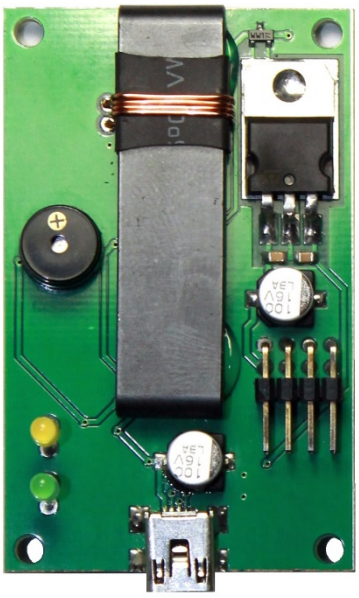
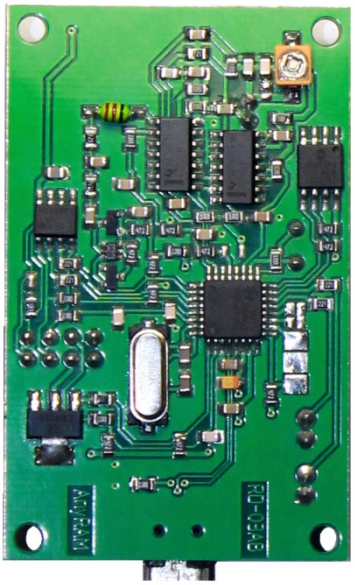
AnyRAM, «ARD Trade» Production
«ARD Trade», 1, 2-Priberezhnaya St, Vitebsk, Belarus
Tel: +375-29-5968565
Fax: +375-21-2586250
Website: http://www.anyram.net
E-Mail: client@anyram.net

Ридер бесконтактных карт стандарта ISO-14443A/B RD-03AB Rel. 2.4





V1.7$_1$

Технические характеристики

Ридер RD-03AB стандарта ISO-14443A/B обеспечивает обмен данными с бесконтактными картами, брелоками, жетонами и т.п., поддерживающими обмен данными по протоколу ISO-14443A, часть 3, а также с устройствами стандарта Mifare®. Обмен по протоколу ISO-14443B и нестандартным протоколам поддерживают только заказные прошивки ридера. Ридер обычно поставляется в исполнении, согласованном с заказчиком.

Ридер обеспеспечивает:
- 4 режима работы, 3 из которых автономные;
- поддержку нестандартных протоколов обмена;
- чтение “сырого” кода обмена;
- поддержку 4х интерфейсов обмена данными (UART, USB, RS485, iButton);
- звуковую индикацию процесса обмена данными (бипер);
- световую индикацию процесса обмена данными (два светодиода);
- управление внешним устройством (замком);
- сохранение настроек в энергонезависимой памяти;
- сохранение ключей доступа во внешней энергонезависимой памяти;
- недоступность для чтения ключей авторизации.

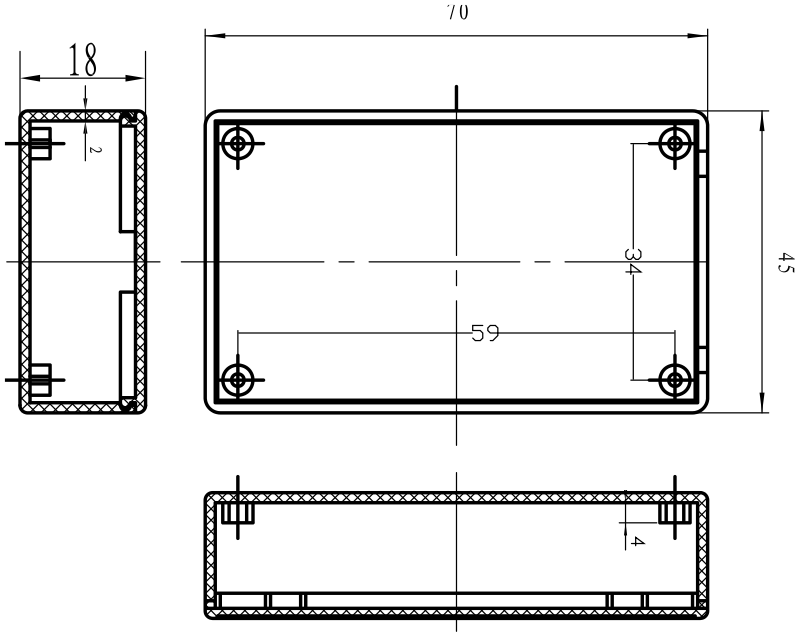
Основное назначение ридера – использование в профессиональных системах авторизованного доступа, транспортных системах, системах оплаты.

Ридер позволяет производить анализ уязвимостей бесконтактных карт, выявлять особенности функционирования карт различных производителей, анализировать карты на уязвимость к кибер-атакам, взлому и т.п.. Совместим с ***mfoc*** и ***mfcuk***.

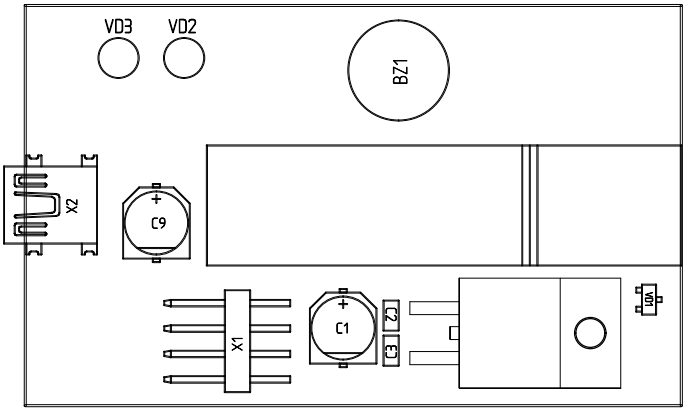
Напряжение питания	- 5В ± 10% или 8…12В (зависит от исполнения).
Потребляемый ток	- не более 250мА (типовое потребление 150мА).
Рабочая частота	- 13.560МГц ± 50ppm.
Напряжённость поля	- не менее 10А·м² в точке считывания.
Тип модуляции	- модуляция типа A, от 106 кбит/с до 817 кбит/с;
	- модуляция типа B, от 106 кбит/с до 817 кбит/с.
Параметры интерфейсов:	
iButton	- открытый сток (+5В max, 5mA), имитация DS1990A;
USB	- LS, поддержка класса HID, режим Control;
RS485	- 8N1, скорость от 1200 бит/с до 115кбит/с (MAX3483);
UART	- 8N1, скорость от 1200 бит/с до 115кбит/с,
	RX – 0…5В, Rвх>1K, TX – 0…3.3В, Rвых<0.5K.
Выход управления внешним устройством:	
SW	- OD (открытый сток), 15мА, Rpull-up = 4k7, +5В макс.;
	- мощный OD (открытый сток), 1.7А, 50В (исполнение).
Ридер сохраняет работоспособность при температуре окружающей среды	
	- от -25°C до +65°C;
	- специальное исполнение от -40°C до +65°C.
Габаритные размеры:	
плата	- 65х40мм;
корпус	- 70х45х18мм (не входит в комплект поставки).

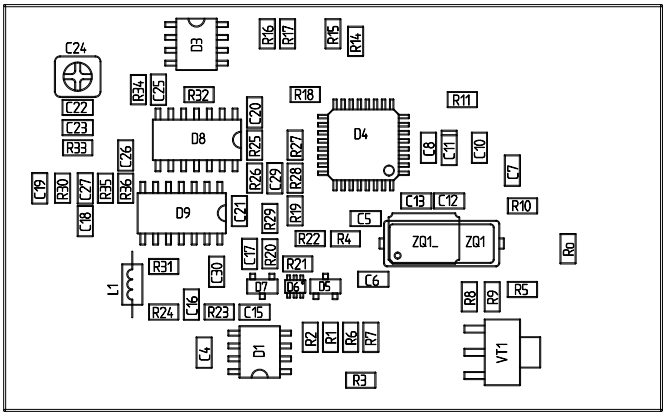
Ридер может быть установлен в пластиковый корпус 20-13 из ABS ф.**Ningbo Sanhe Enclosure** (http://www.china-mould.com). Конструкция верхней крышки корпуса (со стороны светодиодов) позволяет крепить самоклеящуюся наклейку с изображением логотипа.

Чертёж корпуса



Расположение элементов на плате





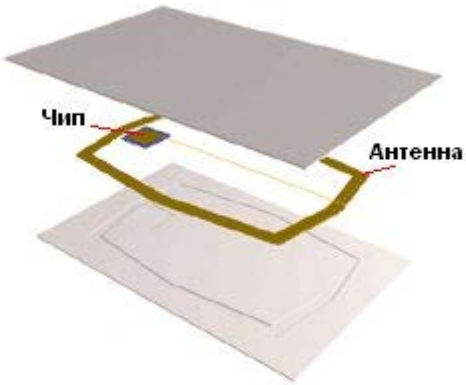
Описание устройства

Расположение элементов на плате ридера

После подачи питания и выхода контроллера в рабочий режим, антенна ридера создаёт направленное электромагнитное поле с частотой 13.56МГц сложной конфигурации. Модуляция электромагнитного поля производится согласно стандарту ISO-14443. Конфигурация антенны выбрана несимметричной для создания оптимального потокосцепления с картой.

При поднесении к антенне ридера бесконтактной карты, карта получает энергию, достаточную для её работы из электромагнитного поля, создаваемого ридером, и через это же поле обменивается с ридером данными. Протокол обмена устроен так, чтобы обмен шёл небольшими порциями данных, чтобы карта в паузах между обменом успевала подзаряжаться.

Устройство бесконтактной карты



Передача данных в направлении карта-ридер производится, путём намеренного искажения картой электромагнитного поля ридера, а ридер фиксирует искажения поля посредством фазового детектора. Очень важно, чтобы при детектировании был правильно установлен опорный уровень на входе детектора: от этого зависит расстояние, на котором будет читаться бесконтактная карта. В ридере есть функция автоматической установки уровня фазового детектора, но для более сложных приложений этот уровень также можно устанавливать вручную.

Все бесконтактные карты стандартов ISO-14443 и Mifare® устроены по-разному и обладают различными функциональными особенностями, однако, у всех этих карт есть общая для них функция: чтение уникального номера карты – UID (**U**nique **ID**entifier). Этот номер может быть использован в качестве электронного ключа. В ридере реализована функция автоматического считывания UID карты.

Ридер также позволяет организовать полноценный протокол обмена с бесконтактной картой. При таком подходе ридер позволяет организовать сложную пропускную систему с записью, чтением, шифрованием данных на карте-пропуске. В зависимости от типа бесконтактной карты, на неё можно записывать от 64 байт до 4 Кбайт данных.

При типовом использовании, ридер обычно программируется на автоматическую работу в режиме пропускного устройства (режим **ACS**) или работу в режиме имитации интерфейса iButton (режим **IBE**), если ридер предназначен для замены брелоков-таблеток iButton бесконтактными картами.

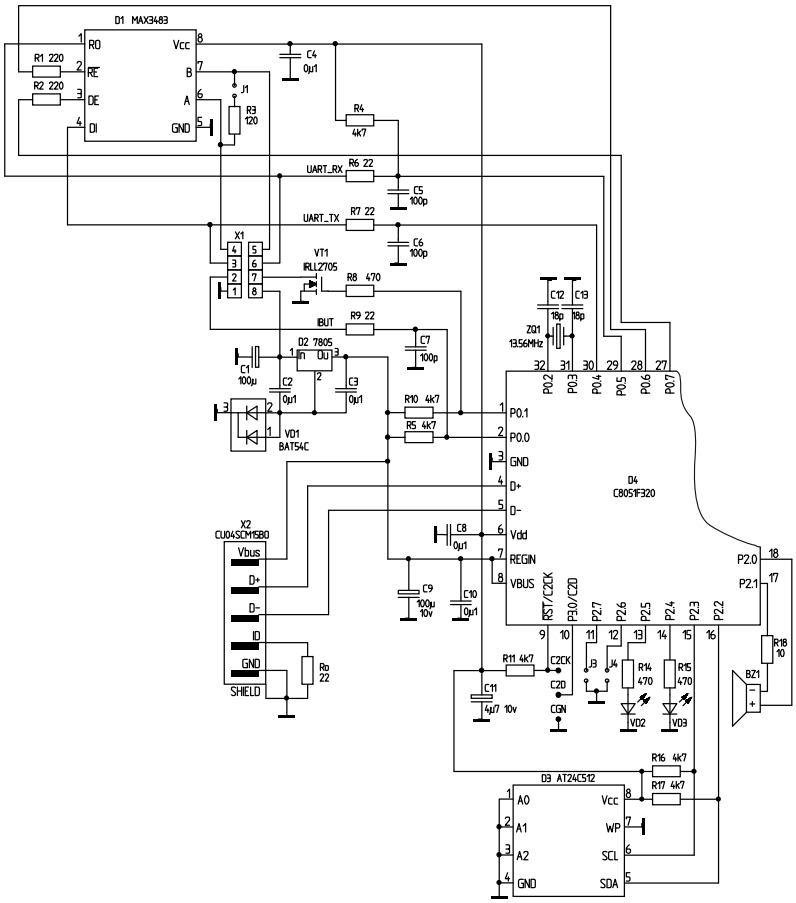
Когда карта подносится к ридеру, ридер читает UID карты и, если идентификатор карты находится в памяти, разрешает проход. В том случае, когда ридер имитирует работу брелока iButton, реакция охранного контроллера будет такой же, как если бы к приёмному терминалу коснулись брелком-таблеткой.

Ридер поставляется в различных исполнениях, в зависимости от требований заказчика: по договорённости с заказчиком на плату ридера может устанавливаться *энергонезависимая память* для увеличения объёма памяти ACS-ключей (допустимы типы 24C32, 24C64, 24C128, 24C256, 24C512), *драйвер интерфейса RS-485* (MAX3483), *источник DC/DC* MC7805 для питания ридера нестабилизированным напряжением 8В…15В, *силовой транзистор* IRLL014 для силового управления электрозамком. В данном документе приводится описание максимальной конфигурации ридера.

Подключение

Ридер может быть подключен к управляющему контроллеру по интерфейсам USB, UART, RS485 или iButton, а также может быть использован как автономное устройство. Ридер может управлять исполнительным механизмом (вывод **SW**), к ридеру могут быть подключены дополнительные кнопочные выключатели, расширяющие возможности устройства.

Схема входных/выходных цепей ридера



Назначение выводов разъёма X1:
1 – корпус;
2 – вывод для подключения к терминалу iButton / в режимах **ACS**/**SRD** кнопка **OPEN**;
3 – выход данных UART TxD (+3.3V max, Push-Pull);
4 – вывод A**(+)** RS485;
5 – вывод B**(-)** RS485;
6 – вход данных UART RxD (+5V max);
7 – выход управления внешним устройством;
8 – питание ридера +5В.

Разъём X2 – стандартный разъём mini-USB.

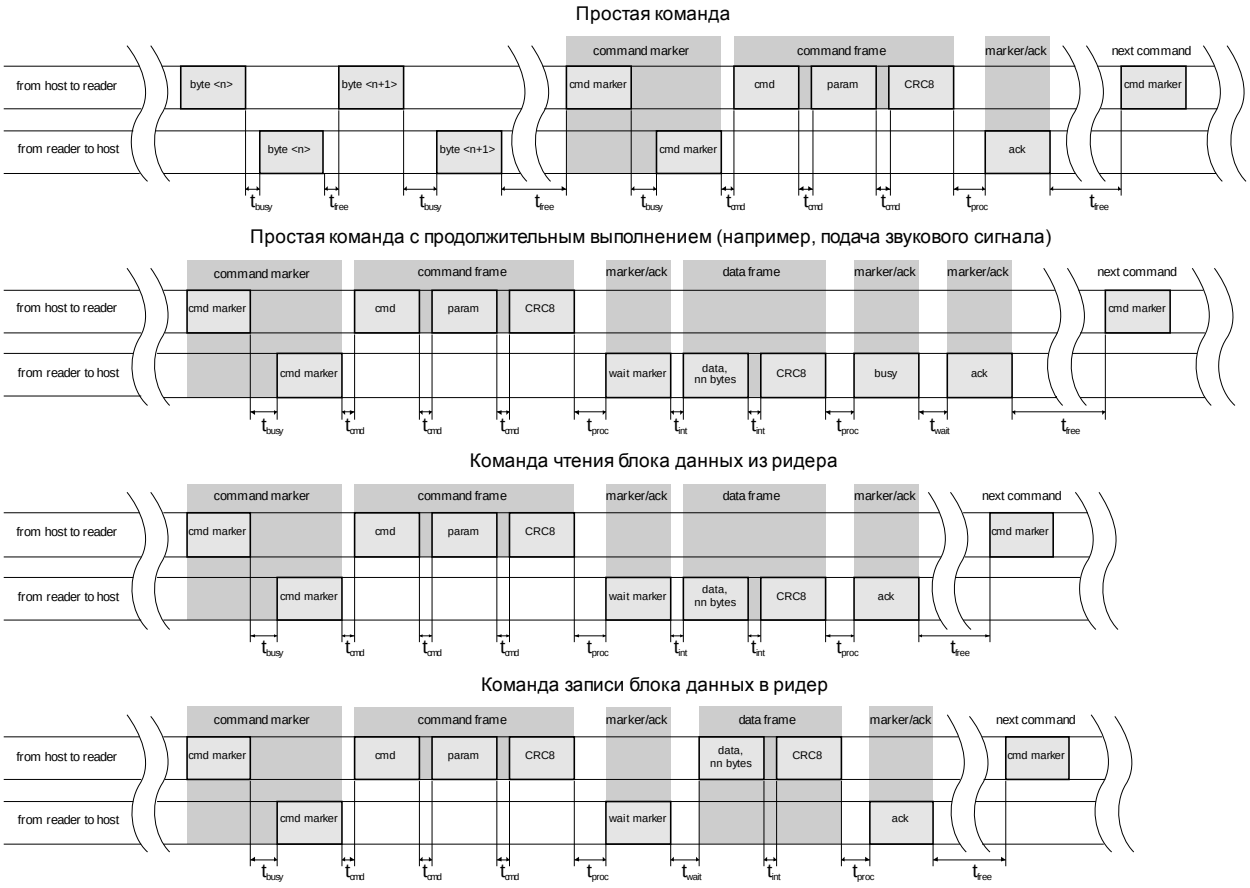
Подключение SCL D3 к GND в режиме ACS активирует режим записи мастер-карты.
Подключение SDA D3 к GND в режиме ACS равносильно нажатию кнопки OPEN.
J1 – подключение терминатора RS485.
J3 – технологическая перемычка.
J4 – при замкнутой перемычке, активен режим RS485, при разомкнутой – UART; при использовании RS485, UART интерфейс следует отключить от ридера.

Описание протокола обмена

Ридер имеет 2 интерфейса для управления и обмена данными: USB и UART. Оба интерфейса равнозначны для устройства, но когда активен один интерфейс, другой переводится в неактивное состояние. Приоритетным является USB-подключение. Протокол обмена по UART допускает подключение к симплексным линиям передачи, например RS485.

При обмене как по USB, так и по UART используется потоковое соединение (stream). Обмен данными с ридером ведётся посредством кадров. Кадры(frames) обмена бывают командными (command frame) и кадрами данных (data frame). Проверка целостности кадров производится с помощью контрольной суммы.

Общая структура обмена



t_{free} – время между передачей произвольных байт или команд, $0 < t_{free} < \infty$.

t_{busy} – время задержки передачи эха, $0 < t_{busy} < 0.6$с.

t_{cmd} – интервал между передачей байт команды, $0 < t_{cmd} < 0.3$с.

t_{proc} – время выполнения команды, $0 < t_{proc} < 0.6$с.

t_{int} – интервал между байтами, когда пауз не требуется, $0 < t_{int} < 0.3$с.

t_{wait} – время ожидания маркера, $0 < t_{wait} < 0.3$с, если не было маркера ‘**B**’;
после прихода маркера ‘**B**’ $0 < t_{wait} < 2.0$с.

cmd_marker равен ‘**U**’.
Значения **wait_marker**: ‘**R**’- ready, ‘**D**’- data, ‘**B**’- busy, **Q** – request, ‘**+**’- ack, ‘**-**‘ - nack.
CRC8 для командного кадра равен сумме по модулю 2 включая маркер!
CRC8 для кадра данных равен сумме по модулю 2 всех байт данных, исключая маркер!

В режиме ожидания команды ридер принимает байт данных через активный интерфейс (UART/USB) и посылает принятый байт обратно, т.е. реализует “эхо” обмена. По наличию-отсутствию эха можно судить о подключении ридера к линии UART. Поскольку ридер может быть занят выполнением какой-либо операции, эхо может быть передано не сразу, а с задержкой **tbusy**. Теоретически, максимальное время задержки выдачи эха равно максимальному времени выполнения самой длительной операции – 0.6с.

Основная идея командного протокола обмена в том, что любая команда состоит ***не более чем из трёх*** кадров обмена: обязательного командного кадра и двух необязательных - кадра передачи данных и кадра приёма данных. Подробности протокола обмена можно прочитать далее или посмотреть пример реализации в исходных текстах.

Для передачи команды используется код команды и параметр (командный кадр). Если этого недостаточно, то параметры передаются в блоке данных (кадр данных). Приём данных может происходить только посредством приёма кадра данных. Передача в направлении ридера обозначается ‘**T**’ – ***T****ransmit*(передача в ридер), приём данных от ридера обозначается символом ‘**R**’ – ***R****eceive*(приём данных из ридера). Тип кадра обозначается символом ‘**C**’- ***C****ommand*, если это командный кадр, ‘**D**’ – ***D****ata*, если это кадр данных. Маркеры в записи не отображаются для упрощения синтаксиса.

Упрощённая структура обмена

TC: командный кадр	
команда	параметр

TD: кадр данных
данные

RD: кадр данных
данные

Обмен в направлении хост-ридер всегда начинается с командного кадра, т.е. в команде всегда есть командный кадр. После приёма маркера команды ридер переходит в режим приёма командного кадра: код команды, параметр и контрольная сумма принимаются без выдачи эха. Если интервал между байтами команды $\mathbf{t_{cmd}}$ превышает 0.3с, режим приёма командного кадра сбрасывается, и ридер посылает символ ошибки передачи ‘**-**‘. После этого ридер переходит в режим ожидания команды.

Простые команды без параметров или команды с одним параметром могут быть переданы посредством командного кадра, однако, команды, которые требуют передачи или приёма данных включают в обмен кадры данных. Если команда требует передачи блока данных в сторону ридера, то ***кадр данных для передачи ридеру всегда будет следовать после командного кадра***, а ***кадр принимаемых от ридера данных будет последним***! Т.е. возможны следующие последовательности протокола: **TC**, **TC-TD**, **TC-RD**, **TC-TD-RD**. Последовательность вида **TC-RD-TD** – невозможна!

