумевает программирование процессора или микропроцессора. Аппаратура становится универсальной и пригодной для реализации целого класса функций.

В данном случае под программированием понимается: процедурное программирование, то есть задание последовательности действий по

реализации алгоритма. Но существует возможность и структурного программирования - указание путей следования потоков данных от одних аппаратных средств к другим. Программное и структурное программирование не исключают друг друга.

Сложность функций практически не влияет на сложность аппаратуры. Все сводится лишь к изменению сложности программы и емкости ЗУ. То есть, чем больше обращений к памяти, а, следовательно, больше время решения и ниже производительность ЭВМ.

Для переналадки устройства достаточно сменить программу и устройство сопряжения.

Недостаток – быстродействие аппаратуры ограничено тактовой частотой процессора.

Учитывая все вышесказанное и требования ТЗ по скорости передачи информации, в данном дипломном проекте, как наиболее оптимальный, выбирается программный способ построения УЗО, обеспечивающий требуемые параметры при низких аппаратных затратах и малых габаритах устройства.

## 2 Разработка программного обеспечения модуля помехоустойчивого кодирования малогабаритной аппаратуры передачи данных

### 2.1 Программная модель малогабаритной аппаратуры передачи данных

Программная структура МАПД состоит из следующих основных модулей (узлов):

- модуль автоматического коммутатора пакетов (АКП) реализующий стык с ООД для ввода (вывода) пакетов информации и команд управления, а также осуществляющий адресование и маршрутизацию введенных пакетов информации по специальным протоколам взаимодействия, принятым в ВС РФ;
- модуль криптозащиты, обеспечивающий криптозащиту информации;
- модуль УЗО, реализующий помехоустойчивое кодирование пакетов информации в соответствии с алгоритмами помехоустойчивого кодирования, принятыми в ВС РФ, реализация при передаче пакетов информации по каналу связи канальных протоколов взаимодействия, принятых в ВС РФ, а также взаимодействие с различными радиосредствами для передачи (приёма) пакетов информации.

Структура МАПД представлена на рисунке 14.

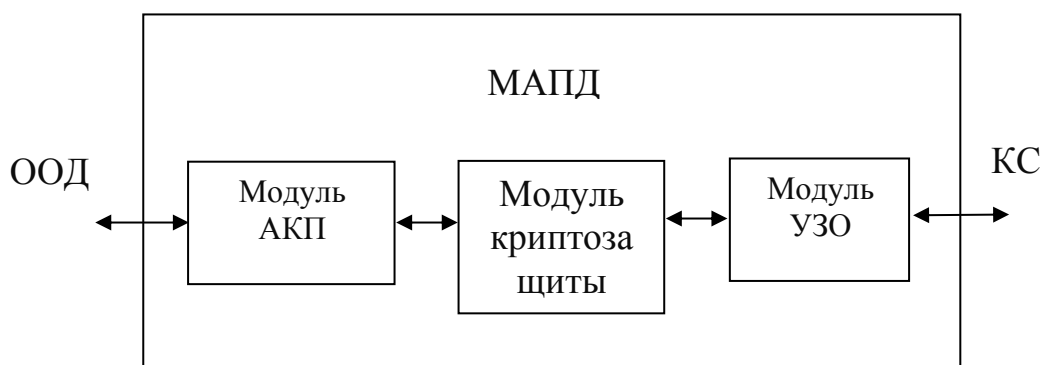


Рисунок 14 – Структура МАПД

### 2.2 Описание межмодульного взаимодействия

Взаимодействие между программными модулями, осуществляется пакетами информации через специально выделенные буфера обмена согласно

внутренним протоколам информационно-логического взаимодействия. Все межмодульные пакеты информации содержат специальный заголовок, содержащий адрес модуля отправителя пакета и адрес модуля получателя пакета, код команды, длину команды и защищаются контрольной суммой.

Формат межмодульного пакета приведён в таблице 5.

Таблица 5 - Формат межмодульного пакета

|                                                          |                            |
|----------------------------------------------------------|----------------------------|
| Адрес<br>модуля источника                                | Адрес<br>модуля получателя |
| Код команды                                              |                            |
| Размер пакета, байт                                      |                            |
| Информационный кадр для<br>помехоустойчивого кодирования |                            |
| Контрольная сумма                                        |                            |

Модуль ЗАС после осуществления криптозащиты записывает зашифрованный пакет информации, сопровождаемый соответствующим заголовком, в специально выделенный буфер обмена с модулем УЗО. Таким образом, формируется входной информационный кадр для модуля УЗО, который включает зашифрованный пакет информации, предназначенный для передачи в канал связи, и эталон квитанции (ЭКВТ). Эталон квитанции в канал связи не передаётся и хранится в модуле УЗО для сравнения с квитанцией, принимаемой из канала связи.

Структура входного кадра на входе УЗО представлена в таблице 6.

Таблица 6 – Структура входного информационного кадра модуля УЗО

|                      |      |
|----------------------|------|
| Зашифрованные данные | ЭКВТ |
|----------------------|------|

Размеры кадров информации на стыке модулей УЗО - ЗАС приведены в таблице 7.

Таблица 7 - Размеры кадров информации на стыке модулей УЗО - ЗАС

| Название ГДП | Размер (бит) |
|--------------|--------------|
| ГДПН         | 688          |
| ГДП0         | 528          |
| ГДП1         | 1360         |
| ГДП2         | 2256         |
| ГДП3         | 3600         |

После кодирования помехоустойчивым каскадным кодом введенного информационного кадра и формирования ПЦС длиной 256 бит модуль УЗО выдаёт закодированные кадры последовательно в канал связи. Формат закодированного информационного кадра данных выдаваемого в канал связи представлен в таблице 8.

Таблица 8 - Формат информационного кадра данных, выдаваемого в канал связи

|    |    |                                    |
|----|----|------------------------------------|
| ЗП | ФП | Закодированный информационный кадр |
|----|----|------------------------------------|

### 2.3 Обоснование выбора языка программирования

Одним из основных требований к проектируемому модулю помехоустойчивого кодирования информации в целом является его переносимость без существенных затрат на другие аппаратные платформы. На практике основным языком программирования сигнальных процессоров является *Assembler*. *Assembler* представляет собой язык низкого уровня, который используется для преобразования исходной программы на машинный язык. На ассемблере машинный язык представляется проще, в виде символики, благодаря чему программы пишутся легче относительно высокоуровневых языков. Язык ассемблер дает доступ к регистрам, позволяет указывать методы адресации и использовать терминологию команд процессора для описания операций. Однако, среди массы достоинств языка *Assembler* имеется такой

недостаток, как сложность перенесения ПО, написанного на нем, на другую платформу.

В данной дипломной работе предлагается решение задачи по программированию на языке высокого уровня СИ, поскольку данный язык не имеет привязки к аппаратному обеспечению, а, следовательно, позволяет удовлетворить основное требование, предъявляемое к разрабатываемому программному обеспечению модуля помехоустойчивого кодирования.

## 2.4 Описание программного обеспечения модуля помехоустойчивого кодирования

Программное обеспечение модуля помехоустойчивого кодирования включает:

- программу кодирования внутренним кодом, занимающуюся кодированием информационных кадров кодом (24,16);
- программу кодирования внешним кодом, занимающуюся кодированием блоков внутреннего кода кодом Рида – Соломона;
- программу, осуществляющую приём посылки цикловой синхронизации;
- программу, осуществляющую прием информационных кадров из канала связи;
- программу декодирования внутреннего кода, занимающуюся декодированием блоков кодом (24,16);
- программу декодирования внешнего кода, занимающуюся декодированием информационных кадров кодом Рида – Соломона.

## 2.5 Описание алгоритма кодирования кодом Рида - Соломона

Процедура кодирования начинается с выполнения начальных установок, а именно, с преднамеренного искажения проверочной части путем заполнения проверочной части блоков нулями и заполнения таблиц локаторов стираний локаторами символов проверочных частей.

Далее осуществляется вычисление коэффициентов многочлена локаторов стираний  $\lambda(x)$ .

После этого вычисляется производная  $\lambda'(x)$  и значение искажений  $E(x)$ .

По вычисленным значениям искажений происходит исправление искажений  $e_i$  и осуществляется коррекция синдрома  $S'$ .

Далее проверяется все ли стирания исправлены и, если нет, то осуществляется возврат к процедуре вычисления значения искажения  $E(x)$ . В противном случае. Если все стирания были исправлены, то осуществляется выход из процедуры кодирования.

Более подробно все перечисленные процедуры описаны в п. 2.8.

Алгоритм кодирования приведен на рисунке 15 и на плакате “Алгоритм кодирования”.

Текст программы приведен в приложении А.

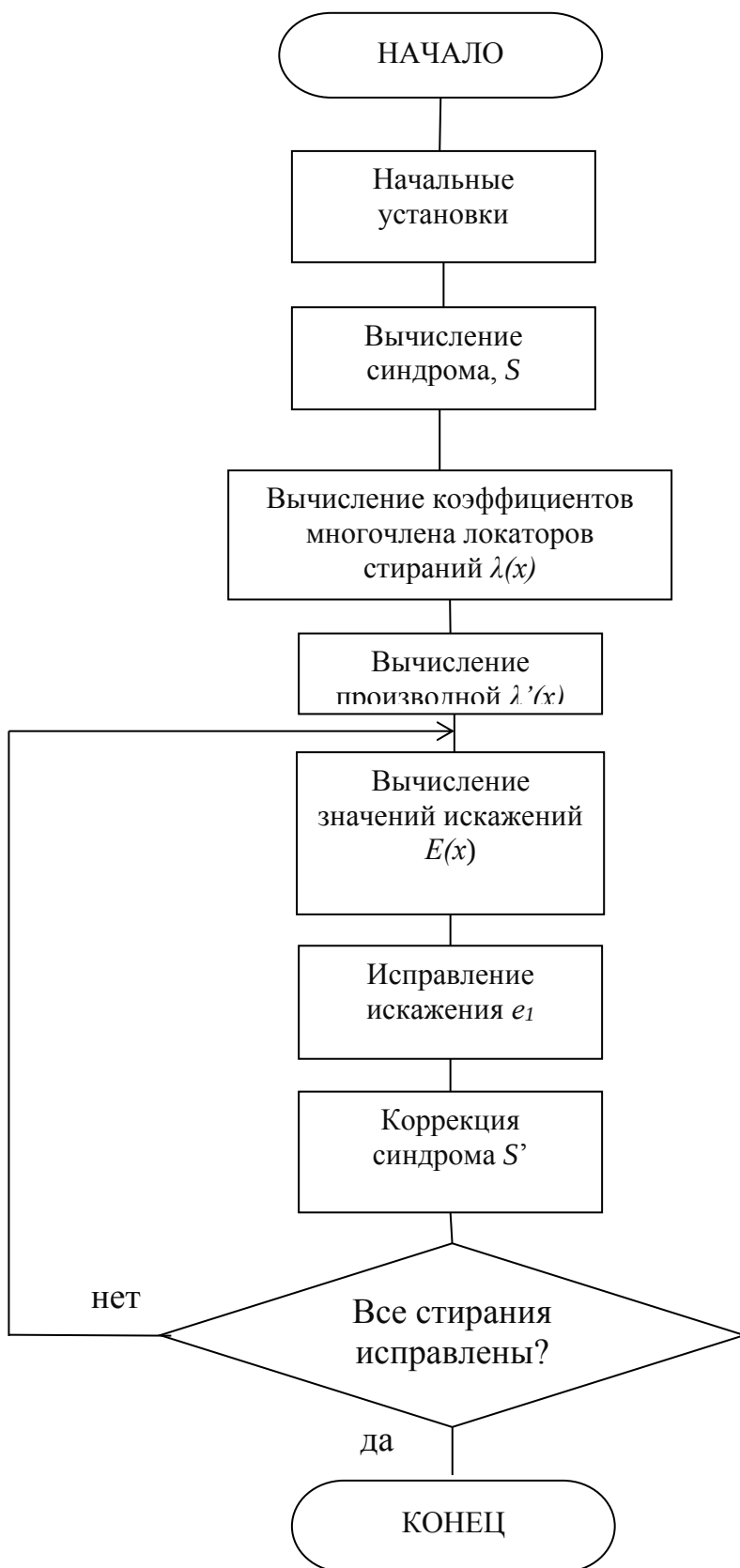


Рисунок 15 – Алгоритм кодирования

## 2.6 Описание алгоритма приема информации

Прием информации начинается с приема фазирующей посылки. Если фазирующая посылка не принята, то принимаемые биты продолжают анализироваться на наличие таковой.

