Министерство общего и профессионального образования РФ

Московский энергетический институт

(Технический Университет)

Филиал в городе Смоленске

Кафедра Вычислительной Техники

Расчётно-пояснительная записка к курсовой работе

по дисциплине:

" Передача информации "

Тема : " Кодирующее устройство для кода Файра".

 Преподаватель: Каевченко М. А.

 Группа: ВМ-1-97

 Студент: Иванов А. Е.

К защите:

Смоленск 2000 г.

**Аннотация.**

 Курсовая работа по курсу "Передача информации".

Преподаватель : Каевченко Михаил Антонович.

Автор работы: Иванов Алексей Евгеньевич.

Название работы: “Кодирующее устройство для кода Файра ”

Количество страниц:

Количество иллюстраций:

Цель курсовой работы: Построить математическую модель заданного корректирующего кода, найти образующую матрицу кода, технически реализовать средства для его кодирования/декодирования (на уровне принципиальной схемы).

В главе 1 расчетно-пояснительной записки выполнен обзор теоретических аспектов, связанных с циклическими кодами.

В главе 2 расчетно-пояснительной записки выполнена разработка принципиальной схемы кодирующего устройства.

В главе 3 расчетно-пояснительной записки выполнен анализ технического задания.

Глава 4 расчетно-пояснительной записки содержит описание работы кодирующего устройства.

Глава 5 расчетно-пояснительной записки обосновывается выбор элементов и узлов в принципиальной схеме.

Глава 6 расчетно-пояснительной записки содержит спецификации на разрабатываемую систему (модульная структура, описание пользовательского интерфейса, спецификация на программные модули).

В главе 7 расчетно-пояснительной записки выбран метод тестирования, приведены результаты тестирования.

Глава 8 расчетно-пояснительной записки содержит инструкцию пользователя по работе в системе и ограничение на разработанную программу.

**Введение.**

Два научных направления призваны сыграть особую роль в научно-техни-

ческом прогрессе. Это- теория систем и теория информации. Особенность указанных научных направлений состоит в их всеобщности. Действительно, теория систем и теория информации имеют прямое отношение ко всем другим наукам, к явлениям любой физической природы и ко всем видам деятельности человека. Достаточно привести такое категорическое утверждение по этому поводу: “Информация есть всеобщее свойство материи и мера организация систем”. В ходе научно-технической революции наука об информации развивалась как дисциплина, имеющая ряд направлений. 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, получившее наибольшее развитие на этапе развития и применения электронных вычислительных машин (ЭВМ) и автоматизированных систем управления (АСУ). В ряду новых дисциплин (исследование операций, системотехника, административное управление) информационные наука и техника занимают одно из базовых положений. К информационной технике относятся средства, служащие для восприятия, подготовки, передачи, переработки, хранения и представления какой-либо информации, получаемой от человека, природы, машины, вообще от какого-либо объекта наблюдения и управления. Комплексное применение этих средств приводит к созданию больших и сложных информационных систем. 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. Если материал (вещество) и энергия сравнительно полно изучены, то законы получения, преобразования и использования информации еще являются не известной областью, таящей в себе много неожиданных проявлений.

 Современные системы телемеханики лучше защищены от помех за счет более совершенных кодов, а сжатие данных позволяет увеличить объем передаваемой информации по тем же каналам связи.

 В данной работе будет рассмотрен помехозащищенный (или корректирующий) код – код Файра. Это циклический код, обнаруживающий и исправляющий пакеты ошибок. Особенности этого кода будут рассмотрены дальше.

**1. Теоретическое введение.**

**1.1. Постановка задачи.**

Построить математическую модель заданного корректирующего кода, найти образующую матрицу кода, технически реализовать средства для его кодирования/декодирования (на уровне принципиальной схемы).

Тип кода: Файра

Число передаваемых сообщений: 63

Кодирующая способность кода: bs=3 br=4

 **1.2. Понятие двоичных циклических кодов.**

 **1.2.1. Общие понятия и определения.**

 Любой групповой (*n, k*) – код может быть записан в виде матрицы, включающей  *k* линейно-независящих строк по *n* символов, и, наоборот, любая совокупность *k* линейно-независящих n-разрядных кодовых комбинаций может рассматриваться как образующая матрица некоторого группового кода. Среди всего многообразия таких кодов можно выделить коды, у которых строки образующих матриц связаны дополнительным условием цикличности.

 Все строки образующей матрицы такого кода могут быть получены циклическим сдвигом одной комбинации, называемой образующей для данного кода. Коды, удовлетворяющие этому условию, получили название **циклических.**

 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 не приводимым. Неприводимые многочлены играют роль, сходную с простыми числами в теории чисел. Неприводимые многочлены *P(X)* можно записать в виде десятичных или двоичных чисел, либо в виде алгебраического многочлена.

Многочлен в поле двоичных чисел называется неприводимым, если он делится без остатка только на себя или на единицу.

В основу циклического кодирования положено использование неприводимого многочлена *P(X),* который применительно к циклическим кодам называется образующим, генератором или производящим многочленом (полиномом) .

 **1.2.2. Методы построения циклических кодов.**

 В качестве информационных символов *k* для построения циклических кодов берут комбинацию двоичного кода на все сочетания. В общем случае, если заданную кодовую комбинацию *G(X)* умножить на образующий многочлен *P(X)* , получится циклический код, обладающий теми или иными корректирующими свойствами в зависимости от выбора *P(X).* 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, т. е. после информационных символов.

 Для этой цели удобно воспользоваться следующим методом.

1. Умножаем кодовую комбинацию *G(X)*, которую мы хотим закодировать, на одночлен  *Xm ,* имеющий ту же степень, что и образующий многочлен  *P(X).*
2. Делим произведение *G(X)Xm*  на образующий многочлен *P(X)*.

*G(X)Xm / P(x)=Q(X)+R(X)/P(X), (1.1)*

 *г*де *Q(X)* - частное от деления; *R(X)* – остаток.

 Умножая выражение (1) на *P(X)* и перенося *R(X)* в другую часть равенства, согласно правилам алгебры двоичного поля, т. е. без перемены знака на обратный, получаем

 *F(X)=G(X)P(x)= G(X)Xm+R(X). (1.2)*

Таким образом, согласно (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)* к той же группе того же кода, что и заданная комбинация *G(X)]*  на образующий многочлен *P(X).*
2. умножением заданной кодовой комбинации *G(X)* на одночлен *Xm,* имеющий ту же степень, что и образующий многочлен *P(X),* с добавлением к этому произведению остатка *R(X),* полученного после деления произведения G(X)Xm на образующий многочлен *P(X).*

Свойства образующего многочлена:

1. Все разрешенные комбинации циклического кода делятся на образующий многочлен без остака.
2. На образующий многочлен делится без остатка двучлен Xn+1.

**1.3. Технические средства кодирования для двоичных циклических кодов.**

 Основу кодирующих и декодирующих устройств двоичных циклических кодов составляют регистры сдвига с обратными связями, позволяющие осуществлять как умножение, так и деление многочленов с приведением коэффициентов по модулю 2.

 Все известные кодирующие устройства для любых типов циклических кодов, выполненные на регистрах сдвига, можно к двум типам схем согласно методам кодирования рассмотренным выше.

