МИНИСТЕРСТВО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Санкт-Петербургский государственный университет аэрокосмического приборостроения

А. А. Вострикov, В. П. Калюжный, М. Б. Сергеев

ПЛАСТИКОВЫЕ КАРТЫ С ОТКРЫТОЙ ПАМЯТЬЮ

Учебное пособие

Санкт-Петербург
2002

УДК 681.3.04 (075)
ББК 32.973-04

В78

Востриков А. А., Калюжный В. П., Сергеев М. Б.
В78 Пластиковые карты с открытой памятью: Учебное пособие/ СПбГУАП, СПб., 2002.

Рассматриваются технологические вопросы организации смарт-карт, стандарты на их параметры. Изложены основные вопросы взаимодействия универсального контроллера смарт-карт с компьютером, в том числе при работе в сетях. Описаны протоколы I2C, RS-232, ASK-BUS, команды универсального контроллера, алгоритмы шифрации по ГОСТ 28147-89.

Учебное пособие предназначено для студентов старших курсов, обучающихся по специальностям 220101, 201600 и по направлению 552800.

Рецензенты:
Кафедра информатики и информационных таможенных технологий Российской таможенной академии им. В. Б. Бобкова;
доктор технических наук, профессор З. М. Юлдашев

Утверждено
редакционно-издательским советом университета в качестве
учебного пособия

СПИСОК ИСПОЛЬЗУЕМЫХ СОКРАЩЕНИЙ

ПУ – периферийное устройство

DTE – терминальное оборудование

DCE – связное оборудование

DMA – прямой доступ к памяти

PPI – программируемый интерфейс периферии

NMI – немаскируемое прерывание

UART – универсальный асинхронный приемопередатчик

BIOS – базовая система ввода/вывода

DOS – дисковая операционная система

DES – стандартный алгоритм шифрования данных (США)

ASCII – американский стандартный код обмена информацией

POST – тест, выполняемый после включения питания

COM port – COMunication port, последовательный порт

ISO – международная организация по стандартам (МОС)

LAN – локальная сеть

EIA – ассоциация электронной промышленности

FIFO – First-In, First-Out. Первый вошел, первый вышел

EEPROM – энергонезависимая память, допускающая возможность оперативной записи и считывания информации.

1. ОБЩИЕ СВЕДЕНИЯ И КЛАССИФИКАЦИЯ ПЛАСТИКОВЫХ КАРТ

Понятие «пластиковая карта с интегрированным чипом» или «смарт-карта» («smart» от английского – умный, смекалистый) плавно вошли в современную жизнь как олицетворение новейших технологий, базирующихся на их основе. Первая пластиковая карта с интегрированным в нее чипом была создана в 1974 г. независимым французским изобретателем Роландом Морено (Roland Moreno). Такие карточки пришли на смену карточкам с магнитной полосой, что и определило довольно тернистый путь «умных» собратьев «магниток». Прошло много лет прежде, чем это, без сомнения технически более совершенное устройство, заняло достойное место в роду пластиковых карт.

1.1. Сферы использования пластиковых карт

Появившись, как альтернатива карточкам с магнитной полосой, сматр-карты вытесняют последних из традиционных сфер их использования, которые можно разделить на следующие группы.

Первая группа объединяет в себе финансовые приложения: хранение денег или эквивалента денег, электронная коммерция, ведения банковских счетов и баланса предприятия, другие банковские операции. Главное достоинство карточек, входящих в эту группу, – способность противостоять попыткам взлома или кражи. Идентификация владельца производится при помощи персонального идентификационного номера (Personal identification number) – PIN-кода.

Вторая – приложения, удостоверяющие права пользователя в момент обращения к ресурсам общего пользования (и поэтому не столь привлекательные для взлома), не требующие уникальной идентификации и, возможно не столь надёжные. Это обеспечение безопасной работы в сети учебных учреждений, интеллектуальные дисконтные карты на товары повседневного проса (любой из членов семьи, организации или её подразделения в праве воспользоваться дисконтной картой), библиотечные, гостиничные, стояночные и клубные карточки, карточки сотовых телефонов и подобных назначений.

Третья группа – приложения, для которых абсолютно критична индивидуальность владельца смарт-карты, но хранимая информация вообще не носит финансового характера. Это контроль прав доступа к определённым ресурсам (доступ на стратегические объекты, в системы расчетов банк-клиент и др.), предотвращение не санкционированных действий и др.

Следует отметит что смарт-технология находит и другие области применения, такие как замена ключей, железнодорожных и авиационных билетов, медицинских карт, студенческих удостоверений. Разрабатываемые уже

сегодня электронные паспорт и медицинская карта – аналоги бумажных «паспорта» и «истории болезни» в обычном понимании. Объем информации которая может быть размещена на смарт карте в настоящее время составляет десятки килобайт, что вполне достаточно для всех перечисленных применений.

Принципиально новые возможности открывает объявленная фирмой Sun инициатива по поддержке смарт-карт Web-браузерами, в первую очередь браузером HotJava. Помимо идентификации личности самого пользователя, избавляющий его от обременительной процедуры введения/запоминания пароля, такой браузер может проверять цифровые подписи и контролировать полномочия и действия всех загружаемых им приложений из среды Internet.

На выставке Comdex2000 компания Humetrix представила особую смарт-карту доступа Netissimo, которая, работая с устройством, имеющим выход в интернет (мобильный телефон, портативный компьютер и др.), обеспечивает автоматическое соединение и загружает какой-либо из запрограммированных сайтов. При этом используются записанные в не настройки, а не установки провайдера. В памяти Netissimo содержаться URL-адреса, логины и пароли доступа к заданным страницам. Таким образом смарт-карта является электронным пропуском для посещения определенных сайтов, осуществляющей загрузку без ее владельца. Приведенные примеры использования смарт-технологии, учитывая нарастающий бум мобильного Internet'а, очень актуальны и современны.

Перечисленные и многочисленные другие применения не вызывают лавинообразного роста числа узкоспециализированных разновидностей смарт-карт, а удовлетворяются существующими их видами и обслуживаются одними и теми же устройствами. Согласно прогнозам в течение ближайших 5 лет в мире будет изготавливаться и использоваться до1000 000 000 пластиковых карт в год, хотя такой объем производства был зафиксирован уже в 1998 г. Для сравнения, количество находящихся в обращении кредитных карт достигает 900 млн. При этом 95% общего количества эмиссии смарт-карт приходится на развитые страны Европы и Северной Америки, Южную Африку и Юго-Восточную Азию. Последняя известна как регион наиболее взрывного роста популярности сарт-карт. По ряду причин использование сарт-карт в России более предпочтительно, чем магнитных карточек. Основной причиной можно назвать неразвитость и низкое качество электронных коммуникаций, так необходимых для авторизации и проведения транзакций с использованием карточек с магнитной полосой.

В начале третьего тысячелетия, по мнению экспертов, в мире единовременно в обращении будет находиться около 3 млрд смарт-карт. По использованию будут доминировать различные финансовые сервисы, а так же обеспечение безопасности в приложениях для сотовых телефонов, компьютерных игр и телевидения, хотя возможно и коренное изменение этих пропорций. Эксперты предсказывают, например, неминуемую интеграцию технологии смарт-карт с новейшими изобретениями в области идентификации личности на основе распознавания уникальных биологических параметров человеческого организма.

Мировой опыт использования смарт-технологий указывает на то, что наиболее впечатляющие результаты от расширения сфер их использования ожидаются не от усложнения элементной базы смарт-карт, а от развития ее интеллектуальных качеств.

1.2. Характеристики смарт-карт

Сарт-карты, в противовес магнитным карточкам, имеют ряд существенных преимуществ.

Во-первых, стойкость к внешним воздействиям. Смарт-карту можно окунать в воду, держать в кармане, ронять, класть на постоянный магнит и т.д. кроме того, стандарт на них специфицирует защиту от воздействия статического электричества. Единственное механическое движение, которое происходит при помещении карты в устройство считывания (считыватель, смарт-ридер) – это собственно помещение, а для бесконтактных смарт-карт нет даже и этого.

Во-вторых, безопасность и защищенность информации. В отличии от магнитных карт, которые можно переделать путём копирования на оборудовании стоимостью менее 1000 долларов США, процесс создания смарт-карт под силу только крупной промышленной компании. На этапе производства чипов в них вносятся уникальные коды, позволяющие в дальнейшем производить действия с ними. Партия смарт-карт и партия их кодов доставляются потребителю разными путями, что гарантирует утечку «болванок» и использование их для подделок. На этапе непосредственного использования смарт-карт действия становятся возможным только при вводе PIN-кода владельца. В противном случае карта «молчит». Против попыток подбора PIN-кода (более оговоренного заранее числа раз) используется система блокировки карточки. Кроме того, современна смарт-карта содержит в себе специальные программно-аппаратные средства криптографической защиты всей содержащейся информации.

В-третьих, удобство работы. Смарт-карта содержит всю необходимую информацию о ее владельце (группе владельцев), количестве средств его правах и др., которыми он (они) могут пользоваться. Работа со смарт-картой не требует режима on-line.

Есть и другие преимущества смарт-карт – это стоимость и гибкость. Когда считают общую стоимость системы, использующей пластиковые карты, принимают во внимание стоимость изобретения, издания, замены и обновления самих карт и ридеров (терминалов), их надежность и долговечность, необходимость использования телекоммуникационного оборудования. С учетом сказанного смарт-карта предлагает самое экономичное решение. Смарт-карта является и самым гибким звеном в цепи дальнейших расширений системы, использующей широкий диапазон электрически и физически совместимых смарт-карт которые могут выполнять самые различные функции: от простого идентификатора личности, до носителя файлов с высоким уровнем важности и секретности информации. Выбор смарт-карты это часть

большой политики достижения высокого уровня и эффективности приложения с одновременным снижением его себестоимости.

Смарт-технология бурно развивается и в обозримом будущем можно ожидать того, что смарт-карты лишаться таких характерных особенностей – наличие только чипа и контактного способа работы с ним. Уже происходит интенсивное вытеснение контактных карт их бесконтактными вариантами, в том числе и имеющими собственный источник питания, жидкокристаллический дисплей, встроенную антенну для бесконтактного ввода информации и клавиатуру (так называемые суперсмарт-карты). Одна из первых промышленных реализаций этой технологии, носящая название Micro 680,была продемонстрирована компанией Gemplus на выставках Smart Card ’98, прошедшей в апреле в Лондоне, и на CeBIT’98 в Ганновере в марте, значительно более широкий спектр был представлен на выставке JavaOne в Сан-Франциско в марте 1998г.

1.3. Обзор смарт-карт и перспективы развития смарт-технологии

Важнейшее преимущество смарт-карт перед обыкновенными пластиковыми картами с магнитной полосой состоит в том, что смарт-карта все необходимое для выполнения операций может производить самостоятельно, в то время как для пассивной магнитной карты требуется обслуживание со стороны удаленного компьютера-сервера (как правило, доступ к базам данных для проведения транзакции) при участии торгового терминала (POS), банкомата (ATM) или другого оборудования в режиме прямого доступа. Сеанс непрерывной связи, который необходим при работе пластиковой магнитной карты, делает ее нерентабельной в станах с высокими тарифами на сетевой трафик. В Европе к таким государствам относится, например, Франция. Именно в этой стране смарт-карты впервые были применены на практике. В России интерес к использованию смарт-карт обусловлен несколькими важными для нашей страны причинами:

во-первых, это плохое качество телефонных сетей, делающих режим on-line не надежным;

во-вторых – высокий криминальный менталитет среднестатистического российского пользователя таких карточек;

в-третьих, использование технологий магнитных карт пришло на Российский рынок с опозданием на много лет, в период расцвета смарт-технологии, что способствовало ее выбору как более новой и надежной.

Наиболее ярко выражен интерес к смарт-картам на азиатском рынке, участники которого до сих пор отдают предпочтение наличности перед использованием платежей в счет кредитов (что характерно для большинства стран запада, исключая, как ни странно, Швейцарию). Сегодня 4/5 всех платежей в Гонконге осуществляется через наличность. Но к 2000 г. азиатский регион обещает обеспечить приблизительно треть всего обращения смарт-карт.

Обычная смарт-карта – это пластиковая карта с расположенными на одной из ее сторон контактами. Размер, расположение и значение контактов

смарт-карт, физические характеристики пластмассы, типы температурных допусков, а так же другие требования определяет международный стандарт ISO-7618. Как правило, все производители смарт-карт гарантируют 100 000 циклов перезаписи и 10 лет хранения последней записи. Стоимость смарт-карточки у различных производителей колеблется от $1.5 до $12. Стоимость услуг, представляемых эмитентами таких карточек, определяется их назначением и спекртом предоставляемых услуг.

Имеется несколько типов пластмасс, используемых для смарт-карт. Основные типы – это PVC и ABS. Карточка из PVC может быть эмбоссирована (может содержать надписи и имена выпуклыми словами), но не может быть утилизирована и переработана. Карточка из ABS не может быть эмбоссирована, но может быть переработана.

Не смотря на чрезвычайную широту спектра типов и подтипов существующих смарт-карт, приведем классификацию этого вида пластиковых карт. Она представлена на рис.1.1.

Контактные смарт-карты с открытой памятью реально представляют собой чипы электрически перепрограммируемой энергонезависимой памяти (EEPROM или ЭППЗУ). Чаще всего для обмена информацией с такой картой (т.е. сохранение информации и, затем, ее чтение) используется протокол I^2C фирмы Philips. Это протокол последовательной передачи данных между двумя устройствами – “Master” (хозяин) и “Slave” (раб). Применение последовательного протокола передачи данных обусловлено ограниченным числом контактов.

Многие фирмы-изготовители чипов предлагают готовые кристаллы EEPROM для размещения на смарт-картах. Фирме-разработчику таких карт остается только разместить чип на ее поверхности и закрыть некоторым подобием мирокорпуса с металлическими контактами на поверхности.

Карты с защищенной памятью отличается от описанных выше тем, что доступ к ячейкам памяти возможен только при вводе (передаче) заданной кодовой последовательность (пароля).

Наибольший интерес, естественно, представляют смарт-карты, обладающие собственным процессором, способным производить вычисления, криптографическое кодирование/декодирование данных, идентификацию, а так же умеющим оптимально использовать память. По физическим параметрам, а это до 16Кбайт ПЗУ, до 512байт ОЗУ и 8-ми разрядный процессор, такие карточки близки исходному варианту персонального компьютера IBM XT, хотя значительно уступают ему по объему памяти. Ввиду таких вычислительных мощностей смарт-карты с микропроцессором имеют встроенную микрооперационную систему, управляющую многими процессами внутри карты, а так же обменом информацией со считывателем.

Микропроцессор на смарт-карте – это, как правило, восьмиразрядный процессор с RISC-архитектурой. Однако уже в 1998 г. было объявлено о разработке и выпуске для смарт-карт шестнадцати- и даже тридцатидвухразрядных микропроцессоров (в том числе с процессором, выполняющим только криптографическое преобразование данных).

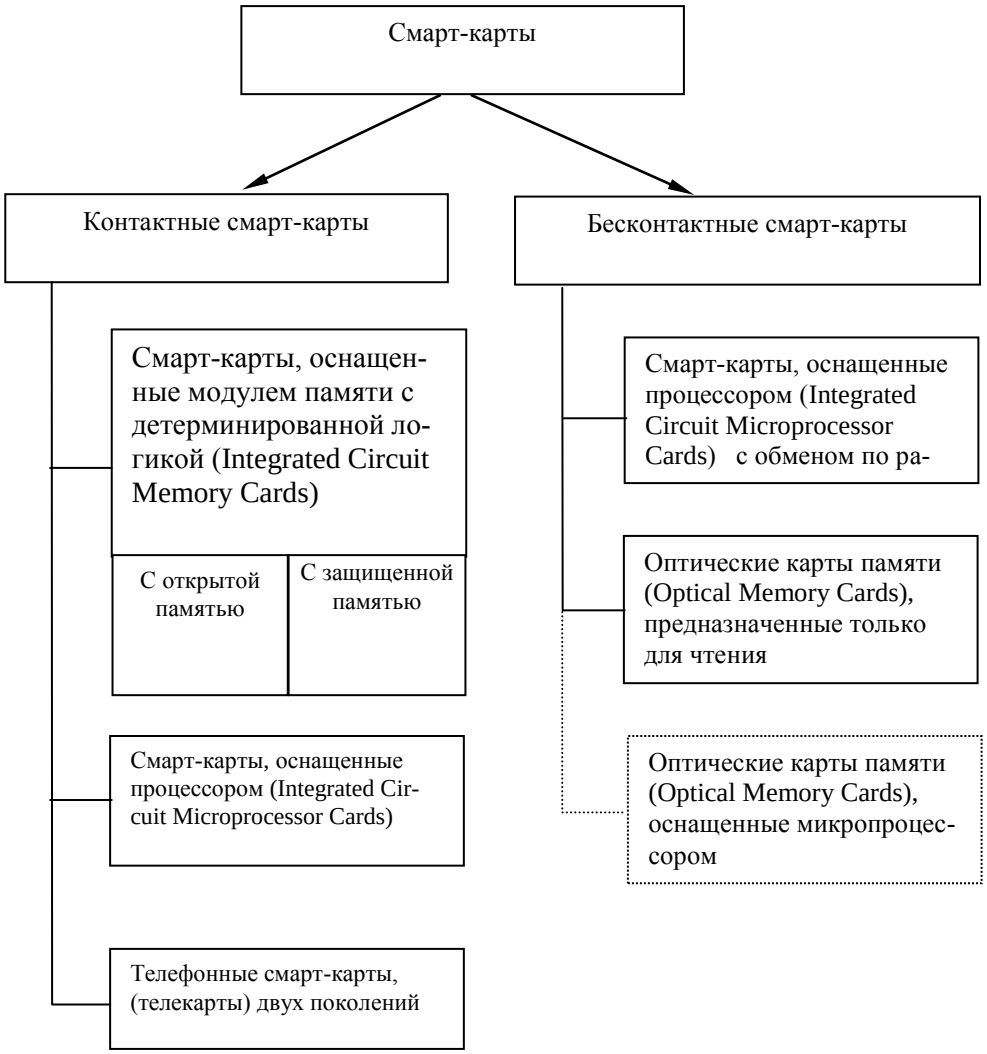


Рис.1.1. Классификация смарт-карт

Программное обеспечение для микропроцессора на смарт-карте при разработке проходит те же этапы, что и при разработке программного обеспечения для микропроцессорных встроенных систем управления. Одной из ведущих операционных систем на рынке смарт-карт является операционная

система MULTOS компании Mondex, которая позволяет осуществлять быструю разработку конечных приложений и использует язык C. Однако существенным недостатком MULTOS является то, что она не опирается на открытый стандарт, из-за чего в последнее время неоднократно подвергалась критике. В частности, Mondex пока не опубликовала технические детали протокола, используемого смарт-картой и интерпретирующим устройством (банкоматом). Большие надежды на то, чтобы стать настоящим взрывом для приложений разработки программного обеспечения для смарт-карт, дарит проходящая сейчас этапы стандартизации технология Java Card.

Телекарты (телефонные смарт-карты) в приведенной классификации выделены в отдельный тип не случайно. В силу специфики работы этих смарт-карт и очень узкой их специализации они получили существенные отличия от своих ближайших родственников во внутренней организации. Телекарты так же располагают некоторым объемом памяти, лишь пять байт из которой участвуют непосредственно в процессе фиксации количества оставшихся на карте единиц. Для этих карт был специально разработан протокол обмена, позволяющий только уменьшать кредит и читать все содержимое памяти карты.

Бесконтактные смарт-карты внешне выглядят как и простые кредитные карты. Однако отличие состоит в ее «начинке» – внутри кроме чипа расположена встроенная антенна. Для того чтобы заставит работать такую карту, достаточно расположить ее в непосредственной близости от антенны считывателя. Обмен информацией между считывателем и картой ведется по радиоканалу без физического контакта. Бесконтактные карты идеальны для приложений, где считывание должно производиться очень быстро, например, в массово посещаемых местах.

Идентификация карты ведется как выбором по частоте, так и с помощью заданных кодовых последовательностей. Максимальное расстояние между считывателем и картой колеблется от 10 см до нескольких метров.

Оптические карты памяти выглядят как магнитные карточки с миниатюрным CD-диском, прикрепленным сверху. Внешнее сходство не является обманчивым, так как именно при помощи CD-технологии оптические карты способны хранить до 4Мбайт данных. Однако информация может быть записана на поверхность карты только один раз и не подлежит стиранию или перезаписи. Такие устройства идеально подходят для замены медицинских карт, водительских удостоверений, паспортов, аттестатов и др. Сегодня оптические смарт-карты сопоставимы по цене со смарт-картами других типов, однако требуют существенно более дорогостоящих считывающих устройств, а это значит, что разработка стандарта на такие устройства еще далека от завершения. В будущем смарт-карты с оптическим носителем предполагается оснастить собственным процессором.

Очевидно, что во всем многообразии пластиковых карт карточки с открытой памятью представляют не самую большую группу и имеют не самое широкое применение, однако без детального их изучения понимание работы других видов смарт-карт существенно затруднено.

2. ПОСЛЕДОВАТЕЛЬНЫЙ ИНТЕРФЕЙС PC

Основное назначение периферийного устройства (ПУ) – обеспечить поступление в ЭВМ из окружающей среды программ и данных для обработки, а также выдачу результатов работы ЭВМ в виде, пригодном для восприятия человека или для передачи на другую ЭВМ, или в иной, необходимой форме. ПУ в немалой степени определяют возможности применения ЭВМ.

ПУ ЭВМ включают в себя внешние запоминающие устройства, предназначенные для сохранения и дальнейшего использования информации, устройства ввода-вывода, предназначенные для обмена информацией между оперативной памятью машины и носителями информации, либо другими ЭВМ, либо оператором. Входными устройствами могут быть: клавиатура, дисковая система, манипуляторы типа "мышь", модемы, микрофон; выходными – дисплей, принтер, дисковая система, модемы, звуковые системы, другие устройства. С большинством этих устройств обмен данными происходит в цифровом формате. Для работы с разнообразными датчиками, предоставляющими информацию в аналоговой форме, а также некоторыми исполнительными устройствами используются аналого-цифровые и цифро-аналоговые преобразователи для преобразования цифровых данных в аналоговые и наоборот.

Цифровой интерфейс проще по сравнению с цифро-аналоговым. Однако для него также требуются специальные схемы, преобразующие при передаче исходный информационный алфавит (т.е. полный набор информационных символов, которые встречаются в блоках передаваемой информации) в алфавит сигналов, использующийся для передачи по физическому каналу, а при приеме, наоборот, – из сигнального в информационный. Необходимость такого преобразования объясняется непригодностью форматов, в которых информация храниться, к передаче по реальным физическим каналам, поскольку последние обладают различными характеристиками, уменьшающими вероятность правильного выделения на приемной стороне содержимого переданной информации. Кроме того, не всегда возможно использовать в качестве линии передачи наборы проводников, количество которых равно разрядности передаваемых информационных символов.

Для передачи по физическому каналу различают последовательную и параллельную передачу данных, с синхронизацией (т.е. с использованием специальных синхронизирующих импульсов, обеспечивающих своевременную фиксацию передаваемой информации на приемной стороне) и без нее. Как правило, для передачи данных на небольшие расстояния при сравнительно высоких требованиях на скорость передачи используют синхронные параллельные методы. При передаче на большие расстояния чаще применяют последовательные интерфейсы. Причем, с увеличением длины физического канала, обычно, уменьшается скорость передачи, и возрастают требования к техническим характеристикам самого канала и аппаратуре приемо-передатчиков.

2.1.Стандарт RS-232C

Один из наиболее распространенных стандартов последовательной асинхронной передачи данных – RS-232C (**R**eference **S**tandard № **232** Revision **C**). Данный интерфейс, определенный стандартом *Ассоциации электронной промышленности (EIA)*, подразумевает наличие оборудования двух видов: *терминального DTE* и *связного DCE*. Чтобы не составить неправильного представления об интерфейсе RS–232C, необходимо отчетливо понимать различие между этими видами оборудования. *Терминальное оборудование*, например микрокомпьютер, может посылать и (или) принимать данные по последовательному интерфейсу. Оно как бы оканчивает (terminate) последовательную линию. *Связное оборудование* – устройства, которые могут упростить передачу данных совместно с терминальным оборудованием. Наглядным примером связного оборудования служит модем (модулятор-демодулятор). Он оказывается соединительным звеном в последовательной цепочке между компьютером и телефонной линией.

