# **Аннотация**

Тема: Передача информации по каналу с решающей обратной связью.

Данная выпускная работа посвящена использованию помехоустойчивых кодов, в частности, циклического кода, у которого кодовое расстояние равно 4.

В работе излагаются этапы и результаты разработки программы, реализующей кодирование циклическим кодом (14,9), приводятся анализ поставленной задачи, ход разработки, результаты решения. Также разработаны структурная, функциональная и принципиальная схемы, реализующие кодирование, декодирование, исправление и обнаружение ошибок циклическим кодом (14,9).

**Содержание**

Аннотация 1

Введение 3

1. Постановка задачи 5

1.1 Анализ технического задания 5

1.2 Теоретическое введение 6

1.2.1 Способы передачи информации по каналам связи 6

1.2.2 Основные понятия о помехозащищенном кодировании 12

1.2.3 Помехозащищенные коды. Циклический код 18

1.2.4 Методы построения циклических кодов 19

1.2.5 Методы декодирования циклических кодов и обнаружения ошибок 33

1.3 Математическая модель 35

1.4 Построение образующей матрицы 36

1.5 Расчет достоверности передаваемых сообщений 38

1.6 Выводы 40

2. Техническая реализация кодера, декодера и решателей 41

2.1 Модульная структура кодера и его работа 41

2.2 Модульная структура декодера и его работа 43

2.3 Модульная структура решателя кодера и его работа 46

2.4 Модульная структура решателя декодера и его работа 48

2.5 Выбор микросхем для реализации кодера, декодера и решателей 49

2.6 Описание функциональной схемы кодера и решателя кодера 56

2.7 Описание функциональной схемы декодера и решателя декодера 57

2.8 Описание принципиальной схемы кодера и решателя кодера 58

2.9 Описание принципиальной схемы декодера и решателя декодера 61

2.10 Выводы 62

3. Описание программных средств, разработанных в ходе реализации проекта 64

3.1 Структура системы 64

3.2 Входные данные, форма представления результатов 64

3.3 Спецификация на программу в целом. 64

3.4 Системные требования 66

3.5 Спецификация на программу в целом. 66

4. Результативная часть 67

4.1 Тестирование 67

4.2 Описание пользовательского интерфейса 68

4.3 Инструкция пользователю. 68

4.4 Выводы 68

5. Заключение 69

Текст программы на языке VHDL для решателя декодера 70

Документированный текст программы 71

# **Введение**

Деятельность людей связана с переработкой и использованием материалов, энергии и информации. Соответственно развивались научные технические дисциплины, отражающие вопросы технологии, энергетики и информатики. Информационная техника является сравнительно новой отраслью, получающей наибольшее развитие на этапе разработки и применения электронных вычислительных машин (ЭВМ) и автоматизированных систем управления (АСУ).

Информационная наука теперь находит применение в самых разнообразных областях теории и практики. Центральной ветвью является теория связи, созданная Шенноном на основе теории вероятностей.

С передачей и обработкой информации связаны действия любого автоматического устройства, поведение живого существа, творческая деятельность человека, развитие науки и техники, экономические и социальные преобразования в обществе и сама жизнь. Для более эффективного использования информации необходимо обмениваться информацией, что невозможно без её передачи по каналам связи.

При передаче информации по каналам связи может происходить искажение передаваемой информации. Для предотвращения потерь полезной информации существуют различные методы защиты. Одним из них является кодирование информации при помощи помехозащищённых кодов.

Двоичный код на все комбинации не является помехозащищённым, так как его комбинации отличаются друг от друга лишь в одном разряде, что не позволяет на приёмной стороне обнаружить и исправить возникшие ошибки. В этой связи возникает необходимость построения помехозащищённого кода.

Помехозащищённые коды – это коды, которые позволяют обнаруживать и исправлять ошибки, то есть корректировать полученные сообщения. Для достижения помехозащищенности можно ввести избыточность добавлением дополнительных контрольных разрядов.

Существует много различных алгоритмов построения помехозащищённых кодов. В данной работе рассматривается код (14,9). Он относится к группе циклических кодов. А именно, он относится к циклическим кодам, обнаруживающим 1 и исправляющим 1 ошибку.

Циклические коды являются основным классом групповых помехоустойчивых кодов и используются для исправления и обнаружения ошибок, возникающих при передаче информации по каналу связи. Устройства, обнаруживающие и исправляющие ошибки, построенные на основе циклического кода, часто применяются в различных информационных системах.

# **1. Постановка задачи**

## 

## 1.1 Анализ технического задания

В соответствии с техническим заданием требуется построить математическую модель циклического кода с кодовым расстоянием 4, для количества букв алфавита сообщений 256. Требуется рассчитать длину кода и его корректирующую способность, найти образующую матрицу кода.

Разработать программу, реализующую кодирование и декодирование разработанным кодом.

Реализовать схему для его кодирования и декодирования на уровне функциональной и принципиальной схемы.

Также необходимо определить функции решателей, включенных в канал обратной связи, разработать функциональную и принципиальную схему решателей.

Расчетно-пояснительная записка содержит:

Введение;

Содержание;.

1. Постановочную часть;

2. Разработку схемы кодирующего, декодирующего и решающих устройств;

3.Описание разработанной программы;

4.Результативную часть;

5.Выводы и заключение;

6.Список литературы;

7.Приложения;

В записке содержатся следующие приложения:

1. Структурную схему передачи данных с РОС;
2. Функциональная схема кодера и решателя кодера;
3. Функциональная схема декодера и решателя декодера;
4. Принципиальная схема кодера и решателя кодера
5. Перечень элементов;
6. Принципиальная декодера и решателя декодера;
7. Перечень элементов;
8. Документированный текст программы решателя декодера;
9. Временные диаграммы;
10. Документированный текст программы;
11. Экранные формы;
12. Техническое задание

## 

## 1.2 Теоретическое введение

### 

### 1.2.1 Способы передачи информации по каналам связи

*Передача информации с повторением (накоплением)*. Такой метод передачи применяют для повышения достоверности при отсутствии обратного канала, хотя нет принципиальных ограничений для его использования и при наличии обратной связи. Иногда такой метод классифицируют как прием сообщений с накоплением. Сущность метода заключается в передаче одного и того же сообщения несколько раз, запоминании принятых сообщений, сравнении их поэлементно и составлении сообщения, включая элементы, выбранные «по большинству». Предположим, что трижды передана одна и та же кодовая комбинация 1010101. Во всех трех передачах она подверглась воздействию помех и была искажена:

1000100

1111101

1010001

1010101

Приемник поразрядно сравнивает три принятых символа и проставляет те символы (под чертой), количество которых в данном разряде преобладает.

Существует и другой метод передачи информации с накоплением, при котором производится не посимвольное сравнение, а сравнение всей комбинации в целом. Этот метод проще реализуется, но обеспечивает более плохие результаты.

Таким образом, высокая помехоустойчивость метода передачи информации с повторением (накоплением) основана на том, что сигнал и помехи в канале не зависят друг от друга и изменяются по разным законам (сигнал периодичен, а помеха случайна), поэтому повторяющаяся комбинация в каждой передаче, как правило, будет искажаться по-разному. Вследствие этого на приеме накопление, то есть суммирование сигнала, возрастает пропорционально числу повторений, тогда как сумма помехи возрастает по другому закону. Если считать, что помехи и сигнал независимы, то суммируются средн-ие квадраты и средний квадрат суммы возрастает пропорционально первойстепени. Поэтому при n повторениях отношение сигнал/помеха увеличивается в n раз, причем это происходит без увеличения мощности сигнала. Однако это достигается за счет усложнения аппаратуры и возрастания времени передачи или полосы частот в случае, если сигнал передается на нескольких частотах одновременно во времени. Кроме того, при зависимых ошибках и пачках ошибок помехоустойчивость системы снижается.

*Передача информации с обратной связью*. Помехоустойчивость передачи без обратной связи (ПБОС) обеспечивается следующими способами: помехоустойчивым кодированием, передачей с повторением, одновременной передачей по нескольким параллельным каналам. В ПБОС применяются обычно коды с исправлением ошибок, что связано с высокой избыточностью и усложнением аппаратуры. Передача с обратной связью (ПОС) во многом устраняет указанные недостатки, так как позволяет применять менее помехоустойчивые коды, обладающие, как правило, меньшей избыточностью. В частности, можно использовать коды с обнаружением ошибок. Преимуществом обратного канала является также возможность контроля работоспособности объекта, принимающего информацию.

При ПОС вводят понятие прямого канала, т.е. канала от передатчика к приемнику, например передается сигнал команды с пункта управления (ПУ) на контролируемый пункт (КП). Обратным каналом при этом явится передача сообщения с КП на ПУ о принятии сигнала команды, причем по обратному каналу могут передаваться как сообщение только о том, что сигнал принят на входе КП (в этом случае контролируется лишь прохождение сигнала по каналу связи), так и сведения о полном выполнении команды. Возможна и обратная связь, дающая сведения о поэтапном прохождении сигнала команды по тракту приема.

Рассмотрим отдельные виды передачи с обратной связью.

*Передача с информационной обратной связью (ИОС)*. Если сообщение передается в виде непомехозащищенного кода, то в кодирующем устройстве данный код может быть преобразован в помехозащищенный. Однако, поскольку в этом обычно нет необходимости, кодирующее устройство представляет собой регистр для превращения простого параллельного кода в последовательный. Одновременно c передачей по прямому каналу сообщение запоминается в накопителе на передатчике (рис.1.1а). На контролируемом пункте принятое сообщение декодируется и также запоминается в накопителе. Однако получателю сообщение передается не сразу: сначала оно поступает через обратный канал на пункт управления. В схеме сравнения ПУ происходит сравнение принятого сообщения с переданным. Если сообщения совпадают, то формируется сигнал «Подтверждение» и происходит передача последующих сообщений (иногда перед посылкой последующего сообщения на КП сначала посылается сигнал «Подтверждение» о том, что предыдущее сообщение было принято верно и с накопителя можно передать информацию получателю). При несовпадении сообщений, что свидетельствует об ошибке, формируется сигнал «Стирание». Этот сигнал запирает ключ для прекращения передачи очередного сообщения и посылается на КП для уничтожения записанного в накопителе сообщения. После этого с ПУ производится повторная передача сообщения, записанного в накопителе.

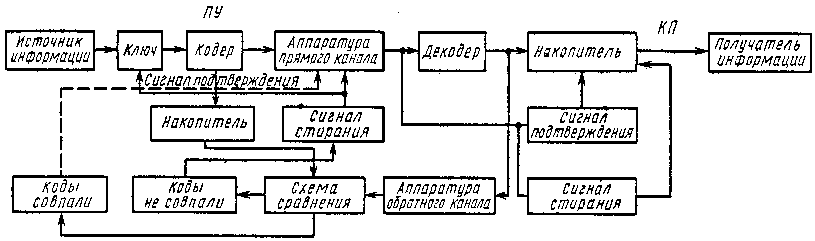


Рис.1.1а. Способ передачи информации с ИОС.

В системах с ИОС ведущая роль принадлежит передающей части, так как она определяет наличие ошибки, приемник только информирует передатчик о том, какое сообщение им получено. Имеются различные варианты передачи с ИОС. Так, существуют системы с ИОС, в которых передача сигналов происходит непрерывно и прекращается лишь при обнаружении ошибки: передатчик посылает сигнал «Стирание» и повторяет передачу. Системы с ИОС, в которых по обратному каналу передается вся информация, переданная на КП, называются системами с ретрансляционной обратной связью. В некоторых системах с ИОС передается не вся информация, а только некоторые характерные сведения о ней (квитанции). Например, по прямому каналу передаются информационные, а по обратному каналу — контрольные символы, которые будут сравниваться на передатчике с предварительно записанными контрольными символами. Имеется вариант, в котором после проверки принятого по обратному каналу сообщения и обнаружения ошибки передатчик может либо повторить его (дублирование сообщения), либо послать дополнительную информацию, необходимую для исправления (корректирующая информация). Число повторений может быть ограниченным или неограниченным.

Обратный канал используют для того, чтобы определить, необходима ли повторная передача информации. В системах с ИОС повышение достоверности передачи достигается путем повторения информации только при наличии ошибки, тогда как в системах без обратной связи (при передаче с накоплением) повторение осуществляется независимо от искажения сообщения. Поэтому в системах с ИОС избыточность информации значительно меньше, чем в системах с ПБОС: она минимальна при отсутствии искажений и увеличивается при ошибках. В системах с ИОС качество обратного канала должно быть не хуже качества прямого во избежание искажений, которые могут увеличить число повторений.

*Передача с решающей обратной связью (РОС)*. Переданное с передатчика по прямому каналу сообщение принимается на приемнике (рис.1.1б), где оно запоминается и проверяется в декодирующем устройстве (декодере). Если ошибок нет, то из накопителя сообщение поступает к получателю информации, а через обратный канал на передатчик подается сигнал о продолжении дальнейшей передачи (сигнал продолжения). Если ошибка обнаружена, то декодер выдает сигнал, стирающий информацию в накопителе. Получателю сообщение не поступает, а через обратный канал на передатчик подается сигнал о переспросе или повторении передачи (сигнал повторения или переспроса). На передатчике сигнал повторения (иногда называемый решающим сигналом) выделяется приемником решающих сигналов, а переключающее устройство отключает вход кодера от источника информации и подключает его к накопителю, что позволяет повторить переданное сообщение. Повторение сообщения может происходить несколько раз до его правильного приема.