 С помощью схем первого типа вычисляются значения проверочных символов путем непосредственного деления многочлена *G(X)Xm* на образующий многочлен *P(X).* Это делается с помощью регистра сдвига, содержащего *n-k* разрядов (рис.1.1). В этой схеме коэффициенты кодируемого многочлена участвуют в обратной связи не через *n-k* сдвигов, а сразу с первого такта. Это позволяет устранить разрыв между информационными и проверочными символами.

Вход

g2

gn-k-1

g1

g0

К2

Выход

К1

2

1

M2

M2

M2

M2

Рис. 1.1. Кодирующее устройство для циклического кода на основе (*n-k*) – разрядного регистра сдвига.

 В исходном состоянии ключ К1 находится в положении 1, а ключ К2 замкнут. Информационные символы одновременно поступают как в линию связи, так и в регистр сдвига, где за *k* тактов образуется остаток. Затем ключ К2 размыкается, ключ К1 переходит в положение 2 и остаток поступает в линию связи.

 С помощью схем второго типа вычисляют значения проверочных символов как линейную комбинацию информационных символов, т. е. построены на основе систематических кодов. Кодирующее устройство строится на основе *k*-разрядного регистра сдвига. Выходы ячеек памяти подключаются к сумматору в цепи обратной связи в соответствии с видом генераторного многочлена

=H0 + H1X + ... + HkXk

H(X)=

G(X)

xn+1

 **1.4. Коды Файра.**

 Под пакетом ошибок длинной **b** понимают такой вид комбинации помехи, в котором между крайними разрядами, пораженными помехами, содержится **b-2** разряда.

 Коды Файра могут исправлять пакет ошибок длинной **bs** и обнаруживать пакет ошибок длинной **br** (в кодах Файра понятие кодового расстояния **d** не используются).

 Образующий многочлен кода Файра *P(X)ф*  определяется из выражения

 *P(X)ф= P(X)(Xc-1), (1.3)*

*Где P(X)* – неприводимый многочлен степени L.

Из принципа построения кода следует, что

 L ≥ **bs,** (1.4)

с ≥ **bs+ br-1** (1.5)

При этом ***с*** не должно делится нацело на число ***e,*** где

*e=2L -1 (1.6)*

 Неприводимый многочлен *P(X)* выбирают из таблицы, согласно уравнению (4), но так, чтобы удовлетворялось условие (6). Длинна слова ***n*** равна наименьшему общему кратному чисел ***c*** и ***e***, так как только в этом случае многочлен *Xn+*1 делится на *P(X)ф* без остатка:

 ***n***=НОК(***e,c***) (1.7)

Число контрольных символов

 *m=c+L (1.8)*

ВЫВОДЫ. В данной главе были рассмотрены теоретические аспекты построения двоичных циклических кодов. Также было выбрано кодирующее устройство на основе *n-k* разрядного регистра. Выяснили, что сперва необходимо найти образующий многочлен (по соответствующим формулам), а потом на основе этого многочлена строить кодирующее устройство. В последующих главах будет приведена реализация кодирующее устройство кода Файра на ЭВМ и приведена принципиальная схема кодера.

**2. Разработка схемы кодирующего устройства.**

**2.1. Построение кода Файра.**

Согласно техническому заданию и в соответствии с данными, полученными на курсовую работу образующий многочлен Файра *P(X)ф.* Согласно формуле (1.4) находим L ≥ 3 , откуда можно принять L=3. Из соответствующих таблиц выбираем неприводимый многочлен *P(X)*=x3+x+1= 1011.

 В соответствии с формулой (1.5):

 c ≥ 3+4-1 ≥ 6 , откуда можно принять с=6.

По формуле (1.6) получаем е=23-1=7. Видим, что **с** на **е** нацело не делится.

Число проверочных разрядов, подставляя в формулу (1.8) значения L и C, получим m=6+3=9 .

Тогда длинна кода в соответствии с (1.7) равна

 n = НОК(6,7) = 42

Тогда код Файра имеет вид (42,33).

Образующий многочлен Файра *P(X)ф* равен

 *P(X)ф*=(x3+x+1)(x6+1)=x9+x7+x6+x3+x+1=1011001011 (2.1)

 **2.2. Структурная схема кодирующего устройства.**

 В соответствии с многочленом (2.1) схема получения проверочных символов будет иметь 9-ти разрядный регистр с двумя ключами.

 Структурная схема кодера представлена на рис. 2.1

#### Кодер

Счетчик

## Г

 Рис 2.1. Структурная схема кодера.

ВЫВОДЫ. В данной главе получен образующий многочлен Файра, а также количество информационных символов и проверочных. По виду образующего многочлена определяется расположение сумматоров в кодирующем устройстве, а также количество разрядов в регистре сдвига. Далее будет рассмотрена работа принципиальной схемы и реализованной программы.

1. **Анализ технического задания.**

Согласно техническому заданию на курсовую работу необходимо построить математическую модель заданного корректирующего кода, найти образующую матрицу кода, технически реализовать средства для его кодирования/декодирования (на уровне принципиальной схемы), разработать многомодульную программу реализующую кодирующее устройство на ЭВМ..

 Программа должна иметь стандартный интерфейс, удовлетворяющий принятым нормам, работать правильно на всех допустимых персональных ЭВМ, текст программы должен быть удобно-читаемый и понятный пользователю.

 С целью упрощения процесса разбора текста программы, программа состоит из нескольких модулей, каждый из которых выполняет определённые функции.

 Можно выделить пять основных модуля:

* **Основная программа.**

В нём происходит обработка результатов выполнения остальных модулей программы. А именно: интерфейс, модуль ввода данных, модуль вывода данных, модуль обработки ошибок. Происходит выполнение алгоритма.

* **Модуль ввода.**

В нём происходит диалог пользователя с программой. В частности здесь происходит ввод информационных символов.

* **Модуль вывода.**

Здесь реализован вывод результатов выполнения программы в удобной для чтения форме. Результаты представлены в виде таблицы. Также здесь реализован вывод образующей матрицы.

* **Модуль обработки ошибок.**

В нём обрабатываются ошибки при вводе и реализована защита программы от сбоев.

* **Интерфейс.**

В этом блоке реализован интерфейс программы взаимодействия пользователя с ЭВМ.

Интерфейс состоит из горизонтального меню и строки статуса.

Основные пункты меню:

* **О программе.**

В этом пункте представлена краткая информация о программе.

* **Задание.**

Изложена полная постановка задачи.

* **Работа.**

Здесь реализован ввод данных и выполнение основного алгоритма, получение результатов и вывод их на экран в удобной для чтения форме.

Программа должна корректно заканчивать свою работу, без ошибочных ситуаций.

ВЫВОДЫ. В данной главе рассмотрены условия реализации программы и поставлены конкретные цели. Здесь было упоминание о модульной структуре программы, в следующей главе модульная структура будет рассмотрена подробнее. Также будут рассмотрены аспекты, связанные с реализацией пользовательского интерфейса.

###### Работа устройства.

Принципиальная схема кодирующего устройства приведена в приложении.