Различие между терминальными и связными устройствами довольно расплывчато, поэтому возникают некоторые сложности в понимании того, к какому типу оборудования относится то или иное устройство. Рассмотрим ситуацию с принтером. К какому оборудованию его отнести? Как связать два компьютера, когда они оба действуют как терминальное оборудование. Для ответа на эти вопросы следует рассмотреть физическое соединение устройств. Произведя незначительные изменения в линиях интерфейса RS–232C, можно заставить связное оборудование функционировать как терминальное. Чтобы разобраться в том, как это сделать, нужно проанализировать функции сигналов интерфейса RS–232C, приведенные в табл.2.1.

Линии 11, 18, 25 обычно считают незаземленными. Приведенная в таблице спецификация относится к спецификациям Bell 113B и 208A. Линии 9 и 10 используются для контроля отрицательного *(MARK)* и положительного *(SPACE)* уровней напряжения. Во избежание путаницы между *RD* (*Read* – считывать) и *RD* (*Received Data* – принимаемые данные) будут использоваться обозначения *RXD* и *TXD*, а не *RD* и *TD*.Стандартный последовательный порт RS–232C имеет форму 25–контактного разъема типа *D* (рис.2.1).

На практике вспомогательный канал RS–232C применяется редко, и в асинхронном режиме вместо 25 линий используются 9 линий (табл.2.2). Терминальное оборудование обычно оснащено разъемом со штырьками, а связное – разъемом с отверстиями (но могут быть и исключения).

Сигналы интерфейса RS–232C подразделяются на следующие классы.

Последовательные данные (например, *TXD, RXD*). Интерфейс RS–232C обеспечивает два независимых последовательных канала данных: первичный (главный) и вторичный (вспомогательный). Оба канала могут работать в дуплексном режиме, т.е. одновременно осуществляют передачу и прием информации.

1.1.1 Таблица 2.1

Функции сигнальных линий интерфейса RS–232C

Номер контакта	Сокращение	Направление	Полное название
1	*FG*	—	Основная или защитная земля
2	*TD (TXD)*	К DCE	Передаваемые данные
3	*RD (RXD)*	К DTE	Принимаемые данные
4	*RTS*	К DCE	Запрос передачи
5	*CTS*	К DTE	Сброс передачи
6	*DSR*	К DTE	Готовность модема
7	*SG*	—	Сигнальная земля
8	*DCD*	К DTE	Обнаружение несущей
9	—	К DTE	(Положительное контрольное напряжение)
10	—	К DTE	(Отрицательное контрольное напряжение)
11	*QM*	К DTE	Режим выравнивания
12	*SDCD*	К DTE	Обнаружение несущей вторичных данных
13	*SCTS*	К DTE	Вторичный сброс передачи
14	*STD*	К DCE	Вторичные передаваемые данные
15	*TC*	К DTE	Синхронизация передатчика
16	*SRD*	К DTE	Вторичные принимаемые данные
17	*RC*	К DTE	Синхронизация приемника
18	*DCR*	К DCE	Разделенная синхронизация приемника
19	*SRTS*	К DCE	Вторичный запрос передачи
20	*DTR*	К DCE	Готовность терминала
21	*SQ*	К DTE	Качество сигнала
22	*RI*	К DTE	Индикатор звонка
23	—	К DCE	(Селектор скорости данных)
24	*TC*	К DCE	Внешняя синхронизация передатчика
25	—	К DCE	(Занятость)

Управляющие сигналы квитирования (например, *RTS, CTS*). Сигналы квитирования – средство, с помощью которого обмен сигналами позволяет *DTE* начать диалог с *DCE* до фактической передачи или приема данных по последовательной линии связи.

Рис.2.1. Назначение линий 25–контактного разъема типа *D* для интерфейса RS–232C

Сигналы синхронизации (например, *TC, RC*). В синхронном режиме (в отличие от более распространенного асинхронного) между устройствами необходимо передавать сигналы синхронизации, которые упрощают синхронизм принимаемого сигнала в целях его декодирования.

Таблица 2.2

Основные линии интерфейса RS–232C

Номер контакта	Сигнал	Выполняемая функция
1	*FG*	Подключение земли к стойке или шасси оборудования
2	*TXD*	Последовательные данные, передаваемые от *DTE* к *DCE*
3	*RXD*	Последовательные данные, принимаемые *DTE* от *DCE*
4	*RTS*	Требование *DTE* послать данные к *DCE*
5	*CTS*	Готовность *DCE* принимать данные от *DTE*
6	*DSR*	Сообщение *DCE* о том, что связь установлена
7	*SG*	Возвратный тракт общего сигнала (земли)
8	*DCD*	*DTE* работает и *DCE* может подключится к каналу связи

2.2.Виды сигналов

В большинстве схем, содержащих интерфейс RS–232C, данные передаются асинхронно, т.е. в виде последовательности пакета данных. Каждый пакет содержит один информационный символ (например, код ASCII), причем информация в пакете достаточна для его декодирования без отдельного сигнала синхронизации.

RS-232C предполагает передачу информации двумя уровнями напряжения "без возврата к нулю". При этом перед началом информационной последовательности (количество информационных бит от 5 до 8) следует так называемый "старт-бит" определенной длительности (высокий уровень), а по окончании – "стоп-бит" одинарной, полуторной или двойной длины (низкий уровень).

Кроме того, перед стоп-битом иногда добавляется так называемые биты "четности" или "нечетности", которые позволяют обнаруживать нечетные количества ошибок в информационной части пакета данных. Следует отметить, что поскольку передача асинхронная, то очень большое значение для безошибочной передачи играет правильная настройка по скорости приемной и передающей аппаратуры.

Для примера рассмотрим формат передачи по RS-232C одного из кодов символов ASCII. Символы кода ASCII представляются семью битами, например буква **А** имеет код 1000001. Чтобы передать букву **А** по интерфейсу RS–232C, необходимо ввести дополнительные биты, обозначающие начало и конец пакета. Кроме того, как указывалось выше, желательно добавить лишний бит для простого контроля ошибок по паритету (четности).

Наиболее широко распространен формат, включающий в себя один стартовый бит, один бит паритета и два стоповых бита. Начало пакета данных всегда отмечает низкий уровень стартового бита. После него следует 7 бит данных символа кода ASCII. Бит четности содержит 1 или 0 так, чтобы общее число единиц в 8–битной группе было нечетным. Последним передаются два стоповых бита, представленных высоким уровнем напряжения. Эквивалентный ТТЛ–сигнал при передаче буквы **А** показан на рис.2.2.

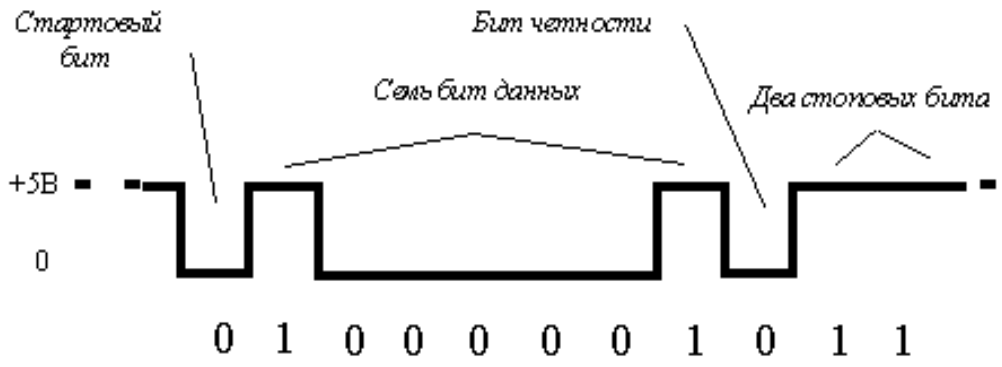


Рис.2.2. Представление кода буквы А сигнальными уровнями ТТЛ

Таким образом, полное асинхронно передаваемое слово состоит из 11 бит (фактически данные содержат только 7 бит) и записывается в виде 01000001011.

Используемые в интерфейсе RS–232C уровни сигналов отличаются от уровней сигналов, действующих в компьютере. Логический 0 *(SPACE)* представляется положительным напряжением в диапазоне от +3 до +25 В, логическая 1 *(MARK)* — отрицательным напряжением в диапазоне от –3 до –25 В (за счет такой большой разности потенциалов между уровнями нуля и единицы данный стандарт достаточно устойчив к внешним помехам.). На рис.2.3 показан сигнал в том виде, в каком он существует на линиях *TXD* и *RXD* интерфейса RS–232C.

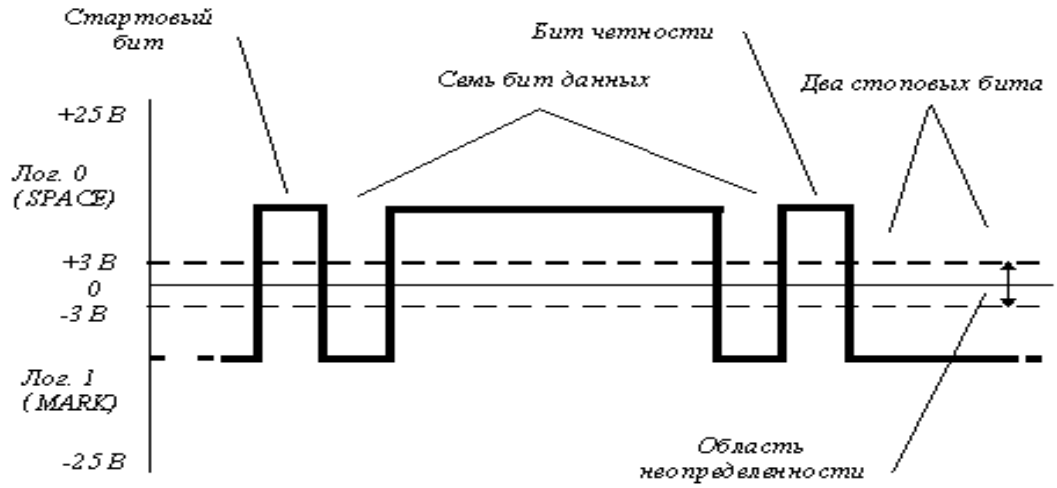


Рис.2.3. Вид кода буквы А на сигнальных линиях *TXD* и *RXD*

Сдвиг уровня, т.е. преобразование ТТЛ уровней в уровни интерфейса RS–232C и наоборот производится специальными микросхемами *драйвера линии и приемника линии*.

На рис.2.4 представлен типичный микрокомпьютерный интерфейс RS–232C. Программируемая микросхема *DD1* последовательного ввода осуществляет параллельно–последовательные и последовательно–параллельные преобразования данных.

Микросхемы *DD2* и *DD3* производят сдвиг уровней для трех выходных сигналов *TXD, RTS, DTR*, а микросхема *DD4* – для трех входных сигналов *RXD, CTS, DSR*. Микросхемы *DD2* и *DD3* требуют напряжения питания ±12 В, хотя сейчас выпускаются чипы преобразователей уровня, не требующие дополнительного источника (только +5В), а имеющие встроенные повышающие преобразователи постоянного напряжения (например, MAX232 (фирма MAXIM) или ADM232 (фирма Analog Devices)).

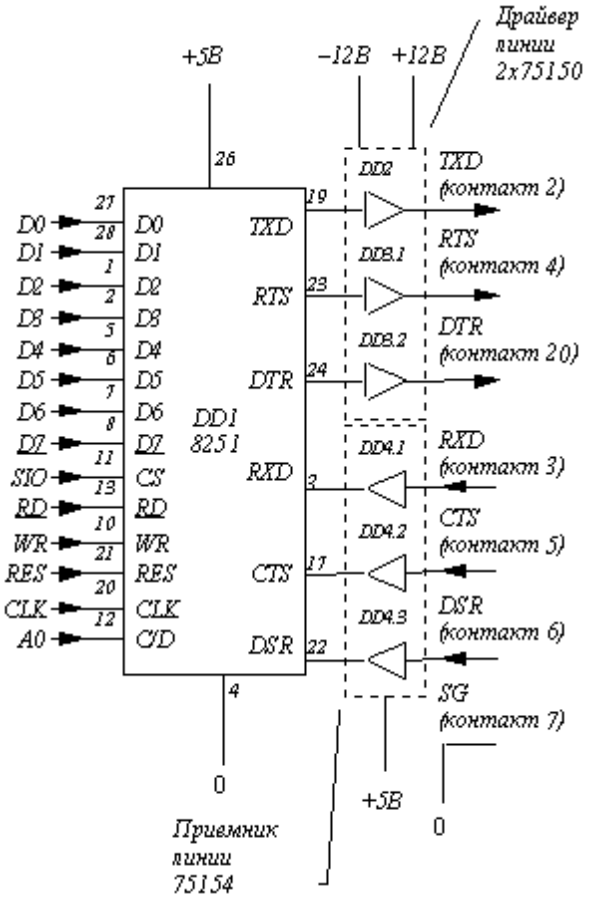


Рис.2.4. Типичная схема интерфейса RS–232C

2.3.Усовершенствования стандарта

Разработано несколько новых стандартов, направленных на устранение недостатков первоначальных спецификаций интерфейса RS–232C. Среди них можно отметить интерфейс RS–422 (балансная система, допускающая импеданс линии до 50 Ом), RS–423 (небалансная система с минимальным импедансом линии 450 Ом) и RS–449 (стандарт с высокой скоростью передачи данных, в котором несколько изменены функции схем и применяется 37–контактный разъем типа *D*).

Обмен информацией с периферийными устройствами у IBM PC – совместимых компьютеров осуществляется через регистры ввода/вывода (порты). Таким образом, архитектура данных ПК предполагает раздельный обмен с оперативной памятью и портами ввода/вывода, так как они имеют различные адресные пространства. Ввиду этого в системе команд процессора введены две команды записи в порт (OUT) и команда чтения из порта (IN).

Через порты передается и получается информация об устройствах – клавиатуре, видеоконтроллере, дисковых контроллерах и т.д. Для одного устройства можно иметь несколько портов различного назначения.

Каждый порт идентифицируется 16-битным адресом от 0 до 65535. Адрес порта задается как операнд инструкций IN и OUT. Как при обращении к памяти, так и при обращении к портам процессор использует шину адреса и шину данных. Чтобы разграничить эти два вида сообщений, процессор посылает сигнал на шину управления. Он информирует все устройства ввода/вывода, что адрес, установленный на шине адреса – это адрес порта. Устройство, имеющее порт с таким адресом, открывает его для заявки процессора. В табл.2.3 описаны основные порты IBM PC совместимого компьютера.

Таблица 2.3

Адреса портов PC

Адреса	Назначение
000-00F	Контроллер I8237A-5 прямого доступа к памяти (DMA)
020-021	Контроллер прерывания I8259A
040-043	Часы I8253-5
060-063	Программируемый интерфейс периферии (PPI)
080-083	DMA контроллер
0A0	Для маскирования извне немаскируемого прерывания (NMI)
0C0-0CF	Управление кириллицей (сейчас не используется)
200-20F	Джойстик
278-27F	Параллельный интерфейс номер 2 (номер 3, если установлен монохроматический адаптер с параллельным интерфейсом)
2F8-2FF	Последовательный интерфейс RS-232 номер 2
320-32F	Жесткий диск
378-37F	Параллельный интерфейс номер 1 (номер 2, если установлен монохроматический адаптер с параллельным интерфейсом)
3B0-3BF	Монохроматический адаптер (если он с параллельным интерфейсом, для него порты 3BC, 3BD, 3BE)
3D0-3DF	Цветной графический адаптер
3F0-3F7	Дискетное устройство
3F8-3FF	Последовательный интерфейс RS-232 номер 1

Устройства со стандартным последовательным интерфейсом RS-232C подключаются к ПК через адаптеры последовательного интерфейса (далее для краткости – последовательные адаптеры). Такой адаптер может представлять собой самостоятельный модуль или часть модуля, который выполняет и другие функции. У современных ПК последовательный адаптер расположен на материнской плате и входит в состав ее логики. Широкое применение и относительная сложность последовательного интерфейса привели к созданию микросхем специального типа, которые осуществляют формирова-

ние и перенос данных и облегчают управление. Схемы этого типа называются UART (Universal Asynchronous Receiver Transmitter).

Операционная система DOS может управлять двумя последовательными адаптерами с именами COM1 и COM2. Драйвер последовательного интерфейса BIOS может управлять четырьмя последовательными адаптерами с номерами 0 (COM1), 1 (COM2), 2 и 3.

Управление одним последовательным адаптером осуществляется через семь портов ввода-вывода с последовательными адресами. Адрес первого порта называется *базовым адресом адаптера.* Для последовательных адаптеров резервированы четыре группы по восемь адресов портов, приведенных в табл.2.4. Для управления адаптером используются первые семь портов в группе. Некоторые адаптеры можно настроить для любой из четырех групп адресов, а другие - только для первых двух групп.

Таблица 2.4

Группы портов ввода-вывода для последовательных адаптеров

Группа	Адреса портов	Группа	Адреса портов
1	3F8H - 3FEH	3	3E8H - 3EFH
2	2F8H - 2FEH	4	2E8H - 2EFH

Во время POST (Power On Self Test – самотестирование по включению питания) выполняется специальная процедура для обнаружения монтированных последовательных адаптеров. Поиск адаптеров осуществляется в последовательности групп, представленных в табл.2.4.

Базовые адреса обнаруженных адаптеров записываются в последовательные слова памяти, начиная с адреса 400H (там резервированы для этого 4 слова). Первый обнаруженный адаптер становится адаптером 0 (COM1), второй становится адаптером 1 (COM2) и т.д. Если, например, монтирован только один адаптер, его базовый адрес будет записан в слово по адресу 400H, и он становится адаптером 0, независимо для какой группы он настроен. Остальные три слова с адресами 402H, 404H, и 406H останутся без изменений, т.е. сохранят нулевые значения. Записанные таким образом базовые адреса используются позднее драйвером последовательного интерфейса. При получении запроса для работы с определенным адаптером (от 0 до 3), драйвер извлекает его базовый адрес из соответствующего слова и выполняет запрос. Если значение слова равно 0, драйвер немедленно возвращает управление.

К приведенной выше схеме необходимо сделать одно уточнение. В процедуру для обнаружения адаптеров включены только первые две группы, т.е. BIOS может обнаружить не более двух адаптеров. Это ограничение можно обойти. Если, например, хотим работать с тремя последовательными адаптерами, можем настроить их соответственно для первой, второй и третьей групп. BIOS обнаружит первые два, и драйвер последо-

вательного интерфейса будет распознавать их под номером 0 и 1. Затем нужно записать базовый адрес третьего адаптера (3E8H) в слово по адресу 404H. Далее драйвер последовательного интерфейса будет распознавать его под номером 2.

Число последовательных адаптеров, обнаруженных BIOS, записываются в биты 9 - 11 слова по адресу 410H и его можно получить при помощи прерывания 11H BIOS. BIOS считает, что адаптер данной группы существует, если после чтения из третьего порта группы (3FAH или 2FAH) биты 3-7 равны нулю.

Во время POST в байты по адресам 47CH, 47DH, 47EH и 47FH записывается единица. Эти байты соответствуют адаптерам с номерами 0, 1, 2 и 3. Они определяют интервал времени, по истечении которого BIOS считает, что соответствующий адаптер не выполнил запрос (таймаут). Проверка окончания операции осуществляется в цикле, и значение соответствующего байта используется как счетчик цикла. При обнулении счетчика BIOS принимает, что операция не выполнена. В BIOS нет обслуживающей функции для изменения этих байтов, но пользовательские программы могут изменять их непосредственно как адреса памяти.

BIOS имеет четыре обслуживающие функции для работы с последовательным интерфейсом. Они обособлены в отдельный модуль BIOS – драйвер последовательного интерфейса. Все функции выполняются посредством прерывания 14H, а номер функции задается в регистре AH. Нужно обратить внимание, что при инициализации последовательного адаптера (функция 00H) BIOS запрещает ему выполнять аппаратное прерывание. Объясним это подробнее. Последовательный адаптер можно программировать так, чтобы он выполнял аппаратное прерывание в определенных случаях: при получении данных, при изменении состояния линии и др. Это позволяет программам выполнять другую работу, до наступления ожидаемого изменения интерфейса. Однако BIOS работает подругому. Для обнаружения изменения она проверяет в цикле один из портов адаптера (поулинг). Следовательно, для осуществления асинхронной работы с последовательным интерфейсом необходимо программировать адаптер непосредственно. Последовательные адаптеры с базовыми адресами 3F8H и 3E8H (первая и третья группы) выполняют прерывание 0CH (уровень 4), в адаптеры с базовыми адресами 2F8H и 2E8H (вторая группа) – прерывание 0BH (уровень 3).

2.4. Обслуживающие функции BIOS для работы с последовательным интерфейсом

Функция 00H. Параметризация.

Эта функция задает различные параметры последовательного интерфейса и инициализирует последовательный адаптер. В DX задается номер адаптера. Параметры кодируются в AL, как показано в табл.2.5. После за-

вершения функции AX содержит информацию о состоянии линии и модема (точнее линии и осведомительных сигналов), как и при функции 03H.

Таблица 2.5

Параметры последовательного интерфейса

Номера битов	Значение
7,6,5	Скорость: 110; 150; 500; 600; 1200; 2400; 4800; 9600 бит/с
4,3	00 и 10 – без контроля; 11 – по четности; 01 – по нечетности
2	Стоп-биты: 0 – один стоп-бит; 1 – два стоп-бита
1,0	Размер символа: 10 – семь битов; 11 – восемь битов

Функция 01H. Передача символа.

Эта функция посылает один символ по интерфейсу. Символ должен быть в AL. После завершения функции AH содержит информацию о результате операции. Если бит 7 AH равен 1, операция неуспешна. В этом случае дополнительная информация имеется в битах 4, 5 и 6 (см. функцию 03H). Если бит 7 AH равен 0, операция успешна. В этом случае остальные биты показывают состояние линии, как и при функции 03H.

Функция 02H. Получение символа.

Эта функция ожидает получение символа по интерфейсу. Символ получается в AL. Функция завершает свое выполнение, как при получении символа, так и при получении сигнала об ошибке или при состоянии таймаута. После завершения функции AH содержит информацию о результате операции. Если AH равно 0, операция успешна, и AL содержит полученный символ. Если AH не равно 0, операция неуспешна. В этом случае дополнительная информация имеется в битах 1, 2, 3, 4 и 7 (см. функцию 03H). Если бит 7 равен 1, остальные биты недействительны.

Функция 03H. Информация о состоянии.

Эта функция возвращает в AH информацию о состоянии линии и в AL – информацию о состоянии модема. Значения отдельных битов AH и AL показаны в табл.2.6 и 2.7.

3. СПОСОБЫ ЗАЩИТЫ ИНФОРМАЦИИ ОТ НЕСАНКЦИОНИРОВАННОГО ДОСТУПА. РОССИЙСКИЙ СТАНДАРТ ШИФРОВАНИЯ ДАННЫХ ГОСТ 28147-89

Криптостойкость, заложенная в рассматриваемом в этом разделе алгоритме, значительно превосходит стандартизованный алгоритм США DES. Кроме того, применяя разумные приемы программирования можно достичь

превосходства в быстродействии по сравнению с реализацией на языке С, приведенной в ГОСТе, в 8 раз.

Таблица 2.6

Состояние линии, отраженное в регистре AH

Номер бита	Значение при единичном состоянии бита
7	Таймаут
6	Регистр для передачи пуст (TX SHIFT EMPTY)
5	Буфер для передачи пуст (TX HOLD EMPTY)
4	Получен BREAK (BREAK DETECT ERROR)
3	Ошибка формата данных (FRAMING ERROR)
2	Ошибка четности (PARITY ERROR)
1	Потеря данных (OVERRUN ERROR)
0	Получены данные (DATA READY)

Таблица 2.7

Состояние модема (осведомительных сигналов адаптера), отраженное в регистре AL

Номер бита	Значение при единичном состоянии бита
7	DCD активен (DATA CARRIER DETECT)
6	RI активен (RING INDICATOR)
5	DSR активен (DATA SET READY)
4	CTS активен (CLEAR TO SEND)
3	Осуществлена смена DCD
2	Осуществлена смена RI
1	Осуществлена смена DSR
0	Осуществлена смена CTS

3.1 Основные понятия

Официально ГОСТ называется *"Алгоритм криптографического преобразования данных ГОСТ 28147-89"* — это несколько шире, чем просто зашифровывание или расшифровка данных.

Все режимы криптопреобразований данных, согласно ГОСТ базируются на трех циклах алгоритма (далее – базовые циклы):

- цикл зашифровывания (32-З);
- цикл расшифровки (32-Р);
- цикл выработки имитоприставки (16-З);

Смысл кода, приведенного в скобках, будет пояснен несколько позже.

Кроме ГОСТа используется дополнительная информация, секретность которой обеспечивает секретность шифрованного сообщения. Эта информация представляет собой 2 массива данных – *ключ* и *таблицу замен*, приведем их характеристики.