После приема фазирующей посылки производятся начальные установки для приема пакета и начинается накопление блока (24,16). Процесс накопления продолжается до тех пор, пока не будет принят весь блок.

Если весь блок принят, то осуществляется вычисление синдрома внутреннего кода (24,16) методом деления информационного многочлена на порождающий.

После вычисления синдрома, если выключено синхронное накопление блоков, осуществляется проверка полученного значения. В противном случае, если синхронное накопление включено, то среди полученных блоков выбирается более надежный и осуществляется проверка выбранного значения синдрома.

Если получено не нулевое значение, то выполняется исправление блока. В противном случае, если получено нулевое значение, то выполняется анализ принятых блоков. Если не все блоки приняты, то происходит возврат к накоплению блоков (24,16).

В случае, когда блок исправлен, локатор блока помещается в таблицу локаторов ненадежных блоков. Если исправление блока не было выполнено, то блок помечается как стертый, счетчик стертых блоков увеличивается на единицу и локатор такого блока заносится в таблицу локаторов стертых блоков.

После этого выполняется анализ принятых блоков. Если не все блоки приняты, то происходит возврат к накоплению блоков (24,16).

Если все блоки приняты, то выполняется проверка значений счетчика стираний. В случае, когда его значение больше 16 происходит отказ от дальнейшего декодирования, включается синхронное накопление и осуществляется поиск фазирующей посылки.

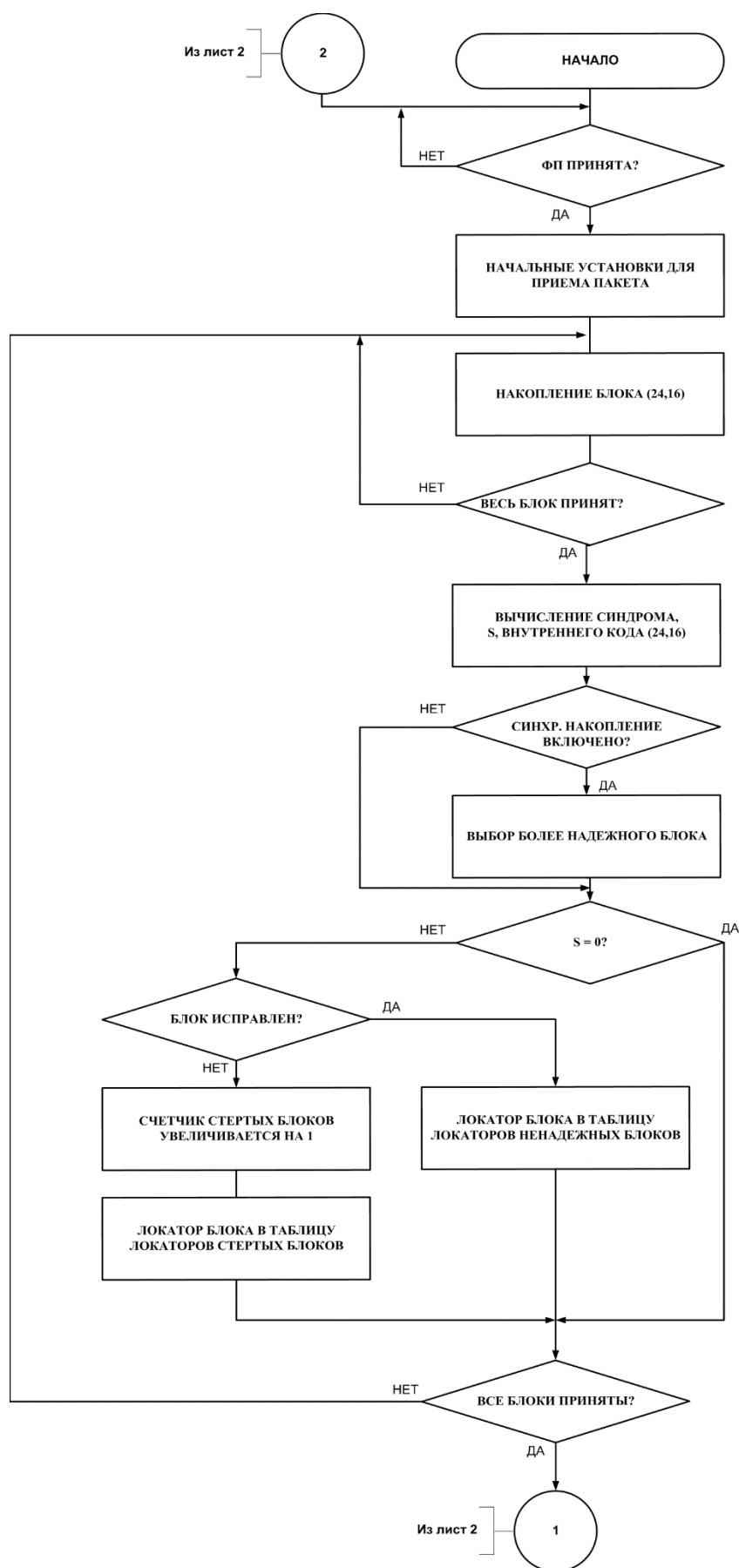
В случае, когда значение счетчика стираний меньше 10 происходит дополнение локаторов стираний из таблицы локаторов ненадежных блоков и выполняется декодирование кода РС. Если значение счетчика стираний больше 10, то выполняется декодирование кода РС.

Если произошел отказ от декодирования, то выполняется включение синхронного накопления и осуществляется поиск фазирующей посылки.

Если отказа не произошло, то осуществляется выдача принятого сообщения в модуль криптозащиты для расшифровки.

Алгоритм приема информации представлен на рисунке 16 и на плакате “Алгоритм приема информации”.

Текст программы приведен в приложении А.



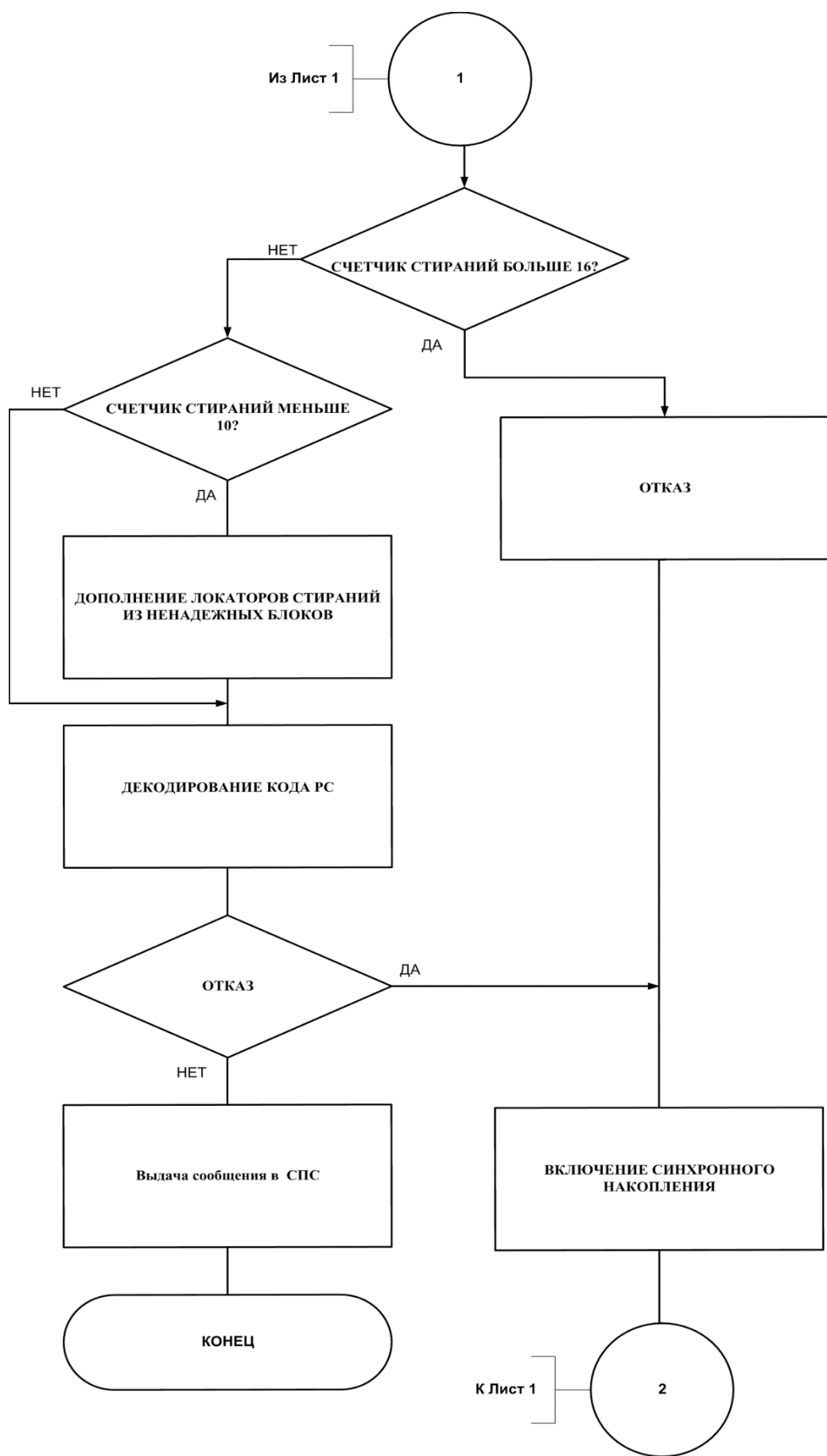


Рисунок 16 – Алгоритм приема информации, лист 2

## 2.7 Описание алгоритма декодирования кодом Рида–Соломона

Циклические коды Рида-Соломона не двоичные и для них характерны операции на основе арифметики полей Галуа  $GF(q)$ , где  $q$  - основание кода.

