Динамическая библиотека PRP18.DLL

Руководство программиста

Версия 2.2 февраль 2023 г.

Оглавление

История документа

Таблица 1:

Версия	Дата	Изменения
1.0	28.05.2018	Первая редакция документа
1.1	21.05.2019	Передана в производство
1.2	24.07.2019	Исправления в активации карт уровня 4 (ошибка типа ATS)
2.0	01.10.2019	Функции поддержки карт военного, ASN банковских, I-Code, Parsec smart.
2.1	16.09.2020	Добавлена функция чтения ПАН номера банковских карт
2.2	16.02.2023	Расширение функционала для поддержки PR-X18

Общие положения

Назначение

Данный документ представляет собой руководство для программистов, разрабатывающих приложения с поддержкой настольных считывателей PR-P18 в режиме FTDI устройства (без использования виртуального COM-порта) в операционной системе Windows.

С версии библиотеки 5.6.6.6 добавлена поддержка мультиформатных настольных считывателей PR-X18, совместимых по функционалу работы с ВЧ картами с моделью PR-P18, которые на момент написания документа планируются к снятию с производства. Как USB устройство, новый считыватель (PR-X18) имеет другой PID (0xE3B6 вместо 0xE3B5).

Отграничения документа

При описании функционала библиотеки некоторые функции, связанные с реализацией специальных проектов, опущены, поскольку для общего применения они не имеют смысла.

Дополнительная информация

Для упрощения понимания работы считывателя с данной библиотекой в составе SDK поставляется демонстрационный пример, написанный на языке Object Pascal с исходными кодами.

Считыватели PR-X18 с версии программного обеспечения 6.x при работе с PAN номером банковских карт поддерживают и карты типа B (в частности, производства завода «МИКРОН», Зеленоград).

Функции библиотеки

Общие положения

При компиляции все функции библиотеки описаны с атрибутом (соглашением о вызовах) **cdecl**.

Все функции библиотеки возвращают код результата. При успешном выполнении функции код результата равен нулю. Значение, отличное от нуля, говорит об ошибке.

Типы параметров

Библиотека написана на языке Паскаль. Для вызова функций из программы на C/C++ следует корректно привести типы данных.

Параметры типов uint8_t, uint16_t, uint32_t комментариев не требуют. Если перед параметром стоит модификатор var, то параметр передается не по значению, а по ссылке.

Тип pbytearray соответствует указателю на массив байтов, аналог void*. Параметр типа pchar соответствует указателю на нуль — терминированную строку байтовых символов, аналог char*.

Коды ошибок

Коды ошибок определены в файле Errors.pas следующим образом:

```
// ERRORS from NXP library
PH_ERR_IO_TIMEOUT               = $01; //**< No reply received, e.g. PICC removal. */
PH_ERR_INTEGRITY_ERROR          = $02; //**< Wrong CRC or parity detected. */
PH_ERR_COLLISION_ERROR          = $03; //**< A collision occured. */
PH_ERR_BUFFER_OVERFLOW          = $04; //**< Attempt to write beyond buffer size. */
PH_ERR_FRAMING_ERROR            = $05; //**< Invalid frame format. */
PH_ERR_PROTOCOL_ERROR           = $06; //**< Received response violates protocol. */
PH_ERR_AUTH_ERROR               = $07; //**< Authentication error. */
PH_ERR_READ_WRITE_ERROR         = $08; //**< A Read or Write error occured in RAM/ROM */
PH_ERR_TEMPERATURE_ERROR        = $09; //**< The RC sensors signal overheating. */
PH_ERR_RF_ERROR                 = $0A; //**< Error on RF-Interface. */
PH_ERR_INTERFACE_ERROR          = $0B; //**< An error occured in RC communication. */
PH_ERR_LENGTH_ERROR             = $0C; //**< A length error occured. */
PH_ERR_INTERNAL_ERROR           = $7F; //**< An internal error occured. */
//
PH_ERR_INVALID_DATA_PARAMS      = $20; //**< Invalid data parameters supplied */
PH_ERR_INVALID_PARAMETER        = $21; //**< Invalid parameter supplied. */
PH_ERR_PARAMETER_OVERFLOW       = $22; //**< Read/Write a parameter produce overflow. */
PH_ERR_UNSUPPORTED_PARAMETER    = $23; //**< Parameter not supported. */
PH_ERR_UNSUPPORTED_COMMAND      = $24; //**< Command not supported. */
PH_ERR_USE_CONDITION            = $25; //**< Condition of use not satisfied. */
PH_ERR_KEY                      = $26; //**< A key error occured. */
//
PH_ERR_SUCCESS_CHAINING         = $71; //**< Rx chaining is not complete */
PH_ERR_SUCCESS_INCOMPLETE_BYTE  = $72; //**< An incomplete byte was received. */

// In the library PH_ERR_CUSTOM_BEGIN = 0x80
// and uses about 10 codes - so, we start from 0xA0
PH_ERR_CUSTOM_BEGIN             = $80;
// For Mifare defined codes
PHPAL_MIFARE_ERR_NAK0           = (PH_ERR_CUSTOM_BEGIN+0);
PHPAL_MIFARE_ERR_NAK1           = (PH_ERR_CUSTOM_BEGIN+1);
PHPAL_MIFARE_ERR_NAK4           = (PH_ERR_CUSTOM_BEGIN+2);
PHPAL_MIFARE_ERR_NAK5           = (PH_ERR_CUSTOM_BEGIN+3);
PHPAL_MIFARE_ERR_NAK6           = (PH_ERR_CUSTOM_BEGIN+4);
```

```
PHPAL_MIFARE_ERR_NAK7           = (PH_ERR_CUSTOM_BEGIN+5);
PHPAL_MIFARE_ERR_NAK8           = (PH_ERR_CUSTOM_BEGIN+6);
PHPAL_MIFARE_ERR_NAK9           = (PH_ERR_CUSTOM_BEGIN+7);
// For Mifare Plus defined codes
PHAL_MFP_ERR_AUTH               = (PH_ERR_CUSTOM_BEGIN+0); /*< MFP Authentication Error. */
PHAL_MFP_ERR_CMD_OVERFLOW       = (PH_ERR_CUSTOM_BEGIN+1); /*< MFP Command Overflow Err. */
PHAL_MFP_ERR_MAC_PCD            = (PH_ERR_CUSTOM_BEGIN+2); /*< MFP MAC Error. */
PHAL_MFP_ERR_BNR                = (PH_ERR_CUSTOM_BEGIN+3); /*< MFP Blocknumber Error. */
PHAL_MFP_ERR_EXT                = (PH_ERR_CUSTOM_BEGIN+4); /*< MFP Extension Error. */
PHAL_MFP_ERR_CMD_INVALID        = (PH_ERR_CUSTOM_BEGIN+5); /*< MFP Invalid Command Error. */
PHAL_MFP_ERR_FORMAT             = (PH_ERR_CUSTOM_BEGIN+6); /*< MFP Authentication Error. */
PHAL_MFP_ERR_GEN_FAILURE        = (PH_ERR_CUSTOM_BEGIN+7); /*< MFP Generic Error. */

// Internal communication errors
ERR_COMM_TIMEOUT      = $A0;
ERR_BAD_PACKET_CRC    = $A1;
ERR_OPEN_PORT         = $A2;
ERR_WRONG_COMPORT     = $A3;
ERR_COM_CLOSED        = $A4;
ERR_COM_WRITEERR      = $A5;
ERR_INVALID_HANDLE    = $A6;
ERR_RESULT_EMPTY      = $A7;

// error code answers from reader
ERR_INVALID_DATA      = $A8;
ERR_INVALID_COMMAND   = $A9;
ERR_INVALID_PARAM     = $AA;
```

