

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Политехнический институт
Факультет информационных
технологий и электроники

Специальность 10.05.03 –

Специализация -

Кафедра
«Информационная безопасность
систем и технологий»

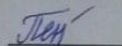
Информационная безопасность
автоматизированных систем
Защищенные автоматизированные
системы управления

ДИПЛОМНЫЙ ПРОЕКТ

на тему

ПРОГРАММНОЕ СРЕДСТВО, ИМИТИРУЮЩЕЕ ВОЗНИКНОВЕНИЕ,
ОБНАРУЖЕНИЕ И УСТРАНЕНИЕ ОТКАЗОВ В АППАРАТУРЕ
ПЕРЕДАЧИ ДАННЫХ

Студент



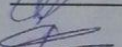
Пеньшин Роман Петрович

Руководитель



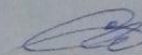
Егорова Наталья Алексеевна

Нормоконтролёр



Фатеев Алексей Григорьевич

Рецензент заместитель генерального
директора АО «ПНИЭИ»



Сомкин С.А.

Работа допущена к защите (протокол заседания кафедры от

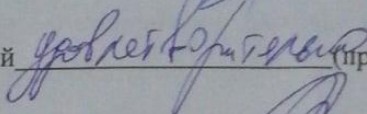
25.01.21 № 16)

Заведующий кафедрой



С.Л. Зефирова

Работа защищена с отметкой



(протокол заседания ГЭК

от 29.06.21 № 14)

Секретарь ГЭК



А.П. Иванов

Пенза 2021

Оглавление

Введение	4
Термины и определения.....	7
1 Обзор отказов в аппаратуре, приводящих к нарушению свойств информационной безопасности данных.....	9
2. Анализ принципов построения и алгоритмов работы встроенных систем контроля	15
2.1 Общий принцип автоматического контроля сложных систем	15
2.2 Метод повторения вычислений.....	15
2.3 Контроль по четности единиц.....	16
2.4 Проверочные коды.	16
2.5 Тестовый контроль.	16
3. Описание и выбор реализации моделируемой аппаратуры передачи данных	18
3.1 Кодирующие и декодирующие устройства циклических кодов	18
3.2 Схемы кодирующих устройств	21
3.3 Схемы декодирующих устройств, обнаруживающих ошибки.....	23
4 Разработка алгоритма, имитирующего работу помехоустойчивого кодера, помехоустойчивого декодера, модулятора, демодулятора	30
5 Разработка программного средства	36
6 Разработка инструкции пользователя.....	59
7 Вредные психофизиологические факторы, влияющие на опорно-двигательный аппарат специалиста по защите информации. Методы и средства физической культуры, снижающие их воздействие.....	60
Заключение	64
Список используемых источников	66

Приложение А Разработка руководства пользователя	69
Приложение Б Код программы	73

Введение

Данная работа необходима для использования в учебном процессе кафедры «Информационная безопасность систем и технологий» по дисциплине «Эксплуатация защищенных телекоммуникационных систем».

В связи с тем, что в настоящее время актуальность проведения дистанционных занятий резко возросла, то необходимо разработать программное средство, которое имитирует возникновение, обнаружение и устранение неисправностей в аппаратуре передачи данных. Наличие подобных программных средств будет позволять студентам выполнять лабораторные работы дистанционно и будет наглядно показывать, как работает встроенная система контроля.

В ходе выполнения лабораторных работ, студенты должны более подробно изучить различные проблемы, связанные с эксплуатацией оборудования, способы, сводящие к минимуму риск серьезных ошибок в его работе, появление которых приводит к нарушению целостности и доступности принимаемой и передаваемой информации. Примеры таких ошибок будет включать в себя:

- неправильное проектирование (неверные исходные данные, ошибки в расчётах, неправильный выбор сырья, материалов, технологии изготовления, неточности в проектировании и так далее);

- неправильное изготовление (некорректная замена сырья, материалов, комплектующих, несоблюдение размеров деталей, узлов и комплектующих, нарушение технологических режимов изготовления и обработки изделий, использование бракованных комплектующих и деталей и тому подобное), то есть элементы изделия или конструкции не соответствуют разработанному, верному проекту;

- неправильная эксплуатация (эксплуатация на нерасчётных режимах, перегрузки, использование при производстве продукции не качественного сырья и тому подобное);

— организационные ошибки, вызванные тем, что руководитель при проектировании, производстве, эксплуатации не предусмотрел организационные меры, предотвращающие перечисленные технические ошибки;

— недостаток квалификации, в результате чего руководитель или ответственное лицо не были достаточно качественно подготовлены, обучены по направлению существования продукции на всех этапах его жизненного цикла для того, чтобы избежать технических и организационных ошибок.

Указанные ошибки минимизируются путём проведения технического контроля процессов проектирования, производства, эксплуатации, ремонта, утилизации изделий и объектов, то есть путём технического контроля всех этапов жизненного цикла продукции.

Цель работы: разработка программного средства, имитирующего возникновение и обнаружение неисправностей в аппаратуре связи для проведения лабораторной работы по дисциплине «Эксплуатация защищенных телекоммуникационных систем».

Лабораторная работа относится к теме «Обнаружение неисправностей в аппаратуре по встроенной системе контроля». В рамках этой лабораторной студенты должны ознакомиться с работой встроенной системы контроля и с процедурами предпускового и динамического контроля. А также в результате выполнения лабораторной работы студенты должны:

- узнать к чему приводят необнаруженные отказы;
- понять, какие отказы система может обнаружить, а какие отказы система обнаружить не сможет;
- высказать свои предложения по поводу ее улучшения.

Для достижения поставленной цели в дипломе необходимо решить следующие задачи:

- проанализировать литературу, чтобы выбрать вариант построения аппаратуры передачи данных;
- выбрать неисправности, которые будут моделироваться;

- составить алгоритм;
- реализовать алгоритм на языке программирования «C#»;
- проверить работоспособность программы;
- проанализировать тему вредные психофизиологические факторы, влияющие на опорно-двигательный аппарат специалиста по защите информации. Методы и средства физической культуры, снижающие их воздействие.

Термины и определения

Циклический код. Название этого кода происходит от их основного свойства: если кодовая комбинация принадлежит циклическому коду, то комбинации, полученные циклической перестановкой элементов, также принадлежат этому коду. Общим свойством всех разрешенных кодовых комбинаций циклического кода (как полиномов) является их делимость без остатка на некоторый выбранный полином, называемый производящим [11].

Синдромом ошибки в циклических кодах является наличие остатка от деления принятой кодовой комбинации на этот полином [11].

Исправляющей способностью называется количество ошибок, которое может исправить данный код. Обнаруживающая и исправляющая способности кодов определяются величиной кодового расстояния [12].

Блочный код – в информатике тип канального кодирования. Он увеличивает избыточность сообщения так, чтобы в приемнике можно было расшифровать его с минимальной (теоретически нулевой) погрешностью, при условии, что скорость передачи информации (количество передаваемой информации в битах в секунду) не превысила бы канальную производительность. Главная характеристика блочного кода состоит в том, что это – канальный код фиксированной длины. Обычно, система блочного кодирования получает на входе « k »-значное кодовое слово « W », и преобразовывает его в « n »-значное кодовое слово « $C(W)$ ». Это кодовое слово называется блоком [13].

Класс эквивалентности – варианты полиномов ошибок, обработка которых приводит к одному и тому же результату [14].

Разделение на классы эквивалентности – это техника, при которой функционал (диапазон возможных входных значений) разделяется на группы значений эквивалентных по воздействию на систему [14].

Кодовое расстояние – это минимальное число элементов, в которых любая кодовая комбинация отличается от другой (по всем парам кодовых слов) [18].

Класс эквивалентности – варианты полиномов ошибок, обработка которых приводит к одному и тому же результату [14].

Разделение на классы эквивалентности – это техника, при которой функционал (диапазон возможных входных значений) разделяется на группы значений эквивалентных по воздействию на систему [14].

Предпусковой контроль — это контроль, который производится перед работой[14].

Динамический контроль — это проверка в режиме работы значений выходных напряжений элементов, которые входят в состав устройства [15].

Сигнатурный анализ сводится к сопоставлению реальной сигнатуры конкретного узла, отображаемой дисплеем анализатора, с образцовой сигнатурой этого узла, указанной на схеме или в таблице руководства по обслуживанию испытуемого устройства. Несовпадение сигнатур свидетельствует о неисправности и неправильном функционировании устройства [17].

Надежность – это свойство технического объекта сохранять свои характеристики (параметры) в определенных пределах при данных условиях эксплуатации [16].

Отказ – это событие, заключающееся в нарушении работоспособного состояния объекта. Отказ может быть полным или частичным. Полный отказ характеризуется переходом объекта в неработоспособное состояние. Частичный отказ характеризуется переходом объекта в частично неработоспособное состояние [16].

Критерий отказа – это признак или совокупность признаков нарушения работоспособного состояния объекта, установленные в документации [16].

1 Обзор отказов в аппаратуре, приводящих к нарушению свойств информационной безопасности данных

Одной из основных задач функционального контроля в цифровых системах является оперативное обнаружение отказов технических средств. Для решения этой задачи необходимо контролировать состояние каждого ТС и сам процесс передачи и обработки информации. Контроль процесса в целом является системным, в большинстве случаев он оказывается более простым в реализации и достаточно полным, его элементы включаются во все протоколы обмена. В существующих протоколах передачи информации предусмотрен контроль верности информации, благодаря которому обнаруживается возникновение любого технического отказа, вызывающего нарушения процесса передачи и обработки информации [21].

Одним из недостатков контроля процесса в целом является задержка с обнаружением отказа на интервале времени от момента возникновения отказа до его обнаружения. С этой точки зрения определенные преимущества имеет функциональный контроль состояния каждого ТС системы, благодаря которому отказавшее ТС может быть заблокировано в момент возникновения отказа. При этом отказ должен обнаруживаться и устраняться в точке технологического процесса, наименее удаленной во времени и пространстве от точки возникновения этого отказа. В более общем случае реальная система функционального контроля фиксирует отказы лишь с некоторой вероятностью. Отказы, не фиксируемые контролем, обнаруживаются с задержкой времени, которая является в общем случае величиной случайной. В силу аддитивности эта задержка прибавляется к времени восстановления, где случайное время восстановления, исчисляемое от момента обнаружения отказа до момента полного восстановления; - указанное выше случайное время задержки обнаружения отказа, исчисляемое от момента, когда фактически произошел отказ, до момента его обнаружения [21].

Поэтому за один из показателей качества функционального контроля ТС принимается вероятность оперативного (т.е. в момент возникновения или с

заданной допустимой задержкой) обнаружения отказа [21].

Для обеспечения единой стратегии контроля и диагностики цифровых систем целесообразно использование двух уровней: верхний уровень - контроль и диагностика с точностью до типового элемента замены на базе встроенных средств контроля, нижний уровень - диагностики неисправностей с помощью средств технической диагностики до неисправного элемента в типовом элементе защиты [21].

В этой связи одним из эффективных решений проблемы контроля цифровой системы представляется использование принципа встроенного контроля, который заключается в том, что цифровая система и ее составные части разрабатываются таким образом, чтобы обеспечивалась возможность встроенного контроля без участия какого-либо внешнего оборудования. Методы встроенного контроля можно иерархически перераспределить между различными уровнями от составных частей до цифровой системы в целом. Встроенный контроль позволяет производить проверку цифровой системы в процессе выполнения основных функций и по существу повышает эксплуатационную надежность системы, поскольку позволяет обнаруживать отказы сразу же при их возникновении [21].

Встроенным средствам контроля присущи следующие основные преимущества:

- значительное сокращение времени восстановления работоспособности системы и, соответственно, повышение общей эксплуатационной готовности;
- уменьшение численности обслуживающего персонала, обеспечивающего ремонтно-восстановительные работы;
- сокращение видов ремонта и ЗИП за счет повышения достоверности контроля [21].

Однако необходимо учитывать, что средства встроенного оперативного контроля двояко влияют на характеристики контролируемой системы: с одной стороны повышается достоверность контроля и уменьшается время обнаружения неисправности, с другой стороны, возрастает объем дополнительного оборудования, что в свою очередь приводит к снижению надежности самой

системы. Таким образом, встроенные средства оперативного контроля, обеспечивая выигрыш в достоверности контроля, приводят к определенному проигрышу в безотказности, контролируемой аппаратуры. В этой связи поиск разумного оптимума между полнотой охвата встроенным контролем системы и объемом средств встроенного контроля является актуальной задачей. Учет влияния объема встроенного контроля на эксплуатационные характеристики системы позволит оптимально перераспределять ресурсы между встроенными и внешними средствами контроля и диагностики. Поэтому для обоснованного выбора встроенного контроля необходимо проведение исследования влияния объема средств встроенного контроля на такие характеристики, как коэффициент готовности, вероятность обнаружения неисправности и среднее время восстановления цифровой системы [21].

Существуют следующие параметры эффективности системы встроенного контроля:

- коэффициент готовности контролируемой системы с системой встроенного контроля;
- вероятность обнаружения неисправности контрольным устройством;
- прорыв в безотказности контролируемого устройства с системой контроля;
- выигрыш в достоверности при использовании встроенного контроля;
- среднее время наработки на отказ контролируемой системы с системой встроенного контроля;
- среднее время восстановления контролируемой системы с системой встроенного контроля [21].

Уязвимости ИС.

Угрозы, как правило, появляются не сами по себе, а через уязвимости, приводящие к нарушению безопасности в ИС. Уязвимости присущие ИС, неотделимы от нее, и обуславливаются недостатками процесса функционирования, свойствами архитектуры ИС, протоколами обмена и интерфейсами, применяемым

программным обеспечением и аппаратной платформой, условиями эксплуатации и расположения[22].

Источники угроз могут использовать уязвимости для нарушения безопасности информации, получения незаконной выгоды (нанесения ущерба собственнику, владельцу, пользователю информации) [22].

Устранение или существенно ослабление уязвимостей, влияет на возможности реализации угроз безопасности информации[22].

Существуют следующая классификация уязвимостей:

- сопутствующие техническим средствам излучения:
 - электромагнитные (побочные излучения элементов технических средств, кабельных линий технических средств, излучения на частотах работы генераторов, на частотах самовозбуждения усилителей);
 - электрические (наводки электромагнитных излучений на линии и проводки, просачивание сигналов в сети электропитания, в цепи заземления, неравномерность потребления тока электропитания);
 - звуковые (акустические, виброакустические);
- активизируемые:
 - аппаратные закладки (устанавливаемые в телефонные линии, в сети электропитания, в помещениях, в технических средствах);
 - программные закладки (вредоносные программы, технологические выходы из программ, нелегальные копии ПО) [22];
- определяемые особенностями элементов:
 - элементы, обладающие электроакустическими преобразованиями (телефонные аппараты, громкоговорители, микрофоны);
 - элементы подверженные воздействию электромагнитного поля (магнитные носители, микросхемы);
- определяемые особенностями защищаемого объекта:

- местоположением объекта (отсутствие контролируемой зоны, наличие прямой видимости объектов, удаленных и мобильных элементов объекта);

- организацией каналов обмена информацией (использование радиоканалов, глобальных информационных сетей, арендуемых каналов) [22].

Субъективные – зависят от действий сотрудников, в основном устраняются организационными и программно-аппаратными методами:

- ошибки:

- при подготовке и использовании программного обеспечения (при разработке алгоритмов и программного обеспечения, инсталляции и загрузке программного обеспечения, эксплуатации программного обеспечения, вводе данных);

- при управлении сложными системами (при использовании возможностей самообучения систем, организация управления потоками обмена информации);

- при эксплуатации технических средств (при включении/выключении технических средств, использовании технических средств охраны, использование средств обмена информацией) [22];

- нарушения:

- режима охраны и защиты (доступа на объект, доступа к техническим средствам);

- режима эксплуатации технических средств (энергообеспечения, жизнеобеспечения);

- режима использования информации (обработка и обмен информацией, хранение и уничтожение носителей информации, уничтожения производственных отходов и брака);

- режима конфиденциальности (сотрудники в не рабочее время, уволенные сотрудники; обиженные сотрудники) [22].

Случайные – зависят от особенностей окружающей ИС среды и непредвиденных обстоятельств:

- сбои и отказы:
 - отказы и неисправности технических средств (обрабатывающих информацию, обеспечивающих работоспособность средств обработки информации, обеспечивающих охрану и контроль доступа);
 - старение и размагничивание носителей информации (дискет и съемных носителей, жестких дисков, микросхем, кабелей и соединительных линий);
 - сбои программного обеспечения (операционных систем и СУБД, прикладных программ, сервисных программ, антивирусных программ);
 - сбои электроснабжения (оборудования, обрабатывающего информацию; обеспечивающего и вспомогательного оборудования) [22];
- повреждения:
 - жизнеобеспечивающих коммуникаций (электро-, водо-, газо-, теплоснабжения, канализации; кондиционирования и вентиляции);
 - ограждающих конструкций (внешних ограждений территорий, стен и перекрытий зданий; корпусов технологического оборудования) [22].

2. Анализ принципов построения и алгоритмов работы встроенных систем контроля

2.1 Общий принцип автоматического контроля сложных систем

В системах автоматического контроля при обнаружении неисправности обычно используется метод контроля прохождения сигнала. На вход проверяемого устройства подаются сигналы, обычно совпадающие с реальными входными сигналами. В контрольных точках производится измерение параметров реакции устройства на входные сигналы. Если информация, снимаемая в контрольных точках, непригодна для непосредственного использования при оценке работы устройства, то она проходит через преобразователь. Обычно информация преобразуется в цифровую форму, так как схемы сравнения и логические схемы, как правило, являются цифровыми. Далее, преобразованная информация поступает в компаратор, т.е. сравнивающее устройство, где происходит сравнение полученной информации с эталонными значениями измеряемых величин. Компаратор может вычислять отклонение измеряемой величины от эталонной и выдать величину отклонения или может выдать сигнал с оценкой «годен – не годен». Существует большое количество методов контроля работы ЦВМ и поиска неисправностей [20].