Алгоритм декодирования включает семь этапов. Кодирование аналогично декодированию, только последние  $R$  символов блока исправляемые являются проверочными и, согласно специальным протоколам взаимодействия, принятым в ВС РФ, их количество постоянно и равно 16. То есть кодирование сводится к декодированию при искажениях, места которых известны (стирания).

Первый этап - вычисление синдромов (проверок), с одновременным запоминанием номера (локатора) символов, в которых при декодировании внутренним кодом были обнаружены ошибки (стирания).

Второй этап - вычисление многочлена локаторов стираний (ошибок)  $\lambda(x)$ , корнями которого являются локаторы стираний  $z_j$ . Здесь же вычисляется производная от многочлена локаторов стираний.

Третий этап - вычисление модифицированных синдромов  $T$ , число которых соответствует числу исправляемых ошибок.

Четвертый этап - вычисление коэффициентов многочлена локаторов ошибок.

Пятый этап - определение локаторов (номеров позиций) ошибок.

Шестой этап - вычисление многочлена локаторов искажений (ошибок и стираний)

Седьмой этап - определение значений искажений (исправление стираний).

На первом этапе вычисление синдромов осуществляется по формуле Горнера

$$S_n = \prod_{i=1}^n (c_i \alpha^n + c_{i+1}), \quad (2.1)$$

где  $c_i$  – информационный символ.

Алгоритм вычисления коэффициентов многочлена синдромов представлен на рисунке 17.

На втором этапе вычисляются коэффициенты многочлена локаторов стираний  $\lambda(x)$ , корнями которого являются локаторы стираний  $z_i$ , которые представляются в виде степеней  $\alpha$ .

Многочлен локаторов стираний имеет следующий вид

$$\lambda(x) = (x + z_1)(x + z_2) \dots (x + z_v), \quad (2.2)$$

где коэффициенты многочлена равны:

$$\begin{aligned} \lambda_1 &= z_1 + z_2 + \dots + z_v; \\ \lambda_2 &= z_1 z_2 + z_2 z_3 + \dots + z_{v-1} z_v; \\ &\dots \\ \lambda_v &= z_1 z_2 \dots z_v. \end{aligned} \quad (2.3)$$

В общем случае вычисление коэффициентов многочлена локаторов стираний осуществляется по следующей формуле:

$$\lambda_i^j = \lambda_i^{j-1} + \lambda_{i-1}^{j-1} \cdot Y_j \quad (2.4)$$

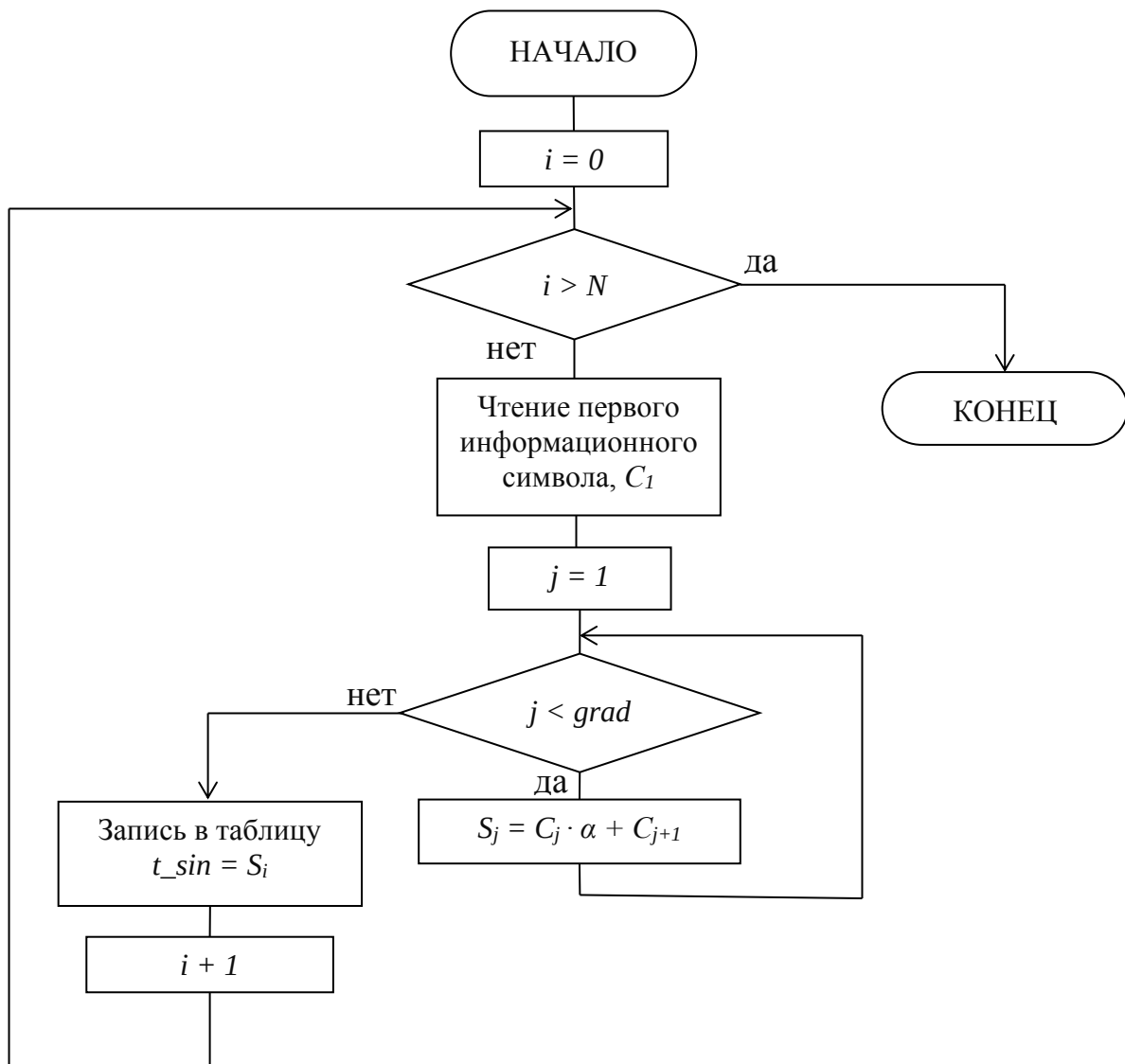


Рисунок 17– Вычисление коэффициентов  
многочлена синдромов

где  $Y_j$  – локатор стирания.

Алгоритм вычисления коэффициентов многочлена локаторов стираний представлен на рисунке 18.

На третьем этапе для исключения из многочлена синдромов  $S(x)$  информации о стираниях вычисляется модифицированный синдром путем перемножения многочлена синдромов  $S(x)$  на многочлен локаторов стираний  $\lambda(x)$ , т.е.

$$T(x) = S(x) \cdot \lambda(x). \quad (2.5)$$

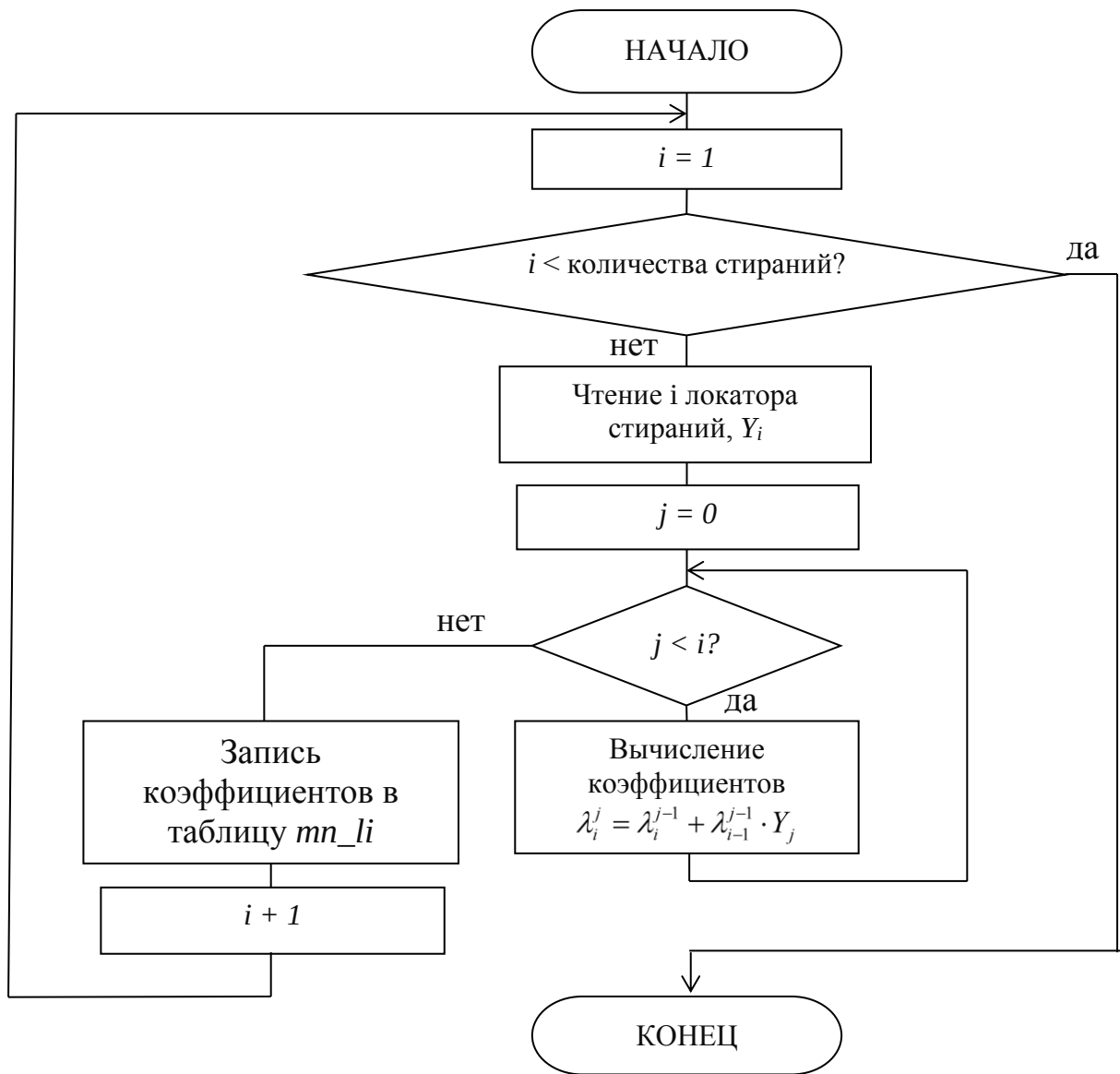


Рисунок 18– Алгоритм вычисления коэффициентов  
многочлена локаторов стираний

Каждый коэффициент многочлена  $T(x)$  представляется в виде:

$$T_j = \sum_{i=1}^t e'_i \cdot x_i^j, \quad (2.6)$$

где  $e'_i$  – модифицированное значение ошибки.