Данные на вход кодирующего устройства поступают начиная со старшего разряда. Информация поступает на DD12.1 сумматор по модулю 2, выполненный на ИМС К555ИП5. Как только мы включили питание на схему, то через цепочку R4, C3, DD17.2, DD17.3 происходит начальный сброс счетчиков DD10, DD11. Это получается потому что в начальный момент времени, при резком подключении питания конденсатор С3 не успевает зарядится. На нем получается логический ноль. Этот ноль через инверторы DD14.2, DD14.3, выполненные на К155ЛИ1, преобразуется в единицу и поступает на входы счетчиков РЕ, счетчик переходит в режим записи, результате чего происходит запись в счетчики данных со входов D1-D4. Но так как эти входы заземлены, то на выходы счетчика запишутся нули. Произошло начальное обнуление. По истечении короткого промежутка времени конденсатор С3 заряжается до уровня логической единицы и счетчики переходят в режим счета импульсов. Так как счетчики обнулялись то на триггер DD15.2 на инверсный вход R поступает ноль и триггер установится в ноль. Этот ноль на выходе инвертора DD14.1становится единицей и поступает на инверсный вход R триггера DD15.1. Триггер хранит свое предыдущее нулевое значение, так как вход K заземлен, а на J поступает ноль со счетчика. Тогда на элемент DD13.3 будет поступать нулевой уровень. Но DD13.3 является элементом “И”, поэтому на его выходе всегда будет ноль. С другой стороны с инверсного выхода триггера на “И” поступает логическая единица и выходной сигнал будет зависеть от входного. Эта же единица поступает на DD13.5. Элементы DD13.3, DD13.4, DD13.5 выполняют роль ключей. Когда на выходе триггера DD15.1 логический ноль, то замкнут ключи К1, а К2 находится в положении 1 [см. Теоретическое введение], сигнал со входа напрямую поступает на выход, а в регистрах формируется остаток. Если на выходе триггера DD15.1 логическая единица, то ключи К1 разомкнут, а К2 находится в положении 2 .

#####  Входной сигнал через ключ DD13.4 поступает на выход ,а через ключ DD13.5 на вход регистра DD1. Запись в регистр происходит последовательно со сдвигом влево, в соответствии с тактируемыми импульсами. В схеме девять регистров (Степень образующего многочлена). Каждый регистр выполняет задержку на один такт. Пока триггер DD15.1 находится в нуле в регистрах за K-тактов вычисляются проверочные символы.

 Триггер DD15.2 управляет работой счетчиков и триггером DD15.1.

Как только счетчики досчитали до k ,т.е. прошло k символов то DD15.1 перебрасывается в единицу, ключи DD13.5, DD13.4 открываются, а DD13.3 закрывается и из регистров на вход поступают проверочные символы за n-k тактов. Когда счетчик досчитает до n триггер DD15.2 перебросится в единицу и произойдет обнуление регистров, счетчиков и триггера DD15.1, т. е. схема готова принимать новую комбинацию.

 Временные диаграммы работы кодера приведены в приложении .

ВЫВОДЫ. В данной главе была подробна описана работа принципиальной схемы. Далее будет рассмотрена отдельные элементы принципиальной схемы и их описание.

## **5.Выбор узлов принципиальной схемы.**

## **5.1. Расчет генератора тактовых импульсов**

Генератор тактовых импульсов на ИМС DD1 К555АГ3. Принципиальная схема генератора приведена на рис.5.1. Период колебаний импульсов получаемых с автогенератора Т = 1 мс.

Длительность импульса: tи = 0.5Т = 0.5 мс.

Длительность паузы: tп = 0.5Т = 0.5 мс.

Рассчитаем параметры RC элементов генератора:

;

;

Задаваясь С1 = С2 = 0.1 мкФ, имеем:

Рис. 5.1. Принципиальная схема генератора.

R1 = 5 кОм; R2 = 5 кОм.

**5.2. Счетчик К555 ИЕ10**

Для подсчета числа импульсов используем два счетчика К555 ИЕ10.

Микросхема К555 ИЕ10 - это синхронный, четырехразрядный, реверсивный, двоично-десятичный счетчик с предварительной записью и выходом переноса. Условное обозначение и цоколевка даны на рис. 5.2.

Рис.5.2 Счетчик К555 ИЕ10.

Особенностью данного счетчика является его построение по синхронному принципу, т. е. все триггеры переключаются одновременно от одного тактового импульса. Счетчик запускается положительным перепадом тактового импульса, подаваемым на вход синхронизации С. Сброс всех триггеров в нулевое состояние асинхронный по общему входу сброса R, т. е. осуществляется напряжением низкого уровня, независимо от сигналов действующих на входах С, РЕ, СЕР, СЕТ.

 Счетчик устанавливается в предварительное состояние при наличии на входе параллельной загрузки РЕ напряжения низкого уровня. В этом случае разрешена подача сигналов на триггер через входы параллельной установки D0 - D3 в момент прихода положительного перепада на вход С.

Для синхронного каскадирования микросхема имеет два входа разрешения СЕР и СЕТ, а также выход ТС (окончания счета). Счетчик считает тактовые импульсы если на входах СЕТ и СЕР напряжения высокого уровня.

Вход СЕТ последующего счетчика соединяется с выходом ТС предыдущего. На выходе ТС появится напряжение высокого уровня, если выходной код счетчика 1111=15, а на входе СЕТ присутствует напряжение высокого уровня. Состояния счетчика приведены в таблице 1.

Таблица 1.

|  |  |  |
| --- | --- | --- |
| Режим работы | Входы | Выходы |
| R | C | CЕР | СЕТ | РЕ | Dn | Qn | ТС |
| Сброс | 0 | Х | Х | Х | Х | Х | 0 | 0 |
| Параллельная загрузка | 1 | ↑ | Х | Х | 0 | 0 | 0 | 0 |
| 1 | ↑ | Х | Х | 0 | 1 | 1 | 1 |
| Счет | 1 | ↑ | 1 | 1 | 1 | Х | Счет | 1 |
| Хранение | 1 | Х | Х | 0 | 1 | Х | Qn | 1 |
| 1 | Х | 0 | Х | 1 | Х | Qn | 1 |

**5.3. Триггер К555 ТВ6**

Микросхема К555 ТВ6 содержит два J-K триггера с общим входом питания и изображена на Рис .5.3 . Вход синхронизации С у всех триггеров инверсный динамический, поэтому данные со входов J и K переносятся на выход Q по отрицательному перепаду импульса С. Когда импульс переходит от высокого уровня к низкому сигналы на входах J и K не должны меняться. Состояния для триггера приведены в таблице 2 .

**Рис. 5.3 . Триггер К555 ТВ6.**

Таблица 2. Состояния триггера К555 ТВ6.

|  |  |  |
| --- | --- | --- |
| Режим работы | Входы | Выходы |
|  R |  J | K | C |  |  |
| Асинхронный сброс | 0 | Х | Х | Х | 0 | 1 |
| Переключение | 1 | 1 | 1 |  | =1 | =0 |
| Хранение | 1 | 0 | 0 |  | 1 | 0 |
| Загрузка 1 | 1 | 1 | 0 |  | 1 | 0 |
| Загрузка 0 | 1 | 0 | 1 |  | 0 | 1 |

**5.4. Регистр К155ИР13.**

Микросхема К155ИР13 является универсальным восьмиразрядным регистром сдвига с большими функциональными возможностями. Регистр может работать в следующих режимах: последовательного ввода информации с о сдвигом вправо;

последовательного ввода информации с о сдвигом влево; параллельного ввода; хранение; установка нулей (очистка, сброс).

