***Программный интерфейс ядра L1 для устройства aQsi Cube***

1. **Работа с бесконтактными картами**
	1. **Документация**
	* ISO/IEC 14443-4:2008(E)
	* EMV® Level 1 Specifications for Payment Systems: EMV Contactless Interface Specification Version 3.0
	1. **Библиотека L1**

Библиотека L1 (далее L1) реализовано внутри библиотеки "libaqc\_emv\_ctls\_l1.so" Библиотека поставляется вместе с заголовочным файлом "contactless.h"

* 1. **Определены следующие коды возврата:**
* EMVL1\_STATUS\_OK = 0 - Ошибок нет
* EMVL1\_STATUS\_TIMEOUT = -1 - таймаут при обмене с картой
* EMVL1\_STATUS\_NOT\_INITIALIZED = -99 - Не инициализирована библиотека или один из ее компонентов
* EMVL1\_STATUS\_FAILED = -8 - Операция не выполнена
* EMVL1\_STATUS\_NOT\_PRESENT = -9 - Карта не оsec-serv
* EMVL1\_STATUS\_COLLISION\_ERROR = -13 - Коллизия, в поле более одной карты
* EMVL1\_STATUS\_PROTOCOL\_ERROR = -14 - Нарушен протокол передачи, тип карты не соответствует требуемой
* EMVL1\_STATUS\_TRANSMISSION\_ERROR = -15 - Ошибка передачи данных
	1. **Функции для работ с картами EMVco:**
		1. *uint32\_t contactless\_getversion();*

Возвращает версию бесконтактной библиотеки L1.

Старшее слово – номер сборки. В младшем слове старший байт – старший номер версии, младший байт – младший номер версии.

В случае ошибки возвращает 0.

Перед вызовом библиотека должна быть открыта.

* + 1. *int contactless\_open ();*

Инициализирует контактный интерфейс.

Возвращает EMVL1\_STATUS\_OK в случае успеха или ненулевой код ошибки.

* + 1. *int contactless\_close ();*

 Завершает работу с контактным интерфейсом.

Возвращает EMVL1\_STATUS\_OK в случае успеха или ненулевой код ошибки.

* + 1. *int contactless\_probe ();*

Проверяет наличие карты EMV в поле ридера, выполняя однократно поллинг карты.

Возвращает EMVL1\_STATUS\_OK, если карта обнаружена, EMVL1\_STATUS\_NOT\_PRESENT, если карта не обнаружена, EMVL1\_STATUS\_COLLISION\_ERROR в случае если в поле обнаружено несколько карт или ненулевой код ошибки.

Если поле выключено, включает поле.

Если функция вернула любую ошибку, отличную от EMVL1\_STATUS\_NOT\_PRESENT, необходимо вызвать contactless\_stop() и только после этого продолжать поллинг карты.

* + 1. *int contactless\_poweron ();*

Включает поле ридера.

Возвращает EMVL1\_STATUS\_OK в случае успеха или ненулевой код ошибки.

Вызывать не обязательно, сохранено для совместимости.

* + 1. *int contactless\_poweroff ();*

Выключает поле ридера.

Возвращает EMVL1\_STATUS\_OK в случае успеха или ненулевой код ошибки.

* + 1. *int contactless\_apdu (const uint8\_t\* req, uint32\_t qsz, uint8\_t\* rsp, uint32\_t\* psz);*

Отправляет карте запрос APDU и возвращает ответ.

req указывает на буфер содержащий запрос. qsz задает размер запроса. rsp указывает на заранее выделенный буфер для ответа. В psz возвращается размер ответа.

Вызывающий должен предоставить буфер, достаточный для сохранения ответа.

Возвращает EMVL1\_STATUS\_TIMEOUT в случае успеха, EMVL1\_STATUS\_FAILED в случае таймаута, EMVL1\_STATUS\_TRANSMISSION\_ERROR в случае ошибки передачи данных, EMVL1\_STATUS\_PROTOCOL\_ERROR в случае ошибки в формате запроса или ответа, или EMVL1\_STATUS\_FAILED в случае другой ошибки.

* + 1. *int contactless\_start ();*

 Аналогично contactless\_poweron(). Вызывать не обязательно, сохранено для совместимости.

* + 1. int contactless\_stop (uint32\_t timeout);\*\*

Выполняет процедуру извлечения карты из поля и выключает поле. Ожидает извлечения заданное в мс время.