Отличие от старых версий

Основными отличиями от версии для COM-порта являются функции обнаружения и открытия интерфейса считывателя. Если в старой версии использовалось всего две функции:

- OpenPort()
- ClosePort()

то в данной версии эти функции удалены за ненадобностью, но добавлены следующие функции:

- Enumerate() получает количество подключенных считывателей
- GetFirstReader() получает первый считыватель из списка
- GetNextReader() получает следующий считыватель из списка
- ReaderOpened() проверяет, открыт ли интерфейс считывателя
- GetReaderSerial() получает серийный номер во внутреннем формате
- GetReaderSerialStr() получает серийный номер в виде строки
- OpenByNum() открывает считыватель по номеру в листе
- CloseByNum() закрывает считыватель по номеру в листе

Следует иметь в виду, что одновременно поддерживается работа только с одним считывателем. При необходимости работы с другим считывателем текущий требуется предварительно закрыть.

В следующем разделе все функции библиотеки рассмотрены более подробно.

Функции общего назначения

Получение списка считывателей

Функция сканирует USB порты ПК для обнаружения всех подключенных считывателей PR-P18, помещает их во внутренний список и возвращает количество обнаруженных считывателей.

```
function Enumerate(): uint16;
```

Параметры

Входных параметров не имеет.

Результат

Возвращает число обнаруженных считывателей. Нумерация внутреннего списка начинается с нулевого индекса.

Получение первого считывателя из списка

Функция возвращает описатель первого считывателя из внутреннего списка вместе с его серийным номером.

```
function GetFirstReader(desc: pbytearray; maxlen: short): uint16;
```

Параметры

На входе получает указатель на строку символов и ее максимальную длину. Рекомендуется выделять не менее 48 байт (включая завершающий ноль).

Результат

Помещает описатель считывателя с серийным номером в переменную . Если выделено недостаточно места (maxlen менее требуемого значения), то возвращаемое значение усекается. Пример возвращаемой строки:

Parsec Desktop Reader PR-P18 (810-00-0001)

Если список считывателей пуст (нет подключенных считывателей, либо предварительно не вызывалась функция Enumerate), то возвращается код ошибки ERR_RESULT_EMPTY.

Получение следующего считывателя из списка

Функция возвращает описатель очередного (начиная с первого) считывателя из внутреннего списка вместе с его серийным номером.

```
function GetNextReader(desc: pbytearray; maxlen: short): uint16;
```

Параметры

На входе получает указатель на строку символов и ее максимальную длину. Рекомендуется выделять не менее 48 байт (включая завершающий ноль).

Результат

Помещает описатель считывателя с серийным номером в переменную . Если выделено недостаточно места (maxlen менее требуемого значения), то возвращаемое значение усекается.