Общий алгоритм обмена данными с ридером следующий:

1) инициализация счётчика попыток обмена с ридером (обычно 6 попыток);
2) посылка байта **маркера команды** (**cmd_marker** = ‘**U**’);
3) ожидание эха в течение таймаута $\mathbf{t_{busy}}$<0.3с; если эха нет, и счётчик попыток обмена не исчерпан, то делаем паузу на время $\mathbf{t_{cmd}}$ =0.3с, затем переход к п.2, иначе выход с ошибкой потери связи;
4) передача байтов командного кадра **без ожидания эха**;
5) ожидание символа (маркера) в течение таймаута $\mathbf{t_{wait}=t_{proc}}$<0.3с; в зависимости от синтаксиса команды это может быть

   маркер ‘**D**’ – “**data**”, начало передачи блока данных от ридера хосту;

маркер ‘**R**’ – “**ready**”, готовность ридера к приёму блока данных из хоста;
маркер ‘**B**’ – “**busy**”, указание на увеличение таймаута до $\mathbf{t_{wait}}$=1.6с;
маркер ‘**+**’ – “**ack**”, подтверждение успешного выполнения команды, выход;
маркер ‘**-**’ – “**nack**”, указание на ошибку выполнения команды, выход;

6) после приёма маркера ‘**B**’, хост ожидает следующего маркера в течение $\mathbf{t_{wait}}$<2.0с, затем переход к п.5;
после приёма маркера ‘**D**’, хост принимает кадр данных и переходит к п.5;
после приёма маркера ‘**R**’, хост передаёт кадр данных и переходит к п.5;
если за время таймаута пришёл ошибочный символ или вообще ничего не пришло – выход с ошибкой синхронизации.

Управление ридером может быть сведено к вызову всего ***одной*** функции, которая возвращает код ошибки:

DWORD link_packet(BYTE cmd, BYTE parm, PBYTE IN outbuf, WORD outlen,PBYTE OUT inbuf, WORD inlen);

Синтаксис вызова простой команды, передаётся только параметр:

if ((err=link_packet(0xF4, 0x02, NULL, 0, NULL, 0)) != LERR_SUCCESS) goto err_proc;

Функция *link_packet* с такими параметрами вызовет передачу только командного кадра и ожидание приёма маркера подтверждения выполнения команды.

Синтаксис вызова команды с приёмом блока данных длиной **6 байт** в буфер *inbuf*:

if ((err=link_packet(0xFF, 0x00, NULL, 0, inbuf, 6)) != SUCCESS) goto err_proc;

Функция *link_packet* с такими параметрами вызовет передачу командного кадра и последующий приём блока данных.

Синтаксис вызова команды с передачей блока данных длиной **6 байт** из буфера *outbuf*:

if ((err=link_packet(0xEC, 0x00, outbuf, 6, NULL, 0)) != SUCCESS) goto err_proc;

Функция *link_packet* с такими параметрами вызовет передачу командного кадра, а затем передачу блока данных.

Синтаксис вызова команды с передачей блока данных длиной **2 байта** из буфера *outbuf* и последующим приёмом блока данных длиной **1 байт** в буфер *inbuf*:

if (err=link_packet(0xDB, 0x02, outbuf, 2, inbuf, 1)) != SUCCESS) goto err_proc;

Функция *link_packet* с такими параметрами вызовет передачу командного кадра, передачу блока данных и последующий приём блока данных.

В пример обмена с ридером на языке C/C++ приводится в исходных текстах демонстрационных программ, поставляемых вместе с ридером.

Особенности программирования

Все исходные тексты взаимодействия с ридером открыты и свободны для использования. Низкоуровневые функции ввода-вывода сведены в файл ***rd0xAB.cpp***. На базе файла ***rd0xAB.cpp*** написаны функции для работы с картами Mifare, которые сведены в файл ***mf0xAB.cpp***.

Все примеры для Windows компилируются в среде WinDDK для 32х и 64х разрядных процессоров, но примеры также могут быть без изменений перенесены в среду VC ≥ 6.0.

Для удобства тех, кто не использует VC и WinDDK, файлы ***rd0xAB.cpp*** и ***mif0xAB.cpp*** откомпилированы в одну DLL – ***rd0xAB.dll***, что позволяет использовать ридер без перекомпилирования исходников.

Примеры для Linux, в силу особенностей этой операционной системы, распространяются только в виде исходных текстов консольных приложений на С.

Взаимодействие с ридером строится на базе 7ми функций;
link_hidopen - открытие ридера, подключенного по USB;
link_comopen - открытие ридера, подключенного по COM/TTY (UART/RS-485);
link_close - закрытие ридера;
link_echobyte - пинг ридера: ридер ответит, если находится в режиме ожидания;
link_rqwait - ожидание и приём запроса от ридера на проход карты;
link_rqack - передача ответа на запрос прохода по карте;
link_packet - основная командная функция для работы с ридером.

Синтаксис функций одинаков для Windows и Linux, кроме функции подключения к ридеру по UART-интерфейсу *link_comopen*, которая в Windows в качестве входного параметра использует номер COM-порта (1…255), а в Linux – имя TTY-устройства.

Для организации обмена по USB ридер использует протокол обмена USB HID устройства. Подобный подход позволяет организовать программный интерфейс с ридером без установки специальных драйверов, т.е. программный интерфейс организуется с помощью встроенных в систему функций. Результатом подобного подхода является то, что ридер поддерживается почти платформами и версиями ОС, которые поддерживают работу с USB HID устройствами, например, все версии Windows (XP…W10), все версии Windows CE/RE, Linux (включая embedded-версии и Android), FreeBSD, MacOS и т.п.. В POSIX системах взаимодействие с ридером происходит через сервис, предоставляемый стандартным драйвером ***hidraw***. В дополнение к HID, в Windows и Linux реализован интерфейс взаимодействия с ридером через функции библиотеки ***libusb-1.0***.

В режиме USB ридер поддерживает автоматическое переключение спецификаций HID устройства: **HID_Interrupt** и **HID_Control**, несмотря на то, что в дескрипторе устройства объявлена только поддержка режима **HID_Control**.

Режим **HID_Interrupt** включен в ридере по умолчанию и позволяет организовать непрерывный приём данных через приёмный тред или процесс путём считывания данных из внешнего блокового устройства (для Windows данные в режиме **HID_Interrupt** считываются функцией *ReadFile*). Запись данных в ридер осуществляется путём записи данных во внешнее блоковое устройство (для Windows данные в режиме **HID_Interrupt** записываются функцией *WriteFile*). Если процесс приёма данных прерывается (например, используется не непрерывный, а периодический вызов *ReadFile*), то данные могут быть утеряны. Для организации обмена данными с ридером путём периодического опроса следует использовать режим **HID_Control**.

Режим **HID_Control** включается после обмена стандартными пакетами **GET_REPORT** или **SET_REPORT:** для Windows это функции *HidD_GetInputReport* и *HidD_SetOutputReport*, для Linux – ***ioctl(…, HIDIOCGFEATURE, …)*** и ***ioctl(…, HIDIOCSFEATURE, …)***.

После активации режима **HID_Control**, ридер накапливает данные для передачи хосту в буфере и выдаёт очередную пачку данных только по стандартному запросу **GET_REPORT**: этим обеспечивается непрерывность потока данных при периодическом опросе ридера. Если периодичность опроса ридера будет недостаточно высокой, то внутренний кэш ридера может переполниться. Объёмы входного и выходного буфера ридера одинаковы - 64 байта.

В случае отсутствия в течение 5с запросов **GET_REPORT** или **SET_REPORT** ридер из режима **HID_Control** перейдёт в режим **HID_Interrupt**. Досрочно перевести ридер в режим **HID_Interrupt**, можно подав команду блоковой записи (для Windows функция *WriteFile*). Низкоуровневые функции ввода-вывода в файле ***rd0xAB.cpp*** взаимодействуют с ридером только в режиме **HID_Control**. Следует учитывать, что не все версии Windows поддерживают «необъявленную» работу в режиме **HID_Interrupt**.

При подключении к ридеру сразу следует дать команду **CMG_VER** для получения аппаратной версии ридера и установки протокола обмена. Сразу после получения версии, рекомендуется перевести его в режим **DIR** без записи режима в энергонезависимую память, чтобы избежать рассинхронизации обмена, если ридер настроен на выдачу запросов при прикладывании карты. После этого можно безопасно работать с ридером, не боясь разрыва соединения. Подробности программной реализации обмена с ридером можно посмотреть в исходных текстах демонстрационных примеров.

Сборка примеров для OpenWrt сделана на основе примеров для Linux с интерфейсом на базе функций библиотеки ***libusb***. В связи с тем, что OpenWrt развивающийся нестабильный проект, функции ядра то работают, то неработают от релиза к релизу проекта: примеры полностью работоспособны в стабильных ветках любых POSIX-проектов.

Принципы функционирования

Ридер может управлять замком, подавать звуковые сигналы, обеспечивает световую индикацию своего состояния. Настройки ридера сохраняются в энергонезависимой памяти. Ридер обеспечивает работу в 4х режимах:

0) режим **ACS** - **A**ccess **C**ontrol **S**ystem, или СКУД (система контроля и управления доступом); ридер работает как независимое устройство, если за заданный промежуток времени не приходит разрешение или запрет прохода; идентификаторы карт хранятся в памяти ридера, т.е. количество пропусков – ограничено, новые пропуска надо вносить в память ридера;
1) режим **SRD** - **S**imple **R**ea**d**, простое чтение данных; в этом режиме читается заданная группа секторов, и считанные данные пересылаются внешнему контроллеру, который принимает решение разрешить проход или нет; возможна автономная работа ридера, но только в режиме авторизации; количество пропусков в этом режиме не ограничено, для создания новых пропусков перепрограммирование ридера не требуется;
2) режим **IBE** – **iB**utton **e**mulation, режим имитации брелоков-таблеток iButton DS1990A и подобных, широко используемых в охранных сигнализациях, на бензоколонках и т.п.; ридер в этом режиме может работать как автономное или как управляемое внешним контроллером устройство, заменяющее брелоки-таблетки на современные бесконтактные карты;
3) режим **DIR** – **Dir**ect access, режим, обеспечивающий прямой доступ к картам на низком уровне.

Запрограммировать ридер для использования в автономных режимах, не вдаваясь в подробности программирования, можно с помощью программы ***ccr.exe***.

Режимы **ACS** и **SRD** предназначены для построения систем авторизованного доступа: **ACS** позволяет организовать пропускную систему по UID карты, режим **SRD** позволяет организовать многоуровневую проверку: UID, ключ авторизации, а также провести проверку считанных данных.

Режим **IBE** позволяет использовать вместо легко копируемых контактных ключей *iButton* и бесконтактных *eMarine*, современные некопируемые ключи стандартов ISO-14443 и Mifare: ридер подключается прямо к контактору.

Для “настольного” использования ридера предусмотрена смена режима работы и подачи звука без сохранения этих настроек в энергонезависимой памяти - это позволяет избежать частой перезаписи flash-памяти и удобно при написании программ для ридера.

При управлении ридером по RS485 и UART следует правильно установить скорость обмена. Если скорость обмена неизвестна, то проще всего её изменить, подключив ридер по USB-интерфейсу, где этот параметр безразличен. Если же такая возможность отсутствует, то следует пробовать подключаться к ридеру на скоростях 1200, 2400, 4800, 9600, 14400, 19200, 38400, 56000, 57600, 115200 бит/с. По умолчанию и после сброса настроек скорость обмена равна 9600 бит/с.

Жёлтый светодиод (Yellow LED) отражает состояние вывода управления замком: включенный жёлтый индикатор указывает на то, что замок открыт (вывод **SW** находится в активном состоянии). Зелёный светодиод (Green LED) – индикатор гаснет при выполнении операции.

Динамичные режимы работы обоих индикаторов отражают работу ридера в разных режимах: при имитации iButton на короткий промежуток времени меняет состояние жёлтый индикатор, при вводе мастер-карты одновременно мигают оба светодиода, и т.п..

В ридере предусмотрена установка полярности сигнала управления замком, а также времени открытия замка. Нулевое время удержания замка открытым, соответствует триггер-режиму, когда каждое новое прикладывание карты приводит к изменению состояния замка на противоположное.

Управление и изменение настроек ридера производится командным протоколом функцией *link_packet*, т.е. обычно ридер является slave-устройством. Однако, в режимах **ACS**, **SRD**, **IBE** ридер может быть настроен на выдачу запросов при поднесении карты, выступая инициатором обмена: приём запросов производится с помощью *link_rqwait*, а ответ на запросы – с помощью *link_rqack*. Обслуживание запросов ридера показано в примерах ***acs-demo***, ***ibe-demo***, ***srd-auth***, ***srd-data***, ***srd-ultra***.

Настройки ридера объединены в группы, которые можно читать и записывать. Поскольку энергонезависимая память имеет хоть и большое, но ограниченное количество циклов перезаписи, рекомендуется вначале считывать настройки, а переписывать их в случае отличия от требуемых. Наиболее часто изменяемые настройки, режим работы ридера и флаг подачи звукового сигнала, имеют “дублёров” в RAM (ОЗУ). Для оперативного изменения режима ридера и режима подачи звукового сигнала, при использовании ридера в качестве “настольного” устройства, рекомендуется пользоваться RAM-дублёрами (см. команды **CMG_MOD**, **CMS_MOD**).

В режиме **ACS** ридер использует внутреннюю базу ключей (ACS-ключей), которым разрешён проход. Длина записи ключа – 8байт. Память для хранения ACS-ключей организована странично с размером страницы 512 байт. На одной странице помещается 512 / 8 = 64 ключа. Нумерация ключей начинается с нуля с самой младшей страницы с самого младшего адреса. Ключ с номером 0 – мастер-ключ. Ключ считается удалённым, когда все его байты равны 0FFh.

Распределение памяти ACS-ключей

Страница 0

смещение	№ ключа	структура данных
+000h	0	мастер-ключ, позволяет прописывать новые ключи
+008h	1	ключ, если поле пустое, то все байты равны 0FFh
+010h	2	ключ, если поле пустое, то все байты равны 0FFh
…	…	…
+1F8h	63	ключ, если поле пустое, то все байты равны 0FFh

……

Страница n=max-1

смещение	№ ключа	структура данных
+000h	0+n*64	ключ, если поле пустое, то все байты равны 0FFh
+008h	1+n*64	ключ, если поле пустое, то все байты равны 0FFh
…	…	…
+1F8h	63+n*64	ключ, если поле пустое, то все байты равны 0FFh

Страница n=max

смещение	№ ключа	структура данных
+000h	0+n*64	ключ, если поле пустое, то все байты равны 0FFh
+010h	2+n*64	ключ, если поле пустое, то все байты равны 0FFh
…	…	…
+1F8h	63+n*64	***ключ для ROM, сигнатура для внешней NVM***

При отсутствии внешней памяти (ROM-режим) для записи ключей используется весь доступный объём памяти. При наличии внешней памяти (NVM-режим) последний ключ на последней странице не будет записан. Узнать установлена или нет внешняя память и полный объём памяти можно с помощью команды **CMG_VER** (см. описание команд). Память ACS-ключей может быть стёрта командой **CMF_KME**, при этом мастер-ключ не стирается.

Для быстрого чтения базы ACS-ключей в ридере предусмотрена команда получения последней занятой ключами страницы (см. **CMF_KME**). Узнав последнюю занятую страницу, можно считать память только до занятой страницы, не читая память ACS-ключей целиком.

При работе с картами Mifare Mini, Mifare Classic, Mifare Plus. Smart MX и другими, поддерживающими авторизованный доступ по алгоритму Crypto1, требуется сохранение ключей авторизации. В ридере предусмотрена энергонезависимая память на 84 ключа авторизации: Authorization Key memory или AK-память.

В AK-память можно записывать ключи авторизации, а вот возможности чтения этих ключей не предусмотрено – это стандартный механизм обеспечения безопасности.

Распределение памяти ключей авторизации (AK-памяти)

№ ключа	структура данных
0	***AK0, RAM-ключ, в энергонезависимой памяти не сохраняется***
1	ключ авторизации AK1, сохраняется в ROM
2	ключ авторизации AK2, сохраняется в ROM
…	…
84	последний ключ авторизации AK84, сохраняется в ROM

Из соображений безопасности при работе с картами, требующими авторизации, никогда не указывается содержимое ключа авторизации. Вместо этого указывается номер ячейки памяти, где хранится ключ. Такой подход обеспечивает высокую безопасность при написании приложений.

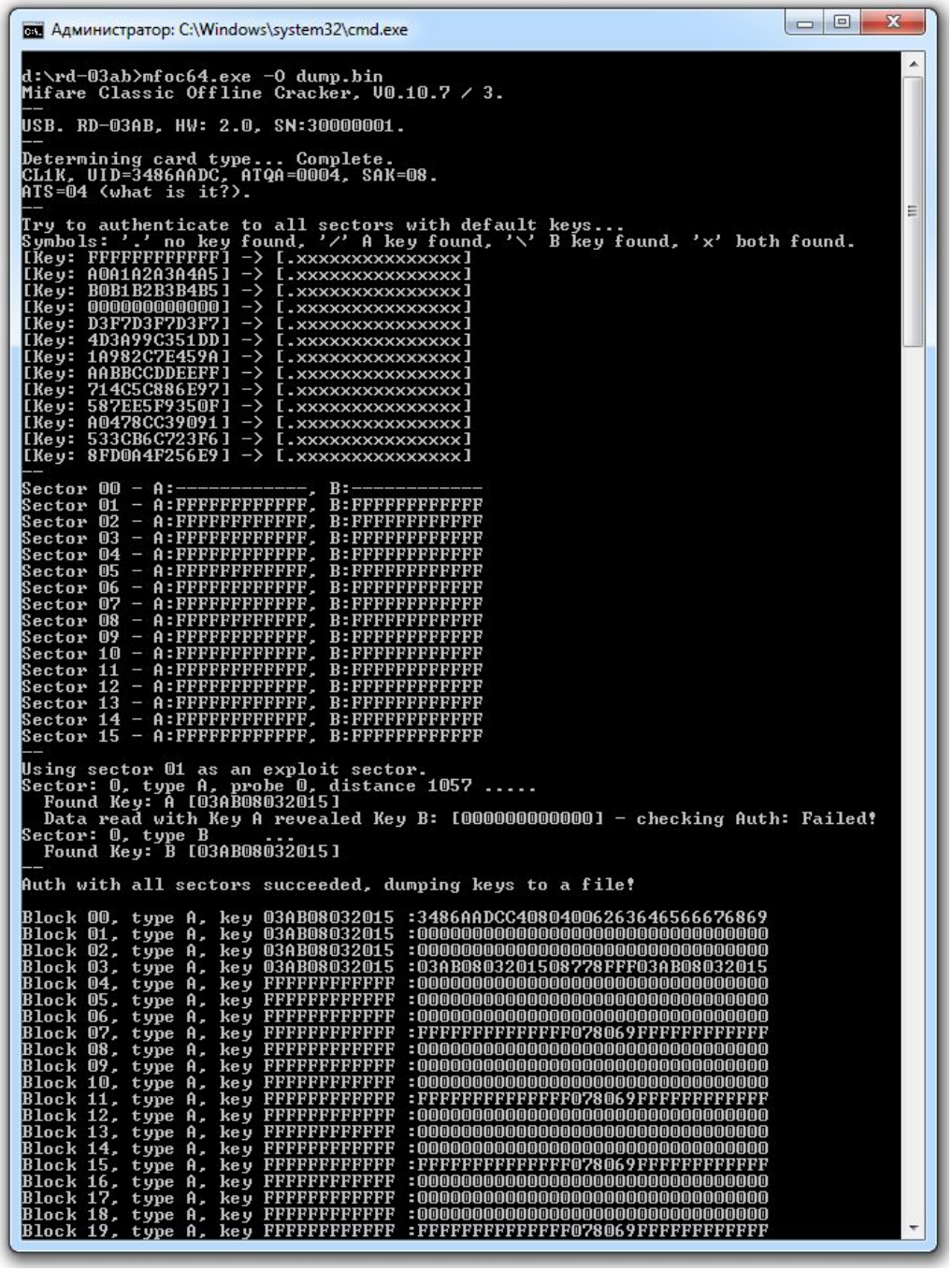
Например, требуется написать транспортное приложение, количество поездок хранится на карте. Как дать разработчику приложения доступ к данным на карте не раскрывая ключа авторизации? Очень просто. Для этого ридер передаётся в отдел безопасности, где в ридер прошиваются ключи. Например, в 1ю ячейку – ключ доступа к транспортным блокам, во 2ю – ключ доступа к финансовым блокам, в 3ю – ключ доступа к блокам с именем и фамилией владельца, и т.д.. Причём работники отдела безопасности сами могут и не знать ключей авторизации, если их прошивает специализированный софт. Далее ридер возвращается разработчику, который знает, что для того, чтобы считать транспортную информацию из заданных блоков, надо установить авторизацию по 1му ключу, для того, чтобы считать количество денег на карте – следует установить авторизацию по 2му ключу. При этом само значение ключа разработчику неизвестно, а чтение ключа из карты – невозможно.

При оперативной работе с ключами авторизации, каждый раз переписывать ключ в энергонезависимой памяти неудобно: сохранение ключа авторизации в ROM требует времени. Для оперативной работы ключ следует сохранять в ячейку 0 – этот ключ сохраняется в RAM (ОЗУ) и пропадает при обесточивании ридера.

В ридере предусмотрены команда подачи звука (6 звуковых сигналов) и команда прямого управления замком. Состояние управления замком не сохраняется в энергонезависимой памяти, и после повторной подачи питания или смене режима ридера, замок устанавливается в закрытое состояние.

В режиме **DIR** набор команд ридера расширяется: добавляются специальные команды от **CMG_BUF** до **CMF_LCMD** для чтения/записи внутренних буферов, проверки и декодирования сырых данных, посылки специальных команд и т.п.. В режиме **DIR** можно включать/выключать несущую, вручную устанавливать уровень фазового детектора, формировать любую последовательность команд. Именно в этом режиме работают программы взлома ключей ***mfoc*** и ***mfcuk***.

Скриншот mfoc.exe



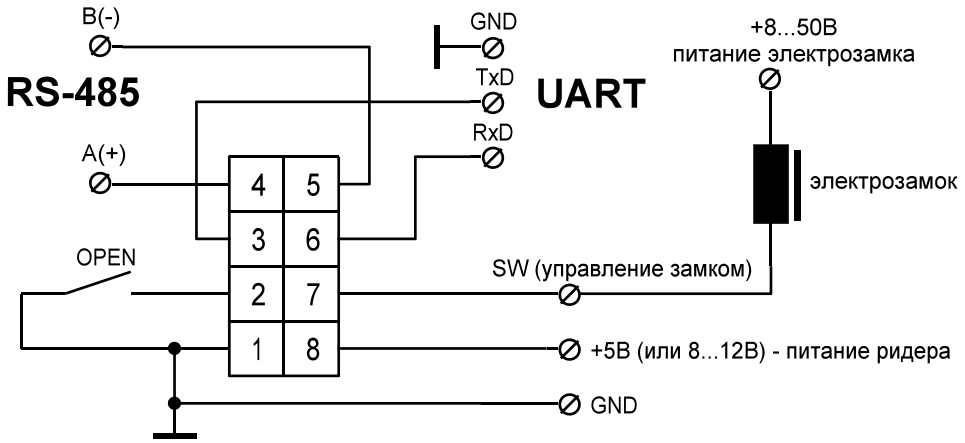
На скриншоте ***mfoc.exe*** видно, что ридер, благодаря конструктивным особенностям, работает на коротких дистанциях, что сильно ускоряет процесс подбора ключей даже при использовании медианного фильтра. ***Mfoc*** адаптирована для работы с ридером под Windows и Linux без использования сторонних библиотек – ***mfoc*** не использует ни ***libnfc***, ни ***libusb***: полный функциональный набор обеспечивается стандартными ***rd0xAB.cpp*** и ***mif0xAB.cpp***.