2.2 Метод повторения вычислений

При применении данного метода делают двойной просчет и сравнивают результаты. При решении больших задач проверяются результаты отдельных частей. Если при сравнении результаты двойного просчета не совпадают, то производится повторное решение. Можно одну и ту же задачу решать на двух машинах с сравнением промежуточных результатов. При несовпадении результатов может предусматриваться двойной просчет. При двукратном несовпадении 16 результатов может предусматриваться остановка с последующим включением поиска неисправностей. Если результаты двух решений совпадают, то

осуществляется переход к решению следующей части. Этот метод контроля не эффективен, так как для его осуществления требуется или дорогостоящее оборудование, или двойное полезное машинное время. Но он достаточно надежен, так как достигается избыточность информации за счет повторения вычислений. [20].

2.3 Контроль по четности единиц

В двоичном коде всегда содержится четное или нечетное количество единиц. При четном количестве единиц двоичный код можно обозначить символом «1», а при нечетном – символом «0» или наоборот. Для контроля вычислений эти символы записываются в контрольные разряды. При выполнении арифметических операций нужно учитывать наличие двоичных переносов, иначе контрольная сумма будет неверной. Данный метод контроля обнаруживает единичную ошибку и совершенно не пригоден при появлении большого числа ошибок [20].

2.4 Проверочные коды

Сущность данного метода состоит в том, что для обнаружения ошибки необходим код, который для перехода от одного числа, изображенного этим кодом, к представлению этим же кодом другого числа требовал бы минимум два изменения в разрядах числа. Контроль по данному методу позволяет не только обнаружить ошибки, но и исправить их. Однако при осуществлении данного метода контроля резко увеличивается разрядность числа, существенно усложняется логика машины [20].

2.5 Тестовый контроль

Тестовый контроль является наиболее распространенным методом контроля ЦВМ. Под тестом будет пониматься процесс задания некоторой совокупности входных значений и наблюдения за выходами проверяемой машины. В зависимости от назначения тесты подразделяются на наладочные, проверочные, диагностические и контрольные тест – задачи. Наладочные тесты применяются в

процессе наладки при установке машины с целью обнаружения грубых ошибок в монтаже и логике машины. Прогоном отработанных проверочных тестов завершается наладка машины. В них предусматривается более полный контроль узлов машины с разнообразными режимами работы. Они более полно учитывают логическую схему узлов машины. Контрольные тест – задачи применяются для периодического контроля в процессе основной работы машины [20].

Тестовый диагностический контроль является профилактическим, т.е. он может производиться в период подготовки к решению задачи. Диагностический тест составляется с учетом частоты отказов каждого типа элементов и ориентировочного времени на их отыскание [20].

Профилактический контроль заключается в решении задач, ответы которых известны. При этом элементы машины включаются в наиболее трудные режимы работы путем изменения напряжения питания в пределах допустимого. Для каждого устройства вычислительной машины имеется свой проверочный тест. Тесты отдельных устройств сводятся в единый комплексный тест, определяющий последовательность выполнения отдельных тестов и продолжительность их выполнения. Неисправность обнаруживается при работоспособности определенного минимума устройств вычислительной машины. Для точного указания места неисправности при тестовом контроле требуется вмешательство оператора[20].

3. Описание и выбор реализации моделируемой аппаратуры передачи данных

3.1 Кодирование и декодирование устройств циклических кодов

В данном разделе выпускной квалификационной работы рассматриваются типовые процедуры кодирования и декодирования циклических кодов, а также соответствующие им схемы кодирующих и декодирующих устройств.

Задача кодирования заключается в формировании, по информационным словам, комбинаций циклических «n» кода и «k» кода, которые по своей структуре могут быть разделимыми и неразделимыми. Построение кодовых комбинаций может производиться либо по порождающему $g(x)$, либо по проверочному $h(x)$ многочленам. Существует три типовых способа кодирования информационных слов:

— способ кодирования с использованием порождающего многочлена $g(x)$ и формированием неразделимых кодовых комбинаций. Пусть имеется некоторое k элементарное информационное слово $a(x)$. Тогда процедура построения разрешенной комбинации циклического «n» кода и «k» кода $b(x)$ алгебраически представляет собой произведение двух многочленов $a(x)$ и $g(x)$, то есть:

$$b(x) = a(x) \cdot g(x),$$

для практической реализации этого способа необходима схема умножения двух многочленов[8];

— способ кодирования с использованием $g(x)$ и формированием различных кодовых комбинаций. Алгебраическое представление разрешенной комбинации имеет вид:

$$b(x) = x^{n-1} \cdot a(x) \cdot r(x),$$

где $r(x)$ это остаток от деления многочлена x в степени n минус k умноженное на $a(x)$ поделенное на $g(x)$. Для практической реализации способа необходима схема деления одного многочлена на другой [8];

— способ кодирования с использованием $h(x)$ и формированием делимых кодовых комбинаций. Данный способ основан на построении и решении системы рекуррентных соотношений. Для циклических кодов справедливо:

$$h(x) = \frac{x^{n-1}}{g(x)}$$

или:

$$h(x) \cdot g(x) = x^n + 1 = 0.$$

Так как разрешенная комбинация $b(x)$ кратна $g(x)$, то выполняется равенство:

$$b(x) \cdot h(x) = 0.$$

Некоторый произвольный коэффициент c_j при x в степени j рассчитывается по формуле:

$$c_j = h_0 \times b_j + h_1 \times b_{j-1} + \dots + h_k \times b_{j-k} = \sum_{i=0}^k h_i \times b_{j-i} = 0,$$

где h_i и b_i это коэффициент при степени x в степени i многочленов $h(x)$ и $b(x)$.

Так как h_0 равняется h_k равняется одному, то последнее выражение уточняется:

$$c_j = b_j + b_{j-k} + \sum_{i=0}^{k-1} h_i \times b_{j-i} = 0,$$

тогда:

$$b_{j-k} = b_j + \sum_{i=1}^{k-1} h_i \times b_{j-i}.$$

Если j равняется n минус один, то:

$$b_{n-k-1} = b_{n-1} + h_1 \times b_{n-2} + \dots + h_{k-1} \times b_{n-k} = b_{n-1} \times \sum_{i=0}^{k-1} h_i \times b_{n-i-1}.$$

Если j равняется n минус два, то:

$$b_{n-k-2} = b_{n-1} \times \sum_{i=0}^{k-1} h_i \times b_{n-i-2}.$$

Если j равняется n минус k , то:

$$b_0 = b_k \times \sum_{i=0}^{k-1} h_i \times b_{k-i}.$$

В комбинации $b(x)$ делимого n кода и k кода коэффициенты от b_{n-k} , b_{n-k+1} , до b_{n-1} соответствуют информационному слову, а от b_0 , b_1 , до b_{n-k-1} является проверочным элементом. Поэтому если известно информационное слово $a(x)$, то коэффициенты кодовой комбинации $b(x)$ определяются по следующему правилу [8]:

$$b_{n-k+i} = a_i,$$

$$i = \overline{0, k-1},$$

$$b_{n-k-i} = b_{n-i} \times \sum_{j=0}^{k-1} h_j \times b_{n-i-j}, \quad i = \overline{0, k-1}.$$

Для практической реализации способа необходима схема решения системы рекуррентных соотношений. [8]

Из рассмотренных способов кодирования для реализации в дипломном проекте был выбран способ кодирования с использованием порождающего многочлена $g(x)$ и формированием неразделимых кодовых комбинаций [8].

3.2 Схемы кодирующих устройств

Схемное построение кодирующих устройств циклических кодов определяется принятым способом кодирования информационных слов[8].

Существует несколько способов кодирования. Рассмотрим, подходящий для рассматриваемого в дипломной работе способа кодирования[8].

Кодирование реализуется с помощью кодера, основным элементом которого является схема умножения информационного многочлена $a(x)$, поступающего на вход, на порождающий многочлен $g(x)$. Существует несколько вариантов построения схем умножения примерно одинаковой сложности. Будет рассмотрен один из таких вариантов [8].

Пусть:

$$g(x) = \sum_{i=0}^{n+k} g_i \cdot x^i,$$

где g_i – это коэффициенты, которые могут принимать значения ноль или единица.

В общем виде схема умножения многочленов $a(x)$ и $g(x)$, а, следовательно, и кодирующего устройства представляет собой регистр сдвига без обратной связи. Схема умножения многочленов $a(x)$ и $g(x)$ представлена на рисунке 1 [8].

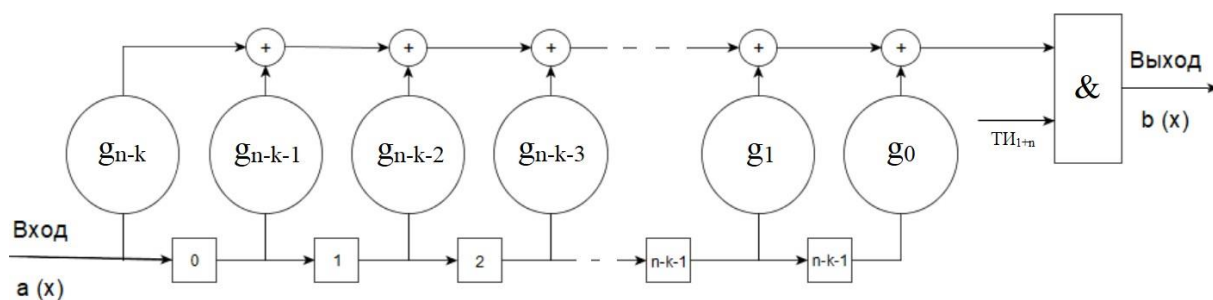


Рисунок 1 - Схема умножения многочленов $a(x)$ и $g(x)$

Полная схема кодера включает в себя регистр сдвига на n минус k разрядов, n минус k сумматоров по модулю два, n минус k плюс 1 условных дополнительных связей, соединяющих входы (выходы) ячеек с сумматорами и конъюнктурами. На вход поступают коэффициенты многочлена $a(x)$, начиная

с коэффициентов старших порядков, а затем следует n минус k нулей. Процесс умножения осуществляется в течении n тактов. Для конкретной реализации кодера при g_i равное нулю соответствующая условная связь и сумматор из схемы исключается [8].

Режим обнаружения ошибок. Процедура декодирования, которая основана на использовании свойства делимости без остатка комбинаций циклического n кода и k кода на порождающий многочлен $g(x)$. Поэтому алгоритм декодирования включает в себя деление принятой комбинации, описываемой многочленом $s(x)$ на $g(x)$, вычисление и анализ остатка от деления $r(x)$. Если $r(x)$ равняется 0, то комбинация $s(x)$ считается разрешенной и ее информационные элементы выдаются получателю, в противном случае комбинация бракуется и формируется сигнал «Ошибка» или «Стирание». При декодировании делимого кода и получателю поступает k элементарное информационное слово, соответствующее старшим степеням полинома $s(x)$, а при использовании неразделимого кода возникает дополнительная задача выделения из принятой комбинации k информационных элементов [8].

Режим исправления ошибок. Декодирование в данном режиме производится по методу максимального правдоподобия. Исправление ошибок циклическим кодом осуществляется в следующей последовательности:

- а) вычисление $r(x)$ комбинации синдрома;
- б) определение по таблице декодирования образца ошибки (образующего смежного класса) и сложение по $\text{mod} 2$ комбинации и образца ошибки [8].

Указанные операции выполняются с использованием специальных свойств циклических кодов, позволяющих строить достаточно простые и эффективные декодирующие устройства. Свойства циклических кодов:

— синдром S некоторой n -элементарной комбинации, описываемой многочленом $s(x)$, тождественен остатку $r(x)$ от деления полинома $s(x)$ на порождающий многочлен $g(x)$ [8].

3.3 Схемы декодирующих устройств, обнаруживающих ошибки

Декодирование комбинаций неразделимого циклического кода осуществляется с помощью кодера, в состав которого входит схема деления многочлена $s(x)$, соответствующего n элементарной комбинации, принятой из дискретного канала связи, на образующий многочлен $g(x)$, двух конъюнкторов и дизъюнктора [8]. Схема деления реализуется в виде регистра сдвига с обратными связями, структура которого определяется видом:

$$g(x) = \sum_{i=0}^{n-k} g_i \times x^i.$$

Схема деления представлена на рисунке 2.

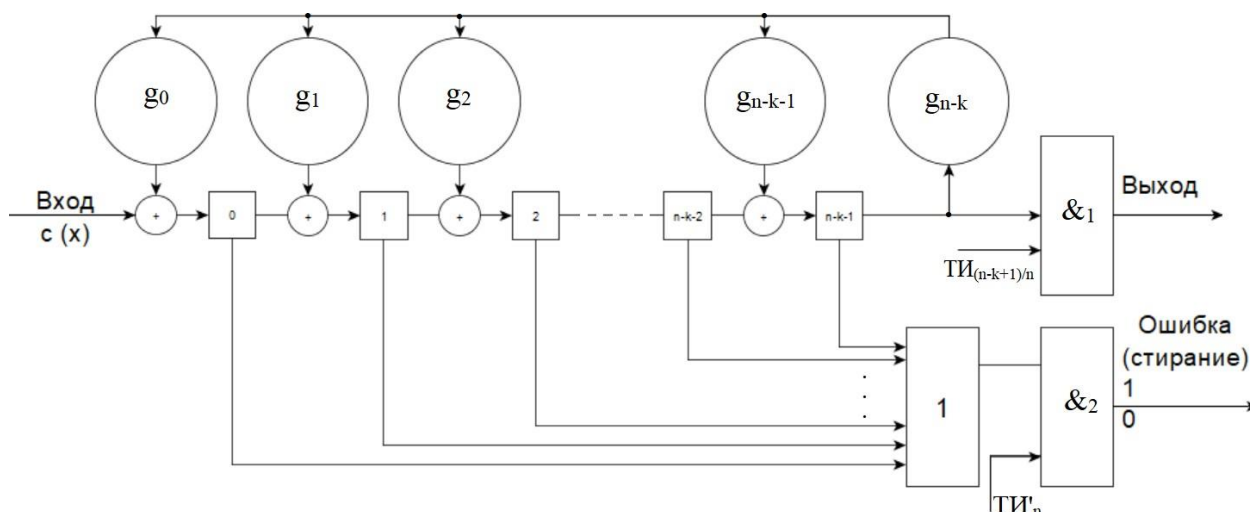


Рисунок 2 – Схема деления

В течение первых n минус k тактов коэффициенты старших степеней $s(x)$ вводятся в регистр сдвига, а затем в течение последующих k тактов с помощью обратных связей реализуется процесс деления с формированием информационного слова (частного) и остатка от деления $r(x)$. Процедура декодирования занимает $n - k$ тактов, после чего сформированный на регистре остаток $r(x)$ параллельно считывается на вход элемента ИЛИ. Если $r(x) \neq 0$, то на выходе U_2 появляется сигнал “Ошибка” или “Стирание” в виде логической единицы [8].

Информационное k – элементное слово разрешается выдавать получателю

при отсутствии обнаруженных ошибок[8].

Исправление ошибок осуществляется с помощью стандартной процедуры, основанной на вычислении синдрома и использовании таблицы декодирования, с учетом специальных свойств циклического кода. Учет алгебраической связи между синдромом и образцом ошибки позволяет избежать необходимости построения таблиц и существенно упростить структуру декодера [8].

Укрупненная структурная схема декодирующего устройства представлена на рисунке 3.

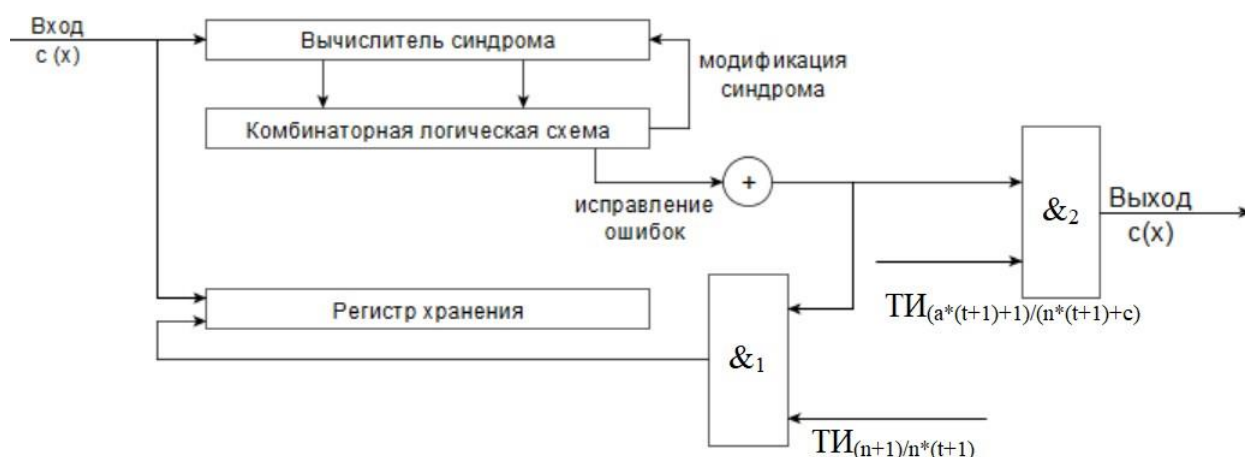


Рисунок 3 - Структурная схема декодирующего устройства

Декодер включает в себя следующие элементы:

- вычислитель синдрома, построенный на основе схемы умножения x в степени n минус k на $s(x)$ поделенное на $g(x)$;
- комбинаторная логическая схема, представляющая собой совокупность дешифраторов, каждый из которых нанесен на синдром класса эквивалентности (число дешифраторов равно числу классов);
- регистр хранения с обратной связью;
- два конъюнктора [8].