Если список считывателей пуст (нет подключенных считывателей, либо предварительно не вызывалась функция Enumerate), либо следующего считывателя в списке нет, то возвращается код ошибки ERR_RESULT_EMPTY.

Проверка интерфейса считывателя

Сервисная функция, позволяющая получить состояние считывателя, заданного по номеру во внутреннем списке.

```
function ReaderOpened(num: short): uint16;
```

Параметры

На входе получает номер считывателя num.

Результат

Возвращает 0, если интерфейс считывателя не открывался, либо единицу, если интерфейс открыт для работы.

Ноль также возвращается, если список пуст, либо задан номер, превыщающий количество считывателей в списке.

Получение серийного номера

Служебная функция, возвращающая серийный номер заданного считывателя во внутреннем представлении.

```
function GetReaderSerial(num: short; var ser: dword): uint16;
```

Параметры

На входе передается номер считывателя в списке и указатель на четырехбайтовую переменную для помещения ответа.

Результат

Возвращает серийный номер во внутреннем представлении. Если считывателя с заданным номером не существует, либо список пуст, то возвращается код ошибки ERR_RESULT_EMPTY.

Интерфейс считывателя должен быть открыт для работы.

Получение серийного номера в виде строки

Функция, возвращающая серийный номер заданного считывателя в виде NULL-терминированной строки.

```
function GetReaderSerialStr(num: short; var serstr: PAnsiChar): uint16;
```

Параметры

На входе передается номер считывателя в списке и указатель на байтовый массив для помещения ответа.

Результат

Возвращает серийный номер в виде строки символов, завершаемой нулевым байтом. Если считывателя с заданным номером не существует, либо список пуст, то возвращается код ошибки ERR_RESULT_EMPTY.

Интерфейс считывателя должен быть открыт для работы. Размер массива для помещения результата должен быть не менее 48 байтов.

Открытие заданного считывателя

Открывает считыватель для выполнения дальнейших операций с ним.

```
function OpenByNum(num: short): uint16;
```

Параметры

Получает номер считывателя во внутреннем списке.

Результат

Если интерфейс считывателя уже открыт, то возвращается код ошибки ERR_OPEN_PORT. Если считывателя с указанным номером в списке нет, то возвращается код ошибки ERR_RESULT_EMPTY. В случае возникновения внутренней ошибки при открытии интерфейса возвращается код ошибки ERR_INVALID_HANDLE.

Закрытие заданного считывателя

Закрывает интерфейс считывателя, после чего его можно открыть снова, либо открыть другой считыватель из списка (если подключено более одного считывателя к одному ПК).

```
function CloseByNum(num: short): uint16;
```

Параметры

Получает номер считывателя во внутреннем списке.

Результат

Закрывает интерфейс считывателя. Если считывателя с указанным номером в списке нет, то возвращается код ошибки ERR_RESULT_EMPTY

Пересчет номера блока в номер сектора

Пересчитывает номер блока в соответствующий номер сектора с учетом разницы размеров секторов для карт с объемом памяти 4 килобайта.

```
function BlockToSector4k(block: uint16): uint16
```

Параметры

Принимает номер блока карты Mifare.

Результат

Возвращает номер сектора, в котором расположен заданный блок.

Пересчет номера сектора в номер блока

Возвращает номер первого (нулевого) блока заданного сектора.

```
function SectorToBlock4k(sect: uint16): uint16
```

Параметры

Принимает номер сектора карты Mifare.

Результат

Возвращает номер нулевого блока заданного сектора.

Программная перезагрузка

Вызов функции приводит к программной перезагрузке считывателя. Параметров функция не имеет.

```
function ReaderSoftReset(): uint16
```

При наличии связи со считывателем всегда возвращает успешный результат.

Проверка связи

Функция служит для проверки обмена со считывателем в режиме эха.

```
function TestTranceive(var len: uint16; data: PByteArray): uint16
```

Параметры

len	при вызове — число байтов для обмена, при возврате — длина ответа.
data	указатель на буфер с данными для обмена.

Индикация

Прграммно включает бипер и/или светодиоды считывателя на заданное время.

```
function BeepBlink(led: uint8; beep: uint8; time: uint8): uint16
```

Параметры

led	при нулевом значении светодиод не включается, иначе включается.
beep	при нулевом значении бипер не включается, иначе включается.
time	время в квантах по 100 мсек, на которое включается индикация.

Следует иметь в виду, что время работы функции пропорционально заданному времени индикации.

Чтение регистра

Низкоуровневая функция чтения регистра микросхемы CLRC663.

```
function ReadRegister(addr: uint16; var data: uint16): uint16
```

Параметры

addr	адрес регистра микросхемы CLRC663.
data	данные, прочитанные из регистра микросхемы CLRC663.

Запись регистра

Низкоуровневая функция записи регистра микросхемы CLRC663.

```
function WriteRegister(addr, data: uint16): uint16
```

Параметры

addr	адрес регистра микросхемы CLRC663.
data	данные, записываемые в регистр микросхемы CLRC663.

Функцией следует пользоваться с осторожностью, поскольку некорректная запись в регистры может нарушить работу микросхемы и даже полностью вывести ее из строя.

Управление несущей считывателя

Функция предназначена для управления несущей считывателя.

```
function RfFieldControl(control: uint8): uint16
```

Параметры

control	значение 0 выключает несущую, значение 1 включает несущую, а при значении 2 производится пересброс несущей: выключение примерно на 10 мсек с последующим включением.