Режим ACS

В режиме **ACS** ридер позволяет организовать систему контроля и управления доступом, когда вход в контролируемую зону и/или выход из контролируемой зоны осуществляется по пропускам-картам. Решение пропускать карту или нет, может принимать как внешний контроллер, так и сам ридер по внутренней базе данных. В режиме **ACS** предполагается следующий алгоритм работы:

1) считывается UID, ATQA, SAK поднесённой карты, проверяется, является ли карта перезаписываемой (RW-свойство карты); ATQA и SAK могут быть использованы для определения типа карты;
2) из считанных данных формируется запрос внешнему контроллеру, который должен решить, что делать с картой; *если посылка запроса и ожидание подтверждения неактивны, то ридер сам принимает решение о пропуске карты, руководствуясь наличием UID карты во внутренней базе ключей и разрешением работы с RW-картами*;
3) если ридеру разрешена посылка запроса, то запрос передаётся на анализ внешнему контроллеру по активному интерфейсу – UART, RS-485 или USB, а ридер переходит в режим ожидания ответа с отсчётом таймаута; *если посылка запроса и ожидание подтверждения неактивны, или установлен режим посылки запроса без подтверждения, то ридер сам принимает решение о пропуске карты, руководствуясь наличием UID карты во внутренней базе ключей и разрешением работы с RW-картами*;
4) на основании принятых данных, внешний контроллер посылает один из 3х вариантов ответа: 0 - запретить проход, 1 - разрешить проход, 2 - ридер должен пропустить карту, руководствуясь своей внутренней базой ключей и разрешением работы с RW-картами;
5) если в течение таймаута ожидания ридер принимает ответ от внешнего контроллера, то решение о пропуске карты будет принято на основании ответа; если до истечения таймаута ответ не будет принят, ридер сам примет решение о пропуске карты, руководствуясь наличием UID карты во внутренней базе ключей и разрешением работы с RW-картами.

Подключение ридера в режиме ACS



Любые настройки ридера могут быть произведены внешним контроллером по интерфейсам USB, UART или RS485, однако, некоторые оперативные манипуляции с ридером можно производить и в автономном режиме, подав только питание.

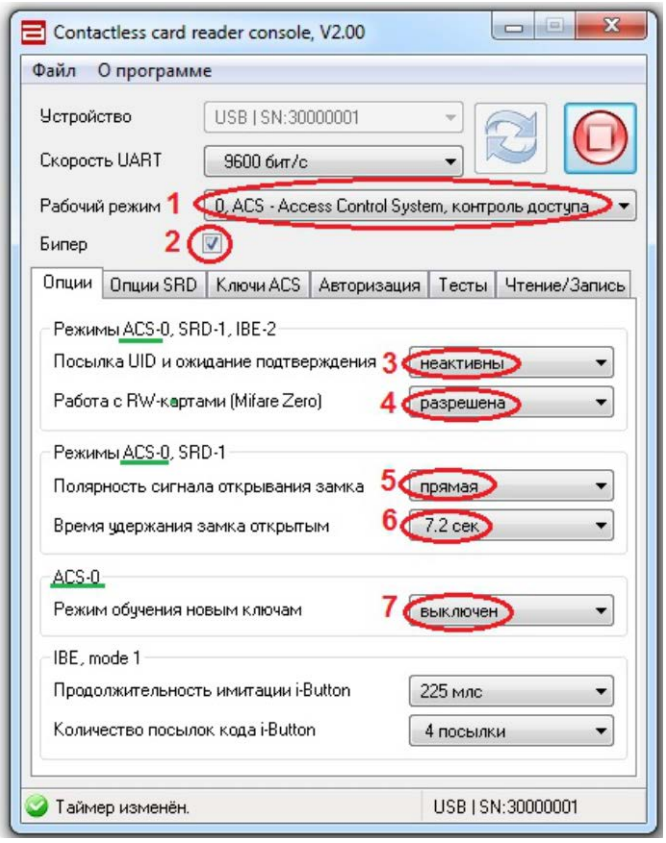
В режиме **ACS** предусмотрена работа с мастер-картой. Мастер-карта переводит ридер в режим обучения: находясь в этом режиме, ридер добавляет каждую поднесённую карту во внутреннюю базу ключей. Для выхода из режима обучения надо поднести мастер-карту ещё раз. *В режиме обучения мигает жёлтый индикатор, а замок переводится в открытое состояние.*

Открыть замок без прикладывания карты можно нажатием кнопки “Open” (удерживать в течение 1 с), а записать мастер-карту можно кратковременным замыканием вывода SCL (выв. 6 D3 или выв.15 D4) на корпус GND. В режиме записи мастер карты раздаётся звуковой сигнал, и одновременно мигают *жёлтый* и *зелёный* индикаторы. Выход из режима записи мастер-карты происходит через 10с после прикладывания карты. Замыкание вывода SDA (выв. 5 D3 или выв.16 D4) на корпус GND является эквивалентом нажатия кнопки “Open”.

При одновременном замыкании выводов SDA (или нажатие на кнопку “Open”) и SCL на корпус GND, ридер издаст трель, а затем сотрёт все **ACS**-ключи в энергонезависимой памяти, кроме мастер-ключа.

Режим ACS в примерах

Пример 1. Простейшей **ACS**-системой является домофон. UID карт, которым разрешён вход в подъезд, хранятся в энергонезависимой памяти контроллера. При поднесении карты, ридер считывает её UID, ищет совпадение в памяти и, если совпадение найдено, открывает замок двери на непродолжительное время (вывод **SW** устанавливается в активное состояние). Открытие замка может сопровождаться звуковым сигналом. Настроить ридер для работы в этом режиме можно с помощью программы ***ccr.exe***.



После подключения к выбранному ридеру с помощью ***ccr.exe***, следует установить:

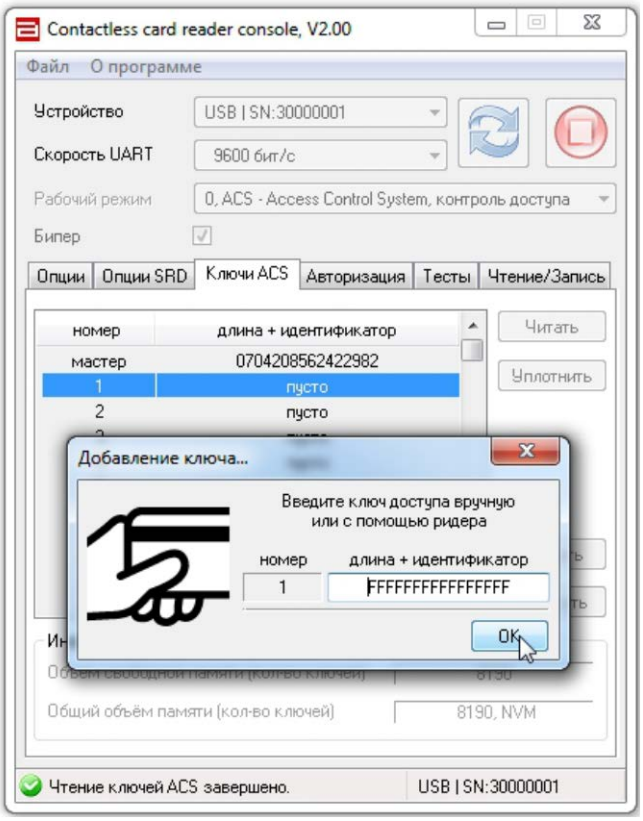
1) рабочий режим – **ACS**;
2) звуковой сигнал, если он нужен;
3) в автономном режиме ридер ничего не посылает и не передаёт;
4) разрешение прохода по RW-картам, если требуется;
5) полярность сигнала открывания замка так, чтобы в режиме ожидания дверь была закрыта;
6) время отпускания замка, за которое можно открыть дверь;
7) режим обучения, если требуется, хотя его можно активировать и позже с помощью мастер-карты.

Ключи доступа для режима ACS можно записать заранее, перейдя на соответствующую вкладку.

Для добавления нового ключа или редактирования значения старого, вначале следует считать базу ключей в память программы (кнопка “Читать”) или стереть базу ключей в памяти ридера (кнопка “Стереть”).

Удалённому ключу соответствует шестнадцатеричное значение “FF FF FF FF FF FF FF FF” (8 байт). Добавить новое значение ключа в окне диалога можно как вручную, так и поднеся новую карту к ридеру. Если карта уже была внесена в базу ключей, то она не будет добавлена второй раз, о чём уведомит звуковой сигнал.

Отметить несколько записей для произведения операций над ними можно с помощью комбинаций клавишей *Shift + стрелки*, *Ctrl + левая кнопка мыши*, а также другими стандартными комбинациями Windows.



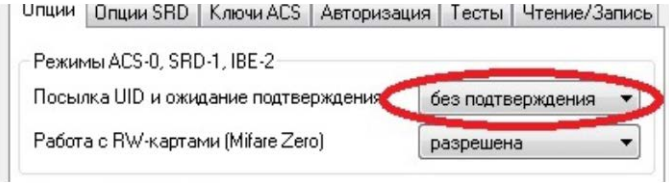
При ручном вводе значения ключа следует учитывать, что первый байт записи – это длина последующего *UID*, т.е. *может принимать значения 04, 07, 0A*. Другие значения первого байта ошибочны, за исключением FF при вводе пустой записи. Второй, третий и последующие байты – это UID карты. *Если длина UID равна 4 байтам, то последние 3 байта ключа следует заполнить нулями*. Длина записи ACS-ключа – 8 байт.

После завершения операций с ключами их следует записать в энергонезависимую память ридера (кнопка “Запись”). Внутренний алгоритм записи вначале стирает все ключи в памяти ридера, а затем записывает новые данные. Подобный алгоритм позволяет ускорить запись ключей на низких скоростях обмена.

Нажатие на кнопку “Уплотнить” приведёт к оптимизации данных в памяти программы: будут удалены повторяющиеся ключи, убраны пропуски, связанные с удалением записей, все ключи будут “подвинуты” к началу, чтобы не было пропусков. После уплотнения, ключи следует записать в память ридера.

Для корректного завершения настройки, отключаться от ридера следует нажатием кнопки с красным квадратом (“Стоп”) или закрытием окна программы ***ccr.exe***.

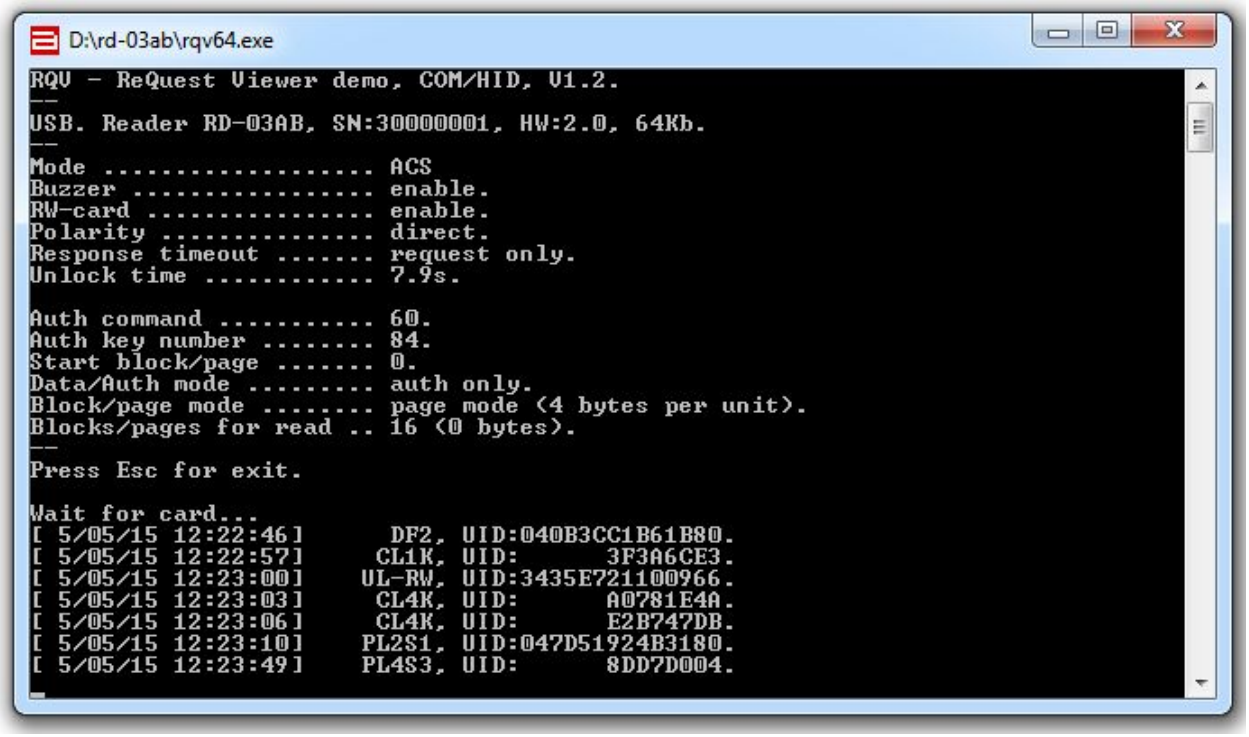
Пример 2. Для организации простой пропускной системы в магазине или офисе с учётом рабочего времени, можно воспользоваться настройками из примера 1, изменив значение настройки “*Посылка **UID** и ожидание подтверждения*” на “*без подтверждения*”. В этом режиме ридер будет посылать внешнему логгеру данные о считанных картах без ожидания подтверждения об их пропуске. К принятым данным логгеру потребуется только добавить время прохода.



Логгером может быть как простейшая программа на компьютере, когда ридер подключен через USB интерфейс, так и простейший логгер, подключенный по UART или RS-485 и сохраняющий данные на SD-карте.

Демонстрационная программа ***rqv*** показывает принцип построения программы-логгера. ***rqv*** считывает параметры ридера, и, если установлен режим передачи UID без ожидания подтверждения, выводит на экран данные поднесённых карт.

Скриншот rqv



Программа строится на циклическом вызове функции *link_rqwait(handle, buf, SZ_SCCD)*. Длина данных, передаваемых ридером в режиме **ACS**, всегда равна *SZ_SCCD* = 14 байтам: ридер передаёт описатель карты ***CD*** – ***Card Descriptor***. Если запрос успешно принят, то функция возвращает код ***LERR_SUCCESS***.

Описание структуры данных CD в буфере после успешного вызова link_rqwait

смещение	назначение	длина	структура данных
+00h	uidlen	1	бит 7 – равен 1, если RW-карта биты 6…4 – зарезервированы, равны 0 биты 3…0 – длина данных в поле uid (4, 7, 10)
01h	uid	10	UID карты, незначащие байты заполнены нулями
+0Bh	sak	1	SAK (**Select acknowledge**, см.ISO14443A), значение SAK используется при определении типа карты
+0Ch	atqa	2	ATQA (**Answer to** request **A**, см.ISO14443A), значение ATQA используется при определении типа

После считывания карты **Mifare Ultralight** uidlen = 7, uid содержит UID карты длиной 7 байт, sak = 0, atqa = 0044h (little endian).

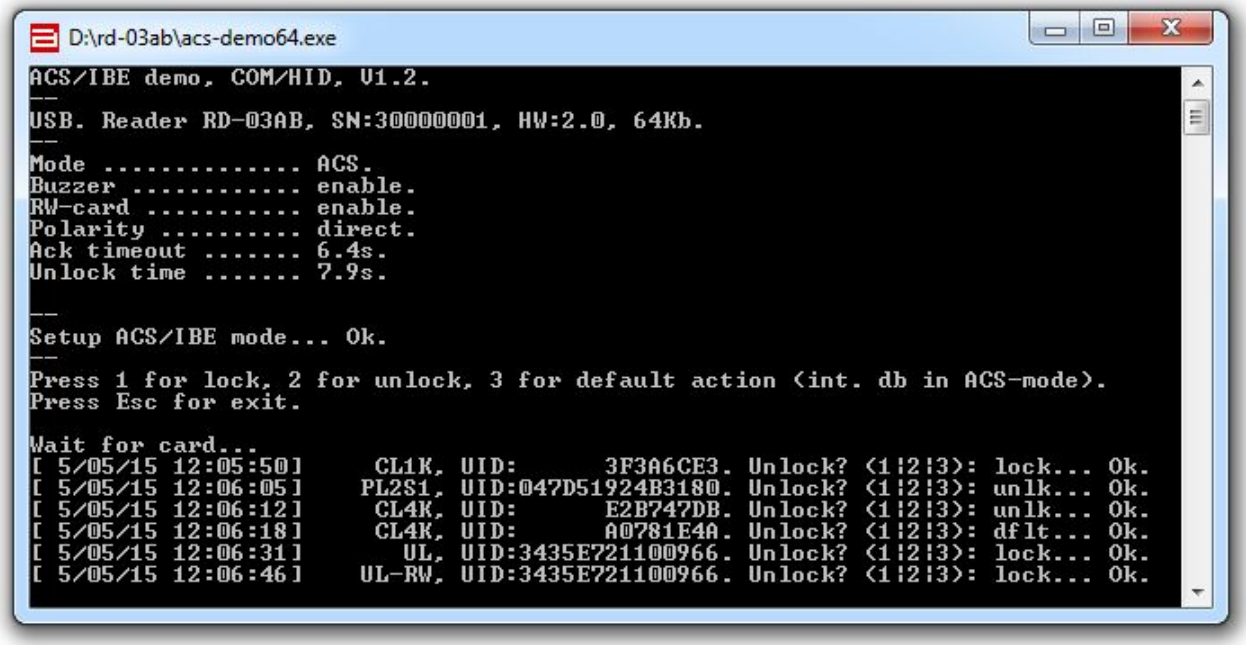
После считывания **Mifare Classic 4K** – uidlen = 4, uid содержит UID карты длиной 4 байта, sak = 18h, atqa = 0002h (little endian).

Пример 3. В классической турникетной пропускной системе турникет, как правило, открывается после прикладывания пропуска к считывателю. Однако ничто не мешает передать карту-пропуск другому лицу, поэтому на такой пропуск обычно наносится фотография владельца, чтобы при подозрениях охранник мог сверить личность на пропуске. Но, ничто не мешает переклеить фотографию на пропуске…

Ридер, в режиме **ACS** с подтверждением запроса, позволяет организовать пропускную систему с внешним решателем и выводом фотографии владельца пропуска на экран монитора охраны. В случае потери связи с внешним контроллером, ридер сам примет решение о пропуске карты, руководствуясь наличием UID карты во внутренней базе ключей и разрешением работы с RW-картами. Также, на случай аварии, вместо прописывания всех возможных карт-пропусков во внутренней базе ридера, можно прописать только несколько “аварийных” пропусков охраны, чтобы охранники могли провести работников через турникет под своим присмотром.

Демонстрационная программа ***acs-demo*** показывает принцип построения такой системы, исходники находятся в каталоге ***acs-ibe***.

Скриншот acs-demo



Программа инициализирует ридер для работы в **ACS**-режиме с подтверждением прохода карты внешним контроллером и входит в петлю ожидания запросов от ридера. Принятый запрос выводится на экран и пользователю предлагается принять решение о пропуске карты за время таймаута (6.0с): нажатие **1** – блокировка прохода, нажатие **2** – разрешение прохода (замок открывается на 7.2с), нажатие **3** – разрешение прохода по базе ключей ридера.

В реальном программном обеспечении на основании принятых от ридера данных можно выводить фото на экран монитора и разрешать проход, если карта с полученными данными присутствует в базе ключей компьютера; а при появлении RW-карты с данными, совпадающими с данными реального пропуска включать “тихую” тревогу.

Программа строится на циклическом вызове функции *link_rqwait(handle, buf, 14)*. Длина данных, передаваемых ридером в режиме **ACS**, всегда равна 14 байтам. Если запрос успешно принят, то функция возвращает код ***LERR_SUCCESS***.

Структура данных в буфере после вызова *link_rqwait* подробно описана в предыдущем примере.

После посылки запроса, ридер ждёт ответа в течение времени таймаута (“Ack timeout” на скриншоте ***acs-demo***). За время таймаута программа должна послать ответ-подтверждение на запрос ридера, иначе ридер примет решение о пропуске карты сам. Ответ ридеру даётся с помощью команды ***link_rqack(handle, ack)***, где допустимые значения **ack** - 0 (нет прохода), 1 (разрешение прохода), 2 (по базе ключей ридера). При вычислении времени таймаута в программе, следует учитывать время передачи запроса и ответа, т.к. запрос и ответ не передаются мгновенно. Рекомендуемую формулу расчёта таймаута можно найти в исходнике ***acs-ibe.cpp***.

Пример 4. Один вариантов построения профессиональной пропускной системы – пропускная система с решателем на внешнем контроллере. Эта система подходит для больших офисных комплексов, предприятий, складов с большим числом ридеров (ридер может быть установлен на каждом помещении), когда надо быстро ограничить или разрешить доступ в группу помещений или во всех помещениях сразу.

Для объединения ридеров в общую сеть используются Wi-Fi роутеры P3 (Ralink) или WR-703N (TP-Link) с изменённой прошивкой. Роутеры могут соединяться между собой проводной сетью или через Wi-Fi с шифрованием.

Ридер RD-03AB, подключенный к Wi-Fi / LAN модулю

К одному роутеру через USB-хаб может быть подключено несколько ридеров. При приёме запроса от ридера, программа в роутере добавляет к пришедшему запросу заголовок в виде номера ридера от которого пришёл запрос, и отправляет в сеть широковещательный UDP пакет. Сервер, ответственный за решение о пропуске карты, принимает пакет, записывает его в лог и даёт подтверждение прохода таким же широковещательным пакетом, указав в заголовке номер ридера.

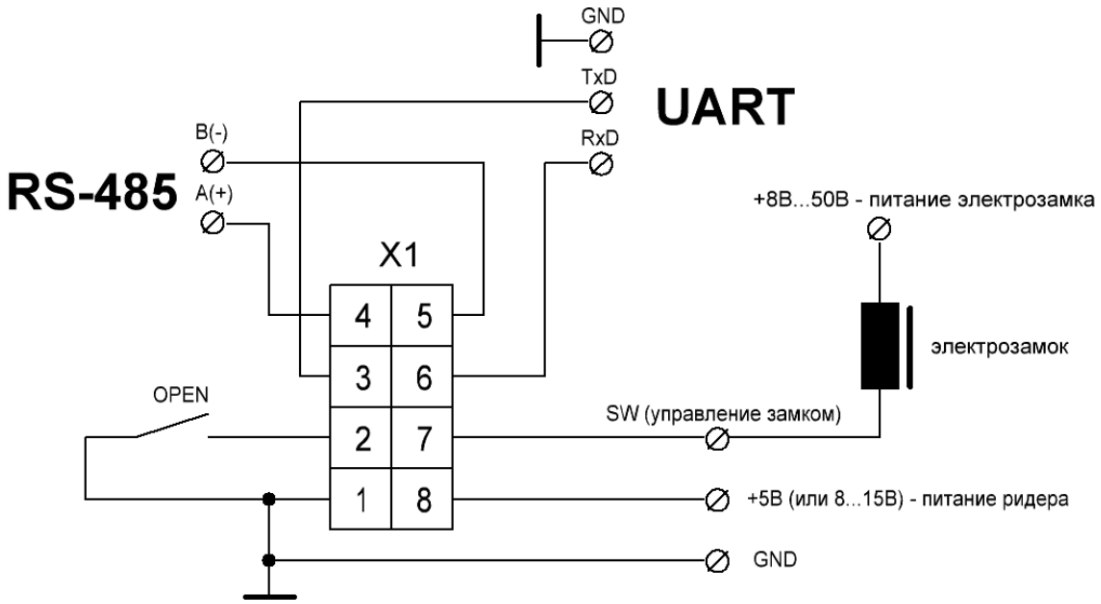
Построение пропускной системы на базе Wi-Fi модулей позволяет объединить множество ридеров в одну защищённую беспроводную сеть, которая отличается низкой стоимостью и высокой защищённостью от взлома. Подробности программирования коммерческих пропускных систем в данном документе не рассматриваются.