1. *Ключ* – это массив из 8 32-битовых элементов, обозначаемых в дальнейшем X_i, где i изменяется от 0 до 7. Таким образом, размер ключа составляет 32x8=256 битов, или 32 байта.

2. *Таблица замен* – двумерная таблица – набор из 8 одномерных массивов (узлов замен), каждый из которых содержит 16 различных 4-битовых чисел (от 0 до 15) в произвольном порядке. Обозначим $K_m(1)$ значение 1-го элемента в m-ом узле замен. При этом m изменяется в пределах 0...7, а 1 – в пределах 0... 15. Таким образом, общий объем таблицы замен равен 8 узлов x16 элементов x4 бита/элемент = 512 битов = 64 байта.

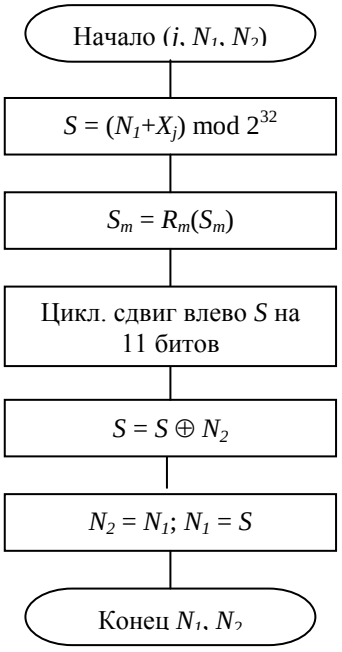


Рис.3.1. Схема алгоритма основного шага преобразования

Рассмотрим основной шаг криптопреобразования; структура алгоритма которого представлена на рис.3.1. На входе шага заданы два 32-битовых элемента данных — N_1, N_2, с этими элементами выполняются следующие манипуляции:

1) добавление к N_1 элемента ключа — сложение по модулю 2^32.
2) поблочная замена результата по 4 бита по таблице замен;
3) циклический сдвиг результата на 11 битов влево;
4) побитовое сложение результата по модулю 2 с элементом N_2;
5) перестановка элементов: $N_2 \leftarrow$ старое N_1, $N_1 \leftarrow$ результат.

После этого новые элементы N_1, N_2 выдаются в качестве результата шага. Так как в основном шаге используется только один элемент ключа, еще одним параметром шага является номер этого элемента.

Рассмотрим теперь базовые циклы криптоалгоритма ГОСТа. Они отличаются друг от друга только числом повторений основного шага и порядком просмотра элемента ключа. В обозначении цикла *nn-X* первый элемент (*nn*) — это число повторений основного шага, а второй кодирует порядок просмотра элементов ключа (буква З — порядок зашифровывания или Р — расшифровки). Кроме того, в конце циклов шифрования предусмотрена дополнительная перестановка элементов.

Приведем порядок использования элементов ключа для трех базовых циклов:

1) цикл зашифровывания (32-З) — 3 раза вперед, 1 раз назад:
0,1,2,3,4,5,6,7,0,1,2,3,4,5,6,7,0,1,2,3,4,5,6,7,7,6,5,4,3,2,1,0
2) цикл расшифровки (32-Р) — 1 раз вперед, 3 раза назад:
0,1,2,3,4,5,6,7,7,6,5,4,3;2,1,0,7,6,5,4,3,2,1,0,7,6,5,4,3,2,1,0
3) цикл выработки имитоприставки (16-З) — 2 раза вперед:
0,1,2,3,4,5,6,7,0,1,2,3,4,5,6,7

Блок-схемы базовых циклов приведены на рис.3.2 – 3.4. Каждый из циклов получает на входе два 32-битовых слова и после серии основных шагов выдает в качестве результата также два 32-битовых слова.

3.2. Основные режимы шифрования

ГОСТ 28147-89 предусматривает четыре режима шифрования данных:

1) простая замена;
2) гаммирование;
3) гаммирование с обратной связью и дополнительный режим;
4) выработка имитоприставки.

В любом из этих режимов данные обрабатываются блоками по 64 бита — именно поэтому ГОСТ относится к блочным шифрам. Кратко опишем основные режимы шифрования.

Простая замена. Зашифровывание заключается в применении цикла 32-З к блокам открытого текста, расшифровка — в применении цикла 32-Р к блокам шифротекста. Это наиболее простой режим шифрования, и он имеет следующие недостатки:

– с точки зрения стойкости шифра, одинаковые блоки исходных данных дают одинаковые блоки шифротекста; понятно, что для такой характеристики как криптостойкость алгоритма это очень плохо;

– с точки зрения удобства применения, если длина массива информации не кратна 8 байтам, то возникают 2 небольшие проблемы:

- чем и как дополнять последний блок до полных 8 байтов;
- после зашифровывания неполного блока в нем все 8 байтов станут значащими, то есть вместе с шифротекстом надо хранить количество байтов в последнем блоке исходного текста.

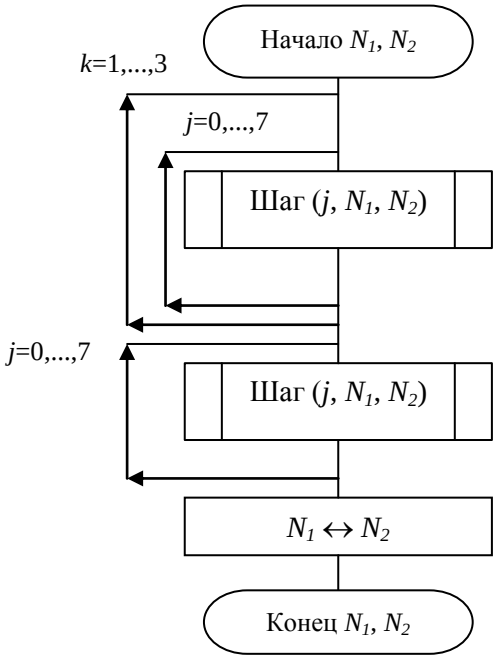


Рис.3.2. Схема алгоритма базового цикла зашифровывания 32-З

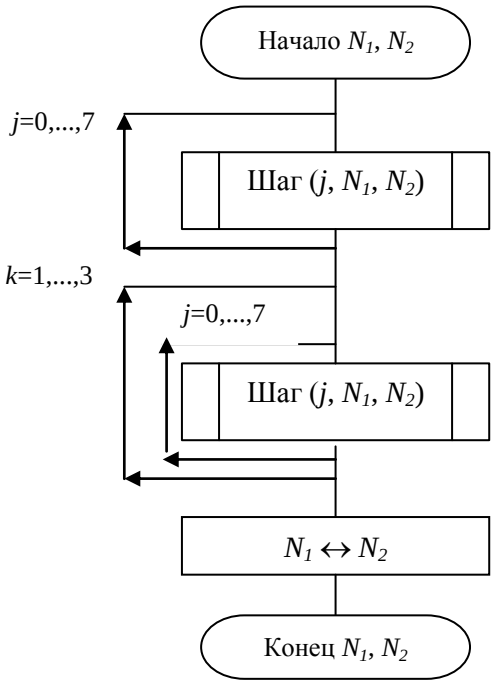


Рис.3.3. Схема алгоритма базового цикла расшифровки 32-Р

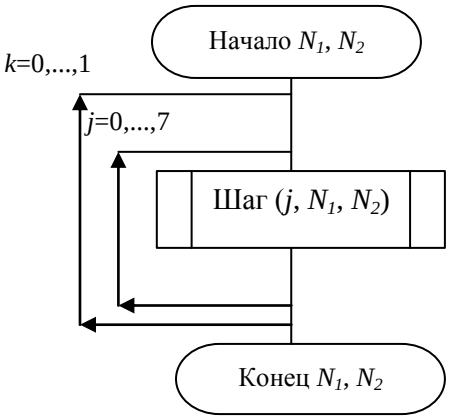


Рис.3.4. Схема алгоритма базового цикла выработки имитовставки 16-З

ГОСТ ограничивает возможные случаи применения простой замены шифрованием ключевой информации (ключи и таблицы замен).

Гаммирование. Этот режим заключается в наложении на открытые данные гаммы с помощью побитовой функции "исключающее или". Зашифровывание и расшифровка в этом режиме не отличаются друг от друга.

$$T^{u}_{i} = T^{o}_{i} \oplus \Gamma_i$$

$$T^o_i = T^ш_i \oplus Г_i$$

для i = l,...,K (К — число блоков данных в шифруемом массиве).

Здесь T^o_i , $T^ш_i$, $Г_i$ — 64-битовые блоки открытого текста, шифротекста, гаммы соответственно. Блоки гаммы получаются зашифровыванием в режиме простой замены некоторой последовательности 64-битовых блоков, вырабатываемых датчиком псевдослучайных чисел. От этого датчика не требуется обеспечения никаких статистических характеристик выходной последовательности, а нужен лишь максимальный период повторения данных. Схема алгоритма шифрования в режиме гаммирования приведена на рис.3.5. В качестве входного параметра алгоритма шифрования в этом режиме используется 64-битовый блок данных S, именуемый синхропосылкой, или начальным заполнением. Константы C_1, C_2 из блока генерации гаммы имеют следующие значения в шестнадцатеричном представлении:

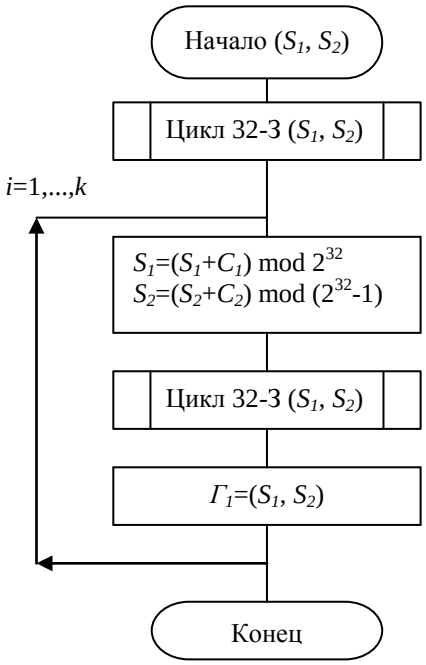


Рис.3.5. Схема алгоритма выработки гаммы для шифрования

C1 = 01010101h

C2 = 01010104h

Гаммирование с обратной связью. Данный режим похож на режим гаммирования и отличается от него только тем, что для выработки блока гаммы для шифрования следующего блока данных используется блок шифротекста, полученный на предыдущем шаге. Этим достигается зацепление блоков — каждый блок при шифровании зависит от всех предыдущих. Какое это имеет значение, поясним на примере: пусть почтальон-злоумышленник (лицо, имеющее полный доступ к шифрованным данным, но не владеющее ключом и не способное расшифровать сообщение) узнал, что в передаваемом сообщении 13-й байт от начала является старшим разрядом числа, например суммы некоторого контракта. Пусть также ему известно значение этой цифры, допустим, это будет 1. Тогда он элементарно может заменить ее любой другой цифрой, например на 2, не расшифровывая сообщения: $A_{13} = A_{13} \oplus$ '1' $\oplus$ '2'

Если бы данные шифровались в режиме простой замены, это привело бы к порче всего блока — после расшифровки мы получили бы совершенно случайную комбинацию символов на месте с 9-го по 16-й байты нашего текста. Если зашифровывание выполнено в режиме гаммирования, то после расшифровки мы получим текст, в котором в 13-й позиции вместо верной цифры 1 стоит цифра 2 и больше никаких изменений нет! Очевидно, не надо объяснять, насколько потенциально опасной может быть

такая подмена. Если же зашифровывание выполнено в режиме гаммирования с обратной связью, то после расшифровки мы получим в блоках данных до испорченного включительно тот же результат, что и при гаммировании, но все последующие блоки окажутся запорченными, то есть после расшифровки их содержимое окажется бессмысленной комбинацией расширенных кодов ASCII. Отсюда становится ясно, что шифрование в режиме гаммирования с обратной связью делает очевидным нарушение целостности данных, если только изменения не внесены лишь в последний блок данных шифротекста. Схемы алгоритмов зашифровывания и расшифровки в режиме гаммирования с обратной связью приведены на рис.3.6, 3.7.

Имитоприставка — это контрольная комбинация, зависящая от открытых данных и секретной ключевой информации. Цель использования имитоприставки — обнаружение всех случайных или преднамеренных изменений в массиве информации. Проблема, описанная в предыдущем пункте, может быть успешно решена с помощью добавления к шифрованным данным имитоприставки.

Для потенциального злоумышленника две следующие задачи, если он не владеет секретным ключом, практически неразрешимы:

- вычисление имитоприставки для заданного открытого массива информации;
- подбор открытых данных под заданную имитоприставку;

Схема алгоритма выработки имитоприставки приведена на рис.3.8. В качестве имитоприставки берется не весь блок, полученный на выходе последнего цикла 16-З, а только его часть. Как правило, это 32 младших бита блока, т.е. N_1.

3.3. Криптостойкость алгоритма

При выборе криптоалгоритма для использования в конкретной разработке одним из определяющих факторов является его криптостойкость. Вопрос о стойкости шифра при ближайшем рассмотрении сводится к двум взаимосвязанным вопросам:

- можно ли вообще раскрыть данный шифр;
- если да, то насколько это трудно сделать практически.

Шифры, которые вообще невозможно раскрыть, называются абсолютно или теоретически стойкими. Существование подобных шифров доказывается теоремой Шеннона.

Цена этой стойкости настолько велика, что они могут использоваться только в каналах передачи сообщений исключительной важности, задействованных нечасто.

Это значит, что наиболее употребительные схемы шифрования теоретически нестойки, т.е. могут быть раскрыты за конечное время или, что, точнее, за конечное число шагов. Тем не менее, это не мешает с успехом их использовать. Для таких алгоритмов не существует понятие

практической стойкости, отражающее практическую трудность их раскрытия.

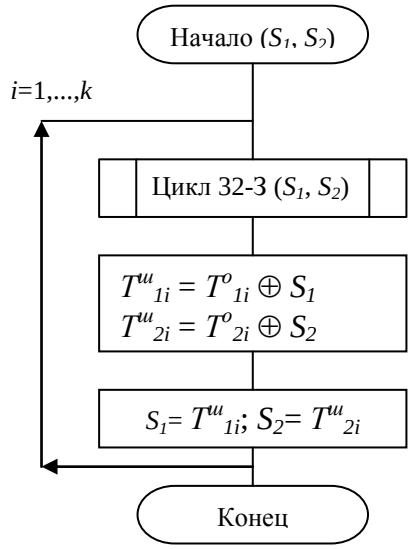


Рис.3.6. Схема алгоритма зашифровывания блока данных в режиме гаммирования с обратной связью

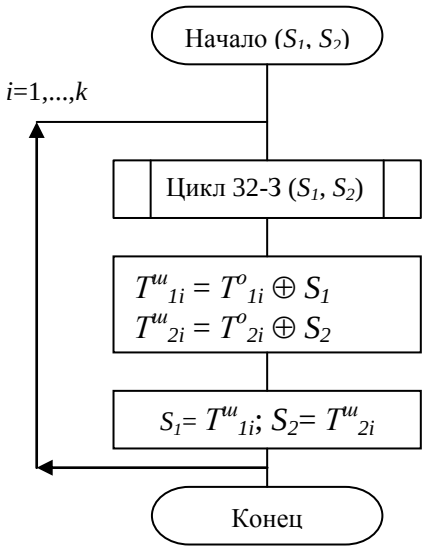


Рис.3.7. Схема алгоритма расшифровки данных в режиме гаммирования с обратной связью

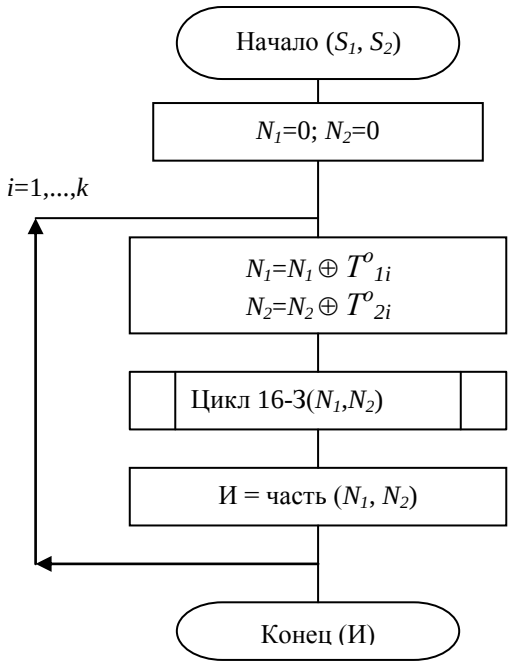


Рис.3.8. Схема алгоритма выработки имитопристав ки

Количественной мерой этой трудности может служить число в элементарных арифметических и логических операций, которые необходимо выполнить, чтобы раскрыть шифр, то есть, чтобы для заданного шифротекста с вероятностью, не меньшей заданной величины, выдать соответствующий открытый текст. При этом в дополнение к дешифруемому тексту криптоаналитик может располагать блоками открытого текста и соответствующего шифротекста или даже возможностью получить для любого выбранного им открытого текста шифротекст.

Все современные криптосистемы построены по принципу Кирхгоффа, т.е. секретность зашифрованных сообщений определяется секретностью ключа. Это значит, что даже если сам алгоритм шифрования известен криптоаналитику, он, тем не менее, не в состоянии расшифровать сообщение, если не располагает соответствующим ключом. Все классические блочные шифры, в том числе DES (американский алгоритм шифрования) и ГОСТ, построены по этому принципу и спроектированы таким образом, чтобы не было пути вскрыть их более эффективным способом, чем полным перебором по всему ключевому пространству, т.е. по всем возможным значениям ключа. Ясно, что стойкость таких шифров определяется размером используемого в них ключа.

В криптоалгоритме ГОСТ используется 256-битовый и ключ, и объем ключевого пространства составляет 2^{256}.

Российский стандарт проектировался с большим запасом и по стойкости на много порядков превосходит американский стандарт DES с его размером ключа в 56 битов и объемом ключевого пространства 2^{56}. В свете прогресса современных вычислительных средств этого явно недостаточно. В конце 80-х годов стоимость аппаратуры для вскрытия DES оценивалась специалистами в несколько миллионов долларов США [4]. С учетом постоянного прогресса в области микроэлектроники сейчас эта величина на порядок ниже, что вполне по плечу спецслужбам даже не очень крупного государства.

В этой связи создание новых программных реализаций DES может представлять скорее исторический или спортивный, чем практический интерес. Но этого мало, DES уступает ГОСТу не только в криптостойкости, он также гораздо хуже приспособлен для программной реализации.

3.4. Качество ключевой информации

Еще один вопрос, оставшийся без рассмотрения — это выработка ключевой информации и ее качество. Исчерпывающий ответ на вопрос о критериях качества ключей и таблиц замен ГОСТа можно получить только у разработчиков алгоритма. Соответствующие данные не были опубликованы в открытой печати, однако косвенным свидетельством в пользу существования таких критериев является практика использования государственными организациями ключевой информации, полученной из специализированных ис-

точников. Для обычного же пользователя вполне достаточно статистического качества ключевой информации, заключающегося в следующем:

- узлы таблицы замен должны быть случайными независимыми;
- подстановками в 16-элементном множестве;
- ключ должен быть массивом случайных независимых равновероятных битов.

Термины "случайный" и "независимый" понимаются на практике как "удовлетворяющий определенным тестам на случайность и независимость". Например, равновероятность значений битов ключа проверяется с помощью *критерия Пирсона*, а независимость – с помощью *критерия серий*.

При написании программ, использующих криптографические средства, необходимо позаботиться об утилитах, вырабатывающих ключевую информацию, а для таких утилит необходим источник случайных чисел (СЧ) высокого статистического качества и криптостойкости. Наилучшим подходом здесь было бы использование аппаратных датчиков СЧ, однако, это не всегда приемлемо по экономическим соображениям. В качестве разумной альтернативы возможно (и очень широко распространено) использование различных программных датчиков СЧ.

4. СМАРТ-КАРТЫ С ОТКРЫТОЙ ПАМЯТЬЮ GFM-2K. ПРОТОКОЛ СИНХРОННОГО ОБМЕНА I^2C

4.1. Общие сведения

Смарт-карты с открытой памятью, с точки зрения внутренней организации, представляют собой перепрограммируемую энергонезависимую память (EEPROM). Доступ к этой памяти ведется с использованием протоколов последовательной передачи данных. Причиной последнего является то, что такие карты ориентированы на стандарт ISO 7816-2. Вследствие этого ограничено количество контактов.

Одним из наиболее крупных поставщиков смарт-карт на российский рынок является фирма "Gemplus". Эта фирма разрабатывает и изготовляет самые разные смарт-карты: от самых простых и дешевых (с открытой памятью) до самых дорогих и, естественно, наиболее защищенных (микропроцессорные асинхронные карты). В общем случае стоимость и область применения любой смарт-карты определяется, прежде всего, ее защищенностью от несанкционированного доступа к записанной на нее информации.

Используемый в лабораторных работах контроллер смарт-карт предназначен для работы со смарт-картами с открытой памятью фирмы "Gemplus" - GFM-2K. Некоторые технические характеристики этих карт приведены в табл.4.1.

Таблица 4.1

Характеристики смарт-карт GFM-2K

Параметр	Значение	Единица	Примечание

		измерения	
Уровни сигналов	ТТЛ	–	–
Питающее напряжение	5 ± 25%	В	–
Время выборки	< 10	мкс	–
Время записи	Около 10*	мс	–
Протокол работы	I^2C (Phillips)	–	–
Объем защищенной памяти	–	–	–
Объем незащищенной памяти	2048	Бит	Организация памяти: 256 x 8 бит или 32 x 8 байт **

** – зависит от значения напряжения питания.*

******* *– такая организация памяти определяется протоколом работы карт (см. ниже).*

GFM-2K соответствует стандартам ISO 7816-1,2. Расположение контактов карты представлено на рис.4.1.

Контакты Vcc и Vdd предназначены для подачи питания на схему смарт-карты (Vdd – нулевой потенциал источника, Vcc – положительный потенциал). Контакты SCL и SDA используются при формировании синхронного протокола информационного обмена с картой. Остальные контакты смарт-карты не используются.

Функциональная схема чипа смарт-карт GFM-2K изображена на рис.4.2. Ниже описаны функции каждого из представленных на данной схеме блоков.

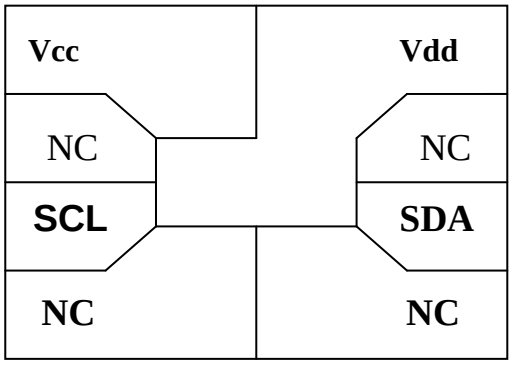


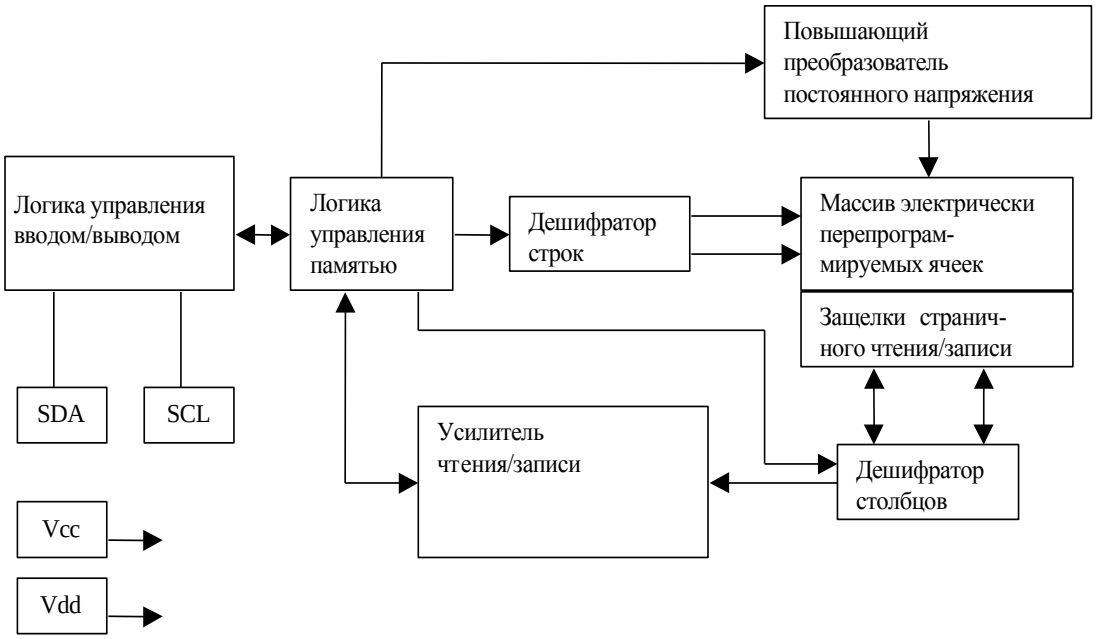
Рис.4.1. Расположение и наименование контактов смарт-карты с открытой памятью GFM-2K

Логика управления вводом/выводом является логической схемой, анализирующей состояние выводов SCL и SDA при ожидании передачи от считывателя и формирующую выходной протокол при передаче данных.