Получение наименования и версии

Функция позволяет получить наименование считывателя и его версию в виде строки.

```
function GetReaderId(data: pbytearray): uint16;
```

Параметры

data	указатель на приемный массив для возвращения строкового результата.

Работа с картами на уровне 3 и 4 ISO-14443

Активирование карты до уровня 3

Функция пытается активировать карту, выполняя основные операции — request, anticollision, select.

```
function ActivateCardA3(var uidlen: uint8; uid: pbytearray;
                        var sak: uint8; var mca: uint8): uint16
```

Параметры

uidlen	длина UID карты (4 или 7 байтов).
uid	указатель на буфер для UID карты
sak	значение параметра SAK
mca	0 — карт в поле больше нет, другие значения — в поле имеются другие карты.

При вызове функции значение uidlen должно устанавливаться в ноль.

Результат

Если функция возвращает не нулевое значение, значит карты в поле нет или карту прочитать невозможно по какой-то причине.

Активирование карты до уровня 3 расширенная

Добавлена с версии 1.1.2 библиотеки. Функция пытается активировать карту, выполняя основные операции — request, anticollision, select.

```
function ActivateCardA3Ex(var uidlen: uint8; uid: pbytearray;
                 var atqa: uint16; var sak: uint8; var mca: uint8): uint16
```

Параметры

uidlen	длина UID карты (4 или 7 байтов).
uid	указатель на буфер для UID карты
atqa	значение параметра ATQA
sak	значение параметра SAK
mca	0 — карт в поле больше нет, другие значения — в поле имеются другие карты.

Результат

Если функция возвращает не нулевое значение, значит карты в поле нет или карту прочитать невозможно по какой-то причине.

При вызове функции значение uidlen должно устанавливаться в ноль.

Активирование карты до уровня 4

Функция пытается активировать карту до уровня 4, подготовив ее для дальнейшей работы.

```
function ActivateCardA4(usepps: uint8; var uidlen: uint8; uid: pbytearray;
                        var sak: uint8; var mca: uint8;
                        // Level 4 ISO params
                        var ats: uint8; var ciden: uint8; var cid: uint8;
                        var fsdi: uint8; var fsci: uint8): uint16
```

Параметры

usepps	считыватель попытается выполнить команду PPS и подготовиться к обмену с максимальными параметрами.
uidlen	длина UID карты (4 или 7 байтов).
uid	указатель на буфер для UID карты
sak	значение параметра SAK
mca	0 — карт в поле больше нет, другие значения — в поле имеются другие карты.
ats	длина ответа карты ATS.
ciden	если не ноль — карта поддерживает CID
cid	значение CID, если предыдущий параметр не ноль
fsdi, fsci	множители, характеризующие максимальный размер фрейма карты на прием и передачу.

При вызове функции значение uidlen должно устанавливаться в ноль.

Активирование карты до уровня 4 расширенная

Добавлена с версии 1.1.2 библиотеки. Функция пытается активировать карту до уровня 4, подготовив ее для дальнейшей работы.

```
function ActivateCardA4Ex(usepps: uint8; var uidlen: uint8; uid:
                  pbytearray; var atqa: uint16; var sak: uint8; var mca:
                  uint8;
                  // Level 4 ISO params
                  var ats: uint8; var ciden: uint8; var cid: uint8;
                  var fsdi: uint8; var fsci: uint8): uint16
```

Параметры

usepps	считыватель попытается выполнить команду PPS и подготовиться к обмену с максимальными параметрами.
uidlen	длина UID карты (4 или 7 байтов).
uid	указатель на буфер для UID карты
atqa	значение параметра ATQA
sak	значение параметра SAK

mca	0 — карт в поле больше нет, другие значения — в поле имеются другие карты.
ats	длина ответа карты ATS.
ciden	если не ноль — карта поддерживает CID
cid	значение CID, если предыдущий параметр не ноль
fsdi, fsci	множители, характеризующие максимальный размер фрейма карты на прием и передачу.

При вызове функции значение uidlen должно устанавливаться в ноль.

Команда HaltA

Команда предназначена для перевода активированной в данный момент до уровня 3 ISO-14443A карты в состояние HALT.

```
function HaltA(): uint16
```

Команда параметров не имеет, возвращает всегда код успешного завершения.

Прозрачный обмен на уровне 3 ISO-14443A

Команда «прозрачного» обмена для карт уровня 3 ISO-14443A. Позволяет реализовать любую команду, не реализованную в библиотеке. Передаваемые в команду данные напрямую передаются карте.

Возвращенные картой данные функция возвращает вызвавшей программе.

```
function Exchange3A(InData: pbytearray; inlen: uint16; OutData: pbytearray;
                    var outlen: uint16): uint16
```

Параметры

InData	указатель на буфер с командой и/или данными, которые требуется передать карте
inlen	длина данных в буфере.
OutData	указатель на буфер для приема данных от карты
outlen	размер возвращенных в буфере данных

Результат

Возвращается код ошибки в случае не успешного обмена, либо если размер передаваемых данных нулевой или превышает 256 байт.