Режим SRD

Режим **SRD** похож на режим **ACS**, однако, в отличие от режима **ACS**, позволяет организовать профессиональную пропускную систему с внешним контроллером с несколькими уровнями защиты: 1) проверка UID и типа карты, 2) проверка ключа авторизации, 3) проверка данных, записанных на карту.

При организации автономной пропускной системы, ридер в режиме SRD позволяет организовать пропуск по ключам авторизации, не учитывая UID, т.е. подобная *система не требует прописывания карт-пропусков в память ридера*: возможность прохода по карте зависит от правильности ключа авторизации, записанного в карту; ключ авторизации считать из карты невозможно.

Подключение ридера в режиме SRD



В режиме **SRD** используется на 4 байта больше дополнительных параметров, чем в режиме **ACS**. Дополнительные 4 байта параметров сохраняются в энергонезависимой памяти и описываются аббревиатурой *AKSD*: ***A****uthorization command*, ***K****ey number*, ***S****tart block/page*, ***D****ata counter & mode*.

A – ***команда авторизации***; если авторизация не требуется, например для карт Mifare Ultralight, то A = 0; для авторизация по ключу A карт Mifare Classic, A = 60h, а для авторизации по ключу B, A = 61h.

K – ***номер ключа авторизации*** в памяти ридера (**это не ключ ACS!**); если A = 0, то значение этого параметра безразлично, т.к. авторизация не производится; принимает значения 0…84, хотя значение 0 бессмысленно, т.к. ключ авторизации AK0 находится в RAM.

S – ***стартовый блок/страница*** с которого начнётся авторизация/чтение данных; размер блока – 16 байт, размер страницы – 4 байта.

D – декрементированное ***количество блоков/страниц и режимы чтения данных***, которые надо прочитать; если установлен блоковый режим, то максимальное количество считываемых блоков, которое можно указать - 4, если установлен страничный режим, то максимальное количество страниц для считывания - 16; если установлен только режим авторизации без чтения данных, то значение счётчика считываемых данных безразлично.

Подробное описание настроек *AKSD* можно найти в описании команд **CMG_SRD**, **CMS_SRD**.

В режиме **SRD** предполагается следующий алгоритм работы:

1) из поднесённой карты считываются не только UID, SAK, ATQA, но записанные данные; если используются карты с авторизацией, то перед чтением данных проходит обязательная авторизация по указанному в *AKSD* ключу;
2) из считанных данных формируется запрос внешнему контроллеру, который должен решить, что делать с картой; если посылка запроса и ожидание подтверждения неактивны, а ридер установлен в режим чтения данных, то ридер будет всегда блокировать проход по карте – такие установки не имеют смысла!
3) если ридеру разрешена посылка запроса, то запрос передаётся на анализ внешнему контроллеру по активному интерфейсу – UART, RS-485 или USB, а ридер переходит в режим ожидания ответа с отсчётом таймаута;
4) на основании принятых данных, внешний контроллер посылает один из 3х вариантов ответа: 0 - запретить проход, 1 - разрешить проход, 2 - ридер должен пропустить карту, руководствуясь реакцией по умолчанию;
5) если в течение таймаута ожидания ридер принимает ответ от внешнего контроллера, то решение о пропуске карты будет принято на основании ответа; если до истечения таймаута ответ не будет принят, то реакцией ридера будет реакция по умолчанию: ридер блокирует все карты, проход которых не был подтверждён внешним контроллером; однако, если установлен режим “только авторизация”, то карта будет пропущена - ведь карта успешно прошла авторизацию, но её проход также должен удовлетворять режиму пропуска RW-карт.

Открыть замок без прикладывания карты можно нажатием кнопки “Open” (удерживать в течение 1с).

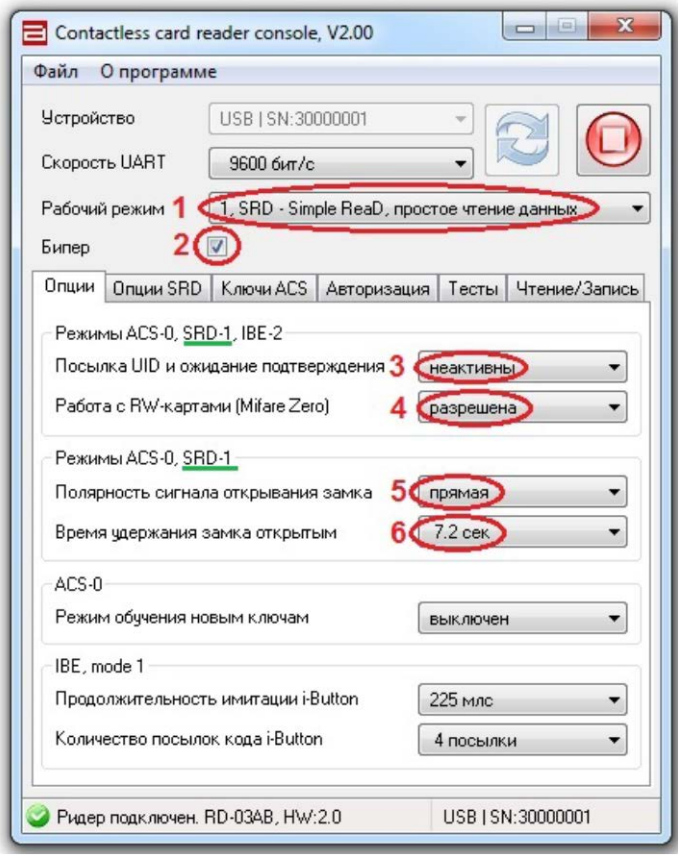
Режим SRD в примерах

Пример 1. Построение автономной пропускной системы для домофона без записи пропускаемых карт в ридер (аналогично *примеру 1* **ACS**): зная ключ авторизации, можно создавать новые пропуска, не перепрограммируя ридер. В режиме **SRD** нет ограничений на количество изготовленных карт-пропусков, т.к. карта пройдёт, лишь бы совпадал ключ авторизации.

В режиме **SRD** ридер может работать автономно только в режиме авторизации без чтения данных. В таком режиме ридер пытается авторизоваться в заданном блоке поднесённой карты. Если авторизация успешна, то ридер открывает замок двери на непродолжительное время (вывод **SW** устанавливается в активное состояние). Открытие замка может сопровождаться звуковым сигналом. Необходимость авторизации накладывает некоторые ограничения на выбор типа карт: использовать Mifare Ultralight и Mifare DesFire использовать не получится: Ultralight не поддерживает авторизацию, DesFire не использует авторизацию Crypto1.

С точки зрения пользователя, автономный режим **SRD** не отличается от режима **ACS**, однако, новые пропуска можно создавать прямо в отделе обслуживания, без выезда на место установки ридера.

Настроить ридер для работы в этом режиме можно с помощью программы ***ccr.exe***.

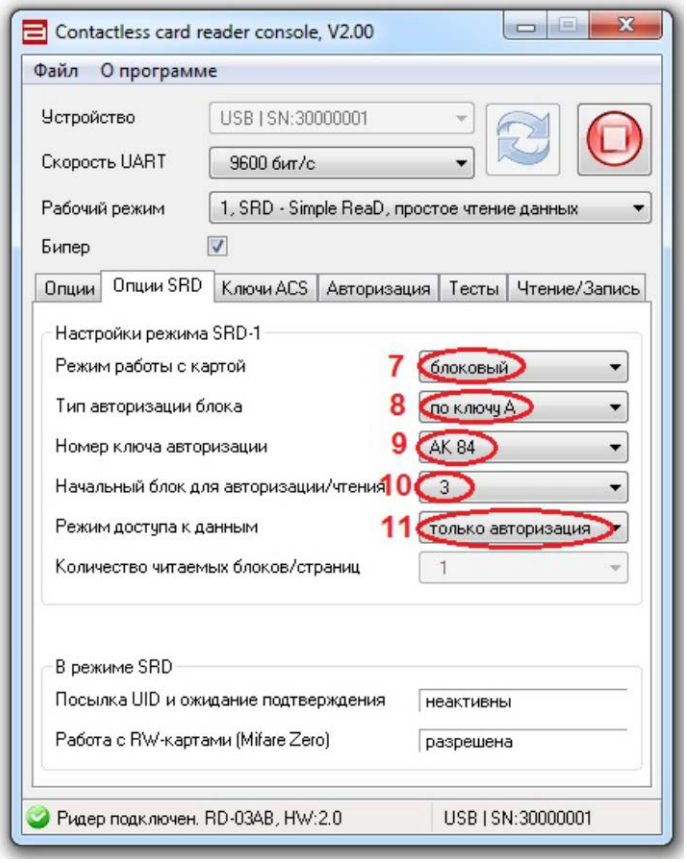


После подключения к выбранному ридеру с помощью ***ccr.exe***, следует установить:

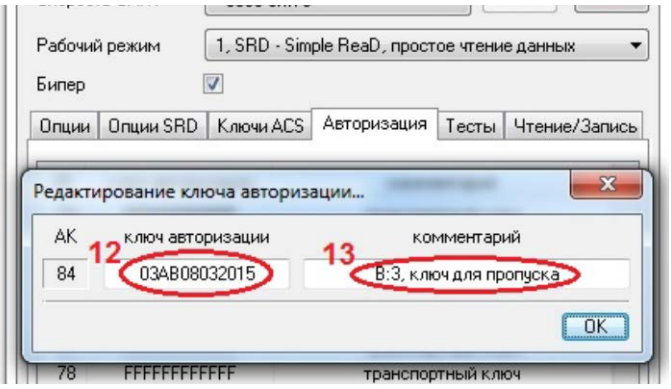
1) рабочий режим – **SRD**;
2) звуковой сигнал, если он нужен;
3) в автономном режиме ридер ничего не посылает и не передаёт;
4) разрешение прохода по RW-картам, если требуется;
5) полярность сигнала открывания замка так, чтобы в режиме ожидания дверь была закрыта;
6) время отпускания замка, за которое можно открыть дверь.

Далее переходим на вкладку “Опции **SRD**”.

7) режим работы с картой: для карт с авторизацией только блоковый (длина блока 16 байт);
8) тип авторизации блока: можно выбрать ключ A или B, но лучше выбирать A, т.к. работа с ключом B может быть запрещена;
9) номер ключа авторизации в памяти ридера: выбран последний ключ, но можно было выбрать любое другое место;
10) блок на карте, для которого производится авторизация;
11) режим доступа к данным: для автономного режима допустима только авторизация.



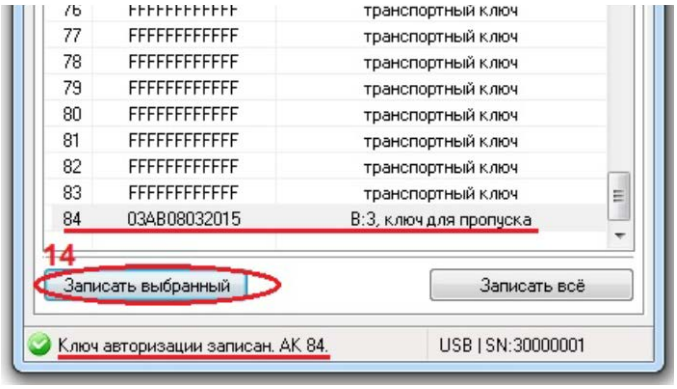
Далее, надо создать ключ авторизации AK 84 и сохранить его в памяти ридера.



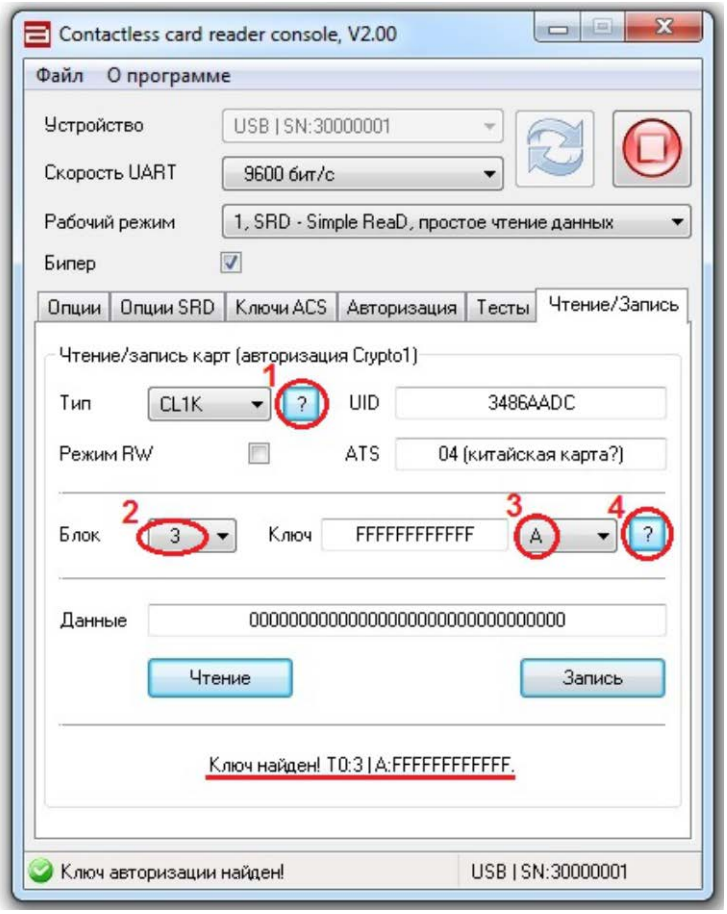
12) придумываем ключ авторизации AK 84 длиной 6 байт: *03AB08032015*; подойдёт любое сочетание шестнадцатеричных цифр;

13) устанавливаем комментарий, чтобы не забыть от какого блока ключ и его назначение;

14) после ввода ключа, не забываем сохранить его в памяти ридера.



На этом настройка ридера закончена. Теперь надо создать карту-пропуск. Поскольку ридер позволяет производить над картами любые операции, с его же помощью создадим карту-пропуск.



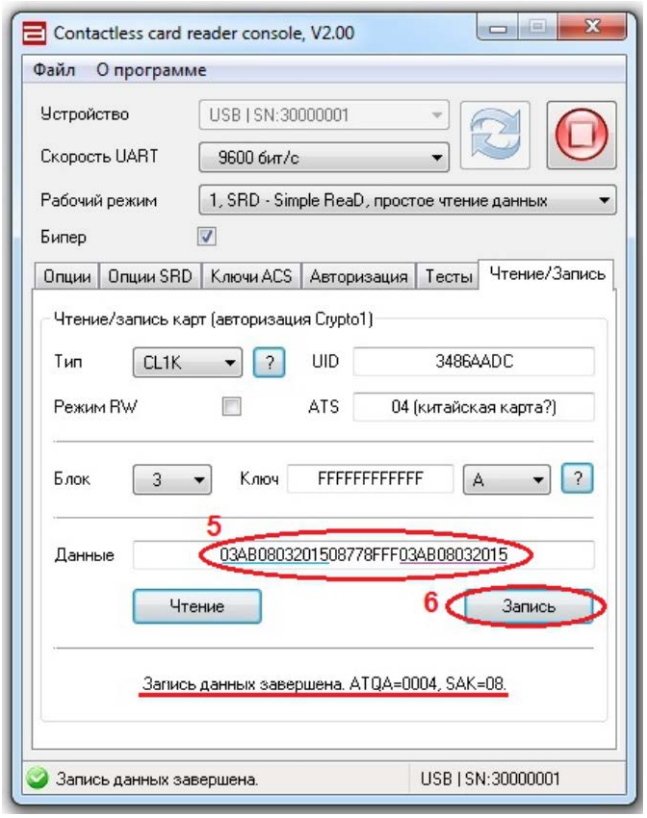
1) Карту следует положить на ридер и нажать кнопку определения типа карты; карты Mifare Ultralight (UL) и Mifare DesFire (DF1, DF2) не подходят;

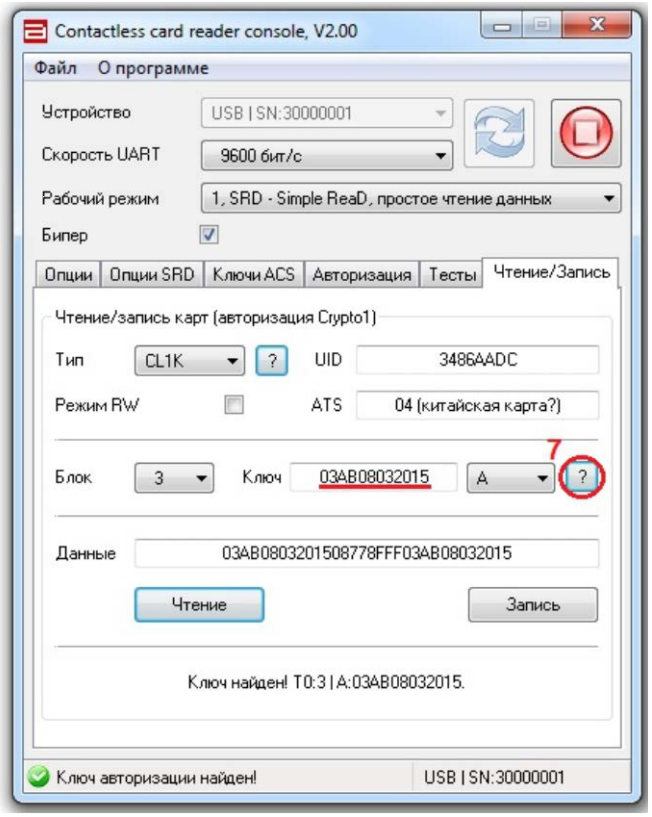
2) для авторизации выбираем блок 3: в блоке 3 хранятся ключи доступа сектора 0 (сектор состоит из 4х блоков – 0, 1, 2, 3), блок, где хранятся ключи, называется трейлером;

3) выбираем тип авторизации: новые карты обычно закрыты транспортным ключом A со значением FFFFFFFFFFFF; для карт использовавшихся ранее, ключ придётся подобрать;

4) нажимаем кнопку подбора ключа и ждём нахождения ключа доступа; в статусе указано, что текущий блок 3 является трейлером сектора 0 (T0:3);

5) формируем данные для трейлера, они состоят из двух ключей – A и B, и 4х байт режима (расположены между ключами): устанавливаем ключ A равным *03AB08032015* (подчёркнут голубой чертой), далее идут байты режима *08778FFF*, далее устанавливаем значение ключа B, такое же, как и ключа A *03AB08032015*, хотя можно было указать другое значение; байты атрибутов указывают, что ключ A является ключом по чтению, ключ B является ключом записи-чтения, ключи не видны при чтении;
6) записываем ключи в карту: всё, пропуск создан!

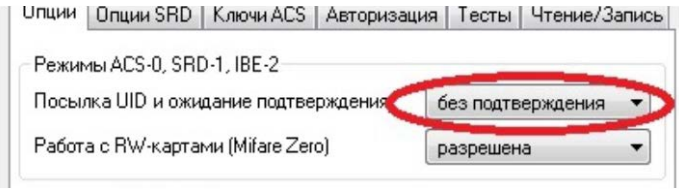




7) Для того, чтобы убедиться, что ключ создан и записан правильно, нажимаем кнопку поиска ключа и убеждаемся, что это значение ключа AK 84.

Теперь можно отключить ридер: всё настроено и готово к работе. Все последующие пропуска создаются аналогично. При записи трейлера надо следить за правильной установкой ключей и байтов режима: неправильная установка может сделать нерабочим весь сектор на карте.

Пример 2. Изменив настройку “*Посылка* ***UID*** *и ожидание подтверждения*” на “*без подтверждения*” в примере 1, получаем простую пропускную систему с записью лога и времени прохода (систему, аналогичную *примеру 2* режима **ACS**).

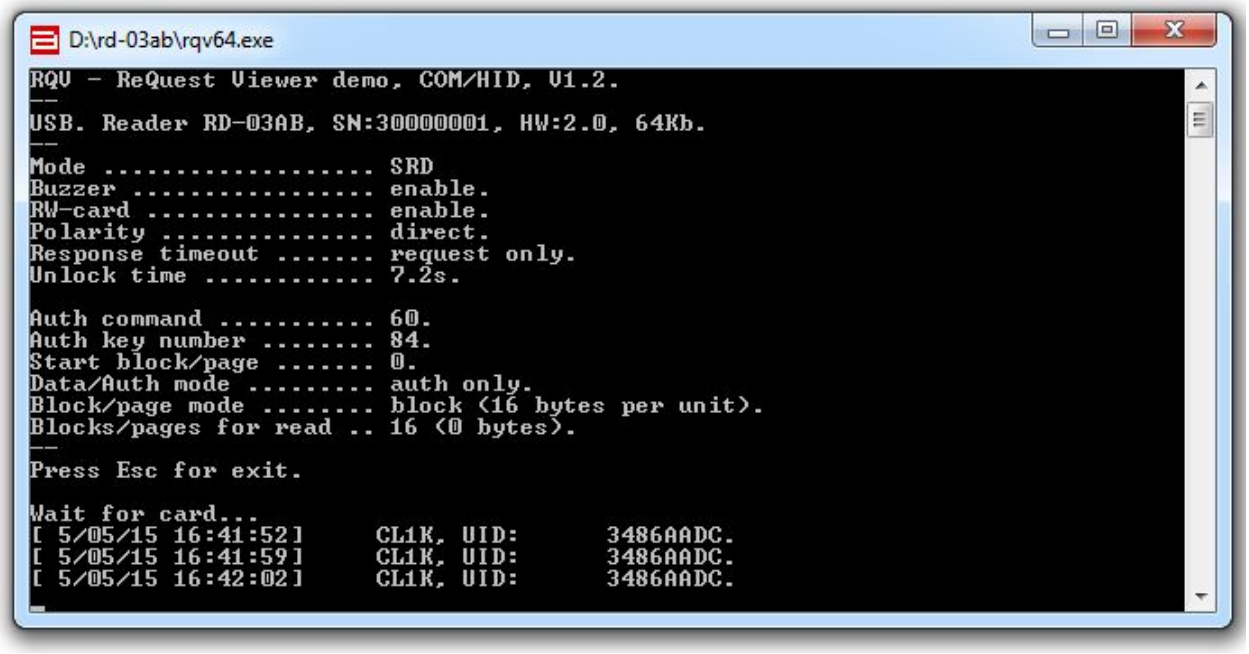


В этом режиме ридер будет посылать внешнему логгеру данные о считанных картах без ожидания подтверждения об их пропуске. К принятым данным логгеру потребуется только добавить время прохода.