В соответствии с текущей операцией схема управления вводом/выводом выдает управляющие и информационные сигналы остальным функциональным блокам.

Логика управления памятью является центральным блоком, так как под его управлением происходят все операции с памятью карты (запись, считывание, стирание).

При выполнении операций записи и считывания на дешифраторы строк и столбцов выдается необходимая кодовая последовательность, зависящая от адреса ячейки и позволяющая адресоваться к конкретным электрически перепрограммируемым ячейкам в соответствии с этим адресом. При этом при считывании с выхода массива электрически перепрограммируемых ячеек формируются импульсы, характеризующие состояние выбранных ячеек. После усиления в усилителе чтения/записи данные импульсы воспринимаются как информационные и сформированные биты данных передаются внешнему считывающему устройству.



При записи слова усилитель чтения/записи формирует импульсы записи, управляющие переводом выбранных ячеек памяти в нулевое или единичное состояние. Процесс записи в электрически перепрограммируемую память происходит с помощью высокого напряжения, получаемого повышающим преобразователем постоянного напряжения по команде от логики управления памятью.

Блок защелок страничного чтения/записи используется при реализации так называемого режима постраничной записи (под страницей понимается 8 байт памяти, начинающихся с адреса, кратного восьми) и представляет собой набор регистров-защелок, т.е. регистров, предназначенных для кратковременной фиксации данных. Данный режим позволяет значительно уменьшить время чтения и записи содержимого памяти.

Как было указано выше, смарт-карты с открытой памятью GFM-2K работают по протоколу последовательной передачи данных I^2C, формируемым логикой управления вводом/выводом. Данный протокол – двухпровод-

ной, т.е. использует две линии: одну – для передачи данных, другую – для синхронизации. Протокол разработан фирмой ‘Philips’ и используется уже достаточно длительное время.

4.2. Описание протокола I^2C

Шина, с которой работают два (или более) устройства для передачи данных, представляет собой совокупность синхронизирующей линии (SCL) и линии передачи данных (SDA). Устройства работают по принципу "Master – Slave" ("Хозяин" – "Раб"). Устройство Master генерирует синхроимпульсы, управляет доступом в шину и реализует состояния "Старт" и "Стоп". Любое из устройств может работать как передатчик данных и как приемник, однако именно Master определяет какой из режимов будет выбран в следующий момент.

Метод доступа к шине следующий:

- передача данных может быть начата только, когда шина свободна;
- при передаче данных состояние линии передачи данных не должно изменяться.

В противном случае это будет воспринято как сигналы "Старт" или "Стоп".

Далее описаны основные состояния шины (см. рис.4.3).

Шина не занята.

SCL и SDA имеют высокий уровень.

Начало передачи данных ("Старт").

Изменение SDA от высокого уровня к низкому, когда SCL имеет высокий уровень. Любая команда должна начинаться со "Старта".

Конец передачи данных ("Стоп").

Изменение SDA от низкого уровня к высокому, когда SCL имеет высокий уровень. Все операции должны заканчиваться "Стопом".

Когда после "Старта" состояние линии SDA не меняется на время высокого значения сигнала SCL, считается, что передан очередной бит данных команды или адреса. Подготовка к следующей передаче бита (т.е. изменение состояния линии SDA) должна осуществляться во время низкого уровня на линии SCL.

Количество передаваемых бит между сигналами "Старт" и "Стоп" задается устройством Master и, теоретически, не ограничено. Однако только последние 16 бит будут зафиксированы приемником.

Таким образом, память, предварительно накапливающая принимаемые биты, работает по принципу FIFO. (First-In, First-Out). Передача бит в составе байта осуществляется старшими разрядами вперед.

Устройство, работающее на прием, после приема каждого байта данных обязано послать сигнал уведомления (Acknowledge). Master специально генерирует для этого лишний тактовый импульс. Во время осуществления цикла записи в EEPROM управляющий блок запоминающего устройства Slave не генерирует сигнал уведомления до окончания цикла записи. Сигналом уведомления является удержание в низком состоянии линии SDA на все

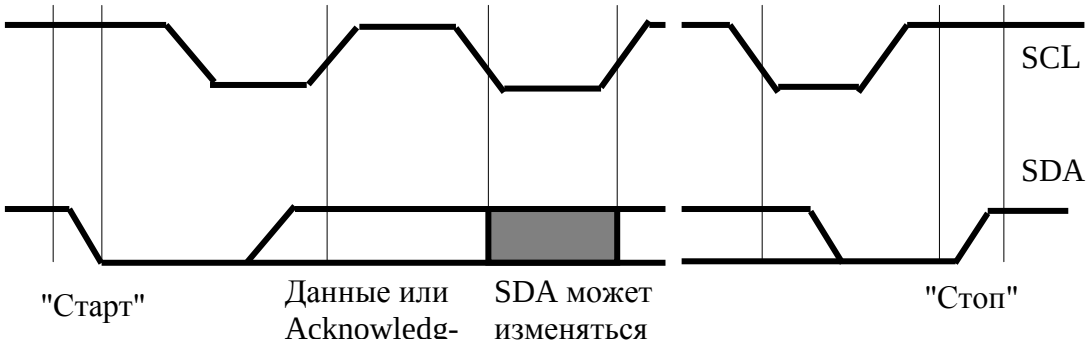


Рис.4.3. Состояния шины данных

время присутствия специального такта. Master должен сигнализировать устройству Slave об окончании получения данных путем удержания (отсутствия генерации) сигнала уведомления. Slave должен ответить на это установкой линии SDA в высокое состояние, чтобы позволить устройству Master послать "Стоп".

Протокол I^2C предполагает наличие на шине множества устройств. Тогда для обращения к конкретному устройству необходимо сгенерировать некоторый "Сигнал выбора". Однако, поскольку, I^2C – последовательный протокол, то это должно быть реализовано в последовательном виде. Делается это следующим образом. Каждая операция начинается с посылки "Старта" и, вслед за ним, управляющего байта. Структура управляющего байта показана на рис.4.4. Первые четыре бита адреса устройства Slave показывают тип устройства, с которым планируется осуществлять работу. Последние три бита позволяют выбирать среди устройств одного типа. Следует отметить, что протокол I^2C имеет еще один формат с расширенным полем адресов устройств. Для смарт-карт GFM-2K адрес показан на рис.4.4. Восьмой бит управляющего байта задает режим работы: чтения или записи.

Запись байта.

После передачи управляющего байта и получения уведомления Master должен отправить восьмибитный адрес, который Slave записывает в свой адресный указатель, отвечая на прием адреса сигналом уведомления. После получения данного сигнала Master посылает восьмиразрядное слово, которое Slave запишет по указанному адресу и, затем, отправит сигнал уведомления. В конце операции Master обязан отправить "Стоп" для уведомления устройства Slave (рис.4.5).

Последний сигнал уведомления отправляется только после того, как запись произведена. Таким образом, для увеличения пропускной способно-

сти шины I^2C целесообразно после отправки последнего бита записываемого слова поллингом (т.е. постоянным опросом) ожидать появления уведомления, после чего немедленно продолжать работу.

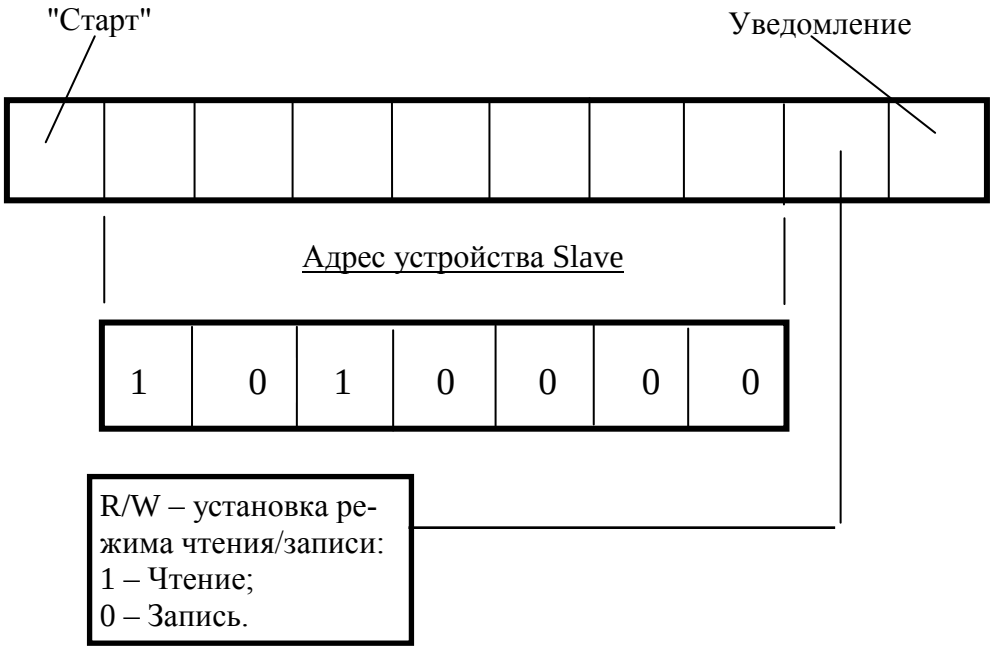


Рис.4.4. Управляющее слово смарт-карт GFM-2K.

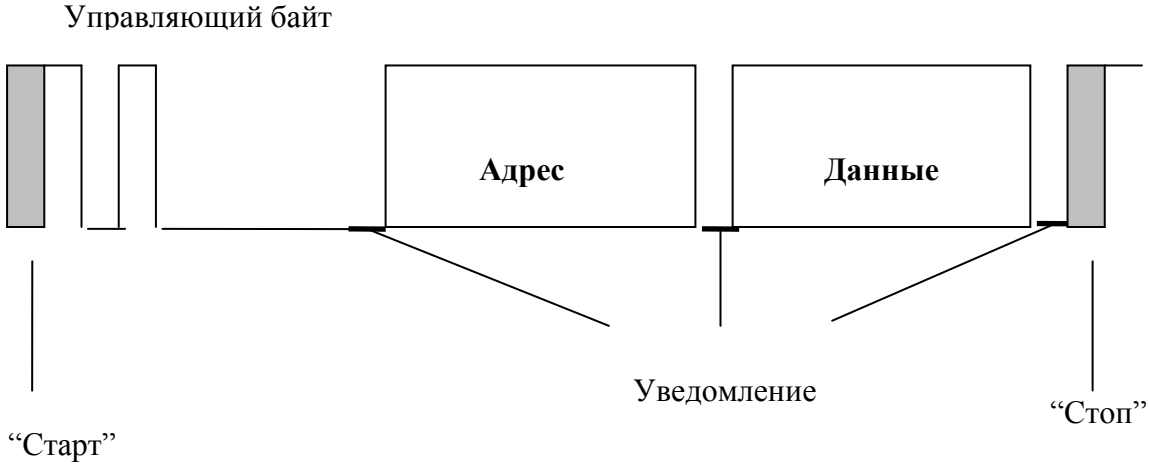


Рис.4.5. Временная диаграмма записи байта в картах GFM-2K

Запись страницы.

С точки зрения быстродействия, операция записи байта нерациональна при работе с большими объемами информации. Для уменьшения временных затрат предусмотрена запись страницами. В нашем случае страница представляет собой восемь последовательно расположенных в памяти байт. Причем, первый байт из этой серии расположен по адресу, младшие 3 бита которого – нулевые. Вследствие этого всю память GFM-2K можно представить

как последовательность из 256 байт или 32 неперекрывающихся страниц по 8 байт.

Если Master начинает запись не с начала какой-либо страницы, то даже в этом случае выхода из пределов этой страницы не произойдет. Причиной этого является то, что при инкрементировании адреса старшеие 5 бит его не изменяются.

Таким образом, начав запись с середины страницы, после переполнения младших трех бит запись продолжится с ее начала.

Протокол этой операции отличается от протокола операции записи байта следующим:

- после отправки первого байта информации и получения уведомления Master не отправляет "Стоп", а продолжает посылку еще семи байт, получая уведомление после каждого из них;
- при отправке более восьми байт младшие три бита адреса будут циклически инкрементироваться, затирая записанные ранее там значения.

Цикл записи полученной информации непосредственно в EEPROM начинается после получения бита "Стоп". Однако необходимо помнить, что уведомление о получении очередного байта команды будет послано не ранее, чем закончится цикл записи.

Чтение по текущему адресу.

Как было указано выше, чип устройства Slave имеет внутренний адресный указатель. Этот указатель после каждой операции чтения и записи инкрементируется на 1, т.е. если, например, операция записи была произведена по адресу n, то следующая за ней операция чтения по текущему адресу, прочитает байт, содержащийся в ячейке (n+1).

После получения управляющего байта с R/W = 1 устройство Slave посылает сигнал уведомления о получении команды, а затем байт, адрес которого содержится в данный момент во внутреннем счетчике. Master устройство после получения байта данных не уведомляет об этом устройство Slave, но обязано сгенерировать сигнал "Стоп" (рис.4.5).

Произвольное чтение.

Операция произвольного чтения дает устройству Master возможность доступа к любой ячейке памяти Slave. Для реализации этой операции, прежде всего, необходимо установить интересующий адрес во внутренний указатель Slave. Это делается отправкой адреса, как части операции записи байта. Таким образом, после отправки управляющего байта с R/W = 0 и адреса, а также получения уведомления, Master снова генерирует "Старт". Это прерывает операцию записи, но требуемый адрес оказывается в адресном указателе устройства Slave.

Далее (рис.4.6) осуществляется описанная выше операция чтения по текущему адресу

В протоколе I^2C предусмотрена также операция последовательного чтения, которая позволяет за одно обращение прочитать целиком всю память. Однако разработчики чипов смарт-карт GFM-2K сочли использование подобной операции излишним.

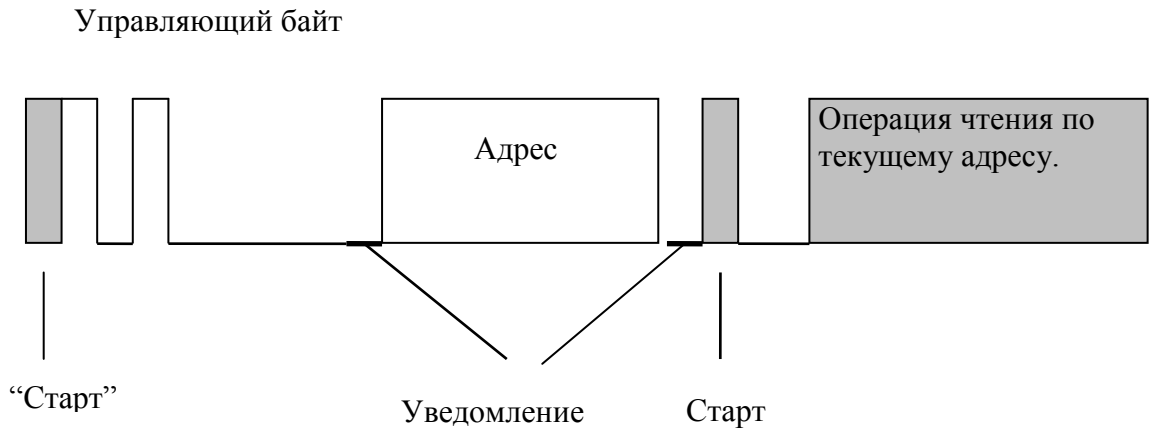


Рис.4.6. Временная диаграмма чтения по текущему адресу в картах GFM-2K

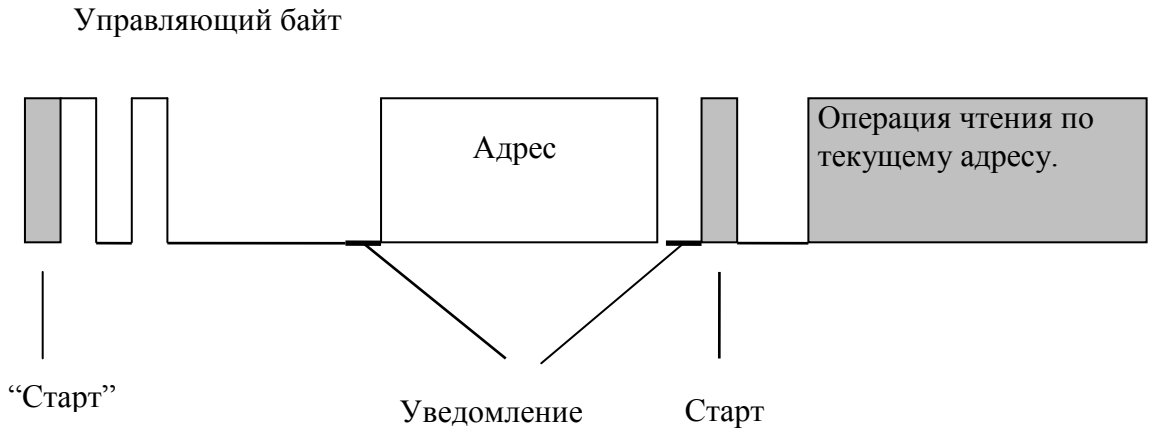


Рис.4.7. Временная диаграмма чтения по произвольному адресу в картах GFM-2K

5. УНИВЕРСАЛЬНЫЙ КОНТРОЛЛЕР СМАРТ-КАРТ

В этой главе приводится описание универсального контроллера смарт-карт *Card Lab* (далее – смарт-контроллер), предназначенного для выполнения лабораторных работ.

5.1. Техническое описание

Технические параметры смарт-контроллера приведены в табл.5.1.

Таблица 5.1

Параметры контроллера

Интерфейс связи с компьютером	RS-232
Скорость обмена с компьютером	9600 бит/с
Протокол обмена	ASK BUS (v. 2.0)
Алгоритм шифрации данных	ГОСТ 28147-89
Длина ключа шифрации	256 бит
Максимальное время срабатывания электронной пломбы	0.2с
Напряжение питания	~220 В / 50 Гц
Потребляемая устройством мощность не более	1 Вт
Габаритные размеры	200 х 112х50 мм
Температурный диапазон	0..+40 ˚С

Обмен с компьютером производится через стандартный COM-порт (RS-232) на скорости 9600 бит/с. В системе применяется протокол обмена ASK BUS (v. 2.0) между смарт-контроллером и компьютером. Описание интерфейса **ASK BUS** приведено в Приложении 1. Для защиты данных от произвольного чтения в смарт-контроллере реализован алгоритм шифрации данных. Ключ шифрации и таблица замен заносятся в контроллер при производстве.

Питание смарт-контроллера осуществляется от сети переменного тока с частотой 50–60 Гц и напряжением 220 В. Потребляемая при этом мощность не превышает 1Вт.

Функциональная схема смарт-контроллера приведена на рис.5.1

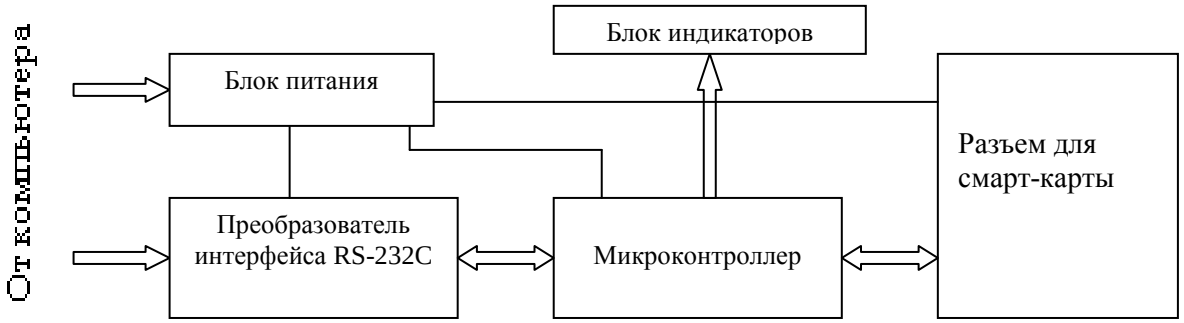


Рис.5.1. Функциональная схема контроллера смарт-карт

5.2. Подготовка к работе

В рассматриваемом варианте исполнения смарт-контроллер предназначен для работы со смарт-картами типа GFM-2K через программную оболочку верхнего уровня.

Смарт-контроллер позволяет производить запись данных на карту, чтение данных с карты и при необходимости автоматическую шифрацию/дешифрацию данных.

На лицевой панели расположена горизонтальная щель для смарт-карт и три светодиодных индикатора, позволяющих определить состояние смарт-контроллера.

Зеленый светодиод с надписью **"Питание"** зажигается после включения смарт-контроллера в сеть. Этот светодиод индицирует наличие питания на схеме контроллера. Если светодиод не горит, то следует проверить положение выключателя, работоспособность шнура питания, и наличие номинального напряжения в сети. Если смарт-контроллер не удается подключить, то необходимо обратится к инженеру.

Средний зеленый светодиод с надписью **"Готов"** зажигается в случае, когда смарт-контроллер готов к работе. Зажигание светодиода происходит после самотестирования отдельных узлов смарт-контроллера при включении питания.

Красный светодиод с надписью **"Авария"** зажигается, если была обнаружена ошибка при самотестировании внутренних узлов смарт-контроллера. Данная ситуация возникает в результате нарушения нормального режима работы внутренних узлов смарт-контроллера.

На задней стороне смарт-контроллера расположен разъем для подключения контроллера в сеть и интерфейсный разъем для работы с компьютером.

На первом этапе необходимо подготовить смарт-контроллер к работе. Вначале производится подключение питающего и интерфейсного проводов к смарт-контроллеру. Шнур питания подключается через разъем с надписью ~220 В / 50 Гц. Интерфейсный провод подключается через разъем с надписью RS-232. Второй конец интерфейсного провода должен быть подключен к COM - порту компьютера.

Внимание! *Подключение интерфейсного провода к COM-порту компьютера производится только при выключенном питании компьютера и смарт-контроллера во избежание перегрузки интерфейсных микросхем и соответственно выхода их из строя.*

Если после включения правый зеленый светодиод не загорается, то следует проверить подключение сетевого провода и положение выключателя.

При включении питания смарт-контроллер начинает тестирование своих внутренних узлов. Если после тестирования загорается светодиод "**Авария"**, то смарт-контроллер не готов к работе и необходимо обратится к персоналу лаборатории.

При успешном завершении тестирования происходит загорание светодиода "**Готов"** на передней панели смарт-контроллера, после чего смарт-контроллер готов к работе.

Программная оболочка для работы со смарт-контроллером может быть загружена как заблаговременно до включения питания смарт-контроллера, так и после готовности смарт-контроллера к работе. При помощи программы верхнего уровня необходимо установить связь со смарт-контроллером.

Пользователь должен быть заранее ознакомлен с программной оболочкой верхнего уровня.

Смарт-карту необходимо вставлять **чипом вверх** (чип показан на рис.4.1) в специальную щель считывателя смарт-карт на корпусе смарт-контроллера.

Смарт-карта должна входить в считыватель без усилий примерно на 50мм. Затем смарт-карта упирается в пружину, и с небольшим усилием карту следует вставить до упора, а затем отпустить. При этом карта должна отойти обратно не более, чем на 5 мм.

Это говорит о том, что карта зафиксирована в считывателе. Если смарт-карта отходит обратно примерно на 20мм, то это означает, что карта в считывателе не зафиксирована и необходимо повторно нажать на карту до упора. Если все попытки вставить карту в смарт-контроллер неудачны, то необходимо обратится к персоналу.

Изъятие карты из смарт-контроллера производится в том же порядке. Вначале необходимо надавить на карту до упора (примерно 4–5 мм при нормально вставленной карте), а затем отпустить карту. При этом карта выйдет из считывателя на 20 мм. Это говорит о том, что смарт-карта снята с фиксатора. После чего смарт карту можно вынуть из смарт-контроллера.