Обмен с картой на уровне 4 ISO-14443A

Команда обмена информацией с картой на уровне 4 ISO-14443A. Позволяет обменяться с картой APDU в соответствии с ISO-7816.

Возвращенные картой данные функция возвращает вызвавшей программе.

```
function ExchangeApdu(InData: pbytearray; inlen: uint16; OutData:
                      pbytearray; var outlen: uint16): uint16
```

Параметры

InData	указатель на буфер с командой и/или данными, которые требуется передать карте
inlen	длина данных в буфере.
OutData	указатель на буфер для приема данных от карты
outlen	размер возвращенных в буфере данных

Результат

Возвращается код ошибки в случае не успешного обмена, либо если размер передаваемых данных нулевой или превышает 256 байт.

Если карту вернула код ошибки операции, отличный от 0x9000, то длина возвращенных данных будет равна двум байтам, в которых будет содержаться указанный код завершения операции обмена.

Функции поддержки Mifare Classic

Аутентификация с заданным ключом

Функция предназначена для аутентификации доступа к заданному блоку карты Mifare Classic.

Описание

```
function MfcAuthenticateKey(block: uint16; key_type: uint8; pUid:
pbytearray;pKey: pbytearray): uint16;
```

Параметры

block	абсолютный номер блока карты.
key_type	тип ключа: 0x0A — ключ A, 0x0B — ключ B
pUid	указатель на 4 байта UID карты для аутентификации
pKey	указатель на 6 байтов значения ключа доступа

Возвращаемые значения

Функция возвращает код ошибки, ноль при успешном выполнении операции.

Аутентификация с ключом из EEPROM

Микросхема считывателя содержит специальную область EEPROM, доступную только для записи и предназначенную для хранения ключей доступа к карте. Ключи заносятся с помощью функции, описание которой следует за описанием данной функции.

Описание

```
function MfcAuthentKeyNo(block: uint16; key_type: uint8; pUid: pbytearray;
 key_no: uint16; key_version: uint16): uint16;
```

Параметры

block	абсолютный номер блока карты.
key_type	тип ключа: 0x0A — ключ A, 0x0B — ключ B
pUid	указатель на 4 байта UID карты для аутентификации
key_no	номер ключа в EEPROM, начиная с нуля
key_version	версия ключа, должна быть равна нулю

Возвращаемые значения

Функция возвращает код ошибки, ноль при успешном выполнении операции.

Запись ключа в EEPROM

С помощью данной функции в EEPROM микросхемы считывателя записывается передаваемый в качестве параметра ключ. Записанный ключ используется предыдущей функцией.

При записи ключей используется следующее соглашение:

- Ключ 00 - key A sector 0
- Ключ 01 - key B sector 0
- Ключ 02 - key A sector 1
- Ключ 03 - key B sector 1
- … и так далее

За один раз может быть записано более одного ключа (например, пара для конкретного сектора).

Описание

```
function MfcStoreKeyE2(key_no: uint8; num_keys: uint8; keys: pbytearray): uint16;
```

Параметры

key_no	номер ключа в EEPROM, начиная с нуля
num_keys	количество передаваемых функции ключей
key_type	тип ключа: 0x0A — ключ A, 0x0B — ключ B
keys	указатель на массив ключей для записи

Возвращаемые значения

Функция возвращает код ошибки, ноль при успешном выполнении операции.

Чтение блока Mifare Classic

Функция читает заданный блок карты Mifare Classic. Предварительно должна быть произведена аутентификация для доступа к заданному блоку.

Допускается последовательное чтение нескольких блоков в рамках текущего сектора карты.

Описание

```
function MfcReadBlock(block: uint16; pBlockData: pbytearray): uint16;
```

Параметры

block	абсолютный номер блока карты.
pBlockData	указатель на область памяти, в которую считывается блок

Возвращаемые значения

Функция возвращает код ошибки, ноль и 16 байтов данных при успешном выполнении операции.

Запись блока Mifare Classic

Функция записывает данные в указанный блок карты Mifare Classic. Предварительно должна быть произведена аутентификация для доступа к заданному блоку.

Допускается последовательная запись нескольких блоков в рамках текущего сектора карты.

Описание

```
function MfcWriteBlock(block: uint16; pBlockData: pbytearray): uint16;
```

Параметры

block	абсолютный номер блока карты.
pBlockData	указатель на область памяти, из которой данные записываются в блок

Возвращаемые значения

Функция возвращает код ошибки, ноль при успешном выполнении операции.

Чтение сектора Mifare Classic

Функция предназначена для чтения полного блока карты Mifare Classic. Для карт объемом памяти 4К последние 8 секторов имеют размер данных 240 байт (против 48 байт для остальных секторов).

Описание

```
function MfcReadSector(sector: uint16; var size: uint16; data: pbytearray):
uint16;
```

Параметры

sector	абсолютный номер сектора карты
size	размер прочитанных данных (48 или 240 байт в зависимости от номера сектора)
data	указатель на область памяти, в которую читаются данные сектора

Возвращаемые значения

Функция возвращает код ошибки, ноль и прочитанные из сектора данные при успешном выполнении операции.

Запись сектора Mifare Classic

Функция предназначена для записи полного блока карты Mifare Classic. Для карт объемом памяти 4К последние 8 секторов имеют размер данных 240 байт (против 48 байт для остальных секторов).