Логгером может быть как простейшая программа на компьютере, когда ридер подключен через USB интерфейс, так и простейший логгер, подключенный по UART или RS-485 и сохраняющий данные на SD-карте.

Демонстрационная программа ***rqv*** показывает принцип построения программы-логгера. ***rqv*** считывает параметры ридера, и, если установлен режим передачи UID без ожидания подтверждения ***и передачи данных***, выводит на экран данные поднесённых карт.

Скриншот rqv

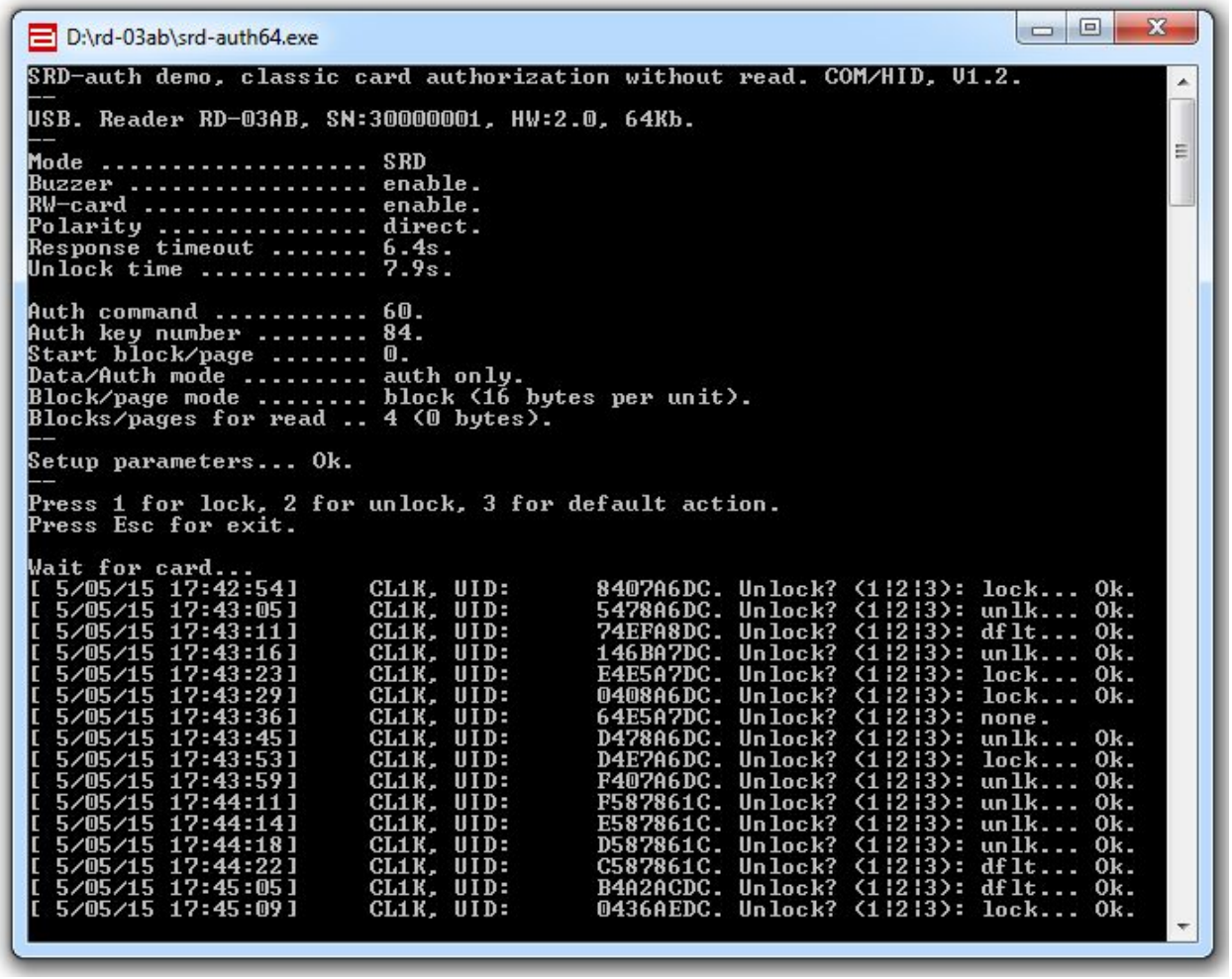


Программа строится на циклическом вызове функции *link_rqwait(handle, buf, SZ_SCCD)*. Логгер ориентирован на приём UID без данных, длина пакета в этом случае всегда равна *SZ_SCCD* = 14 байтам. Если запрос успешно принят, то функция возвращает код ***LERR_SUCCESS***. Подробное описание данных в буфере приводится в примере 2 для режима **ACS**.

Пример 3. Построение системы в режиме **SRD** с авторизацией без чтения данных, но с подтверждением запроса, позволяет организовать пропускную систему с внешним решателем, аналогичную системе, предложенной в *примере 3* для режима **ACS**, но в отличии от режима **ACS**, в режиме **SRD** новые карты не надо прописывать в ридер: достаточно установить правильный ключ авторизации. В случае потери связи с внешним контроллером, ридер “пропустит” карту только в случае успешной авторизации, руководствуясь разрешением работы с RW-картами.

Демонстрационная программа ***srd-auth*** показывает принцип построения такой системы.

Скриншот srd-auth



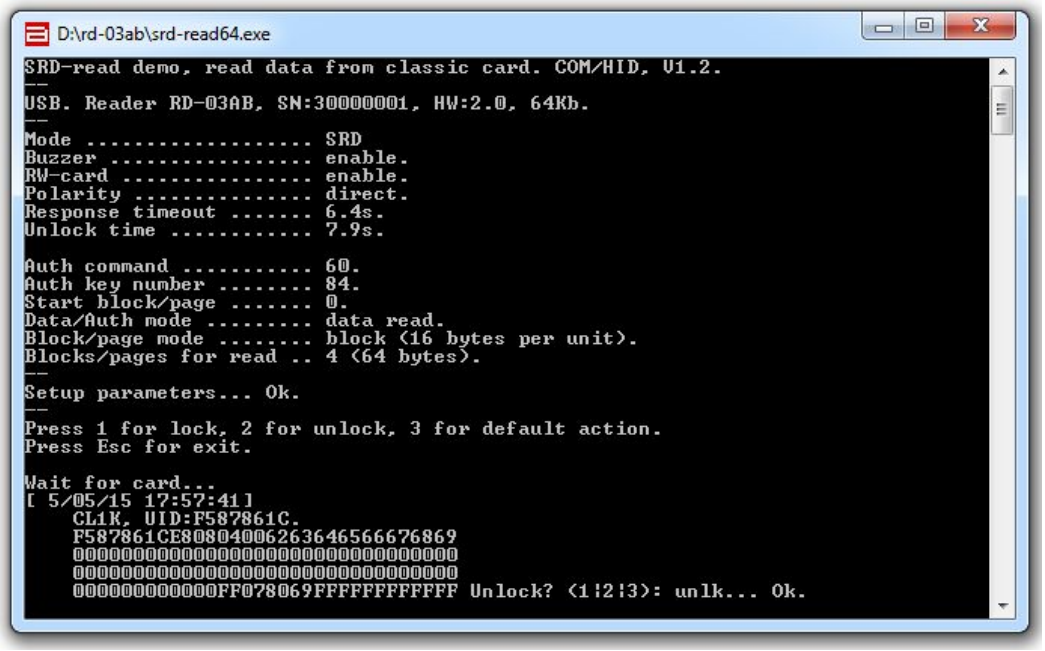
Для авторизации используется ключ AK84, который должен быть заранее внесён в память ридера. Если авторизация карты прошла успешно, ридер посылает запрос внешнему решателю и ожидает от решателя подтверждения прохода. В случае потери связи с решателем, карта ***будет*** пропущена.

Пример 4. Построение высокозащищённой системы авторизации основывается на чтении данных из карты и обработки их внешним решателем. При таком построении системы, решатель может учитывать UID и тип карты, использовать хэш для определения валидности карты и т.п.. Такая система может применяться на транспорте, в банковской сфере, в системах авторизованного доступа с повышенной криптостойкостью и т.д..

В качестве пропуска в режиме **SRD** могут применяться карты как с авторизацией считываемых данных (Classic, Plus, Mini и т.п.), так и без авторизации считываемых данных (Ultralight, Ultralight C, SM1K и т.п.). Использование карт с авторизацией считываемых данных предпочтительнее, т.к. в этом случае степень защиты выше в связи с тем, что данные не могут быть считаны без знания ключа авторизации. Для получения исключительно высокой степени защиты данных рекомендуется использование карт Mifare Plus с 7ми-значным UID, карт Mifare Classic нового поколения (MF1S50) или смарт-карт (например, SAM AV2).

Демонстрационная программа ***srd-read*** показывает принцип построения системы с авторизацией считываемых данных. Исходники находятся в каталоге ***srd***.

Скриншот srd-read



Для авторизации используется ключ AK84, который должен быть заранее внесён в память ридера. Если считывание данных прошло успешно, ридер посылает запрос с типом карты и считанными данными внешнему решателю и ожидает от решателя подтверждения прохода. В случае потери связи с решателем, карта ***не будет*** пропущена.

Демонстрационная программа ***srd-ultra*** показывает принцип построения системы без авторизации считываемых данных, например, на картах Ultralight.

Скриншот srd-ultra

Если считывание данных прошло успешно, ридер посылает запрос с типом карты и считанными данными внешнему решателю и ожидает от решателя подтверждения прохода. В случае потери связи с решателем, карта ***не будет*** пропущена.

Пример 5. Для связи с решателем, для объединения ридеров в общую сеть можно использовать Wi-Fi роутеры P3 (Ralink) или WR-703N (TP-Link) с изменённой прошивкой. Роутеры могут соединяться между собой проводной сетью или через Wi-Fi с шифрованием.

Ридер RD-03AB, подключенный к Wi-Fi / LAN модулю

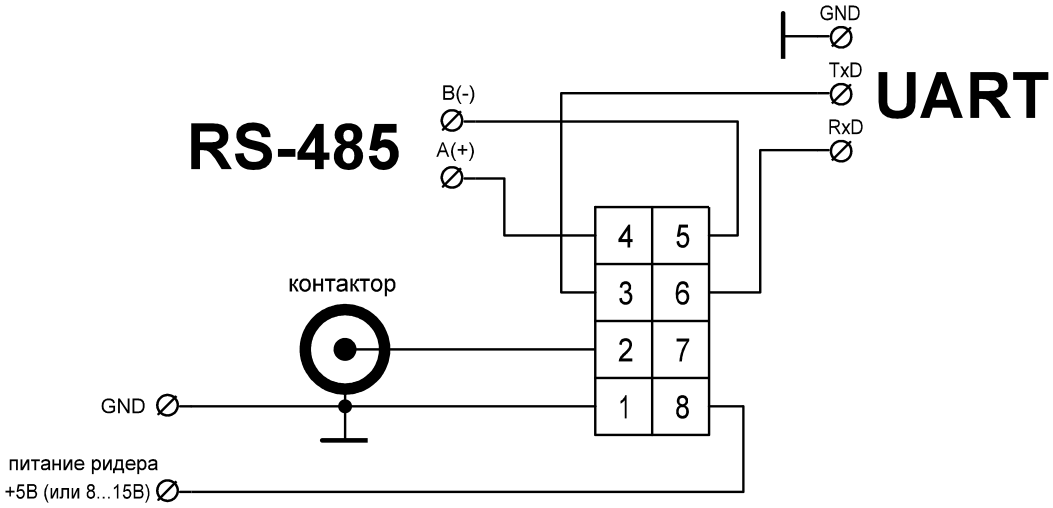
К одному роутеру через USB-хаб может быть подключено несколько ридеров. При приёме запроса от ридера, программа в роутере добавляет к пришедшему запросу заголовок в виде номера ридера от которого пришёл запрос, и отправляет в сеть широковещательный UDP пакет. Сервер, ответственный за решение о пропуске карты, принимает пакет, записывает его в лог и даёт подтверждение прохода таким же широковещательным пакетом, указав в заголовке номер ридера.

Построение пропускной системы на базе Wi-Fi модулей позволяет объединить множество ридеров в одну защищённую беспроводную сеть, которая отличается низкой стоимостью и высокой защищённостью от взлома. Подробности программирования коммерческих пропускных систем в данном документе не рассматриваются.

Режим IBE

Режим **IBE** предназначен для имитации брелоков-таблеток **iButton** DS1990A с помощью бесконтактных карт. Подобное использование ридера позволяет использовать преимущества бесконтактных систем без переделки существующей системы путём простого подключения ридера к контактору. Ридер имитирует серийный номер брелока-таблетки DS1990A, используя UID поднесённой карты.

Подключение ридера в режиме IBE



Для управления имитацией DS1990A используются два параметра: *ib_tmr* и *ib_rpt*. Считать параметры можно командой **CMG_IB**, установить новое значение параметров можно с помощью команды **CMS_IB**.

Параметр *ib_tmr* задаёт максимальное время имитации iButton. Этот параметр требуется для предотвращения зависания ридера в случае неполадок с интерфейсом или контроллером, обслуживающим интерфейс iButton. В связи с тем, что DS1990A является подчинённым устройством (slave), для того, чтобы считать код из DS1990A внешний контроллер посылает в DS1990A тактовые импульсы, по которым DS1990A выдаёт ответ. Однако, если по какой-либо причине контроллер перестанет выдавать тактовые импульсы, DS1990A будет бесконечно долго оставаться в “недочитанном” состоянии, что допустимо для отдельно взятого брелока и недопустимо для ридера. Время имитации расчитывается по формуле:

ib_time = (ib_tmr + 1) * 25млс, минимальное время 25млс, максимальное – 400млс.

Счётчик *ib_rpt* устанавливает необходимое количество повторов серийного номера, т.к. однократной передачи сымитированного кода для большинства устройств недостаточно. Если *ib_rpt* = 0, то при поднесении карты будет передан только один код (0 повторов), если *ib_rpt* = 1, то при поднесении карты будет передано 2 одинаковых кода (1 повтор), и т.д..

Ридер имитирует посылку серийного номера по двум командам интерфейса **iButton**: по команде 0x0F и по команде 0x33. Выход из режима имитации происходит или по таймауту *ib_tmr*, или по исчерпанию счётчика кодов *ib_rpt*, в зависимости от того, какое событие наступит раньше.

Соответствие байтов серийного номера DS1990A байтам UID

FAM	SN0	SN1	SN2	SN3	SN4	SN5	CRC8
0x01	UID0	UID1	UID2	UID3	UID4	UID5	CRC8

Если длина UID менее 6ти байт, недостающие байты UID (UID4 и UID5) заполняются нулями.

Как и в режиме **ACS**, в режиме **IBE** решение посылать код карты по интерфейсу **iButton** или нет, может принимать как внешний контроллер, так и сам ридер. В режиме **IBE** предполагается следующий алгоритм работы:

6) считывается UID, ATQA, SAK поднесённой карты, проверяется, является ли карта перезаписываемой (RW-свойство карты); ATQA и SAK могут быть использованы для определения типа карты;
7) из считанных данных формируется запрос внешнему контроллеру (*CD – **C**ard **D**escriptor*), который должен решить, что делать с картой; *если посылка запроса и ожидание подтверждения неактивны, то ридер имитирует подключение DS1990A к контактору, руководствуясь разрешением работы с RW-картами*;
8) если ридеру разрешена посылка запроса, то запрос передаётся на анализ внешнему контроллеру по активному интерфейсу – UART, RS-485 или USB, а ридер переходит в режим ожидания ответа с отсчётом таймаута; *если посылка запроса и ожидание подтверждения неактивны, то ридер имитирует подключение DS1990A к контактору, руководствуясь разрешением работы с RW-картами*;
9) на основании принятых данных, внешний контроллер посылает один из 3х вариантов ответа: 0 - запретить проход, 1 - разрешить проход, 2 - ридер должен пропустить карту, руководствуясь разрешением работы с RW-картами;
10) если в течение таймаута ожидания ридер принимает ответ от внешнего контроллера, то решение о пропуске карты будет принято на основании ответа; если до истечения таймаута ответ не будет принят, ридер сам примет решение о пропуске карты, руководствуясь наличием UID карты во внутренней базе ключей и разрешением работы с RW-картами.

Несмотря на возможность подключения внешнего решателя, режим IBE рекомендуется рассматривать как автономный режим с записью или без записи лога.

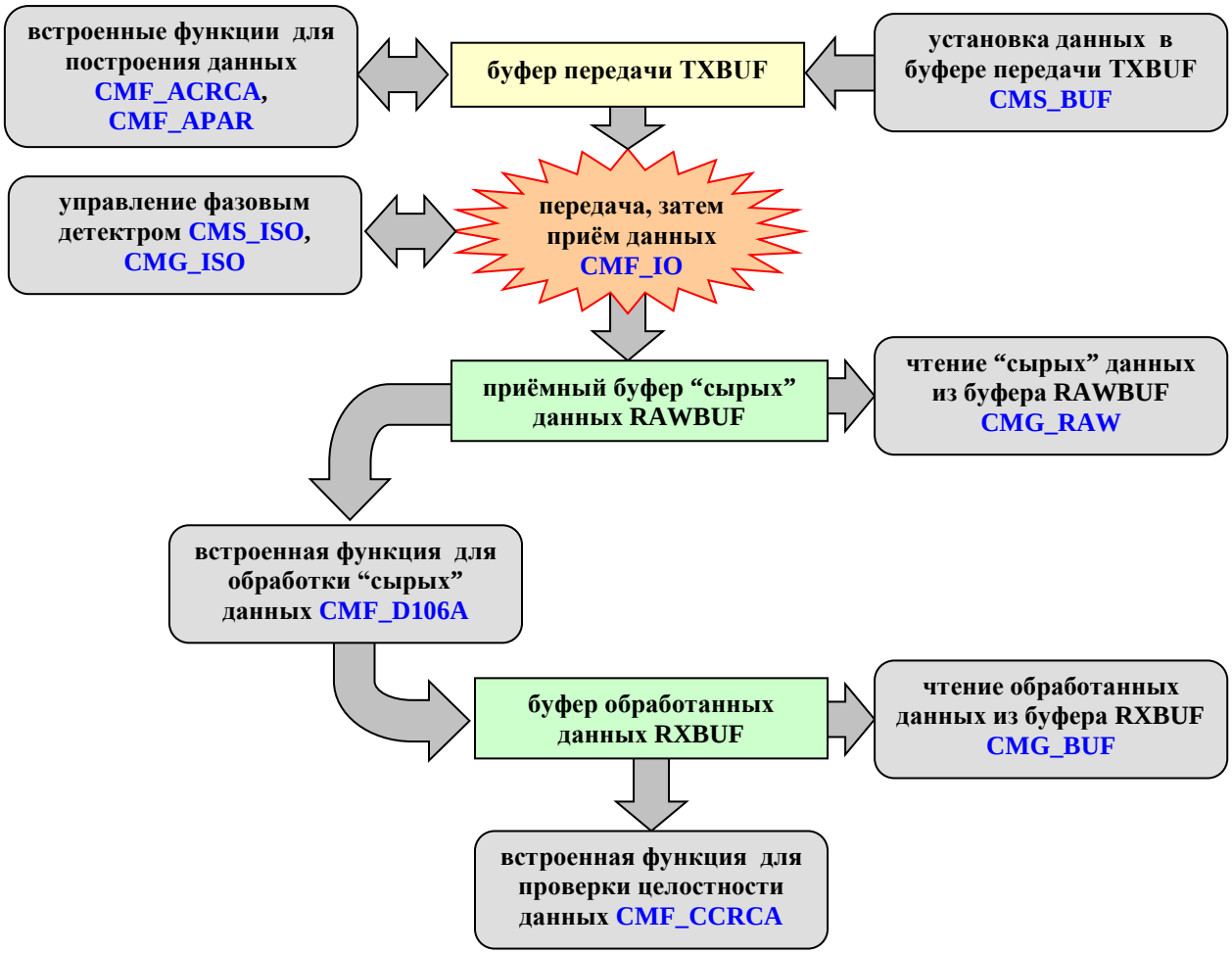
Демонстрационная программа ***ibe-demo*** показывает принцип построения такой системы.

Режим **DIR**

Режим **DIR** позволяет работать с бесконтактными картами не только как обычный ридер настольного типа, но и взаимодействовать с картами на уровне “сырого” кода. В этом режиме одинаково просто работать с картами как на уровне примитивов, типа “считать блок – записать блок”, так и формируя “вручную” кадр обмена произвольного формата.

Ридер в этом режиме легко использовать для изучения уязвимостей бесконтактных карт, отработки алгоритмов обмена, вскрытия ключей защиты и т.п.. В частности, ридер в режиме DIR использовался при реверсивном анализе проприетарного алгоритма Crypto1 в картах Mifare. Ридер отлично работает с программами ***mfoc*** и ***mfcuk*** – лучшими программами, использующими уязвимости Mifare.

Структурная схема программной модели ридера изображена на схеме.



Базовой функцией для построения обмена с бесконтактными картами является функция **CMF_IO**. Эта функция передаёт данные из командного буфера TXBUF и заполняет буфер “сырых” данных RAWBUF ответом карты. Этой функции достаточно для организации обмена с картой, однако, для удобства использования ридера добавлены вспомогательные функции, которые выполняют битовые операции и упрощают работу с ридером.

В TXBUF[0] хранится длина передаваемых данных в битах, начиная с TXBUF[1] хранятся передаваемые данные. Порядок передачи битов в байте - младший-старший. Например, если передаётся команда 0x26 длиной 7 бит, то TXBUF[0]=7, TXBUF[1]=0x26, при этом бит 7 в TXBUF[1] не используется, т.е. при TXBUF[1]=0xA6, также будет передана правильная команда. Если требуется передать 9 битов (TXBUF[0]=9), то в TXBUF[1] будут храниться биты 0…7, а в TXBUF[2] в младшем бите будет храниться 8ой передаваемый бит. Длина TXBUF 33 байта: 1байт длины данных + 32 байта данных.

Ответ карты записывается в RAWBUF, порядок записи байтов и битов - младший-старший. Один бит, записанный в RAWBUF соответствует выборке из демодулятора на скорости 847кбит/с. Длина буфера RAWBUF 340 байт.

В случае обмена по стандарту ISO-14443, в RAWBUF будет ответ карты в манчестерском коде. В зависимости от стандарта и фазы обмена, выбирается алгоритм декодирования. Карты Mifare отвечают на скорости 106кбит/с, т.е. для декодирования кодирования 1го бита ответа требуется обработать 2 байта RAWBUF. В ридере есть вспомогательная функция **CMF_D106A**, которая декодирует содержимое RAWBUF по стандарту ISO-14443A 106кбит/с. Функция **CMF_D106A** может декодировать ответ как с проверкой паритета, так и без проверки. Декодированные данные помещаются в RXBUF.

RXBUF организован также, как и TXBUF: в RXBUF[0] находится длина принятых данных в битах, данные хранятся начиная с RXBUF[1]. В RXBUF помещаются данные, выровненные на границу байта. Длина RXBUF 86 байт. С данными в RXBUF работает функция проверки CRC **CMF_CRCA** по стандарту ISO-14443A.