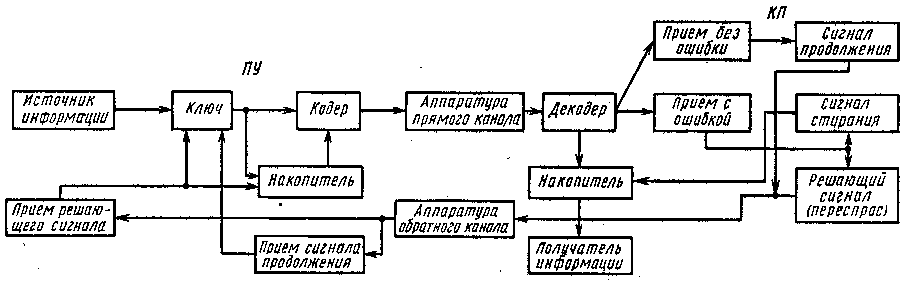


Рис.1.1б. Способ передачи информации с РОС.

При передаче с РОС ошибка определяется приемником. Для этого передаваемое сообщение должно кодироваться обязательно помехозащищенным кодом, что позволяет приемнику выделить разрешенную комбинацию (сообщение) из неразрешенных. Это означает, что передача с РОС осуществляется с избыточностью. Достоверность передачи в системах РОС определяется выбором кода и защитой решающих сигналов повторения и продолжения. Последнее не представляет особых трудностей, так как эти сигналы несут одну двоичную единицу информации и могут передаваться достаточно помехоустойчивым кодом.

Системы с РОС, или системы с переспросом, подразделяют на системы с ожиданием решающего сигнала и системы с непрерывной передачей информации.

В системах с ожиданием передача новой кодовой комбинации или повторение переданной происходит только после поступления на передатчик сигнала запроса.

В системах с непрерывной передачей происходит непрерывная передача информации без ожидания сигнала запроса. Скорость передачи при этом выше, чем в системах с ожиданием. Однако после обнаружения ошибки по обратному каналу посылается сигнал переспроса и за время прихода на передатчик с последнего уже будет передано какое-то новое сообщение. Поэтому системы с непрерывной передачей необходимо усложнять соответствующей блокировкой приемника, чтобы он не принимал информацию после обнаружения ошибки.

Для сравнения эффективности системы без обратной связи, в которой применяется код Хэмминга с исправлением одной ошибки, и системы с РОС, использующей простой код, вводят понятие коэффициента эффективности. Этот коэффициент учитывает уменьшение вероятности ошибочного приема и затраты на его достижение, выигрыш в защите от ошибок (в случае применения указанных кодов), относительное снижение скорости передачи и схемную избыточность, связанные с использованием разных кодов. Итоговое сравнение показало, что в отличие от системы без обратной связи, использующей сложный код, система с РОС дает выигрыш в 5,1 раза. Высокая эффективность систем с РОС обеспечила их широкое распространение.

Сравнительный анализ достоверности передачи систем с ИОС и РОС, показал, что:

1) системы с ИОС и РОС обеспечивают одинаковую достоверность передачи при одинаковых суммарных затратах энергии сигналов в прямом и обратном каналах при условии, что эти каналы симметричны и имеют одинаковый уровень помех;

2) системы с ИОС обеспечивают более высокую достоверность передачи, чем Системы с РОС при относительно слабых помехах в обратном канале в отличие от прямого. При отсутствии помех в обратном канале системы с ИОС обеспечивают безошибочную передачу сообщений по основному каналу;

3) при сильных помехах в обратном канале более высокую достоверность обеспечивают системы с РОС;

4) при пачках ошибок в прямом и обратном каналах более высокую достоверность обеспечивают системы с ИОС.

### 1.2.2 Основные понятия о помехозащищенном кодировании

Помехозащищенными (или корректирующими) называются коды, позволяющие обнаружить и исправить ошибки в кодовых комбинациях. Отсюда и деление этих кодов на две большие группы: 1) коды с обнаружением ошибок; 2) коды с обнаружением и исправлением ошибок.

Принципы обнаружения и исправления ошибок кодами хорошо иллюстрируются с помощью геометрических моделей. Любой n-элементный двоичный код можно представить n-мерным кубом (рис.1.2), в котором каждая вершина отображает кодовую комбинацию, а длина ребра куба соответствует одной единице. В таком кубе расстояние между вершинами (кодовыми комбинациями) измеряется минимальным количеством ребер, находящихся между ними, обозначается d и называется кодовым расстоянием Хэмминга.

Таким образом, кодовое расстояние — это минимальное число элементов, в которых любая кодовая комбинация отличается от другой (по всем парам кодовых слов). Например, код состоит из комбинаций 1011, 1101, 1000 и 1100. Сравнивая первые две комбинации, путем сложения их по модулю 2 находим, что d=2. Сравнение первой и третьей комбинаций показывает, что и в этом случае d=2. Наибольшее значение d=3 обнаруживается при сравнении первой и четвертой комбинаций, а наименьшее d=1 — второй и четвертой, третьей и четвертой комбинаций. Таким образом, для данного кода минимум расстояния dmin=l.

При n=1 n-мерный куб превращается в прямую длиной d=1, на одном конце которой располагается 1, а на другом — 0. При n = 2 четыре возможные комбинации (N=22=4) располагаются на четырех вершинах квадрата. При этом комбинации 00 и 11, а также 10 и 01 отличаются друг от друга в двух разрядах, т.е. d=2.

Кодовое расстояние между двумя комбинациями двоичного кода равно числу единиц, полученных при сложении этих комбинаций по модулю 2, например 1001=11 и 0011=11. Такое определение кодового расстояния удобно при большой разрядности кодов. Так, складывая комбинации

10110101101



10000000010

00110101111,

определяем, что кодовое расстояние между ними d = 7.

При n = 3 восемь кодовых комбинаций размещаются в вершинах трехмерного куба.

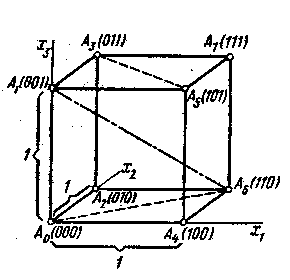


Рис.1.2. Геометрическая модель двоичных кодов

Трехмерный куб строится так (рис.1.2), что одна из его вершин лежит в начале координат. Каждой вершине куба приписывается кодовая комбинация по следующему правилу: на i-м месте кодовой комбинации ставится 0, если проекция этой вершины на i-ю ось координат равна нулю, и 1, если проекция равна единице. Например, требуется узнать, какую следует записать комбинацию в вершине A6 (рис.1.2). Проецируя эту вершину на ось Х1, получим единицу. На втором месте комбинации запишется также 1 (проекция на ось X2 равна единице). Так как проекция на ось Х3 равна нулю (проекция в начало координат), то на третьем месте комбинации запишется 0. Следовательно, вся комбинация в вершине A6 запишется как 110.

Если использовать все восемь слов, записанных в вершинах куба, то образуется двоичный код на все сочетания. Как было показано, такой код является непомехоустойчивым. Если же уменьшить число используемых комбинаций с восьми до четырех, то появится возможность обнаружения одиночных ошибок. Для этого выберем только такие комбинации, которые отстоят друг от друга на расстояние d = 2, например 000, 110, 011 и 101. Остальные кодовые комбинации не используются. Если будет принята комбинация 100, то очевидно, что при ее приеме произошла одиночная ошибка.

Представленные комбинации построены по определенному правилу, а именно содержат четное число единиц, а принятая комбинация 100 – нечетное. Можно утверждать, что комбинация 100 образовалась при искажении разряда одной из разрешенных комбинаций, но определить, какая именно комбинация искажена, невозможно. Поэтому такие или подобные им коды называют кодами с обнаружением ошибок.

Кроме указанной группы комбинаций в том же трехмерном кубе может быть получена еще одна группа комбинаций с кодовым расстоянием d=2 (111, 001, 010 и 100). В этих кодовых комбинациях нечетное число единиц и каждая из комбинаций могут быть использованы для обнаружения ошибки, возникшей при передаче, так как при одиночном искажении в комбинации будет четное число единиц. Однако, если необходимо получить код с обнаружением одиночной ошибки, в передаче может участвовать только одна группа, т. е. четыре комбинации из возможных восьми. В противном случае получится непомехоустойчивый код, в котором будут встречаться комбинации с d=l.

Таким образом, в помехозащищенных кодах есть комбинации разрешенные, составленные по определенному правилу, и запрещенные, не соответствующие этому правилу.

Так, если из восьми комбинаций трехразрядного кода образованы четыре комбинации, позволяющие обнаружить одиночную ошибку (например, 111, 001, 010 и 100), то эти комбинации являются разрешенными, а остальные четыре (000, 011, 101 и 110) — запрещенными, которые должны фиксироваться на приеме как искаженные. Иногда совокупность разрешенных кодовых комбинаций, которые при заданных возможных искажениях не могут перейти друг в друга, называют системой непереходящих сигналов.

Итак, видно, что построение помехоустойчивого кода (а код с обнаружением ошибки является простейшим типом такого кода) связано с недоиспользованием кодовых комбинаций, приводящим к так называемой избыточности. Избыточность означает, что из исходных символов можно построить больше комбинаций, чем их применено в данном коде.

Таким образом, установлено, что уменьшение числа используемых комбинаций приводит к повышению помехоустойчивости кода. Если идти дальше по этому пути и еще больше ограничить число разрешенных комбинаций, то можно создать код не только с обнаружением, но и с исправлением ошибки.

Выберем в трехмерном кубе такие вершины, кодовые обозначения которых отличались бы друг от друга на d=3. Такие вершины расположены на концах пространственных диагоналей куба. Их может быть только четыре пары: 000 и 111, 001 и 110, 100 и 011, 010 и 101. Однако из этих четырех пар для передачи можно брать только одну любую пару, так как большее число пар приведет к тому, что в передаче будут использоваться комбинации, отличающиеся друг от друга на d<3.

Код, образованный по такому правилу, может исправить одиночную ошибку или обнаружить две ошибки без их исправления. Пусть, например, передается код, состоящий из комбинаций 001 и 110. На приеме получена комбинация 100. Сравнение ее с исходными комбинациями показывает, что от комбинации 110 она отличается в одном (втором) разряде, а от комбинации 001 — в двух разрядах. Если считать, что сделана одна ошибка, то полученную комбинацию 100 следует исправить на 110.

От разрешённой комбинации 001 отличаются нa d=l комбинации 011, 000 и 101, а от комбинации 110 — комбинации 111, 100 и 010. Они и являются своеобразными комбинациями-спутниками, которые после приема можно относить к той или иной исходной комбинации.

Когда говорят об исправлении одиночной ошибки, считают, что вероятность двойной ошибки в канале связи пренебрежимо мала. Если такая вероятность достаточно велика, то код с d = 3 можно использовать для обнаружения двойных ошибок, но при этом исправить одиночную ошибку он уже не может. Действительно, если в нашем примере была принята комбинация 100, то нельзя утверждать, что была передана комбинация 110, так как при двойных ошибках это могла быть и искаженная комбинация 001.

Таким образом, дальнейшее повышение помехоустойчивости кода связано с увеличением кодового расстояния d, что приводит к увеличению избыточности (вместо восьми комбинаций используются только две).

Корректирующая способность кода зависит от кодового расстояния: а) при d=1 ошибка не обнаруживается; б) при d = 2 обнаруживаются одиночные ошибки; в) при d = 3 исправляются одиночные ошибки или обнаруживаются двойные ошибки. В общем случае

d = r + s + 1,

где d — минимальное кодовое расстояние; r — число обнаруживаемых ошибок; s — число исправляемых ошибок.

При этом обязательным условием является r≥s. Так, в нашем примере d = 3, и если r = s = l, то код может обнаружить одну ошибку и исправить ее. Если r =2, s=0, то код может только обнаружить две ошибки. Как следует из уравнения, для исправления одной ошибки и обнаружения двух ошибок необходимо, чтобы d = 2 + 1 + 1 =4. При d=4 может быть также вариант, когда r=3, s=0. Если d=5, то могут быть три варианта: r = s = 2; r=3, s=l; r = 4, s=0.

Если код только обнаруживает ошибки, то

d=r + l, т.е. r = d - 1.

Если код только исправляет ошибки, то

d=2s+1, т.е. s=(d – 1)/2

Итак, геометрические модели позволяют просто строить малоразрядные корректирующие коды. Однако при длине кода n>3 геометрической моделью пользоваться трудно, так как она должна быть многомерной. Поэтому для построения многоразрядных помехоустойчивых кодов используют различные правила и методики, к рассмотрению которых и перейдем.

### 1.2.3 Помехозащищенные коды. Циклический код

Циклические коды относятся к числу блоковых систематических кодов, в которых каждая комбинация кодируется самостоятельно (в виде блока) таким образом, что информационные k и контрольные m символы всегда находятся на определенных местах.

Возможность обнаружения и исправления практически любых ошибок при относительно малой избыточности по сравнению с другими кодами, а также простота схемной реализации аппаратуры кодирования и декодирования сделали эти коды широко распространенными.

Теория циклических кодов базируется на теории групп и алгебре многочленов над полем Галуа.

Многочлен (полином), который можно представить в виде произведения многочленов низших степеней, называют приводимым (в данном поле), в противном случае — неприводимым. Неприводимые многочлены играют роль, сходную с простыми числами в теории чисел. Неприводимые многочлены Р(Х) можно записать в виде десятичных или двоичных чисел либо в виде алгебраического многочлена (табл. 1.1).