Описание

```
function MfcWriteSector(sector: uint16; data: pbytearray): uint16;
```

Параметры

sector	абсолютный номер сектора карты
data	указатель на область памяти, из которой данные записываются в сектор

Возвращаемые значения

Функция возвращает код ошибки, ноль при успешном выполнении операции.

Функции поддержки Mifare Plus

Персонализация карт Mifare Plus

Команда предназначена для записи любого блока карты Mifare Plus в процессе персонализации. Работает только с картой уровня SL0, после перехода на любой другой уровень команда выполнена быть уже не может.

```
function WritePerso(block: uint16; key: pbytearray): uint16
```

Параметры

block	абсолютный номер блока карты.
key	16 байтов данных для записи в указанный блок.

С помощью команды можно записывать данные как в область ключей, так и в блоки данных.

Возвращаемые значения

Функция возвращает код ошибки, ноль при успешном выполнении операции.

Закрыть персонализацию

Команда закрывает персонализацию карты с переводом ее на следующий уровень безопасности (после пересброса поля и последующей активации карты).

Персонализацию можно проводить в несколько приемов, между ними удаляя карту из поля считывателя. До получения данной команды карта ведет себя как обыкновенная память EEPROM (с учетом секторной структуры) с возможностью многократной перезаписи данных.

```
function CommitPerso(): uint16
```

Параметров команда не имеет.

Возвращаемые значения

Функция возвращает код ошибки, ноль при успешном выполнении операции.

Аутентификация на уровне SL3

Команда используется для аутентификации доступа к карте на уровне безопасности SL3. Если до этого карта была на более низком уровне безопасности, то она будет переведена на уровень SL3 без возможности понизить его обратно.

```
function MifPlusAuthSl3(first: uint8; keyblock: uint16;
                        keyind: uint16): uint16
```

Параметры

first	при ненулевом значении трактуется как первичная аутентификация.
keyblock	номер блока, к которому требуется доступ с ключом AES128

keyind	индекс ключа во внутренней EEPROM CLRC663 (ключи должны быть занесены заранее).

Возвращаемые значения

Функция возвращает код ошибки, ноль при успешном выполнении операции.

Аутентификация на уровне SL3 с заданным ключом

Как и предыдущая, команда используется для аутентификации доступа к карте на уровне безопасности SL3. Если до этого карта была на более низком уровне безопасности, то она будет переведена на уровень SL3 без возможности понизить его обратно. Отличие в том, что ключ доступа задается непосредственно при вызове функции.

```
function MifPlusAuthSl3Key(first: uint8; keyblock: uint16;
                           key: pbytearray): uint16
```

Параметры

first	при ненулевом значении трактуется как первичная аутентификация.
keyblock	номер блока, к которому требуется доступ с ключом AES128.
key	значение ключа AES128 (16 байтов).

Возвращаемые значения

Функция возвращает код ошибки, ноль при успешном выполнении операции.

Чтение блоков карты Mifare Plus

После успешной аутентификации к сектору с помощью данной функции можно прочитать один или более блок данных из этого сектора.

```
function MifPlusReadBlocks(encripted: uint8; readmaced: uint8;
                           maconcmd: uint8; block: uint16;
                           blockcount: uint8; data: pbytearray): uint16
```

Параметры

encripted	0 — без шифрования, 1 — с шифрованием обмена.
readmaced	0 – чтение без MAC, 1 – с MAC.
maconcmd	0 – без MAC в команде, 1 — с MAC.
block	номер блока для чтения (в абсолютных значениях).
blockcount	число блоков для чтения (от 1 до 4-х)
data	указатель на буфер для прочитанных данных.

Возвращаемые значения

Функция возвращает код ошибки, ноль при успешном выполнении операции.

Запись блоков карты Mifare Plus

После успешной аутентификации к сектору с помощью данной функции можно записать один или более блок данных в этот сектор.

```
function MifPlusWriteBlocks(encripted: uint8; maconresp: uint8;
                            block: uint16; blockcount: uint8ж
                            data: pbytearray): uint16
```

Параметры

encripted	0 — без шифрования, 1 — с шифрованием обмена.
maconresp	0 – ответ без MAC, 1 – с MAC.
block	номер блока для чтения (в абсолютных значениях).
blockcount	число блоков для чтения (от 1 до 4-х)
data	указатель на буфер для записываемых данных.

Возвращаемые значения

Функция возвращает код ошибки, ноль при успешном выполнении операции.

Чтение сектора Mifare Plus

Назначение

Функция предназначена для чтения данных сектора карты Mifare Plus. Читается только область данных, без сектора трайлера.

Функция сама определяет размер данных (число блоков) для чтения в зависимости от номера сектора: для секторов от 0 до 31 читется 48 байт данных, для секторов от 32 до 39 читается 240 байтов данных.

Описание

```
function MifPlusReadSector(encripted: uint8; readmaced: uint8; maconcmd:
      uint8; sector: uint16; var size: uint16; data: pbytearray): uint16
```

Параметры

encripted	0 — без шифрования, 1 — с шифрованием обмена.
readmaced	0 – чтение без MAC, 1 – с MAC.
maconcmd	0 – без MAC в команде, 1 — с MAC.
sector	номер сектора для чтения (от 0 до 39).
size	возвращает количество прочитанных байтов
data	указатель на буфер для прочитанных данных.