При работе ридера в режиме **DIR**, управляющая программа должна сама заботиться о правильном опорном уровне фазового детектора, иначе считывание данных с карты будет невозможным. Установка уровня фазового детектора производится командой **CMS_ISO** в поле PHDL, при этом PHDF должен быть равен 1.

Наиболее прост инкрементальный (декрементальный) алгоритм установки уровня фазового детектора, когда устанавливается минимально (максимально) возможный PHDL=0x00 (0x1F), затем посылается команда подключения карты (PICC_REQIDL = 0x26 или PICC_REQALL = 0x52). Если принят корректный ответ карты, то уровень PHDL фиксируется и дальнейший обмен производится с неизменным уровнем фазового детектора. В случае неудачи, PHDL увеличивается (уменьшается) на единицу, а затем производится новая попытка подключения к карте. Ридер использует инкрментальный алгоритм управления.

Описание команд

При описании команд используется следующая схема:

- если для описания используется обозначение **xx** – значение данных безразлично;
- если для описания используется обозначение **nn** – данные обрабатываются командой;
- жёлтый цвет соответствует передаваемым в ридер данным;
- зелёный цвет соответствует принимаемым из ридера данным;
- аббревиатура TC означает передачу ридеру командного кадра (**T**ransmit **C**ommand);
- аббревиатура TD означает передачу кадра данных (**T**ransmit **D**ata);
- аббревиатура RD означает приём кадра данных (**R**eceive **D**ata).

CMG_VER

код 0xFF - получение версии, объёма памяти, серийного номера

Команда **CMG_VER** предназначена для получения версии ридера, длины памяти ключей и серийного номера ридера. Параметр безразличен.

Протокол: TC-RD.

Команда:

FF	xx

Ответ (длина 7 байт):

смещение	назначение	длина	структура данных
+00h	версия ридера	1	номер аппаратной реализации ридера: старшая тетрада – major version, младшая тетрада – minor version
+01h	количество ключей	1	бит 7 – тип памяти ключей: 0 – внешняя энергонезависимая память (NVM), 1 – внутренняя память контроллера (ROM) биты 6…0 - номер последней страницы памяти ключей
+02h	серийный номер	4	серийный номер ридера в BCD формате (старший-младший): 1…0x99999999; старшая тетрада содержит модель ридера: 0x1nnnnnnn - RD-01A, 0x2nnnnnnn - RD-02AB, 0x3nnnnnnn - RD-03AB

Пример вызова команды:

err = link_packet(CMG_VER, 0, NULL, 0, rxbuf, 6); // получение версии ридера

CMF_RST

код 0xFE - сброс настроек ридера в настройки по умолчанию

Команда **CMF_RST** предназначена для возврата настроек ридера к заводским установкам. Параметр должен быть равен 0xA5. Ключи доступа и ключи авторизации не сбрасываются.

Протокол: TC.

Команда:

FE	A5

Пример вызова команды:

err = link_packet(CMF_RST, 0xA5, NULL, 0, NULL, 0); // сброс настроек ридера

CMG_MOD

код 0xFD - чтение режима бипера и режима ридера

Команда **CMG_MOD** предназначена для получения флага состояния бипера и режима ридера. Бит 7 параметра указывает, какое значение запрошено:

0 – чтение текущего состояния бипера и текущего режима ридера (RAM-режим),

1 – чтение сохранённого состояния бипера и сохранённого режима (ROM-режим).

Сохранённый(ROM) и текущий(RAM) режимы могут не совпадать. ROM режим будет установлен после сброса ридера, например, после выключения и последующей подачи питания.

Протокол: TC-RD.

Команда:

FD	xx

Ответ (длина 1 байт):

смещение	назначение	длина	структура данных
+00h	флаг бипера и режим ридера	1	бит 3 – флаг разрешения работы бипера: 0 – подача звуковых сигналов запрещена, 1 – подача звуковых сигналов разрешена биты 1…0 - номер текущего режима: 0 – режим ACS, 1 – режим SRD, 2 – режим IBE, 3 – режим DIR биты 7…4, 2 зарезервированы

Пример вызова команды:

```
err = link_packet(CMG_MOD, 0x00, NULL, 0, rxbuf, 1);      // чтение текущего режима
err = link_packet(CMG_MOD, 0x80, NULL, 0, rxbuf, 1);      // чтение ROM - режима
```

CMS_MOD

код 0xFC - установка режима бипера и режима ридера

Команда **CMS_MOD** предназначена для установки нового флага состояния бипера и режима ридера. Бит 7 параметра указывает, куда записывается значение:

0 – установка текущего состояния бипера и текущего режима (RAM-режим),
1 – установка состояния бипера и режима ридера в ROM.

Сохранённый(ROM) и текущий(RAM) режимы могут не совпадать. ROM режим будет установлен после сброса ридера, например, после выключения и последующей подачи питания.

ВНИМАНИЕ! Запись режима в ROM одновременно устанавливает такой же режим в RAM, т.е. для установки режима в ROM, отличающегося от режима в RAM, следует выполнить 2 команды: 1) установить требуемый ROM-режим (бит 7 равен 1), 2) установить требуемый RAM-режим (бит 7 равен 0).

Протокол: TC.

Команда:

FC	nn

смещение	назначение	длина	структура данных
+00h	флаг бипера и режим ридера	1	бит 7: 0 – установка текущего режима (RAM), 1 – сохранение режимов в ROM бит 3 – флаг разрешения работы бипера: 0 – подача звуковых сигналов запрещена, 1 – подача звуковых сигналов разрешена биты 1…0 - номер текущего режима: 0 – режим ACS, 1 – режим SRD, 2 – режим IBE, 3 – режим DIR биты 6…4, 2 зарезервированы

Пример вызова команды:

err = link_packet(CMS_MOD, 8 + 3, NULL, 0, NULL, 0) // звук включен, текущий режим DIR
err = link_packet(CMS_MOD, 0x80 + 8 + 0, NULL, 0, NULL, 0); // ROM – режимы: звук + ACS

CMG_UBR

код 0xFB - чтение скорости обмена по UART/RS485

Команда **CMG_UBR** предназначена для получения текущей скорости обмена по последовательному интерфейсу UART/RS485. Параметр безразличен. Соответствие индекса скорости – табличное.

Протокол: TC-RD.

Команда:

FB	xx

Ответ (длина 1 байт):

смещение	назначение	длина	структура данных
+00h	BR idx	1	индекс скорости обмена 0…9 (реальная скорость обмена вычисляется по таблице)

BR idx	baud rate
0	1200 бит/сек
1	2400 бит/сек
2	4800 бит/сек
3	9600 бит/сек
4	14400 бит/сек
5	19200 бит/сек
6	38400 бит/сек
7	56000 бит/сек
8	57600 бит/сек
9	115200 бит/сек

Пример вызова команды:

```
err = link_packet(CMG_UBR, 0, NULL, 0, rxbuf, 1)    // чтение UART Baud Rate
```

CMS_UBR

код 0xFA - установка скорости обмена по UART/RS485

Команда **CMS_UBR** предназначена для установки текущей скорости обмена ридера по последовательному интерфейсу UART/RS485. Параметр **nn** – индекс новой скорости обмена 0…9. Параметр сохраняется в энергонезависимой памяти ридера.

Протокол: TC.

Команда:

FA	nn

nn	baud rate
0	1200 бит/сек
1	2400 бит/сек
2	4800 бит/сек
3	9600 бит/сек
4	14400 бит/сек
5	19200 бит/сек
6	38400 бит/сек
7	56000 бит/сек
8	57600 бит/сек
9	115200 бит/сек

Пример вызова команды:

err = link_packet(CMS_UBR, 3, NULL, 0, NULL, 0) // установка Baud Rate = 9600 бит/сек

CMG_OPT

код 0xF9 – чтение управляющих флагов (опций)

Команда **CMG_OPT** предназначена для чтения управляющих флагов ридера, влияющих на его работу. Параметр безразличен. Команда возвращает текущее состояние флагов:

rwc_en - разрешение работы с RW-картами в режимах **ACS**, **SRD**, **IBE**;
sto_en – флаг разрешения обучения в режиме ACS;
polar – флаг полярности управления замком (прямая/инверсная).

Упаковка флагов в байт

бит 7	бит 6	бит 5	бит 4	бит 3	бит 2	бит 1	бит 0
не исп.	не исп.	не исп.	не исп.	не исп.	**rwc_en**	**sto_en**	**polar**

Протокол: TC-RD.

Команда:

F9	xx

Ответ (длина 1 байт):

смещение	назначение	длина	структура данных
+00h	байт флагов	1	бит 2 – **rwc_en**, 0 - запрещено, 1 - разрешено бит 1 – **sto_en**, 0 - запрещено, 1 - разрешено бит 0 – **polar**, 0 – прямая, 1 - инверсная биты 7…3 зарезервированы

Пример вызова команды:

err = link_packet(CMG_OPT, 0, NULL, 0, rxbuf, 1) // чтение флагов

CMS_OPT

код 0xF8 – установка управляющих флагов (опций)

Команда **CMS_OPT** предназначена для установки флагов ридера, влияющих на его работу. Флаги сохраняются в энергонезависимой памяти. Параметр – новое значение флагов **rwc_en**, **sto_en**, **polar**:

rwc_en - разрешение работы с RW-картами в режимах **ACS**, **SRD**, **IBE**;

sto_en – флаг разрешения обучения в режиме ACS;

polar – флаг полярности управления замком (прямая/инверсная).

Упаковка флагов в байт nn

бит 7	бит 6	бит 5	бит 4	бит 3	бит 2	бит 1	бит 0
не исп.	не исп.	не исп.	не исп.	не исп.	**rwc_en**	**sto_en**	**polar**

Протокол: TC.

Команда:

F8	nn

Пример вызова команды:

err = link_packet(CMS_OPT, 4 + 2 + 0, NULL, 0, NULL, 0) // установка rwc_en и sto_en

CMG_TMR

код 0xF7 – чтение сервисных таймеров

Команда **CMG_TMR** предназначена для чтения значения двух сервисных таймеров: таймера удержания замка открытым и таймера ожидания подтверждения. Параметр безразличен.

Упаковка таймеров в байт

бит 7	бит 6	бит 5	бит 4	бит 3	бит 2	бит 1	бит 0
link_timer				*unlock_timer*			

Старшая тетрада возвращаемого байта – таймер ожидания подтверждения *link timer = 0…15*; младшая тетрада – таймер удержания замка открытым *unlock timer = 0…15*. Некоторые значения таймеров выполняют специальные функции и не могут использоваться для расчёта времени:

link_timer = 0 запрещает посылку запроса для внешнего управления;

link_timer = 1 разрешает посылку запроса без ожидания ответа;

unlock_timer = 0 переводит ридер в триггерный режим: каждое валидное поднесение карты меняет состояние замка с открытого на закрытое и наоборот. При всех других значениях, время можно рассчитать по следующим формулам:

link_time = (n*16+15)*25млс; минимальное значение 1.175с, максимальное – 6.375с.

unlock_time = (n*16+15)*50млс; минимальное значение 1.55с, максимальное – 12.75с.

Протокол: TC-RD.

Команда:

F7	xx

Ответ (длина 1 байт):

смещение	назначение	длина	структура данных
+00h	*link_timer* *unlock_timer*	1	биты 7…4 – *link_timer*, значения 0, 1 для специальных функций биты 3…0 – *unlock_timer*; значение 0 включает триггер-режим замка

Пример вызова команды:

err = link_packet(CMG_TMR, 0, NULL, 0, rxbuf, 1) // чтение таймеров

CMS_TMR

код 0xF6 – установка сервисных таймеров

Команда **CMS_TMR** предназначена для установки значений двух сервисных таймеров: таймера удержания замка открытым и таймера ожидания подтверждения. Установленные значения сохраняются в энергонезависимой памяти.

Упаковка таймеров в байт

бит 7	бит 6	бит 5	бит 4	бит 3	бит 2	бит 1	бит 0
link_timer				*unlock_timer*			

Старшая тетрада параметра – таймер ожидания подтверждения *link timer = 0…15*; младшая тетрада – таймер удержания замка открытым *unlock timer = 0…15*. Некоторые значения таймеров выполняют специальные функции и не могут использоваться для расчёта времени:

link_timer = 0 запрещает посылку запроса для внешнего управления;

link_timer = 1 разрешает посылку запроса без ожидания ответа;

unlock_timer = 0 переводит ридер в триггерный режим: каждое валидное поднесение карты меняет состояние замка с открытого на закрытое и наоборот. При всех других значениях, время можно рассчитать по следующим формулам:

link_time = (n*16+15)*25млс; минимальное значение 1.175с, максимальное – 6.375с.

unlock_time = (n*16+15)*50млс; минимальное значение 1.55с, максимальное – 12.75с.

Протокол: ТС.

Команда:

F6	nn

параметр	назначение	структура данных
nn	*link_timer*	биты 7…4 – *link_timer*, значения 0, 1 для специальных функций
	unlock_timer	биты 3…0 – *unlock_timer*; значение 0 включает триггер-режим замка

Пример вызова команды:

err = link_packet(CMS_TMR, 0x08, NULL, 0, NULL, 0) // link_timer=0, unlock_timer=8

CMF_SW

код 0xF5 – управление замком

Команда **CMF_SW** предназначена для принудительной установки вывода управления замком в активное или неактивное состояние. Параметр – упакованные флаги **fsw**, **ssw**:

fsw – ***F****orce* ***SW****itch*, включение режима принудительной установки вывода управления замком,

ssw – ***S****tate* ***SW****itch*, устанавливает неактивное/активное состояние вывода управления замком.

Упаковка флагов в байт

бит 7	бит 6	бит 5	бит 4	бит 3	бит 2	бит 1	бит 0
не исп.	не исп.	не исп.	не исп.	не исп.	не исп.	**fsw**	**ssw**

В упакованном байте **nn**, биты 7…2 незначащие, бит 1 – **fsw**, бит 0 – **ssw**. Если **fsw**=0, то состояние **ssw** безразлично, т.к. принудительное управление выводом замка выключено.

ВНИМАНИЕ! При подаче питания, сбросе ридера или переключении режима работы ридера **fsw** и **ssw** сбрасываются в 0!

Протокол: ТС.

Команда:

F5	nn

Пример вызова команды:

err = link_packet(CMF_SW, 3, NULL, 0, NULL, 0) // принудительное открытие замка

CMF_BUZ

код 0xF4 – подача звукового сигнала

Команда **CMF_BUZ** предназначена для подачи ридером звукового сигнала. Параметр **nn**=0…5 устанавливает тип подаваемого звукового сигнала:

0 – **check**, звук, используемый при успешном завершении операции;
1 – **fail** - звук, используемый при ошибочном завершении операции;
2 – **good** - короткий звук при успешном завершении операции;
3 – **short_fail** - короткий звук при ошибочном завершении операции;
4 – **trill**, трель, используется для обозначения важных событий;
5 – **wow**, нарастающий звук сирены, используется при ошибке важной операции.

ВНИМАНИЕ! Перед маркером завершения операции подаётся маркер ожидания '**B**', после которого подаётся звуковой сигнал.

Протокол: TC.

Команда:

F4	nn

Пример вызова команды:
err = link_packet(CMF_BUZ, 2, NULL, 0, NULL, 0) // подача короткого бипа "good"

CMF_KME

код 0xF3 – стирание ключей доступа и получение данных о последней занятой странице

Команда **CMF_KME** предназначена для стирания всех ключей доступа из памяти ридера (мастер-ключ не стирается): освободившееся место заполняется байтом 0xFF. Также команда умеет возвращать номер последней занятой ключами страницы, т.е. страница, следующая за последней занятой страницей, будет заполнена 0xFF. Сведения о последней занятой странице позволяют резко сократить объём передаваемых данных при редактировании ключей доступа, что особенно важно при низких скоростях обмена или большом объёме внешней памяти: считывать страницы с ключами доступа можно не все (значение, возвращаемое **CMG_VER**), а до значения, возвращаемого **CMF_KME**, включительно.

Команда **CMF_KME** – это по сути две разные команды, протокол которых определяется параметром: стирание будет выполнено при параметре 0xA5, возврат последней занятой страницы – при параметре 0x00.

Стирание

Протокол: TC.

Команда:

F3	A5

При установленной внешней памяти (NVM), стирание может занять значительное время (до 10сек), в течение которого ридер не выходит на связь. Проверить окончание цикла стирания можно по приходу эха, посылая любой байт, кроме командного маркера: для этого можно использовать функцию API *link_echobyte*.

Стирание внутренней памяти (ROM) занимает не более 1.5сек, поэтому команду можно подавать в обычном режиме. Узнать установленный тип памяти можно с помощью команды **CMG_VER**.

Пример вызова команды:

err = link_packet(CMF_KME, 0xA5, NULL, 0, NULL, 0) // стирание памяти ключей

Информация о последней занятой странице

Протокол: TC.

Команда:

F3	00

Ответ:

lstpg

Пример вызова команды:

err = link_packet(CMF_KME, 0, NULL, 0, lstpg, 1) // чтение последней занятой страницы

CMG_KM

код 0xF2 – получение страницы ключей

Команда **CMG_KM** предназначена для чтения страницы памяти ключей доступа - ACS-ключей. Параметр **nn** указывает номер считываемой страницы. Номер последней страницы памяти возвращает команда **CMG_VER**.

Формат ACS-ключа

байт 0	**байт 1**	**байт 2**	**байт 3**	**байт 4**	**байт 5**	**байт 6**	**байт 7**
длина UID	UID0	UID1	UID2	UID3	UID4	UID5	UID6

Ключи доступа используются только в режиме **ACS**, однако, читать и записывать их можно в любом режиме. Длина записи ACS-ключа – 8байт. Память для хранения ACS-ключей организована странично с размером страницы 512 байт. На одной странице помещается 512 / 8 = 64 ключа. Нумерация ключей начинается с нуля с самой младшей страницы с самого младшего адреса. Ключ с номером 0 – мастер-ключ. Ключ считается удалённым, когда все его байты равны 0FFh.

Страница 0

смещение	№ ключа	структура данных
+000h	0	мастер-ключ, позволяет прописывать новые ключи
+008h	1	ключ, если поле пустое, то все байты равны 0FFh
+010h	2	ключ, если поле пустое, то все байты равны 0FFh
…	…	…
+1F8h	63	ключ, если поле пустое, то все байты равны 0FFh

……

Страница n=max

смещение	№ ключа	структура данных
+000h	0+n*64	ключ, если поле пустое, то все байты равны 0FFh
+010h	2+n*64	ключ, если поле пустое, то все байты равны 0FFh
…	…	…
+1F8h	63+n*64	***ключ для ROM, сигнатура для внешней NVM***

Протокол: TC-RD.

Команда:

F2	nn

Ответ (длина 512 байт):

смещение	назначение	длина	структура данных
+00h	страница ACS-ключей	512	страница памяти ACS-ключей

Пример вызова команды:

err = link_packet(CMG_KM, 3, NULL, 0, keypg, 512) *// чтение 3ей страницы ключей*

CMS_KM

код 0xF1 – запись страницы ключей

Команда **CMS_KM** предназначена для записи страницы памяти ключей доступа - ACS-ключей. Параметр **nn** указывает номер записываемой страницы. Номер последней страницы памяти возвращает команда **CMG_VER**.

Описание структуры страницы ключей приводится в описании команды **CMG_KM**. Если в ридере используется внешняя NVM для хранения ACS-ключей, то при записи последней страницы надо сохранять считанную сигнатуру, иначе при подаче питания внешняя NVM будет отформатирована, а её содержимое будет стёрто (кроме мастер-ключа). Т.е. перед записью, последняя страница NVM обязательно должна быть считана, чтобы сохранить правильную сигнатуру, установленную при форматировании.

Протокол: TC-TD.

Команда:

F1	nn

Передаваемый кадр данных (длина 512 байт):

смещение	назначение	длина	структура данных
+00h	страница ACS-ключей	512	страница памяти ACS-ключей

Пример вызова команды:

err = link_packet(CMS_KM, 0, keypg, 512, NULL, 0) // запись 0ой страницы ключей

CMG_IB

код 0xF0 – чтение параметров имитации iButton

Команда **CMG_IB** предназначена для чтения параметров имитации iButton в режиме **IBE**. Параметр безразличен. Команда возвращает состояние *ib_tmr* (старшая тетрада) и *ib_rpt* (младшая тетрада):

ib_tmr – 0…15, максимальное время имитации iButton в тиках по 25млс,

ib_rpt – 0…15, число посылок ID, которые будут переданы за время *ib_tmr*.

Упаковка параметров в байт

бит 7	бит 6	бит 5	бит 4	бит 3	бит 2	бит 1	бит 0
ib_tmr				*ib_rpt*			

При имитации iButton, ридер выступает в качестве ведомого устройства, к которому ведущее устройство, спрашивающее код, может и не обратиться вследствие неисправности или занятости. Таймер *ib_tmr* ограничивает время пребывания в режиме имитации, чтобы избежать «подвисания» ридера в случае неактивности ведущего устройства. Время имитации расчитывается по формуле:

ib_time = (ib_tmr + 1) * 25млс, минимальное время 25млс, максимальное – 400млс.

Счётчик *ib_rpt* устанавливает необходимое количество повторов кода, т.к. однократной передачи сымитированного кода для большинства устройств недостаточно. Если *ib_rpt* = 0, то при поднесении карты будет передан только один код (0 повторов), если *ib_rpt* = 1, то при поднесении карты будет передано 2 одинаковых кода (1 повтор), и т.д..

Протокол: TC-RD.

Команда:

F0	xx

Ответ (длина 1 байт):

смещение	назначение	длина	структура данных
+00h	*ib_tmr* *ib_rpt*	1	биты 7…4 – *ib_tmr* биты 3…0 – *ib_rpt*

Пример вызова команды:

err = link_packet(CMG_IB, 0, NULL, 0, rxbuf, 1) // чтение параметров iButton

CMS_IB

код 0xEF –установка параметров имитации iButton

Команда **CMS_IB** предназначена для установки параметров имитации iButton в режиме **IBE**. Команда устанавливает *ib_tmr* (старшая тетрада) и *ib_rpt* (младшая тетрада):

ib_tmr – 0…15, максимальное время имитации iButton в тиках по 25млс,

ib_rpt – 0…15, число посылок ID, которые будут переданы за время ib_tmr.

Упаковка параметров в байт

бит 7	бит 6	бит 5	бит 4	бит 3	бит 2	бит 1	бит 0
ib_tmr				*ib_rpt*			

При имитации iButton, ридер выступает в качестве ведомого устройства, к которому ведущее устройство, спрашивающее код, может и не обратиться вследствие неисправности или занятости. Таймер *ib_tmr* ограничивает время пребывания в режиме имитации, чтобы избежать «подвисания» ридера в случае неактивности ведущего устройства. Время имитации расчитывается по формуле:

ib_time = (ib_tmr + 1) * 25млс, минимальное время 25млс, максимальное – 400млс.

Счётчик *ib_rpt* устанавливает необходимое количество повторов кода, т.к. однократной передачи сымитированного кода для большинства устройств недостаточно. Если *ib_rpt* = 0, то при поднесении карты будет передан только один код (0 повторов), если *ib_rpt* = 1, то при поднесении карты будет передано 2 одинаковых кода (1 повтор), и т.д..

Устанавливая количество повторов и/или время имитации можно менять время подключения виртуального брелка к контактору, как если бы брелок был реальным.

Протокол: TC-RD.