**Таблица 1.1 Неприводимые многочлены и их эквиваленты**

|  |  |
| --- | --- |
| Р(Х1)=Х+1→3→11 | Р(Х5)= Х5+Х3+Х2+Х+1→47→101111 |
| Р(Х2)=Х2+Х+1→7→111 | Р(Х5)= Х5+Х4+Х2+Х+1→55→110111 |
| Р(Х3)=Х3+Х+1→11→1011 | Р(Х5)= Х5+Х4+Х3+Х+1→59→111011 |
| Р(Х3)= Х3+Х2+ 1→13→1101 | Р(Х5)= Х5+Х4+Х3+Х2+1→61→111101 |
| Р(Х4)=Х4+Х+1→19→10011 | Р(Х6)= Х6+Х+1→67→1000011 |
| Р(Х4)=Х4+Х3+1→25→11001 | Р(Х7)= Х7+Х3+1→137→10001001 |
| Р(Х4)= Х4+Х3+ Х2+Х+1→31→11111 | Р(Х8)= Х8+Х4+Х3+ Х2+1→285→100011101 |
| Р(Х5)= Х5+Х2+1→37→100101 | Р(Х9)= Х9+Х4+1→1041→1000010001 |
| Р(Х5)= Х5+Х3+1→41→101001 | Р(Х10)= Х10+Х3+ 1→2057→10000001001 |

В табл. 1.1 указаны все неприводимые многочлены до пятой степени; включительно, используемые для построения циклических кодов. Многочлены более высоких степеней приводятся лишь выборочно.

Многочлен в поле двоичных чисел называется неприводимым, если он делится без остатка только на себя или на единицу; что касается многочленов, приведенных в табл. 1.1, то это определение справедливо только для конечного поля двоичных чисел.

В основу циклического кодирования положено использование неприводимого многочлена Р(Х), который применительно к циклическим кодам называется образующим, генераторным или производящим многочленом (полиномом).

### 

### 1.2.4 Методы построения циклических кодов

В качестве информационных символов k для построения циклических кодов берут комбинации двоичного кода на все сочетания. В общем случае, если заданную кодовую комбинацию Q(X) умножить на образующий многочлен Р(Х), получится циклический код, обладающий теми или иными корректирующими свойствами в зависимости от выбора Р(Х). Однако в этом коде контрольные символы m будут располагаться в самых разнообразных местах кодовой комбинации. Такой код не является систематическим, что затрудняет его схемную реализацию. Ситуацию можно значительно упростить, если контрольные символы приписать в конце кода, то есть после информационных символов. Для этой цели целесообразно воспользоваться следующим методом.

1. Умножаем кодовую комбинацию G(X), которую мы хотим закодировать, на одночлен , имеющий ту же степень, что и образующий многочлен Р(Х).

2. Делим произведение  на образующий многочлен Р():

(1.1)

где Q(X) — частное от деления; R(X) — остаток.

Умножая выражение (1.1) на Р(Х) и перенося R(X) в другую часть равенства, согласно правилам алгебры двоичного поля, т. е. без перемены знака на обратный, получаем

 (1.2)

Таким образом, согласно равенству (1.2), циклический код, т. е. закодированное сообщение F(X), можно образовать двумя способами:

1 ) умножением одной из комбинаций двоичного кода на все сочетания [комбинация Q(X) принадлежит к той же группе того же кода, что и заданная комбинация G(X)] на образующий многочлен Р(Х);

2) умножением заданной кодовой комбинации G(X) на одночлен , имеющий ту же степень, что и образующий многочлен Р(Х), с добавлением к этому произведению остатка R(X), полученного после деления произведения  на образующий многочлен Р(Х).

Пример 1.1. Требуется закодировать одну из комбинаций двоичного кода 1101, что соответствует .

Не останавливаясь на выборе образующего многочлена Р(Х), о чем будет сказано далее, возьмем из табл. 1.1 многочлен Р(Х3)=Х3+Х+1→11→1011.

Умножая G(X) на , который имеет третью степень, получим

.

От умножения степень каждого многочлена повысилась, что эквивалентно приписыванию трех нулей к многочлену, выраженному в двоичной форме.

Разделив произведение  на образующий многочлен , согласно (1.1) получим



или в двоичном эквиваленте



Таким образом, в результате деления получаем частное Q(X) той же степени, что и G(X):



и остаток:

.

В итоге комбинация двоичного кода, закодированная циклическим кодом, согласно (1.2) примет вид

F(X)= 1111•1011 = 1101000 + 001 = 1101001.

Действительно, умножение 1111•1011 (первый способ) дает тот же результат, что и сложение 1101000 + 001 (второй способ).

*Циклические коды, обнаруживающие одиночную ошибку (d=2)*. Код, образованный многочленом Р(Х)=Х+1, обнаруживает не только одиночную ошибку, но и любое нечетное число ошибок.

Предположим, что необходимо закодировать сообщение G(Х)=Х3+ +Х2+X+1→1101 с помощью образующего многочлена P(Х)=Х+1→11.

Умножим G(X) на Хm, что эквивалентно добавлению нуля справа, так как m=1, поскольку Р(Х) имеет первую степень: (Х3+Х2+1)X= Х4+Х3+X→11010.

Разделив полученное выражение на Р(Х), найдем, что остаток R(X)=1. Следовательно, кодовый многочлен циклического кода в соответствии с уравнением (1.2) будет иметь вид

F(X)= G(X)Хm + R(X)= Х4 + Х3 +X + 1→ 11010 + 1 = 11011.

Таким образом, в этом закодированном сообщении 11011 n = 5, k=4 и m=1, то есть информационным символом является комбинация 1101, а контрольным — единица в младшем разряде.

Сообщение, которое было закодировано (1101), является одной из 16 комбинаций четырехразрядного кода. Если требуется передать все эти сообщения в закодированном виде, то каждое из них следует кодировать так же, как и комбинацию 1101. Однако проделывать дополнительные 15 расчетов (в общем случае 2k расчетов) нет необходимости. Это можно сделать проще, путем составления образующей (порождающей) матрицы.

Образующая матрица составляется из единичной транспонированной матрицы и матрицы дополнений.

Число столбцов транспонированной единичной матрицы равно числу информационных символов k. Матрица дополнений получается из остатков от деления единицы с нулями на образующий многочлен Р(Х), выраженный в двоичном эквиваленте. Число остатков равно числу информационных символов.

Как следует из результатов деления единицы с нулями на Р(Х)=Х+1→11, все остатки оказываются равными единице. Поэтому образующая матрица имеет вид



Единичная Матрица

транспо дополнений

нирован

ная матрица

Четыре кодовые комбинации, из которых состоит образующая матрица, являются первыми кодовыми комбинациями циклического кода. Пятая комбинация нулевая, а так как в четырехразрядном непомехозащищенном коде всего N=24 =16 комбинаций, то остальные 11 ненулевых комбинаций находят суммированием по модулю 2 всевозможных комбинаций строк образующей матрицы:

5. 000009. 13. 

6. 10. 14. 

7. 11. 15. 

8. 12. 16. 

Сгруппируем полученные комбинации следующим образом:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1. | 00011 | 2. | 00101 | 12. | 01111 |
| 6. | 00110 | 7. | 01010 | 16. | 11110 |
| 9. | 01100 | 10. | 10100 | 13. | 11101 |
| 11. | 11000 | 3. | 01001 | 14. | 11011 |
| 4. | 10001 | 8. | 10010 | 15. | 10111 |

Видно, что в первом столбце от комбинации к комбинации две рядом стоящие единицы сдвигаются на один символ влево, во втором столбце циклически сдвигаются две единицы, не стоящие рядом друг с другом, а в третьем столбце происходит циклический сдвиг четырех единиц. Этот циклический сдвиг одной комбинации по отношению к другой и определил название кодов — циклические.

Заметим, что циклический сдвиг является результатом умножения кодовой комбинации на X. Действительно, вторую комбинацию можно записать как 00101→ Х2 + 1, седьмую — как (Х2 + 1)Х = X3 + X→01010 и т. п. Если при умножении на X степень становится равной Xm+l, то полученный результат нужно разделить на Х+1. Например, если комбинацию 10101→ Х4+ +Х2 + 1 умножить на X, то получим Х5 + Х3 + X. Деля полученное выражение на Х5 + 1, найдем остаток Х3 + Х + 1 → 01011. Многочлен 01011 и является результатом циклического сдвига на один разряд влево многочлена 10101.

Рассмотрение полученных комбинаций показывает, что все они имеют четное число единиц. Действительно, контрольные символы оказываются равными единице при нечетном числе единиц в исходной комбинации и нулю при четном числе единиц. Таким образом, циклический код с обнаружением одиночной ошибки является обычным кодом с четным числом единиц.

Циклические коды с d = З. Эти коды могут обнаруживать одиночные и двойные ошибки или обнаруживать и исправлять одиночные ошибки.

1. Выбор числа контрольных символов. Есть два способа выбора числа m. При первом способе исходят из того, что число контрольных символов m= = n — k зависит от числа информационных символов, а значит, и от длины всей кодовой комбинации. Выбор m производится, как и для кода Хэмминга, с исправлением одиночной ошибки. Условие  может быть записано в виде

(1.3)

где Е" — знак округления в сторону большего значения.

При втором способе число контрольных символов определяется по эмпирической формуле

m=Е" Iog2[(k +1) + E" Iog2 (k + 1)]. (1.4)

В основу выбора m в последнем выражении положено значение числа информационных разрядов. Это удобно, так как первое, что известно в начале кодирования, — именно число разрядов информационных символов. Уравнение (1.4) дает тот же результат, что и (1.3).

Из (1.3) вытекает, что наиболее экономичными являются коды, для которых log2(n +1) выражается целым числом, то есть когда длина кодовой комбинации

n = 2m – 1,(1.5)

где m должно быть целым числом. Так, при k=11, n=15 и m=4 без всяких округлений. Но при k=12, n=17, так как m = 5 выбрано с округлением в сторону большего значения, что увеличивает избыточность кода: в первом случае И=4/15=0,266, во втором И=5/17=0,295.

2. Выбор образующего многочлена Р(Х). Степень образующего многочлена l не может быть меньше числа контрольных символов m. Это значит, что если m=3, то из табл. 1.1 можно выбрать любой образующий многочлен Р(Х) начиная с третьей степени и выше. Для упрощения технической реализации кодирования степень Р(Х) следует выбирать равной числу m, т. е. l=m. Если в таблице имеется ряд многочленов с данной степенью, то из них следует выбрать самый короткий. Однако число ненулевых членов многочлена Р(Х) не должно быть меньше кодового расстояния d.

3. Нахождение элементов дополнительной матрицы. Дополнительную матрицу находят путем деления единицы с нулями на выбранный многочлен R(X) и выписывания всех промежуточных остатков. При этом должны быть соблюдены следующие условия:

а) число остатков должно быть равно числу информационных символов k;

б) для дополнительной матрицы пригодны лишь остатки с весом W, не меньшим числа обнаруживаемых ошибок r, т. е. в данном случае не меньшим 2 (W≥2); так обнаруживается не менее двух ошибок.

Из условий а) и б) определяется количество нулей, приписываемых к единице при делении ее на многочлен Р(Х);

в) так как элементы дополнительной матрицы являются для данной комбинации контрольными символами, то число разрядов дополнительной матрицы равно числу контрольных символов m. Вследствие того, что степень образующего многочлена выбирают равной m, число разрядов дополнительной матрицы равно также степени образующего многочлена. Например, если m=3, а остаток равен 11, то он должен быть записан как 011. Из сказанного вытекает, что разрядность остатка равна степени образующего многочлена.

4. Составление образующей матрицы. Берут транспонированную единичную матрицу и справа приписывают к ней элементы дополнительной матрицы. Пример составления образующей матрицы был дан при рассмотрении циклического кода с обнаружением одиночной ошибки.

5. Нахождение всех комбинаций циклического кода данной группы. Это достигается суммированием по модулю 2 всевозможных сочетаний строк образующей матрицы, как было показано при рассмотрении циклического кода с обнаружением одиночной ошибки.

Пример 1.2. Образовать циклический код, позволяющий обнаруживать двукратные ошибки или исправлять одиночную ошибку из всех комбинаций двоичного кода на все сочетания с числом информационных символов k = 4.

По уравнению (1.4) находим число контрольных символов:

m = Е" Iog2 [(4 + 1 ) + E"log2 (4 + 1 )] = Е" Iog2 (5 + 3)= 3.

Из табл.1.1 выбираем один из образующих многочленов третьей степени. Пусть Р(Х)=Х3 + Х + 1→ 1011. Находим остатки от деления единицы с нулями на Р(Х), которые соответственно равны 011, 110, 111, 101. Остатков должно быть четыре согласно числу информационных символов. Выписывая транспонированную единичную матрицу и приписывая к ней справа матрицу дополнений в виде остатков, получаем образующую матрицу

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| k4 | | k3 | k2 | k1 | m3 | m2 | m1 |
| a1 | 0 | 0 | 0 | 1 | 0 | 1 | 1 |
| a2 | 0 | 0 | 1 | 0 | 1 | 1 | 0 |
| a3 | 0 | 1 | 0 | 0 | 1 | 1 | 1 |
| a4 | 1 | 0 | 0 | 0 | 1 | 0 | 1 |

Так как все члены единичной матрицы являются комбинациями заданного четырехразрядного двоичного кода, то четыре комбинации образующей матрицы представляют собой четыре комбинации требуемого циклического кода. Остальные 11 комбинаций циклического кода (начиная с пятой) могут быть получены путем суммирования по модулю 2 этих четырех комбинаций образующей матрицы так, как было проделано для кода с d=2:

5. 

6. 

7. 

8. 

9. 

10. 

11. 

12. 

13. 

14 

15 

Заметим, что комбинация 13 была получена при выводе уравнения (1.2). Если сложить комбинации , то получим циклический код 1011000, в котором контрольными символами являются одни нули. Нулевая комбинация может быть также использована: у нее все символы — нули.