Возвращает EMVL1\_STATUS\_OK в случае успеха или ненулевой код ошибки.

* 1. **Функции для индикации:**
		1. *int contactless\_beep(bool ok);*

Подает звуковой сигнал успеха или ошибки операции.

Возвращает EMVL1\_STATUS\_OK в случае успеха или ненулевой код ошибки.

* + 1. *int contactless\_setled(int mode);*

Управляет светодиодной индикацией согласно спецификации EMVco:

* EMVL1\_LEDS\_STATE\_NOT\_READY - ридер не готов
* EMVL1\_LEDS\_STATE\_IDLE - ридер в режиме ожидания
* EMVL1\_LEDS\_STATE\_READY\_TO\_READ - поле включено, ридер в режиме обнаружения карты
* EMVL1\_LEDS\_STATE\_PROCESSING - карта обнаружена, идет обмен
* EMVL1\_LEDS\_STATE\_CR\_SUCCESSFULLY - карта успешно обработана
* EMVL1\_LEDS\_STATE\_ERROR - ошибка в работе с картой

Возвращает EMVL1\_STATUS\_OK в случае успеха или ненулевой код ошибки.

* 1. **Функции для работы с картами Mifare:**
		1. *int contactless\_mifare\_probe(uint8\_t \*output)*

Проверяет наличие карты Mifare в поле ридера, выполняя однократно поллинг карты.

Коды возврата аналогично *contactless\_probe ();*

Если поле выключено, включает поле.

Если функция вернула любую ошибку, отличную от EMVL1\_STATUS\_NOT\_PRESENT, необходимо вызвать contactless\_stop() и только после этого продолжать поллинг карты.

output указывает на заранее выделенный буфер, в который возвращаются следующие данные:

* дилна UID, 1 байт;
* UID, 4,7 или 10 байт;
* ATQA, 2 байта;
* SAK, 1 байт;
	+ 1. *int contactless\_mifare\_auth(uint8\_t block\_no, char key\_type, uint16\_t key\_number, uint16\_t key\_version, uint8\_t\* uid, uint8\_t uid\_length);*

Выполняет аутентификацию в карте на ключе из SAM AV2.

Вызывающий должен пройти авторизацию к SAM AV2, смотри contactless\_mifare\_sam\_enable();

Входные параметры

|  |  |
| --- | --- |
| block\_no | Номер блока в карте, к которому осуществляется аутентификация |
| key\_type | Тип ключа. Допустимые значения A a B b |
| key\_number | Номер ключа в хранилище |
| key\_version | Версия ключа в хранилище |
| uid | Указатель на буфер, содержащий UID |
| uid\_length | Длина UID |

Возвращает EMVL1\_STATUS\_OK в случае успеха или ненулевой код ошибки.

* + 1. *int contactless\_mifare\_auth\_key(uint8\_t block\_no, char key\_type, uint8\_t\* key, uint8\_t\* uid, uint8\_t uid\_length);*

Выполняет аутентификацию в карте на известном ключе.

Входные параметры

|  |  |
| --- | --- |
| block\_no | Номер блока в карте, к которому осуществляется аутентификация |
| key\_type | Тип ключа. Допустимые значения A a B b |
| key | Указатель на буфер, содержащий ключ |
| uid | Указатель на буфер, содержащий UID |
| uid\_length | Длина UID |

Возвращает EMVL1\_STATUS\_OK в случае успеха или ненулевой код ошибки.

* + 1. *int contactless\_mifare\_reprobe (uint8\_t\* output)*

Переактивация ранее активированной карты майфир в случае неуспешной авторизации. Функция аналогична contactless\_mifare\_probe, но должна вызываться ТОЛЬКО в случае ошибки аутентификации (т.е. только после неуспешной *contactless\_mifare\_auth\_key, 1.6.3.* В других случаях её вызывать не нужно!)

* + 1. *int contactless\_mifare\_read\_block(uint8\_t block\_no, uint8\_t\* block\_data);*

Выполняет чтение блока карты.

Входные параметры

|  |  |
| --- | --- |
| block\_no | Номер блока в карте |
| *block\_data* | Указатель на буфер, куда будут помещены прочитанные данные |

Возвращает EMVL1\_STATUS\_OK в случае успеха или ненулевой код ошибки.