 Микросхема имеет следующие выводы (рис. 5.4): информационные входы последовательного ввода информации – DR при сдвиге вправо и DL при сдвиге влево; восемь входов D0-D7 для параллельного ввода; тактовый вход С; управляющие входы S0 и S1 для выбора режима; вход R для установки триггеров в нулевое состояние и восемь выходов от разрядов Q0-Q7.

#### Рис. 5.4. Регистр К155ИР13.

 Работа регистра в режиме последовательного ввода со сдвигом вправо происходит при S1=0 и S0=1. Информация в последовательном коде подается на вход, начиная с младших разрядов. Ввод и сдвиг всего числа на один разряд происходит с каждым перепадом 0,1 тактовых импульсов.

Последовательный ввод со сдвигом влево осуществляется при S1=1 и S0=0. Входная информация должна поступать на вход DL со старших разрядов.

Для параллельного ввода со входов D0-D7 на обоих управляющих входах должно быть S1=S0=1. Информация со входов D0-D7 будет записана в триггеры и появится на выходах Q0-Q7 по перепаду 0,1 тактового импульса.

Во избежание сбоев, смена состояний управляющих входов S1 и S0 должна происходить при С=1. Когда на обоих управляющих входах S1= S0=0, триггеры не переключаются, т. е. имеет место режим хранения. Установка нулей (очистка регистра ) осуществляется импульсом U0 на входе R. Очистка регистра происходит независимо от состояния остальных входов. Во время действия импульса R=0 регистр бездействует. При выполнении всех остальных операций необходимо поддерживать R=1. Работа регистра приведена в таблице 3.

Таблица 3.

|  |  |  |
| --- | --- | --- |
| Входы | Выходы | Режим работы |
| R | C | S0 | S1 | DR | DL | Di | Q0 | Q1-Q6 | Q7 |
| 1 | ↑ | 1 | 1 | 1 | X | Dni | Dn0 | Dn1... Dn6 | Dn7 | Параллельный ввод. Хранение  |
| 1 | ↑ | 0 | 0 | X | X | X | Qn0 | Qn1... Qn6 | Qn7 |
| 1 | ↑ | 1 | 0 | X | 0 | X | Qn1 | Qn2... Qn7 | 0 | Сдвиг влево |
| 1 | ↑ | 1 | 0 | X | 1 | X | Qn1 | Qn2... Qn7 | 1 |
| 1 | ↑ | 0 | 1 | 0 | X | X | 0 | Qn0... Qn5 | Qn6 | Сдвиг вправо |
| 1 | ↑ | 0 | 1 | 1 | X | X | 1 | Qn0... Qn5 | Qn6 |
| 0 | x | x | x | X | x | X | 0 | 0 ... 0 | 0 | Сброс |

ВЫВОДЫ. В данной главе были описаны основные элементы, применяемые при реализации принципиальной схемы кодера. Также была дана их краткая характеристика и способы включения элементов. Все используемые элементы -ТТЛ. В следующих главах будет рассмотрена программная реализация кодирующего устройства (его математическая модель).

**6. Описание разработки системы.**

6.1. Модульная структура системы.

Как уже говорилось во второй главе программа состоит из пяти модулей:

* Основная программа.
* Модуль ввода.
* Модуль вывода.
* Интерфейс.
* Модуль обработки ошибок.

 Модульная структура программы представлена на рисунке 6.1

**Интерфейс**

**Модуль ввода**

**Основная программа**

Модуль вывода

**Обработка ошибок**

Рис 6.1 Модульная структура программы.

Пользователь ведёт диалог с программой через интерфейс. От интерфейса управление передаётся в **Основную программу**, где анализируются команды пользователя и выполняется передача управления соответствующим модулям. Если пользователь выбрал пункт меню <Работа> и пункт подменю <Ввод>, то управление от **Основной программы** передаётся в **Модуль ввода** и дальше пользователь ведет диалог с этим модулем. Как только нажата клавиша <Enter>, то управление берёт  **Модуль обработки ошибок**. Данный модуль проверяет выполнение ограничений и выявляет ошибки при вводе. Если ошибок нет, то управление передаётся в **Основную программу** и дальше в **Модуль вывода.** А иначе пользователю предоставляется шанс ещё раз ввести данные.

**6.2. Описание пользовательского интерфейса.**

**6.2.1. Меню системы.**

Интерфейс системы выполнен в графическом режиме с соблюдением всех установленных ГОСТОВ. Экран разбит на три части: верхняя часть - горизонтальное меню с выпадающими окнами; нижняя часть - строка состояния; средняя часть - окно диалога.

Горизонтальное меню состоит из трёх пунктов:

* Работа.

В этом пункте предусмотрено выпадающее меню для облегчения поиска соответствующего пункта. В подменю находятся следующие пункты:

* Ввод.
* Вывод
* Выход.
* О программе.
* Помощь.

Передвижение по меню осуществляется с помощью клавиш управления курсором. Каждый пункт меню активизируется при нажатии клавиши <Enter>. Обработку пунктов меню осуществляет Основная программа. В неё передаётся управление и соответствующие координаты пунктов на выполнение.

В пункте меню <Работа>, в подпункте <Ввод > осуществляется ввод данных, необходимых для работы программы. При этом в **Модуле ввода** предусмотрено редактирование вводимого результата. В подпункте <Вывод> происходит выполнение алгоритма и вывод образующей матрицы на экран в удобном для визуализации виде.

В подпункте <Выход> происходит предупреждение пользователя о выходе из программы с соответствующими запросами.

В пункте <О программе > дана краткая аннотация программы.

В пункте <Помощь> представлена помощь по обработке пунктов меню и по соответствующим клавишам.

**6.2.2. Обработка ошибочных ситуаций.**

 В программе предусмотрена защита от ввода нестандартных данных, а также защита от нажатия различных клавиш. Для обработки этих ситуаций предусмотрен **Модуль обработки ошибок.** Пользователь может вводить только 0 или 1.

 Если пользователь ввел по ошибке неправильные данные, то у него есть возможность исправиться на стадии ввода. Для этих целей в программе предусмотрен повторный ввод с предыдущими данными.

**6.3. Спецификация на программные модули.**

 **Процедура initgrf.**

Входные параметры:Нет.

Выходные параметры:Нет.

### Выполняемые функции:Инициализация графики.

### Особенностей:Нет.

 **Функция**  Sum(F,P : Byte) : Byte;

Входные параметры: F,P : Byte

Выходные параметры: Sum

 Выполняемые функции: Суммирование по модулю 2

### Особенностей:Нет.

 **Процедура** Dopoln(Var F : Mass1);

Входные параметры: F : Mass1

Выходные параметры: F : Mass1

 Выполняемые функции: Умножение на старшую степень

 образующего многочлена .