Как следует из табл. 1.1, в качестве образующего можно было бы взять и многочлен Р(Х)=Х3 + Х2 + 1→ 1101. В этом случае образующая матрица приняла бы вид

a10001101

a20010111

a30100011

a41000110

Многочлен Р(Х)=Х3 + Х + 1→ 1011 называется обратным или двойственным многочленом многочлена Р(Х)=Х3 + Х2 + 1→ 1101. Действительно, сравнивая записанные в двоичной форме выражения обоих многочленов, видим, что нули и единицы в обратном многочлене расположены зеркально относительно основного многочлена, т. е. младший разряд становится старшим. Так, многочлен 1110101 является обратным многочлену 1010111. Двойственный многочлен можно записать в виде

Р\*(Х)=Хn-1Р(Х-1). (1.6)

В нашем примере Р\*(Х)= Х3(Х-3 + Х-2 + 1) = Х3 + Х + 1. Использование двойственных многочленов расширяет возможности построения циклических кодов, так как если Р(Х) — неприводимый многочлен, то и многочлен Р\*(Х) также неприводим.

Циклическое кодирование можно осуществлять не только путем составления образующей матрицы из транспонированной матрицы и матрицы дополнения. Тот же результат достигается, если каждый из членов единичной транспонированной матрицы умножить на образующий многочлен. Так, если образующий многочлен Р(Х)=Х3 + Х + 1→ 1011, то умножение транспонированной единичной матрицы на этот многочлен даст

0001X1011=0001011

0010Х1011=0010110

0100X1011=0101100

1000X1011=1011000

Заметим, что, например, умножение 0100X1011 эквивалентно 1011X X100=101100. Нуль слева (0101100) приписывается для комплектности кода. Результатом умножения явился циклический сдвиг образующего многочлена. Сложением полученных комбинаций можно образовать те же комбинации, что и с помощью двух предыдущих образующих матриц.

Нами был выбран в качестве исходного четырехэлементный двоичный код на все сочетания (k = 4), что позволило образовать 24=16 комбинаций циклического кода. Эти комбинации являются разрешенными, так как после кодирования разрядность кода из-за наличия контрольных символов m = 3 увеличилась до n = 7. Из 128 комбинаций семиразрядного двоичного кода 112 будут неразрешенными. При этом сравнение комбинаций, полученных с помощью образующей матрицы обоими многочленами, показывает, что из 32 комбинаций совпадают только нулевые и составленные из одних единиц.

Таким образом, из двоичного кода на все сочетания (k=4) были образованы два циклических кода с помощью различных образующих многочленов: Р(X)=1011 и P(X)=1101. При этом, несмотря на то что в каждом коде комбинации различны, оба кода вполне правомочны, так как комбинации в каждом из них отличаются друг от друга на кодовое расстояние d=3. В то же время сравнение кодов, составленных образующей матрицей [многочлен Р(Х)=Х3 + Х + 1] и умножением транспонированной матрицы на тот же многочлен, показывает полную идентичность комбинации этих кодов.

Теперь, когда ясна роль образующего многочлена при составлении циклических кодов, вырисовываются также следующие его свойства, которые могут помочь при изучении более сложных циклических кодов.

Первое свойство образующего многочлена заключается в том, что все разрешенные комбинации делятся на него без остатка. Это свойство следует из (1.2), и его можно проверить, разделив любую комбинацию кода на образующий ее многочлен. Таким образом, многочлен Р(Х) как бы позволяет образовать или выбрать из большего числа комбинации, удовлетворяющие только заданному закону построения кода, т. е. разрешенные. Поэтому многочлен Р(Х) и называется образующим.

Второе свойство образующего многочлена таково, что на него делится без остатка не только разрешенная комбинация, имеющая степень n — 1, но и двучлен Хn + 1. В нашем примере n = 7. При делении числа 10000001 на 1011 получается частное 10111 без остатка. Это значит, что образующий многочлен входит в качестве сомножителя в разложение двучлена Хn + 1, который с учетом равенства (1.5) можно записать в виде . Так для двучлена  составляющие сомножители разложения должны быть неприводимыми многочленами, степени которых являются делителями числа m = 3. К числам, на которые m = 3 делится без остатка, относятся 1 и 3. Из табл. 1.1 выпишем все неприводимые многочлены первой и третьей степеней, которые и явятся сомножителями в разложении двучлена Х7+1:

Х7+1= (Х+1)(Х3 + Х +1)(Х3 + Х2 + 1)(1.7)

Один из неприводимых многочленов третьей степени и должен быть выбран для кодирования, если k=4. Заметим, что такое разложение двучлена Хn+1 является одним из методов выбора образующего многочлена.

В рассмотренном примере при k=4 и m=3 n=k+m=7. В литературе циклические коды такого типа называются кодами (7,4). Из примера не следует, что для всех циклических кодов с обнаружением двойной ошибки образующий многочлен будет всегда иметь третью степень. Чем больше длина кода, тем выше степень образующего многочлена, что объясняется увеличением числа контрольных символов. Так, при k=26 согласно уравнению (1.4) m = 5. Это значит, что степень образующего многочлена должна быть не меньше пятой. Такой код обозначают как код (31,26).

*Циклические коды с d=4*. Эти коды могут обнаруживать одиночные, двойные и тройные ошибки или обнаруживать двойные и исправлять одиночные ошибки.

1. Выбор числа контрольных символов. Число контрольных символов в этом коде должно быть на единицу больше, чем для кода с d=3:

(1.8)

Для нахождения  можно воспользоваться уравнением (1.4). Если число контрольных символов определяется, как в коде Хэмминга, то уравнение (1.3) примет вид

 (1.9)

2. Выбор образующего многочлена. Образующий многочлен : равен произведению двучлена (Х+1) на многочлен 

(1.10)

Это объясняется тем, что двучлен (Х+1) позволяет обнаружить все одиночные и тройные ошибки, а многочлен Р(Х) — двойные ошибки. Так, для кода (7,3), обнаруживающего все трехкратные ошибки, можно было бы выбрать .

В общем случае степень l многочлена  равна числу m. Дальнейшая процедура кодирования остается такой же, как и при образовании кода с обнаружением двойной ошибки.

Пример 1.3. Требуется закодировать сообщение 10101010101010 циклическим кодом с d = 4:

G(X)= Х13 + Х11 + Х9 + Х7 + Х5 + Х3+ Х→10101010101010.

Определяем число контрольных символов по уравнению (1.4):



Из уравнения (1.8) следует, что . Выбираем из табл. 1.1 образующий многочлен для d=3. Пусть . Тогда



Так как необходимо закодировать только одно сообщение G(X), а не весь ансамбль двоичных кодов с k=14, то будем придерживаться процедуры кодирования, выполняемой по уравнению (1.2). Выбираем одночлен  . Тогда



Разделив полученное выражение на , находим остаток:



Следовательно, передаваемая закодированная комбинация будет иметь вид F(X) = (Х19 + Х17 + Х15 + Х13 + Х11 + Х9 + Х7) + (Х4 + Х3 + Х2 + X + 1).

10101010101010 011111

Информационные Контрольные

символы символы

### 1.2.5 Методы декодирования циклических кодов и обнаружения ошибок

*Обнаружение ошибок*. Идея обнаружения ошибок в принятом циклическом коде заключается в том, что при отсутствии ошибок закодированная комбинация F(X) делится на образующий многочлен Р(Х) без остатка. При этом контрольные символы m отбрасываются, а информационные символы k используются по назначению. Если произошло искажение принятой комбинации, то эта комбинация F(X) преобразуется в комбинацию Н(Х), которую можно представить как сумму двух многочленов:

H(X)=F(X) + E(X),(1.11)

где Е(Х)—многочлен ошибок, содержащий столько единиц, сколько элементов в принятой комбинация не совпадает с элементами переданной комбинации.

Пусть, например, была передана комбинация кода (7,4) ==1101001, закодированная с помощью Р(Х)=1011. Если она принята правильно, то деление на Р(Х) даст остаток, равный нулю. Если же комбинация принята как Н(Х)=1101011, то при делении на Р(Х) образуется остаток 010, что свидетельствует об ошибке, и принятая комбинация бракуется.

*Обнаружение и исправление ошибок*. Существует несколько вариантов декодирования циклических кодов. Один из них заключается в следующем.

1. Вычисление остатка (синдрома). Так же как и в кодах с обнаружением ошибок, принятую комбинацию делят на образующий многочлен Р(Х). Остаток R(X)=0 означает, что комбинация принята без ошибок. Наличие остатка свидетельствует о том, что комбинация принята искаженной. Дальнейшая процедура исправления ошибок протекает таким образом.

2. Подсчет веса остатка W. Если вес остатка равен или меньше числа исправляемых ошибок, т.е. , то принятую комбинацию складывают по модулю 2 с остатком и получают исправленную комбинацию.

3. Циклический сдвиг на один символ влево. Если W>s, то производят циклический сдвиг на один символ влево и полученную комбинацию снова делят на образующий многочлен. Если вес полученного остатка , то циклически сдвинутую комбинацию складывают с остатком и затем циклически сдвигают ее в обратную сторону вправо на один символ (возвращают на прежнее место). В результате получают исправленную комбинацию.

4. Дополнительные циклические сдвиги влево. Если после циклического сдвига на один символ по-прежнему W>s, то производят дополнительные циклические сдвиги влево. При этом после каждого сдвига сдвинутую комбинацию делят на Р(Х) и проверяют вес остатка. При  выполняют действия, указанные в п. 3, с той лишь разницей, что обратных циклических сдвигов вправо делают столько, сколько их было сделано влево.

Пример 1.4. Принят код 1101110, закодированный образующим многочленом Р(Х)=1011 и с s = l. Проверить наличие ошибки и в случае обнаружения исправить ее.

Делим комбинацию 1101110 на 1011 и находим, что остаток R(X)=111. Так как это не удовлетворяет равенству W=s, сдвигаем комбинацию 1101110 циклически на один символ влево. Получаем 1011101. В результате деления этой комбинации на Р(Х) находим остаток R1(X)=101. Вес этого остатка равен двум, т.е. больше s. Осуществляем новый циклический сдвиг влево. Получаем 0111011. Деление на Р(Х) дает остаток R2(X)=001, вес которого равен s. Складываем: 0111011 001 = 0111010. Теперь осуществляем два циклических сдвига последней комбинации вправо: после первого она принимает вид 0011101, после второго — 1001110, т.е. получается уже исправленная комбинация. Проверка показывает, что эта комбинация делится на Р(Х) без остатка.

## 

## 1.3 Математическая модель

Исходя из технического задания d = 4, а согласно формуле

d = r + s + 1, где

d — минимальное кодовое расстояние;

r — число обнаруживаемых ошибок;

s — число исправляемых ошибок.

имеем 2 варианта:

1. r = 2, s = 1 – обеспечивает обнаружение двух ошибок и исправление одной;
2. r = 3, s = 0 – обеспечивает обнаружение тройных ошибок;

Выбираем вариант 1, так как вариант номер 2 не допустим по ТЗ (нет исправления ошибок).

Имеем алфавит в 256 символов, что потребует 9 разрядов, так как комбинацию 00000000 использовать не будем. Имеем k = 9.

Опеределим число контрольных символов :





n = k + m

Так как k = 9, то



Тогда для :



n = 9 + 5 = 14

Найдём образующий многочлен:



Выберем  из таблицы 1.1. Пусть 



## 

## 1.4 Построение образующей матрицы

Из выше полученных расчетов мы знаем, что число информационных символов (бит) равно 9. Следовательно размерность единичной матрицы будет 9. Число проверочных символов m = 5, следовательно получим дополнительную матрицу, имеющую 9 строк и 5 столбцов.

Найдём дополнительную матрицу:

100000000|100111

100111 |—————————————————————

——————————

00111000111: 1-й остаток

————————

01110001110: 2-й остаток

————————

11100011100: 3-й остаток

100111

———————

11111011111: 4-й остаток

100111

————————

11001011001: 5-й остаток

100111

————————

10101010101: 6-й остаток

100111

————————

0110101101: 7-й остаток

————————

11010011010:8-й остаток

100111

————————

10011010011: 9-й остаток

100111

————————

000001

Итак, дополнительная матрица имеет вид:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| m5 | m4 | m3 | m2 | m1 |
| 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 1 | 1 | 0 |
| 1 | 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 0 | 1 |
| 1 | 1 | 0 | 1 | 0 |
| 1 | 0 | 0 | 1 | 1 |

Составим образующую матрицу:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| k9 | k8 | k7 | k6 | k5 | k4 | k3 | k2 | k1 | m5 | m4 | m3 | m2 | m1 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 1 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 |
| 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 |
| 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 |
| 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 1 |

Нахождение всех комбинаций циклического кода достигается суммированием по модулю 2 всевозможных сочетаний строк образующей матрицы.

## 1.5 Расчет достоверности передаваемых сообщений

Достоверность – степень соответствия принятой информации переданной. Оценкой достоверности служит вероятность правильного приема, равная отношению числа правильно принятых символов сообщения к общему числу переданных символов.

1. Для симметричного канала с независимыми ошибками.

Согласно ТЗ, P10=P01 =0,5.



Для одиночных ошибок:



Для двух ошибок:



Общая вероятность:



Это означает, что на 1000 переданных символов 7 будут с ошибкой. Тогда для 1000 сиволов достоверность будет 993/1000=0,993 или 99,3%.

2. Для несимметричного канала с независимыми ошибками.

Согласно ТЗ, P10=0,3

P01 =0,7.

Пусть сообщение будет следующим G=11001010101011, а искаженное G1=01101010101011.



Общая вероятность:



Это означает, что на 1000 переданных символов 2 будут с ошибкой. Тогда для 1000 сиволов достоверность будет 998/1000=0,998 или 99,8%.