Будет рассматриваться декодера при исправлении t – кратной ошибки.

В течение первых n тактов принимается последовательность $s(x)$ записывается в регистр хранения, а вычислитель синдрома формирует соответствующий синдром. Затем производятся одновременные сдвиги по тактам как синдрома, так и принятой комбинации, причем на каждом такте

сравниваются синдромы в вычислителе комбинаторной логической схемы. При совпадении синдромов на выходе комбинаторной логической схемы вырабатывается единичный сигнал, по которому инвертируется (с помощью сумматора по mod2) искаженный элемент в принятой комбинации и модифицируется синдром в вычислителе. Процесс повторяется t раз, и признаком, отмечающим исправление всех ошибок, является формирование в последнем цикле нулевого синдрома. После завершения процедуры исправления информационные элементы выводятся из регистра хранения через u_2 [8].

Наиболее сложной задачей при построении декодирующих устройств является рациональное разбиение образцов исправляемых ошибок на классы эквивалентности и правильный выбор синдрома в каждом классе [8].

Полная схема представлена на рисунке 4.

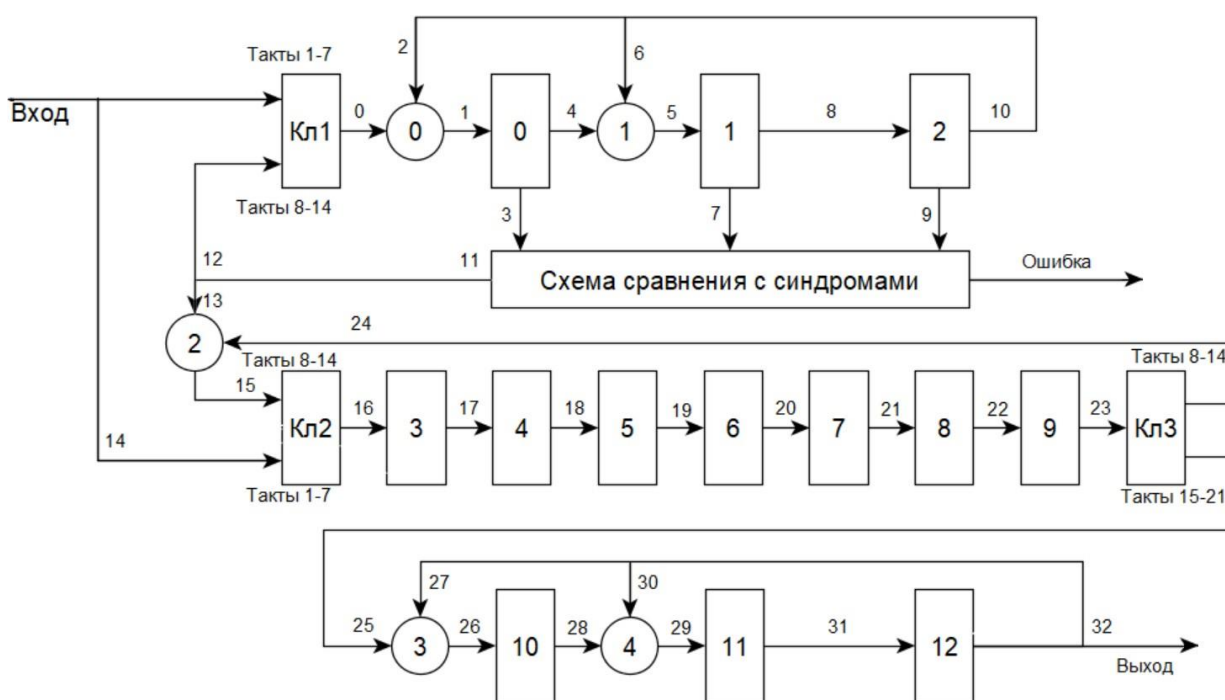


Рисунок 4 – Полная схема декодера

Кружочком обозначается сумматор по модулю два, прямоугольником обозначаются ячейки регистра сдвига, прямоугольник Кл1, Кл2, Кл3 и Кл4 это ключ, а стрелки от нуля до тридцати двух — это цепи.

В течении первых n и k тактов коэффициенты при старших степенях s

$x(i)$ вводятся в регистр сдвига, а затем в течение последующих k тактов с помощью обратных связей реализуется процесс деления с формированием информационного слова (частотного) и остатка от деления $r(x)$. Процедура декодирования занимает n тактов, после чего сформированный на регистре остаток $r(x)$ параллельно считывается на вход элемента «ИЛИ». Если $r(x)$ не равен ноль, то на выходе u_2 появляется сигнал «Ошибка» или «Стирание» в виде логической единицы [8].

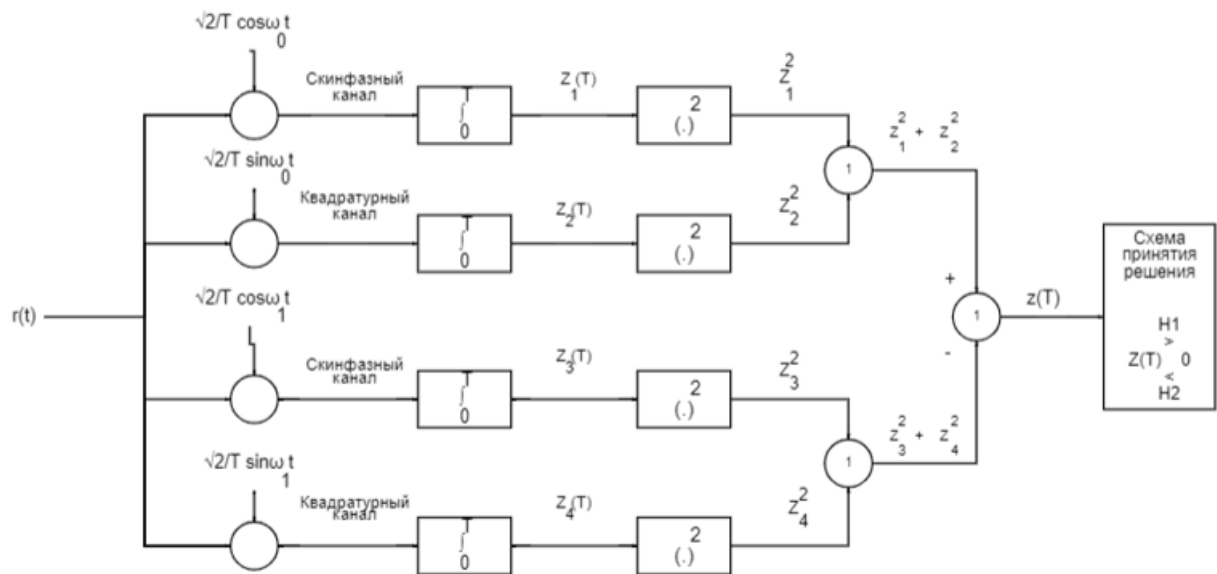


Рисунок 5 – Схема демодулирующего устройства ЧМ сигналов

На рисунке 5 показаны синфазный и квадратурный каналы, используемые для некогерентного детектирования набора сигналов бинарной частотной модуляции. Две верхние ветви настроены на детектирование сигнала с частотой ω_1 .

Для синфазной ветви опорный сигнал имеет вид:

$$\sqrt{\frac{2}{T}} \cos \omega_1 t.$$

для квадратурной:

$$\sqrt{\frac{2}{T}} \sin \omega_1 t.$$

Аналогично две нижние ветви настроены на детектирование сигнала с частотой ω_2 , когда опорные сигналы имеют вид синфазная ветвь:

$$\sqrt{\frac{2}{T}} \cos \omega_2 t;$$

и квадратурная ветвь:

$$\sqrt{\frac{2}{T}} \sin \omega_2 t [19].$$

Предположительно, принятый сигнал $r(t)$ будет имеет вид:

$$\cos \omega_1 t + n(t),$$

то есть фаза точно равна нулю. Следовательно, сигнальный компонент принятого сигнала точно соответствует (по частоте и фазе) опорному колебанию верхней ветви. В такой ситуации максимальный выход должен дать интегратор произведений верхней ветви. Вторая ветвь должна дать нулевой выход (шум, по предположению, имеет нулевое среднее значение), поскольку его опорное колебание ортогонально сигнальному компоненту аддитивной смеси $r(t)$. При ортогональной передаче сигналов третья и четвертая ветви также должны дать близкие к нулю выходы, поскольку их опорные колебания также ортогональны сигнальному компоненту смеси $r(t)$. [19].

Информационное k элементное слово разрешается выдавать получателю при отсутствии обнаруженных ошибок[19].

Полная схема структуры моделируемой аппаратуры передачи данных включает в себя 5 составляющих блоков, а именно:

- кодер;
- декодер;
- модулятор;
- демодулятор;
- встроенная система контроля.

Результат представлен на рисунке 6.

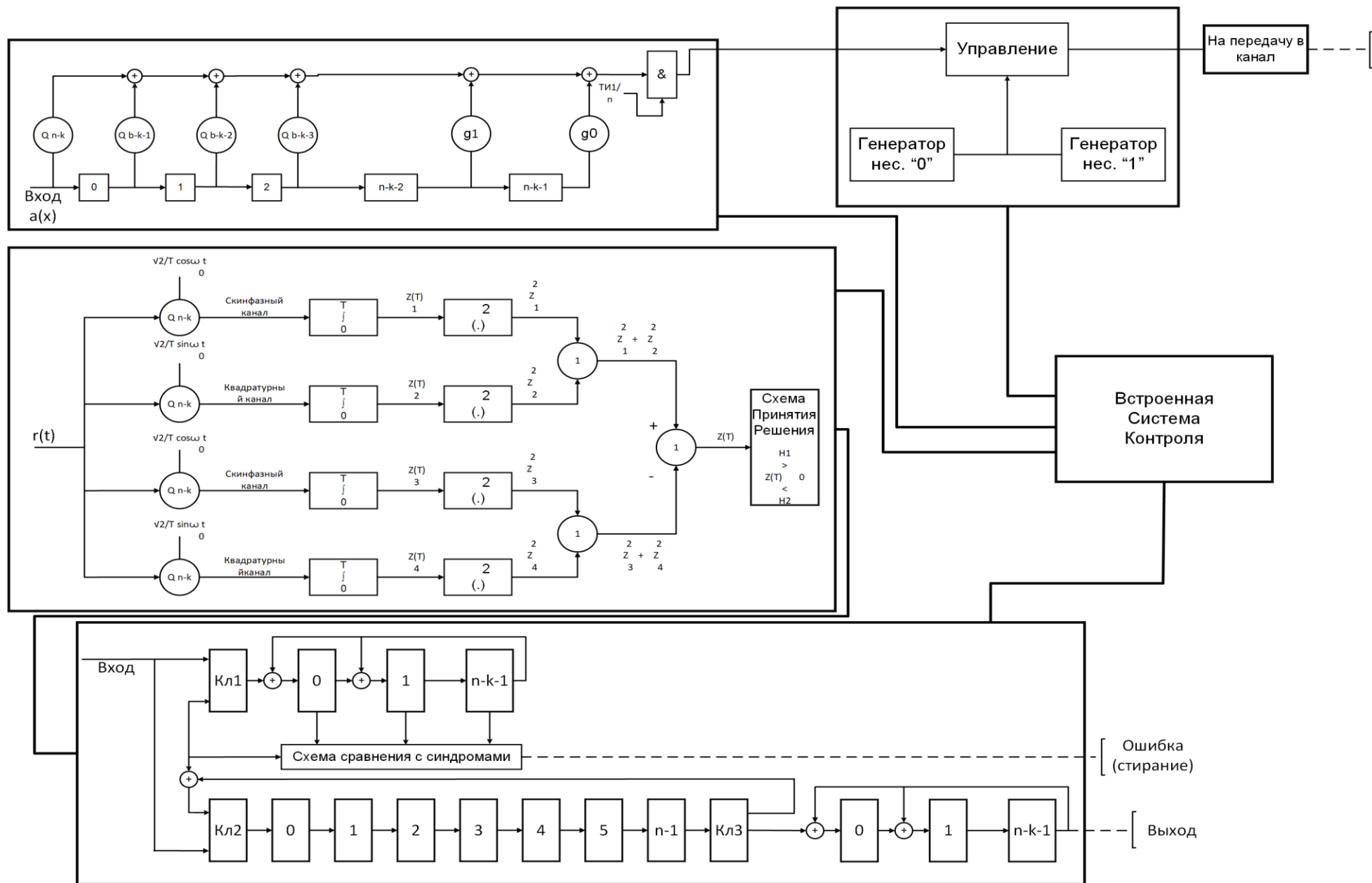


Рисунок 6 – Полная схема структуры моделируемой аппаратуры передачи данных

В данной работе моделируется аппаратура связи, состоящая из 4 блоков:

- помехоустойчивый кодер;
- помехоустойчивый декодер;
- модулятор;
- демодулятор.

В этой аппаратуре реализован циклический помехоустойчивый код, который исправляет одну ошибку. В ВКР моделируется аппаратная реализация кодера и декодера, показанная на рисунке 6. В кодере идет умножение информационного слова на многочлен. Декодер реализует операции деления кодового слова на порождающий многочлен. В результате процедуры, от деления, в линии задержки формируется остаток от деления. Этот остаток подается в схему сравнения с синдромом. Если остаток нулевой, то ошибок в слове нет и полученное в результате деления подается дальше. Если остаток содержит хотя бы одну единицу, то начинается процедура исправления ошибок. Это осуществляется путем поиска в таблице синдромов того синдрома, который соответствует текущему синдрому. Если такой синдром найден, то ошибка исправляется. В аппаратуре происходит частотная модуляция. В данной аппаратуре смоделированы следующие отказы:

- в модуляторе генератор несущей частоты вырабатывает частоту, отличающуюся от номинальной на 100 Гц;
- в кодере ячейка № 1, в регистре сдвига не изменяет своего состояния;
- в демодуляторе схема сравнения отказывает таким образом, что на ее выходе всегда установлена логическая единица;
- в декодере первая ячейка линии задержки не изменяет своего состояния.

4 Разработка алгоритма, имитирующего работу помехоустойчивого кодера, помехоустойчивого декодера, модулятора, демодулятора

В данном разделе выпускной квалификационной работы рассматриваются разработанные алгоритмы программного средства, имитирующего возникновение, обнаружение и устранение отказов в аппаратуре передачи данных. На рисунке 7 представлен алгоритм, который реализует моделирование работы аппаратуры передачи данных и возникновение отказов в ней.

На вход подаются данные. Работа алгоритма начинается с вызова цикла `for`, который, пошагово, заполняет таблицу синдромов. Далее с помощью условий `if/else` происходит проверка на длину помехоустойчивого кода. После срабатывания, одной из трех проверок на длину кода, происходит установка длины регистра в кодере, затем заполняется порождающий многочлен. Определяется длина посылки(сообщения). Результат представлен на рисунке 7. Лист 1.

Далее происходит обнуление линии задержки регистра в кодере. Затем начинается процесс формирования кодового слова с помощью цикла `for` и нескольких условий `if/else`. На каждой шаге цикла выполняется построение кодового слова путем сложения кодового слова с регистром задержки и последующим умножением на порождающий многочлен. Затем выполняется кодирование слова. Далее для выполнения процесса модуляции вызывается цикл `for`. Затем снова обнуляется линия задержки регистра в кодере. Результат представлен на рисунке 7. Лист 2.

На следующем шаге вычисление синдрома и сравнение его с таблицей синдромов. Затем выполняется сдвиг регистра хранения. Завершающим этапом работы алгоритма является вычисление принятого информационного слова с помощью циклов `for` и нескольких условий `if/else`. Результат алгоритма представлен на рисунке 7. Лист 3.