### Особенностей: НЕТ .

 **Процедура** Delenye(F : Mass1;P : Mass2);

 Входные параметры: F : Mass1;P : Mass2

### Выходные параметры: нет.

 Выполняемые функции: Деление многочлен на многочлен

### Особенностей:Имеет свои особенности.

 **Процедура** Ed\_Matrix(Var A : Two\_Matrix);

 Входные парамеры: Var A : Two\_Matrix

### Выходные параметры: Var A : Two\_Matrix

### Выполняемые функции: Составление единичной матрицы.

### Особенности: нет.

 **Процедура** Obr\_Matrix(Var A : Two\_Matrix);

 Входные параметры A : Two\_Matrix

Выходные параметры: A : Two\_Matrix

 Выполняемые функции: Получение образующей матрицы

### Особенностей: нет.

 **Процедура** Visual(Var sa:mass);

Входные параметры: Var sa:mass

 Выходные параметры: Var sa:mass

 Выполняемые функции: Ввод информационных символов.

### Особенностей: нет

 **Процедура** OutPutObr\_Matrix(x,y : Integer;Obr\_Matr :

 Two\_Matrix );

 Входные параметры: x,y : Integer;Obr\_Matr :

 Two\_Matrix );

Выходные параметры:Нет.

### Выполняемые функции: Вывод образующей матрицы.

### Особенностей:Является универсальной.

 **Процедура** OutPut(x,y : Integer;F,A : Mass1);

Входные параметры: x,y : Integer;F,A : Mass1.

### Выходные параметры:Нет.

Выполняемые функции: Вывод полученной кодовой

комбинации.

### Особенностей: нет.

 **Функция** \_Exit(Fon,Color : Integer;Col\_Simv : Byte) :

 integer;

### Входные параметры Fon,Color : Integer;Col\_Simv : Byte

### Выходные параметры: \_Exit : integer.

Выполняемые функции: выход из программы

### Особенностей: нет

 **Основная программа**

Входные параметры:нет.

### Выходные параметры:Нет.

Выполняемые функции:Обьединяет в себя все процедуры и

управляет работой.

### Особенностей:Нет.

ВЫВОДЫ. В данной главе были описаны принципы разработки программы и интерфейса. Дана модульная структура программы и межмодульные связи. Описан принцип работы программы и дана спецификация на программные модули. Если это всё объединить, то нетрудно будет разобраться в отдельных деталях программы. Дальше будет рассмотрено одно из основных мероприятий при программировании - тестирование.

**7. Тестирование.**

**7.1. Выбор методики тестирования.**

 Без тестирования программ невозможно создать правильно работающую программу, так как всегда существуют какие-то граничные условия, за пределами которых программа даёт ошибки. При этом не следует путать отладку программ и тестирование.

Отладка программ - используется при неправильной работе программы.

Тестирование - ищет ошибки.

Существует много методов тестирования:

1. *Статическое тестирование* является наиболее формализованным и автоматизируемым методом проверки программ. В качестве эталонов применяются правила структурного построения программных модулей и обработки данных. Проверка степени этих правил проводится без использования объектного кода программы путем формального анализа текста программы на языке программирования. Операторы и операнды текста программ при этом анализируется в символьном виде, поэтому такой метод называют символьным тестированием.
2. *Детерминированное тестирование* является наиболее трудоёмким и детализирующим. При детерминированном тестировании контролиру-

 естся каждая комбинация исходных эталонных данных и соответствующая ей комбинация результатов функционирования программы. Это позволяет выявлять отклонение результатов от эталона и реализующих данных, при которых это отклонение произошло.

1. *Стохастическое тестирование* применяется в тех случаях, когда невозможно в сложных задачах перебрать все комбинации исходных данных и проконтролировать результаты функционирования программы на каждом из них. При этом исходные тестовые данные задаются множеством случайных величин с соответствующими распределениями и для сравнения полученных результатов используются также распределения случайных величин. Стохастическое тестирование применяется в основном для обнаружения ошибок.
2. *Тестирование в реальном масштабе времени.* В процессе такого тестирования проверяется исполнение программ и обработка исходных данных с учетом времени их поступления , длительности и приоритетности обработки , динамика использования памяти и т.д.
3. При *восходящем тестировании* прежде всего проверяются модули нижних иерархических уровней, к которым постепенно подключаются вызывающие их модули. При этом обеспечивается работоспособность вызываемых компонент и функции группы программ проверяются в их естественном исполнении. Основные трудности состоят в необходимости полного обновления тестовых наборов при подключении каждой новой программы более высокого уровня.
4. При *нисходящем тестировании*  проверки начинаются с программ управления и организации вычислительного процесса. Первоначально тестируется управляющее ядро комплекса программ и программы решения функциональных задач, размещенных на высших иерархических уровнях. К ним постепенно подключаются для тестирования программы последующих более низких иерархических уровней. Преимуществом такого метода является возможность сохранения и развития наборов тестовых данных по мере подключения программ нижних уровней.

На практике обычно используются три стратегии тестирования:

1. Тестирование программ как "Чёрного ящика ". Имеет цель выяснения обстоятельств, в которых поведение программ не соответствует спецификации. Тестовые данные составляются и используются без учета знаний о внутренней структуре программы.
2. Тестирование программ как "Белого ящика ". Тестовые наборы данных проектируются на основе внутренней логики программы. Цель тестирования - проверить каждую ветвь, каждый путь и каждый оператор. Спецификация программы при этом не используется.
3. Реальная стратегия обычно сочетает оба метода. При помощи метода "чёрного ящика" делается:
* Проверка в нормальных условиях.
* Анализ граничных значений.
* Проверка в исключительных ситуациях.
* Предположение об ошибке.

При помощи метода "Белого ящика" делается:

* Покрытие операторов.
* Покрытие решений.
* Покрытие условий.

**7.2. Результаты тестирования**

При тестировании программы был использован восходящий метод тестирования. С помощью этого метода сначала были протестированы отдельные модули программы, а затем и вся программа. Результаты тестирования показаны на рисунке Приложения.

Тестирование системы включало в себя:

- тестирование ввода различных набора данных;

- получение комбинаций для кода с любым образующим многочленом;

Тестирование ввода различных наборов кодовых комбинаций не показало ни одной исключающей ситуации.

ВЫВОДЫ. В данной главе был выбран метод тестирования системы. В результате тестирования были обнаружены ошибки, которые впоследствии были устранены. В результате тестирования было получено, что программа является работоспособной. Программа правильно находит образующую матрицу и строит код Файра..

**8. Ограничения на разработку.**

Для выполнения программы необходимо соблюдать некоторые требования, а именно:

1. Программа может работать на персональных компьютерах с процессором не ниже 80486 .
2. Для реализации интерфейса необходимо иметь цветной монитор с адаптером не ниже VGA.
3. RAM не ниже 8MB.
4. Windows 95/98/NT

**8.1. Инструкция пользователю.**

 В программе разработан интерфейс, подходящий под стандарты. Интерфейс реализован в графическом режиме. Меню горизонтальное, с выпадающими окнами. Главное окно программы изображено на рис.. Приложения 2.

Главное меню программы содержит следующие пункты:

1. **О программе -** дана краткая аннотация выполненной программы.
2. **Работа** - содержит подменю с пунктами:
3. **Ввод** - позволяет пользователю ввод соответствующих данных. Здесь предусмотрена защита от некорректного ввода. В этом пункте можно ввести любую информационную комбинацию.
4. **Вывод** - выполняется алгоритм построения образующей матрицы, с последующем выводом ее на экран.
5. **Выход** - обеспечивает выход из программы. Используется защита от случайного выхода.