* + 1. *int contactless\_mifare\_write\_block(uint8\_t block\_no, uint8\_t\* block\_data);*

Выполняет запись блока карты.

Входные параметры

|  |  |
| --- | --- |
| block\_no | Номер блока в карте |
| *block\_data* | Указатель на буфер, содержащий данные для записи в карту |

Возвращает EMVL1\_STATUS\_OK в случае успеха или ненулевой код ошибки.

* + 1. *int contactless\_mifare\_read\_value(uint8\_t block\_no, uint32\_t\* value, uint8\_t\* addr);*

Выполняет чтение value-блока карты. Формат блока должен соответствовать формату value-блока.

Входные параметры

|  |  |
| --- | --- |
| block\_no | Номер блока в карте |
| *value* | Указатель на ячейку памяти, куда будет прочитан value |
| *addr* | Указатель на ячейку памяти, куда будет прочитан addr |

Возвращает EMVL1\_STATUS\_OK в случае успеха или ненулевой код ошибки.

* + 1. *int contactless\_mifare\_write\_value(uint8\_t block\_no, uint32\_t value, uint8\_t addr);*

Выполняет запись value-блока карты.

Входные параметры

|  |  |
| --- | --- |
| block\_no | Номер блока в карте |
| *value* | Значение value |
| *addr* | Значение addr |

Возвращает EMVL1\_STATUS\_OK в случае успеха или ненулевой код ошибки.

* + 1. *int contactless\_mifare\_operate(char type, uint8\_t block\_no, uint32\_t value, uint8\_t update\_block\_no);*

Выполняет операции над value-блоком карты.

Входные параметры

|  |  |
| --- | --- |
| *type* | Тип операции. Может быть + - или = |
| block\_no | Номер блока-источника в карте |
| *value* | Значение value |
| *update\_block\_no* | Номер блока-приемника в карте |

Возвращает EMVL1\_STATUS\_OK в случае успеха или ненулевой код ошибки.

* + 1. *int contactless\_mifare\_restore\_transfer(uint8\_t block\_no, uint8\_t update\_block\_no);*

Выполняет операцию restore-transfer над value-блоком карты.

Входные параметры

|  |  |
| --- | --- |
| block\_no | Номер блока-источника в карте |
| *update\_block\_no* | Номер блока-приемника в карте |

Возвращает EMVL1\_STATUS\_OK в случае успеха или ненулевой код ошибки.

* + 1. *int* *contactless\_mifareul\_auth\_pwd(uint8\_t\* pwd, uint8\_t\* pack);*

Выполняет аутентификацию к карте ultralight-c.

Входные параметры

|  |  |
| --- | --- |
| *pwd* | Указатель на буфер, содержащий pwd |
| *pack* | Указатель на буфер, содержащий pack |

Возвращает EMVL1\_STATUS\_OK в случае успеха или ненулевой код ошибки.

* + 1. *int contactless\_mifareul\_read(uint8\_t page, uint8\_t\* data);*

Выполняет чтение блока карты ultralight.

Входные параметры

|  |  |
| --- | --- |
| *page* | Страница в карте |
| *data* | Указатель на буфер, куда будут помещены прочитанные данные |

Возвращает EMVL1\_STATUS\_OK в случае успеха или ненулевой код ошибки.

* + 1. *int contactless\_mifareul\_write(uint8\_t page, uint8\_t\* data);*

Выполняет запись блока карты ultralight.

Входные параметры

|  |  |
| --- | --- |
| *page* | Страница в карте |
| *data* | Указатель на буфер, содержащий данные для записи |

Возвращает EMVL1\_STATUS\_OK в случае успеха или ненулевой код ошибки.

* + 1. *int contactless\_mifareul\_read\_sign(uint8\_t page, uint8\_t\* data);*

Выполняет чтение подписи карты ultralight.

Входные параметры

|  |  |
| --- | --- |
| *page* | Страница в карте |
| *data* | Указатель на буфер, куда будут помещены прочитанные данные |

Возвращает EMVL1\_STATUS\_OK в случае успеха или ненулевой код ошибки.

* + 1. *int contactless\_mifareul\_read\_cnt(uint8\_t cnt\_n, uint32\_t\* cnt\_value);*

Выполняет чтение счетчика карты ultralight.

Входные параметры

|  |  |
| --- | --- |
| *cnt\_n* | Номер счетчика в карте |
| *cnt\_value* | Указатель на ячейку, куда будут помещены прочитанные данные |

Возвращает EMVL1\_STATUS\_OK в случае успеха или ненулевой код ошибки.