## 1.6 Выводы

В данной главе былы освещены теоретические основы для решения технического задания. Были описаны структура и специфика циклических кодов и методов кодирования. Таким образом, была подведена база для последующей реализации поставленной задачи на языке программирования, а также схемной реализации.

# **2. Техническая реализация кодера, декодера** **и решателей**

## 2.1 Модульная структура кодера и его работа

Основу кодирующих устройств двоичных циклических кодов составляют регистры сдвига с обратными связями, позволяющие осуществлять как умножение, так и деление многочленов с приведением коэффициентов по модулю 2. Такие регистры также называют многотактными линейными переключательными схемами и линейными кодовыми фильтрами Хаффмена. Они состоят из ячеек памяти, сумматоров по модулю 2 и устройств умножения на коэффициенты многочленов множителя или делителя. В случае двоичных кодов для умножения на коэффициент, равный 1, требуется только наличие связи в схеме. Если коэффициент равен 0, то связь отсутствует. Сдвиг информации в регистре осуществляется импульсами, поступающими с генератора продвигающих импульсов. На вход устройств поступают только коэффициенты многочленов, причем начиная с коэффициента при переменной в старшей степени.

Как указывалось выше, образование циклического кода состоит из двух операций: умножения комбинации обычного двоичного кода G(X) на одночлен Xm и последующего деления этого произведения на выбранный образующий многочлен P(X). Полученные в остатке от деления контрольные символы приписываются к кодируемой комбинации. Таким образом, кодирующее устройство должно совмещать функции умножения и деления.

Рассмотрим методику построения кодирующего устройства. Требуется составить схему кодирующего устройства для многочлена:

P(X)=X5+X2+X+1.

Схематичное изображение кодирующего устройства представлено на рисунке 2.1.



Рис.2.1. Схематичное изображение кодирующего устройства

Схема, изображенная на рис. 2.1, работает следующим образом. В исходном состоянии ключ К1 находится в положении 1, а ключ К2 замкнут. Все подлежащие кодированию информационные символы, начиная со старшего разряда, поступают одновременно на выход и через сумматор на входе в схему кодирования. После того как пройдет последний символ k, ключ К1 переключится в положение 2, а ключ К2 размыкается. После этого регистр делает m шагов, равных числу ячеек, т.е. пять шагов. И весь остаток поступает на выход. Этот остаток представляет собой контрольные символы, следующие за информационными символами.

Рассмотрим подробнее процесс кодирования комбинации

Процесс кодирования комбинации G(X)= 000100000 с помощью кодера на рисунке 2.1, а показан в таблице 2.1.

В тактах 1-3 на вход поступают нули, поэтому в регистре ничего не меняется. Только в такте 4 единица кодируемого записывается в ячейки X0, X1, X2 и поступает на выход. В такте 5 на вход поступает нуль, поэтому в X0 поступает 0, и на выходе тоже 0. Из ячеек X0, X1, X2 единицы перемещаются в ячейки X1, X2, X3.

Аналогично и в такте 6, три единицы перемещаются далее вправо. На такте 7 единица из ячейки X4 поступает на сумматор по модулю 2 и складывается там с 0, поступающим с входа. Тогда, в результате сложения 1 и 0 по модулю 2 получается 1, которая поступает на остальные суммирующие элементы по модулю 2. В итоге во всех ячейках будут записаны 1. В тактах 7, 8, 9 просходит аналогично такту 6.

Таблица 2.1. Образование циклического кода

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Номер такта | Вход | Состояние ячеек регистра | | | | | Выход |
| X0 | X1 | X2 | X3 | X4 |
| 1  2  3  4  5  6  7  8  9 | 0  0  0  1  0  0  0  0  0 | 0  0  0  0  1  0  0  1  1  1 | 0  0  0  0  1  1  0  1  0  0 | 0  0  0  0  1  1  1  1  0  1 | 0  0  0  0  0  1  1  1  1  0 | 0  0  0  0  0  0  1  1  1  1 | 0  0  0  1  0  0  0  0  0 |
| 10  11  12  13  14 |  | 0  0  0  0  0 | 1  0  0  0  0 | 0  1  0  0  0 | 1  0  1  0  0 | 0  1  0  1  0 | 1  0  1  0  1 |

После такта 9 остаток R(X) оказывается записанным в ячейках регистра. После переключения ключа K1 в положение 2 и выключения ключа K2 этот остаток в последующие четыре такта переписывается на выход вслед за информационными символами.

## 2.2 Модульная структура декодера и его работа

Декодирование циклического-кода с обнаружением и исправлением нескольких ошибок. Метод такого декодирования был изложен в теоретическом введении. Рассмотрим теперь схемную реализацию декодирования комбинации 10000000010011, искаженную одним символом и принявшей вид 10010000010011. Декодер (рис.2.2) состоит из делителя, выполненного для деления на многочлен P(X)=X5+X2+X+1, и запоминающего устройства, представляющего собой регистр с сумматором символов k. Комбинация поступает одновременно на делитель и запоминающее устройство начиная со старшего разряда. Искаженный символ в комбинации отмечен почеркиванием. Вначале ключ K1 замкнут, а ключ К2 разомкнут. В таблице 2.2 показан процесс деления начиная с такта 6, так как в первые пять тактов происходит заполнение делителя и обратная связь еще не проявляется.



Рис. 2.2. Сруктурная схема декодирования циклического кода с исправлением одной ошибки.

В такте 6 единица с Х4 поступает на сумматоры по модулю 2, и в X0 записывается единица, нуль, находившийся в X0, сложившийся с единицей даст 1 и она запишется в X1 , единица из X1 сложившись с 1 даст нуль и этот нуль запишется в X2, нуль из X2 перейдёт в X3, а единица из X3 в X4. Далее аналогично.

Таблица 2.2. Работа делителя

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Номер такта | Делимое | Состояние ячеек делителя | | | | | Вес  остатка |
| X0 | X1 | X2 | X3 | X4 |
| 6  7  8  9  10  11  12  13  14  15  16  17  18 | 0  0  0  0  1  0  0  1  1 | 1  0  0  0  0  1  0  0  1  1  0  1  1 | 1  1  0  0  1  1  1  1  0  0  1  1  0 | 0  1  1  0  1  0  1  0  1  1  0  0  0 | 0  0  1  1  0  1  0  1  0  1  1  0  0 | 0  0  0  1  1  0  1  0  1  0  1  1  0 | 3  3  3  3  1 |

В такте 14 синдром (остаток от деления) оказывается записанным в ячейках регистра (10101). Однако его вес W=3 больше числа исправляемых ошибок s, поэтому делитель делает еще один шаг (такт 15), в процессе которого снова осуществляется деление на многочлен Р(Х). Синдром 10110 опять имеет вес W=3. Только после 18 такта W=1=s. В этот момент ключ К1 размыкается, а ключ К2 замыкается и синдром с делителя начинает поступать на сумматор запоминающего устройства, у которого ключ К3 замкнут, а ключ К4 разомкнут.

Это устройство в такте 14 первого этапа полностью заполнилось, а на втором этапе его работы начался циклический сдвиг записанной информации (таблица 2.3). Так в такте 1 единица из ячейки X8 информационных символов переместилась в ячейку X0 контрольных символов m. В такте 2 эта единица передвинулась в ячейку X1, а ее место в ячейке занял следующий нуль и т. д.

Таблица 2.3. Работа ЗУ декодера

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Номер такта | Символы m | | | | | Символы k | | | | | | | | |
| X0 | X1 | X2 | X3 | X4 | X0 | X1 | X2 | X3 | X4 | X5 | X6 | X7 | X8 |
| 14  1  2  3  4  5  6  7  8  9  10  11  12  13  14 | 1  1  0  0  1  0  0  0  0  0  1  0  0  1  1 | 1  1  1  0  0  1  0  0  0  0  0  1  0  0  1 | 0  1  1  1  0  0  1  0  0  0  0  0  1  0  0 | 0  0  1  1  1  0  0  1  0  0  0  0  0  1  0 | 1  0  0  1  1  1  0  0  1  0  0  0  0  0  1 | 0  1  0  0  1  1  1  0  0  0  0  0  0  0  0 | 0  0  1  0  0  1  1  1  0  0  0  0  0  0  0 | 0  0  0  1  0  0  1  1  1  0  0  0  0  0  0 | 0  0  0  0  1  0  0  1  1  1  0  0  0  0  0 | 0  0  0  0  0  1  0  0  1  1  1  0  0  0  0 | 1  0  0  0  0  0  1  0  0  1  1  1  0  0  0 | 0  1  0  0  0  0  0  1  0  0  1  1  1  0  0 | 0  0  1  0  0  0  0  0  1  0  0  1  1  1  0 | 1  0  0  1  0  0  0  0  0  1  0  0  1  1  1 |

Первые четыре нуля синдрома, поступающие на сумматор, не влияют на работу запоминающего устройства. Лишь в такте 9 единица синдрома, складываясь по модулю 2 с ошибочной единицей символов k (обозначена подчеркиванием), «уничтожают» её, т. е. исправляют ошибку. Регистр запоминающего устройства продолжает переключаться до окончания второго цикла (этапа) его работы. После такта 14 ключи К2 и К3 размыкаются, а ключи К1 и К4 замыкаются: начинается считывание исправленной комбинации и одновременная запись новой.

Таким образом, декодирование состоит их двух этапов. На первом этапе осуществляются нахождение остатка и запись кодовой комбинации, на втором — ее исправление и расстановка символов k и m на свои места.

## 2.3 Модульная структура решателя кодера и его работа

Решатель кодера должен выполнять следующие задачи: обеспечить связь с передающим устройством, передачу сигналов задержек при исправлении ошибок и повторных передачах информации. Также должен выполнять свою основную функцию – приём сигнала продолжения. Структурная схема решателя приведена на рисунке 2.3.



Рис. 2.3. Сруктурная схема решателя кодера.

Рассмотрим все случаи более подробно.

1. Сообщение проходит без ошибок. Решатель принимает сигнал продолжения (F\_NXT). Ключ Кр открывается и начинает поступать следующее сообщение. Так как длина сообщения составляет 9 символов, то в конце такта 9 ключ Кр размыкается, и на передающее устройство поступает сигнал ожидания посылки сообщения (WAIT14). Далее решатель начинает ожидать сигнал продолжения.

2. Сообщение прошло с ошибкой. Пусть искажение произошло только в одном символе. Тогда решатель на 14 такте не получает сигнал продолжения. Ключ Кр размыкается. Решатель посылает сигнал ожидания на передатчик в течение 14 тактов (этап 2). За это время искаженная комбинация исправляется, и с решателя декодера приходит сигнал продолжения, Кр замыкается.

3. Сообщение содержит более одной ошибки. Всё происходит аналогично пункту 2, за исключением того, что и после 14 такта второго этапа ожидания он не получает сигнал продолжения и ключ Кр остается разомкнутым. Тогда ещё в течение 14 тактов (этап 3) решатель посылает на передатчик сигнал ожидания (WAIT14), но на кодер посылается сигнал повтора сообщения из буфера (RPT\_CODE).

## 2.4 Модульная структура решателя декодера и его работа

Решатель декодера должен выполнять следующие функции: он должен после принятия сообщения декодером определить, есть ли ошибки в сообщении, если их нет, то отправить сигнал продолжения (F\_NXT), в противном случае попытаться исправить ошибку и если ошибка не будет исправлена, то «отправить» сигнал повтора, вернее не отправить сигнал продолжения.

Структурная схема решателя декодера приведена на рисунке 2.4.

Определение ошибки заключается в нахождении остатка от деления R(X). Если остаток равен 0, в случае данной работы (00000), то сообщение было принято без ошибок, и посылается сигнал продолжения (F\_NXT).

Если остаток будет иметь вес равный 1, то есть это следующие комбинации: 10000, 01000, 00100, 00010, 00001, тогда ошибка исправима, а значит не потребуется повтора передачи. После исправления ошибки будет послан сигнал продолжения.



Рис. 2.4. Сруктурная схема решателя декодера.

Если вес остатка будет больше 1, то исправление ошибки невозможно и решатель отправляет сигнал повтора (сигнал F\_NXT = 0).

Входы R1…R5 подключаются к регистру делителя, а выходы (NXT, ERCOR) на блок коррекции ошибок.

## 2.5 Выбор микросхем для реализации кодера, декодера и решателей

Согласно технического задания кодер, декодер и решатели выполняются на ПЛИС (программируемые логические интегральные схемы). ПЛИС являются наиболее перспективными элементами, так как они вполне могут заменить десятки и сотни микросхем старых типов. Они может немного и уступают им по скорости, но в современных микросхемах этот недостаток практически устранен. Однако ПЛИС обладают огромным преимуществом перед обычными логическими схемами, что отражено в их названии «программируемые». Это означает, что теперь, легко производить модернизацию схем, так как при незначительной переработке какого-либо устройства, достаточно с помощью специального оборудования (программаторов) перезаписать ПЛИС. А при сборке на обычных элементах, может потребоваться полная переработка схемы, вплоть до изменения печатной платы и т.п., что значительно повысит расходы на перепроектирование схемы. Или легко будет в кодере/декодере использовать более сложные или наоборот более простые алгоритмы кодирования, в зависимости от помех, возникающих в линии связи.

Поэтому использование ПЛИС очень удобно в подобных системах передачи данных.

Ещё одним плюсом ПЛИС является компактность устройств, а также меньшее количество соединений на плате, что в свою очередь повышает надёжность устройства.

Важным достоинством ПЛИС является также создание собственных логических элементов на языках AHDL, VHDL и на уровне временных диаграмм. Например, решатель декодера был реализован именно на языке VHDL, текст программы которого приведен в приложении.