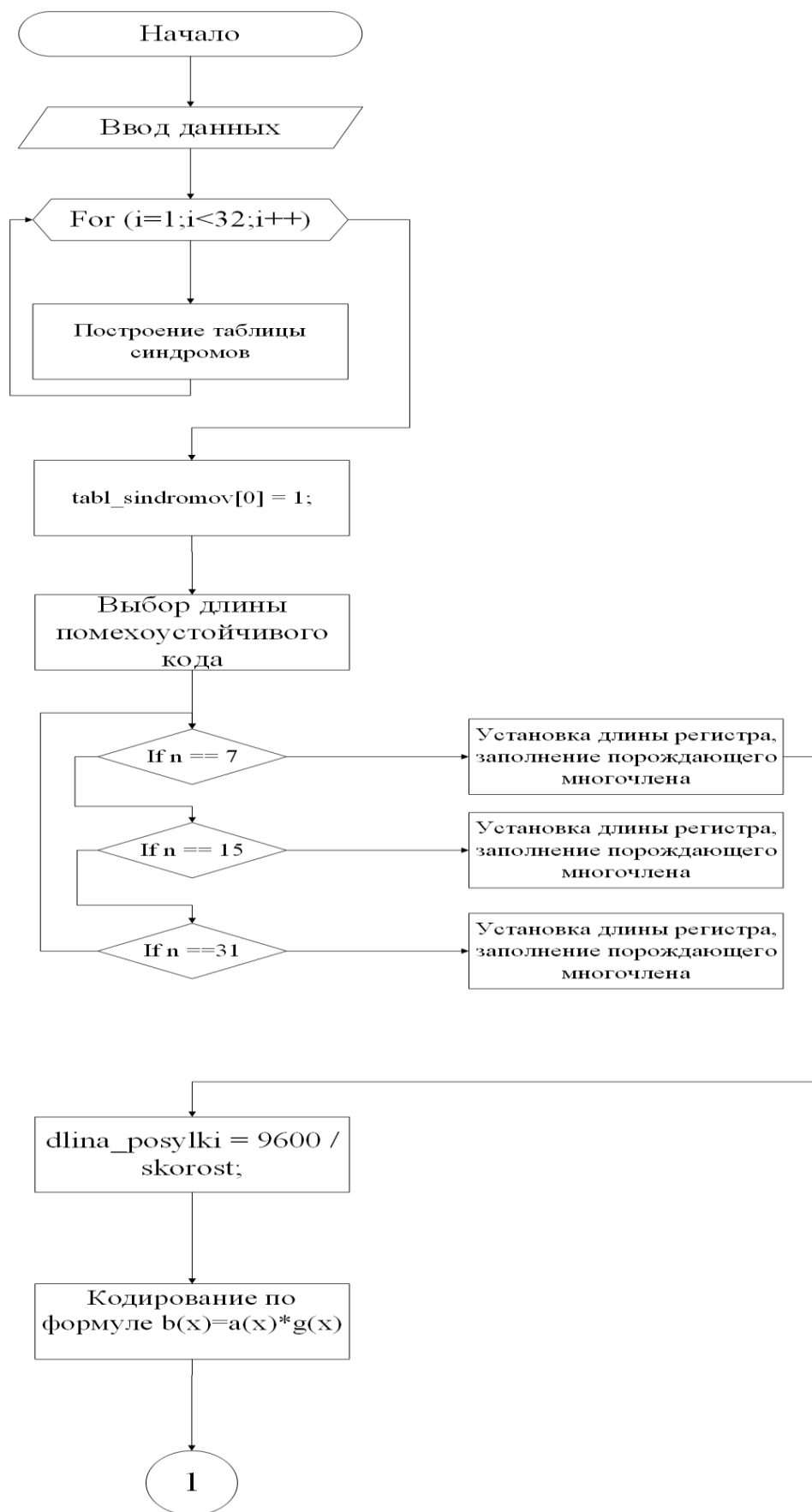


Рисунок 7 – Алгоритм имитации работы помехоустойчивого кодера, помехоустойчивого декодера, модулятора, демодулятора. Лист 1

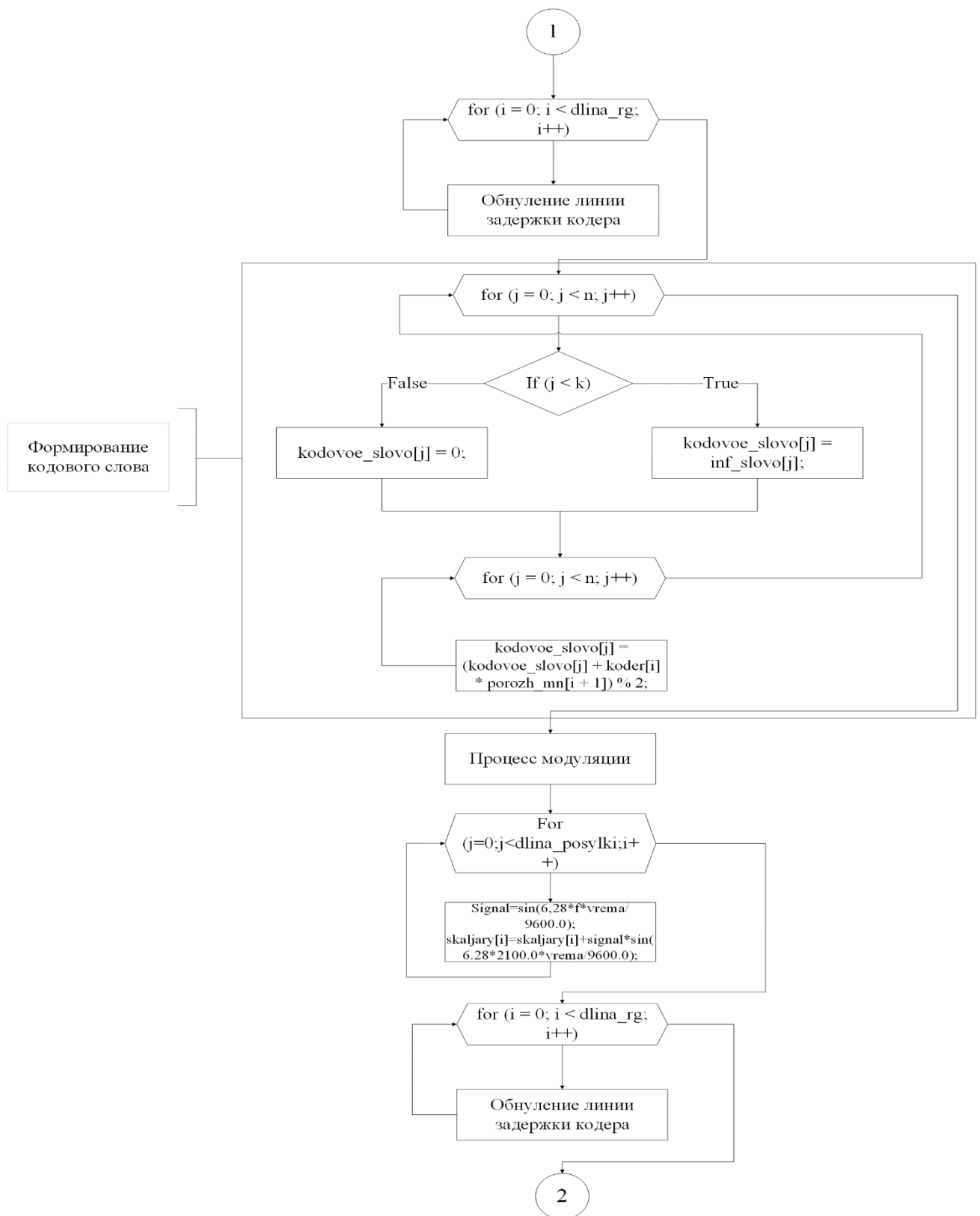


Рисунок 7. Лист 2

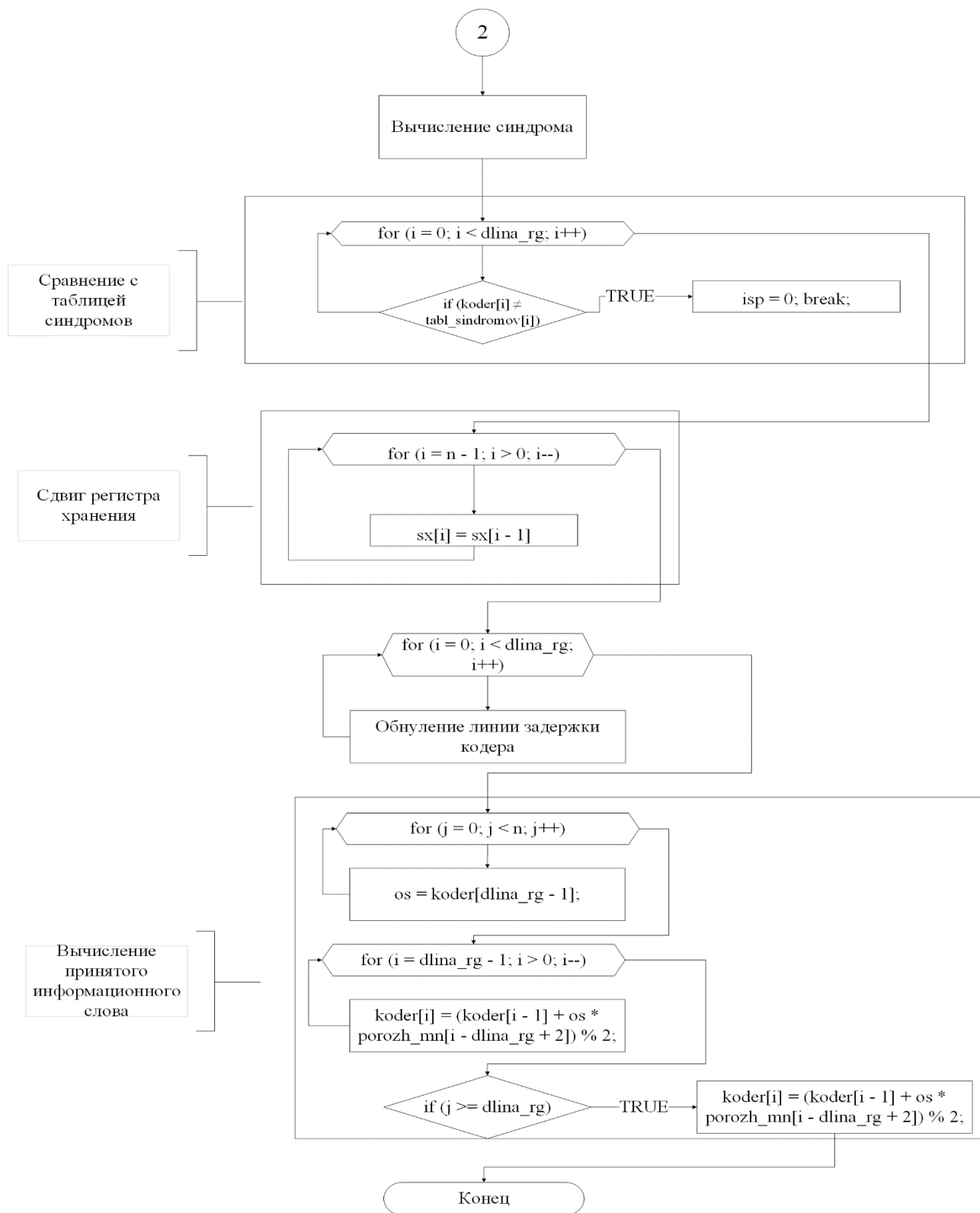


Рисунок 7 . Лист 3

Следующий алгоритм описывает имитацию работы встроенной системы контроля. Результат представлен на рисунке 8.

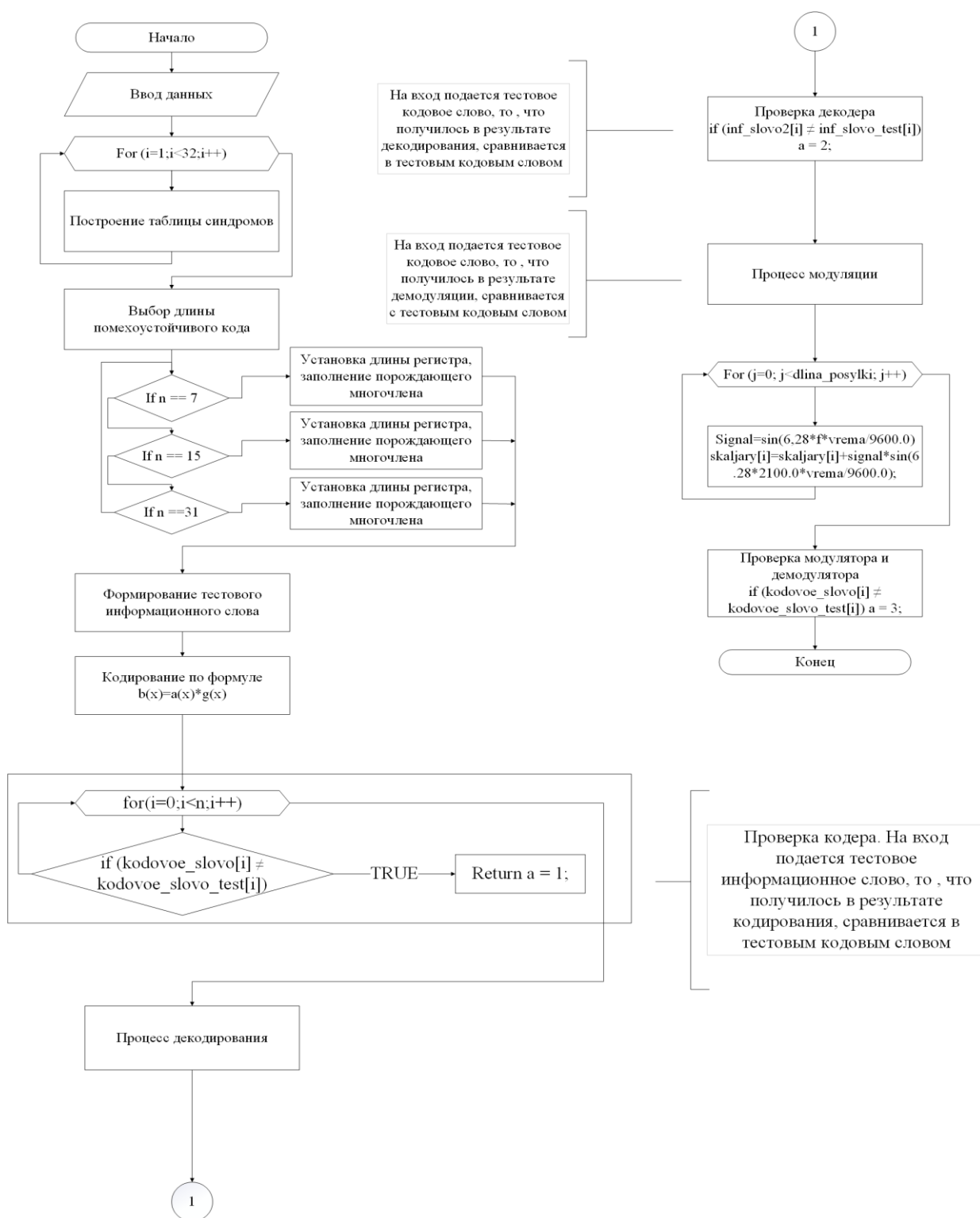


Рисунок 8 – Алгоритм имитации работы встроенной системы контроля

На вход подаются данные. Работа алгоритма начинается с вызова цикла `for`, который, пошагово, заполняет таблицу синдромов. Далее с помощью условий `if/else` происходит проверка на длину помехоустойчивого кода. После срабатывания, одной из трех проверок на длину кода, происходит установка длины регистра в кодере, затем заполняется порождающий многочлен. Затем формируется тестовое информационное слово. Обнуляется линия задержки регистра в кодере. Затем формируется кодовое слово и выполняется его кодирование. Далее с помощью цикла `for` и условия `if/else` выполняется проверка кодера. Для проверки на вход подается тестовое информационное слово, то, что получилось в результате кодирования, сравнивается с тестовым кодовым словом. Затем выполняется процесс декодирования и выполняется проверка декодера. Для проверки на вход подается тестовое кодовое слово, то, что получилось в результате декодирования, сравнивается с тестовым информационным словом. Далее последовательно выполняется процесс модуляции и демодуляции. На последнем этапе происходит выполнение проверки модулятора и демодулятора. Для этого на вход подается тестовое кодовое слово, то, что получилось в результате демодуляции, сравнивается с тестовым кодовым словом.

5 Разработка программного средства

В данном разделе выпускной квалификационной работы описывается разработанное программное средство, имитирующее, возникновение, обнаружение и устранение отказов в аппаратуре передачи данных.

Программа была написана на языке «С#» с использованием библиотек Windows Forms. Программа состоит из модулей: «Form1.cs», «Program.cs», «Form1.Designer.cs». В модуле «Program.cs» находится текст функции main, имеющий стандартное содержание для проектов с использованием Windows Forms, а именно вызов методов класса Application. В модуле «Form1.cs» находится описание класса окна программы. В модуле «Form1.Designer.cs» описаны элементы интерфейса, используемые на окне программы.

Код разработанного программного средства, имитирующее, возникновение, обнаружение и устранение отказов в аппаратуре передачи данных представлен в приложении Б.

Класс «Form» описывает окно программы. Члены класса переменные приведены в таблице 1.

Таблица 1 – Члены класса переменные

Тип	Название	Назначение
System::Windows::Forms::Button^	button1	Кнопка «Диагностика»
System::Windows::Forms::Button^	button2	Кнопка «Отправить сообщение»
System::Windows::Forms::Button^	button3	Кнопка «Заменить неисправный блок»
System::Windows::Forms::Label^	label1	Надпись «Состояние системы»
System::Windows::Forms::Label^	label2	Надпись «Вид отказа»
System::Windows::Forms::Label^	label3	Надпись «Скорость передачи»
System::Windows::Forms::Label^	label4	Надпись «Длина помехоустойчивого кода»
System::Windows::Forms::Label^	label5	Надпись «Ввод сообщения»

Продолжение таблицы 1

Тип	Название	Назначение
System::Windows::Forms::Label^	label6	Надпись «Вывод сообщения»
System::Windows::Forms::TextBox^	textBox1	Поле для ввода передаваемого сообщения
System::Windows::Forms::radioButton3^	radioButton3	Выбор длины помехоустойчивого кода -31 бит
System::Windows::Forms::radioButton4^	radioButton4	Выбор частотного вида модуляции
System::Windows::Forms::radioButton5^	radioButton5	Выбор относительной фазовой вида модуляции
System::Windows::Forms::radioButton6^	radioButton6	Выбор скорости передачи - 300 бит/с
System::Windows::Forms::radioButton7^	radioButton7	Выбор скорости передачи - 600 бит/с
System::Windows::Forms::RadioButton8^	radioButton8	Выбор скорости передачи - 1200 бит/с
System::Windows::Forms::RadioButton9^	radioButton9	Выбор вида отказа: отказ кодера
System::Windows::Forms::RadioButton10^	radioButton10	Выбор вида отказа: отказ модулятора
System::Windows::Forms::RadioButton11^	radioButton11	Выбор вида отказа: отказ декодера
System::Windows::Forms::RadioButton12^	radioButton12	Выбор вида отказа: отказ демодулятора
System::Windows::Forms::RadioButton13^	radioButton13	Выбор вида отказа: отказов нет
System::Windows::Forms::TextBox^	textBox3	Поле для ввода состояния системы
System::Windows::Forms::listBox^	listBox1	Поле для вывода передаваемого сообщения
System::Windows::Forms::panel1^	panel1	Блок выбора длины помехоустойчивого кода

Продолжение таблицы 1

Тип	Название	Назначение
System::Windows::Forms::panel2^	panel2	Блок выбора вида модуляции
System::Windows::Forms::panel3^	panel3	Блок выбора скорости передачи
System::Windows::Forms::panel4^	panel4	Блок вида выбора отказа
System::Windows::Forms::progressBar1^	progressBar1	Индикатор проведения диагностики
System::Windows::Forms::radioButton1^	radioButton1	Выбор длины помехоустойчивого кода – 7 бит
System::Windows::Forms::radioButton2^	radioButton2	Выбор длины помехоустойчивого кода – 15 бит

Члены класса методы приведены в таблице 2.