Внимание! *Запрещается вытаскивать карту из смарт-контроллера, если она не снята с фиксатора. Это может привести к преждевременному выходу из строя считывателя смарт-карт.*

После того, как смарт-карта помещена в смарт-контроллер, с ней можно начинать работу с помощью программы верхнего уровня.

После завершения работы со смарт-картой она должна быть вынута из смарт-контроллера.

Внимание! *Запрещается отключение смарт-контроллера из сети, либо изъятие смарт-карты из считывателя при работе с этой картой. Это может привести к потере данных на карте.*

При завершении работы с смарт-контроллером достаточно отключить его питание.

Внимание! *При необходимости отключения смарт-картроллера от компьютера, необходимо вначале отключить питание компьютера и смарт-контроллера, а затем производить отключение интерфейсного провода.*

5.3. Система команд смарт-контроллера

Смарт-контроллер предназначен для работы со смарт-картами типа GFM-2K, представляющих собой энергонезависимую память эквивалентную EEPROM 24LC02. Она имеет тот же интерфейс (I^2C) и протокол обмена, что и 24LC02, и тот же объем памяти 2048 бит (256 байт).

В рассматриваемой версии смарт-контроллера вся карта, на которую производится запись данных, логически разбита на 4 блока по 64 байта в каждом. Поэтому если на карту необходимо произвести запись 256 байт ин-

формации, то записываемая информация должна быть разбита на 4 части по 64 байта.

Если на карту необходимо записать менее 64 байт информации в каждый блок, то вначале необходимо произвести чтение данных с карты из блока, в котором необходимо произвести модификацию данных, затем произвести модификацию данных в нужных ячейках и произвести запись блока на карту.

В смарт-контроллере используется семь команд приведенных в табл.5.2.

Команда 90h. Запись данных на карту без шифрации.

Кадр для записи данных на карту должен иметь длину 75 байт, из которых соответственно первые 8 байт – служебные, затем следуют 65 байт данных и 2 байта контрольной суммы. В поле данных нулевой байт определяет банк памяти, в который производится запись данных.

Таблица 5.2

Комманды контроллера

№ команды	Действие
90h	Запись данных без шифрации
91h	Чтение данных без дешифрации
92h	Стирание карты
93h	Запись данных с шифрацией
94h	Чтение данных с дешифрацией
95h	Формирование протокола для записи на карту
96h	Формирование протокола для чтения с карты
97h	Звуковой сигнал

Всего контроллером поддерживается четыре банка памяти с 0 по 3 длиной 64 байта. Если невозможно произвести запись на карту, то контроллер возвращает ответный кадр длиной 16 байт, а в нулевом байте содержится код ошибки (табл.5.2). В случае успешного завершения команды в ответном кадре, в нулевом байте поля данных содержится 0, при этом кадр имеет длину 75 байт и в поле данных содержит записанную на карту информацию.

Команда 91h. Чтение данных с карты без дешифрации.

Кадр для чтения данных с карты может иметь длину от 16 до 75 байт, из которых соответственно первые 8 байт – служебные, затем следует поле данных и 2 байта контрольной суммы. В поле данных нулевой байт определяет банк памяти, с которого производится чтение данных. Всего контроллером поддерживается четыре банка памяти с 0 по 3 длиной 64 байта. Если невозможно произвести чтение данных с карты, то контроллер возвращает ответный кадр длиной 16 байт, а в нулевом байте содержится код ошибки (табл.5.2). В случае успешного завершения команды в ответном кадре, в нулевом байте поля данных содержится 0, при этом

кадр имеет длину 75 байт и в поле данных содержит прочитанную с карты информацию.

Команда 92h. Стирание карты.

Данная команда производит стирание всей памяти на карте. После выполнения этой команды все ячейки памяти содержат FFh. Длина кадра может изменяться от 16 до 75 байт. Длина ответного кадра имеет такую же длину, как и кадр запроса. Если невозможно произвести чтение данных с карты, то в нулевом байте содержится код ошибки (табл.5.3). В случае успешного завершения команды в ответном кадре, в нулевом байте поля данных содержится 0.

Команда 93h. Запись данных на карту с шифрацией.

Эта команда полностью аналогична команде 90, но перед записью данных производится их шифрация по ГОСТ28147-89.

Команда 94h. Чтение данных с карты с дешифрацией.

Эта команда полностью аналогична команде 91, но перед передачей данных производится их дешифрация по ГОСТ28147-89.

Команда 95h. Формирование протокола для записи на карту**.**

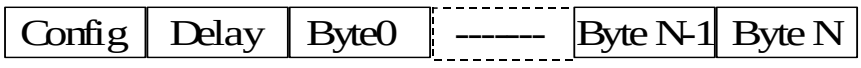
Команда 96h. Формирование протокола для чтения с карты.

Таблица 5.3

Коды состояний контроллера

Значение нулевого байта поля данных	Действие
00h	Команда успешно завершена
81h	Ошибка чтения EEPROM
82h	Ошибка записи EEPROM
83h	Ошибка чтения карты
84h	Ошибка записи карты
85h	Данные записаны не верно
86h	Карта отсутствует
87h	Карта вставлена
88h	Банк памяти не поддерживается

Данные команды предназначены для формирования любых протоколов обмена с картами, не поддерживаемыми контроллером. Формат данных, пересылаемых на карту следующий:



– config – содержит побитовую конфигурацию портов: 1 – вход, 0 – выход;
– delay – содержит задержку между передаваемыми на карту данными в микросекундах.

Byte0..ByteN – соответственно данные, которые необходимо переслать на карту. Каждый бит байта соответствует выводу порта. Длина кадра запроса может быть от 16 до 75 байт. При записи данных на карту ответный кадр в поле данных содержит записанную информацию, а при чтении соответственно прочитанную.

6. ПРОГРАММНАЯЙ ОБОЛОЧКА ДЛЯ РАБОТЫ С УНИВЕРСАЛЬНЫМ КОНТРОЛЛЕРОМ СМАРТ-КАРТ *CARD LAB*

6.1. Назначение и возможности программы

Программная оболочка (далее – программа) предназначена для работы только с универсальным контроллером смарт-карт *Card Lab*. Без него она может быть использована для просмотра и изменения файлов в характерном только для данной программы формате.

Программа позволяет:

- создавать, просматривать созданный ранее, записывать и читать из файла, а также редактировать дамп смарт-карты с открытой памятью в двоичном или шестнадцатеричном виде;
- читать дамп смарт-карты, изменять его и записывать на карту;
- устанавливать связь с контроллером, выполняя все необходимые действия в соответствии с интерфейсом ASK-BUS (V. 2.0);
- отображать все переданные контроллером данные о своих характеристиках и прочую служебную информацию;
- передавать контроллеру команды на генерацию звукового сигнала, стирание содержимого памяти карты в соответствии с интерфейсом ASK-BUS (V. 2.0);
- формировать запросный кадр ASK-BUS (V. 2.0), осуществлять информационный обмен с контроллером и отображать ответный кадр;
- просмотреть помощь по программе и макету, а также рекомендации по совместной работе макета и компьютера.

Для корректной работы программной оболочки необходимо подключение контроллера к компьютеру через исправный правильно распаянный кабель последовательного (прямое включение сигналов TXD, RXD) порта, а также правильное подключение питания.

Чтобы убедиться в исполнении данных требований:

- посмотрите, горит ли светодиод ‘Питание’ (крайний, зеленого свечения) на плате контроллера. Если нет, то необходимо проверить подключение шнура питания как к сети, так и к контроллеру;
- установите связь с контроллером из программы, выбрав пункт меню <Контроллер – Установить связь>. При этом выдача программой сообщения об ошибке при горящем светодиоде макета "Готов" (средний, зеленого свечения) будет свидетельствовать о неисправности интер-

фейсного соединения или о неправильной настройке последовательного порта.

6.2. Работа с программой

Запуск программы при работе под управлением или при эмуляции MS-DOS осуществляется набором строки ‘SMART-11.EXE’ с указанием пути к программе (если соответствующий каталог не является текущим) и нажатием <Enter>.

При запуске с пустой командной строкой устанавливаются следующие параметры последовательного интерфейса:

Номер COM-порта	2
Скорость	9600
Параметры пакета	1 старт, 8 инф., 1 стоп, без контроля

При необходимости настройки интерфейса на другие параметры необходимо задать командную строку в соответствии с шаблоном:

<кол-во инф. бит> <кол-во стоп-бит> <COM-порт> <контроль> <скорость>
(5,6,7,8) (1,2) (COM1, COM2) (NO,ODD,EVEN) (Любая из стандартного ряда).

Например, если контроллер подключен к первому COM-порту, необходимо набрать: SMART-11.EXE 8, 1, COM1, NO, 9600.

При первоначальной загрузке программы доступны следующие команды меню: <Загрузить дамп>, <Установить связь> и команда на получение помощи. Командой <Загрузить дамп> можно просмотреть и изменить дамп памяти смарт-карты, содержащийся в ранее сохраненном файле текстового формата с расширением .dmp .

Для начала работы с контроллером необходимо убедиться, что соответствующий порт компьютера настроен на работу с внешней периферией.

Универсальный контроллер смарт-карт *Card Lab* представляет собой интеллектуальное периферийное устройство с обменом по последовательному интерфейсу RS-232C. Протоколом, по которому обмениваются компьютер и контроллер, является ASK-BUS (v. 2.0). Данный протокол является внутренним стандартом ASK Lab DC и разработан для обмена управляющего ПК с периферийными устройствами, объединенными в сеть ASK LAN или единичными, с целью сбора информации с интеллектуальных датчиков, управления удаленными объектами и т.д. Начало работы с устройствами в соответствии с ASK-BUS (V. 2.0) состоит из следующих действий управляющего ПК.

1. Обнулить(сбросить) адреса всех узлов сети.

2. Запросить в сети устройство, не имеющее сетевого адреса. Если не один из узлов не ответил, то работа по распределению сетевых адресов закончена. При этом отвечает первый узел без адреса по пути прохождения кадре. В ответном содержится идентификационный номер данного устройства и некоторые прочие данные о нем.

3.Присвоить ответившему узлу сетевой адрес. Адрес состоит из пяти байт. Протокол ASK-BUS (V. 2.0) предполагает использование в структуре сети иерархию "колец". Максимальное количество уровней - 5. Таким образом, каждый байт адресует узел внутри соответствующего кольца.

После распределения адресов управляющий ПК может производить обмен с любым из узлов сети.

Вся описанная последовательность действий выполняется программой при выборе пункта меню <Установить связь>.

Настоящая версия программы (1.X) позволяет работать только с одним контроллером смарт-карт (при выполнении контроллеров в виде сетевых узлов возможна работа со многими такими устройствами, объединенными в сеть).

После присвоения адреса контроллеру программа выдает в специальном окне все известные данные о контроллере.

При этом становятся доступными следующие пункты меню:

– <Загрузить дамп>;
– <Установить связь>;
– <Звуковой сигнал>;
– <Информационный обмен>;
– <Параметры>;
– <Читать дамп без расшифровки>;
– <Читать дамп с расшифровкой>;
– <Стереть информацию>,

а также команды на выдачу помощи.

Пункты меню:

– <Сохранить дамп>;
– <Сохранить как...>;
– <Запись без шифровки>;
– <Запись с шифровкой>,

становятся доступными при открытом окне шестнадцатеричного дампа или открытом окне бинарного дампа.

6.3. Описание меню программной оболочки

Пункт меню <Загрузить дамп>

Позволяет выбрать с любого диска и из любого каталога файл с дампом для чтения (расширение по молчанию .dmp). После выбора имени файла нажмите кнопку <Открыть> или клавишу <Enter>. Прочитанный дамп (при условии целостности файла) будет отображен в окне шестнадцатеричного дампа или в окне бинарного дампа – в зависимости от того, в каком виде последний раз отображался дамп.

Ниже описано содержимое и действия с окнами шестнадцатеричного и бинарного дампа.

Окно шестнадцатеричного дампа. Данное окно предназначено для отображения прочитанного из файла или со смарт-карты дампа памяти в

шестнадцатеричном виде в строках по 16 байт. В каждый момент может быть открыто не более одного окна шестнадцатеричного дампа или окна бинарного дампа.

Для просмотра всего дампа используйте курсорные клавиши, а также <Home>,<End>,<PageUp>,<PageDown>.

В окне также возможно изменение дампа. Для этого, установив подсвеченную строку на необходимую позицию нажмите клавишу <Ins>. После этого можно изменить содержимое первого байта дампа в текущей строке.

Для перехода к другим байтам используйте курсорные клавиши <Влево>, <Вправо>, а также клавиши <Tab>, <Enter>. Переход к другой строке осуществляется курсорными клавишами и клавишами <Home>, <End>, <PageUp>, <PageDown>, после чего необходимо снова нажать клавишу <Ins>.

Измененный дамп можно сохранить в файле или записать на карту. Для этого необходимо из основного меню выбрать соответствующий пункт.

По окончанию работы с дампом для закрытия окна нажмите кнопку <Закрыть>.

Окно бинарного дампа. Данное окно предназначено для отображения прочитанного из файла или со смарт-карты дампа памяти в двоичном виде в виде списка двоичных строк.

Для просмотра всего дампа используйте курсорные клавиши, а также <Home>, <End>, <PageUp>, <PageDown>.

Кроме того, в этом окне возможно изменение дампа. Для этого, установив подсвеченную строку на необходимую позицию нажмите клавишу <Ins>. После этого можно изменить любое количество двоичных разрядов выбранного байта. Для перехода к другой строке используйте курсорные клавиши и клавиши <Home>, <End>, <PageUp>, <PageDown>, с последующим нажатием клавиши <Ins>.

Измененный дамп можно сохранить в файле или записать на карту. Для этого необходимо из основного меню выбрать соответствующий пункт.

По окончанию работы с дампом для закрытия окна нажмите кнопку <Закрыть>.

Пункт меню <Установить связь>

По выбору этого пункта выполняется вся последовательность действий по назначению подключенному контроллеру смарт-карт сетевого адреса. Настоящая версия программы (1.X) позволяет работать только с одним контроллером смарт-карт (при выполнении контроллеров в виде сетевых узлов возможна работа со многими такими устройствами, объединенными в сеть).

После присвоения адреса контроллеру программа выдает в специальном окне все известные данные о контроллере.

Пункт меню <Звуковой сигнал>

Выбор данного пункта меню приводит к передаче команды подключенному контроллеру на генерацию кратковременного звукового сигнала. Если информационный обмен проходит успешно, то никаких сообщений не выдается.

Пункт меню <Информационный обмен>

При выборе этого пункта меню разворачивается окно информационного обмена, которое предназначено для ввода полей запросного кадра, направляемого контроллеру, и отображения ответного кадра, полученного от контроллера.

В окне с помощью строк ввода отображена структура запросного и ответного кадров в соответствии с внутренним стандартом ASK Lab DC ASK-BUS (V. 2.0). Не показаны только контрольная сумма кадров, которая представляет собой простую сумму по модулю 2^{16} и для запросного кадра формируется автоматически перед передачей, а для ответного – подсчитывается и проверяется при приеме. Перемещаться между строками ввода можно кл. <Tab> или с использованием манипулятора "мышь". Рекомендуется при изменении полей запросного кадра использовать режим текстовой вставки, который включается нажатием клавиши <Insert>. Это упрощает процесс редактирования содержимого длинных полей.

При задании необходимого содержимого полей нажмите кнопку <Передать>. Программа сформирует запросный кадр и передаст его в требуемом формате контроллеру.

Если после этого контроллер отвечает также передачей кадра, то содержимое полей этого кадра отображается в нижней части окна, а пользователю сообщается о приеме ответной информации. Если же контроллер в течение примерно секунды не отвечает, то фиксируется ошибка истечения времени ожидания ответа, о чем сообщается в специальном окне.

Пункт меню <Параметры>

Отображает все известные параметры контроллера смарт-карт, с которым последний раз было установлено соединение.

Пункт меню <Читать дамп без расшифровки>

При выборе этого пункта меню у пользователя запрашивается тип карты, которая помещена в считыватель контроллера, а также система счисления, в которой следует отображать прочитанный дамп. После получения данных параметров контроллеру передаются команды на чтение содержимого карты без расшифровки по ГОСТ 28147-89 и его передачи управляющему ПК.

Перед выбором этого пункта необходимо поместить смарт-карту в считыватель!

Пункт меню <Читать дамп с расшифровкой>

При выборе этого пункта меню у пользователя запрашивается тип карты, которая помещена в считыватель контроллера, а также система счисления, в которой следует отображать прочитанный дамп. После получения данных параметров контроллеру передаются команды на чтение содержимого карты с расшифровкой по ГОСТ 28147-89 при использовании ключа и таблицы замен, содержащихся в энергонезависимой памяти программ контроллера, и передачи полученных данных управляющему ПК.

Пункт меню <Стереть информацию>

Выбор этого пункта меню вызывает передачу команду контроллеру на заполнение энергонезависимой памяти смарт-карты фиксированным значением (FFh). О результатах выполнения команды сообщается в специальном окне.

Пункт меню <Сохранить дамп>

Данный пункт меню необходимо использовать в случаях, когда текущий дамп понадобится для дальнейшей работы или сравнения с последующими изменениями. Сохранению в файле подвергается дамп, отображенный в настоящее время в окне шестнадцатеричного дампа или окне бинарного дампа. Если ранее файл не сохранялся и не был прочитан из файла, то у пользователя запрашивается имя файла, в который будет записан дамп.

Дамп сохраняется в текстовом виде, удобном для просмотра из любого текстового редактора под DOS.

Причем, если необходимо изменить содержимое дампа для дальнейшего использования в программе, то это также можно сделать из любого редактора, а затем открыть из программы командой <Загрузить дамп>.

Внимание! При изменении самого дампа нельзя изменять другие строки программы, а также дописывать недопустимые символы в строки с дампом.

(Допустимые символы: 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F)

Пункт меню <Сохранить как...>

Позволяет сохранить дамп под заданным, отличным от текущего именем. (См. описание пункта<Сохранить дамп>).

Пункт меню <Запись без шифровки>

Выбор данного пункта меню вызывает запись отображенного в настоящее время дампа в окне шестнадцатеричного дампа или в окне бинарного дампа с помощью контроллера на смарт-карту. Запись производится без какого-либо преобразования информации.

Перед выбором этого пункта необходимо поместить смарт-карту в считыватель!

Пункт меню <Запись с шифровкой>

Выбор данного пункта меню вызывает запись отображенного в настоящее время дампа в окне шестнадцатеричного дампа или в окне бинарного дампа с помощью контроллера на смарт-карту. Запись производится с шифрованием по ГОСТ 18147-89 при использовании ключа и таблицы замен, хранящихся в энергонезависимой памяти программ контроллера.

Перед выбором этого пункта необходимо поместить смарт-карту в считыватель!

При необходимости окончить работу с программой, если открыто окно шестнадцатеричного дампа или окно бинарного дампа, убедитесь в том, что изменения дампа сохранены в файле или на карте и закройте окно. После этого можно выйти из программы нажатием сочетания клавиша <Alt-X> или выбором пункта меню <Файл - Выход>.

7. КОЛЬЦЕВАЯ СТРУКТУРА МИКРОКОНТРОЛЛЕРНОЙ СЕТИ ASK-LAN1.2

Для организации взаимодействия множества терминальных устройств и управляющего устройства сети разработана спецификация кольцевой сети ASK-LAN 1.2. Эта сеть обеспечивает полудуплексный обмен между участниками. В качестве базового физического канала используется RS-485 на "витой паре". Протокол обмена – ASK-BUS (v. 2.0).

7.1. Топология сети

Сеть представлена на рис.7.1. Она состоит из следующих компонентов:

– управляющее устройство (например, IBM PC с сетевой картой ASK-NC1, вставленной в свободный слот расширения);

– контроллер сети (КС) – устройство, включенное в разрыв любого кольца сети любого уровня и поддерживающее установленный протокол обмена;

– периферийное устройство (ПУ) – терминал пластиковых карт или любое другое цифровое или аналоговое устройство с любым интерфейсом, подключенное к контроллеру сети;

– узел сети – контроллер сети со всеми подключенными к нему периферийными устройствами.

Физическая топология сети – двунаправленное кольцо, логическая топология – звезда. Несмотря на кольцевую структуру, выход из строя одного узла или нарушение линии связи в одном месте не влечет за собой потерю управления всеми ПУ за счет наличия "запасного" выхода в сеть через порт B.

Контроллеры сети обеспечивают двунаправленный информационный

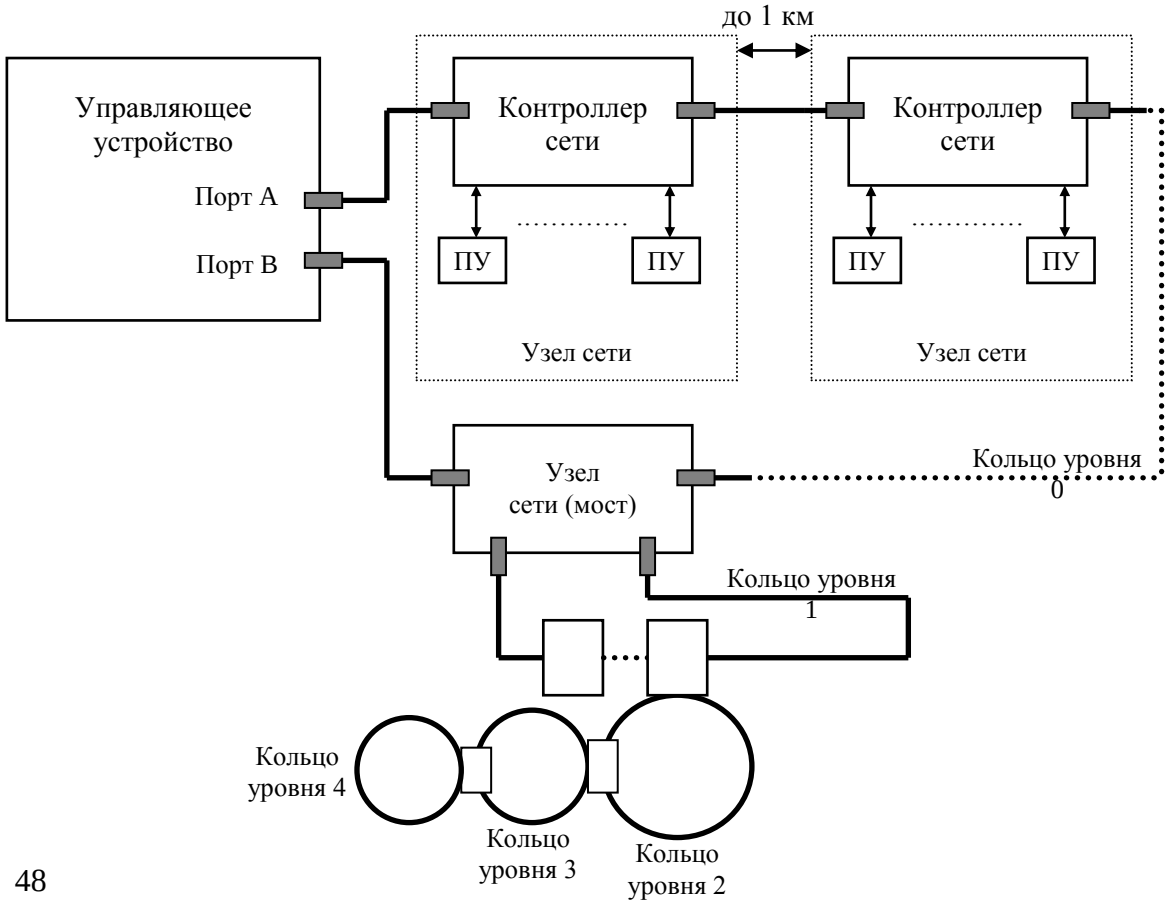


Рис.7.1. Топология сети ASK-LAN1

обмен между управляющим устройством и большим числом разнообразных ПУ по кольцевой линии связи. КС могут быть разнесены друг от друга на значительные расстояния (до ~1 км) и состоять из нескольких разнородных ПУ (в зависимости от типа КС). Возможно подключение к КС кроме ПУ сети следующего уровня и т.д. до пяти уровней (в этом случае узел носит название моста).

Внутренняя конструкция узла сети в самом сложном случае (мост с поддержкой нескольких ПУ) представлена на рис.7.2. Он состоит из следующих элементов:

– устройство отключения узла от сети осуществляет замыкание кольца сети и отключение внутренних приемопередатчиков от сети при пропадании питания на узле или при нажатии специальной кнопки исключения узла из сети;

– приемопередатчики RS485 обеспечивают преобразование дифференциального сигнала в линии в цифровой сигнал 0 – 5В и наоборот;

– ретранслятор кадров обеспечивает прохождение сетевых кадров сквозь узел в обоих направлениях, коррекцию временных параметров кадров, искажающихся за время прохождения от узла к узлу по длинной линии, ответвление проходящих кадров к микроконтроллеру, передачу ответных кадров микроконтроллера в сеть и защиту кадров от наложения друг на друга;

– микроконтроллер обеспечивает поддержку установленного протокола обмена, обмен данными с ПУ и управление кольцом сети нижестоящего уровня.