Команда:

EF	nn

параметр	назначение	структура данных
nn	*ib_tmr* *ib_rpt*	биты 7…4 – *ib_tmr* биты 3…0 – *ib_rpt*

Пример вызова команды:

err = link_packet(CMS_IB, 0xF3, NULL, 0, rxbuf, 1) // установка ib_tmr=15, ib_rpt=3

CMG_SRD

код 0xEE –чтение параметров SRD (режим простого чтения)

Команда **CMG_SRD** предназначена для чтения параметров режима **SRD** – *AKSD*: ***A****uthorization command*, ***K****ey number*, ***S****tart block/page*, ***D****ata counter & mode* (см. раздел “Режим **SRD**”). Параметр безразличен.

A – ***команда авторизации***; если авторизация не требуется, например, для карт Mifare Ultralight, то A = 0; для авторизация по ключу A карт Mifare Classic, A = 60h, а для авторизации по ключу B, A = 61h.

K – ***номер ключа авторизации*** в памяти ридера (**это не ключ ACS!**); если A = 0, то значение этого параметра безразлично, т.к. авторизация не производится; принимает значения 0…84, хотя значение 0 бессмысленно, т.к. ключ авторизации AK0 находится в RAM.

S – ***стартовый блок/страница*** с которого начнётся авторизация/чтение данных; размер блока – 16 байт, размер страницы – 4 байта.

D – декрементированное ***количество блоков/страниц и режимы чтения данных***, которые надо прочитать; если установлен блоковый режим, то максимальное количество считываемых блоков, которое можно указать - 4, если установлен страничный режим, то максимальное количество страниц для считывания - 16; если установлен только режим авторизации без чтения данных, то значение счётчика считываемых данных безразлично.

Упаковка параметра D в байт

бит 7	бит 6	бит 5	бит 4	бит 3	бит 2	бит 1	бит 0
blkcnt - 1				не исп.	не исп.	*auth_mo*	*blk/pg*

Протокол: TC-RD.

Команда:

EE	xx

Ответ:

смещение	назначение	длина	структура данных
+00h	**A**	1	команда авторизации
+01h	**K**	1	номер ключа авторизации
+02h	**S**	1	стартовый блок/страница
+03h	**D**	1	биты 7…4 - количество блоков/страниц бит 1 – если *auth_mo* = 0, то считываются данные, если *auth_mo* = 1, то производится только авторизация стартового блока/страницы бит 0 – если *blk/pg* = 0, то производятся блоковые операции (длина блока 16 байт), если *blk/pg* = 1, то производятся страничные операции (длина страницы 4 байта)

Пример вызова команды:

err = link_packet(CMG_SRD, 0, NULL, 0, buf_aksd, 4) // чтение AKSD в buf_aksd

CMS_SRD

код 0xED –установка параметров SRD (режим простого чтения)

Команда **CMS_SRD** предназначена для установки параметров режима **SRD** – *AKSD*: ***A****uthorization command*, ***K****ey number*, ***S****tart block/page*, ***D****ata counter & mode* (см. раздел “Режим **SRD**”). Параметр безразличен. Установки сохраняются в энергонезависимой памяти.

A – ***команда авторизации***; если авторизация не требуется, например, для карт Mifare Ultralight, то A = 0; для авторизация по ключу A карт Mifare Classic, A = 60h, а для авторизации по ключу B, A = 61h.

K – ***номер ключа авторизации*** в памяти ридера (**это не ключ ACS!**); если A = 0, то значение этого параметра безразлично, т.к. авторизация не производится; принимает значения 0…84, хотя значение 0 бессмысленно, т.к. ключ авторизации AK0 находится в RAM.

S – ***стартовый блок/страница*** с которого начнётся авторизация/чтение данных; размер блока – 16 байт, размер страницы – 4 байта.

D – декрементированное ***количество блоков/страниц и режимы чтения данных***, которые надо прочитать; если установлен блоковый режим, то максимальное количество считываемых блоков, которое можно указать - 4, если установлен страничный режим, то максимальное количество страниц для считывания - 16; если установлен только режим авторизации без чтения данных, то значение счётчика считываемых данных безразлично.

Протокол: TC-TD.

Команда:

ED	xx

Передаваемый кадр данных (длина 4 байта)

смещение	назначение	длина	структура данных
+00h	**A**	1	команда авторизации
+01h	**K**	1	номер ключа авторизации
+02h	**S**	1	стартовый блок/страница
+03h	**D**	1	биты 7…4 - количество блоков/страниц бит 1 – если *auth_mo* = 0, то считываются данные, если *auth_mo* = 1, то производится только авторизация стартового блока/страницы бит 0 – если *blk/pg* = 0, то производятся блоковые операции (длина блока 16 байт), если *blk/pg* = 1, то производятся страничные операции (длина страницы 4 байта)

Пример вызова команды:

err = link_packet(CMS_SRD, 0, buf_aksd, 4, NULL, 0) // установка AKSD из buf_aksd

CMS_AKEY

код 0xEC –установка ключа авторизации (AK)

Команда **CMS_AKEY** предназначена для записи 6ти байтного ключа авторизации в заданную ячейку энергонезависимой памяти ридера (кроме ключа 0). Считать записанное значение ключа невозможно для обеспечения секретности. Параметр указывает номер ячейки, в которую сохраняется ключ AK, допустимы значения 0…84.

Ключ авторизации AK0 сохраняется в RAM и пропадает при отключении питания. AK0 используется для “быстрых” операций с известными ключами, когда не требуется повышенная секретность.

Протокол: TC-TD.

Команда:

EC	nn

Передаваемый кадр данных (длина 6 байт)

смещение	назначение	длина	структура данных
+00h	значение ключа авторизации	6	6 байт ключа Crypto1, например, 0xA0, 0xA1, 0xA2, 0xA3, 0xA4, 0xA5

Пример вызова команды:

err = link_packet(CMS_AKEY, 2, buf_akey, 6, NULL, 0) // установка значения AK2

CMG_BUF

код 0xEB – чтение приёмного буфера

Команда **CMG_BUF** предназначена для чтения декодированных данных из приёмного буфера (**RXBUF**) в режиме прямого доступа к карте **DIR**. В параметре **nn** указывается длина принимаемого блока 1…86 байт. Команда доступна только в режиме **DIR**.

Протокол: TC-RD.

Команда:

EB	nn

Ответ (длина 1…86 байт, указывается в параметре **nn**):

смещение	назначение	длина	структура данных
+00h	длина данных в буфере	1	длина данных в буфере: 0…85 байт
+01h	данные	1…85	декодированные данные после обмена с картой

Пример вызова команды:

err = link_packet(CMG_BUF, 17, NULL, 0, rxbuf, 17)// чтение байта длины и 16 байт данных

CMS_BUF

код 0xEA – установка данных в буфере передачи

Команда **CMS_BUF** предназначена для установки данных в буфере передачи (**TXBUF**). В параметре **nn** указывается длина передаваемого блока: 2…33 байта. Команда доступна только в режиме **DIR**.

Протокол: TC-TD.

Команда:

EA	nn

Передаваемый кадр данных (длина 2…33 байта, указывается в параметре **nn**):

смещение	назначение	длина	структура данных
+00h	длина	1	длина данных в буфере: 1…32 байт
+01h	данные	1…32	данные для передачи карте

Пример вызова команды:

err = link_packet(CMS_BUF, 2, txbuf, 2, NULL, 0) // установка байта длины и команды

CMG_RAW

код 0xE9 – чтение “сырого” приёмного буфера

Команда **CMG_RAW** предназначена для чтения “сырых” данных после обмена с картой из приёмного буфера (**RAWBUF**). Длина блока данных всегда 340 байт. Параметр безразличен. Команда доступна только в режиме **DIR**.

Протокол: TC-RD.

Команда:

E9	xx

Ответ (длина 340 байт):

смещение	назначение	длина	структура данных
+00h	данные	340	“сырые” данные после обмена с картой

Пример вызова команды:

err = link_packet(CMG_RAW, 0, NULL, 0, rawbuf, 340) // чтение буфера “сырого” кода

CMF_IO

код 0xE8 – обмен данными с картой

Команда **CMF_IO** предназначена для инициирования обмена с картой: в карту передаётся буфер передачи (**TXBUF**), принятые от карты данные помещаются в буфер “сырого” кода (**RAWBUF**). Команда возвращает статус *iost* (ответ длиной 1 байт): если *iost* = 1, то буфер “сырого” кода (**RAWBUF**) содержит данные для декодирования, если *iost* = 0, то данные не были приняты. Команда доступна только в режиме **DIR**.

Упаковка байта статуса операции

бит 7	бит 6	бит 5	бит 4	бит 3	бит 2	бит 1	бит 0
не исп.	не исп.	не исп.	не исп.	не исп.	не исп.	не исп.	*iost*

Протокол: TC-RD.

Команда:

E8	xx

Ответ (длина 1 байт):

смещение	назначение	длина	структура данных
+00h	статус	1	биты 7…1 – незначащие; бит 0 – *iost*, статус операции: если *iost* = 0, то данные не были приняты, если *iost* = 1, то RAW-буфер содержит данные

Пример вызова команды:

err = link_packet(CMF_IO, 0, NULL, 0, buf_st, 1) // обмен с картой и чтение статуса

CMF_D106A

код 0xE7 – декодирование сырого кода ISO14443A, 106КБит/с

Команда **CMF_DECA106** предназначена для декодирования принятых данных в буфере “сырого” кода (**RAWBUF**) и помещения их в приёмный буфер (**RXBUF**). Команда доступна только в режиме **DIR**.

Декодирование выполняется по стандарту ISO-14443A, скорость 106КБит/с. В параметре **nn** упакованы флаги *pm* (бит 1) и *cm* (бит 0):

если pm = 0, то выходной буфер заполняется декодированными данными;
если pm = 1, то выходной буфер заполняется декодированными битами паритета;
если cm = 0, то проверка паритета декодируемых данных не производится;
если cm = 1, то паритет принятых данных будет проверен.

Команда возвращает статус bitlen (ответ длиной 1 байт) – количество декодированных битов в **RXBUF**; RXBUF[0] также содержит длину принятых данных в битах.

Упаковка параметра nn

бит 7	бит 6	бит 5	бит 4	бит 3	бит 2	бит 1	бит 0
не исп.	не исп.	не исп.	не исп.	не исп.	не исп.	*pm*	*cm*

Протокол: TC-RD.

Команда:

E7	nn

Ответ (длина 1 байт):

смещение	назначение	длина	структура данных
+00h	bitlen	1	длина декодированных данных в RX-буфере

Пример вызова команды:

err = link_packet(CMF_D106A, 1, NULL, 0, bitlen, 1) // декодирование данных с паритетом

CMF_ACRCA

код 0xE6 – добавление CRC по ISO14443A к данным в TX - буфере

Команда **CMF_ACRCA** предназначена для добавления CRC по алгоритму стандарта ISO14443A к данным, находящимся в буфере передачи (**TX**-буфере). Команда возвращает статус *iost* (ответ длиной 1 байт): если *iost* = 1, то CRC успешно добавлена к данным **TX**-буфера, если *iost* = 0, то добавить CRC не удалось. При добавлении CRC, также корректируется длина данных в **TX**-буфере. Добавлять паритет можно только к “байтным” данным (паритет вычисляется по 8ми битам). Команда доступна только в режиме **DIR**. Параметр безразличен.

Упаковка байта статуса операции

бит 7	бит 6	бит 5	бит 4	бит 3	бит 2	бит 1	бит 0
не исп.	не исп.	не исп.	не исп.	не исп.	не исп.	не исп.	*iost*

Протокол: TC-RD.

Команда:

E6	xx

Ответ (длина 1 байт):

смещение	назначение	длина	структура данных
+00h	статус	1	биты 7…1 – незначащие; бит 0 – *iost*, статус операции: если *iost* = 0, то добавить CRC не удалось, если *iost* = 1, то CRC успешно добавлена

Пример вызова команды:

err = link_packet(CMF_ACRCA, 0, NULL, 0, buf_st, 1) // добавление CRC

CMF_CCRCA

код 0xE5 – проверка CRC данных в RX - буфере по ISO14443A

Команда **CMF_CCRCA** предназначена для проверки правильности CRC в данных в **RX**-буфере по стандарту ISO14443A. Команда возвращает статус *iost* (ответ длиной 1 байт): если *iost* = 1, то проверка CRC успешна, если *iost* = 0, то проверка CRC закончилась неудачей. Команда доступна только в режиме **DIR**. Параметр безразличен.

Упаковка байта статуса операции

бит 7	бит 6	бит 5	бит 4	бит 3	бит 2	бит 1	бит 0
не исп.	не исп.	не исп.	не исп.	не исп.	не исп.	не исп.	*iost*

Протокол: TC-RD.

Команда:

E5	xx

Ответ (длина 1 байт):

смещение	назначение	длина	структура данных
+00h	статус	1	биты 7…1 – незначащие; бит 0 – *iost*, статус операции: если *iost* = 0, то проверка CRC неудачна, если *iost* = 1, то проверка CRC успешна

Пример вызова команды:

err = link_packet(CMF_CCRCA, 0, NULL, 0, buf_st, 1) // проверка CRC в RX-буфере

CMF_APAR

код 0xE4 – добавление паритета к данным в TX - буфере

Команда **CMF_APAR** предназначена для добавления битов паритета к данным в буфере передачи (**TX**-буфере). Команда возвращает статус *iost* (ответ длиной 1 байт): если *iost* = 1, то биты паритета успешно добавлены, если *iost* = 0, то добавление битов паритета закончилось неудачей. Команда доступна только в режиме **DIR**. Параметр безразличен.

Упаковка байта статуса операции

бит 7	бит 6	бит 5	бит 4	бит 3	бит 2	бит 1	бит 0
не исп.	не исп.	не исп.	не исп.	не исп.	не исп.	не исп.	*iost*

Протокол: TC-RD.

Команда:

E4	xx

Ответ (длина 1 байт):

смещение	назначение	длина	структура данных
+00h	статус	1	биты 7…1 – незначащие; бит 0 – *iost*, статус операции: если *iost* = 0, то добавление паритета неудачно, если *iost* = 1, то биты паритета успешно добавлены

Пример вызова команды:

err = link_packet(CMF_APAR, 0, NULL, 0, buf_st, 1) // добавление паритета к TX-данным

CMG_ISO

код 0xE3 – чтение параметров ввода-вывода, чтение расширенных флагов ошибок

Команда **CMG_ISO** предназначена для получения состояния фазового детектора, флага активности несущей 13.56МГц и расширенных флагов ошибок. Команда доступна только в режиме **DIR**. Параметр безразличен.

Описание считанных параметров ввода-вывода по смещению +00h:

coff - ***c****arrier* ***off***, флаг выключения несущей, если 1, то несущая выключена;
phdf - ***ph****ase* ***d****etector level* ***f****orce*, флаг принудительной установки детектора;
phd - ***ph****ase* ***d****etector level*, уровень фазового детектора 0...31.

Описание считанных расширенных флагов ошибок ***erf*** по смещению +01h:

wr_fail - устанавливается в 1, если была ошибка записи;
wr_ok - устанавливается в 1, если запись была успешно завершена;
rd_fail - устанавливается в 1, если была ошибка чтения;
rd_ok - устанавливается в 1, если чтение было успешно завершено;
auth_fail - устанавливается в 1, если была ошибка авторизации;
auth_ok - устанавливается в 1, если авторизация была успешно пройдена;
sel_fail - устанавливается в 1, если была ошибка выбора карты;
sel_ok - устанавливается в 1, если выбор карты успешно завершён.

Расширенные флаги ошибок ***erf*** (***e****xtended* ***r****ror* ***f****lags*) используются для диагностирования ошибок при выполнении сложных операций: SELECT – ANTICOLLISION, AUTH, READ, WRITE.

Перед подключением к карте (фаза SELECT - ANTICOLLISION), расширенные флаги ошибок обнуляются. Далее, если фаза ANTICOLLISION проходит успешно, устанавливается флаг *sel_ok*, если фаза ANTICOLLISION завершается неудачей, устанавливается флаг *sel_fail*.

Прохождение фазы авторизации устанавливает флаги *auth_ok* или *auth_fail*, прохождение чтения устанавливает *rd_ok* или *rd_fail*, прохождение записи *wr_ok* или *wr_fail*.

Протокол: TC-RD.

Команда:

E3	xx

Ответ (длина 2 байта):

Упаковка флагов в байте по смещению +00h: расширенные флаги ошибки - erf

бит 7	бит 6	бит 5	бит 4	бит 3	бит 2	бит 1	бит 0
wr_fail	wr_ok	rd_fail	rd_ok	auth_fail	auth_ok	sel_fail	sel_ok

Упаковка флагов в байте по смещению +01h

бит 7	бит 6	бит 5	бит 4	бит 3	бит 2	бит 1	бит 0
coff	не исп.	*phdf*	*phd*				

Пример вызова команды:

err = link_packet(CMG_ISO, 0, NULL, 0, buf_iso, 2) // чтение спец. параметров

CMS_ISO

код 0xE2 – установка параметров ввода-вывода, сброс расширенных флагов ошибок

Команда **CMS_ISO** предназначена для установки нового состояния фазового детектора и флага активности несущей 13.56МГц. В параметре указываются три значения: *coff*, *phdf*, *phd*. Вызов команды обнуляет расширенные флаги ошибок (см. **CMG_ISO**). Команда доступна только в режиме **DIR**.

Описание параметров ввода-вывода **nn**:

coff - ***c****arrier* ***off***, флаг выключения несущей, если 1, то несущая выключена;
phdf - ***ph****ase* ***d****etector level* ***f****orce*, флаг принудительной установки детектора;
phd - ***ph****ase* ***d****etector level*, уровень фазового детектора 0...31.

Упаковка параметра nn

бит 7	бит 6	бит 5	бит 4	бит 3	бит 2	бит 1	бит 0
coff	не исп.	*phdf*	*phd*				

Протокол: ТС.

Команда:

E2	nn

Пример вызова команды:

```
err = link_packet(CMS_ISO, 0x80, NULL, 0, NULL, 0)          // выключение несущей
```

CMS_CBLK

код 0xE1 – блоковая запись (длина блока 16 байт)

Команда **CMS_CBLK** высокоуровневая команда для записи блока или группы блоков (не более 4х) по команде 0xA0 (PICC_WRITE16). Команда ориентирована на запись блоков карт Mifare Classic, Mifare Plus. Вызов команды вызывает выключение-включение несущей 13.56МГц, подключение к карте (фаза SELECT - ANTICOLLISION), авторизацию (если требуется), запись блока или группы блоков по 16 байт. В случае неудачи записи, по расширенным флагам ошибок можно установить, что послужило причиной ошибки (см. **CMG_ISO**). Команда доступна только в режиме **DIR**.

В параметре команды указывается количество блоков *blknum* для записи, таймаут операции *timeout* и флаг работы с произвольной картой или только с картой с заданным UID - *anyuid*. Если UID задан (*anyuid* = 0), то он должен быть передан в ридер в кадре данных вместе с параметрами записи. Запись нескольких блоков производится последовательно, начиная со стартового блока, переходя к блоку с большим номером. В случае сбоя записи, выполняется повторное подключение к карте с авторизацией ещё не записанного блока.

Упаковка параметра nn

бит 7	бит 6	бит 5	бит 4	бит 3	бит 2	бит 1	бит 0
не исп.	не исп.	*blknum*		*anyuid*	*timeout*		

Количество записываемых блоков определяется декрементированным значением *blknum*: если записывается 1 блок, то *blknum* = 0; если записываются 2 блока, то *blknum* = 1, и т.д.. Время таймаута вычисляется по формуле: *time* = (*timeout* * 4 + 3) * 50млс, минимальное время – 0.15с, максимальное – 1.55с.

Длина кадра данных, передаваемых в ридер, зависит от флага *anyuid* и количества записываемых блоков *blknum*. Порядок формирования кадра данных TD такой:

1) если *anyuid* = 0, то передаваемый кадр начинается с 11ти байт UID: uidlen(1б) + uid(10б);
2) далее идут 4 байта параметров ***CAKS***, если *anyuid* = 1, то кадр начнётся с ***CAKS***, а не с UID, т.к. установка UID не требуется;
3) далее идут данные записываемых блоков, длина данных рассчитывается как *datalen* = (*blknum* + 1) * 16.

Суммарная длина кадра TD: *framelen* = ((*anyuid* ==0) ? 15 : 4) + (*blknum* + 1) * 16;

Параметры ***CAKS*** (4 байта) используются в высокоуровневых командах чтения-записи блоков: ***C****onnect command*, ***A****uthorization command*, ***K****ey number*, ***S****tart block/page*.

C – ***команда подключения***; для стандартных карт это код 0x26 (PICC_REQIDL) или 0x52 (PICC_REQALL); однако, для активации возможностей RW-карт, карту следует активировать с помощью кода 0x40 (PICC_REQRW);

A – ***команда авторизации***; если авторизация не требуется, например для карт Mifare Ultralight, то A = 0; для авторизация по ключу A карт Mifare Classic, A = 60h, а для авторизации по ключу B, A = 61h;

K – ***номер ключа авторизации*** в памяти ридера (**это не ключ ACS!**); если A = 0, то значение этого параметра безразлично, т.к. авторизация не производится; принимает значения 0…84, хотя значение 0 бессмысленно, т.к. ключ авторизации AK0 находится в RAM;

S – ***стартовый блок/страница*** с которого начнётся авторизация/чтение данных; размер блока – 16 байт, размер страницы – 4 байта.

CMS_CBLK *- продолжение*

код 0xE1 – блоковая запись

Структура кадра, передаваемого ридеру: ***UID***(0 | 11б) + ***CAKS***(4б) + ***DATA***(16…64б). Данные ***UID*** отсутствуют при *anyuid* = 1, т.е. кадр начинается с ***CAKS***. Минимальная длина кадра – 20 байт (***CAKS***(4б) + ***DATA***(16б)), максимальная длина кадра – 79 байт.

В случае успешной записи, команда вернёт 14 байт описателя карты ***CD*** (***C****ard* ***D****escriptor*), в которую произведена запись: uidlen(1б), uid(10б), sak(1б), atqa(2б).

Упаковка параметра nn

бит 7	бит 6	бит 5	бит 4	бит 3	бит 2	бит 1	бит 0
не исп.	не исп.	*blknum*		*anyuid*	*timeout*		

Протокол: TC-TD-RD.