В данной работе были выбраны ПЛИС фирмы Altera. Кодер, декодер и решатели были смоделированы на ЭВМ с помощью специальной программы MAX+plus II фирмы Altera. Использование программы моделирования MAX+plus II позволило очень быстро спроектировать рабочие варианты кодера, декодера и решателей, а также смоделировать их работу.

Были выбраны следующие микросхемы: серии MAX 7000: EPM7032LC44-6 – на них реализован кодер и решатель, и серии MAX 9000: EPM9320LC84-15 – на ней реализован декодер и решатель. Для декодера была выбрана более «емкая» микросхема, так как, схема декодера гораздо сложнее схемы кодера, это также позволит в будущем реализовывать на ней и более сложные декодирующие устройства, например для кода с кодовым расстоянием более 5 (коды БЧХ), которые широко применяются в настоящее время, а кодер реализуется намного проще, что позволило применить менее «емкую» микросхему.

*Выбор моделей микросхем*.

Для реализации кодера, декодера и решателей нам понадобятся следующие элементы: ИЛИ (2, 3, 4, 6, 8, 12 входовые), И (2, 6 входовые), НЕ, 4-х разрядный счетчик типа 7493, двоично-десятичный дешифратор, D, RS и T – триггеры, 2-х входовое исключающее ИЛИ (XOR), решатель декодера, мультиплексор на 2 канала.

Опишем кратко каждый элемент.

1. ИЛИ. Выход равен 0, только когда все входы нулевые.



X1X2Y

000

011

101

111

2. И. Выход равен единице, только когда все входы равны 1.



X1X2Y

000

010

100

111

3. НЕ. Отрицание



X1Y

01

10

4. Счетчик.



Представляет собой двоичный четырехразрядный счетчик. Выход QA должен быть соединен со входом CLKB. CLKA подключается к генератору тактовых импульсов (ГТИ).

Счет Выходы

QAQBQCQD

00000

10001

20010

30011

40100

50101

60110

70111

81000

91001

101010

111011

121100

131101

141110

151111

Если RO1 и RO2 одновременно равны 1, то происходит сброс счетчика в 0, при любых других комбинациях RO1 и RO2 счетчик будет считать.

5. Двоично-десятичный дешифратор 4 входа – 16 выходов.

Входы A, B, C, D, выходы Q0-Q15. Преобразует двоичный код в десятичный.



Входы Выходы

D C B A Q15 Q14 Q13 Q12 Q11 Q10 Q9 Q8 Q7 Q6 Q5 Q4 Q3 Q2 Q1 Q0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1

0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0

0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0

0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0

0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0

0 1 0 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0

0 1 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0

0 1 1 1 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0

1 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0

1 0 0 1 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0

1 0 1 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0

1 0 1 1 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0

1 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0

1 1 0 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0

1 1 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0

1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

6. Триггеры.

RS-типа.



Работа триггера. При CLRN = 0 происходит установка триггера в 0, независимо от входа CLK, то есть Q = 0 (очистка).

Входы | Выход

CLRN CLK S R | Q

1 0 x x | хранит

1 0→1 0 0 | хранит

1 0→1 1 0 | 1

1 0→1 0 1 | 0

1 0→1 1 1 | запрещено

D-типа.



Работа триггера. При CLRN = 0 происходит установка триггера в 0, независимо от входа CLK, то есть Q = 0 (очистка).

Входы | Выход

CLRN CLK D | Q

1 0 x | хранит

1 1 x | хранит

1 0→1 1 | 1

1 0→1 0 | 0

6. Элемент исключающее ИЛИ (2х входовое).



Когда входы одинаковы, на выходе 0, если разные, то 1.

X1X2Y

000

011

101

110

7. Решатель декодера.

Представляет собой обыкновенный двоично-десятичный дешифратор на 5 входов – 32 выхода и шестивходовой элемент ИЛИ.

На вход подается остаток от деления. Если он равен 0, то активизируется выход Q0 (это соответствует сигналу NXT – ошибок нет), если вес остатка равен 1, то активны Q1, Q2, Q4, Q8, Q16 (это соответствует сигналу ERCOR – ошибка исправима).



8. Мультиплексор на два канала.

Выполняет роль коммутатора каналов. Используется как в кодере, так и в декодере.



Входы |Выход

S A B |Y

0 х 1 |1

0 х 0 |0

1 1 х |1

1 0 х |0

## 2.6 Описание функциональной схемы кодера и решателя кодера

Работа кодера и решателя кодера была описана выше. Поясним некоторые моменты.

При начале передачи передатчик начинает посылать новый пакет сообщений. Сначала посылается сигнал очистки памяти кодера и решателя (CLRN = 0). Затем он становится равен 1.

Далее начинает передаваться сообщение № 1 на вход решателя кодера (INF\_IN), состоящее из 9 символов. Оно поступает на ключ Кр, который выполнен на элементах DD1.21 и DD1.22. Одновременно оно поступает и на кодер через ключ К2, реализованный на элементах DD2.13 и DD2.14.

В кодере предусмотрен буфер на 14 бит, в который записывается закодированное сообщение. Это позволит на аппаратном уровне осуществлять повтор сообщения, не «отвлекая» передатчик, в роли которого выступает ЭВМ. Это позволит при ошибках, не прерывать другие задачи, которые могут выполняться на данной ЭВМ и использовать её только в качестве источника информации, не нагружая дополнительными задачами. Можно было его сделать и на 9 бит, но емкость примененной микросхемы позволяет применить и 14 битный буфер. Буфер управляется ключами, которые выполняют необходимые коммутации, соединение с кодером, запись уже закодированного сообщения и т.п. Ключи реализованы на элементах DD2.4, DD2.3, DD2.2, DD2.34 и DD2.35. Элементы памяти выполнены на триггерах D-типа (элементы DD2.20 – DD2.33).

Отсчет количества бит сообщения производится с помощью счетчиков: DD1.9, DD1.17 – в решателе кодера и DD2.16 – в кодере.

Делитель предназначен для получения контрольных символов. Его работа была рассмотрена выше. Он выполнен на элементах DD2.5 – DD2.12.

Дешифратор и другие логические элементы используются для работы ключей и включения/выключения каналов передачи.

## 

## 2.7 Описание функциональной схемы декодера и решателя декодера

Работа декодера и решателя декодера была описана выше. Поясним некоторые моменты.

При приеме первого сообщения пакета посылается сигнал очистки памяти кодера и решателя (CLRN = 0). Затем он становится равен 1.

Далее начинает приниматься сообщение № 1 на вход делителя и в память декодера (INFA), состоящее из 14 бит. Оно поступает на сумматор по модулю 2, который выполнен на элементе DD1.1 и соответственно оно поступает и в ОЗУ декодера. Память предназначена для хранения полученного сообщения.

Отсчет количества бит сообщения производится с помощью счетчиков: DD1.16, DD1.23.

Делитель предназначен для получения остатка от деления. С выходов делителя на дешифратор DD1.12 поступает остаток, соответственно при остатке с весом 0 посылается сигнал продолжения (NXT). Если вес остатка равен 1, то посылается сигнал ERCOR (ошибка исправима). Делитель выполнен на элементах DD1.1 – DD1.8.

Дешифратор (DD1.32) и другие логические элементы используются для работы ключей и включения/выключения каналов передачи.

## 

## 2.8 Описание принципиальной схемы кодера и решателя кодера

Полученные кодер и решатель представляет собой микросхемы, с функциями, которые были описаны в функциональной схеме.

В итоге имеем две микросхемы, одна из них выполняет функции решателя, а другая функции кодера.

У решателя имеются следующие входы:

INF\_IN – информационный;

CLRN – очистка памяти решателя;

CLK – вход генератора тактовых импульсов;

F\_NXT – принимает сигнал о продолжении;

И имеются следующие выходы:

INF\_OUT – информационный выход;

RPT\_CODE – сигнал повтора;

WAIT14 – сигнал приостановки посыла информации с передатчика;

Далее идут выходы, которые несут чисто информационный характер для показа внутренней работы микросхемы и непосредственно не используются:

RES\_CO – сигнал сброса счетчиков;

CN1…CN5 – разряды счетчика решателя;

Далее рассмотрим кодер.

Входы:

RPT – прием сигнала RPT\_CODE;

CLRN – очистка памяти кодера;

CLK – вход генератора тактовых импульсов;

INFA – информационный вход;

Выход один, это OUT – посыл закодированного сообщения либо в линию связи (ЛС), либо на модулятор (в зависимости от вида устройства: если ЛС аналоговая, то на модулятор, если цифровая, то непосредственно в ЛС).

Также имеется разъем для соединения с ЛС и передающим устройством, а также для подвода питания.

Опишем работу полученного устройства.

Решатель.

Пусть с передатчика пришло сообщение 011001010. Сигнал F\_NXT=0. Всё это время сигнал очитки памяти выключен (CLRN = 1). На такте 9, счетчик досчитает до 8 (01000), так как он начинает считать с нуля. За эти девять тактов всё полученное сообщение поступает на выход INF\_OUT. После девятого такта включается сигнал WAIT14, потому что необходимо ещё закодировать сообщение.

Кодер кодирует сообщение, и посылает в течение 5 тактов пять контрольных символов. Допустим в декодер сообщение поступило с ошибкой. Сигнал F\_NXT не пришел, то есть F\_NXT=0 в течение всех 14 тактов. Тогда счетчик считает далее до 27 (11011), и всё это время на передатчик поступает сигнал приостанова посылки сообщения, так как идёт попытка исправления искаженного сообщения в декодере.

Пусть в декодере сообщение не исправилось, тогда сигнал F\_NXT не приходит, то есть опять F\_NXT=0. И соответственно после 9 такта на информационный выход ничего не поступает. Не получив на 28 такте сигнала F\_NXT решатель сбрасывает счетчик решателя RES\_CO=1, и послав на передатчик сигнал приостанова, отправляет на кодер сигнал повтора RPT\_CODE = 1.

С буфера кодера в течение 14 тактов сообщение заново посылается в ЛС.

Пусть теперь сообщение в декодере было принято без ошибок. На 14 такте приходит сигнал F\_NXT, решатель снова сбрасывает счетчик и принимает новое сообщение с передающего устройства, предварительно сбросив сигнал WAIT14 в нуль.

Соответственно далее всё происходит подобным образом. Более подробно работа решателя приведена в приложении на временной диаграмме.

Кодер.

Пусть с решателя кодера пришло сообщение 011001010. Сигнал повтора сообщения RPT\_CODE=0. Всё это время сигнал очитки памяти выключен (CLRN = 1).

Кодер в начале первые 9 тактов просто выводит сообщение на выход. Одновременно оно записывается в память кодера. Затем посылает в течение 5 тактов пять контрольных символов. Допустим с решателя кодера поступает сигнал повтора посылки сообщения RPT\_CODE=1. Тогда в течение 14 тактов кодер уже из памяти отсылает в ЛС (вывод OUT) сообщение заново.

Соответственно далее всё происходит подобным образом. Более подробно работа решателя приведена в приложении на временной диаграмме.

## 2.9 Описание принципиальной схемы декодера и решателя декодера

Полученные декодер и решатель представляет собой микросхему, с функциями, которые были описаны в функциональной схеме.

В итоге имеем микросхему, которая выполняет функции решателя и декодера.

У ней имеются следующие входы:

INFA – информационный;

CLK – вход генератора тактовых импульсов;

И имеются следующие выходы:

OUT – информационный выход;

F\_NXT – посыл сигнала о продолжении передачи;

Далее идут выходы, которые несут чисто информационный характер для показа внутренней работы микросхемы и непосредственно не используются:

F\_ERCOR – сигнал об исправимости искаженного сообщения;

ERCOR – промежуточный сигнал об исправимости искаженного сообщения;

CLRN\_OUT – сигнал очистки памяти декодера;

ТТК34 – управление ключами;

RES\_CO – сигнал сброса счетчиков;

CN1…CN5 – разряды счетчика декодера;

Также имеется разъем для соединения с ЛС и передающим устройством, а также для подвода питания.

Опишем работу полученного устройства.

1. Пусть из ЛС (INFA) поступило сообщение 01100000010111. Всё это время сигнал очистки памяти выключен (CLRN\_OUT = 1). На 14 такте проверяется остаток от деления. Он получился равным 0, так как в конце 14 такта появляется сигнал NXT. Это означает, что сообщение принято без ошибок, поэтому вырабатывается импульс F\_NXT и поступает в обратный канал, и RES\_CO, который сбрасывает счетчик в нуль.

2. Теперь рассмотрим исправление ошибки. Допустим поступило искаженное сообщение 01000000010111. Искаженный символ отмечен подчеркиванием. До 14 такта всё происходит аналогично пункту 1, но на 14 такте нет сигнала NXT, что означает ненулевой остаток. Также и сигнал ERCOR (ошибка исправима) не равен 1, поэтому происходят циклические сдвиги сообщения в памяти декодера и происходит деление. Вот на такте 17 появляется сигнал ERCOR = 1. И появляется сигнал F\_ERCOR (для дальнейшего срабатывания NXT). На такте 22 сообщение исправляется и появляется сигнал NXT. И соответственно на 28 такте появляется импульс F\_NXT и RES\_CO.

Далее в следующие 14 тактов происходит отправка исправленного сообщения на выход в приемник 01100000010111 (OUT) и одновременное чтение следующего сообщения.

3. И теперь самый «плохой» случай, когда искажения более, чем в одном символе, то есть сообщение нельзя будет исправить. Пусть на вход поступило следующее сообщение: 11000000010111

До 28 такта всё аналогично пункту 2, за исключением того, что не появляются сигналы NXT, ERCOR и соответственно F\_NXT, F\_ERCOR. На 28 такте происходит сброс счетчика, очистка памяти CLRN\_OUT = 0 и чтение сообщения заново (так как на решатель кодера не приходит импульс F\_NXT).