Возвращаемые значения

Функция возвращает код ошибки, ноль при успешном выполнении операции.

Запись сектора Mifare Plus

Назначение

Функция предназначена для записи данных сектора карты Mifare Plus. Записывается только область данных, без сектора трайлера.

Функция сама определяет размер данных (число блоков) для записи в зависимости от номера сектора: для секторов от 1 до 31 пишется 48 байт данных, для секторов от 32 до 39 пишется 240 байтов данных.

Для нулевого сектора пишется только 32 байта, первые 16 байтов данных игнорируются, поскольку блок 0 имеет атрибут «только для чтения».

Описание

```
function MifPlusWriteSector(encripted: uint8; maconresp: uint8; sector:
                            uint16; data: pbytearray): uint16;
```

Параметры

encripted	0 — без шифрования, 1 — с шифрованием обмена.
maconresp	0 – без MAC, 1 – с MAC.
sector	номер сектора для записи (от 0 до 39).
data	указатель на буфер с данными для записи.

Возвращаемые значения

Функция возвращает код ошибки, ноль при успешном выполнении операции.

Занесение ключей в EEPROM

Ключи шифрования можно предварительно занести во внутреннюю EEPROM CLRC663 для последующего их использования в процедуре аутентификации (например, с функцией MifPlusAuthSl3).

```
function WriteAesKeyToEe(sector: uint8; keytype: uint8;
                         key: pbytearray): uint16
```

Параметры

sector	номер сектора, для которого предназначен ключ.
keytype	0 – ключ A, 1 — ключ B.
key	указатель на буфер с записываемым ключом.

Возвращаемые значения

Функция возвращает код ошибки, ноль при успешном выполнении операции.

Перезапись секторных ключей AES

Добавлена с версии 1.1.2 библиотеки. Функция предназначена для перезаписи секторных ключей AES-128 в карту, переведенную на уровень безопасности SL3.

```
function MifPlusChangeAesKey(maced: uint8; block: uint16; key: pbytearray):
uint16
```

Параметры

maced	задает режим обмена с картой. При нуле MAC не используется.
block	номер блока, в который записывается новое значение ключа.
key	указатель на буфер с записываемым ключом.

Возвращаемые значения

Функция возвращает код ошибки, ноль при успешном выполнении операции.

Замечания

Ключи хранятся в блоках с адреса 0x4000, например, в блоке 0x4000 хранится ключ A для нулевого сектора, в блоке 0x4001 хранится ключ B для нулевого сектора.

Перед вызовом функции следует активировать карту до уровня 4 и провести аутентификацию с ключом на запись для желаемого сектора, после чего можно поменять оба ключа (как на запись, так и на чтение). Как правило, на запись используется ключ B, что в приведенном выше примере подразумевает аутентификацию к блоку 0x4001.

Комплексные функции

В данном разделе рассмотрены функции, добавленные в версии 2.0 библиотеки. Для их поддержки версия по считывателя PR-P18 должна быть также не ниже 2.0.

Следует иметь в виду, что перечисленные в данном разделе функции самостоятельно включают несущую, но не выключают ее по окончании работы — это должен сделать программист, вызывавший функцию.

Получение ASN банковской карты

Команда предназначена для получения ASN специального не финансового приложения банковской карты. На карте должно быть размещено не финансовое приложение SmartLab Solutions.

```
function ReadBankAsn(var size: uint16; data: pbytearray): uint16;
```

Параметры

size	размер возвращаемого ID карты.
data	ID карты размером 8 байтов.

Возвращаемые значения

Функция возвращает код ошибки, ноль при успешном выполнении операции.

Получение ASN банковской карты с хэшированием

Команда предназначена для получения ASN банковской карты с хэшированием для обеспечения уникальности кода при использовании коротких ID карт в системе. На карте должно быть размещено не финансовое приложение SmartLab solutions.

```
function ReadBankAsnCoded(var size: uint16; data: pbytearray): uint16;
```

Параметры

size	размер возвращаемого ID карты.
data	ID карты размером 4 байта.

Возвращаемые значения

Функция возвращает код ошибки, ноль при успешном выполнении операции.

Получение ID смартфона с ОС Андроид

Команда предназначена для получения уникального идентификатора смартфона с ОС Андроид. На смартфоне должно работать приложение Parsec Card Emulator.

```
function ReadParsecSmart(var size: uint16; data: pbytearray): uint16;
```

Параметры

size	размер возвращаемого ID смартфона.
data	ID карты размером 8 байтов.

Возвращаемые значения

Функция возвращает код ошибки, ноль при успешном выполнении операции.

Получение ID карты типа B

Команда предназначена для получения уникального идентификатора карты типа ISO-14443-B. Данный тип карт не имеет постоянного UID как у карт типа A, поэтому идентификатор извлекается из специальной области карты. Работает с картами определенного производителя, используется в заказных проектах.

```
function ReadCardB(var size: uint16; data: pbytearray): uint16;
```

Параметры

size	размер возвращаемого ID карты.
data	ID карты размером 7 байтов.

Возвращаемые значения

Функция возвращает код ошибки, ноль при успешном выполнении операции.