Пункты меню активизируются нажатием клавиши <Enter>.

При появления каждого окна в строке статуса появляется сообщение о возможных действиях.

Передвижение по меню осуществляется клавишами перемещения курсора на клавиатуре. Также используется "горячая клавиша" Esc. При нажатии на неё происходит автоматический выход из программы.

Заключение

В данной работе был рассмотрен один из множества методов кодирования. Достоинство данного метода в том, что он может находить исправлять пачки ошибок. Также все циклические коды легко поддаются схемной реализации.

Все элементы принципиальной схемы имеют серии К155 или К55. Это одни из самых распространенных и довольно не дорогих отечественных микросхем, легко доступная любому пользователю.

Программа реализована на языке Turbo Pascal 7.0. Это наиболее распространенный на сегодняшний день язык программирования. Поэтому в программе разобраться трудностей не будет.

Кодирование информации наиболее актуальна в наши дни в связи с развитием компьютерных сетей. Необходимо передавать информацию быстро и точную.

 Программа тестировалась восходящим методом. С помощью этого метода сначала были протестированы отдельные модули программы, а затем и вся программа.

Проведено тестирование программы необходимое для проверки работоспособности программы, которое показало, что программа устойчиво работает на все режимах работы и выполняет поставленную задачу.

Программа является универсальной и может быть усовершенствована любым пользователем, разбирающимся в языке программирования Pascal 7.0 .

Дана общая модульная структура программа, которая может помочь ползо-

вателю разобраться в общей структуре программы и усовершенствовать программу.

**Список литературы.**

1. Тёмников Ф. Е..

Теоретические основы информационной техники:

Учеб.пособие для вузов-2-е изд., перераб. и доп.-М.:Энергия, 1979.-

 512 с., ил.

2. Липаев В.В.

Тестирование программ.- М.: Радио и связь,1986. - 296 с.: ил.

3. Тутевич В. Н.

Телемеханика:

Учеб.пособие для студентов вузов спец. “Автоматика и

телемеханика”-2-е изд., перераб. и доп.-М.: Высш. Шк., 1985.- 423 с., ил.

4. Зельдин Е. А.

Цифровые интегральные микросхемы в информационн-измерительной

аппаратуре. –Л.: Энергоатомиздат. Ленингр. отд-ние, 1986.- 280 с.: ил.

**Приложения.**

**Приложение 1. Перечень элементов**

|  |  |  |  |
| --- | --- | --- | --- |
| Поз.Обозн. | Наименование | Кол |  Примечания |
|  | Конденсаторы |  |  |
| С3 | К10-60 – 10 мКф + 10% | 1 |  |
|  С1,С2 | К10-17а – 0.1 мКф + 0.1% | 2 |  |
|  | Микросхемы. |  |  |
|  | К555АГ3 | 1 |  |
|  | К155ИР13 | 9 |  |
|  | К555ТВ6 | 1 |  |
|  | К555ИЕ10 | 2 |  |
|  | К155ЛН1 |  |  |
|  | К155ЛИ1 |  |  |
|  | Резисторы |  |  |
| R1-R2 | С2-29В 5Ком±0.1% | 2 |  |
| R4 | МЛТ-0.125 – 30 Ком + 10% | 1 |  |
| R5 | МЛТ-0.125 – 1.5Koм + 10% | 1 |  |

**Приложение 2. Блок-схема алгоритма.**

**Приложение 3. Текст програмных модулей.**

Uses Crt,Graph,AlexUnit;

Const

 \_N = 33;

 \_M = 10;

Type

 Delim\_Mas = array[1 .. 100] of byte;

 Mass1 = array[1 .. \_N+\_M-1] of byte;

 Mass2 = array[1 .. \_M] of byte;

 Mass3 = array[1 .. \_M-1] of byte;

 Mas\_Exit=array[1..2] of String;

 Two\_Matrix=array[1 .. \_N,1 .. \_N+\_M-1] of byte;

Const

 P : Mass2 = (1,0,1,1,0,0,1,0,1,1);

 Y\_No : Mas\_Exit = ('Да','Нет');

Var

 F,Cicle\_Kod : Mass1; R : Mass1;

 Delimoe : Delim\_Mas;

 Obraz\_Matrix : Two\_Matrix;

 Mas : Mass;

 grDriver,grMode,ErrCode: Integer;

 flag : boolean;

 \_t,c,n,m,i,schot,N0,Code : integer;

Function Sum(F,P : Byte) : Byte; {Суммирование по модулю 2}

 Var

 i : Byte;

 Begin

 If ((F=1) and (P=1)) or ((F=0) and (P=0)) Then Sum:=0

 Else Sum:=1;

 End;

{-------------------------------}

Procedure Dopoln(Var F : Mass1); {Умножение на старшую степень образующего многочлена}

 Var

 i : Byte;

 Begin

 for i:=\_N+1 to \_N+\_M-1 do F[i]:=0;

 End;

{--------------------------------}

Procedure Delenye(F : Mass1;P : Mass2); {Деление многочлен на многочлен}

 Var

 i,j,t : Byte; K : Mass1;

 Begin

 For i:=1 to \_N do

 Begin

 IF F[i]=1 Then Begin

 t:=1;

 For j:=i to i+\_M-1 do

 Begin

 K[j]:=Sum(F[j],P[t]);

 F[j]:=K[j];

 t:=t+1;

 End;

 End;

 End;

 t:=1;

 For i:=\_N+1 to \_N+\_M-1 do

 Begin

 R[t]:=F[i];

 t:=t+1;

 End;

 End;

{----------------------------------------}

Procedure Ed\_Matrix(Var A : Two\_Matrix); {Составление еденичной матрицы}

Var

 i : Integer;

 Begin

 For i:=1 to \_N do

 Begin

 A[i,\_N+1-i]:=1

 End;

 End;

{----------------------------------------}

Procedure Obr\_Matrix(Var A : Two\_Matrix); {Получение образующей матрицы}

 Var

 i,j,t,Schot,l,m : Byte; K : Mass1;

 Begin

 Delimoe[1]:=1;

 Schot:=1;

 For i:=1 to \_N do

 Begin

 IF Delimoe[i]=1 Then Begin

 t:=1;

 For j:=i to i+\_M-1 do

 Begin

 K[j]:=Sum(Delimoe[j],P[t]);

 Delimoe[j]:=K[j];

 t:=t+1;

 End;

 l:=1;

 For m:=i+1 to i+\_M-1 do

 Begin

 A[Schot,\_N+l]:=Delimoe[m];

 l:=l+1;

 End;

 End

 Else Begin

 l:=1;

 For m:=i+1 to i+\_M-1 do

 Begin

 A[Schot,\_N+l]:=Delimoe[m];

 l:=l+1;

 End;

 End;

 Schot:=Schot+1;

 End;

 End;

Procedure InitGrf; {Инициализация графики}

Begin

 grDriver := Detect;

 InitGraph(grDriver, grMode,'c:\sub\bp\bgi');

 if GraphResult <> grOk then Begin

 Halt(1);

 End;

End;

Procedure Visual(Var sa:mass); {Ввод информационных символов}

 Var i,x,y,k,Fon,Color,a:Integer;

 Code: Integer;

 ch,chi:Char;