* + 1. *int contactless\_mifareul\_inc\_cnt(uint8\_t cnt\_n, uint32\_t inc\_value);*

Выполняет инкремент счетчика карты ultralight.

Входные параметры

|  |  |
| --- | --- |
| *cnt\_n* | Номер счетчика в карте |
| *inc\_value* | Значение инкремента |

Возвращает EMVL1\_STATUS\_OK в случае успеха или ненулевой код ошибки.

* + 1. *int* *contactless\_mifareulc\_auth(uint16\_t key\_n, uint16\_t key\_v);*

Выполняет аутентификацию к карте ultralight-c на ключе из SAM AV2.

Вызывающий должен пройти авторизацию к SAM AV2, смотри contactless\_mifare\_sam\_enable();

Входные параметры

|  |  |
| --- | --- |
| *key\_n* | Номер ключа в хранилище |
| *key\_v* | Версия ключа в хранилище |

Возвращает EMVL1\_STATUS\_OK в случае успеха или ненулевой код ошибки.

* + 1. *int* *contactless\_mifare\_sam\_set\_host\_key(uint8\_t\* key, uint8\_t key\_nmb, uint8\_t key\_type, uint8\_t key\_pos, uint8\_t key\_ver);*

Устанавливает ключ хоста для аутентификации к SAM AV2 в программное хранилище ключей.

Хранилище имеет 2 ячейки по 2 позиции каждая.

Входные параметры

|  |  |
| --- | --- |
| *key* | Указатель на буфер, сожержащий значение ключа |
| *key\_nmb* | Номер ключа в хранилище, 0..1 |
| *key\_type* | Тип ключа:SAM\_KEY\_TYPE\_AES128 SAM\_KEY\_TYPE\_AES192SAM\_KEY\_TYPE\_AES256SAM\_KEY\_TYPE\_DESSAM\_KEY\_TYPE\_2K3DESSAM\_KEY\_TYPE\_3K3DESSAM\_KEY\_TYPE\_MIFARE |
| *key\_pos* | Позиция ключа в хранилище, 0..1 |
| *key\_ver* | Версия ключа |

Возвращает EMVL1\_STATUS\_OK в случае успеха или ненулевой код ошибки.

* + 1. *int* *contactless\_mifare\_sam\_enable(uint8\_t port, uint8\_t auth\_type, uint8\_t host\_key\_nmb, uint8\_t host\_key\_ver, uint8\_t sam\_key\_nmb, uint8\_t sam\_key\_ver);*

Выполняет аутентификацию к SAM AV2 и включает режим работы через SAM-модуль.

Вызывающий должен предварительно загрузить ключ для аутентификации хоста в программное хранилище, смотри *contactless\_mifare\_sam\_set\_host\_key.*

Входные параметры

|  |  |
| --- | --- |
| *port* | Номер слота SAM, 1..2 |
| *auth\_type* | Тип аутентификацииSAM\_AUTH\_MODE\_PLAINSAM\_AUTH\_MODE\_CMAC SAM\_AUTH\_MODE\_FULL |
| *value* | Значение value |
| *host\_key\_nmb* | Номер ключа в программном хранилище ключей хоста |
| *host\_key\_ver* | Версия ключа хоста |
| *sam\_key\_nmb* | Номер ключа в SAM, на котором производится аутентификация |
| *sam\_key\_ver* | Версия ключа SAM |

Возвращает EMVL1\_STATUS\_OK в случае успеха или ненулевой код ошибки.

* + 1. *int contactless\_mifare\_sam\_disable();*

Выключает режим работы через SAM-модуль.

Входные параметры

|  |  |
| --- | --- |
| *cnt\_n* | Номер счетчика в карте |
| *inc\_value* | Значение инкремента |

Возвращает EMVL1\_STATUS\_OK в случае успеха или ненулевой код ошибки.

* + 1. *int contactless\_get\_peer\_info (uint8\_t param, uint8\_t\* output)*

возвращает параметры последней активированной карты.