Получение ID карты типа B с хэшированием

Команда предназначена для получения уникального идентификатора карты типа ISO-14443-B. Данный тип карт не имеет постоянного UID как у карт типа A, поэтому идентификатор извлекается из специальной области карты. Работает с картами определенного производителя, используется в заказных проектах. Хэширование используется для обеспечения уникальности ID, когда его длина меньше оригинального значения.

```
function ReadCardBCoded(var size: uint16; data: pbytearray): uint16;
```

Параметры

size	размер возвращаемого ID карты.
data	ID карты размером 4 байта.

Возвращаемые значения

Функция возвращает код ошибки, ноль при успешном выполнении операции.

Получение ID карты I-Code SLI

Команда предназначена для получения уникального идентификатора карты типа I-Code SLI (ISO-15693).

```
function ReadCardICode(var size: uint16; data: pbytearray): uint16;
```

Параметры

size	размер возвращаемого ID карты.
data	ID карты размером 8 байтов.

Возвращаемые значения

Функция возвращает код ошибки, ноль при успешном выполнении операции.

Получение PAN номера банковской карты

Команда предназначена для получения в качестве уникального идентификатора (ID) части PAN номера банковской карты, в том числе с виртуальных карт на смартфонах.

Данная функция читает часть PAN номера карты с с частичным преобразованием, при котором выдаваемый функцией код не совпадает со значением, обозначенном на карте.

По данному алгоритму работают считыватели серии PNR-P/X.

```
function ReadBankCardPan(var size: uint16; data: pbytearray): uint16;
```

Параметры

size	размер возвращаемого ID карты.
data	ID карты размером 5 байт.

Возвращаемые значения

Функция возвращает код ошибки, ноль при успешном выполнении операции.

Получение полного PAN номера карты

Команда служит для получения полного PAN номера с реальной или виртуальной банковской карты в виде строки. Полученная строка повторяет изображенный на карте номер и позволяет на стороне хоста осуществить необходимые дополнительные преобразования (например, хэширование, поскольку использование реального PAN номера в приложениях считается недопустимым).

```
function ReadCardPanFull(var size: uint16; data: pbytearray): uint16;
```

Параметры

size	размер возвращаемого ID карты.
data	ID карты в виде текста размером 16 байт.

Возвращаемые значения

Функция возвращает код ошибки, ноль при успешном выполнении операции.

Функции, специфичные для PR-X18

Функции из данного раздела поддерживаются только в модели PR-P18X, имеющей низкочастотный радио тракт.

Чтение карты формата EM Marin

Функция возвращает полный (5 байт) код карты формата EM Marin.

```
function ReadCardEM(var size: uint16; data: pbytearray): uint16;
```

Параметры

size	размер возвращаемого ID карты.
data	ID карты размером 5 байт.

Возвращаемые значения

Функция возвращает код ошибки, ноль при успешном выполнении операции.

Чтение карты формата HID

Функция возвращает полный (6 байт) код карты формата HID. Дальнейшая трактовка формата данных на карте предоставляется пользователю.

```
function ReadCardHid(var size: uint16; data: pbytearray): uint16
```

Параметры

size	размер возвращаемого ID карты.
data	ID карты размером 6 байт.

Возвращаемые значения

Функция возвращает код ошибки, ноль при успешном выполнении операции.

Чтение карты формата CheckPoint

Функция возвращает полный (4 байта) код карты формата CheckPoint, включая контрольную сумму

```
function ReadCardCP(var size: uint16; data: pbytearray): uint16
```

Параметры

size	размер возвращаемого ID карты.
data	ID карты размером 4 байта.

Возвращаемые значения

Функция возвращает код ошибки, ноль при успешном выполнении операции.

Приложение 1

Распределение секторов карты

Рекомендуемое распределение секторов первого килобайта памяти карты Mifare Plus в системе доступа с возможностью дополнения других приложений показано в следующей таблице.

Таблица 2:

Сектор	Назначение
0	MAD
1	Данные персонализирующей организации
2	Данные держателя карты
3	CID — идентификатор карты в СКУД
4	PACS — данные для ССКУД
5	резерв
6	резерв
7	свободен
8	свободен
9	свободен
10	свободен
11	свободен
12	свободен
13	свободен
14	свободен
15	свободен

Таким образом, мы оставляем 9 свободных секторов для распределения между другими приложениями.

Приложение 2

Содержимое MAD

С учетом приведенного выше распределения секторов MAD будет иметь следующее содержимое.

Таблица 3:

Байт блок	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
0	Блок производителя, включая UID															
1	AID сектора 0x07		AID сектора 0x06		AID сектора 0x05		AID сектора 0x04		AID сектора 0x03		AID сектора 0x02		AID сектора 0x01		info	CRC
	0x0000		0x0002		0x0002		0x4801		0x4701		0x0004		0x0003		0x01	??
2	AID сектора 0x0F		AID сектора 0x0E		AID сектора 0x0D		AID сектора 0x0C		AID сектора 0x0B		AID сектора 0x0A		AID сектора 0x09		AID сектора 0x08	
	0x0000		0x0000		0x0000		0x0000		0x0000		0x0000		0x0000		0x0000	
3	Сектор трайлер (ключи и условия доступа)															

По спецификации MAD () за системами контроля доступа зарезервированы коды кластера 0x47, 0x48. Коды приложения определяет, как правило, разработчик.

Для заметок