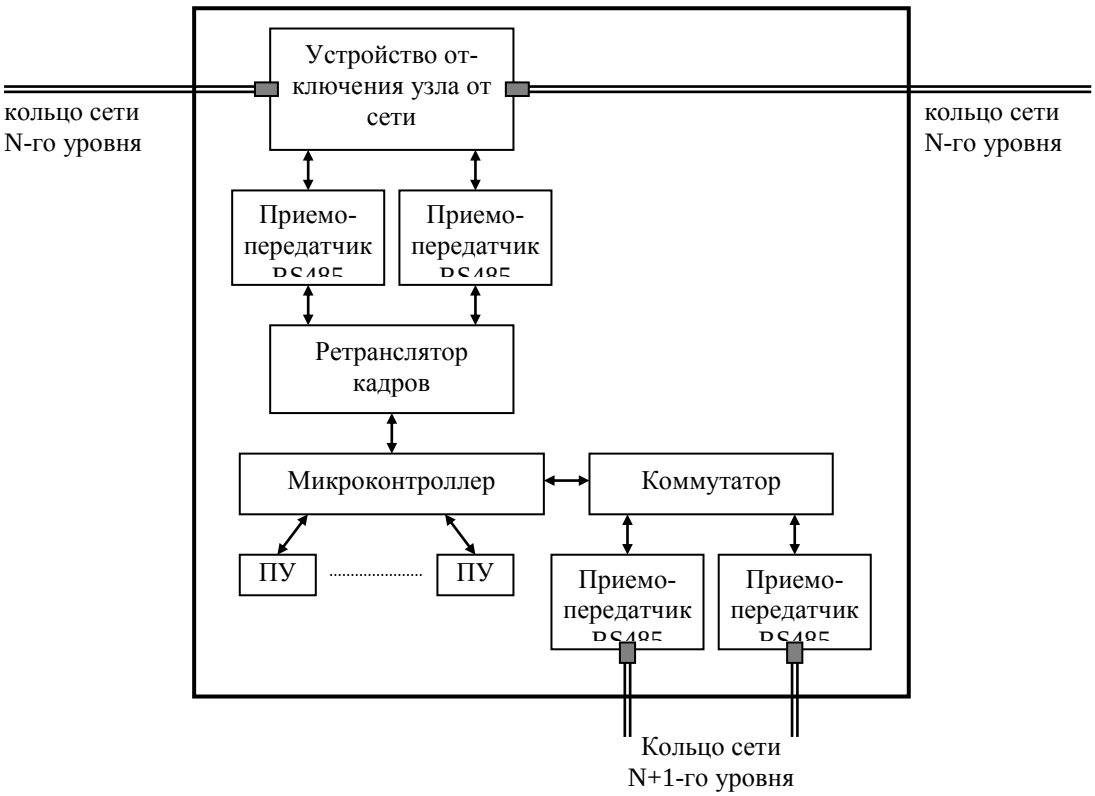


Рис.7.2. Структурная схема узла сети

В приведенной схеме физические интерфейсы и топологии вышестоящей и нижестоящей сетей совпадали и имели кольцевую структуру. Возможен, однако, такой случай, когда применение кольцевой структуры нецелесообразно из экономических, энергетических или каких-либо других соображений, тогда допускается применение другой физической топологии, допускающей логическую топологию звезда (например, можно использовать шину RS-485). В этом случае из структурной схемы узла убирается один приемопередатчик, ретранслятор и, возможно, устройство отключения узла от сети.

Аналогично может возникнуть необходимость в изменении физического канала, например сеть нижестоящего уровня должна работать на RS-422 или ИК-канале. Такое также вполне допускается, однако в некоторых случаях число узлов в сети этого уровня не сможет быть больше 1 (например, интерфейс RS-232).

На рис.7.3 в качестве примера приведена возможная структура 3-х уровневой сети.

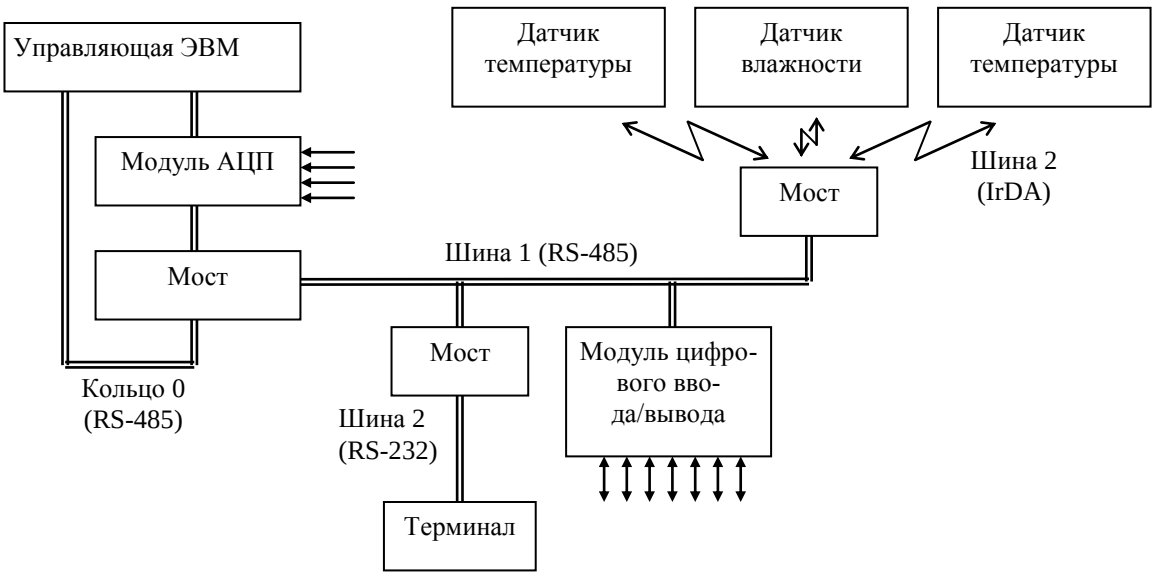


Рис.7.3. Пример построения распределенной системы с разнотипными интерфейсами в рамках сети ASK-LAN1

7.2. Организация обмена информацией

Весь процесс обмена информацией базируется на следующих принципах:

– инициатором любого обмена в сети любого уровня всегда является мастер-устройство этой сети, т.е. управляющее устройство сети (для сети 0-го уровня) или мост (для сетей 1-4 уровней);

– любой единичный обмен информацией состоит из двух транзакций:
1 – передача запроса от мастер-устройства к какому-либо узлу сети;
2 – передача ответа от узла к мастер-устройству.

При работе сети в кольцевом режиме обмен инициируется через один порт мастер-устройства, запрос распространяется последовательно через все узлы и приходит на другой порт. Ответ распространяется от отвечающего узла в обоих направлениях и приходит на оба порта мастер-устройства.

При нарушении кольцевой связи (например, при выходе из строя узла) мастер-устройство выдает информационный кадр в сеть через тот порт, который остался связан с адресуемым узлом. Ответный кадр в этом случае приходит на этот же порт. Если нарушение кольцевой связи произошло из-за неисправности ветви линии связи, то система сможет продолжать работу почти без потери качества обмена. Если же вышел из строя узел, то связь с ПУ, подключенными к нему будет утеряна.

В целях повышения надежности системы возможно использование дистанционного управления сигналом аппаратного сброса узлов. Это осуществляется следующим образом.

В нормальном состоянии при отсутствии обменов мастер-устройство постоянно передает по сети одинаковые байты, не несущие никакой информационной нагрузки. КС узлов детектируют наличие сигнала в сети и удерживают линию аппаратного сброса в неактивном состоянии (в кольцевой сети эту функцию выполняет ретранслятор). При необходимости аппаратного сброса мастер-устройство прекращает выдачу каких-либо байтов в сеть в течение ~5с. Логика КС построена так, что каждый приходящий по сети байт вызывает сброс таймера с временем ожидания ~5с. При отсутствии байтов в течение этого времени таймер активизирует сигнал аппаратного сброса узла.

7.3. Идентификация узлов в сети

Сеть имеет иерархическую структуру с количеством уровней до 5. Число узлов на каждом уровне до 255. Каждый узел "знает" свой номер в сети, к которой он подключен. Адрес любого узла в сети (сетевой адрес) состоит из пяти чисел в диапазоне 1…255. Для определенности будем записывать сетевой адрес узла в виде \\L0\L1\L2\L3\L4, где L0 – номер узла на самом верхнем уровне (1..255), L4 (1..255) – номер узла внутри последнего уровня.

Адрес 0 на каждом уровне зарезервирован для обмена с мастер-устройством этого уровня.Кроме сетевого адреса, с каждым узлом связан тип и серийный номер, которые присваиваются узлу на стадии производства.

7.4. Инициализация сети (присвоение сетевых адресов)

Перед началом работы с сетью после включения питания необходимо произвести нумерацию всех узлов в сетях всех уровней, т.е. присвоить им сетевые адреса в диапазоне 1–255. Для идентификации узлов на этой стадии

используются их серийные номера. Присвоение сетевого адреса осуществляется посылкой специального кадра, содержащего серийный номер узла.

При работе сети в кольцевом режиме за счет наличия ретрансляторов в узлах имеется возможность узнать серийные адреса узлов, которым еще не присвоен адрес, посылкой специального безадресного кадра. В ходе одного обмена всегда будет получен один из серийных номеров, так как ретрансляторы не допустят наложение нескольких ответов.

При необходимости можно аннулировать сетевые адреса всех узлов в сети любого уровня посылкой специального безадресного кадра.

7.5. Протокол обмена между мастер–устройством и узлами сети (версия 2.0)

В этом подразделе описываются типы и форматы кадров, а также команды используемые для обмена между управляющим устройством (PC) и терминалом пластиковых карт.

Форматы сетевых кадров

Обмен может происходить как кадрами фиксированной длины 64 байта, так и кадрами переменной длины 16–255 байтов. Формат кадра схематично можно изобразить следующим образом.

FT/FL	NA/SN	Cmd	S	Data	CS

Ниже приводится расшифровка полей кадра.

Frame type/Frame length (FT/FL) – 1 байт

Данное поле определяет назначение кадра и его длину. Значения 00h – 0Fh означают длину кадра 64 байта и полностью соответствуют версии 1 протокола:

00h – "пустой кадр", не обрабатывается ни одним узлом. Может использоваться для заполнения пауз между информационными обменами, в режиме отладки и т.п.

01h – сброс адреса узла, обрабатывается всеми узлами независимо от содержания остальных полей и приводит к аннулированию текущего адреса узла

02h – запрос идентификационного номера узла с аннулированным адресом

03h – присвоение адреса узлу с указанным идентификационным номером

04h – информационный запрос мастер-устройства

05h – ответ узла

06h – 0Fh – зарезервированы.

Если поле FT/FL содержит число в диапазоне 10h – FFh, то это число равно длине передаваемого кадра в байтах, включая контрольную сумму.

Serial number / Net address (SN/NA) – 5 байтов

Данное поле содержит сетевой адрес узла с которым идет обмен или серийный номер узла на стадии инициализации сети.

NA = |L0|L1|L2|L3|L4| - сетевой адрес;

SN = |NT|SN0|SN1|SN2|SN3|, где NT – тип узла, SN0-SN3 – 4-х байтовый серийный номер узла.

В настоящее время использованы следующие типы узлов:

01h – сетевая плата ISA ASK-NC1;

02h – КС с SPI-интерфейсом для подключения модуля АЦП;

03h – узел сбора данных с квартирных счетчиков;

04h – светодиодное информационное табло.

Command (Cmd) – 1 байт

Номер команды узла. Значения 00h-3Fh зарезервированы для однотипного использования всеми узлами. Подробное описание команд дается в документации конкретных узлов.

Status (S) – 1 байт

Содержит статус выполнения команды. Значения статуса 00h–3Fh зарезервированы для однотипного использования всеми узлами:

00h – команда выполнена успешно;

01h – идет передача запроса в нижестоящем уровне сети;

02h – идет прием ответа в нижестоящем уровне сети;

03h – ожидается ответ в нижестоящем уровне сети;

04h – ожидается готовность нижестоящего уровня сети к обмену;

05h – нет выполняемых команд;

06h – команда принята мостом;

07h – запрошенная команда не поддерживается КС;

08h – кадр содержит длину передаваемого файла;

09h – кадр содержит смещение запрашиваемого блока файла;

0Ah – кадр содержит запрошенный блок файла;

0Bh – кадр содержит длину передаваемого файла и смещение запрашиваемого блока файла;

0Ch – кадр содержит запрошенный блок файла и смещение запрашиваемого блока файла;

0Eh – ошибка при обмене файлами;

В отличие от версии 1 при получении кадра узел производит дешифрацию поля статуса при операциях с файлами.

Значения 40h–FFh отведены для произвольного использования узлами. Подробное описание статуса дается в документации конкретных типов узлов.

Data – 6-245 байтов

Здесь содержится информация, пересылаемая узлу или получаемая от узла. Подробное описание пересылаемых данных см. в описании узлов. Байты данных ответа узла, не имеющие значения, должны оставаться такими же, как в принятом запросе либо равны 0.

Check sum (CS) – 2 байта

Контрольная сумма кадра. Используется для проверки правильности принятого кадра и содержит сумму предыдущих байтов.

Типы кадров

Тип 0 – "пустой" кадр

FT = 0, SN/NA =XXXXX, Cmd = X, S = X, Data = X. Кадр не требует никакой обработки узлом.

Тип 1 – сброс адреса узла

FT = 1, SN/NA =XXXXX, Cmd = X, S = X, Data = X. При получении такого кадра узел должен аннулировать свой сетевой адрес.

Тип 2 – запрос идентификационного номера узла с аннулированным адресом

FT = 2, SN/NA =XXXXX, Cmd = X, S=X, Data = X. При получении такого кадра узел, не имеющий сетевого адреса должен ответить кадром следующего вида:

FT = 5, SN/NA = X, Cmd = X, S = 0, Data = NT|SN0|SN1|SN2|SN3|PV|Dmax|Tmax0|Tmax1, где NT – тип узла, SN0-SN3 – 4-х байтовый серийный номер узла, PV – версия протокола (2 для данного описания), Dmax – максимальный размер поля Data, Tmax – 2-х байтовое число, определяющее максимальное время обработки команды узлом (в миллисекундах).

Тип 3 – присвоение адреса узлу

FT = 3, SN/NA = NT|SN0|SN1|SN2|SN3, Cmd = X, S = X, Data = NA. При получении такого кадра узел должен проверить равенство поля SN своему и при совпадении присвоить себе сетевой адрес NA. Ответный кадр может использовать поле FL и должен содержать S = 0.

Тип 4 – информационный запрос мастер-устройства

FT = 4, SN/NA = NA, Cmd = ?, S= ?, Data = ?. Это обычный кадр мастер-устройства, инициирующий обмен с узлом, использующийся в версии 1 протокола. В версии 2 целесообразнее для этого пользоваться полем FL вместо FT = 4.

Общие команды узлов

Команда 0 – Инициализация

Может использоваться для приведения узла в исходное состояние (более "мягкий" вариант аппаратного сброса).

Команда 1 – Самотестирование

По этой команде узел осуществляет тестирование внутренних цепей и возвращает результат самотестирования. Отсутствие ошибки сигнализирует нулевое значение поля статуса. Детальную информацию о результатах тестирования узел может сообщить в поле Data, ее дешифрирование осуществляется в соответствии с документацией на узел.

Команда 2 – Получить тип и серийный номер КС

В версии 1 по этой команде узел возвращает байт типа и 4-х байтовый серийный номер в поле Data. В настоящей версии узел сообщает следующую информацию:

00h – тип узла (1байт);
01h – серийный номер узла (4байта);
05h – номер версии протокола, поддерживаемой узлом (1байт);
06h – максимальная длина поля Data, поддерживаемая узлом (1байт);
07h – максимальное время обработки команды узлом, мс (2байта).

Команда 3 – Получить результат выполнения последней команды

Данная команда используется для проверки готовности данных в том случае, если в ответ на запрос был возвращен статус 06h. Значение статуса 00h свидетельствует о том, что команда успешно завершена и поле Data действительно.

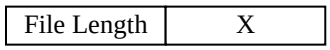
Команда 4 – Получить идентификационную строку КС

По этой команде узел возвращает идентификационную строку в ASCIIZ формате. В настоящее время строка имеет вид: ASK Lab Design Center PT=XXX VN=XXX SN=XXXXXXXXXX, где PT (product type) соответствует типу узла, VN (version number) содержит номер версии узла, SN (serial number) соответствует серийному номеру узла.

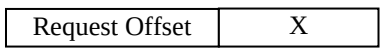
Форматы кадров обмена файлами

Для управления процессом обмена файлами используется поле S кадра. Оно определяет формат поля Data кадра.

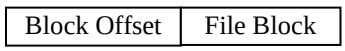
S = 08h – кадр содержит длину передаваемого файла

Data = 

S = 09h – кадр содержит смещение запрашиваемого блока файла

Data = 

S = 0Ah – кадр содержит запрошенный блок файла

Data = 

S = 0Bh – кадр содержит длину передаваемого файла и смещение запрашиваемого блока файла

Data =

File Length	Request Offset	X

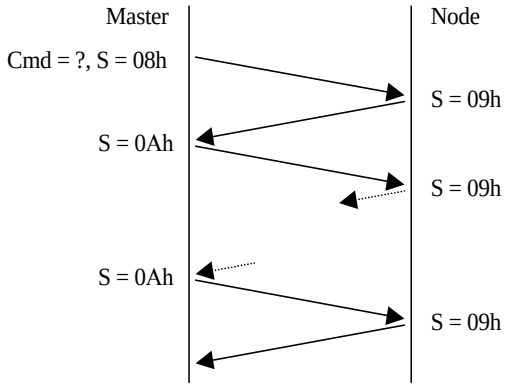
S = 0Ch – кадр содержит запрошенный блок файла и смещение запрашиваемого блока файла

Data =

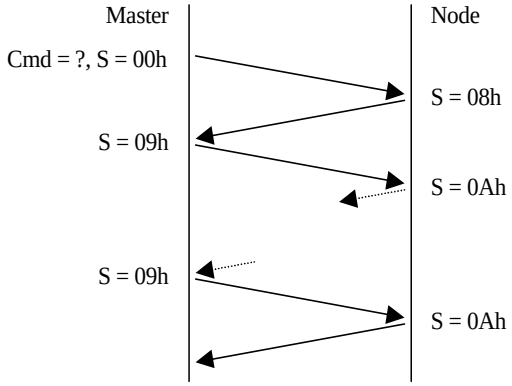
Request Offset	Block Offset	File Block

S = 0Eh – ошибка при обмене файлами
Означает, что нарушен ход обмена файлами, либо параметры файла не устраивают приемную сторону, либо какую-то другую причину, по которой обмен файлами не может быть выполнен.

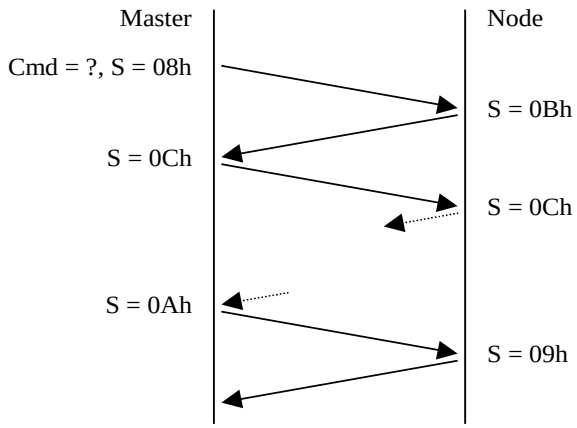
Передача файла от мастер–устройства к узлу



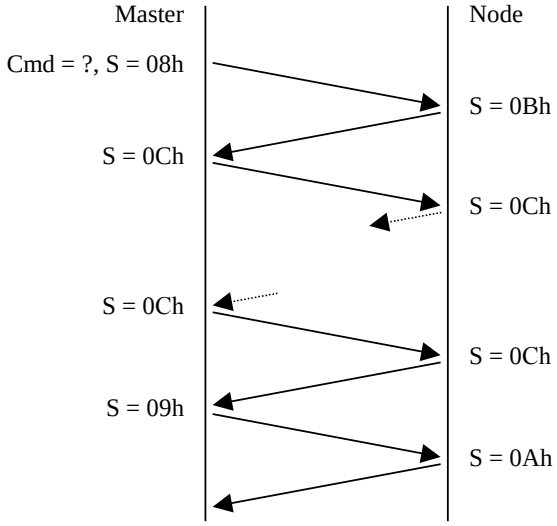
Передача файла от узла к мастер–устройству



Двунаправленная передача файлов (файл от мастера длиннее файла от узла)



Двунаправленная передача файлов(файл от мастера короче файла от узла)



8. ОПИСАНИЕ ЗАДАНИЙ

Процедуре обмена данными с пластиковой картой GFM-2K в режимах простой записи и считывания и с шифрацией по ГОСТ 28147-89 должен предшествовать ряд действий.

Сначала необходимо настроить (инициализировать) COM-порт, аннулировать сетевой адрес терминала, присвоить новый адрес и только после этого можно начинать обмен.

Рассмотрим простой пример. Пусть необходимо вызвать генерацию звука терминалом. Поскольку к вырожденной сети подключен лишь один терминал через интерфейс RS-232C, нет необходимости устанавливать связь с другими терминальными устройствами. В этом случае блок-схема соответствующего алгоритма может выглядеть так, как это представлено на рис.8.1.

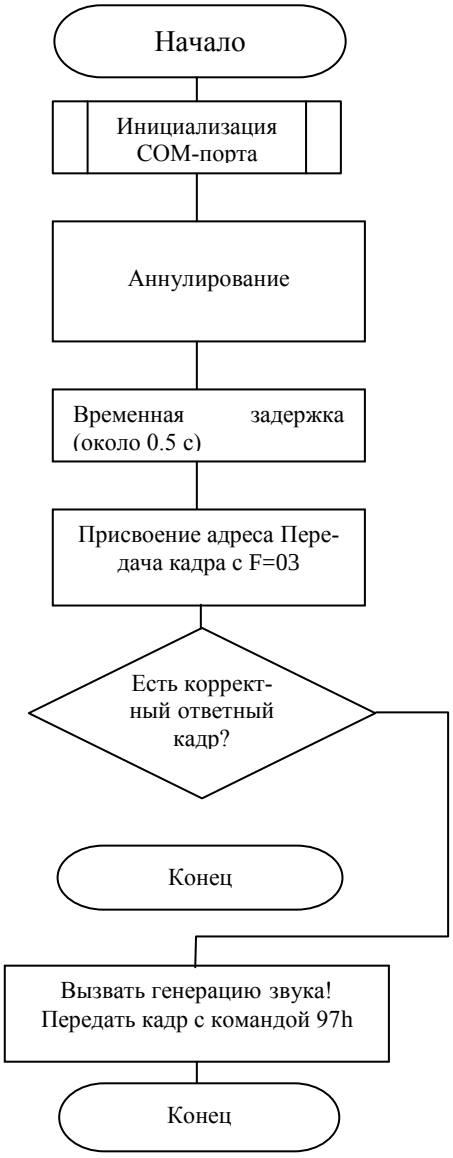


Рис.8.1. Схема алгоритма генерации звука

Задержка необходима, чтобы терминальное устройство успело осуществить обмен с картой и выполнить предписанные действия. При получении адреса FT=03h терминалу присваивается нулевой адрес.

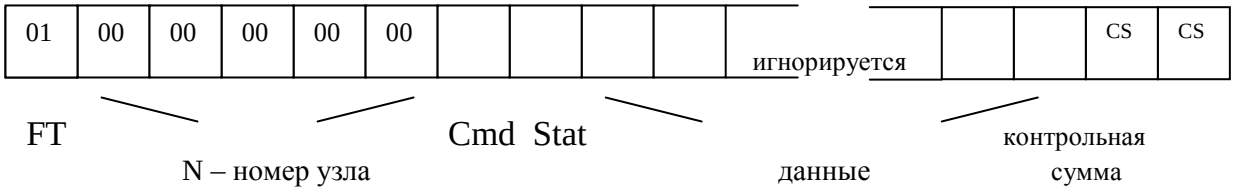
Форматы кадров аннулирования адреса, присвоения адреса и генерации звука представлены на рис.8.2.