Таблица 2 – Члены класса методы

Тип возвращаемого значения	Название и параметры	Назначение
	MyForm(void)	Конструктор
	~MyForm()	Деструктор
System::Void	Button2_Click(System::Object^ sender, System::EventArgs^ e)	Обработчик нажатия кнопки «Отправить сообщение»
System::Void	Button1_Click(System::Object^ sender, System::EventArgs^ e)	Обработчик нажатия кнопки «Диагностика»
System::Void	Button3_Click(System::Object^ sender, System::EventArgs^ e)	Обработчик нажатия кнопки «Заменить неисправный блок»

Продолжение таблицы 2

Тип возвращаемого значения	Название и параметры	Назначение
int	kontrol()	Моделирование контроля в схеме
void	modelir()	Моделирование работы аппаратуры передачи данных с отказами без их обнаружения
void	timer1_Tick	Вывод результатов контроля

В функции `button2_Click` происходит проверка значений, которые выбрал пользователь при помощи радиокнопок:

```

if (textBox1.Text == "")
{
    MessageBox.Show("Вы не ввели сообщение!"); return;
}

if (radioButton1.Checked == true) dlinakoda = 7;

if (radioButton2.Checked == true) dlinakoda = 15;
if (radioButton3.Checked == true) dlinakoda = 31;

if (radioButton4.Checked == true) vidimodulyatsii = 1;

if (radioButton5.Checked == true) vidimodulyatsii = 2;

if (radioButton6.Checked == true) skorost = 300;
if (radioButton7.Checked == true) skorost = 600;
if (radioButton8.Checked == true) skorost = 1200;

if (radioButton9.Checked == true) vidotkaza = 1;

```

```

if (radioButton10.Checked == true) vidotkaza = 2;
if (radioButton11.Checked == true) vidotkaza = 3;
if (radioButton12.Checked == true) vidotkaza = 4;
    if (radioButton13.Checked == true) vidotkaza = 0;

```

После этого происходит очистка поля сообщения:

```
listBox1.Items.Clear();
```

Далее происходит преобразование введенного пользователем сообщения в последовательность бит:

```

String[] ss = textBox1.Text.Split(' ');
    n = ss.Count();
    for (int i = 0; i < n; i++)
    {
        inf_slovo[i] = int.Parse(ss[i]);
    }

```

Вызывается функция моделирования работы аппаратуры передачи данных с отказами:

```
modelir(dlinakoda, vidimodulyatsii, skorost, vidotkaza);
```

После чего полученное информационное слово выводится в соответствующее поле:

```

listBox1.Items.Clear();
    for (int i = 0; i < n; i++)
    {
        listBox1.Items.Add(inf_slovo2[i]);
    }
}

```

В функции `button1_Click` происходит диагностика, то есть имитируется работа встроенной системы контроля. Сначала надпись состояния системы и кнопка «заменить неисправный блок» становятся невидимыми:

```
label1.Visible = false;
```

После этого запускается таймер, который необходим для отображения на экране прогресса выполнения диагностики.

В функции `button3_Click` происходит замена неисправного блока:
`MessageBox.Show("Замена неисправного блока осуществлена",
"Диагностика", MessageBoxButtons.OK);`

Функции `kontrol()` реализует алгоритм встроенной системы контроля. Внутри функции объявлены переменные:

```
int i, j;  
  
int[] tabl_sindromov = new int[32]; //таблица синдромов  
int[] porozh_mn = new int[32]; //порождающий многочлен  
  
int k = 0; //длина информационног слова  
int dlina_rg = 0; //количество проверочных разрядов, оно же  
длина регистра сдвига в кодере  
int t = 1; //количество исправляемых ошибок  
  
int[] koder = new int[32]; //линия задержки при делении и  
умножении информационного слова на порождающий многочлен  
int[] inf_slovo_test = new int[32]; //тестовое информационное  
слово  
int[] kodovoe_slovo = new int[32]; //кодовое слово после  
кодирования  
int[] inf_slovo2 = new int[32]; //информационное слово после  
декодирования  
int[] kodovoe_slovo_test = new int[32]; //тестовое кодовое слово  
после кодирования  
  
int kol_sindr = 1; //количество классов эквивалентности  
double signal;  
double[] skaljary = new double[4];
```

```

double f;
int dlina_posylki;
int vrema = 0;
int i1, os = 0, os1 = 0, isp = 0;
int[] sx = new int[32]; //регистр хранения в декодере

```

Сначала формируется синдром для помехоустойчивого декодера:

```

for (i = 1; i < 32; i++) tabl_sindromov[i] = 0;
    tabl_sindromov[0] = 1;

```

Затем задаются параметры помехоустойчивого кода, которые будут моделироваться (длина информационного слова и порождающий многочлен)

```

    if (n == 7) { dlina_rg = 3; k = 4; porozh_mn[0] = 1; porozh_mn[1] = 0;
porozh_mn[2] = 1; porozh_mn[3] = 1; }
        if (n == 15) { dlina_rg = 4; k = 11; porozh_mn[0] = 1;
porozh_mn[1] = 0; porozh_mn[2] = 1; porozh_mn[3] = 1; porozh_mn[4] = 1; }
            if (n == 31) { dlina_rg = 5; k = 26; porozh_mn[0] = 1;
porozh_mn[1] = 0; porozh_mn[2] = 1; porozh_mn[3] = 1; porozh_mn[4] = 0;
porozh_mn[5] = 1; }

```

После этого формируется тестовое информационное слово:

```

for (i = 0; i < k; i = i + 2) { inf_slovo_test[i] = 0; inf_slovo_test[i + 1] = 1; }

```

Выполняется процедура кодирования, в результате которой формируется тестовое кодовое слово:

```

for (i = 0; i < dlina_rg; i++) koder[i] = 0; //обнулить линию задержки кодера

```

//формирование кодового слова

```

    for (j = 0; j < n; j++)
    {
        if (j < k) kodovoe_slovo_test[j] = inf_slovo_test[j];
        else kodovoe_slovo_test[j] = 0;
        for (i = 0; i < dlina_rg; i++)
        {

```

```

                                kodovoe_slovo_test[j] = (kodovoe_slovo_test[j] +
koder[i] * porozh_mn[i + 1]) % 2;
                                }
                                for (i = dlina_rg - 1; i > 0; i--)
                                {
                                        koder[i] = koder[i - 1];
                                }
                                if (j < k) koder[0] = inf_slovo_test[j];
                                else koder[0] = 0;
                                }

```

После этого запускается первый тест от встроенной системы контроля. На вход кодера подается тестовое информационное слово. Кодер выполняет кодирование, а затем полученное кодовое слово сравнивается с тестовым кодовым словом. Если пользователям было выбрано моделирование отказа кодера, то процедура кодирования происходит с неизменяемым значением элемента массива koder[1].

```

                                for (i = 0; i < dlina_rg; i++) koder[i] = 0; //обнулить линию задержки кодера

//формирование кодового слова
                                for (j = 0; j < n; j++)
                                {
                                        if (j < k) kodovoe_slovo[j] = inf_slovo_test[j];
                                        else kodovoe_slovo[j] = 0;
                                        for (i = 0; i < dlina_rg; i++)
                                        {
                                                kodovoe_slovo[j] = (kodovoe_slovo[j] + koder[i] *
porozh_mn[i + 1]) % 2;
                                        }
                                        for (i = dlina_rg - 1; i > 0; i--)
                                        {

```

```

        koder[i] = koder[i - 1];
    }
    if (j < k) koder[0] = inf_slovo_test[j];
    else koder[0] = 0;
    if (vid_otkaza == 1) koder[0] = 1;
}
//конец кодирования
for (i = 0; i < n; i++)
{ if (kodovoe_slovo[i] != kodovoe_slovo_test[i]) return 1; }

```

Если тестовое кодовое слово и полученное кодовое слово не совпадают, то функция возвращает значение единица.

После этого запускается второй тест от встроенной системы контроля. На вход декодера подается тестовое кодовое слово. Декодер выполняет декодирование, а затем полученное информационное слово сравнивается с тестовым информационным словом. Если пользователем было выбрано моделирование отказа декодера, то процедура декодирования происходит с неизменяемым значением элемента массива koder[1].

```

for (i = 0; i < dlina_rg; i++)
{
    koder[i] = 0; //обнуление линии задержки
}
if (vid_otkaza == 2) koder[0] = 1;

for (j = 0; j < (n * 2); j++) //вычисление синдрома
{
    os = koder[dlina_rg - 1];
    os1 = sx[n - 1];
    for (i1 = 0; i1 < kol_sindr; i1++)
    {
        isp = 1;
    }
}

```

```

        for (i = 0; i < dlina_rg; i++)// сравнение с таблицей
            синдромов
            {
                if (koder[i] != tabl_sindromov[i]) { isp = 0;
break; }

            }
            if (isp == 1) break;
        }
        for (i = dlina_rg - 1; i > 0; i--)
        {
            koder[i] = (koder[i - 1] + os * porozh_mn[i - dlina_rg
+ 2]) % 2;

        }
        for (i = n - 1; i > 0; i--)
        {
            sx[i] = sx[i - 1];//сдвиг регистра хранения
        }
        if (j < n)
        {
            koder[0] = (kodovoe_slovo_test[j] + os) % 2;

            sx[0] = kodovoe_slovo_test[j];
        }
        else
        {
            koder[0] = (os + isp) % 2;

            sx[0] = (os1 + isp) % 2;
        }
        if (vid_otkaza == 2) koder[0] = 1;

```

```

    }
int oshibki = 0;
    for (j = 1; j < dlina_rg; j++)
    {
        koder[0] = koder[0] + koder[j];
    }
    if (koder[0] != 0)
    {
        oshibki = oshibki + 1; return 2;
    }
    else
    {
        for (i = 0; i < dlina_rg; i++)
        {
            koder[i] = 0; // обнуление линии задержки
        }
        if (vid_otkaza == 2) koder[0] = 1;
        for (j = 0; j < n; j++) // вычисление принятого
информационного слова
        {
            os = koder[dlina_rg - 1];

            for (i = dlina_rg - 1; i > 0; i--)
            {
                koder[i] = (koder[i - 1] + os * porozh_mn[i -
dlina_rg + 2]) % 2;

            }
            koder[0] = (sx[n - 1 - j] + os) % 2;
            if (vid_otkaza == 2) koder[0] = 1;

```

```

        if (j >= dlina_rg) inf_slovo2[j - dlina_rg] = os;
    }
}
for (i = 0; i < k; i++)
    { if (inf_slovo2[i] != inf_slovo_test[i]) return 2; }

```

Если тестовая и полученные информационные слова не совпадают, то функция возвращает значение два.

После этого запускается третий тест от встроенной системы контроля. На вход модулятора подается тестовое кодовое слово. Модулятор производит модуляцию, а затем происходит демодуляция, после чего полученное кодовое слово сравнивается с тестом. Если пользователем было выбрано моделирование отказа модулятора, то процедура модуляции происходит с изменением значения несущей частоты на 100 Гц. Если пользователем было выбрано моделирование отказа демодулятора, то на его выходе всегда формируется значение единица. Так как пользователь может выбрать либо частотную, либо относительную фазовую модуляцию, то перед выполнением этих действий происходит проверка выбора пользователей.

```

    if (vid_mod == 1)
    {
        for (i = 0; i < n; i++)
        { //модуляция
            if (kodovoe_slovo_test[i] == 0) f = 2100.0; else f =
1500.0;

            if ((vid_otkaza == 3) && (f == 2100.0)) f = f + 100.0;

            skaljary[0] = 0; skaljary[1] = 0; skaljary[2] = 0;
skaljary[3] = 0;

            for (j = 0; j < dlina_posylki; j++)

```

```

        {
            signal = Math.Sin(6.28 * f * vreme / 9600.0);
            skaljary[0] = skaljary[0] + signal *
Math.Sin(6.28 * 2100.0 * vreme / 9600.0);
            skaljary[1] = skaljary[1] + signal *
Math.Cos(6.28 * 2100.0 * vreme / 9600.0);
            skaljary[2] = skaljary[2] + signal *
Math.Sin(6.28 * 1500.0 * vreme / 9600.0);
            skaljary[3] = skaljary[3] + signal *
Math.Cos(6.28 * 1500.0 * vreme / 9600.0);
            vreme++;
        }
        if ((skaljary[0] * skaljary[0] + skaljary[1] *
skaljary[1]) > (skaljary[2] * skaljary[2] + skaljary[3] * skaljary[3])) kodovoe_slovo[i]
= 0;

        else kodovoe_slovo[i] = 1;
        if (vid_otkaza == 4) kodovoe_slovo[i] = 1;

    }
} //конец
else
{
    skaljary[0] = -1.0;
    for (i = 0; i < n; i++)
    { //модуляция
        if (i == 0) { if (kodovoe_slovo_test[0] == 0) f = 0.0;
else f = 3.14; }

        else { if (kodovoe_slovo_test[i] ==
kodovoe_slovo_test[i - 1]) f = 0.0; else f = 3.14; }

        skaljary[1] = 0;

```

```

        for (j = 0; j < dlina_posylki; j++)
        {
            if (vid_otkaza == 3) signal = Math.Sin(6.28 *
1900.0 * vreme / 9600.0 + f);

            else signal = Math.Sin(6.28 * 1800.0 * vreme /
9600.0 + f);

            skaljary[1] = skaljary[1] + signal *
Math.Sin(6.28 * 1800.0 * vreme / 9600.0);
            vreme++;
        }
        if (vid_otkaza == 4) skaljary[1] = 1.0;

        if (i == 0)
        {
            if (skaljary[1] < 0) kodovoe_slovo[0] = 0;
            else kodovoe_slovo[i] = 1;
            skaljary[0] = skaljary[1];
        }
        else
        {
            if (((skaljary[1] < 0) && (skaljary[0] < 0)) ||
((skaljary[1] > 0) && (skaljary[0] > 0))) kodovoe_slovo[i] = kodovoe_slovo[i - 1];
            else kodovoe_slovo[i] = (kodovoe_slovo[i - 1]
+ 1) % 2;
        }
    }
}

```

```
for (i = 0; i < n; i++)
{ if (kodovoe_slovo[i] != kodovoe_slovo_test[i]) return 3; }
```

Если тестовое кодовое слово и сформированное не совпадают, то функция возвращает значение три. В противном случае, функция возвращает значение ноль (то есть отказы не были обнаружены)

В функции `modeler` происходит моделирование работы аппаратуры передачи данных без обнаружения отказов. Внутри функции объявлены переменные:

```
int i, j;

int[] tabl_sindromov = new int[32]; //таблица синдромов
int[] porozh_mn = new int[32]; //порождающий многочлен

int k = 0; //длина информационного слова
int dlina_rg = 0; //количество проверочных разрядов, оно же
длина регистра сдвига в кодере
int t = 0; //количество исправляемых ошибок
int n_summ = 0; //количество сумматоров в схеме
деления=количество единиц в порождающем многочлене

int maxi1;

int[] koder = new int [32]; //линия задержки при делении и
умножении информационного слова на порождающий многочлен

int[] kodovoe_slovo =new int[32]; //кодовое слово после
кодирования

int n_zn; //значение неисправного элемента
int n_elem; //номер неисправного элемента
```

```
int n_takt;//номер такта, когда произойдет отказ
int schet_taktov;//счетчик тактов работы
int cikl;//периодичность динамического контроля (в кодовых словах)
```

```
int porog05;//количество подряд идущих обнаруженных неисправляемых ошибок, при котором запускается контроль
```

```
int kol_sindr = 1;//количество классов эквивалентности
```

```
double signal;
```

```
double[] skaljary = new double[4];
```

```
double f;
```

```
int dlina_posylki;
```

```
int vreme = 0;
```

```
int i1, os=0, os1=0, isp=0, oshibki=0;
```

```
int[]sx = new int[32];//регистр хранения в декодере
```

Сначала формируется синдром для помехоустойчивого декодера:

```
for (i = 1; i < 32; i++) tabl_sindromov[i] = 0;
```

```
tabl_sindromov[0] = 1;
```

Затем задаются параметры помехоустойчивого кода, которые будут моделироваться (длина информационного слова и порождающий многочлен)

```
if (n == 7) { dlina_rg = 3; k = 4; porozh_mn[0] = 1; porozh_mn[1] = 0;
porozh_mn[2] = 1; porozh_mn[3] = 1; }
```

```
if (n == 15) { dlina_rg = 4; k = 11; porozh_mn[0] = 1;
porozh_mn[1] = 0; porozh_mn[2] = 1; porozh_mn[3] = 1; porozh_mn[4] = 1; }
```

```
if (n == 31) { dlina_rg = 5; k = 26; porozh_mn[0] = 1;
porozh_mn[1] = 0; porozh_mn[2] = 1; porozh_mn[3] = 1; porozh_mn[4] = 0;
porozh_mn[5] = 1; }
```

Осуществляется кодирование информационного слова, которое ввел пользователь:

```
for (i = 0; i < dlina_rg; i++) koder[i] = 0;//обнулить линию задержки кодера
```

```

//формирование кодового слова
for (j = 0; j < n; j++)
{
    if (j < k) kodovoe_slovo[j] = inf_slovo[j];
    else kodovoe_slovo[j] = 0;
    for (i = 0; i < dlina_rg; i++)
    {
        kodovoe_slovo[j] = (kodovoe_slovo[j] +
koder[i] * porozh_mn[i + 1]) % 2;
    }
    for (i = dlina_rg - 1; i > 0; i--)
    {
        koder[i] = koder[i - 1];
    }
    if (j < k) koder[0] = inf_slovo[j];
    else koder[0] = 0;
    if (vid_otkaza == 1) koder[0] = 1;
}

```

Если пользователь выбрал отказ кодера, то в процедуре не изменяется значение элемента массива `koder[1]`.