Входные параметры

|  |  |
| --- | --- |
| *param* | Тип параметров. Возможные значения:0 – Тип карты1 – Параметры 14443-3а2 – Параметры 14443-3b3 – Параметры 14443-3a |

**0 – Тип карты.** Возвращает 4 байта:

enum

{

 E\_PH\_NFCLIB\_UNKNOWN = 0,

 E\_PH\_NFCLIB\_MIFARE\_ULTRALIGHT, /\*\*< Card Type can be any one of MIFARE Ultralight, MIFARE Ultralight C, MIFARE Ultralight EV1. \*/

 E\_PH\_NFCLIB\_MIFARE\_CLASSIC\_1K,

 E\_PH\_NFCLIB\_MIFARE\_CLASSIC\_4K,

 E\_PH\_NFCLIB\_MIFARE\_DESFIRE,

 E\_PH\_NFCLIB\_TYPEA\_LAYER3,

 E\_PH\_NFCLIB\_TYPEA\_LAYER4,

 E\_PH\_NFCLIB\_TYPEB\_LAYER3,

 E\_PH\_NFCLIB\_TYPEB\_LAYER4,

 E\_PH\_NFCLIB\_ISO15693,

 E\_PH\_NFCLIB\_ISO18000p3m3,

 E\_PH\_NFCLIB\_LAST = 0x7FFFFFFF

}

**1 – Параметры 14443-3а.** Возвращает 14 байт:

struct

{

 uint16\_t wAtqa;

 uint8\_t bUidLength;

 uint8\_t aUid[10];

 uint8\_t bSak;

} tIso14443\_3a;

**2 – Параметры 14443-3b.** Возвращает 14 байт:

struct

{

 uint8\_t bAtqbLength;

 uint8\_t aPupi[4];

 uint8\_t aApplicationData[4];

 uint8\_t bIsExtendedAtqb;

 uint8\_t aProtocolInfo[4];

} tIso14443\_3b;

**3 – Параметры 14443-4а.** Возвращает 65 байт:

struct

{

 uint8\_t bDid;

 uint8\_t aAts[PH\_NXPNFCRDLIB\_CONFIG\_ATS\_BUFF\_LENGTH];

} tIso14443\_4;

**Примечание:** сначала надо выяснить тип активированной карты, затем запрашивать параметры. Некорректный запрос (к примеру запрос параметров 14443-3b для карт типа А) приведёт к возврату некорретных данных.

1. **Работа с контактными картами**
	1. **Документация**
	* ISO/IEC 7816-3:2006(E)
	* EMV® Integrated Circuit Card Specifications for Payment Systems Book 1. Application Independent ICC to Terminal Interface Requirements. Version 4.3 November 2011
	1. **Библиотека L1**

Библиотека L1 (далее L1) реализована внутри библиотеки "libaqc\_emv\_ct\_l1.so" Библиотека поставляется вместе с заголовочным файлом "contact.h"

* 1. **Определены следующие коды возврата и константы:**
* EMVL1CT\_STATUS\_OK = 0 - Ошибок нет
* EMVL1CT\_STATUS\_TIMEOUT = -1 - таймаут при обмене с картой
* EMVL1CT\_STATUS\_NOT\_INITIALIZED = -99 - Не инициализирована библиотека или один из ее компонентов
* EMVL1CT\_STATUS\_FAILED = -8 - Операция не выполнена
* EMVL1CT\_STATUS\_NOT\_PRESENT = -9 - Карта не обнаружена
* EMVL1\_SCR\_SLOT = 0 – слот смарт-карты
* EMVL1\_SAM1\_SLOT = 1 – слот SAM1
* EMVL1\_SAM2\_SLOT = 2 – слот SAM2
	1. **Функции для работы со смарт-картами EMVco:**

#### *uint32\_t contact\_getversion ()*

Возвращает версию контактной библиотеки

##### Возвращает:

Старшее слово – номер сборки. В младшем слове старший байт – старший номер версии, младший байт – младший номер версии. В случае ошибки возвращает 0.