Поскольку все три кадра запросные, байт статуса (Stat) игнорируется, байты данных также игнорируются, кроме того, байт команды в первых двух кадрах не значим, поэтому также игнорируется.

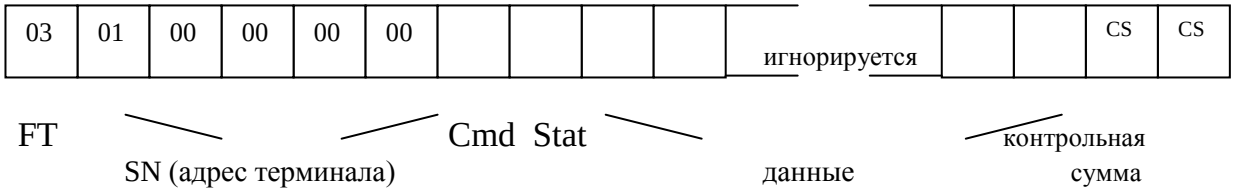
Процедура инициализации COM-порта может быть написана с помощью сервиса BIOS (раздел 2) с использованием прерывания. Это является обязательным требованием при создании коммерческих программ,

поскольку обеспечивает их совместимость. В то же время, использование BIOS может оказаться неудобным если, например, терминал не отвечает на запрос должным образом.

Аннулирование адреса



Присвоение адреса



Генерация звука

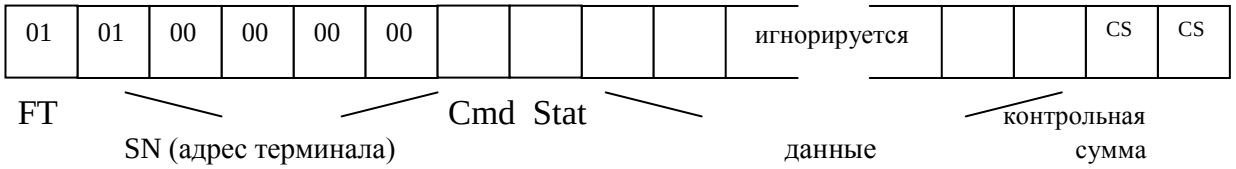


Рис.8.2. Форматы кадров аннулирования адреса, присвоения адреса и генерации звука

В разделе 9 приведена процедура инициализации COM-порта (Function COMInit) написанная на более низком уровне с использованием команды процессора OUT.

Верхнюю часть блок-схемы алгоритма рис.8.1, осуществляющего аннулирование и присвоение нового сетевого адреса терминалу можно условно назвать "установлением связи с терминалом". Практическая работа с картой GFM-2K предусматривает операции записи данных на карту и их чтение.

Рассмотрим пример реализации этой задачи для случая, когда шифрация не производится. Под данными будем понимать строку символов. ЗАПИСЬ строки будем производить в нулевой банк смарт-карты, причем первый байт записываемой последовательности будет указывать на длину строки.

Для записи на смарт-карту контроллеру должен быть передан информационный кадр со следующим значением полей.

Таблица 8.1

Значение полей информационного кадра

FL=4Bh	В поле типа указана длина кадра
SN=0000000001h	Сетевой адрес устройства
Cmd=90h	Команда записи на смарт-карту без шифрации
Data[x]=00h	Нулевой банк смарт-карты
Data[1...64]	Записываемые в банк данные

Формат кадра записи приведен на рис.8.3.

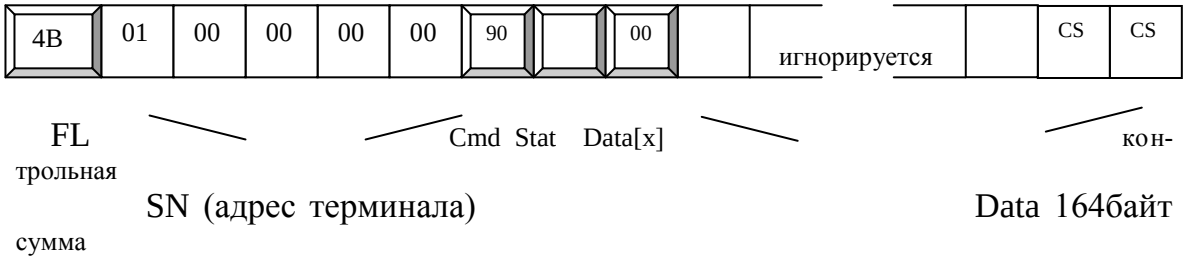


Рис.8.3. Формат кадра данных

Этот кадр принуждает терминал записать на карту 64 байта данных, содержащихся в поле Data. Запись производится в нулевой банк карты. Общая длина кадра 4Bh=75 байт. Служебная информация занимает 75-64=11 байт. Байт статуса игнорируется, контрольная сумма располагается в последнем и предпоследнем байтах кадра и представляет собой простую сумму по модулю 2^{16} всех остальных байтов. Если запись данных произошла успешно, то терминал возвращает кадр, в младшем байте поля данных которого будет содержаться нулевое значение, а остальные 64 байта будут содержать записанную на карту информацию (ретрансляция кадра).

Если терминалу по каким-либо причинам не удалось произвести корректную запись, то он отвечает кадром длиной 16 байт, в младшем байте поля данных которого будет содержаться под ошибки (табл.5.3).

ЧТЕНИЕ данных осуществляется передачей в терминал запросного кадра длиной от 16 до 75 байт. Формат кадра такой же, как и для записи. Младший байт поля данных определяет банк памяти карты, с которого производится чтение. Если терминалу не удается корректно прочесть данные из банка, он возвращает кадр длиной в 16 байт, в младшем байте поля данных которого содержатся под ошибки. Если команда чтения выполнена успешно, терминал отвечает кадром длиной 75 байт, формат которого полностью совпадает с кадром записи. Его поле данных содержит прочитанный банк памяти, а младший его разряд – код успешного выполнения, т.е. 00h.

Для записи строки на смарт-карту и чтения с нее необходима программа, которая выполняла бы действия в соответствии с алгоритмом, блок-схема которого представлена на рис.8.4.

Функция ввода строки символов отображает набираемую на клавиатуре строку на экран монитора. Для передачи этой строки в терминальное устройство, ее сначала необходимо преобразовать в массив чисел размером 64 байта, что соответствует размеру одного банка памяти смарт-карты. Пример программы осуществляющей запись массива в нулевой банк смарт-карты приведен в разделе 9. Эта программа оформлена в виде функции Send-Frame. В ее задачу входит формирование кадра заданной длины, подсчет контрольной суммы и передача кадра с контролем опорожнения сдвигового регистра.

Пример программы чтения нулевого банка памяти также приведен в разделе 9 в виде функции Receive-Frame. Эта функция ожидает кадр, принимает его и проверяет контрольную сумму. Прочитанный банк преобразуется в строку символов и, затем, выводится на экран монитора.

Возможность эффективного шифрования информации на основе стандарта ГОСТ 28147-89 позволяет использовать карты с открытой памятью GFM-2K для предотвращения несанкционированного доступа, и, в частности, для создания электронного паспорта. Рассмотрим эту задачу полагая, что паспорт должен содержать поля, представленные в табл.8.2.

Таблица 8.2

Поля электронного паспорта

№ поля	Наименование поля	Объем, байт
1	Номер электронного паспорта	4
2	Ф.И.О. владельца	60
3	Дата и место рождения	64
4	Место жительства	64
5	Дополнительная информация	64

Как следует из табл.8.2., запись необходимо производить во все банки смарт-карты в четыре приема по одному банку за одну запись. Поскольку при записи служебная информация занимает 11 байт (рис. 7.3), то для записи кадра длиной 64 байт требуется 11+64=75 байт (4Bh). При этом целесообразно использовать кадры, параметры которых приведены в табл.8.3.

Рис.8.4. Схема алгоритма записи и чтения строк со смарт-карты

Таблица 8.3

Параметры для записи кадров длиной 64 байт

FT=4Bh	В поле типа указана длина кадра
SN=0000000001h	Сетевой адрес устройства
Cmd=93h	Команда записи на карту с шифрацией
Data[x]=0Xh	Банк смарт-карты (X – от 0 до 3)
Data[1...64]	Записываемые в банк данные

Блок-схема алгоритма программы, осуществляющей запись и считывание полей электронного паспорта (табл.8.2) с шифрацией может выглядеть так, как это изображено на рис.8.5. Нетрудно видеть, что эта блок-схема содержит как свои части алгоритмы, рассмотренные здесь ранее.

Отличие состоит главным образом в том, что алгоритм (рис.8.5) оперирует не с одной, а с несколькими строками, а также, в том, что в нем реализован простейший интерактивный интерфейс.

Если запись полей паспорта прошла успешно, терминальное устройство (контроллер) ответит кадром с нулевым значением нулевого байта поля данных.

Поскольку кадр, используемый для чтения, как таковых, данных не содержит, его длину целесообразно сделать минимальной, т.е. 10h.

Для чтения следует использовать кадр, значения полей которого приведены в табл.8.4.

Таблица 8.4.

Значение полей кадра для чтения

FT=10h	Данные кадра (может быть любой от 10h до 4Bh)
SN=0000000001h	Сетевой адрес терминала
Cmd=94h	Команда чтения смарт-карты с дешифрацией
Data[0]=0Xh	Банк смарт-карты (X – от 0 до 3)

В ответ контроллер должен вернуть кадр со следующим содержимым поля Data:

- Data [0] = 00h свидетельствует об успешном выполнении команды;
- Data [1...64] прочитанные данные.

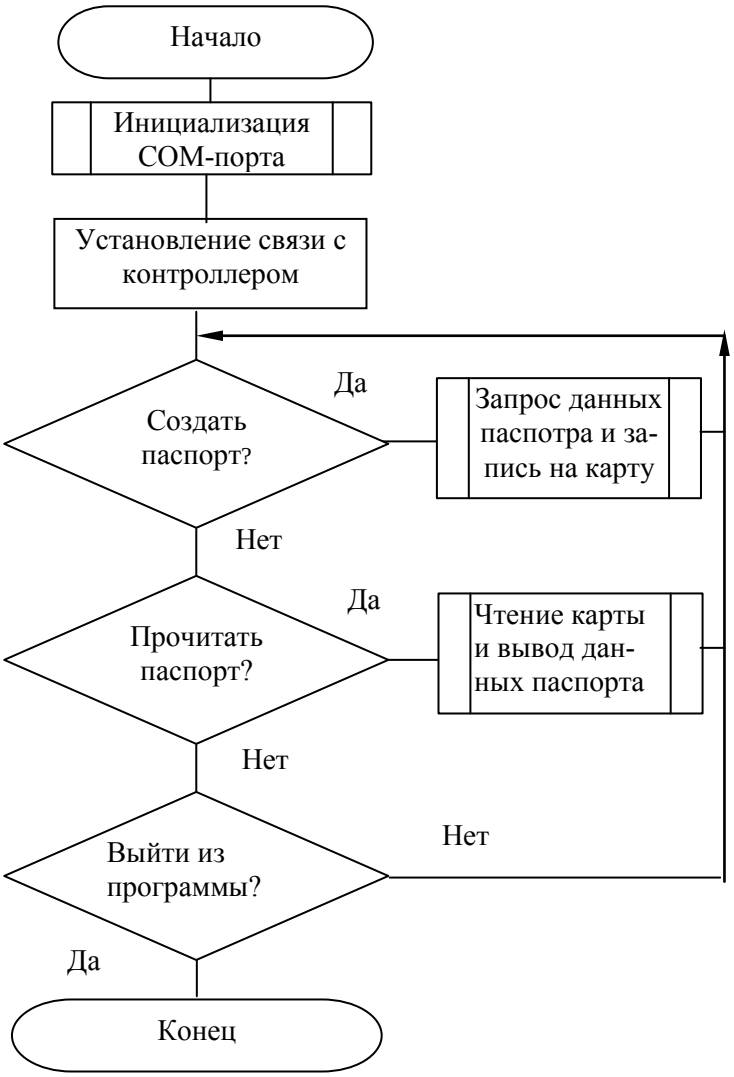


Рис.8.5. Схема алгоритма, осуществляющего запись и считывание полей электронного паспорта с шифрацией

9. ВЫПОЛНЕНИЕ ЗАДАНИЙ

9.1. В режиме эмуляции DOS (или DOS) запустить оболочку с помощью исполняемого файла smart_11.exe. Выбрать пункт меню "Контроллер" и установить связь с терминалом. Определить серийный номер и тип терминала, а также присвоенный ему сетевой адрес. Записать эти данные в протокол, а в отчете представить форматы кадров, используемые для установления связи с терминалом.

9.2. Выбрать пункт меню "Звуковой сигнал". Генерация сигнала будет свидетельствовать о передаче команды терминалу на генерацию звукового сигнала. В отчете представить форматы кадра, с помощью которого эта команда была передана в терминальное устройство.

9.3. Выбрать пункт меню "Параметры". Полученные данные записать в протокол и сравнить их с теми, которые были получены в пункте 9.1.

9.4. Выбрать пункт меню "Загрузить дамп". Загрузить один из имеющихся файлов с расширением dmp. Загрузку произвести в окно шестнадцатеричного дампа. Внести изменения в дамп памяти таким образом, чтобы первая его половина содержала число FFh во всех байтах дампа. Модифицированный дамп сохранить под новым именем. Выйти из программной оболочки, выбрав соответствующий пункт меню.

9.5. Запустить в режиме эмуляции DOS (или DOS) программу example1.exe. Убедиться в наличии звукового сигнала. Определить серийный номер, тип и сетевой адрес терминала и сравнить их с теми, которые были получены в пунктах 9.1. и 9.3. В отчете представить блок схему программы, с помощью которой осуществляется аннулирование текущего сетевого адреса, присвоение нового сетевого адреса и генерация звукового сигнала. Выйти из программы example1.exe.

9.6. Запустить в режиме эмуляции DOS (или DOS) программу example2.exe. На приглашение ввести строку символов для записи на смарт-карту, введите заданную преподавателем строку и нажмите "ввод". На запрос программы нужно ли прочитать записанную на карту строку, ответить (Y). Наблюдать прочитанную с карты строку символов. В отчете представить форматы кадров, которые использовала программа для записи и считывания строки без шифрации. Выйти из программы example2.exe.

9.7. Запустить в режиме эмуляции DOS (или DOS) оболочку swart_11.exe. Выбрать пункт меню "Контроллер" и установить связь с терминалом, убедиться, что связь установлена. Затем выбрать пункт меню "Карта". Вставить пластиковую карту в приемник терминала и выбрать пункт меню "Чтение без расшифровки". Прочитать содержимое банка памяти смарт-карты в шестнадцатеричном представлении. Воспользовавшись таблицей кодировки стандартного кода ASCII, убедиться что прочитанная строка соответствует записанной в пункте 9.6. Выйти из программной оболочки smart_11.exe.

9.8. Запустить в режиме эмуляции DOS (или DOS) программу example3.exe. Вставить карту в приемник терминала. Выбрать пункт меню "Создать новый паспорт". По согласованию с преподавателем, ввести в русском регистре следующие данные паспорта:

- Номер;
- Ф.И.О.;
- Дата и место рождения;
- Адрес места жительства;
- Дополнительные данные.

Выбрать пункт меню "Прочитать паспорт". Убедиться в правильности информации, записанной на карте с использованием шифрации. Выйти из программы example3.exe. Представить в отчете блок-схему программы, осуществляющей информационный обмен с терминалом в режиме записи и чтения данных электронного паспорта с использованием шифрации.

9.9. Запустить в режиме эмуляции DOS (или DOS) оболочку smart_11.exe. Установить связь с терминалом. Выбрать пункт меню "Карта". Вставить смарт-карту в приемник терминала и выбрать пункт меню "Чтение без расшифровки" в шестнадцатеричном представлении. Наблюдать данные паспорта без расшифровки, которые имеют вид массива случайных чисел. Выбрать пункт меню "Чтение с расшифровкой". Убедиться, что данные в этом случае имеют упорядоченный характер (использование русской кодировки дает возможность прочесть паспорт из шестнадцатеричного окна).

9.10. Не выходя из программы smart_11.exe, выбрать пункт меню "Стереть информацию". Прочесть данные с карты и убедиться, что она не содержит данных. Выйти из программной оболочки smart_11.exe.

В качестве заданий повышенной сложности предлагается по усмотрению преподавателя один из вариантов.

1. Написать программу на языке высокого уровня, которая аннулирует текущий сетевой адрес терминала, присваивает ему новый сетевой адрес, а также вызывает генерацию звукового сигнала.

2. Написать программу на языке высокого уровня, которая позволяет ввести с консоли компьютера строку символов, записать эту строку без шифрации в память смарт-карты, а также прочитать в случае необходимости эту строку.

Для выполнения заданий повышенной сложности можно пользоваться блок- схемами алгоритмов соответствующих программ, процедурами и функциями, приведенными в разделе 10, а также листингом программы на языке Паскаль из приложения 2. Эта программа позволяет вводить на экран строки электронного паспорта, редактировать их, записывать с шифрацией на смарт-карту, а также, считывать эти строки с дешифрацией и выводить на экран монитора.

10. ПРИМЕРЫ ПРОЦЕДУР И ФУНКЦИЙ, ОПИСАНИЕ ТИПОВ ДАННЫХ НА ЯЗЫКЕ TURBO PASCAL

Ниже приведены примеры функций инициализации COM-порта, проверки контрольной суммы, формирования кадра, ожидания, а также процедуры чтения контрольной суммы.

Описание типов данных

```
tNA =   Array [1..5] of Byte;
     {Сетевой адрес по ASK-BUS 2.0}
tData = Array [1..MaxFrameLen-10-2] of Byte;
     {Область данных кадра по ASK-BUS 2.0}
tBuffer = Record
{Буфер для содержания кадра по ASK-BUS 2.0 в виде записи}
       FT:  Byte;
       NA:  tNA;
       Cmd: Byte;
       Status: Byte;
       Data: tData;
       End;
tLinkBuffer = Array [0..MAXFrameLen-1] of Byte;
{Буфер для содержания кадра по ASK-BUS 2.0 в виде
 неструктурированного массива}
tNoteData=Record
     {информация об обнаруженном устройстве}
       SN: tNA;
       Ver: Byte;
       ID_String: String;
       Address: tNA;
       Buf_Size: Byte;
       Delay: Word;
      End; { of Record }
tDump=Array [0..511] of Byte;      {Рабочий массив}
```

Процедура инициализации COM-порта

```
Function COMInit(Num: Byte): Boolean;
 {Процедура инициализации COM-порта}
Begin
 COMInit:=False;
 If (Num>4) or (Num=0) Then Exit;          1,2,3,4
 Case Num of  {Базовые адреса в зависимости от номера порта}
   1: COMBase:=$3F8;
   2: COMBase:=$2F8;
   3: COMBase:=$3E8;
   4: COMBase:=$2E8;
```

```
  End; { of Case }                в ассемблере
  asm
        mov     dx,COMBase        адрес порта в DX
        add     dx, 3
        in      al,dx
        or      al,$80
        out     dx,al         {установить бит DLAB}
        mov     ax,$0C        {коэффициент для скорости 9600}
        mov     dx,COMBase
        out     dx,ax         {установить скорость}
        mov     dx,COMBase
        add     dx, 1
        xor     ax, ax
        out     dx,ax         {установить скорость}
        mov     al,$03
        mov     dx,COMBase
        add     dx, 3
        out     dx,al         {установить параметры: без
                               бита  четности, 1 топ-бит}
        mov     al,$00
        mov     dx,COMBase
        add     dx, 1
        out     dx,al         {Запретить прерывания}
        mov     dx,COMBase
        in      al,dx         {Прочитать состояние
                               регистра  данных}
  end; { of asm }
  CurCOM:=Num;
  COMInit:=True;
 End; { of COMInit }
```

Функция проверки контрольной суммы

```
Function CheckLinkCRC : Boolean;
 {Проверить контрольную сумму. False - не совпала}
 Var i: Word;
     Chk_Calc, Chk_Frame: Word;
 Begin
  Chk_Calc:=0;
  For i:=0 To COMFrameLen-3 Do
   Chk_Calc:=Word(Chk_Calc+RXBuf[i]);
  Move(RXBuf[COMFrameLen-2], Chk_Frame, SizeOf(Chk_Frame));
  If Chk_Calc=Chk_Frame Then CheckLinkCRC:=True Else
   CheckLinkCRC:=False;
 End; { of CheckLinkCRC }
```

Процедура чтения контрольной суммы

```
Procedure CalcLinkCRC;
 {Прочитать контрольную сумму и записать в конец кадра}
 Var i: Word;
```

```
    Chk_Calc: Word;
 Begin
  Chk_Calc:=0;
  For i:=0 To COMFrameLen-3 Do
   Chk_Calc:=Word(Chk_Calc+TXBuf[i]);
  Move(Chk_Calc, TXBuf[COMFrameLen-2], SizeOf(Chk_Calc));
 End; { of CalcLinkCRC }
```

Функция формирования кадра

```
Function Send_Frame(FT: Byte; NA: tNA; Cmd: Byte; Status:
Byte; Data: tData): Boolean;
 {Формирование кадра длины COMFrameLen для передачи, подсчет
КС, передача с ожиданием ухода всего кадра}
  Var Res: Boolean; Start: LongInt; i: Word; Symb: Byte;
  Begin
   Send_Frame:=False;
   Out_Buf.FT:=FT;
   Out_Buf.NA:=NA;
   Out_Buf.Cmd:=Cmd;
   Out_Buf.Status:=Status;
   For i:=1 To COMFrameLen-2 Do Out_Buf.Data[i]:=Data[i];
   CalcLinkCRC;
   Start:=Cur_Time;        {Запомнить начало отправки кадра}
   For i:=0 To COMFrameLen-1 Do {Цикл отправления запросного
                                 кадра}
Begin
     Symb:=TXBuf[i];
     asm
      mov  dx, COMBase
      mov  al, Symb
      out  dx, al        {Отправить очередной байт}
      add  dx, 5
   @1:
      in   al, dx
      test al, $40       {Сдвиговый регистр порта пуст?}
      jz   @1            {Нет}
     end; { of asm }
     If (Cur_Time-Start)>Time_Out Then Exit; {Отслеживание
Тайм-Аута}
    End; { of For }
   Send_Frame:=True;
  End; { of Send_Frame }
```

Функция ожидания кадра (Poling)

```
Function Receive_Frame: Boolean;
{Ожидать кадр, пока не тайм-аут. Если принят – проверить КС}
  Var Start_Time: LongInt;
      Cnt, Symb: Byte; Got: Boolean;
  Begin
```

```
 Cnt:=0; Start_Time:=Cur_Time;
 In_Buf.FT:=0;
 While Abs(Cur_Time-Start_Time)<Time_Out Do
  Begin
   asm
    mov  dx, COMBase
    add  dx, 5
    in   al, dx
    mov  Got, 0
    test al, $01
    jz   @1
    mov  Got, 1
    mov  dx, COMBase
    in   al, dx
    mov  Symb, al
@1:
   end; { of asm }
   If Got Then
    Begin
     RXBuf[Cnt]:=Symb;
     Inc(Cnt);
    End; { of If }
  End; { of While }
 If In_Buf.FT>$0F Then COMFrameLen:=In_Buf.FT
                  Else COMFrameLen:=64;
 If Cnt<$10       Then Receive_Frame:=False
                  Else Receive_Frame:=CheckLinkCRC;
End; { of Receive_Frame }
```