Проверяется, выбрал ли пользователь частотную модуляцию. Если выбрал, то происходит формирование отчетов сигнала соответствующих передаваемому кодовому слову, а затем эти отчеты подвергаются процедуре демодуляции. Проверяется условие, выбрал ли пользователь отказ модулятора. Если выбрал, то при выполнении преобразования, значение несущей частоты становится равным 2200 Гц. Если пользователь выбрал отказ демодулятора, то на его выходе всегда формируется значение один:

```

for (i = 0; i < n; i++)
{
    //модуляция

```

```

        if (kodovoe_slovo[i] == 0) f = 2100.0; else f = 1500.0;
        if ((vid_otkaza == 3) && (f == 2100.0)) f = f + 100.0;

        skaljary[0] = 0; skaljary[1] = 0; skaljary[2] = 0;
        skaljary[3] = 0;

        for (j = 0; j < dlina_posylki; j++)
        {
            signal = Math.Sin(6.28 * f * vreme / 9600.0);
            skaljary[0] = skaljary[0] + signal *
Math.Sin(6.28 * 2100.0 * vreme / 9600.0);
            skaljary[1] = skaljary[1] + signal *
Math.Cos(6.28 * 2100.0 * vreme / 9600.0);
            skaljary[2] = skaljary[2] + signal *
Math.Sin(6.28 * 1500.0 * vreme / 9600.0);
            skaljary[3] = skaljary[3] + signal *
Math.Cos(6.28 * 1500.0 * vreme / 9600.0);
            vreme++;
        }
        if ((skaljary[0] * skaljary[0] + skaljary[1] *
skaljary[1]) > (skaljary[2] * skaljary[2] + skaljary[3] * skaljary[3])) kodovoe_slovo[i]
= 0;
        else kodovoe_slovo[i] = 1;
            if (vid_otkaza == 4) kodovoe_slovo[i] = 1;
        }
    }
}

```

Если пользователь выбрал относительную частотную модуляцию, то в случае необходимости моделирования отказа модулятора, значение несущей частоты становится равным 1900 Гц., а в остальном процедура аналогична описанной выше

```

skaljary[0] = -1.0;

```

```

for (i = 0; i < n; i++)
{
    //модуляция
    if (i == 0) { if (kodovoe_slovo[0] == 0) f = 0.0; else f
= 3.14; }

    else { if (kodovoe_slovo[i] == kodovoe_slovo[i - 1])
f = 0.0; else f = 3.14; }

    skaljary[1] = 0;
    for (j = 0; j < dlina_posylki; j++)
    {
        if (vid_otkaza == 3) signal = Math.Sin(6.28 *
1900.0 * vreme / 9600.0 + f);

        else signal = Math.Sin(6.28 * 1800.0 * vreme /
9600.0 + f);

        skaljary[1] = skaljary[1] + signal *
Math.Sin(6.28 * 1800.0 * vreme / 9600.0);
        vreme++;
    }
    if (vid_otkaza == 4) skaljary[1] = 1.0;
    if (i == 0)
    {
        if (skaljary[1] < 0) kodovoe_slovo[0] = 0;
        else kodovoe_slovo[i] = 1;
        skaljary[0] = skaljary[1];
    }
    else
    {
        if (((skaljary[1] < 0) && (skaljary[0] < 0)) ||
((skaljary[1] > 0) && (skaljary[0] > 0))) kodovoe_slovo[i] = kodovoe_slovo[i - 1];
        else kodovoe_slovo[i] = (kodovoe_slovo[i - 1]
+ 1) % 2;
    }
}

```

```

    }
}

```

Далее происходит декодирование кодового слова и исправление обнаруженных в нем ошибок. Если пользователь выбрал отказ кодера, то в процедуре не изменяется значение элемента массива `koder[1]`.

```

for (i = 0; i < dlina_rg; i++)
{
    koder[i] = 0; // обнуление линии задержки
}
if (vid_otkaza == 2) koder[0] = 1;
for (j = 0; j < (n * 2); j++) // вычисление синдрома
{
    os = koder[dlina_rg - 1];
    os1 = sx[n - 1];
    for (i1 = 0; i1 < kol_sindr; i1++)
    {
        isp = 1;
        for (i = 0; i < dlina_rg; i++) // сравнение с таблицей
            синдromов
            {
                if (koder[i] != tabl_sindromov[i]) { isp = 0;
                break; }
            }
        if (isp == 1) break;
    }
    for (i = dlina_rg - 1; i > 0; i--)
    {
        koder[i] = (koder[i - 1] + os * porozh_mn[i - dlina_rg
        + 2]) % 2;
    }
}

```

```

for (i = n - 1; i > 0; i--)
{
    sx[i] = sx[i - 1]; //сдвиг регистра хранения
}
if (j < n)
{
    koder[0] = (kodovoe_slovo[j] + os) % 2;
    sx[0] = kodovoe_slovo[j];
}
else
{
    koder[0] = (os + isp) % 2;
    sx[0] = (os1 + isp) % 2;
}
if (vid_otkaza == 2) koder[0] = 1;
}
//деление закончилось, получился синдром
for (j = 1; j < dlina_rg; j++)
{
    koder[0] = koder[0] + koder[j];
}
if (koder[0] != 0)
{
    oshibki = oshibki + 1;
}
else
{
    for (i = 0; i < dlina_rg; i++)
    {
        koder[i] = 0; // обнуление линии задержки
    }
}

```

```

    }
    if (vid_otkaza == 2) koder[0] = 1;

    for (j = 0; j < n; j++)//вычисление принятого
информационного слова
    {
        os = koder[dlina_rg - 1];
        for (i = dlina_rg - 1; i > 0; i--)
        {
            koder[i] = (koder[i - 1] + os * porozh_mn[i -
dlina_rg + 2]) % 2;
        }
        koder[0] = (sx[n - 1 - j] + os) % 2;
        if (vid_otkaza == 2) koder[0] = 1;
        if (j >= dlina_rg) inf_slovo2[j - dlina_rg] = os;
    }
}

```

В конструкторе класса происходит выделение памяти для массивов переданного информационного слова и принятого информационного слова.

```

inf_slovo = new int[32];//информационное слово
inf_slovo2 = new int[32];

```

В функции `timer` происходит вызов функции `контроль`, а затем значения, которые она вернула, в соответствии с результатом выводится соответствующая надпись и становится видимой строка состояния и кнопка «заменить блок».

```

int a = 0;

a = kontrol(dlinakoda, vidimodulyatsii, skorost, vidotkaza);
if (a == 0) { textBox3.Text = "Норма"; return; }
if (a == 1) textBox3.Text = "Отказ кодера";
if (a == 2) textBox3.Text = "Отказ декодера";

```

```
if (a == 3) textBox3.Text = "Отказ УПС";  
label1.Text = "Состояние системы: обнаружены отказы";  
label1.Visible = true;  
button3.Visible = true;
```

Кроме этой функции реализована остановка таймера и прорисовка индикатора загрузки.

```
progressBar1.Value = progressBar1.Value + 1;
```

```
if (progressBar1.Value == 300)  
{  
    timer1.Stop();  
    Thread.Sleep(100);  
}
```

6 Разработка инструкции пользователя

В данном разделе выпускной квалификационной работы было разработана инструкция пользователя, которая представлено в приложении А «Разработка руководства оператора». В данном руководстве пошагово описаны все действия, которые необходимо выполнить студенту во время выполнения лабораторной работы по дисциплине «Разработка и эксплуатация защищенных автоматизированных систем».

Каждый студент получает свой персональный вариант задания, в котором имеются все необходимые параметры для успешного выполнения лабораторной работы.

7 Вредные психофизиологические факторы, влияющие на опорно-двигательный аппарат специалиста по защите информации. Методы и средства физической культуры, снижающие их воздействие

Любая профессиональная деятельность несет в себе определенную степень риска профессиональных заболеваний.

Казалось бы, у людей, вынужденных проводить перед монитором компьютера долгие часы, работа спокойная и «физически легкая», поэтому профессиональные болезни им не грозят. Однако на самом деле это не так. Любая малоподвижная и сидячая деятельность создает множество предпосылок для развития серьезных заболеваний.

Однако эта деятельность имеет свои специфические минусы, самой большой проблемой специалиста по ИБ является малоподвижный образ жизни. Ведь творческая работа порой настолько поглощает, что ты и не задумываешься, сколько часов проводишь перед монитором.

Основные вредные факторы, влияющие на состояние здоровья людей, работающих за компьютером:

- сидячее положение в течение длительного времени;
- воздействие электромагнитного излучения монитора;
- утомление глаз, нагрузка на зрение;
- перегрузка суставов кистей;
- стресс при потере информации;
- пыль и грязь: аллергия и кишечные инфекции.

Работая за компьютером, человек должен долгое время пребывать в вынужденном положении, двигательная активность сильно ограничивается, что, естественно, оказывает негативное воздействие на позвоночник и мышцы спины. Длительное статическое положение тела приводит к постепенной атрофии мышц шеи, спины, грудной клетки и брюшного пресса. Ослабленные мышцы оказываются не в состоянии поддерживать позвоночник в правильном физиологическом положении и в итоге возникают нарушения осанки,

переходящие в дальнейшем в искривления позвоночника. Чаще всего позвоночник искривляется в боковой плоскости, и это заболевание называется сколиозом. Еще одна причина развития нарушения осанки и искривления позвоночника - сидение на рабочем месте в неправильной позе. Вначале человек стремится контролировать себя и держать правильную осанку, но, погрузившись в процесс работы, утрачивает контроль над положением корпуса, и со временем вообще перестает обращать внимание на то, как он сидит. Все это создает условия, как для возникновения заболевания, так и для его прогрессирования.

Многочасовое нахождение в статичной позе ухудшает обменные процессы в веществе межпозвонковых дисков, что вызывает их преждевременное старение. Межпозвоночные диски утрачивают упругость, теряют воду и постепенно растрескиваются. Этот патологический процесс приводит к развитию остеохондроза. Дальнейшее прогрессирование заболевания заканчивается тем, что через образовавшиеся трещины начинается постепенное выпадение пульпозного ядра, т. е. образуется межпозвоночная грыжа. При межпозвоночной грыже нередко наблюдается сдавление и реактивное воспаление нервных корешков, отходящих от спинного мозга, называемое радикулитом.

Работа с компьютерной мышью и клавиатурой в течение длительного времени вызывает значительное перенапряжение сухожильно-связочного аппарата, мышц предплечья и кистей рук. В процессе работы руки программиста или оператора ПК несут достаточно низкую физическую нагрузку, а перенапряжение связано с выполнением однотипных движений в течение всего трудового дня. Перенапряжение может послужить причиной возникновения достаточно серьезного нервно-мышечного расстройства.

На подушечках пальцев имеется огромное количество нервных окончаний. Эти чувствительные окончания в процессе работы с клавиатурой постоянно раздражаются, что приводит к нарушению передачи нервных импульсов по проводящим путям от кончиков пальцев к коре головного мозга.

В результате появляются судороги мышц предплечья и кистей рук, а также нарушается мелкая моторика кисти. Английские врачи, первые описавшие данное заболевание, назвали его *repetitive strain injury (RSI)*, что переводится как повторяющаяся травма, растяжение.

В регламентированные перерывы с целью снижения развивающегося у пользователей нервно-эмоционального напряжения, утомления зрительного анализатора, целесообразно выполнять комплексы специальных профилактико-реабилитационных упражнений.

В целях профилактики и предупреждения заболеваний специалистов по защите информации необходимо проводить специальную оздоровительную систему профреабилитации от последствий гиподинамии. При этом должны выполняться комплексные меры, включающие физические упражнения, водно-воздушные процедуры, массаж, прием витаминов и микроэлементов, минеральных веществ, аутогенную тренировку.

Лучший способ повышения двигательной активности – спорт и активный отдых.

Существует много форм физической культуры, которые используются для нормализации функционального состояния человека, а также для профилактики заболеваний:

- подвижные спортивные игры являются хорошим средством физического развития, нормализация психоэмоционального состояния, совершенствования опорно-двигательного аппарата и мускулатуры;
- ходьба и бег. Имеют огромное для профилактики опорно-двигательного аппарата и оздоровление организма;
- велосипедный спорт способствует усилению обмена веществ, тренировке кардиореспираторной системы и др. велопрогулки показаны при заболеваниях сердечно - сосудистой, дыхательной систем и нарушении обмена веществ, а также при последствиях травм суставов ног (для разработки тугоподвижности и тренировки мышц). Зимой велопрогулки заменяются упражнениями на велотренажерах;

— плавание - отличное тренирующее средство и закаливает. Если плавание сочетать с прогулками, играми, то оздоравливающий эффект еще более возрастает. Плавание усиливает деятельность кардиореспираторной системы и обмен веществ, а при травмах и заболеваниях позвоночника ведет к исчезновению болей и улучшению подвижности в суставах. Осторожность следует проявлять при заболеваниях сердца, легких, ЛОР - органов, особенно в осенне-зимнее время из-за опасности переохлаждения и обострения или возникновения простуды.

Как видно тихая работа программиста очень опасна, но правильная оценка угроз и здоровый образ жизни уменьшают их до минимума и зачастую исключают болезни у специалистов по защите информации.

Заключение

В ходе выполнения выпускной квалификационной работы были рассмотрены следующие вопросы:

- обзор отказов в аппаратуре, приводящих к нарушению свойств информационной безопасности данных;
- анализ принципов построения и алгоритмов работы встроенных систем контроля;
- описание и выбор реализации моделируемой аппаратуры передачи данных;
- разработка алгоритма имитации работы аппаратуры передачи данных и возникновения отказов в ней;
- разработка программного средства;
- разработка инструкции пользователя;
- результаты тестирования программного средства;
- вредные психофизиологические факторы, влияющие опорно-двигательный аппарат специалиста по защите информации. Методы и средства физической культуры, снижающие их воздействия.

Были выполнены следующие требования:

- программное средство должно быть имитационной моделью аппаратуры передачи данных, в которой в процессе работы возникают отказы;
- должна моделироваться частотная и относительная фазовая модуляции для скоростей передачи данных от 300 бит в секунду и 1200 бит в секунду;
- должен моделироваться циклический помехоустойчивый код с длиной кодового слова от 7 до 31 бита;
- должны моделироваться следующие неисправности:
 - отказ кодера;
 - отказ декодера;
 - отказ модулятора;

- отказ демодулятора;
- интерфейс программного средства должен отображать:
 - результаты работы встроенной системы контроля по обнаружению отказа;
 - последствия, к которым приводят отказы;
- программное средство должно быть реализована на языке программирования «C#»;
- программное средство должно функционировать в операционной системой Windows XP, Windows 7 или Windows 10.

В результате было разработано программное средство, которое имитирует возникновение, обнаружение и устранение отказов в аппаратуре связи.

Таким образом все пункты технического задания выполнены в полном объеме.

Список используемых источников

1 ГОСТ 26656-85 «Техническая диагностика. Контролепригодность. Общие требования» [электронный ресурс]. Режим доступа: <http://docs.cntd.ru/document/1200010775>.

2 ГОСТ 20911-89 «Техническая диагностика. Термины и определения» [электронный ресурс]. Режим доступа: <http://docs.cntd.ru/document/1200009481>.

3 ГОСТ 19919-74 «Контроль автоматизированный технического состояния изделий авиационной техники. Термины и определения» [электронный ресурс]. Режим доступа: <http://docs.cntd.ru/document/1200009359>.

4 Информационная сеть «Техэксперт» [электронный ресурс]. Режим доступа: www.cntd.ru.

5 Практическое задание №19 «Неисправности аппаратуры и их устранение» [учебное пособие]/ М. Тимершин - Москва: Московский государственный технический университет имени Н.Э. Баумана (МГТУ), 2015. - 18 с.

6 «Введение в техническую диагностику и неразрушающий контроль» [учебное пособие] / А. П. Наumenко – Омск: Омский государственный технологический университет (ОмГТУ), 2019. – 153 с.

7 «Методы поиска и устранения неисправностей» [электронный ресурс]. Режим доступа: <http://www.mkuznecov.ru/metodi.html>.