Тогда получается система уравнений:

$$\left\{ \begin{array}{l} T_0 = S_0 \cdot \lambda_s + S_1 \cdot \lambda_{s-1} + \dots + S_{s-1} \cdot \lambda_1 + S_s; \\ T_1 = S_1 \cdot \lambda_s + S_2 \cdot \lambda_{s-1} + \dots + S_s \cdot \lambda_1 + S_{s+1}; \\ \dots \\ T_{15-s} = S_{15-s} \cdot \lambda_s + S_{16-s} \cdot \lambda_{s-1} + \dots + S_{14} \cdot \lambda_1 + S_{15}. \end{array} \right. \quad (2.7)$$

Алгоритм вычисления модифицированных синдромов представлен на рисунке 19.

На четвертом этапе осуществляется вычисление коэффициентов многочлена локаторов ошибок  $\delta(x)$ . Сам многочлен имеет следующий вид

$$\delta(x) = x^t + \delta_1 \cdot x^{t-1} + \dots + \delta_t = (x + X_1) \cdot (x + X_2) \cdot \dots \cdot (x + X_t). \quad (2.8)$$

Для вычисления коэффициентов многочлена локаторов ошибки применяется способ, аналогичный способу нахождения модифицированных синдромов, т. е.  $T(x) \cdot \delta(x)$ . Тогда получается система уравнений:

$$\left\{ \begin{array}{l} T_0 \cdot \delta_t + T_1 \cdot \delta_{t-1} + \dots + T_{t-1} \cdot \delta_1 + T_t = 0, \\ T_1 \cdot \delta_t + T_2 \cdot \delta_{t-1} + \dots + T_t \cdot \delta_1 + T_{t+1} = 0, \\ \dots \\ T_{t-1} \cdot \delta_t + T_t \cdot \delta_{t-1} + \dots + T_{2t-2} \cdot \delta_1 + T_{2t-1} = 0. \end{array} \right. \quad (2,9)$$

Данная система уравнений решается методом Гаусса:

$$\left| \begin{array}{cccc} T_0 & T_1 & \dots & T_{t-1} \\ T_1 & T_2 & \dots & T_t \\ \dots & \dots & \dots & \dots \\ T_{t-1} & T_t & \dots & T_{2t-1} \end{array} \right| \cdot \left| \begin{array}{c} \delta_t \\ \delta_{t+1} \\ \dots \\ \delta_1 \end{array} \right| = \left| \begin{array}{c} T_t \\ T_{t+1} \\ \dots \\ T_{2t-1} \end{array} \right| \quad (2,10)$$

На пятом этапе определение номеров локаторов ошибок сводится к нахождению корней уравнения:

$$x^t + \delta_1 \cdot x^{t-1} + \dots + \delta_t = 0. \quad (2.11)$$

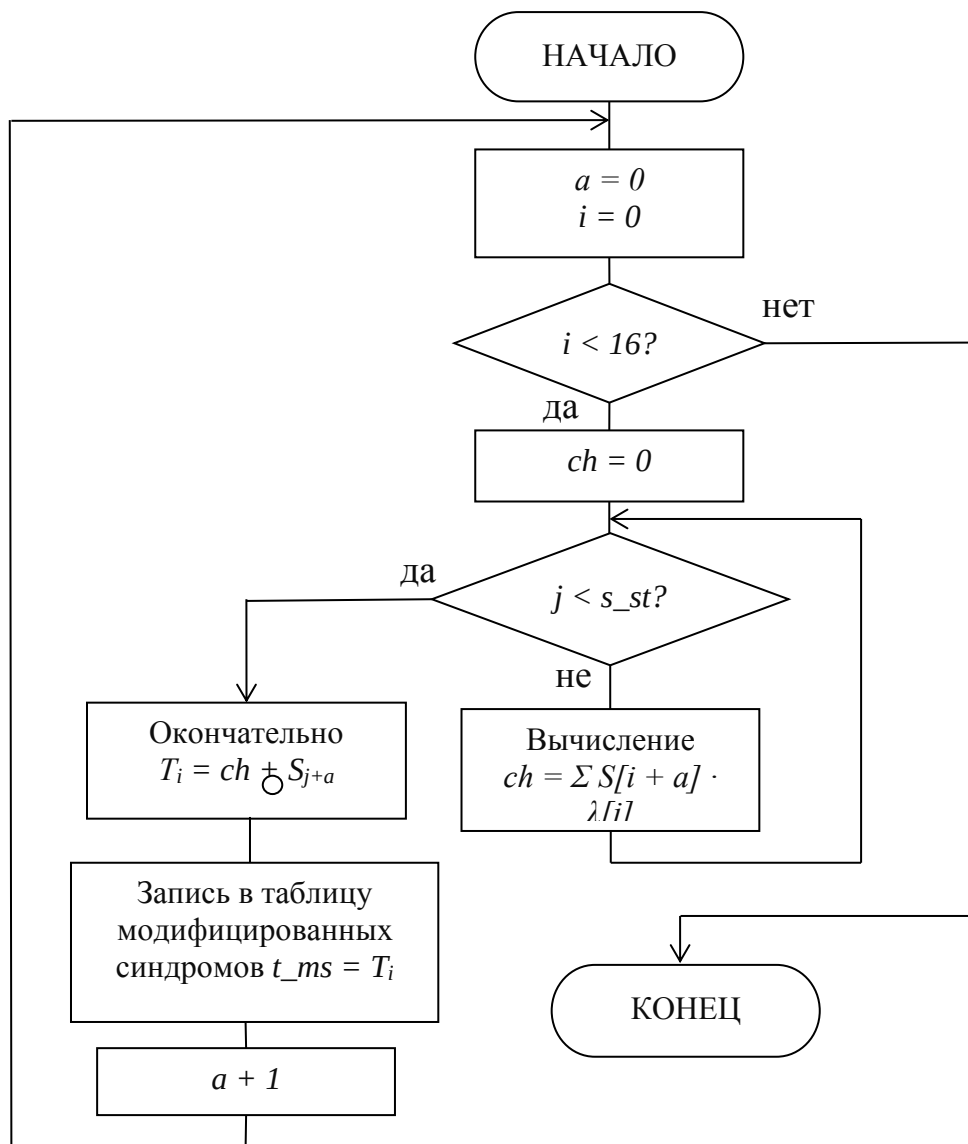


Рисунок 19 – Алгоритм вычисления модифицированных синдромов

Для определения трех локаторов ошибок необходимо решить уравнение:

$$x^3 + \delta_1 \cdot x^2 + \delta_2 \cdot x + \delta_3 = 0. \quad (2.12)$$

Для определения одного локатора ошибки используется соотношение:

$$x + \delta_3 = 0. \quad (2.13)$$

Для решения кубического уравнения используется следующий алгоритм.  
Подставим:

$$x = y + \delta_1. \quad (2,14)$$

Получим:

$$y^3 + \delta_1^3 + y^2 \cdot \delta_1 + y \cdot \delta_1^2 + \delta_1^3 + \delta_1 \cdot y^2 + \delta_2 \cdot \delta_1 + \delta_2 \cdot y + \delta_3 = 0; \quad (2,15)$$
$$y^3 + y \cdot (\delta_1^2 + \delta_2) + \delta_1 \cdot \delta_2 + \delta_3 = 0.$$

Если  $\delta_1^2 + \delta_2 = 0$ , то по таблице TDEL3 определяются корни уравнения

$$y^3 + \delta_1 \cdot \delta_2 + \delta_3 = 0. \quad (2,16)$$

Если  $\delta_1^2 + \delta_2 \neq 0$ , то подставляем

$$y = z \cdot (\delta_1^2 + \delta_2)^{1/2}. \quad (2,17)$$

Получается уравнение:

$$z^3 + z + \frac{\delta_1 \cdot \delta_2 + \delta_3}{(\delta_1^2 + \delta_2)^{3/2}} = 0. \quad (2,18)$$

Тогда по адресу свободного члена считывается из таблицы ТАК два корня

$$z_1 \text{ и } z_2, \text{ а } z_3 = z_1 + z_2. \quad (2,19)$$

Локаторы ошибок определяются из выражения

$$x_i = z_i \cdot (\delta_1^2 + \delta_2)^{1/2} + \delta_1. \quad (2,20)$$

После вычисления локаторов ошибок производится контроль соответствия локаторов градации длины пакета. В случае несоответствия происходит отказ от декодирования.

На шестом этапе осуществляется вычисление многочлена локаторов искажений также как и вычисление многочлена локаторов стираний (ошибок) на втором этапе.

На седьмом этапе осуществляется вычисление значений искажений (исправление стираний) по формуле:

$$e_{s+t} = \frac{\lambda_{s+t-1} \cdot S_0 + \lambda_{s+t-2} + \dots + \lambda_1 \cdot S_{s+t-2} + S_{s+t-1}}{\lambda'(X_{s+t})}. \quad (2.21)$$

Сначала вычисляется значение искажения, локатор которого при получении многочлена локаторов искажений обрабатывался последним. Далее корректируются синдром таким образом, что следующий с конца локатор искажения становится последним.

Общий алгоритм декодирования кодом РС представлен на рисунке 20 и на плакате “Алгоритм декодирования”.

Текст программы приведен в приложении А. Таблицы для кодирования и декодирования приведены в приложении Б.