ПРИЛОЖЕНИЕ

ПРИМЕР ПРОГРАММЫ

```
Program Example1;
Uses CRT;
Const
    MAXFrameLen      = 256;   { Максимальная длина  кадра  }
 Type
      tNA =   Array [1..5] of Byte;
      tData = Array [1..MaxFrameLen-10-2] of Byte;
      tBuffer = Record
       FT:  Byte;
       NA:  tNA;
       Cmd: Byte;
       Status: Byte;
       Data: tData;
      End; { of Record }
      tLinkBuffer = Array [0..MAXFrameLen-1] of Byte;
      tNoteData=Record
       SN: tNA;
       Ver: Byte;
       ID_String: String;
       Address: tNA;
       Buf_Size: Byte;
       Delay: Word;
      End; { of Record }
      tDump=Array [0..511] of Byte;

 Const
    Time_Out: Byte = 18*1; {тайм-аут ожидания кадра  - одна
                            секунда }

{Константы, задающие все типы кадров, команды, статус для
обмена с контроллером смарт-карт в соответствии с ASK-BUS
v.2.0}

{----------------Типы кадров ------------------------------}
    aftEmpty         =         0;
    aftReset         =         1;
    aftIDRequest     =         2;
    aftSetAddress    =         3;
    aftMasterRequest =         4;
    aftNoteAnswer    =         5;
{-----------------------------------------------------------}

{------------------- Команды ------------------------------}
    {Контроллер сети}
    acmNoteInit      =         0;
```

```
 acmNoteSelfTest  =         1;
 acmGetNoteSN     =         2;
 acmLastCom       =         3;
 acmNoteIDString  =         4;
 {Периферийное устройство }
 acmBoardInit     =         $80;
 acmBoardSelfTest =         $81;
 acmGetBoardSN    =         $82;
 acmBoardIDString =         $83;
 {Только для смарт-контроллера }
 acmWriteWoCoding =         $90;
 acmReadWOCoding  =         $91;
 acmClearCard     =         $92;
 acmWriteWCoding  =         $93;
 acmReadWCoding   =         $94;
 acmWriteProtocol =         $95;
 acmReadProtocol  =         $96;
 acmGenerateBeep  =         $97;
{-------------------------------------------------------}
{---------------- Статусы ------------------------------}
 astSuccessNote   =         0;
 astSuccessBoard  =         $80;
 astSendingCom    =         1;
 astGettingCom    =         2;
 astWaitAnswer    =         3;
 astWaitReady     =         4;
 astNoCommands    =         5;
 astLongCommand   =         6;
 astComErrorNote  =         7;
 astComErrorBoard =         $87;
{-------------------------------------------------------}
{---------------- Типы узлов ---------------------------}
 antNetCard       =         1;
 antTimos         =         2;
 antSymmetron     =         3;
 antDIP           =         4;
 antBridge        =         5;
 antSmart         =         6; { Контроллер смарт-карт  }
{-------------------------------------------------------}
{------------ Ошибки (только для смарт-контроллера ) ---}
 sstSuccess       =         0;
 sstEEPROMFailR   =        $81;
 sstEEPROMFailW   =        $82;
 sstCardFailR     =        $83;
 sstCardFailW     =        $84;
 sstWriteDataFail =        $85;
 sstNoCard        =        $86;
 sstCardIn        =        $87;
 sstInvalidBank   =        $88;
{-------------------------------------------------------}
```

```
Var
     RXBuf : tLinkBuffer;        {Приемный буфер кадров}
     TXBuf : tLinkBuffer;        {Буфер для ответных кадров}

     Out_Buf: tBuffer absolute TXBuf; {Структурированный ис-
ходящий буфер}
     In_Buf : tBuffer absolute RXBuf; {Структурированный от-
ветный буфер}
     {-----------------------------}
     Frame_FT: Byte; Frame_NA: tNA; Frame_Cmd: Byte;
Frame_Status: Byte;
     Frame_Data: tData;
     {-----------------------------}
     Cur_Time    : LongInt absolute $0000:$046C; {Текущее
состояние счетчика  системного времени}
     CurCOM      : Byte; {Текущий COM-порт}
     COMBase     : Word; {Базовый адрес текущего COM-порта}
     COMFrameLen : Byte; {Длина принятого или отправляемого
                          кадра}
     {-----------------------------}

     Note   : tNoteData; {Запись с данными о текущем устрой-
стве,
                          с кот. установлена  связь}
     {-----------------------------}
     Dump: tDump;

Function COMInit(Num: Byte): Boolean;
 {Проинициализировать COM-порт}
 Begin
  COMInit:=False;
  If (Num>4) or (Num=0) Then Exit;
  Case Num of        {Базовые адреса в зависимости от номера
                      порта}
   1: COMBase:=$3F8;
   2: COMBase:=$2F8;
   3: COMBase:=$3E8;
   4: COMBase:=$2E8;
  End; { of Case }
  asm
        mov     dx,COMBase
        add     dx, 3
        in      al,dx
        or      al,$80
        out     dx,al       {установить бит DLAB}
        mov     ax,$0C      {коэффициент для скорости 9600}
        mov     dx,COMBase
        out     dx,ax       {установить скорость}
        mov     dx,COMBase
        add     dx, 1
```

```
        xor     ax, ax
        out     dx,ax       {установить скорость}
        mov     al,$03
        mov     dx,COMBase
        add     dx, 3
        out     dx,al       {установить параметры: без бит
                             четности,1 стоп-бит}
        mov     al,$00
        mov     dx,COMBase
        add     dx, 1
        out     dx,al       {Запретить прерывания}
        mov     dx,COMBase
        in      al,dx       {Прочитать состояние регистра
                             данных}
  end; { of asm }
  CurCOM:=Num;
  COMInit:=True;
 End; { of COMInit }

Function CheckLinkCRC : Boolean;
 {Посчитать контрольную сумму и записать в конец кадра}
 Var i: Word;
     Chk_Calc, Chk_Frame: Word;
 Begin
  Chk_Calc:=0;
  For i:=0 To COMFrameLen-3 Do
   Chk_Calc:=Word(Chk_Calc+RXBuf[i]);
  Move(RXBuf[COMFrameLen-2], Chk_Frame, SizeOf(Chk_Frame));
  If Chk_Calc=Chk_Frame Then CheckLinkCRC:=True Else
   CheckLinkCRC:=False;
 End; { of CheckLinkCRC }

Procedure CalcLinkCRC;
 {Проверить контрольную сумму}
 Var i: Word;
     Chk_Calc: Word;
 Begin
  Chk_Calc:=0;
  For i:=0 To COMFrameLen-3 Do
   Chk_Calc:=Word(Chk_Calc+TXBuf[i]);
  Move(Chk_Calc, TXBuf[COMFrameLen-2], SizeOf(Chk_Calc));
 End; { of CalcLinkCRC }

 Procedure TickWait(Const Num: Byte);
 {Ожидание Num тиков, т.е. Num количеств инкрементов систем-
ного
  счетчика}
  Var StartTime: LongInt;
  Begin
   StartTime:=Cur_Time;
```

```
   While (Cur_Time-StartTime)<>Num Do;
  End; { of TickWait }

 Function Send_Frame(FT: Byte; NA: tNA; Cmd: Byte; Status:
Byte;
                     Data: tData): Boolean;
{Формирование кадра длины COMFrameLen для передачи, подсчет
K', передача с ожиданием ухода всего кадра}
  Var Res: Boolean; Start: LongInt;
      i: Word; Symb: Byte;
  Begin
   Send_Frame:=False;
   Out_Buf.FT:=FT;
   Out_Buf.NA:=NA;
   Out_Buf.Cmd:=Cmd;
   Out_Buf.Status:=Status;
   For i:=1 To COMFrameLen-2 Do Out_Buf.Data[i]:=Data[i];
   CalcLinkCRC;
   Start:=Cur_Time;              {Запомнить начало отправки
                                  кадра}
   For i:=0 To COMFrameLen-1 Do  {Цикл отправления
                                  запросного кадра}
    Begin
     Symb:=TXBuf[i];
     asm
      mov  dx, COMBase
      mov  al, Symb
      out  dx, al                {Отправить очередной байт}
      add  dx, 5
   @1:
      in   al, dx
      test al, $40               {Сдвиговый регистр порта
                                  пуст?}
      jz   @1                    {Нет}
     end; { of asm }
     If (Cur_Time-Start)>Time_Out Then Exit; {Отслеживание
                                              Тайм-Аута}
    End; { of For }
   Send_Frame:=True;
  End; { of Send_Frame }

 Function Receive_Frame: Boolean;
 {Ожидать кадр, пока не тайм-аут. Если принят - проверить
K'}
  Var Start_Time: LongInt;
      Cnt, Symb: Byte; Got: Boolean;
  Begin
   Cnt:=0; Start_Time:=Cur_Time;
   In_Buf.FT:=0;
   While Abs(Cur_Time-Start_Time)<Time_Out Do
```

```
    Begin
     asm
      mov  dx, COMBase
      add  dx, 5
      in   al, dx
      mov  Got, 0
      test al, $01
      jz   @1
      mov  Got, 1
      mov  dx, COMBase
      in   al, dx
      mov  Symb, al
  @1:
     end; { of asm }
     If Got Then
      Begin
       RXBuf[Cnt]:=Symb;
       Inc(Cnt);
      End; { of If }
    End; { of While }
   If In_Buf.FT>$0F Then COMFrameLen:=In_Buf.FT
                    Else COMFrameLen:=64;
   If Cnt<$10       Then Receive_Frame:=False
                    Else Receive_Frame:=CheckLinkCRC;
  End; { of Receive_Frame }

 Function Note_Address_Reset: Boolean;
 {Осуществить сброс адреса всех узлов}
  Begin
   COMFrameLen:=64; Note_Address_Reset:=False;
   If Not Send_Frame(aftReset, Frame_NA, Frame_Cmd,
                     Frame_Status, Frame_Data) Then Exit
   Else Note_Address_Reset:=True;
  End; { of Note_Address_Reset }

 Function No_Address_Request(Var Note_Type: Byte; Var
Note_SN: Longint): Boolean;
 {Запросить устройства в сети, не имеющие сетевого адреса.
При
  успехе возвра-щает тип и серийный номер одного из таких
  устройств, а также другие параметры устройств по ASK BUS.
  Параметры заносятся в  глобальную переменную Note}
  Begin
   COMFrameLen:=64; No_Address_Request:=False;
   If Not Send_Frame(aftIDRequest, Frame_NA, Frame_Cmd,
                     Frame_Status, Frame_Data) Then Exit;
   If Receive_Frame Then
      Begin
       If In_Buf.Status<>astSuccessNote Then Exit;
       If In_Buf.FT<>aftNoteAnswer Then Exit;
```

```
       With In_Buf Do
         Begin
          Note_Type:=Data[1];
          Move(Data[2], Note_SN, SizeOf(Note_SN));
          Move(Data[1], Note.SN, SizeOf(Note.SN));
          Move(Data[6], Note.Ver, SizeOf(Note.Ver));
          Move(Data[7], Note.Buf_Size,
SizeOf(Note.Buf_Size));
          Move(Data[8], Note.Delay, SizeOf(Note.Delay));
         End; { of With }
       No_Address_Request:=true;
      End; { of If }
  End; { of No_Address_Request }

 Function Set_Note_Address(Note_Type: Byte; Note_SN:
LongInt; Note_Address: tNA): Boolean;
 {Присвоить узлу сети с серийным номером Note_SN и типом
  Note_Type  дрес Note_Address}
  Begin
   COMFrameLen:=64; Set_Note_Address:=False;
   Frame_NA[1]:=Note_Type;
   Move(Note_SN, Frame_NA[2], SizeOf(Note_SN));
   Move(Note_Address, Frame_Data[1], SizeOf(Note_Address));
   If Not Send_Frame(aftSetAddress, Frame_NA, Frame_Cmd,
                     Frame_Status, Frame_Data) Then Exit;
   If Receive_Frame Then
      Begin
       If In_Buf.Status<>astSuccessNote Then Exit;
       Set_Note_Address:=True; Note.Address:=Note_Address;
      End; { of If }
  End; { of Set_Note_Address }

 Function Generate_Beep: Boolean;
 { Дать команду на генерацию кратковременного звукового сиг-
нала}
  Begin
   COMFrameLen:=16; Generate_Beep:=False;
   Frame_FT:=COMFrameLen;
   Frame_NA:=Note.Address; Frame_Cmd:=acmGenerateBeep;
   If Not Send_Frame(Frame_FT, Frame_NA, Frame_Cmd,
                     Frame_Status, Frame_Data) Then Exit;
   If Receive_Frame Then
      Begin
       If In_Buf.Status<>astSuccessNote Then
         Generate_Beep:=False
       Else
         Generate_Beep:=True;
      End; { of If }
  End; { of Generate_Beep }
```

```
 Function Check_Card(Var Present: Boolean): Boolean;
 { Проверить наличие карты в считывателе путем попытки чте-
ния с карты из нулевого банка}
  Begin
   COMFrameLen:=16; Check_Card:=False;
   Frame_FT:=COMFrameLen;
   Frame_NA:=Note.Address; Frame_Cmd:=acmReadWOCoding;
   Frame_Data[1]:=0;
   If Not Send_Frame(Frame_FT, Frame_NA, Frame_Cmd,
          Frame_Status, Frame_Data) Then Exit;
   If Receive_Frame Then
      Begin
       If In_Buf.Status<>astSuccessNote Then
         Check_Card:=False
       Else
         Check_Card:=True;
       If In_Buf.Data[1]=sstNoCard Then Present:=False
       Else
        Begin
         If (In_Buf.Data[1]=sstSuccess) or
            (In_Buf.Data[1]=sstCardIn) Then Present:=True
         Else Exit;
        End; { of Else }
      End; { of If }
  End; { of Check_Card }

 Function Card_Write(Bank: Byte; Crypto: Boolean): Boolean;
{ Записать из содерж. массива  Dump на карту в соотв. банк.
 rypto=False - без шифрования, Crypto=True - с шифрованием}
  Begin
   COMFrameLen:=75; Card_Write:=False;
   Frame_FT:=COMFrameLen;
   Frame_NA:=Note.Address;
   If Crypto Then Frame_Cmd:=acmWriteWCoding Else
Frame_Cmd:=acmWriteWOCoding;
   Frame_Data[1]:=Bank;
   Move(Dump[Bank*64], Frame_Data[2], 64);
   If Not Send_Frame(Frame_FT, Frame_NA, Frame_Cmd,
Frame_Status, Frame_Data) Then Exit;
   If Receive_Frame Then
      Begin
       If (In_Buf.Status<>astSuccessNote) or
(In_Buf.Data[1]<>sstSuccess) Then
         Card_Write:=False
       Else
         Card_Write:=True;
       If In_Buf.FT<>75 Then Card_Write:=False;
      End; { of If }
  End; { of Card_Write }
```

```
 Function Card_Read(Bank: Byte; Crypto: Boolean): Boolean;
 { Прочитать с карты из соотв. банка в массив Dump.
Crypto=False - без расшифровки, Crypto=True - с расшифров-
кой}
  Begin
   COMFrameLen:=16; Card_Read:=False;
   Frame_FT:=COMFrameLen;
   Frame_NA:=Note.Address;
   If Crypto Then Frame_Cmd:=acmReadWCoding Else
Frame_Cmd:=acmReadWOCoding;
   Frame_Data[1]:=Bank;
   If Not Send_Frame(Frame_FT, Frame_NA, Frame_Cmd,
Frame_Status, Frame_Data) Then Exit;
   If Receive_Frame Then
      Begin
       If (In_Buf.Status<>astSuccessNote) or
(In_Buf.Data[1]<>sstSuccess) Then
         Card_Read:=False
       Else
        Begin
         Card_Read:=True; Move(In_Buf.Data[2],
Dump[Bank*64],
               64);
        End; { of Else}
       If In_Buf.FT<>75 Then Card_Read:=False;
      End; { of If }
  End; { of Card_Read }

Var
 NoteType: Byte; NoteSN: LongInt;
 Card_In: Boolean;
 PPNum: LongInt; Action: Byte;
 FIO, Birthday, Home, Appendix: String;

Begin { --------- Основная часть программы ------------ }

 ClrScr;     {Очистить экран}

 COMInit(2); {Проинициализировать COM2}

 Writeln;
 Writeln('Сброс сетевых  адресов...');
 If Not Note_Address_Reset Then {Сбросить  адрес узла}
  Begin
   Writeln('Ошибка! Не удалось передать кадр на сброс
            адресов!'); Halt(1);
  End; { of If }

 TickWait(9); {Ожидание около 0.5 сек}
```

```
 Writeln('Запрос узлов с не установленным адресом...');
 If Not No_Address_Request(NoteType, NoteSN) Then
  Begin
   Writeln('Ошибка! Контроллер не обнаружен!'); Halt(1);
  End { of If }
 Else
  Begin
   Writeln(' Обнаружено устройство: тип = ', NoteType, '
             серийный номер = ', NoteSN);
  End; { of Else }

 FillChar(Note.Address, SizeOf(Note.Address), 0);
Note.Address[1]:=1;
 Writeln('Присвоение сетевого адреса...');
 If Not Set_Note_Address(NoteType, NoteSN, Note.Address)
Then
  Begin
   Writeln(' Ошибка! Не удалось присвоить сетевой  адрес!');
Halt(1);
  End { of If }
 Else
  Begin
   Writeln(' Присвоен сетевой адрес = 0000000001');
  End; { of Else }

 TickWait(36); {Ожидание около 2 сек}

 While True Do { Бесконечный цикл }
  Begin
   Repeat
    ClrScr; Writeln;
    Writeln('  Выбор:');
    Writeln(' 1. Создать новый паспорт.');
    Writeln(' 2. Прочитать паспорт.');
    Writeln(' 3. Выход.');
    Write(':');
    Readln(Action);
    If Action=3 Then Halt(0);
   Until (Action>0) and (Action<4);
   Writeln;
   Case Action of
    1:
     Begin
      Write('Введите номер паспорта:');
      Readln(PPNum);
      Write('Введите ФИО владельца:');
      Readln(FIO);
      Write('Введите дату и место рождения владельца:');
      Readln(Birthday);
      Write('Введите место жительства владельца:');
```

```
      Readln(Home);
      Write('Введите дополнительную информацию: ');
      Readln(Appendix);
      Move(PPNum, Dump[0], SizeOf(PPNum));
      Move(FIO[0], Dump[4], 60);
      Move(Birthday[0], Dump[64], 64);
      Move(Home[0], Dump[128], 64);
      Move(Appendix[0], Dump[192], 64);
      Repeat
       Check_Card(Card_In);
       Writeln;
       If Not Card_In Then
        Begin
         Write(' Поместите смарт-карту в считыватель и
нажмите
                любую клавишу.');
         While Not KeyPressed Do;
        End; { of If }
      Until Card_In;
      If (Not Card_Write(0, True)) or (Not Card_Write(1,
True)) or
         (Not Card_Write(2, True)) or (Not Card_Write(3,
True)) Then
Begin
       Writeln(' Ошибка! Не удалось записать смарт-
карту!'); Halt(1);
       End { of If }
      Else
       Begin
        Writeln(' Карта записана ...');
       End; { of Else }
     End;
    2:
     Begin
      Repeat
       Check_Card(Card_In);
       Writeln;
       If Not Card_In Then
        Begin
         Write('Поместите смарт-карту в считыватель и нажми-
те любую клавишу.');
         While Not KeyPressed Do;
        End; { of If }
      Until Card_In;
      If (Not Card_Read(0, True)) or (Not Card_Read(1,
True)) or
         (Not Card_Read(2, True)) or (Not Card_Read(3,
True)) Then
       Begin
```

```
        Writeln(' Ошибка! Не удалось прочитать смарт-
карту!'); Halt(1);
       End { of If }
      Else
       Begin
        Move(Dump[0], PPNum, SizeOf(PPNum));
        Move(Dump[4], FIO[0], 60);
        Move(Dump[64], Birthday[0], 64);
        Move(Dump[128], Home[0], 64);
        Move(Dump[192], Appendix[0], 64);
        Writeln('Предъявлен паспорт № ', PPNum);
        Writeln(' ФИО владельца: ', FIO);
        Writeln(' Дата и место рождения владельца: ',
Birthday);
        Writeln(' Место жительства владельца: ', Home);
        Writeln(' Дополнительная информация: ', Appendix);
       End; { of Else }
     End;
   End; { of Case }
   Writeln; Write('  Нажмите любую клавишу...');
   While Not KeyPressed Do;  {Ожидание нажатия любой
клавиши}
  End; { of While }

End.  { ---------------------------------------------- }
```

КОНТРОЛЬНЫЕ ВОПРОСЫ ПО РАЗДЕЛАМ

К разделу 2

2.1.Назовите сигналы интерфейса RS-232C, обеспечивающие синхронизацию, управление, передачу и прием данных соответственно.
2.2.Сколько независимых каналов данных обеспечивает интерфейс RS-232C и в каком режиме они работают?
2.3.Сколько символов содержит один пакет при асинхронной передаче?
2.4.Сколько битов используется для передачи одного символа ASCII по интерфейсу RS-232C?
2.5.Объясните, как можно обнаружить ошибку по паритету (четности)?
2.6.Какие значения напряжений соответствуют "0" (SPACE) и "1" (MARK)?
2.7.Поясните назначение элементов и выводов на схеме рис.1.4.
2.8.Расскажите, как BIOS обнаруживает последовательные адаптеры.
2.9.Является ли интерфейс RS-232C асинхронным?

К разделу 3

3.1.Назовите базовые законы “Алгоритма криптографического преобразования данных” ГОСТ 28147-89.
3.2.Что такое "ключ"?
3.3.Что такое "таблица замен"?
3.4.Расскажите, какие операции содержат алгоритм основного шага преобразования.
3.5.Что общее и в чем отличие алгоритмов базовых циклов зашифрования, расшифровки и выработки имитоприставки?
3.6.Какие режимы шифрования данных Вы знаете?
3.7.Какие достоинства и недостатки у "простой замены"?
3.8.В чем заключается "гаммирование"?
3.9.Чем отличается "гаммирование с обратной связью"?
3.10.Что такое "имитоприставка" и как она используется для повышения криптостойкости ГОСТ 28147-89?
3.11.Что такое "криптостойкость алгоритма" по Кирхгоффу?
3.12.Какой объем ключевого пространства обеспечивает ключ ГОСТ 28147-89?
3.13.В чем достоинства ГОСТ 28147-89 по сравнению с DES?

К разделу 4

4.1. Является ли протокол I^2C асинхронным?
4.2. Расскажите о назначении и взаимодействии блоков на функциональной схеме типа карты GFM-2K.
4.3. Перечислите основные состояния шины I^2C.

4.4. Как протокол I^2C реализует возможность коммуникации нескольких устройств на одной последовательной шине?
4.5. Для какой цели в протоколе I^2C используется сигнал уведомления (Acknowledge)?
4.6. Объясните, как при помощи I^2C осуществляется запись байта?
4.7. Объясните, как при помощи I^2C осуществляется запись страницы?
4.8. Как осуществляется чтение по текущему адресу?
4.9.Как в строке I^2C осуществлена возможность произвольного чтения?
4.10. Как протокол I^2C регламентирует передачу и прием адресов и данных?

К разделу 5

5.1. Какие операции позволяет производить терминал (смарт-контроллер)?
5.2. Поясните назначение светодиодов на плате смарт-контроллера.
5.3. Объясните в каком порядке следует осуществлять коммутацию интерфейсного провода и питания при включении и отключении смарт-контроллера.
5.4. Какой тип памяти использован в микроконтроллере смарт-карт (рис. 4.1) и каков объем этой памяти?
5.5. Как логически организована EEPROM карты GFM-2K?
5.6. Каков алгоритм записи данных в банк памяти GFM-2K, если их объем меньше 64 байт?
5.7. Какова длина кадра записи данных на карту и как распределены байты этого кадра?
5.8. Какова длина кадра чтения данных с карты и как распределены байты этого кадра?
5.9. Как реагирует смарт-контроллер на невозможность прочесть данные с карты?
5.10. Перечислите команды смарт-контроллера карт GFM-2K.

К разделу 6

6.1. Под управлением какой операционной системы работает программная оболочка?
6.2. Какие параметры последовательного интерфейса следует установить перед началом работы смарт-контроллера?
6.3. Какие действия выполняет программная оболочка при выборе пункта меню "Установить связь"?
6.4. Какое расширение должен иметь файл с дампом памяти чтобы его можно было загрузить для работы с ним?
6.5. Как осуществляется редактирование запросного кадра и его передача?
6.6. Перечислите команды, на которые контроллер отвечает передачей ответного кадра.
6.7. Какие данные необходимо внести в программную оболочку для выполнения пункта меню "Читать дамп …"?

6.8. Каким пунктом меню программной оболочки следует воспользоваться для сохранения вновь созданного дампа?
6.9. Используется ли гаммирование при записи дампа с помощью пункта меню "Запись с шифровкой"?

К разделу 7

7.1. Какая сетевая технология использована в сети ASK-LAN 1.2 на физическом уровне?
7.2. Какие устройства входят в состав сети?
7.3. Какая физическая и логическая топология использована в ASK-LAN 1.2 и какие у нее особенности и достоинства?
7.4. Объясните принцип действия и устройство узла сети на структурной схеме (рис.7.2).
7.5. Опишите логику организации обмена информацией в сети ASK-LAN 1.2.
7.6. Опишите логику аппаратного сброса узлов.
7.7. Как происходит присвоение сетевых адресов?
7.8. Изобразите структурную схему вырожденной сети, которая использована в этом цикле лабораторных работ и предусматривает наличие лишь одного терминала (смарт-контроллера).
7.9. В соответствии с используемой терминологией, назовите устройства, входящие в состав вырожденной сети (см. вопрос 7.8).
7.10. В каком режиме (симплексном, дуплексном или полудуплексном) работает интерфейс RS-232C в данном случае?

СОДЕРЖАНИЕ