Соответственно далее всё происходит подобным образом. Более подробно работа решателя приведена в приложении на временных диаграммах.

## 2.10 Выводы

В данной главе представлено описание работы кодирующего, декодирующего и решательных устройств, их функциональное изображение. Приведено их описание, а также представлен набор микросхем, используемых для реализации принципиальных схемы кодера с решателем и декодера с решателем.

# **3. Описание программных средств, разработанных в ходе реализации проекта**

## 

## 3.1 Структура системы

Для решения задачи был применен наиболее простой подход. Был взят наиболее простой язык высокого уровня Паскаль.

Для данной разработки в программе должны содержаться массивы для хранения требуемых для работы переменных:

Для конкретной реализации в программе должны содержаться массивы для хранения требуемых для работы алгоритма переменных: CODE – массив разрядов, входящих в сообщение, должен вводиться пользователем (9 разрядов – информационные). И массив G\_CODE для закодированного сообщения (9 разрядов – информационные, 5 разрядов – проверочные).

Проект также должен содержать справку об авторе, о назначении данной программы, а также о том, как с ней работать.

## 3.2 Входные данные, форма представления результатов

## 

Входными данными является код, который вводит пользователь. Код вводится следующим образом: имеется 9 символов, по умолчанию они нули**.** С помощью курсорных клавиш перемещается курсор по символам и при нажатии клавиши пробел значение символа меняется на противоположное, то есть 1 на 0, а 0 на 1.

Результаты приведены в виде закодированного сообщения с помощью циклического кода (14,9)**.**

## 3.3 Спецификация на программу в целом.

Программа выполняет следующие функции:

1. Кодирование кодовой последовательности с помощью циклического кода (14,9);
2. Вывод закодированного сообщения (информационные и контрольные символы);
3. Возможность пользователя «искажать» закодированное сообщение для дальнейшего декодирования;
4. Декодирование закодированного сообщения и исправление ошибок в искаженном сообщении.

Итак, входные данные:

Code:Array[1..k]Of Boolean; – массив начальной кодовой комбинации;

Выходные данные:

G\_Code:Array[1..n]Of Byte; – закодированная кодовая комбинация;

Константы:

a:Array[1..k,1..n]Of Byte=((0,0,0,0,0,0,0,0,1,0,0,1,1,1),

(0,0,0,0,0,0,0,1,0,0,1,1,1,0),

(0,0,0,0,0,0,1,0,0,1,1,1,0,0),

(0,0,0,0,0,1,0,0,0,1,1,1,1,1),

(0,0,0,0,1,0,0,0,0,1,1,0,0,1),

(0,0,0,1,0,0,0,0,0,1,0,1,0,1),

(0,0,1,0,0,0,0,0,0,0,1,1,0,1),

(0,1,0,0,0,0,0,0,0,1,1,0,1,0),

(1,0,0,0,0,0,0,0,0,1,0,0,1,1)); – образующая матрица;

stepen=6; – степень образующего многочлена=6, т.к. порядковый номер степени начинаем отсчитывать не с 0, а с 1;

Polynom:Array[1..stepen]Of byte=(1,0,0,1,1,1); – образующий многочлен.

## 3.4 Системные требования

Минимальные:

Процессор: 80486-33

Память: 4 Mb RAM

Видеопамять: 512Kb DRAM

Свободное место на жестком магнитном диске: 1 Mb

Операционная система: DOS 3.30 или выше.

## 3.5 Спецификация на программу в целом.

Программа соответствует требованиям технического задания. Она успешно кодирует, декодирует и исправляет введенную двоичную последовательность при помощи кодов. Созданный удобный дружественный интерфейс – понятен и прост. Кроме того комментарии позволяют быстро разобраться в программе и при необходимости внести в нее поправки.

В программе широко использовались элементы технологии TOP-DOWN.

Процедуры написанные для данной программы могут быть в дальнейшем использоваться в других программах.

# **4. Результативная часть**

## 4.1 Тестирование

Тестирование – это процесс, посредством которого проверяется правильность программы. Его цель – показать, что программа правильно работает в соответствии с проектными спецификациями.

При тестировании проверялась работа каждого модуля в отдельности, а также всей программы в целом. Было проведено несколько тестирований, после каждого из которых проводилась доработка программы и устранение ошибок. Тестирование проводилось из расчета на то, программой могут пользоваться неопытные пользователи, которые непредсказуемы в работе с программой.

На первом этапе тестирования вводилось несколько новых данных, и с ними проводились различные операции. Результаты этого тестирования показали правильную работу модуля, обеспечивающего ввод данных (проверялась защита от некорректного ввода и запоминание данных в память),

Результаты тестирования показали устойчивую работу программы .

Тестирование показало, что программа полностью соответствует техническому заданию. Верно разработаны алгоритмы и реализованы процедуры кодирования, декодирования и исправления ошибок.

При тестировании мы получили следующие примеры выполнения программы и алгоритма, что подтверждает правильность задания программы (в данном случае применялся метод черного ящика):

Приведем тестовую таблицу с введенными кодами, закодированными последовательностями, то есть покажем соответствие между входными и выходными данными.

Таблица 4.1. Тестирование программы

|  |  |  |  |
| --- | --- | --- | --- |
| Введенный код | Код с проверочными  символами | Передаваемый код | Декодированный код |
| 101010101 | 10101010111100 | 11101010111100 | 10101010111100 |
| 010000001 | 01000000111101 | 01000000111001 | 01000000111101 |
| 010100000 | 01010000001111 | 01010000000011 | невозможно  декодировать |
| 111111110 | 11111111000101 | 11111111000101 | 11111111000101 |

## 4.2 Описание пользовательского интерфейса

После запуска программы на экране появляется меню, содержащее 4 пункта:

1. Кодировка
2. Помощь
3. О программе
4. Выход

После активации пункта номер 1 открывается окно, отображающее процессы кодирования и декодирования.

## 4.3. Инструкция пользователю.

Изменение кода происходит с помощью курсорных клавиш и клавиши пробел. Нажатием Enter пользователь подтверждает введенную кодовую комбинацию. Далее отображается закодированное сообщение. Его можно «искажать» по усмотрению пользователя. Дальнейшее нажатие клавиши Enter приведет к декодированию сообщения, если это возможно, в противном случае выводится соответствующее сообщение.

## 4.4 Выводы

Написанная программа полностью соответствует ТЗ, правильно кодирует и декодирует циклический код (14,9), а также исправляет ошибки.

# **5. Заключение**

В результате проделанной работы была построена математическая модель помехозащищенного циклического кода (14,9), который кодирует информацию так, что при приеме может быть обнаружено две ошибки и одна из них исправлена. Данный код кодирует передаваемое сообщение из 9 бит, количество различных сообщений – более 256 (согласно ТЗ).

Математическая модель данного кода представляет собой программу, написанную с помощью языка Borland Pascal 7.0. Составленная программа работает в соответствии с техническим заданием и позволяет кодировать и декодировать вводимые сообщения.

Также в данной работе были разработаны структурная схема передачи данных с решающей обратной связью, функциональные и принципиальные схемы кодера и декодера с решателями в соответствии с техническим заданием.

Полное описание проведенной работы с пояснительными рисунками, таблицами и различными расчетами содержатся в данной расчетно-пояснительной записке. Графическая часть записки – структурная, функиональная и принципиальная схемы – выполнена в соответствии с требованиями ЕСКД и вынесены в приложение. А также к расчетно-пояснительной записке прилагаются документированный текст программы, перечень элементов, используемых для построения принципиальных схем, текст программы решателя декодера, написанный на языке VHDL и техническое задание.

Также было проведено моделирование работы кодера, декодера и решателей на программе MAX+plus II, где были получены соответствующие временные диаграммы, которые также вынесены в приложение.

На основании вышеизложенного материала можно сделать вывод, что задача, поставленная в техническом задании, – выполнена.

# 

# **Текст программы на языке VHDL для решателя декодера**

**LIBRARY ieee;**

**USE ieee.std\_logic\_1164.all;**

**USE ieee.std\_logic\_arith.all;**

**ENTITY dec5 IS**

**PORT**

**(R1, R2, R3, R4, R5: IN STD\_LOGIC;**

**ERCOR, NXT: OUT STD\_LOGIC);**

**END dec5;**

**ARCHITECTURE decoder5 OF dec5 IS**

**BEGIN**

**process (R1, R2, R3, R4, R5)**

**begin**

**if (R1='0' and R2='0' and R3='0' and R4='0' and R5='0') then**

**ERCOR<='0';**

**NXT<='1';**

**elsif (R1='1' and R2='0' and R3='0' and R4='0' and R5='0') then**

**ERCOR<='1';**

**NXT<='0';**

**elsif (R1='0' and R2='1' and R3='0' and R4='0' and R5='0') then**

**ERCOR<='1';**

**NXT<='0';**

**elsif (R1='0' and R2='0' and R3='1' and R4='0' and R5='0') then**

**ERCOR<='1';**

**NXT<='0';**

**elsif (R1='0' and R2='0' and R3='0' and R4='1' and R5='0') then**

**ERCOR<='1';**

**NXT<='0';**

**elsif (R1='0' and R2='0' and R3='0' and R4='0' and R5='1') then**

**ERCOR<='1';**

**NXT<='0';**

**else**

**ERCOR<='0';**

**NXT<='0';**

**end if;**

**end process;**

**END decoder5;**

# 

# **Документированный текст программы**

program shhh;

uses Crt;

const

On=516; {курсор включён}

Off=1600; {курсор выключен}

n=14; {общее число символов сообщения}

k=9; {число информационных символов}

s=1; {число исправляемых символов}

{Образующая матрица}

a:Array[1..k,1..n]Of Byte=((0,0,0,0,0,0,0,0,1,0,0,1,1,1),

(0,0,0,0,0,0,0,1,0,0,1,1,1,0),

(0,0,0,0,0,0,1,0,0,1,1,1,0,0),

(0,0,0,0,0,1,0,0,0,1,1,1,1,1),

(0,0,0,0,1,0,0,0,0,1,1,0,0,1),

(0,0,0,1,0,0,0,0,0,1,0,1,0,1),

(0,0,1,0,0,0,0,0,0,0,1,1,0,1),

(0,1,0,0,0,0,0,0,0,1,1,0,1,0),

(1,0,0,0,0,0,0,0,0,1,0,0,1,1));

stepen=6; {степень образующего многочлена=6, т.к.

порядковый номер степени начинаем

отсчитывать не с 0, а с 1 }

{Образующий многочлен :

X^5 + X^2 + X + 1}

Polynom:Array[1..stepen]Of byte=(1,0,0,1,1,1);

MenuColor=3; {цвет активного пункта меню}

GroundColor=white; {цвет фона}

CodeLine=5; {номер строки ввода кода}

G\_CodeLine=6; {номер строки закодированного сообщения}

D\_CodeLine=8; {номер строки искажённого сообщения}

C\_CodeLine=7; {номер строки исправленного сообщения}

Begin\_Line=34; {номер столбца начала строк}

var

menu\_p:array[1..18] of string[19]; {массив названий пунктов меню}

n\_pun,from:byte; {текущий номер пункта меню}

n\_z:integer; {количество записей в базе данных}

key:char; {нажатая клавиша}

i,j,x:byte; {счетчик}

Code:Array[1..k]Of Boolean; {начальная кодовая комбинация}

G\_Code:Array[1..n]Of Byte; {закодированная кодовая комбинация}

(\* ИНИЦИАЛИЗАЦИЯ \*)

PROCEDURE init;

begin

menu\_p[1]:=' КОДИРОВКА';

menu\_p[2]:=' ПОМОЩЬ ';

menu\_p[3]:=' О ПРОГРАММЕ';

menu\_p[4]:=' ВЫХОД ';

menu\_p[5]:=' СПРАВКА ';

menu\_p[6]:=' АВТОР ';

menu\_p[7]:=' ДА ';

menu\_p[8]:=' НЕТ ';

end;

(\* процедура работы с курсором \*)

Procedure Cursor(q:Integer);

Begin Asm

mov AH,01h

mov CX,q

Int 10h

End End;

(\* Процедура рисования простого окна \*)

PROCEDURE win(x1,y1,x2,y2,color:byte);

begin

textbackground(color);

window(x1,y1,x2,y2);

clrscr;

end;

(\* Процедура рисования окна с рамкой, тенью и заголовком \*)

PROCEDURE wind(x1,y1,x2,y2,foncol,textcol:byte;zagl:string);

var pos:byte; {позиция х для заголовка окна}

i,j:integer; {счетчики}

begin

window(1,1,80,25);

textbackground(cyan);

textcolor(darkgray);

for i:=y1 to y2+2 do

begin

gotoxy(x1-1,i);

for j:=x1-1 to x2+4 do

write(chr(177));

end;

win(x1-2,y1-1,x2+2,y2+1,foncol);

textcolor(textcol);

gotoxy(3,1);

for i:=1 to x2-x1+1 do

write(chr(205));

gotoxy(3,3-y1+y2);

for i:=1 to 1+x2-x1 do

write(chr(205));

for i:=1 to y2-y1+1 do

begin

gotoxy(2,i+1);

writeln(chr(186));

end;

for i:=1 to 1+y2-y1 do

begin

gotoxy(4+x2-x1,i+1);

write(chr(186));

end;

gotoxy(2,1);

write(chr(201));

gotoxy(2,y2-y1+3);

write(chr(200));

gotoxy(x2-x1+4,1);

write(chr(187));

gotoxy(x2-x1+4,y2-y1+3);

write(chr(188));

pos:=3+((x2-x1) div 2)-(length(zagl) div 2);

gotoxy(pos,1);

write(zagl);

window(x1,y1,x2,y2);

end;