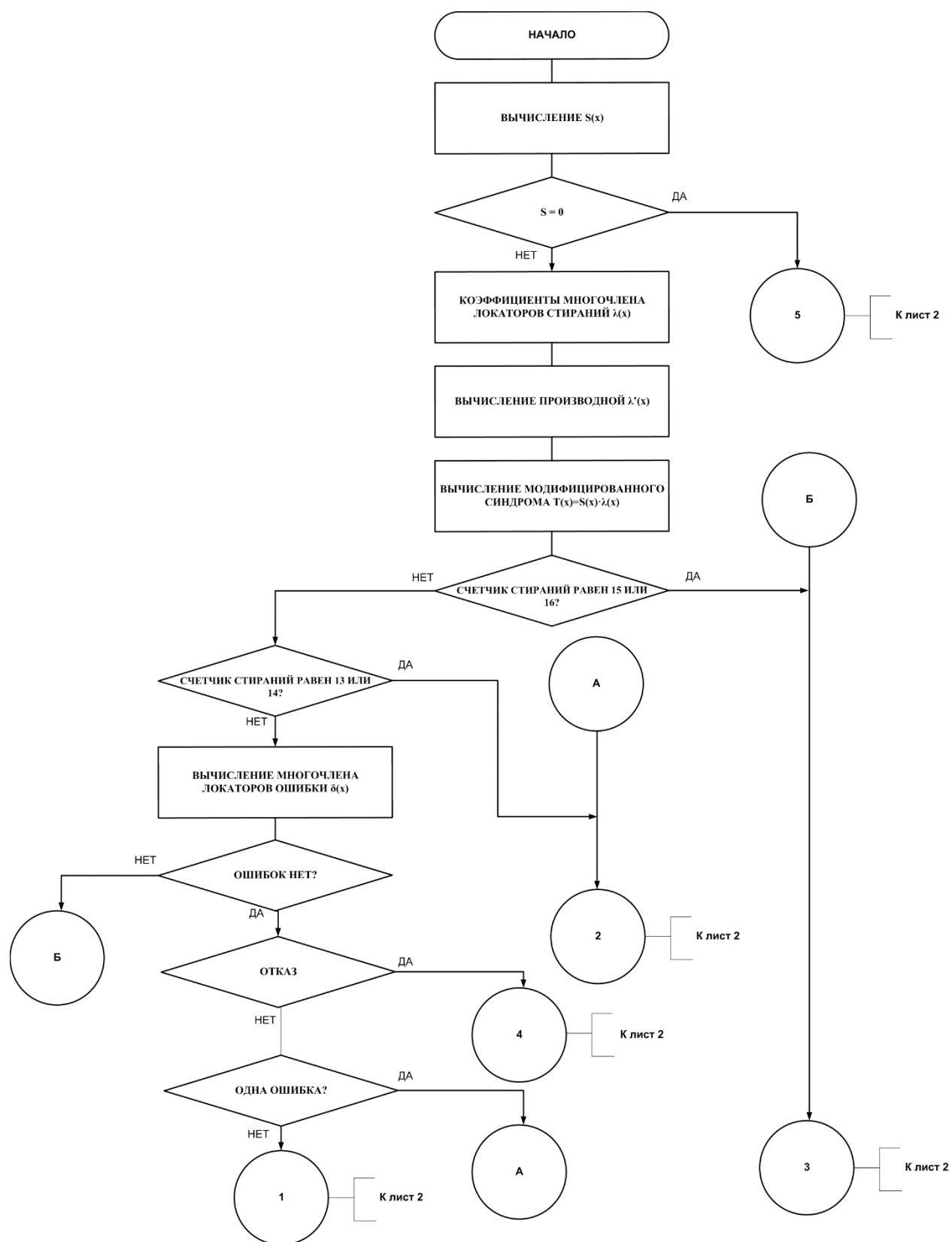


Рисунок 20 – Алгоритм декодирования кода РС, лист 1

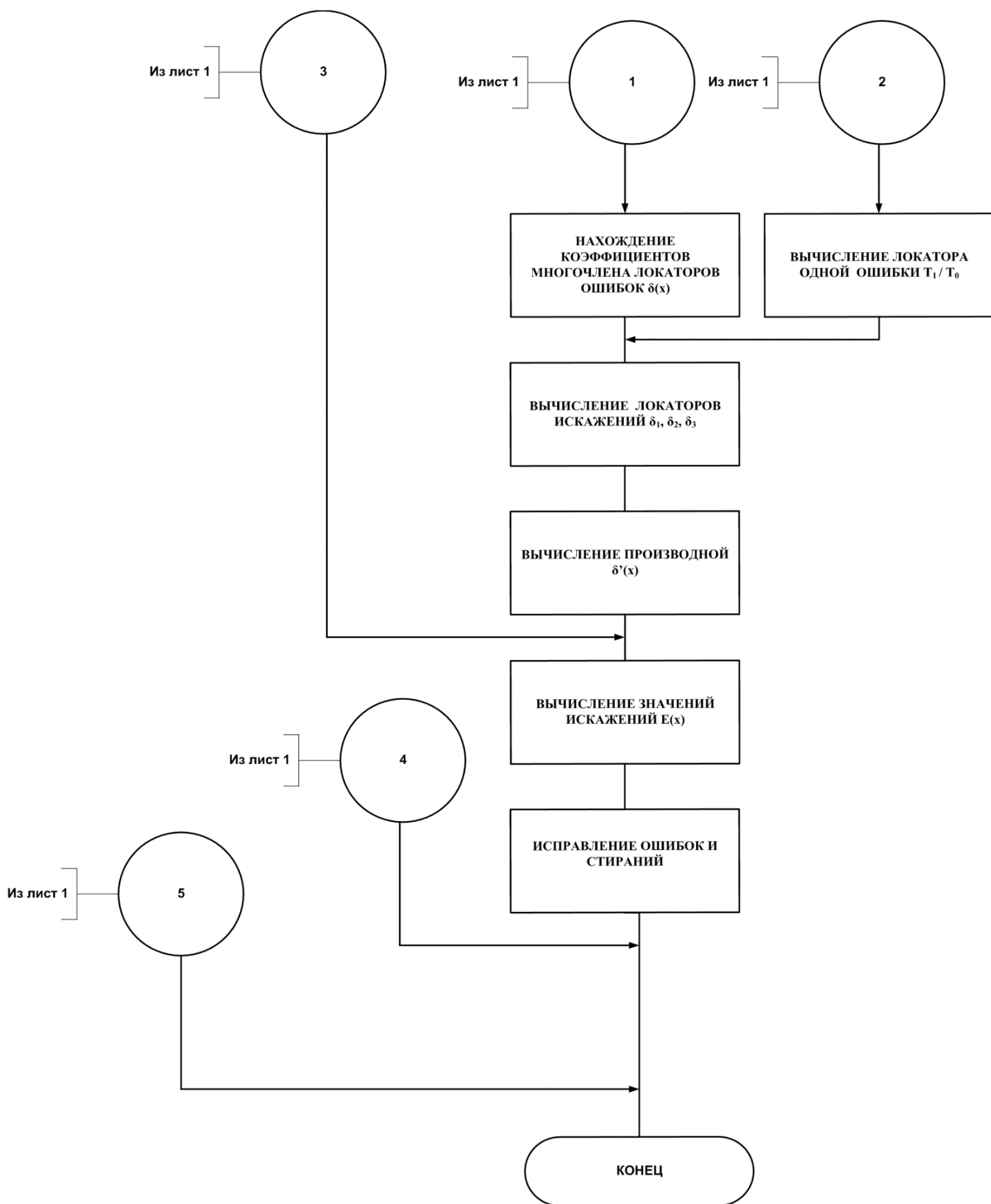


Рисунок 21 – Алгоритм декодирования кода РС, лист 2

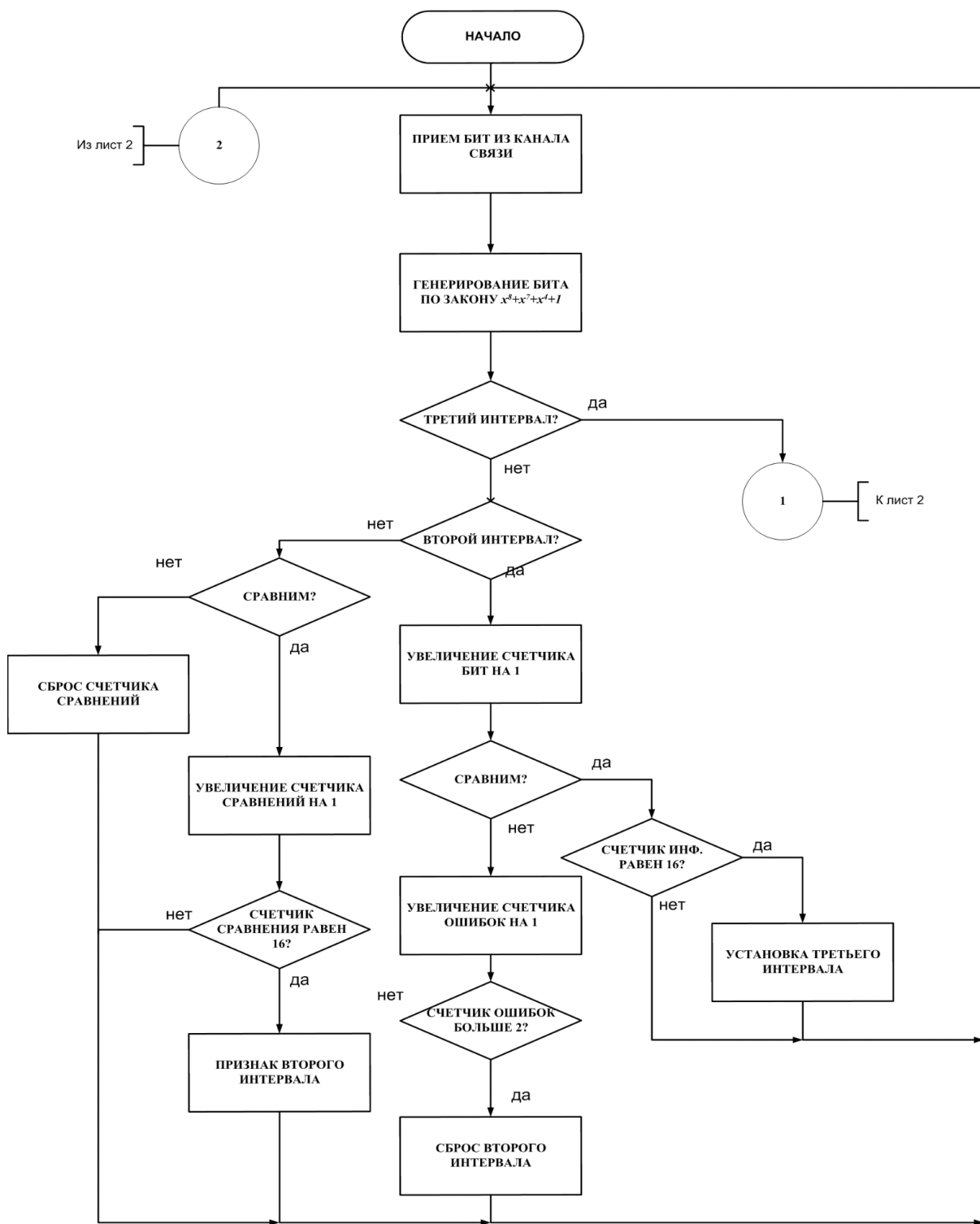


Рисунок 22 – Алгоритм Фазирования, лист 1

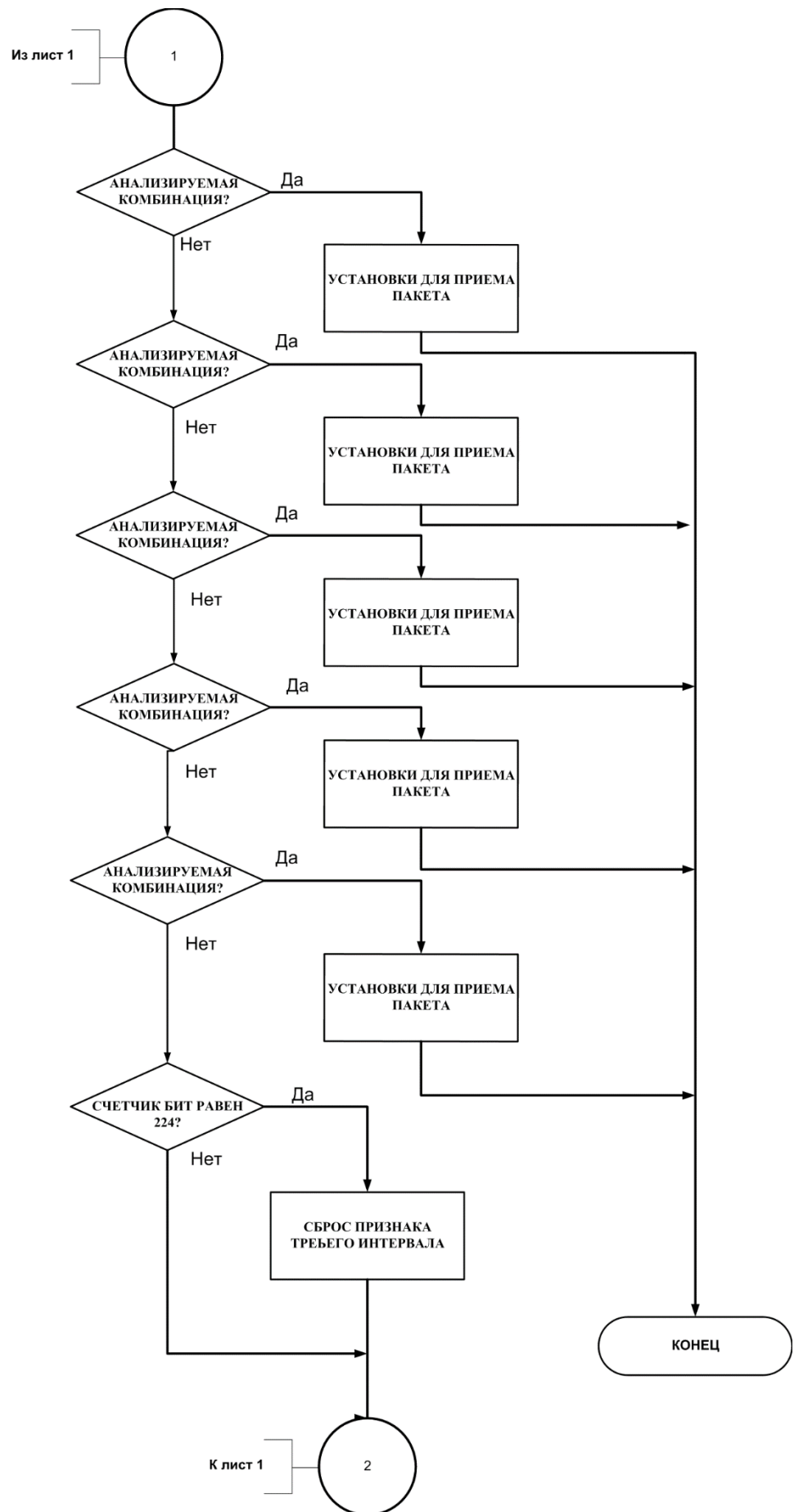


Рисунок 23 – Алгоритм Фазирования, лист 2

### 3 Разработка организационных и технических мероприятий для снижения воздействия психофизических факторов, влияющих на психоэмоциональное состояние специалиста по защите информации

#### 3.1 Вредные психофизиологические факторы, влияющие на психоэмоциональное состояние специалиста по защите информации.

На любой работе человек ежедневно подвергается опасности, которая осуществляется производственной средой. Пусть работа специалиста по защите информации не подразумевает интенсивных физических нагрузок или же взаимодействие с потенциально опасными предметами, вредный производственный фактор невозможно полностью исключить.

Работа специалиста по защите информации так же сопряжена с опасными производственными факторами, один из них – электрический ток.

Вредные и опасные факторы подразделяются на:

- физические;
- химические;
- биологические;
- психофизиологические.

В рамках ВКР будут рассмотрены психофизиологические факторы.

Психофизиологические факторы – это статические и динамические перегрузки, так же нервно-психические перегрузки, а именно: умственное перенапряжение, монотонность труда, эмоциональные перегрузки.

Деятельность специалиста по защите информации обычно включает в себя:

- разработка, определение, составление мер по защите информации;
- организация, ведение, осуществления контроля мер по защите информации;
- анализ, корректировка, тестирования организации, осуществляющей контроль за информацией ограниченного доступа;

- проектирование, разработка, тестирование системы управления информационной безопасности.

В основном специалист по защите информации проводит около сорока часов на своём рабочем месте, сидя за компьютером. Соответственно, динамические нагрузки отсутствуют. Но при специфике работы специалист может выполнять трудовые функции, при нахождении в вынужденных позах. Даже при долгом сидячем положении на одном месте, могут возникнуть ряд осложнений.

Психические процессы составляют основу психической деятельности.

Нервные перегрузки негативно влияют на память. Запоминание связано с забыванием. По статистике, специалист, непривыкший к обработке большого количества информации, забывает до 75% информации за первые 8-9 часов работы.

Нервные перегрузки снижают внимание, то есть специалист со сниженным вниманием, не сохранит сосредоточенность сознания в значимой ситуации.

Все вышеперечисленные перегрузки приводят в гиподинамии.

Гиподинамия – уменьшение силовых показателей, тонуса мышц. Гиподинамия отрицательно влияет на внутренние органы и нервную систему.

Напряжённость внимания характеризуется длительностью сосредоточения наблюдения, числом объектов наблюдения.

Степень напряжённости анализаторских функций для зрительного анализатора зависит от размера объекта различения и расстояния объекта от глаз, различия в контрастности объекта различения и фона, для слухового анализатора – от соотношения между уровнями речи и шума.

Эмоциональное напряжение воздействует на центральную нервную систему. Негативное воздействие возможно из-за работы по напряжённому графику.

Заключительный из психофизиологических факторов – это монотонная работа. Существует два вида монотонности.

Монотонность за счёт информационной перегрузки одних и тех же нервных центров в результате поступления одинаковых сигналов или многократного повторения.

Второй вид монотонности вызван однообразным восприятием, из-за недостатка новой информации. Из-за монотонной работы, специалист теряет интерес, возникает состояние «производственной скуки».

### 3.2 Методы и средства физической культуры, снижающие воздействие психофизиологических факторов, влияющих на психоэмоциональное состояние специалиста по защите информации

Во избежание гиподинамии следует ввести на предприятие производственной гимнастики. При проектировании рабочих мест необходимо стремиться к тому, чтобы рабочая поза была как можно ближе к естественной позе человека. Поэтому целесообразно предусматривать возможность работы как стоя, так и сидя. Так же для снижения воздействия на нервную систему рекомендуется организация комнат психологической разгрузки.

Снижение нервных перегрузок, связанных с плотным графиком работы, следует начинать с пересмотра графика работы и с составлением нового графика. Так же необходимо минимизировать эстетический дискомфорт и физиологический дискомфорт.

Монотонная работа отрицательно сказывается на эффективности производства: ухудшаются экономические показатели, повышается аварийность, травматизм, растет текучесть кадров.

Основные меры по уменьшению влияния монотонности на человека:

- пересмотр выдаваемых задач специалистом;
- автоматизация однотипных процессов;