8 «Теоретические основы передачи данных. Часть 2. Основы построения каналов передачи данных» [учебное пособие]. / Киев – Киевское высшее военное инженерное училище связи имени М. И. Калинина, 1991 г. – 172 с.

9 «Физическая культура студента» [учебное пособие]. / Под ред. В. И. Ильинича, М. Гардарики – Москва: Московский государственный университет имени М. В. Ломоносова, 2000.

- 10 «Психофизиологические элементы условий труда» [электронный ресурс]. Режим доступа: <https://studref.com/431779/tehnika/psihofiziologiche>.
- 11 «Циклические коды (теория)» [электронный ресурс]. Режим доступа: <https://studfile.net/preview/2933526/page:4/>.
- 12 «Корректирующие коды и их свойства» [электронный ресурс]. Режим доступа: http://tors.psuti.ru/metod_web/Lekc_TI_for_RPIS_part3.pdf.
- 13 «Помехоустойчивое кодирование, линейные блочные коды. Адаптивное арифметическое кодирование, полиномиальные коды» [электронный ресурс]. Режим доступа: <http://kpet-ks.ru>.
- 14 «Классы эквивалентности, граничные значения» [электронный ресурс]. Режим доступа: <https://training.qatestlab.com/blog/technical-articles/equivalence-classes-and-boundary-values/>.
- 15 ГОСТ 18421-93 «Аналоговая и аналого-цифровая вычислительная техника. Термины и определения» [электронный ресурс]. Режим доступа: <http://docs.cntd.ru/document/1200015672>.
- 16 ГОСТ 27.002-2015 «Надежность в технике (ССНТ). Термины и определения» [электронный ресурс]. Режим доступа: <http://docs.cntd.ru/document/1200136419>.
- 17 «Общее описание сущности сигнатурного анализа» [электронный ресурс]. Режим доступа: https://studopedia.su/9_57411_obshchee-opisanie-sushchnosti-signaturnogo-analiza.html.
- 18 «Кодовое расстояние и корректирующая способность кода» [электронный ресурс]. Режим доступа: <http://www.techproof.ru/prunids-1200-1.html>.
- 19 Шахтарин Б.И. и др. Синхронизация в радиосвязи и радионавигации: Учебн. Пособие / Б.И. Шахтарин, А.А. Иванов, П.И. Кобылкина, М.А. Рязанова, А.А. Самохвалов, Ю.А. Сидоркина, А.А. Тимофеев. – М.: Гелиос АРВ, 2007 – 256 с.: ил.
- 20 Долгов В.А. Встроенные автоматизированные системы контроля. Учебное пособие - <http://padaread.com/?book=17131&pg=22>
- 21 Встроенный контроль цифровых систем -

https://vuzlit.ru/2239733/vstroennyy_kontrol_tsifrovyyh_sistem.

22 Анализ угроз информационной безопасности -

<https://www.docsity.com/ru/analiz-ugroz-2/1835044/>.

Приложение А

(обязательное)

Разработка руководства пользователя

Запустите приложение «form1». Перед вами высветится окно, как показано на рисунке А.1.

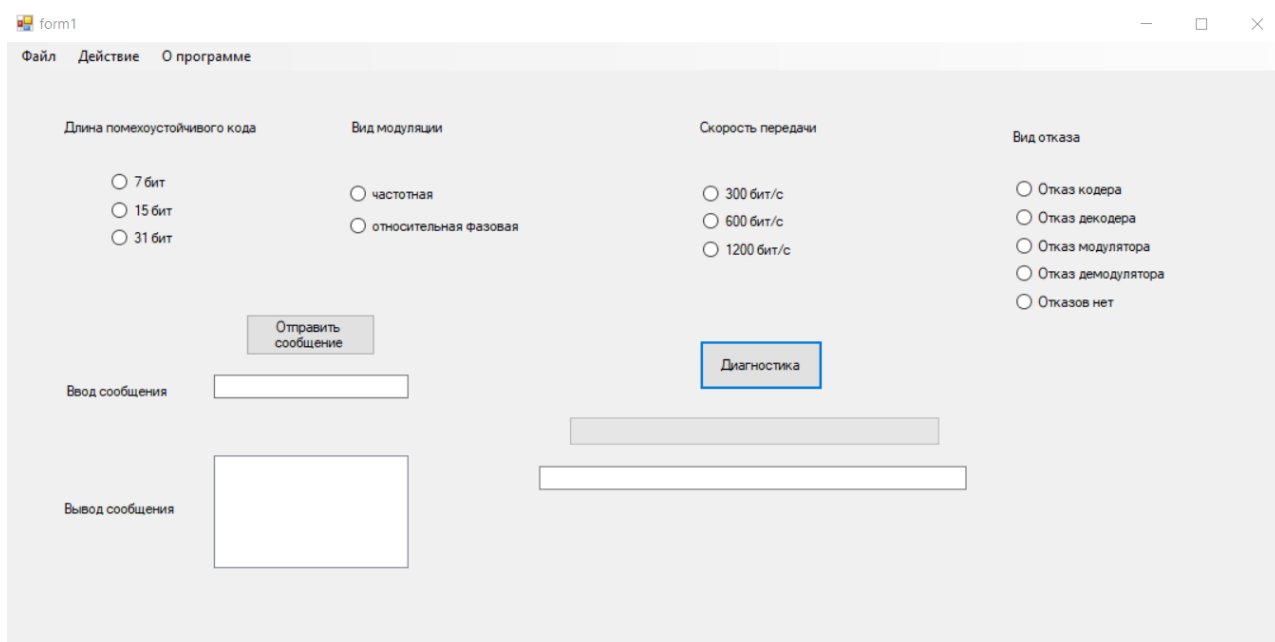


Рисунок А.1 – Общий вид графического интерфейса программы

В панели «Длина помехоустойчивого кода» необходимо выбрать значение, которое соответствует варианту студента. Результат представлен на рисунке А.2.

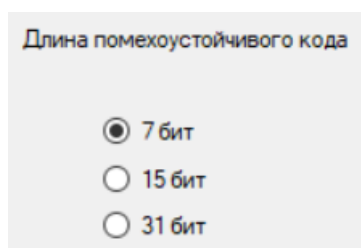


Рисунок А.2 – выбор длины кода

В панели «Вид модуляции» необходимо выбрать значение, которое соответствует варианту студента. Результат представлен на рисунке А.3.

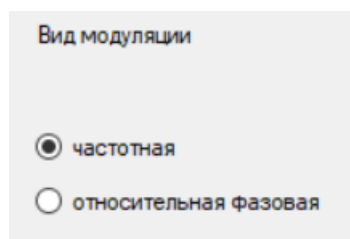


Рисунок А.3 – Выбор вида модуляции

В панели «Скорость передачи» необходимо выбрать значение, которое соответствует варианту студента. Результат представлен на рисунке А.4.

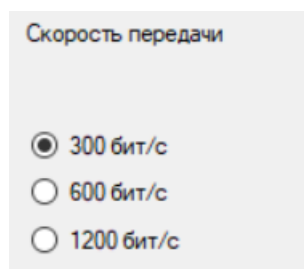


Рисунок А.4 – Выбор скорости передачи

В панели «Вид отказа» необходимо выбрать значение, которое соответствует варианту студента. Результат представлен на рисунке А.5.

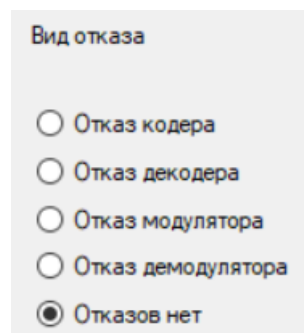


Рисунок А.5 – Выбор вида отказа

В поле «Ввод сообщения» необходимо ввести двоичные комбинации, которые будут передаваться. Длина передающейся комбинации должна соответствовать значению в панели «Длина помехоустойчивого кода», пример: 1001. После этого необходимо нажать кнопку «Отправить сообщение». Программное средство выдаст результат, который будет отражен в поле «Вывод сообщения»

В результате должен получиться интерфейс, как показано на рисунке А.6.

Отправить сообщение

Ввод сообщения 1 0 0 1

Вывод сообщения 1 0 0 1

Рисунок А.6 – Пример отправки сообщения

Далее можно провести диагностику программного средства, для этого следует нажать на кнопку «Диагностика». Если перед тем как была нажата кнопка «Диагностика» вывод сообщения совпал с вводом сообщения, то система выдаст, что программное средство в норме. Результат данного действия приведен на рисунке А.7.

Диагностика

Норма

Рисунок А.7 – Результат проведения диагностики

Для того, чтобы смоделировать отказ в системе, необходимо выбрать в панели «Вид отказа» конкретный отказ, который зависит от варианта студента, выполняющего работу. Пример: Выбрать значение «отказ кодера» и нажать на кнопку «Отправить сообщение». Результат приведен на рисунке А.8.

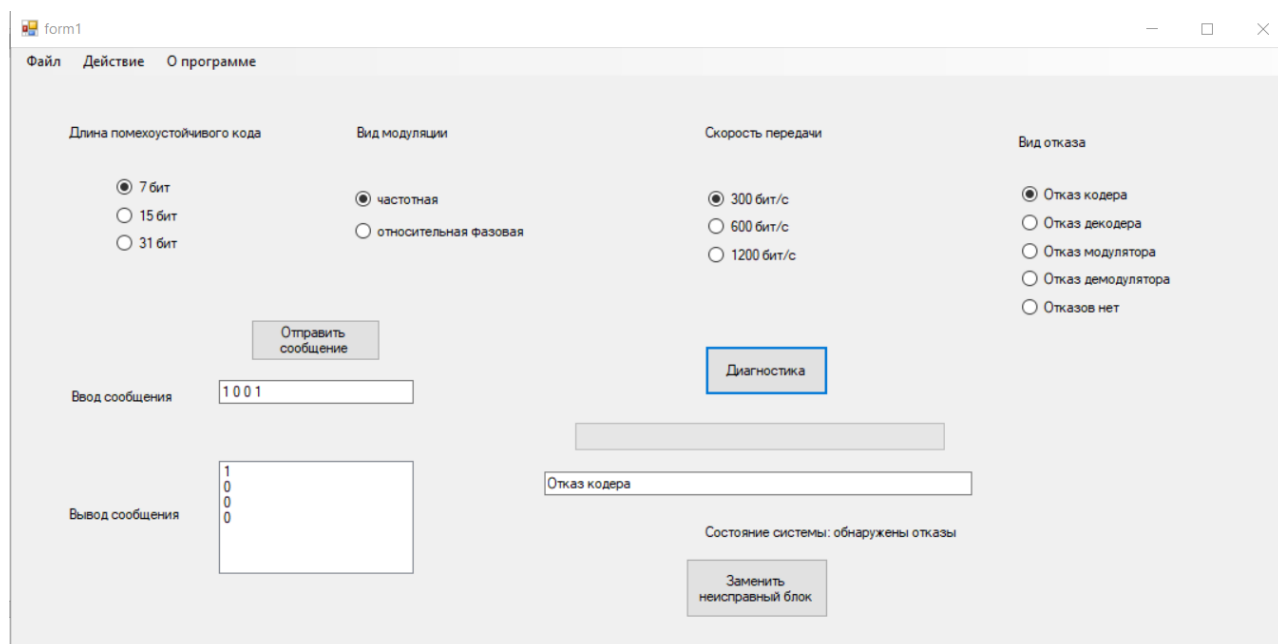


Рисунок А.8 – Результат моделирования системы при отказе кодера

Как показано на рисунке 8, выводимое сообщение отличается от вводимого, значит произошел отказ системы. Для того, чтобы узнать где именно произошел отказ, следует нажать кнопку «Диагностика». В поле вывода будет выведено сообщение «Отказ кодера» Из этого следует, что встроенная система контроля работает исправно и можно заменить неисправный блок. Результат работы представлен на рисунке А.9.

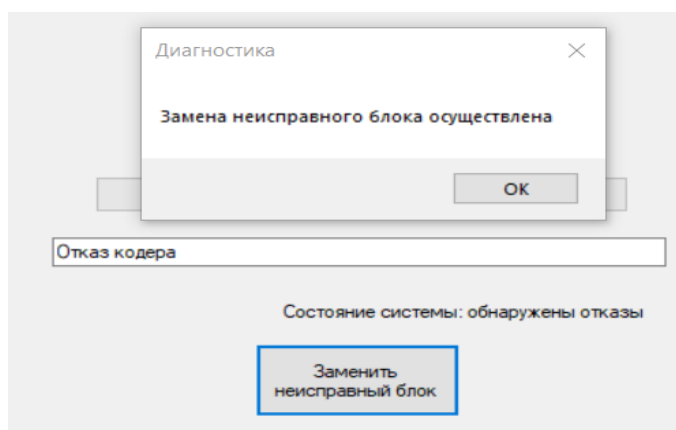


Рисунок А.9 – Замена неисправного блока

Приложение Б
(обязательное)
Код программы

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading;
using System.Threading.Tasks;
using System.Windows.Forms;
namespace WindowsFormsApplication3
{
    public partial class Form1 : Form
    {
        OpenFileDialog ofp = new OpenFileDialog();
        int dlinakoda = 0;
        int vidimodulyatsii = 1; ///  
1 - частотная модуляция 2 - относительная  
фазовая
        int skorost = 300; ///  
скорость передачи
        int vidotkaza = 0; ///  
1 - отказ кодера 2 - отказ декодера 3 - отказ  
модулятора 4 - отказ демодулятора
        int error = 0;
        int[] inf_slovo;
        int[] inf_slovo2;
        public Form1()
        {
```

```

InitializeComponent();
label1.Text = "";
button3.Visible = false;
    inf_slovo = new int[32];//информационное слово
    inf_slovo2 = new int[32];
}
private void modelir(int n, int vid_mod, int skorost, int vid_otkaza)
{
    int i, j;
    int[] tabl_sindromov = new int[32];//таблица синдромов
    int[] porozh_mn = new int[32];//порождающий многочлен
    int k = 0;//длина информационного слова
    int dlina_rg = 0;//количество проверочных разрядов, оно же длина
регистра сдвига в кодере
    int t = 0;//количество исправляемых ошибок
    int n_summ = 0; //количество сумматоров в схеме
деления=количество единиц в порождающем многочлене
    int maxi1;
    int[] koder = new int [32]; //линия задержки при делении и
умножении информационного слова на порождающий многочлен
    int[] kodovoe_slovo =new int[32];//кодовое слово после
кодирования
    int n_zn;//значение неисправного элемента
    int n_elem;//номер неисправного элемента
    int n_takt;//номер такта, когда произойдет отказ
    int schet_taktov;//счетчик тактов работы
    int cikl;//периодичность динамического контроля (в кодовых
словах)
    int porog05;//количество подряд идущих обнаруженных
неисправляемых ошибок, при котором запускается контроль

```

```

int kol_sindr = 1;//количество классов эквивалентности
double signal;
double[] skaljary = new double[4];
double f;
int dlina_posylki;
int vreme = 0;
int i1, os=0, os1=0, isp=0, oshibki=0;
int[]sx = new int[32];//регистр хранения в декодере
for (i = 1; i < 32; i++) tabl_sindromov[i] = 0;
tabl_sindromov[0] = 1;
if (n == 7) { dlina_rg = 3; k = 4; porozh_mn[0] = 1; porozh_mn[1] =
0; porozh_mn[2] = 1; porozh_mn[3] = 1; }
if (n == 15) { dlina_rg = 4; k = 11; porozh_mn[0] = 1; porozh_mn[1]
= 0; porozh_mn[2] = 1; porozh_mn[3] = 1; porozh_mn[4] = 1; }
if (n == 31) { dlina_rg = 5; k = 26; porozh_mn[0] = 1; porozh_mn[1]
= 0; porozh_mn[2] = 1; porozh_mn[3] = 1; porozh_mn[4] = 0; porozh_mn[5] = 1; }
dlina_posylki = 9600 / skorost;
//кодирование по формуле  $b(x)=a(x)*g(x)$ 
for (i = 0; i < dlina_rg; i++) koder[i] = 0;//обнулить линию задержки
кодера

//формирование кодового слова
for (j = 0; j < n; j++)
{
    if (j < k) kodovoe_slovo[j] = inf_slovo[j];
    else kodovoe_slovo[j] = 0;
    for (i = 0; i < dlina_rg; i++)
    {
        kodovoe_slovo[j] = (kodovoe_slovo[j] + koder[i] *
porozh_mn[i + 1]) % 2;

```

```

    }
    for (i = dlina_rg - 1; i > 0; i--)
    {
        koder[i] = koder[i - 1];
    }
    if (j < k) koder[0] = inf_slovo[j];
    else koder[0] = 0;
    if (vid_otkaza == 1) koder[0] = 1;
}
//конец кодирования

if (vid_mod == 1)
{
    for (i = 0; i < n; i++)
    {
        //модуляция
        if (kodovoe_slovo[i] == 0) f = 2100.0; else f = 1500.0;
        if ((vid_otkaza == 3) && (f == 2100.0)) f = f + 100.0;

        skaljary[0] = 0; skaljary[1] = 0; skaljary[2] = 0;
skaljary[3] = 0;

        for (j = 0; j < dlina_posylki; j++)
        {
            signal = Math.Sin(6.28 * f * vreme / 9600.0);
            skaljary[0] = skaljary[0] + signal * Math.Sin(6.28
* 2100.0 * vreme / 9600.0);
            skaljary[1] = skaljary[1] + signal * Math.Cos(6.28
* 2100.0 * vreme / 9600.0);
            skaljary[2] = skaljary[2] + signal * Math.Sin(6.28
* 1500.0 * vreme / 9600.0);
            skaljary[3] = skaljary[3] + signal * Math.Cos(6.28

```

```

* 1500.0 * vreme / 9600.0);

        vreme++;