 Stop:Boolean;

 Elm : String;

 Begin

 moveto(20,465);

 SetTextJustify(0,1);

 SetColor(0);

 outtext('Нажмите Enter');

 x:=30;

 y:=225;

 k:=17;

 Fon:=1;

 Color:=14;

 Window\_(30,150,610,260,1,'Ввод нформационных символов');

 SetTextStyle(1,0,4);

 SetColor(Fon);

 For i:=1 to \_N Do

 Begin

 MoveTo(x+i\*k,y);

 Str(Sa[i],Elm);

 OutText(Elm);

 End;

 SetTextStyle(0,0,1);

 MoveTo(x+17,y-20);

 Str(\_N,Elm);

 OutText(Elm);

 MoveTo(x+\_N\*17,y-20);

 Str(0,Elm);

 OutText(Elm);

 SetTextStyle(1,0,4);

 i:=1;

 MoveTo(x+i\*k,y);

 Setcolor(Color);

 Str(Sa[i],Elm);

 OutText(Elm);

 Stop:=False;

 Repeat

 Begin

 ch:=ReadKey;

 Case ch Of

 #75:Begin

 Setcolor(Fon);

 MoveTo(x+i\*k,y);

 Str(Sa[i],Elm);

 OutText(Elm);

 i:=i-1;

 if(i<1)then i:=\_N;

 SetColor(Color);

 MoveTo(x+i\*k,y);

 Str(Sa[i],Elm);

 OutText(Elm);

 Stop:=False;

 End;{влево}

 #77:Begin

 SetColor(Fon);

 MoveTo(x+i\*k,y);

 Str(Sa[i],Elm);

 OutText(Elm);

 i:=i+1;

 if(i>\_N)then i:=1;

 SetColor(Color);

 MoveTo(x+i\*k,y);

 Str(Sa[i],Elm);

 OutText(Elm);

 Stop:=False;

 End;{вправо}

 '1':Begin

 SetColor(7);

 MoveTo(x+i\*k,y);

 Str(Sa[i],Elm);

 OutText(Elm);

 SetColor(Color);

 MoveTo(x+i\*k,y);

 sa[i]:=1;

 F[i]:=Sa[i];

 Str(Sa[i],Elm);

 OutText(Elm);

 Stop:=False;

 End;

 '2':Begin

 SetColor(7);

 MoveTo(x+i\*k,y);

 Str(Sa[i],Elm);

 OutText(Elm);

 SetColor(Color);

 MoveTo(x+i\*k,y);

 sa[i]:=0;

 F[i]:=Sa[i];

 Str(Sa[i],Elm);

 OutText(Elm);

 Stop:=False;

 End;

 #13:Begin Stop:=True; End;

 End;

 End;

 Until(Stop);

 SetTextStyle(0,0,1);

 End;

{---------------------------------------}

{Вывод образующей матрицы}

Procedure OutPutObr\_Matrix(x,y : Integer;Obr\_Matr : Two\_Matrix );

 Var k,i,j : Integer;

 Elm : String;

 Begin

 k:=12;

 For i:=1 to \_N Do

 Begin

 For j:=1 to \_N Do

 Begin

 MoveTo(x+j\*k,i\*10+y);

 Str(Obr\_Matr[i,j],Elm);

 OutText(Elm);

 End;

 End;

 SetColor(4);

 For i:=1 to \_N Do

 Begin

 For j:=\_N+1 to \_N+\_M-1 Do

 Begin

 MoveTo(x+j\*k,i\*10+y);

 Str(Obr\_Matr[i,j],Elm);

 OutText(Elm);

 End;

 End;

 End;

{----------------------------------}

{Вывод полученной кодовой комбинации}

Procedure OutPut(x,y : Integer;F,A : Mass1);

 Var k,s : Integer;

 Elm : String;

 Begin

 MoveTo(x+10,y-20);

 Str(\_N+\_M-1,Elm);

 OutText(Elm);

 MoveTo(x+(\_N+\_M)\*12-3,y-20);

 Str(0,Elm);

 OutText(Elm);

 k:=12;

 For i:=1 to \_N Do Cicle\_Kod[i]:=F[i];

 s:=1;

 For i:=\_N+1 To \_N+\_M-1 Do

 Begin

 Cicle\_Kod[i]:=A[s];

 s:=s+1;

 End;

 For i:=1 to \_N Do

 Begin

 MoveTo(x+i\*k,y);

 Str(Cicle\_Kod[i],Elm);

 OutText(Elm);

 End;

 SetColor(4);

 For i:=\_N+1 to \_N+\_M-1 Do

 Begin

 MoveTo(10+x+i\*k,y);

 Str(Cicle\_Kod[i],Elm);

 OutText(Elm);

 End;

 End;

{---------------------------------}

{выход из программы}

Function \_Exit(Fon,Color : Integer;Col\_Simv : Byte) : integer;

 Var x,y,k : Integer;

 Stop : Boolean;

 Ch : Char;

Begin

 Window\_(250,200,450,300,1,'Выход');

 x:=225;

 y:=260;

 k:=80;

 SetTextStyle(0,0,1);

 SetColor(Col\_Simv);

 For i:=1 to 2 do

 Begin

 MoveTo(x+i\*k,y);

 OutText(Y\_No[i]);

 End;

 i:=1;

 SetFillStyle(1,Fon);

 Bar(x+i\*k-30,y-15,x+i\*k+30,y+15);

 MoveTo(x+i\*k,y);

 Setcolor(Color);

 OutText(Y\_No[i]);

 Stop:=False;

 Repeat

 ch:=ReadKey;

 Case ch Of

 #75:Begin

 SetFillStyle(1,7);

 Bar(x+i\*k-30,y-15,x+i\*k+30,y+15);

 Setcolor(Col\_Simv);

 MoveTo(x+i\*k,y);

 OutText(Y\_No[i]);

 i:=i-1;

 if(i<1)then i:=1;

 SetFillStyle(1,Fon);

 Bar(x+i\*k-30,y-15,x+i\*k+30,y+15);

 SetColor(Color);

 MoveTo(x+i\*k,y);

 OutText(Y\_No[i]);

 Stop:=False;

 End;{влево}

 #77:Begin

 SetFillStyle(1,7);

 Bar(x+i\*k-30,y-15,x+i\*k+30,y+15);

 Setcolor(Col\_Simv);

 MoveTo(x+i\*k,y);

 OutText(Y\_No[i]);

 i:=i+1;

 if(i>2)then i:=2;

 SetFillStyle(1,Fon);

 Bar(x+i\*k-30,y-15,x+i\*k+30,y+15);

 SetColor(Color);

 MoveTo(x+i\*k,y);

 OutText(Y\_No[i]);

 Stop:=False;

 End;{вправо}

 #13:Begin Stop:=True; \_Exit:=i End;

 End;{Case}

 Until Stop;

 SetTextStyle(0,0,0);

End;

{---------------------------------}

{ОСНОВНАЯ ПРОГРАММА }

{---------------------------------}

Begin

 InitGrf;

Repeat

 flag:=false;

 Fon(3,2,GetMaxX-3,30);

 setcolor(0);

 moveto(20,465);

 SetTextJustify(0,1);

 outtext('Esc - Выход');

 Menu(3,'Работа','О программе','Помощь','','','');