- осуществления перевода персонала на близкую, но отличную от предыдущего вида деятельности, для переключения внимания;
- принимать оптимальные режимы труда и отдыха;

- организация музыкального сопровождения во время неотложной монотонной работы.

В процессе профессионального обучения подготовить работающего к работе в экстремальных и аварийных ситуациях так, чтобы стрессы не помешали ему в этих сложных условиях выполнить свои профессиональные обязанности.

Наведение чистоты и поддержания порядка, так же аутентичное оформление производственных помещений или приборов способствуют повышению настроения и чувству комфорта на рабочем месте.

Выбор мер поощрения, наказания или контроля, необходимо осуществлять согласно с индивидуальными особенностями специалиста. Наказание робкого человека, способно дезорганизовать его работу на длительный период времени.

Для организации дружеского сотрудничества, рекомендуется проводить кооперативы по случаю значимых дат.

#### 4 Технология разработки программного обеспечения малогабаритных устройств передачи данных

Технология разработки описывается несколькими моделью жизненного цикла ПО.

Модель жизненного цикла ПО – это структура, определяющая последовательность выполнения и взаимосвязи процессов, действий и задач на протяжении жизненного цикла.

Жизненный цикл ПО – это события, происходящие с системой в процессе её создания и использования. То есть это весь промежуток времени от начала создания программного продукта, до полной разработки и внедрения.

Ниже представлено описание нескольких основных моделей.

Модель кодирования и устранения ошибок.

Алгоритм модели:

- постановка задачи;
- выполнение;
- проверка результата;
- при необходимости переход к постановке задачи.

Данная модель является устаревшей и в настоящее время не используется, но именно с неё появилось понимание в необходимости введения структуры для разработки программных продуктов.

Каскадная модель жизненного цикла программного обеспечения.

Алгоритм модели:

- выработка системных требований;
- выработка требований к ПО;
- анализ;
- проектирование;
- кодирование;
- тестирование;
- эксплуатация.

Этапы проекты выполняются последовательно, что позволяет оценивать качество продукта на каждом этапе.

V модель или разработка через тестирование.

Алгоритм модели:

- планирование требований;
- анализ требований продукта и спецификаций;
- разработка архитектуры проекта;
- детализированная разработка проекта;
- кодирование
- модульное тестирование;
- тестирование и проверка системы;
- введение в эксплуатацию и проверка.

Спиральная модель.

В данной модели особое внимание уделяется начальным этапам разработки: выработке стратегии, анализу и проектированию, где реализуемость тех или иных технических решений проверяется и обосновывается посредством создания прототипов (макетирования). Каждый виток спирали предполагает создание некой версии продукта или какого-либо его компонента; при этом уточняются характеристики и цели проекта, определяется его качество и планируются работы следующего витка спирали.

Поэтапная модель с промежуточным контролем.

Данная модель была выбрана при разработке программного обеспечения модуля помехоустойчивой защиты информации. Модель состоит из последовательно расположенных этапов с обратной связью. Благодаря промежуточному контролю исправление ошибок происходит на каждом из этапов сразу при выявлении проблемы. Следующий этап не начинается, пока не завершится предыдущий.

Алгоритм модели:

- постановка задачи;
- проектирование;

- алгоритмизация;
- программирование;
- тестирование и отладка;
- эксплуатация и сопровождение.

При постановке задачи в описании должны быть указаны;

- наименование и назначение;
- функциональная подсистема, к которой относится данная задача;
- событие, инициирующее решение задачи или периодичность;
- продолжительность решения задачи: минимальная, максимальная, средняя;
- должностное лицо, осуществляющее решение задачи, подразделение;
- входная информация, ее источники;
- справочная информация;
- выходная информация;
- связь с другими задачами.

На этапе проектирования производится анализ ТЗ. После анализа ТЗ программист выбирает основной метод решения задачи и составляет общий проект. В проект входят:

- основные части, проектируемой программы;
- функции частей, проектируемой программы;
- последовательность выполнения;
- входные данные;
- выходные данные.

Этап алгоритмизации чаще всего не включают в основные этапы модели, так как данный этап является вспомогательным к выполнению этапа программирование. Данный этап необходим для облегчения работы программиста и для минимизации и предотвращения ошибок, связанных с программированием. Это обуславливается тем, что формулировка алгоритма закрепляет последовательность основных шагов в программе, чётко фиксирует

функциональное содержание частей, уделяет необходимое внимание логической структуре.

После разработки алгоритмов, следует процесс перевода созданных алгоритмов на выбранный язык программирования.