    }

    if ((skaljary[0] * skaljary[0] + skaljary[1] * skaljary[1]) >
(skaljary[2] * skaljary[2] + skaljary[3] * skaljary[3])) kodovoe_slovo[i] = 0;
    else kodovoe_slovo[i] = 1;

        if (vid_otkaza == 4) kodovoe_slovo[i] = 1;

    }

} //конец
else
{

    skaljary[0] = -1.0;
    for (i = 0; i < n; i++)
    { //модуляция
        if (i == 0) { if (kodovoe_slovo[0] == 0) f = 0.0; else f =
3.14; }

        else { if (kodovoe_slovo[i] == kodovoe_slovo[i - 1]) f =
0.0; else f = 3.14; }

        skaljary[1] = 0;

        for (j = 0; j < dlina_posylki; j++)
        {

            if (vid_otkaza == 3) signal = Math.Sin(6.28 *
1900.0 * vreme / 9600.0 + f);

            else signal = Math.Sin(6.28 * 1800.0 * vreme /
9600.0 + f);

            skaljary[1] = skaljary[1] + signal * Math.Sin(6.28
* 1800.0 * vreme / 9600.0);

            vreme++;

        }
    }
}

```

```

        if (vid_otkaza == 4) skaljary[1] = 1.0;

        if (i == 0)
        {
            if (skaljary[1] < 0) kodovoe_slovo[0] = 0;
            else kodovoe_slovo[i] = 1;
            skaljary[0] = skaljary[1];
        }
        else
        {
            if (((skaljary[1] < 0) && (skaljary[0] < 0)) ||
((skaljary[1] > 0) && (skaljary[0] > 0))) kodovoe_slovo[i] = kodovoe_slovo[i - 1];
            else kodovoe_slovo[i] = (kodovoe_slovo[i - 1] +
1) % 2;
        }
    }

//декодирование
for (i = 0; i < dlina_rg; i++)
{
    koder[i] = 0;// обнуление линии задержки
}
if (vid_otkaza == 2) koder[0] = 1;

for (j = 0; j < (n * 2); j++)//вычисление синдрома
{
    os = koder[dlina_rg - 1];
    os1 = sx[n - 1];
    for (i1 = 0; i1 < kol_sindr; i1++)
    {

```

синдромов

```
isp = 1;
for (i = 0; i < dlina_rg; i++)// сравнение с таблицей
{
    if (koder[i] != tabl_sindromov[i]) { isp = 0; break;
}

}

if (isp == 1) break;
}
for (i = dlina_rg - 1; i > 0; i--)
{
    koder[i] = (koder[i - 1] + os * porozh_mn[i - dlina_rg +
2]) % 2;

}
for (i = n - 1; i > 0; i--)
{
    sx[i] = sx[i - 1];//сдвиг регистра хранения
}
if (j < n)
{
    koder[0] = (kodovoe_slovo[j] + os) % 2;

    sx[0] = kodovoe_slovo[j];
}
else
{
    koder[0] = (os + isp) % 2;
    sx[0] = (os1 + isp) % 2;
}
if (vid_otkaza == 2) koder[0] = 1;
```

```

    }
    //деление закончилось, получился синдром
    for (j = 1; j < dlina_rg; j++)
    {
        koder[0] = koder[0] + koder[j];
    }
    if (koder[0] != 0)
    {
        oshibki = oshibki + 1;
    }
    else
    {
        for (i = 0; i < dlina_rg; i++)
        {
            koder[i] = 0; //обнуление линии задержки
        }
        if (vid_otkaza == 2) koder[0] = 1;

        for (j = 0; j < n; j++) //вычисление принятого
информационного слова
        {
            os = koder[dlina_rg - 1];
            for (i = dlina_rg - 1; i > 0; i--)
            {
                koder[i] = (koder[i - 1] + os * porozh_mn[i -
dlina_rg + 2]) % 2;
            }
            koder[0] = (sx[n - 1 - j] + os) % 2;
            if (vid_otkaza == 2) koder[0] = 1;
            if (j >= dlina_rg) inf_slovo2[j - dlina_rg] = os;

```

```

        }
    }
    return; ;
}

private int kontrol(int n, int vid_mod, int skorost, int vid_otказа)
{
    int i, j;
    int[] tabl_sindromov = new int[32]; //таблица синдромов
    int[] porozh_mn = new int[32]; //порождающий многочлен

    int k = 0; //длина информационного слова
    int dlina_rg = 0; //количество проверочных разрядов, оно же длина
    регистра сдвига в кодере
    int t = 1; //количество исправляемых ошибок
    int[] koder = new int[32]; //линия задержки при делении и
    умножении информационного слова на порождающий многочлен
    int[] inf_slovo_test = new int[32]; //тестовое информационное
    слово
    int[] kodovoe_slovo = new int[32]; //кодированное слово после
    кодирования
    int[] inf_slovo2 = new int[32]; //информационное слово после
    декодирования
    int[] kodovoe_slovo_test = new int[32]; //тестовое кодированное слово
    после кодирования

    int kol_sindr = 1; //количество классов эквивалентности
    double signal;
    double[] skaljary = new double[4];
    double f;
    int dlina_posylki;
    int vreme = 0;

```

```

int i1, os = 0, os1 = 0, isp = 0;
int[] sx = new int[32]; //регистр хранения в декодере
for (i = 1; i < 32; i++) tabl_sindromov[i] = 0;
tabl_sindromov[0] = 1;
if (n == 7) { dlina_rg = 3; k = 4; porozh_mn[0] = 1; porozh_mn[1] =
0; porozh_mn[2] = 1; porozh_mn[3] = 1; }
if (n == 15) { dlina_rg = 4; k = 11; porozh_mn[0] = 1; porozh_mn[1]
= 0; porozh_mn[2] = 1; porozh_mn[3] = 1; porozh_mn[4] = 1; }
if (n == 31) { dlina_rg = 5; k = 26; porozh_mn[0] = 1; porozh_mn[1]
= 0; porozh_mn[2] = 1; porozh_mn[3] = 1; porozh_mn[4] = 0; porozh_mn[5] = 1; }
dlina_posylki = 9600 / skorost;
for (i = 0; i < k; i = i + 2) { inf_slovo_test[i] = 0; inf_slovo_test[i + 1]
= 1; } //тестовое информационное слово
//кодирование по формуле  $b(x)=a(x)*g(x)$ 
for (i = 0; i < dlina_rg; i++) koder[i] = 0; //обнулить линию задержки
кодера

```

```

//формирование кодового слова
for (j = 0; j < n; j++)
{
    if (j < k) kodovoe_slovo_test[j] = inf_slovo_test[j];
    else kodovoe_slovo_test[j] = 0;
    for (i = 0; i < dlina_rg; i++)
    {
        kodovoe_slovo_test[j] = (kodovoe_slovo_test[j] +
kodovoe_slovo_test[j] * porozh_mn[i + 1]) % 2;
    }
    for (i = dlina_rg - 1; i > 0; i--)
    {
        koder[i] = koder[i - 1];
    }
}

```

```

    }
    if (j < k) koder[0] = inf_slovo_test[j];
    else koder[0] = 0;
}
//в kodovoe_slovo находится тестовое кодовое слово
//Тест 1. Проверка кодера. На вход подается тестовое
информационное слово, то , что получилось в результате кодирования,
сравнивается в тестовым кодовым словом

//кодирование по формуле  $b(x)=a(x)*g(x)$ 
for (i = 0; i < dlina_rg; i++) koder[i] = 0;//обнулить линию задержки
кодера

//формирование кодового слова
for (j = 0; j < n; j++)
{
    if (j < k) kodovoe_slovo[j] = inf_slovo_test[j];
    else kodovoe_slovo[j] = 0;
    for (i = 0; i < dlina_rg; i++)
    {
        kodovoe_slovo[j] = (kodovoe_slovo[j] + koder[i] *
porozh_mn[i + 1]) % 2;
    }
    for (i = dlina_rg - 1; i > 0; i--)
    {
        koder[i] = koder[i - 1];
    }
    if (j < k) koder[0] = inf_slovo_test[j];
    else koder[0] = 0;
    if (vid_otkaza == 1) koder[0] = 1;
}

```

```

//конец кодирования
for (i = 0; i < n; i++)
{ if (kodovoe_slovo[i] != kodovoe_slovo_test[i]) return 1; }

//Тест 2. Проверка декодера. На вход подается тестовое кодовое
слово, то, что получилось в результате декодирования, сравнивается в тестовым
информационным словом

//декодирование
for (i = 0; i < dlina_rg; i++)
{
    koder[i] = 0; // обнуление линии задержки
}
if (vid_otkaza == 2) koder[0] = 1;

for (j = 0; j < (n * 2); j++) //вычисление синдрома
{
    os = koder[dlina_rg - 1];
    os1 = sx[n - 1];
    for (i1 = 0; i1 < kol_sindr; i1++)
    {
        isp = 1;
        for (i = 0; i < dlina_rg; i++) // сравнение с таблицей
            if (koder[i] != tabl_sindromov[i]) { isp = 0; break;
    }

    if (isp == 1) break;
}
for (i = dlina_rg - 1; i > 0; i--)
{

```

```

        koder[i] = (koder[i - 1] + os * porozh_mn[i - dlina_rg +
2]) % 2;

    }
    for (i = n - 1; i > 0; i--)
    {
        sx[i] = sx[i - 1]; //сдвиг регистра хранения
    }
    if (j < n)
    {
        koder[0] = (kodovoe_slovo_test[j] + os) % 2;

        sx[0] = kodovoe_slovo_test[j];
    }
    else
    {
        koder[0] = (os + isp) % 2;

        sx[0] = (os1 + isp) % 2;
    }
    if (vid_otkaza == 2) koder[0] = 1;
}
//деление закончилось, получился синдром

int oshibki = 0;
for (j = 1; j < dlina_rg; j++)
{
    koder[0] = koder[0] + koder[j];
}
if (koder[0] != 0)
{

```

```

        oshibki = oshibki + 1; return 2;
    }
    else
    {
        for (i = 0; i < dlina_rg; i++)
        {
            koder[i] = 0; // обнуление линии задержки
        }
        if (vid_otkaza == 2) koder[0] = 1;

        for (j = 0; j < n; j++) // вычисление принятого
информационного слова
        {
            os = koder[dlina_rg - 1];

            for (i = dlina_rg - 1; i > 0; i--)
            {
                koder[i] = (koder[i - 1] + os * porozh_mn[i -
dlina_rg + 2]) % 2;

            }
            koder[0] = (sx[n - 1 - j] + os) % 2;
            if (vid_otkaza == 2) koder[0] = 1;

            if (j >= dlina_rg) inf_slovo2[j - dlina_rg] = os;
        }
    }
    for (i = 0; i < k; i++)
    { if (inf_slovo2[i] != inf_slovo_test[i]) return 2; }
    //Тест 3. Проверка модулятора и демодулятора. На вход подается

```

тестовое кодовое слово, то , что получилось в результате демодуляции, сравнивается в тестовым кодовым словом

```
        if (vid_mod == 1)
        {
            for (i = 0; i < n; i++)
            {
                //модуляция
                if (kodovoe_slovo_test[i] == 0) f = 2100.0; else f =
1500.0;

                if ((vid_otkaza == 3) && (f == 2100.0)) f = f + 100.0;
                skaljary[0] = 0; skaljary[1] = 0; skaljary[2] = 0;
                skaljary[3] = 0;

                for (j = 0; j < dlina_posylki; j++)
                {
                    signal = Math.Sin(6.28 * f * vreme / 9600.0);
                    skaljary[0] = skaljary[0] + signal * Math.Sin(6.28
* 2100.0 * vreme / 9600.0);

                    skaljary[1] = skaljary[1] + signal * Math.Cos(6.28
* 2100.0 * vreme / 9600.0);

                    skaljary[2] = skaljary[2] + signal * Math.Sin(6.28
* 1500.0 * vreme / 9600.0);

                    skaljary[3] = skaljary[3] + signal * Math.Cos(6.28
* 1500.0 * vreme / 9600.0);

                    vreme++;
                }
                if ((skaljary[0] * skaljary[0] + skaljary[1] * skaljary[1])
> (skaljary[2] * skaljary[2] + skaljary[3] * skaljary[3])) kodovoe_slovo[i] = 0;
                else kodovoe_slovo[i] = 1;
                if (vid_otkaza == 4) kodovoe_slovo[i] = 1;
            }
        }
```

```

    }//конец
else
{
    skaljary[0] = -1.0;
    for (i = 0; i < n; i++)
    {
        //модуляция
        if (i == 0) { if (kodovoe_slovo_test[0] == 0) f = 0.0; else
f = 3.14; }
        else { if (kodovoe_slovo_test[i] ==
kodovoe_slovo_test[i - 1]) f = 0.0; else f = 3.14; }
        skaljary[1] = 0;
        for (j = 0; j < dlina_posylki; j++)
        {
            if (vid_otkaza == 3) signal = Math.Sin(6.28 *
1900.0 * vreme / 9600.0 + f);
            else signal = Math.Sin(6.28 * 1800.0 * vreme /
9600.0 + f);
            skaljary[1] = skaljary[1] + signal * Math.Sin(6.28
* 1800.0 * vreme / 9600.0);
            vreme++;
        }
        if (vid_otkaza == 4) skaljary[1] = 1.0;

        if (i == 0)
        {
            if (skaljary[1] < 0) kodovoe_slovo[0] = 0;
            else kodovoe_slovo[i] = 1;
            skaljary[0] = skaljary[1];
        }
    }
}
else

```

```

        {
            if (((skaljary[1] < 0) && (skaljary[0] < 0)) ||
((skaljary[1] > 0) && (skaljary[0] > 0))) kodovoe_slovo[i] = kodovoe_slovo[i - 1];
            else kodovoe_slovo[i] = (kodovoe_slovo[i - 1] +
1) % 2;
        }
    }
}
for (i = 0; i < n; i++)
{ if (kodovoe_slovo[i] != kodovoe_slovo_test[i]) return 3; }
return 0;
}
private void button1_Click(object sender, EventArgs e)

{
    label1.Visible = false;
    button3.Visible = false;
    timer1.Start();
}

private void помощьToolStripMenuItem_Click(object sender, EventArgs e)
{
    MessageBox.Show("Программное средство, имитирующее
возникновение,\побнаружение и устранение отказов в аппаратуре передачи
данных.", "О программе", MessageBoxButtons.OK);
}

public void button2_Click(object sender, EventArgs e)
{
    if (textBox1.Text == "")

```

```

{
    MessageBox.Show("Вы не ввели сообщение!"); return;
}
if (radioButton1.Checked == true) dlinakoda = 7;

if (radioButton2.Checked == true) dlinakoda = 15;
if (radioButton3.Checked == true) dlinakoda = 31;

if (radioButton4.Checked == true) vidimodulyatsii = 1;
if (radioButton5.Checked == true) vidimodulyatsii = 2;

if (radioButton6.Checked == true) skorost = 300;
if (radioButton7.Checked == true) skorost = 600;
if (radioButton8.Checked == true) skorost = 1200;

if (radioButton9.Checked == true) vidotkaza = 1;
if (radioButton10.Checked == true) vidotkaza = 2;
if (radioButton11.Checked == true) vidotkaza = 3;
if (radioButton12.Checked == true) vidotkaza = 4;
    if (radioButton13.Checked == true) vidotkaza = 0;
    int n;
    listBox1.Items.Clear();
    String[] ss = textBox1.Text.Split(' ');
    n = ss.Count();
    for (int i = 0; i < n; i++)
    {
        inf_slovo[i] = int.Parse(ss[i]);
    }
    modelir(dlinakoda, vidimodulyatsii, skorost, vidotkaza);
    listBox1.Items.Clear();

```

```

        for (int i = 0; i < n; i++)
        {
            listBox1.Items.Add(inf_slovo2[i]);
        }
    }

    private void button3_Click(object sender, EventArgs e)
    {
        if (label1.Text == "")
        {
            MessageBox.Show("Файл не выбран!", "Данные",
                MessageBoxButtons.OK);
            ofp.Filter = "Text files(*.txt)|*.txt|All files(*.*)|*.*";
            ofp.ShowDialog();
            label1.Text = "Файл состояние: загружен ";
            label1.Text +=
                System.IO.Path.GetFileNameWithoutExtension(ofp.FileName + ".txt");
        }
        else
        {
            MessageBox.Show("Файл отправлен!", "Данные",
                MessageBoxButtons.OK);
            label1.Text = "Файл состояние: отправлен";
            label1.Text +=
                System.IO.Path.GetFileNameWithoutExtension(ofp.FileName + ".txt");
        }
    }

    public void timer1_Tick(object sender, EventArgs e)
    {
        progressBar1.Value = progressBar1.Value + 1;
    }

```

```

if (progressBar1.Value == 300)
{
    timer1.Stop();
    Thread.Sleep(100);
    progressBar1.Value = 0;
    error = 1;

    int a = 0;
    a = kontrol(dlinakoda, vidimodulyatsii, skorost, vidotkaza);
    if (a == 0) { textBox3.Text = "Норма"; return; }
    if (a == 1) textBox3.Text = "Отказ кодера";
    if (a == 2) textBox3.Text = "Отказ декодера";
    if (a == 3) textBox3.Text = "Отказ УПС";
    label1.Text = "Состояние системы: обнаружены отказы";
    label1.Visible = true;
    button3.Visible = true;
}
}

private void button3_Click_1(object sender, EventArgs e)
{
    MessageBox.Show("Замена неисправного блока осуществлена",
"Диагностика", MessageBoxButtons.OK);
}

private void label1_Click(object sender, EventArgs e)
{
}
}
}

```