 Case t Of

 1:Begin

 Repeat

 SetFillStyle(1,7); {Строка состтояния}

 Bar(3,450,getmaxx-3,getmaxy-3);

 SetColor(15);

 Line(3,450,getmaxx-3,450);

 Line(3,450,3,getmaxy-3);

 SetColor(0);

 Line(3,getmaxy-3,getmaxx-3,getmaxy-3);

 Line(getmaxx-3,450,getmaxx-3,getmaxy-3);

 {moveto(20,465);

 SetTextJustify(0,1);

 SetColor(0);

 outtext('Нажмите любую клавишу ...');}

 VerMenu(3,'Ввод комбинации','Образующая матрица','Выход','','','');

 Case Np Of

 1 : Begin

 Visual(Mas);

 Dopoln(F);

 Delenye(F,P);

 Window\_(30,300,610,410,1,'Закодированное сообщение. Красные

 символы - контрольные.');

 OutPut(40,370,F,R);

 ReadKey;

 SetFillStyle(1,3);

 Bar(30,150,610,410);

 End;

 2 : Begin

 Window\_(30,50,610,445,1,'Образующая матрица');

 Ed\_Matrix(Obraz\_Matrix);

 Obr\_Matrix(Obraz\_Matrix);

 OutPutObr\_Matrix(40,90,Obraz\_Matrix);

 ReadKey;

 SetFillStyle(1,3);

 Bar(30,50,610,450);

 End;

 3 : Begin

 Case \_Exit(1,15,0) of

 1 : begin

 Np:=3;

 flag:=true;

 end;

 2 : Flag:=False;

 end

 End;

 End;

 Until (Np=3) or (Np=4);

 Ramka\_Off(x1,y1,x2,y2);

 End;

 2:Begin

 SetFillStyle(1,7); {Строка состтояния}

 Bar(3,450,getmaxx-3,getmaxy-3);

 SetColor(15);

 Line(3,450,getmaxx-3,450);

 Line(3,450,3,getmaxy-3);

 SetColor(0);

 Line(3,getmaxy-3,getmaxx-3,getmaxy-3);

 Line(getmaxx-3,450,getmaxx-3,getmaxy-3);

 Window\_(100,80,510,400,1,'О программе');

 moveto(130,130);

 SetTextJustify(0,1);

 outtext('Данная программа является курсовой работой');

 moveto(300,145);

 SetTextJustify(1,1);

 outtext('по курсу :');

 moveto(300,160);

 SetTextJustify(1,1);

 outtext('" Передача информации " ');

 moveto(120,200);

 SetTextJustify(0,1);

 outtext('Задание : Построить кодирующее устройство');

 moveto(185,220);

 outtext('для кода Файра.Кодируемых сообщений 63,');

 moveto(185,240);

 outtext('число обнаруживаемых ошибок Br=4, число');

 moveto(185,260);

 outtext('исправляемых ошибок Bs=3.');

 moveto(120,290);

 outtext('Студент : Иванов А. Е.');

 moveto(120,310);

 outtext('Группа : BM-1-97');

 moveto(120,330);

 outtext('Преподаватель : Каевченко .');

 moveto(300,380);

 SetTextJustify(1,1);

 outtext('СМОЛЕНСК 1999 г.');

 moveto(20,465);

 SetTextJustify(0,1);

 SetColor(0);

 outtext('Нажмите любую клавишу ...');

 Readkey;

 Ramka\_Off(x1,y1,x2,y2);

 End;

 3:Begin

 SetFillStyle(1,7); {Строка состтояния}

 Bar(3,450,getmaxx-3,getmaxy-3);

 SetColor(15);

 Line(3,450,getmaxx-3,450);

 Line(3,450,3,getmaxy-3);

 SetColor(0);

 Line(3,getmaxy-3,getmaxx-3,getmaxy-3);

 Line(getmaxx-3,450,getmaxx-3,getmaxy-3);

 Window\_(100,50,500,350,1,'Помощ');

 moveto(120,110);

 SetTextJustify(0,1);

 outtext('<-- --> ПЕРЕДВИЖЕНИЕ ПО ГОРИЗОНТАЛЬНОМУ МЕНЮ.');

 moveto(120,130);

 outtext('‑ |');

 moveto(120,135);

 outtext('| | ПЕРЕДВИЖЕНИЕ ПО ВЕРТИКАЛЬНОМУ МЕНЮ.');

 moveto(120,140);

 outtext('| ­');

 moveto(114,150);

 moveto(114,155);

 outtext('<Enter> АКТИВАЦИЯ ПУНКТА МЕНЮ.');

 moveto(140,200);

 outtext('В программе использованы соотношения :');

 moveto(120,220);

 outtext('L >= Bs c >= Bs+Br-1');

 moveto(160,235);

 outtext('L');

 moveto(120,240);

 outtext('e = 2 -1 m = c + L');

 moveto(120,260);

 outtext('n = НОК(e,c)');

 moveto(120,280);

 outtext('После расчетов получили образующий многочлен');

 moveto(120,300);

 outtext('Файра : P(x)=1011001011');

 moveto(20,465);

 SetTextJustify(0,1);

 SetColor(0);

 outtext('Нажмите любую клавишу ...');

 ReadKey;

 Ramka\_Off(x1,y1,x2,y2);

 End;

 End; {Case}

Until flag;

closegraph;

End.

**Оглавление.**

|  |  |
| --- | --- |
| **Аннотация** |  |
| **Введение** |  |
| **1. Теоретическое введение** |  |
|  1.1. Постановка задачи.  |  |
|  1.2. Понятие двоичных циклических кодов. |  |
|  1.2.1. Общие понятия и определения. |  |
|  1.2.2. Методы построения циклических кодов**.** |  |
| 1.3.Технические средства кодирования для двоичных циклических кодов. |  |
| 1.4. Коды Файра. |  |
|  **2. Разработка схемы кодирующего устройства**. |  |
| 2.1. Построение кода Файра. |  |
| 2.2. Структурная схема кодирующего устройства. |  |
|  **3. Анализ технического задания.**  |  |
|  **4. Работа устройства** |  |
|  **5.Выбор узлов принципиальной схемы.** |  |
| 5.1. Расчет генератора тактовых импульсов |  |
|  5.2. Счетчик К555 ИЕ10 |  |
|  5.3. Тригер К555 ТВ6 |  |
| 5.4. Регистр К155ИР13. |  |
|  **6. Описание разработки системы.** |  |
| 6.1. Модульная структура системы. |  |
| 6.2. Организация пользовательского интерфейса . |  |
|  6.2.1. Меню системы. |  |
|  6.2.2. Обработка ошибочных ситуаций. |  |
| 6.3. Спецификация на программные модули. |  |
|  **7 . Тестирование.** |  |
| 7.1. Выбор методики тестирования. |  |
| 7.2. Результаты тестирования |  |
|  **8. Ограничения на разработку.** |  |
| 8.1. Инструкция пользователя |  |
|  **Заключение.** |
|  **Список литературы.** |
|  **Приложения** |
| **Приложение 1. Перечень элементов.** |
| **Приложение 2. Блок схема алгоритма.** |
| **Приложение 3. Текст программных модулей.**  |
|  **Приложение 4. Экранные формы.** |