В данном дипломном проекте выбран язык Си, система программирования является Borland.

Традиционные подходы к разработке прикладной программы (ПП) (структурное проектирование и информационное моделирование) всегда подчеркивали различия между данными и процессами их обработки.

Технология структурного подхода ориентирована в первую очередь на процессы обработки данных с последующим установлением необходимых для этого данных и организацию информационных потоков между связанными процессами.

Объектно-ориентированная технология объединяет данные и процессы в логические сущности - объекты, которые имеют способность наследовать характеристики. Следовательно, уменьшаются затраты на создание ПП, повышается эффективность жизненного цикла ПП.

Метод нисходящего проектирования предполагает последовательное разложение общей функции обработки данных на простые функциональные элементы. В результате строится иерархическая схема, отражающая состав и взаимоподчиненность отдельных функций, которая носит название функциональной структуры алгоритма (ФСА) приложения.

Последовательность действий по разработке функциональной структуры алгоритма приложения:

- определяются цели автоматизации предметной области и их иерархия (цель-подцель);
- устанавливается состав приложений (задач обработки), обеспечивающих реализацию поставленных целей;

- уточняется характер взаимосвязи приложений и их основные характеристики (информация для решения задач, время и периодичность решения, условия выполнения и др.);

- определяются необходимые для решения задач функции обработки данных;

- выполняется декомпозиция функций обработки до необходимой структурной сложности, реализуемой предполагаемым инструментарием.

Подобная структура отражает состав и взаимосвязь функций обработки информации для реализации приложений, хотя и не раскрывает логику выполнения каждой отдельной функции, условия или периодичность их вызовов.

Функции ввода-вывода информации рекомендуется отделять от функций вычислительной или логической обработки данных.

По частоте использования функции делятся на:

- однократно выполняемые;
- повторяющиеся.

Модульное программирование основано на понятии модуля - логически взаимосвязанной совокупности функциональных элементов, оформленных в виде отдельных программных модулей.

Модуль характеризуют:

- один вход и один выход - на входе программный модуль получает определенный набор исходных данных, выполняет содержательную обработку и возвращает один набор результатных данных, т.е. реализуется стандартный принцип IPO (Input - Process - Output) – вход-процесс-выход;

- функциональная завершенность - модуль выполняет перечень регламентированных операций для реализации каждой отдельной функции в полном составе, достаточных для завершения начатой обработки;

- логическая независимость - результат работы программного модуля зависит только от исходных данных, но не зависит от работы других модулей;

- слабые информационные связи с другими программными модулями;

- обмен информацией между модулями должен быть по возможности минимизирован;
- обозримый по размеру и сложности программный элемент.

Следующим шагом после этапа программирования является отладка комплекса программ, которая заключается в поиске и устранении неисправностей.

Проверка работоспособности системы предполагает наличие средств, способных установить соответствие параметров и характеристик данной системы требованиям ТЗ. Целью контроля работоспособности сложной системы является выявление фактов отказов или сбоев, анализа данных, характеризующих работоспособность системы. Кроме контроля работоспособности, применяются методы диагностического контроля, основное назначение которых состоит в локализации ошибок, установлении их характера и причин.

Отладка и диагностика должны начинаться с контроля отдельных подпрограмм и взаимодействующих групп в порядке последовательного усложнения и увеличения числа функционирующих компонент. Самым сложным являются задачи диагностики - обнаружения и локализации ошибок.

Программные ошибки по количеству и типам определяются степенью автоматизации программирования и глубиной автоматического контроля записи программ. Устранение таких ошибок начинается, как правило, в процессе ручной автономной отладки без применения ЭВМ. Проверка и корректировка осуществляется по тексту программы с использованием блок - схем алгоритма до начала трансляции программы. При ручной автономной отладке решаются следующие задачи:

- проверяется запись программы на соответствие синтаксическим и семантическим правилам;
- производится проверка логики решения задачи данной программой на соответствие ТЗ и блок-схемы алгоритма;
- проверяется последовательность преобразования информации;

- уточняется содержание и структура входной и выходной информации;
- осуществляется подготовка и предварительная проверка программ по тестам, используемым впоследствии при отладке ЭВМ.

Несмотря на кажущуюся трудоемкость, ручная отладка весьма рентабельна, т.к. позволяет выявить наиболее грубые алгоритмические и системные ошибки, которые трудно обнаружить формализованными методами на ЭВМ.

После завершения ручной отладки программа транслируется на язык машинных кодов. Программа, в которой выявлены и устранены все ошибки записи, передается на автоматический контроль соответствия формализованным правилам структурного построения программы: на отсутствие зацикливаний, тупиков, лишних участков.

После устранения этих ошибок начинается отладка на ЭВМ с использованием интерпретирующей программы, обеспечивающей исполнение отлаживаемой программы. Метод интерпретации состоит в том, что команды программы, подлежащей исполнению, моделируются командами реализующей ЭВМ в момент ее исполнения. С точки зрения доступности для контроля, простоты организации слежения и управления процессом исполнения программ метод интерпретации имеет преимущество перед методом компиляции.

Алгоритмические ошибки труднее поддаются обнаружению методами формального автоматического контроля, чем предыдущий тип ошибок. К алгоритмическим относятся ошибки, обусловленные некорректной постановкой функциональных задач, когда не полностью оговорены условия, необходимые для получения правильного результата. Эти условия формируются и уточняются в процессе тестирования и выявления ошибок в результатах функционирования алгоритмов. К алгоритмическим относятся ошибки сопряжения функциональных подпрограмм. Процесс обнаружения и выявления алгоритмических ошибок во многих случаях сливается с процессом уточнения

постановки задачи и является, по существу, частью процесса разработки алгоритмов.

Системные ошибки в сложных комплексах программ определяются неполной информацией о реальных процессах, происходящих в источниках и потребителях информации. Эти процессы иногда зависят от алгоритмов управления и поэтому не могут быть определены без исследования функционирования комплекса программ во взаимодействии с заказчиком. На начальных стадиях проектирования не всегда удается сформулировать целевую функцию всей системы и отдельных подсистем. В соответствии с этим в процессе разработки уточняются и конкретизируются технические задания на отдельные подпрограммы и выявляются отклонения от уточненного задания, которые могут квалифицироваться как системные ошибки. Завершающей стадией комплексной отладки программ является испытание программ в реальных условиях и демонстрация заказчику полноты выполнения требований ТЗ.

## Заключение

В ходе выполнения выпускной квалификационной работы было разработано программное обеспечение, реализующее помехоустойчивое кодирование информации в малогабаритной аппаратуре передачи данных.

В процессе работы была обоснована актуальность разработки программного обеспечения, реализующего функции помехоустойчивого кодирования информации в малогабаритном устройстве передачи данных, определены основные методы построения помехоустойчивой защиты.

Осуществлён выбор и произведено обоснование каскадного кода. Следующим шагом являлось произведение расчёта длины кодовой комбинации внутреннего кода.

Разработаны алгоритмы отдельных функций программного обеспечения:

- алгоритм кодирования кодом Рида-Соломона;
- алгоритм приёма информации;
- алгоритм декодирования кодом Рида-Соломона;
- алгоритм вычисления коэффициентов многочлена локаторов стираний;
- алгоритм вычисления модифицированных синдромов

Разработано ПО для модуля помехоустойчивого кодирования.

Разработаны организационные и технические мероприятия для снижения воздействия психофизических факторов, влияющих на психоэмоциональное состояние специалиста по защите информации

Рассмотрена технология разработки программного обеспечения малогабаритных устройств передачи данных

Результатом выпускной квалификационной работы является программное обеспечение, реализующее помехоустойчивое кодирование информации в малогабаритной аппаратуре передачи данных.

Таким образом, задание на выпускную квалификационную работу выполнено в полном объеме.

## Список используемых источников

1. Форни Д. Каскадные коды. – М.: Мир, 1970. – 230с.
2. Блейхут Р. Теория и практика кодов, контролирующих ошибки: Пер. с англ. – М.: Мир, 1986. – 576 с.
3. Кодирование и передача информации в вычислительных сетях. Под редакцией С. И. Самойленко.//Вопросы кибернетики. – 1977. – Вып.28. – 168 с.
4. Ю. Г. Сибаров, Н. Н. Сколотнев, В. К. Васин, В. Н. Нагинаев. – М.: Машиностроение, 1990. – 192 с.: ил.
5. Методы и средства криптографической защиты информации: Учебное пособие / Жданов О.Н., Золотарев В.В., Красноярск: СГАУ им. М.Ф. Решетнева, 2007. – 72 с.
6. . ГОСТ Р 52459.1-2009. Совместимость технических средств электромагнитная. Технические средства радиосвязи. Часть 1. Общие технические требования и методы испытаний. – Введ. 01-01-2010. – М.: Стандартиформ, 2010. – 46 с.
7. Основы схемотехники аналого-цифровых устройств: Учебное пособие по курсу «Схемотехника ЭВМ» О. Е. Аверченков, Смоленск ГОУВПО «МЭИ (ТУ)», 2012 — 80 с.

Приложение А Текст программного обеспечения модуля помехоустойчивого  
кодирования

```

int i,j,m,n,k,ii,l,st_m,st,a,b,c;
int inf,sind,kor,st_um,pr;
int pp;
int pols,pols1,pols_t;           //посл.коэф-т мн-на лок-ров
искажений
int s_is1;                       //счетчик количества искажений -
эталон
int s_is;                        //счетчик количества
искажений
int buf;
int x_kv;                       //квадрат степени локатора
искажений
int s_o,s_ol;                   //счетчик ошибок
int a_1[4],a_2[4],a_3[4];       //строки матрицы
int g_1,g_2,g_3;               //коэфф-ты мн-на локаторов
ошибок
int z1,z2,y1,y2,y3,x1,x2,x3;    //корни мн-на локаторов
ошибок
int zn;                         //знаменатель при вычислении корней мн-на
локаторов ошибок
int ch;                         //числитель при вычислении значений
искажений
int e,e_ch;                    //значение искажения (степень,число)
int p_2o;                      //признак 2-х ошибок

int t_sin[10];                 //таблица синдромов (числа)
int mn_li[55];                 //коэф-ты мн-на лок-ров искажений
(степени)
int t_pr[10];                  //таблица производных (числа)
int t_ms[6];                   //модифицир. синдромы (числа)
int t_zn_is[10];               //таблица чисел значений искажений

int p_dec = 1;                 //признак декодирования = 1
int s_st = 4;                  //счетчик стираний
int lok_o = 4;                 //адрес первого локатора ошибок