(\* Процедура "Нажмите любую клавишу" \*)

PROCEDURE wait\_key;

var w\_k:char; {ожидаемая клавиша}

begin

win(1,25,80,25,white);

textcolor(black);

write(' Нажмите любую клавишу');

w\_k:=readkey;

if w\_k=#0 then w\_k:=readkey;

end;

(\* Процедура вывода "справки" \*)

PROCEDURE spravka;

begin

wind(27,3,75,13, white,black,' Справка ');

textcolor(black);

write;

WriteLn('Данная программа позволяет закодировать сооб-');

WriteLn('щение с помощью циклического кода с корректиру-');

WriteLn('ющей способностью d=4. Первые 9 символов -');

WriteLn('информационные, остальные 5 - контрольные.');

WriteLn;

WriteLn('Программа написана студентом 4 курса СФ МЭИ(ТУ)-');

WriteLn('Власовым А.В. в качестве приложения к выпускной');

writeln('работе.');

wait\_key;

writeln;

win(1,1,80,24,cyan);

end;

(\* Процедура вывода помощи-используемые клавиши \*)

PROCEDURE helper;

begin

wind(9,4,59,15,white,black,' Помощь ');

textcolor(0);

writeln('Используемые клавиши:');

writeln;

writeln(' F1 - помощь');

writeln(' Esc - отмена, выход');

writeln(' "Пробел" - ввод символа кода : [0,1]');

writeln(' BackSpace - Удаление предыдущего символа');

writeln;

wait\_key;

win(1,1,80,24,cyan);

end;

(\*Процедура вывода информации об авторе \*)

PROCEDURE avtor;

begin

wind(16,7,60,15,white,black,' Об авторе ');

textcolor(0);

writeln;

writeln(' Студент : Власов А.В.');

writeln(' Группа : ВМ-2-00');

writeln(' Руководитель : Каевченко М.А.');

writeln;

writeln;

writeln(' Смоленск 2004 г.');

wait\_key;

win(1,1,80,24,3);

end;

(\* Процедура вывода подсказки в нижней строке \*)

PROCEDURE vnizu;

begin

win(1,25,80,25,white);

textcolor(black);write(' ',chr(24),chr(25),' │ ',chr(27),chr(26),' │ ');

textcolor(red);write('Enter ');

textcolor(black);write('Выбор │ ');

textcolor(red);write('F1 ');

textcolor(black);write('Помощь │ ');

textcolor(red);write('Esc ');

end;

(\*Процедура выхода из программы \*)

PROCEDURE final(var from:byte);{номер пункта меню, на котором находились}

var n\_p:byte;{номер позиции в меню выхода}

i:integer; {счетчик}

begin

win(4,from+2,20,from+2,white);

textcolor(black);

write(menu\_p[from]);

win(4,6,19,6,3);

textcolor(white);

write(' ВЫХОД');

n\_p:=1;

repeat

repeat

vnizu;textcolor(black);write('Отмена выхода');

wind(29,10,42,11,white,black,'');

for i:=1 to 2 do

begin

if i=n\_p then

begin

textbackground(3);

textcolor(white);

end

else begin

textbackground(white);

textcolor(black);

end;

if i=2 then write(menu\_p[8])

else writeln(menu\_p[7]);

end;

key:=readkey;

if key=#0 then key:=readkey;

case key of

#80:begin {Вниз}

n\_p:=n\_p+1;

if n\_p>2 then n\_p:=1;

end;

#72:begin {Вверх}

n\_p:=n\_p-1;

if n\_p<1 then n\_p:=2;

end;

#27,#75:begin {Esc}

n\_p:=2;

break;

end;

end;

until (key=#13) or (key=#77);

case n\_p of

1:begin

cursor(on);

textcolor(lightgray);

win(1,1,80,25,0);

halt;init;

end;

2:begin

win(1,1,80,25,3);

exit;

end;

end;

until false;

end;

(\*Процедура вывода меню для пункта "О программе" \*)

PROCEDURE o\_progr;

var n\_p:byte;{номер позиции в меню выхода}

i:integer; {счетчик}

begin

n\_p:=1;

repeat

repeat

vnizu;textcolor(black);write('Выход');

wind(26,9,37,10,white,black,'');

for i:=1 to 2 do

begin

if i=n\_p then

begin

textbackground(3);

textcolor(white);

end

else begin

textbackground(white);

textcolor(0);

end;

if i=2 then write(menu\_p[6])

else writeln(menu\_p[5]);

end;

key:=readkey;

if key=#0 then key:=readkey;

case key of

#80:begin {Вниз}

n\_p:=n\_p+1;

if n\_p>2 then n\_p:=1;

end;

#72:begin {Вверх}

n\_p:=n\_p-1;

if n\_p<1 then n\_p:=2;

end;

#27,#75:begin {Esc}

win(1,1,80,24,3);

exit;

end;

end;

until (key=#13) or (key=#77);

case n\_p of

1:begin {справка}

spravka;

exit;

end;

2:begin {отмена выхода}

avtor;

exit;

end;

end;

until false;

end;

(\* процедура выхода \*)

Procedure Quit;

begin

clrscr;

cursor(off);

init;

n\_pun:=1;

win(1,1,80,25,3);

repeat

repeat

vnizu;

textcolor(0);

write('Выход');

wind(4,3,20,6,white,0,'');

for i:=1 to 4 do

begin

if i=n\_pun then

begin

textbackground(3);

textcolor(white);

end

else begin

textbackground(white);

textcolor(0);

end;

if i=4 then write(menu\_p[4])

else writeln(menu\_p[i]);

end;

from:=4;

key:=readkey;

if key=#0 then key:=readkey;

case key of

#27:begin {Esc}

from:=n\_pun;

n\_pun:=4;

break;

end;

#59:helper; {F1}

#80:begin {Вниз}

n\_pun:=n\_pun+1;

if n\_pun>4 then n\_pun:=1;

end;

#72:begin {Вверх}

n\_pun:=n\_pun-1;

if n\_pun<1 then n\_pun:=4;

end;

end;

until (key=#13) or (key=#77);

case n\_pun of

{1:visio;}

2:helper; {помощь}

3:o\_progr; {о программе}

4:final(from); {выход}

end;

until false;

end;

(\* процедура ввода кода \*)

Procedure Input;

Begin

X:=2;

GotoXY(Begin\_Line+x,CodeLine-1);

Write('k9k8k7k6k5k4k3k2k1m5m4m3m2m1');

For i:=1 to k Do

Begin

GotoXY(Begin\_Line+X+2\*(i-1),CodeLine);

If Code[i]Then Write('1')Else Write('0');

End;

TextBackGround(MenuColor);

GotoXY(Begin\_Line+X,CodeLine);

If Code[x div 2]Then Write('1') Else Write('0');

Repeat

Key:=ReadKey;

Case Key of

#0:Begin

Key:=ReadKey;

TextBackGround(GroundColor);

GotoXY(Begin\_Line+X,CodeLine);

If Code[x div 2]Then Write('1') Else Write('0');

Case Key of

#77:If X<2+(k-1)\*2 Then X:=X+2 else x:=2;

#75:If X>2 Then X:=X-2 else x:=2+(k-1)\*2;

End;

TextBackGround(MenuColor);

GotoXY(Begin\_Line+X,CodeLine);

If Code[x div 2]Then Write('1') Else Write('0');

End;

#27:quit;

{begin clrscr; cursor(off);init; final(from); end;}

#32:Begin

Code[x div 2]:=Not Code[x div 2];

GotoXY(Begin\_Line+X,CodeLine);

If Code[x div 2]Then Write('1') Else Write('0');

End;

End;

Until Key=#13;

TextBackGround(GroundColor);

GotoXY(Begin\_Line+X,CodeLine);

If Code[x div 2]Then Write('1') Else Write('0');

End;

(\* процедура кодирования сообщения \*)

Procedure Coder;

Begin

For i:=1 to n do G\_Code[i]:=0;

For i:=1 to k do Begin

If Code[k+1-i] Then Begin

For j:=1 to n Do Begin

G\_Code[j]:=G\_Code[j]+a[i,j];

If g\_Code[j]=2 Then G\_Code[j]:=G\_Code[j]-2;

End;

End;

End;

For i:=1 to n Do Begin

GotoXY(Begin\_Line+2\*i,G\_CodeLine);

Write(G\_Code[i]);

End;

End;

(\* процедура сдвига кода влево \*)

procedure Left;

begin

x:=G\_Code[1];

for i:=1 to n-1 do G\_Code[i]:=G\_Code[i+1];

G\_Code[14]:=x; {!!!!!!!!!!15!!!!!!111}

end;

(\* процедура сдвига кода вправо \*)

Procedure Shift\_Rigth;

Begin

x:=G\_Code[14];{!!!!!!!!!!!15!!!!!!!!1}

For i:=n downto 2 do G\_Code[i]:=G\_Code[i-1];

G\_Code[1]:=x;

End;

(\* процедура декодирования \*)

Procedure decoder;

Var b:Array[1..n] of byte;

w,AmountShift:byte;

Begin

AmountShift:=0;

Repeat

w:=0;

For i:=1 to n Do b[i]:=G\_Code[i];

For i:=1 to n-stepen+1 Do

Begin

If b[i]=1 Then

Begin

For j:=i to stepen+i do

Begin

b[j]:=b[j]+Polynom[j-i+1];

if b[j]=2 Then

b[j]:=0;

End;

End;

End;

For i:=1 to n do If b[i]=1 Then Inc(w);

If w>s Then

Begin

Left;

Inc(AmountShift);

End

Else

Begin

For i:=1 to n Do

Begin

G\_Code[i]:=G\_Code[i]+b[i];

If G\_Code[i]=2 then G\_Code[i]:=0;

End;

w:=0;

End;

Until (w=0) Or (AmountShift=n);

If w=0 Then

Begin

While AmountShift>0 Do

Begin

Dec(AmountShift);

Shift\_Rigth;

End;

For i:=1 to n Do

Begin

GotoXY(Begin\_Line+2\*i,D\_CodeLine);

write(G\_Code[i]);

End;

End

Else

Begin

GotoXY(Begin\_Line+2,D\_CodeLine);

Write('Невозможно расшифровать комбинацию');

End;

End;

(\* процедура изменения кода \*)

Procedure Change\_Code;

begin

x:=2;

For i:=1 to n Do Begin

GotoXY(Begin\_Line+X+2\*(i-1),C\_CodeLine);

If G\_Code[i]=1Then Write('1')Else Write('0');

End;

TextBackGround(MenuColor);

GotoXY(Begin\_Line+X,C\_CodeLine);

If G\_Code[x div 2]=1Then Write('1') Else Write('0');

Repeat

Key:=ReadKey;

Case Key of

#0:Begin

Key:=ReadKey;

TextBackGround(GroundColor);

GotoXY(Begin\_Line+X,C\_CodeLine);

If G\_Code[x div 2]=1Then Write('1') Else Write('0');

Case Key of

#77:If X<2+(n-1)\*2 Then X:=X+2 else x:=2;

#75:If X>2 Then X:=X-2 else x:=2+(n-1)\*2;

End;

TextBackGround(MenuColor);

GotoXY(Begin\_Line+X,C\_CodeLine);

If G\_Code[x div 2]=1Then Write('1') Else Write('0');

End;

#27:Exit;

#32:Begin

G\_Code[x div 2]:=1-G\_Code[x div 2];

GotoXY(Begin\_Line+X,C\_CodeLine);

If G\_Code[x div 2]=1 Then Write('1') Else Write('0');

End;

End;

Until Key=#13;

TextBackGround(GroundColor);

GotoXY(Begin\_Line+X,C\_CodeLine);

If G\_Code[x div 2]=1Then Write('1') Else Write('0');

end;

(\*Процедура ввода и обработки кодового сообщения\*)

procedure visio;

Begin

Cursor(Off);

Init;

Repeat

ClrScr;

wind(2,3,78,24,white,black,' Кодирование ');

GotoXY(3,12);

TextColor(0);

GotoXY(3,codeLine);

Write('Посылаемый код :');

GotoXY(3,G\_codeLine);

Write('Закодированный код :');

GotoXY(3,C\_codeLine);

Write('Измененный код :');

GotoXY(3,D\_codeLine);

Write('Декодированный код :');

Input;

Coder;

Change\_Code;

decoder;

Repeat

Key:=ReadKey;

If Key=#27 Then exit;

Until Key=#13;

Until False;

End;

(\* ГЛАВНЫЙ МОДУЛЬ \*)

(\* Вывод меню.Вызов соответствующих модулей \*)

begin

cursor(off);

init;

n\_pun:=1;

win(1,1,80,25,3);

repeat

repeat

vnizu;

textcolor(0);

write('Выход');

wind(4,3,20,6,white,0,'');

for i:=1 to 4 do

begin

if i=n\_pun then

begin

textbackground(3);

textcolor(white);

end

else begin

textbackground(white);

textcolor(0);

end;

if i=4 then write(menu\_p[4])

else writeln(menu\_p[i]);

end;

from:=4;

key:=readkey;

if key=#0 then key:=readkey;

case key of

#27:begin {Esc}

from:=n\_pun;

n\_pun:=4;

break;

end;

#59:helper; {F1}

#80:begin {Вниз}

n\_pun:=n\_pun+1;

if n\_pun>4 then n\_pun:=1;

end;

#72:begin {Вверх}

n\_pun:=n\_pun-1;

if n\_pun<1 then n\_pun:=4;

end;

end;

until (key=#13) or (key=#77);

case n\_pun of

1:visio;

2:helper; {помощь}

3:o\_progr; {о программе}

4:final(from); {выход}

end;

until false;

end.