Команда:

E1	nn

Передаваемый кадр данных (длина 20…79 байт):

смещение	назначение	длина	структура данных
+00h	uidlen	1 | 0	длина uid, 4…10, передаётся только при *anyuid* = 0
+01h	uid	10 | 0	UID карты в которую производится запись, передаётся только при *anyuid* = 0
+0Bh | +00h	***CAKS***	4	обязательный параметр ***CAKS***, если *anyuid* = 1, то кадр данных начинается с этого параметра
+0Eh/+15h	блоковые данные	16…64	записываемые данные блоков, начиная с младшего

Ответ – ***CD***, ***C****ard* ***D****escriptor* (длина 14 байт):

смещение	назначение	длина	структура данных
+00h	uidlen	1	бит 7 – равен 1, если RW-карта биты 6…4 – зарезервированы, равны 0 биты 3…0 – длина данных в поле uid (4, 7, 10)
01h	uid	10	UID карты, незначащие байты заполнены нулями
+0Bh	sak	1	SAK (**Select acknowledge**, см.ISO14443A), значение SAK используется при определении типа карты
+0Ch	atqa	2	ATQA (**Answer to** request **A**, см.ISO14443A), значение ATQA используется при определении типа

Пример вызова команды (также реализацию можно посмотреть в ***mf0xAB.cpp***):

```
err = link_packet(CMS_CBLK, 0x00 | 8 | 7, txbuf, 4+16, rxbuf, 14)  // запись блока карты
```

CMG_CBLK

код 0xE0 – блоковое чтение (длина блока 16 байт)

Команда **CMG_CBLK** высокоуровневая команда для чтения блока или группы блоков (не более 4х) по команде 0x30 (PICC_READ16). Команда ориентирована на запись блоков карт Mifare Classic, Mifare Plus. Вызов команды вызывает выключение-включение несущей 13.56МГц, подключение к карте (фаза SELECT - ANTICOLLISION), авторизацию (если требуется), чтение блока или группы блоков по 16 байт. В случае неудачи чтения, по расширенным флагам ошибок можно установить, что послужило причиной ошибки (см. **CMG_ISO**). Команда доступна только в режиме **DIR**.

В параметре команды указывается количество блоков *blknum* для чтения, таймаут операции *timeout* и флаг работы с произвольной картой или только с картой с заданным UID - *anyuid*. Если UID задан (*anyuid* = 0), то он должен быть передан в ридер в кадре данных вместе с параметрами чтения. Чтение нескольких блоков производится последовательно, начиная со стартового блока, переходя к блоку с большим номером. В случае сбоя чтения, выполняется повторное подключение к карте с авторизацией ещё не считанного блока.

Упаковка параметра nn

бит 7	бит 6	бит 5	бит 4	бит 3	бит 2	бит 1	бит 0
не исп.	не исп.	*blknum*		*anyuid*	*timeout*		

Количество считываемых блоков определяется декрементированным значением *blknum*: если считывается 1 блок, то *blknum* = 0; если считываются 2 блока, то *blknum* = 1, и т.д.. Время таймаута вычисляется по формуле: *time* = (*timeout* * 4 + 3) * 50млс, минимальное время – 0.15с, максимальное – 1.55с.

Длина кадра данных, передаваемых в ридер, зависит от флага *anyuid*. Порядок формирования кадра данных TD такой:

1) если *anyuid* = 0, то передаваемый кадр начинается с 11ти байт UID: uidlen(1б) + uid(10б);
2) далее идут 4 байта параметров ***CAKS***, если *anyuid* = 1, то кадр начнётся с ***CAKS***, а не с UID, т.к. установка UID не требуется;

Суммарная длина кадра TD: *framelen* = (*anyuid* ==0) ? 15 : 4;

Параметры ***CAKS*** (4 байта) используются в высокоуровневых командах чтения-записи блоков: ***C****onnect command*, ***A****uthorization command*, ***K****ey number*, ***S****tart block/page*.

C – ***команда подключения***; для стандартных карт это код 0x26 (PICC_REQIDL) или 0x52 (PICC_REQALL); однако, для активации возможностей RW-карт, карту следует активировать с помощью кода 0x40 (PICC_REQRW);

A – ***команда авторизации***; если авторизация не требуется, например для карт Mifare Ultralight, то A = 0; для авторизация по ключу A карт Mifare Classic, A = 60h, а для авторизации по ключу B, A = 61h;

K – ***номер ключа авторизации*** в памяти ридера (**это не ключ ACS!**); если A = 0, то значение этого параметра безразлично, т.к. авторизация не производится; принимает значения 0…84, хотя значение 0 бессмысленно, т.к. ключ авторизации AK0 находится в RAM;

S – ***стартовый блок/страница*** с которого начнётся авторизация/чтение данных; размер блока – 16 байт, размер страницы – 4 байта.

***CMG_CBLK** - продолжение*

код 0xE0 – блоковое чтение

Структура кадра, передаваемого ридеру: ***UID***(0 | 11б) + ***CAKS***(4б). Данные ***UID*** отсутствуют при *anyuid* = 1, т.е. кадр начинается с ***CAKS***. Минимальная длина кадра – 4 байта (***CAKS***(4б)), максимальная длина кадра – 15 байт.

В случае успешного чтения, команда вернёт 14 байт описателя карты ***CD*** (***C**ard **D**escriptor*), из которой производилось чтение, и данные, длина которых определяется количеством считываемых блоков: *datalen* = (*blknum* + 1) * 16. Минимальная длина кадра – 30 байт (***CD***(14б) + ***DATA***(16б)), максимальная длина кадра – 78 байт (***CD***(14б) + ***DATA***(64б)).

Упаковка параметра nn

бит 7	бит 6	бит 5	бит 4	бит 3	бит 2	бит 1	бит 0
не исп.	не исп.	*blknum*		*anyuid*	*timeout*		

Протокол: TC-TD-RD.

Команда:

E0	nn

Передаваемый кадр данных (длина 4…15 байт):

смещение	назначение	длина	структура данных
+00h	uidlen	1 | 0	длина uid, 4…10, передаётся только при *anyuid* = 0
+01h	uid	10 | 0	UID карты в которую производится запись, передаётся только при *anyuid* = 0
+0Bh | +00h	***CAKS***	4	обязательный параметр ***CAKS***, если *anyuid* = 1, то кадр данных начинается с этого параметра

Ответ – ***CD***, ***C**ard **D**escriptor* и данные (длина 30…78 байт):

смещение	назначение	длина	структура данных
+00h	uidlen	1	бит 7 – равен 1, если RW-карта биты 6…4 – зарезервированы, равны 0 биты 3…0 – длина данных в поле uid (4, 7, 10)
01h	uid	10	UID карты, незначащие байты заполнены нулями
+0Bh	sak	1	SAK (**Select acknowledge**, см.ISO14443A), значение SAK используется при определении типа карты
+0Ch	atqa	2	ATQA (**Answer to** request **A**, см.ISO14443A), значение ATQA используется при определении типа
+0Eh	блоковые данные	16…64	считываемые данные блоков, начиная с младшего

Пример вызова команды (также реализацию можно посмотреть в ***mf0xAB.cpp***):

err = link_packet(CMG_CBLK, 0x00 | 8 | 7, txbuf, 4, rxbuf, 14+16) // чтение блока карты

CMS_CPGE

код 0xDF – страничная запись (длина страницы 4 байта)

Команда **CMS_CPGE** высокоуровневая команда для записи страницы или группы страниц (не более 16ти) по команде 0xA2 (PICC_WRITE4). Команда ориентирована на запись карт Mifare Ultralight, Mifare Ultralight C. Вызов команды вызывает выключение-включение несущей 13.56МГц, подключение к карте (фаза SELECT - ANTICOLLISION), авторизацию (если требуется), запись страницы или группы страниц по 4 байта. В случае неудачи записи, по расширенным флагам ошибок можно установить, что послужило причиной ошибки (см. **CMG_ISO**). Команда доступна только в режиме **DIR**.

В параметре команды указывается количество страниц *pgnum* для записи, таймаут операции *timeout* и флаг работы с произвольной картой или только с картой с заданным UID - *anyuid*. Если UID задан (*anyuid* = 0), то он должен быть передан в ридер в кадре данных вместе с параметрами записи. Запись нескольких страниц производится последовательно, ***начиная с конечной*** страницы, переходя к странице с ***меньшим*** номером. В случае сбоя записи, выполняется повторное подключение к карте с авторизацией ещё не записанной страницы.

Упаковка параметра nn

бит 7	бит 6	бит 5	бит 4	бит 3	бит 2	бит 1	бит 0
pgnum				*anyuid*	*timeout*		

Количество записываемых страниц определяется декрементированным значением *pgnum*: если записывается 1 страница, то *pgnum* = 0; если записываются 2 страницы, то *pgnum* = 1, и т.д.. Время таймаута вычисляется по формуле: *time* = (*timeout* * 4 + 3) * 50млс, минимальное время – 0.15с, максимальное – 1.55с.

Длина кадра данных, передаваемых в ридер, зависит от флага *anyuid* и количества записываемых страниц *pgnum*. Порядок формирования кадра данных TD такой:

4) если *anyuid* = 0, то передаваемый кадр начинается с 11ти байт UID: uidlen(1б) + uid(10б);
5) далее идут 4 байта параметров ***CAKS***, если *anyuid* = 1, то кадр начнётся с ***CAKS***, а не с UID, т.к. установка UID не требуется;
6) далее идут данные записываемых страниц, длина данных рассчитывается как *datalen* = (*pgnum* + 1) * 4.

Суммарная длина кадра TD: *framelen* = ((*anyuid* ==0) ? 15 : 4) + (*pgnum* + 1) * 4;

Параметры ***CAKS*** (4 байта) используются в высокоуровневых командах чтения-записи блоков: ***C****onnect command*, ***A****uthorization command*, ***K****ey number*, ***S****tart block/page*.

C – ***команда подключения***; для стандартных карт это код 0x26 (PICC_REQIDL) или 0x52 (PICC_REQALL); однако, для активации возможностей RW-карт, карту следует активировать с помощью кода 0x40 (PICC_REQRW);

A – ***команда авторизации***; если авторизация не требуется, например для карт Mifare Ultralight, то A = 0; для авторизация по ключу A карт Mifare Classic, A = 60h, а для авторизации по ключу B, A = 61h;

K – ***номер ключа авторизации*** в памяти ридера (**это не ключ ACS!**); если A = 0, то значение этого параметра безразлично, т.к. авторизация не производится; принимает значения 0…84, хотя значение 0 бессмысленно, т.к. ключ авторизации AK0 находится в RAM;

S – ***стартовый блок/страница*** с которого начнётся авторизация/чтение данных; размер блока – 16 байт, размер страницы – 4 байта.

***CMS_CPGE** - продолжение*

код 0xDF – страничная запись

Структура кадра, передаваемого ридеру: ***UID***(0 | 11б) + ***CAKS***(4б) + ***DATA***(4…64б). Данные ***UID*** отсутствуют при *anyuid* = 1, т.е. кадр начинается с ***CAKS***. Минимальная длина кадра – 8 байт (***CAKS***(4б) + ***DATA***(4б)), максимальная длина кадра – 79 байт.

В случае успешной записи, команда вернёт 14 байт описателя карты ***CD*** (***C**ard **D**escriptor*), в которую произведена запись: uidlen(1б), uid(10б), sak(1б), atqa(2б).

Упаковка параметра nn

бит 7	бит 6	бит 5	бит 4	бит 3	бит 2	бит 1	бит 0
pgnum				*anyuid*	*timeout*		

Протокол: TC-TD-RD.

Команда:

DF	nn

Передаваемый кадр данных (длина 8…79 байт):

смещение	назначение	длина	структура данных
+00h	uidlen	1 | 0	длина uid, 4…10, передаётся только при *anyuid* = 0
+01h	uid	10 | 0	UID карты в которую производится запись, передаётся только при *anyuid* = 0
+0Bh | +00h	***CAKS***	4	обязательный параметр ***CAKS***, если *anyuid* = 1, то кадр данных начинается с этого параметра
+0Eh/+15h	страничные данные	4…64	данные записываемых страниц, начиная с младшей

Ответ – ***CD***, ***C**ard **D**escriptor* (длина 14 байт):

смещение	назначение	длина	структура данных
+00h	uidlen	1	бит 7 – равен 1, если RW-карта биты 6…4 – зарезервированы, равны 0 биты 3…0 – длина данных в поле uid (4, 7, 10)
01h	uid	10	UID карты, незначащие байты заполнены нулями
+0Bh	sak	1	SAK (**Select acknowledge**, см.ISO14443A), значение SAK используется при определении типа карты
+0Ch	atqa	2	ATQA (**Answer to** request **A**, см.ISO14443A), значение ATQA используется при определении типа

Пример вызова команды (также реализацию можно посмотреть в ***mf0xAB.cpp***):

```
err = link_packet(CMS_CPGE, 0x00 | 8 | 7, txbuf, 4+4, rxbuf, 14)   // запись страницы
```

CMG_CPGE

код 0xDE – страничное чтение (длина страницы 4 байта)

Команда **CMG_CPGE** высокоуровневая команда для чтения блока или группы блоков (не более 4х) по команде 0x30 (PICC_READ16). Команда ориентирована на запись карт Mifare Ultralight, Mifare Ultralight C. Вызов команды вызывает выключение-включение несущей 13.56МГц, подключение к карте (фаза SELECT - ANTICOLLISION), авторизацию (если требуется), чтение страницы или группы страницы по 4 байта. В случае неудачи чтения, по расширенным флагам ошибок можно установить, что послужило причиной ошибки (см. **CMG_ISO**). Команда доступна только в режиме **DIR**.

В параметре команды указывается количество страниц *pgnum* для чтения, таймаут операции *timeout* и флаг работы с произвольной картой или только с картой с заданным UID - *anyuid*. Если UID задан (*anyuid* = 0), то он должен быть передан в ридер в кадре данных вместе с параметрами чтения. Чтение нескольких страниц производится последовательно, начиная со стартовой страницы, переходя к странице с большим номером. В случае сбоя чтения, выполняется повторное подключение к карте с авторизацией ещё не считанной страницей (если требуется авторизация).

Упаковка параметра nn

бит 7	бит 6	бит 5	бит 4	бит 3	бит 2	бит 1	бит 0
pgnum				*anyuid*	*timeout*		

Количество считываемых страниц определяется декрементированным значением *pgnum*: если считывается 1 блок, то *pgnum* = 0; если считываются 2 блока, то *pgnum* = 1, и т.д.. Время таймаута вычисляется по формуле: *time* = (*timeout* * 4 + 3) * 50млс, минимальное время – 0.15с, максимальное – 1.55с.

Длина кадра данных, передаваемых в ридер, зависит от флага *anyuid*. Порядок формирования кадра данных TD такой:

1) если *anyuid* = 0, то передаваемый кадр начинается с 11ти байт UID: uidlen(1б) + uid(10б);
2) далее идут 4 байта параметров ***CAKS***, если *anyuid* = 1, то кадр начнётся с ***CAKS***, а не с UID, т.к. установка UID не требуется;

Суммарная длина кадра TD: *framelen* = (*anyuid* ==0) ? 15 : 4;

Параметры ***CAKS*** (4 байта) используются в высокоуровневых командах чтения-записи блоков: ***C****onnect command*, ***A****uthorization command*, ***K****ey number*, ***S****tart block/page*.

C – ***команда подключения***; для стандартных карт это код 0x26 (PICC_REQIDL) или 0x52 (PICC_REQALL); однако, для активации возможностей RW-карт, карту следует активировать с помощью кода 0x40 (PICC_REQRW);

A – ***команда авторизации***; если авторизация не требуется, например для карт Mifare Ultralight, то A = 0; для авторизация по ключу A карт Mifare Classic, A = 60h, а для авторизации по ключу B, A = 61h;

K – ***номер ключа авторизации*** в памяти ридера (**это не ключ ACS!**); если A = 0, то значение этого параметра безразлично, т.к. авторизация не производится; принимает значения 0…84, хотя значение 0 бессмысленно, т.к. ключ авторизации AK0 находится в RAM;

S – ***стартовый блок/страница*** с которого начнётся авторизация/чтение данных; размер блока – 16 байт, размер страницы – 4 байта.

CMG_CPGE *- продолжение*

код 0xDE – страничное чтение

Структура кадра, передаваемого ридеру: ***UID***(0 | 11б) + ***CAKS***(4б). Данные ***UID*** отсутствуют при *anyuid* = 1, т.е. кадр начинается с ***CAKS***. Минимальная длина кадра – 4 байта (***CAKS***(4б)), максимальная длина кадра – 15 байт.

В случае успешного чтения, команда вернёт 14 байт описателя карты ***CD*** (***C****ard* ***D****escriptor*), из которой производилось чтение, и данные, длина которых определяется количеством считываемых страниц: *datalen* = (*pgnum* + 1) * 4. Минимальная длина кадра – 18 байт (***CD***(14б) + ***DATA***(4б)), максимальная длина кадра – 78 байт (***CD***(14б) + ***DATA***(64б)).

Упаковка параметра nn

бит 7	бит 6	бит 5	бит 4	бит 3	бит 2	бит 1	бит 0
pgnum				*anyuid*	*timeout*		

Протокол: TC-TD-RD.

Команда:

DE	nn

Передаваемый кадр данных (длина 4…15 байт):

смещение	назначение	длина	структура данных
+00h	uidlen	1 | 0	длина uid, 4…10, передаётся только при *anyuid* = 0
+01h	uid	10 | 0	UID карты в которую производится запись, передаётся только при *anyuid* = 0
+0Bh | +00h	***CAKS***	4	обязательный параметр ***CAKS***, если *anyuid* = 1, то кадр данных начинается с этого параметра

Ответ – ***CD***, ***C****ard* ***D****escriptor* и данные (длина 18…78 байт):

смещение	назначение	длина	структура данных
+00h	uidlen	1	бит 7 – равен 1, если RW-карта биты 6…4 – зарезервированы, равны 0 биты 3…0 – длина данных в поле uid (4, 7, 10)
01h	uid	10	UID карты, незначащие байты заполнены нулями
+0Bh	sak	1	SAK (**Select acknowledge**, см.ISO14443A), значение SAK используется при определении типа карты
+0Ch	atqa	2	ATQA (**Answer to** request **A**, см.ISO14443A), значение ATQA используется при определении типа
+0Eh	страничные данные	4…64	данные считываемых страниц, начиная с младшего

Пример вызова команды (также реализацию можно посмотреть в ***mf0xAB.cpp***):

```
err = link_packet(CMG_CPGE, 0x30 | 8 | 7, txbuf, 4, rxbuf, 14+16) // чтение 4х страниц
```

CMF_CSEL

код 0xDD – подключение к карте, выполнение антиколлизии

Команда **CMF_CSEL** высокоуровневая команда подключения к карте. Вызов команды вызывает выключение-включение несущей 13.56МГц, подключение к карте (фаза SELECT - ANTICOLLISION). При вызове команды указывается код подключения к карте (7 бит), обычно это команды 0x26 (PICC_REQIDL), 0x52 (PICC_REQALL), 0x40 (PICC_REQRW), но могут быть указаны и другие значения. Фаза антиколлизии выполняется для подключения на скорости 106Кбит/с. В случае неудачи подключения можно сделать вывод об отсутствии карты. Команда сбрасывает и в процессе работы изменяет расширенные флаги ошибок ***erf***: *sel_fail* и *sel_ok*. Команда доступна только в режиме **DIR**.

В параметре команды указывается таймаут операции *timeout* и флаг работы с произвольной картой или только с картой с заданным UID - *anyuid*. Если UID задан (*anyuid* = 0), то он должен быть передан в ридер в кадре TD. В случае сбоя на любой фазе SELECT - ANTICOLLISION, выполняется повторное подключение к карте до истечения времени таймаута.

Упаковка параметра nn

бит 7	бит 6	бит 5	бит 4	бит 3	бит 2	бит 1	бит 0
не исп.	не исп.	не исп.	не исп.	*anyuid*	*timeout*		

Время таймаута вычисляется по формуле: *time* = (*timeout* * 4 + 3) * 50млс, минимальное время – 0.15с, максимальное – 1.55с. Длина кадра данных, передаваемых в ридер, зависит от флага *anyuid*. Порядок формирования кадра данных TD такой:

1) если *anyuid* = 0, то передаваемый кадр начинается с 11ти байт UID: uidlen(1б) + uid(10б);
2) далее идёт урезанный до 1го байта параметр ***CAKS***, состоящий только из кода ***C*** – ***команды подключения***; если *anyuid* = 1, то кадр начнётся с ***C***, а не с UID, т.к. установка UID не требуется;

Суммарная длина кадра TD: *framelen* = (*anyuid* ==0) ? 12 : 1;

Протокол: TC-TD-RD.

Команда:

DD	nn

Передаваемый кадр данных (длина 1…12 байт)

смещение	назначение	длина	структура данных
+00h	uidlen	1 | 0	длина uid, 4…10, передаётся только при *anyuid* = 0
+01h	uid	10 | 0	UID карты в которую производится запись, передаётся только при *anyuid* = 0
+0Bh | +00h	***C***	1	обязательный параметр ***C***, если *anyuid* = 1, то кадр данных начинается с этого параметра

Ответ – ***CD***, ***Card Descriptor*** (см. **CMS_CPGE**, например), длина 14 байт.

Пример вызова команды (также реализацию можно посмотреть в ***mf0xAB.cpp***):

```
err = link_packet(CMF_CSEL, 8 | 7, txbuf, 1, rxbuf, 14)        // подключение к карте
```

CMF_CAUTH

код 0xDC – авторизация блока/страницы по алгоритму Crypto1

Команда **CMF_CAUTH** высокоуровневая команда авторизации. Вызов команды производится после фазы SELECT-ANTICOLLISION. При вызове команды указывается код команды авторизации и номер ключа авторизации AK 0…84. Обычно используется код команды авторизации 0x60 - PICC_AUTHENT1A или 0x61 - PICC_AUTHENT1B, но могут быть указаны и другие значения. В случае неудачи авторизации можно сделать вывод об ошибочном значении ключа. В случае успешной авторизации, команда устанавливает внутренний флаг обязательного шифрования данных, который будет сброшен только при повторном подключении к карте. Команда в процессе работы изменяет расширенные флаги ошибок ***erf***: *auth_fail* и *auth_ok*. Команда доступна только в режиме **DIR**.

В параметре команды указывается код команды авторизации, а в 2х байтах кадра команды TD – номер ключа авторизации AK (0…84) и номер блока/страницы для авторизации.

Упаковка параметра nn

бит 7	бит 6	бит 5	бит 4	бит 3	бит 2	бит 1	бит 0
код команды авторизации							

Протокол: TC-TD.

Команда:

DC	nn

Передаваемый кадр данных (длина 2 байта):

смещение	назначение	длина	структура данных
+00h	***K***	1	номер ключа авторизации ak 0…84
+01h	***S***	1	блок/страница для авторизации 0…255

Пример вызова команды (также реализацию можно посмотреть в ***mf0xAB.cpp***):

err = link_packet(CMF_CAUTH, auth_cmd, txbuf, 2,NULL, 0) // авторизация блока

CMF_LCMD

код 0xDB – выполнение длинной команды

Команда **CMF_LCMD** высокоуровневая команда, служит для передачи команды карте и чтения результатов ответа. Вызов команды производится после фазы SELECT-ANTICOLLISION. При вызове команды указывается параметр (длина команды) и кадр данных – команду. Перед передачей команды, к ней добавляется контрольная сумма, CRC по алгоритму стандарта ISO14443A и биты паритета. Принятые после выполнения команды данные сразу декодируются (паритет не проверяется), после чего возвращается длина декодированных данных в битах. RXBUF[0] также содержит длину принятых данных в битах. Ответ можно считать командой **CMG_BUF**. Команда доступна только в режиме **DIR**.

Протокол: TC-TD-RD.

Команда:

DB	nn

Передаваемый кадр данных (длина nn байт):

смещение	назначение	длина	структура данных
+00h	команда	nn	команда, передаваемая карте

Ответ (длина 1 байт):

смещение	назначение	длина	структура данных
+00h	длина	1	длина ответа карты в битах

Пример вызова команды:

err = link_packet(CMF_LCMD, 2, txbuf, 2, bitlen, 1) *// передача команды карте*