```



```

        fdec_rc(1);
        if (otkaz != 0)      return;
    }          //конец
//=====
void    fdec_rc(pp)

{
    for (i=0; i<16; ++i)
        t_sin[i] = t_sin_c[i];          //перепись синдромов
//===== Вычисление синдромов =====
    for (i=0; i<10; ++i)
        {
            if (pp!=0)      { m = inf_mas[0]; m >>= 6;}
            else      {      m = inf_mas[0] & 0x3f;}
            for (j=1; j<24; ++j)
                { st_m = t_log[m] + t_sin[i];          //степень SINi*Li
                  if (pp!=0)      {inf = inf_mas[j]; inf >>= 6;}

//2-я полоса

                        else      {inf = inf_mas[j] & 0x3f;}

//1-я полоса
                        m = t_anlog[st_m] ^ inf;
                }
            t_sin[i] = m;          //синдром
        }
//-----
    if (p_dec != 0)
        {
            if (pp == 0)
                {
                    ii = 1;
                    s_is = s_is1;
                    a = 0;
                    mn_li[0] = t_li[0];
                    fmn_li();          //выч. коэф. мн. локаторов
стираний(искажений)
                    pols_t = j+a;
                    pols1 = pols;
                    buf = a;
                }
        }

```

```

//===== Вычисление модифицированных синдромов =====
// T0
ms:  a = 0;
      for (i=0; i < 16-s_st; ++i)
        {
          ch = 0;
          for (j=0; j<s_st; ++j)
            { st = t_log[t_sin[j+a]]+ mn_li[pols+s_st-
j]; //ст.SINj * Ls-i
              ch ^= t_anlog[st];
            }
          ch ^= t_sin[j+a];
          //числитель
          t_ms[i] = ch;
          a += 1;
        }

//-----

if (s_is == 15 || s_is == 16)
  { //Сч. стираний =15,=16
    s_o1 = s_o = 1;
    x1 = t_log[t_ms[1]] - t_log[t_ms[0]];
    if (x1 < 0)
      {x1 += 63;}
    if (x1 > 23 || (t_ms[0] == 0 && t_ms[1] != 0))
      {   otkaz = 0xaaaa;   return;}
    if (t_ms[0] == 0 || t_ms[1] == 0)
      {   s_o1 = s_o = 0; goto z;} //Вычисление
производных
    if (t_ms[0] != 0 || t_ms[1] != 0)
      {t_li[lok_o] = x1;}
  }

if (s_is <= 14 && s_is != 12) //Сч. стираний =14,=13
  {
    s_o1 = s_o = 2;
    a_1[0] = 0;    //заполнение матрицы
    a_1[1] = 0;
  }

```

```

        a_1[2] = 0;
        a_1[3] = 0;
        a_2[0] = 0;
        a_2[1] = t_ms[0];
        a_2[2] = t_ms[1];
        a_2[3] = t_ms[2];
        a_3[0] = 0;
        a_3[1] = t_ms[1];
        a_3[2] = t_ms[2];
        a_3[3] = t_ms[3];
        goto met2;
    }
    if (s_is <= 10) //Сч.стираний
        <=10
{
    s_o1 = s_o = 3;
    //===== Вычисление коэфф. многочлена локаторов ошибок =====

aa:
        a_1[0] = t_ms[0]; //заполнение
матрицы

        a_1[1] = t_ms[1];
        a_1[2] = t_ms[2];
...
    a_1[3] = t_log[a_1[3]] - t_log[a_1[0]]+ 63;
    a_1[3] = t_anlog[a_1[3]];
        goto met2;
    }
    if (a_2[0] == 0 && a_3[0] == 0)
        { s_o1 = s_o = 0; goto z;}
    //Вычисление производных
    }//else
if (a_2[0] != 0)
{
    a_2[1] ^= t_anlog[t_log[a_1[1]] + t_log[a_2[0]]];
    a_2[2] ^= t_anlog[t_log[a_1[2]] + t_log[a_2[0]]];
    a_2[3] ^= t_anlog[t_log[a_1[3]] + t_log[a_2[0]]];
}

```

```

met1:          if (a_3[0] != 0)
                {
                    a_3[1] ^= t_anlog[t_log[a_1[1]] + t_log[a_3[0]]];
                    a_3[2] ^= t_anlog[t_log[a_1[2]] + t_log[a_3[0]]];
                    a_3[3] ^= t_anlog[t_log[a_1[3]] + t_log[a_3[0]]];
                }
met2:          if (a_2[1] != 0)
                {
                    a_2[2] = t_log[a_2[2]] - t_log[a_2[1]] + 63;
                    a_2[2] = t_anlog[a_2[2]];
                    a_2[3] = t_log[a_2[3]] - t_log[a_2[1]] + 63;
                    ...
                    p_2o = 1;
                    goto met4;
                }

                if (s_is==5 || s_is==6) {s_o = 0; goto z;} //Вычисление
производных
                else          { otkaz = 0xaaaa; return; }
                }//else
met3:          if (a_1[1] != 0)
                {
                    a_1[2] ^= t_anlog[t_log[a_2[2]] + t_log[a_1[1]]];
                    a_1[3] ^= t_anlog[t_log[a_2[3]] +
t_log[a_1[1]]];
                }
                if (a_3[1] != 0)
                    a_3[2] ^= t_anlog[t_log[a_2[2]] +
t_log[a_3[1]]];
                    a_3[3] ^= t_anlog[t_log[a_2[3]] +
t_log[a_3[1]]];
                }
met4:          if (a_3[2] != 0)
                {
                    a_3[3] = t_log[a_3[3]] - t_log[a_3[2]] + 63;
                    a_3[3] = t_anlog[a_3[3]];
                }
                else
                    {
                        if (a_3[3] != 0)          { otkaz =
0xaaaa; return;}

```

```

...
//===== Вычисление локаторов ошибок=====

g:          ch = t_anlog[g_1 + g_2] ^ t_anlog[g_3];
          zn = t_anlog[g_1 + g_1] ^ t_anlog[g_2];

          if (zn == 0)
              {
                  y1 = t_del3[ch];
                  y2 = y1 + 21;
                  if (y2 >= 63) { y2 -= 63;}
                  y3 = y1 + 42;
                  if (y3 >= 63) { y3 -= 63;}
                  goto zap;
              }
          ...
          s_is1 = s_is = s_st = 10;
          goto zn;          //переход к вычислению значений
искажений
              }

z:  fpr ();                      //Вычисление
производных

//===== Вычисление значений искажений =====

zn:  if (s_is == 1) //если 1 стирание
      {
          if (pp == 0)
              {
                  t_zn_is[0] = t_sin[0];
                  return;
              }
          else
              {
                  e_ch = t_sin[0]; e_ch <= 6;
                  t_zn_is[0] ^= e_ch;
                  return;
              }
      }

```

```

//-----
        for (i=0; i<s_isl-1; ++i)
        {
            s_is = a = b = s_is -1;
            ...
        }
        goto i_o;          //Исправление искажений
    }
//===== Корректировка синдромов =====

        t_sin[0] ^= t_anlog[e];          // число
значения искажения
        kor = e;
        for (k=1; k<b; ++k)              //      k
        {
            ...
            //корректировка синдромов
        }
        if (s_o != 0) {s_o -= 1;}
    }
//===== Исправление искажений =====

i_o:    for (i=0; i<s_isl; ++i)
        {
            inf_mas[23 - t_li[s_isl-i-1]] ^= t_zn_is[i];
            t_zn_is[i] ^= t_zn_is[i];
        }
    }

//+++++
//===== Вычисление коэфф. многочлена локаторов искажений =====
void    fmn_li()
        {
            if (s_is != 1)
            {
                for (i=ii; i<s_is; ++i)
            {
                st_um = t_li[i];          //локатор стирания
                ...
            }
        }
//=====

```

```

//===== Вычисление производных =====
void    fpr()
{
    a = 1;
    b = 2;
    t_pr[0] = 0;                                     //кол-во
производных
    t_pr[1] = mn_li[1]; //коэфф. мн.лок.стираний (L1)
    for (i=2; i<s_is; ++i)
{
    x_kv = t_li[i] + t_li[i];                       //квадрат локатора
        if ( (b & 1) != 0 )
        { a += 2; m = mn_li[a];}
        else
        {m = 0; a += 1;}
        c = b >> 1;
        for (j=0; j<c; ++j)
        {
            a += 2;
            l = m + x_kv;
            pr = t_anlog[l] ^ t_anlog[mn_li[a]];
            m = t_log[pr];
        }
        t_pr[i] = t_log[pr];
        b += 1;
    }
}

//=====

```

## Приложение Б Результат тестирования

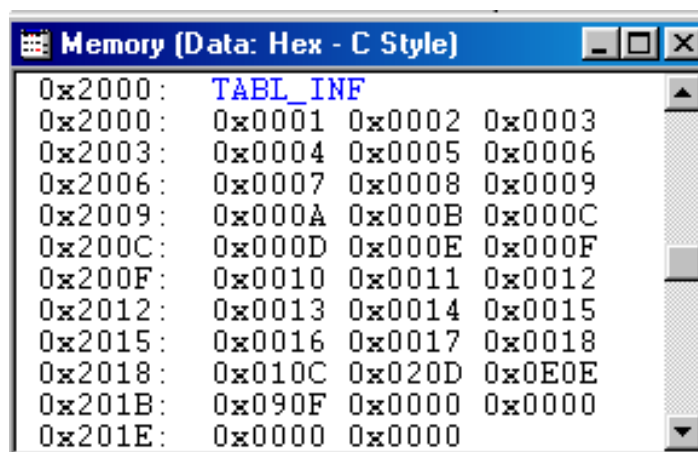


Рисунок Б1 – Входные данные

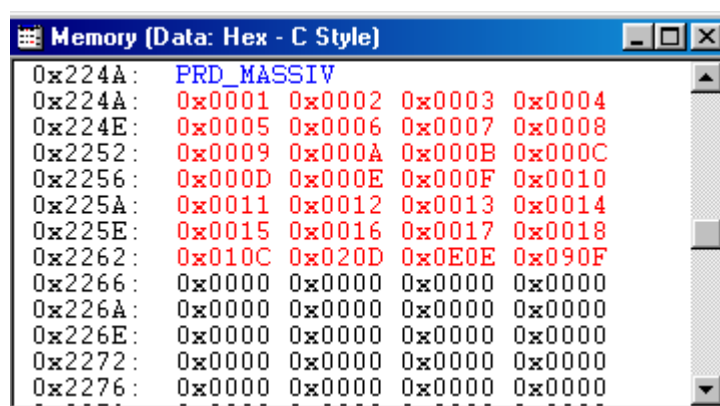


Рисунок Б2 – Закодированные данные

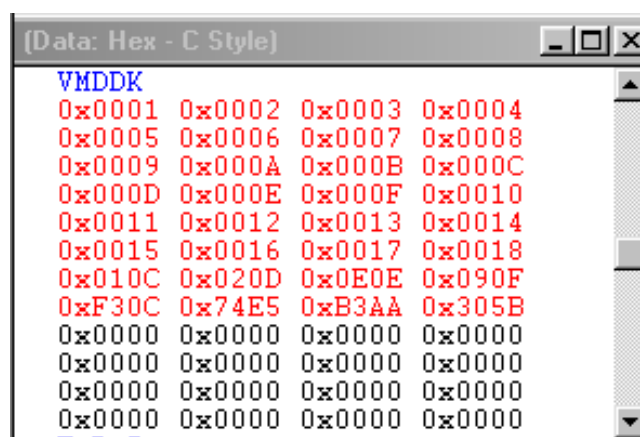


Рисунок Б3 – Декодированные данные