#### *int contact\_probe ()*

Проверяет наличие смарт-карты в считывателе

##### Возвращает:

EMVL1CT\_STATUS\_OK если карта вставлена или EMVL1CT\_STATUS\_NOT\_PRESENT если карта отсутствует

#### *int contact\_open ()*

Конфигурирует оборудование и открывает порт для работы со смарт-картой

##### Возвращает:

EMVL1CT\_STATUS\_OK если ошибок нет или код ошибки

#### *int contact\_close ()*

Закрывает порт смарт-карты

##### Возвращает:

EMVL1CT\_STATUS\_OK если ошибок нет или код ошибки

#### *int contact\_start ()*

Активирует смарт-карту в считывателе. Вызывать надо после **contact\_open()**

Перед вызовом необходимо убедиться, что карта находится в считывателе

##### Возвращает:

Длину ATR или код ошибки

#### *int contact\_stop ()*

Деактивирует ранее активированную смарт-карту

##### Возвращает:

EMVL1CT\_STATUS\_OK если ошибок нет или код ошибки

#### *int contact\_apdu (char \* req, int qsz, char \* rsp, int \* psz)*

Осуществляет обмен APDU со смарт-картой

##### Аргументы:

|  |  |
| --- | --- |
| *req*  | Указатель на буфер C-APDU  |
| *qsz*  | Длина C-APDU  |
| *rsp*  | Указатель на заренее выделенный буфер куда будет принят R-APDU  |
| *psz*  | Указатель на ячейку куда будет возвращена длина R-APDU  |

##### Возвращает:

EMVL1CT\_STATUS\_OK если ошибок нет или код ошибки

#### *int contact\_get\_atr (char \* rsp, int \* psz)*

Возвращает ATR, полученный при активации смарт-карты

##### Аргументы:

|  |  |
| --- | --- |
| *rsp*  | Указатель на заранее выделенный буфер, куда будет скопирован ATR  |
| *psz*  | Указатель на ячейку куда будет возвращена длина ATR  |

##### Возвращает:

EMVL1CT\_STATUS\_OK если ошибок нет или код ошибки

* 1. **Функции для работы с SAM:**

#### *int sam\_open (uint8\_t port)*

Конфигурирует оборудование и открывает порт для работы с SAM.

##### Аргументы:

|  |  |
| --- | --- |
| *port*  | номер слота SAM  |

##### Возвращает:

EMVL1CT\_STATUS\_OK если ошибок нет или код ошибки

#### *int sam\_close (uint8\_t port)*

Закрывает порт SAM.

##### Аргументы:

|  |  |
| --- | --- |
| *port*  | номер слота SAM  |

##### Возвращает:

EMVL1CT\_STATUS\_OK если ошибок нет или код ошибки

#### *int sam\_start (uint8\_t port)*

Активирует SAM в считывателе. Вызывать надо после **sam\_open()**

##### Аргументы:

|  |  |
| --- | --- |
| *port*  | номер слота SAM  |

##### Возвращает:

Длину ATR или код ошибки

#### *int sam\_stop (uint8\_t port)*

Деактивирует ранее активированный SAM.

##### Аргументы:

|  |  |
| --- | --- |
| *port*  | номер слота SAM  |

##### Возвращает:

EMVL1CT\_STATUS\_OK если ошибок нет или код ошибки

#### *int sam\_apdu (uint8\_t port, char \* req, int qsz, char \* rsp, int \* psz)*

Осуществляет обмен APDU с SAM.

##### Аргументы:

|  |  |
| --- | --- |
| *port*  | номер слота SAM  |
| *req*  | Указатель на буфер C-APDU  |
| *qsz*  | Длина C-APDU  |
| *rsp*  | Указатель на заренее выделенный буфер куда будет принят R-APDU  |
| *psz*  | Указатель на ячейку куда будет возвращена длина R-APDU  |

##### Возвращает:

EMVL1CT\_STATUS\_OK если ошибок нет или код ошибки

#### *int sam\_get\_atr (uint8\_t port, char \* rsp, int \* psz)*

Возвращает ATR, полученный при активации SAM.

##### Аргументы:

|  |  |
| --- | --- |
| *port*  | номер слота SAM  |
| *rsp*  | Указатель на заранее выделенный буфер куда будет скопирован ATR  |
| *psz*  | Указатель на ячейку куда будет возвращена длина ATR  |

##### Возвращает:

EMVL1CT\_STATUS\_OK если ошибок нет или код ошибки

**3. Магнитная полоса**

**3.1 open**

**int msr\_open()**

Инициализирует интерфейс ридера магнитной полосы. Возвращает 0 в случае успеха или ненулевой код ошибки.

**3.2 close**

**int msr\_close()**

Завершает работу с ридером магнитной полосы. Возвращает 0 в случае успеха или ненулевой код ошибки.

**3.3 probe**

**int msr\_probe(uint8\_t \* track1, int \* tsz1, uint8\_t \* track2, int \* tsz2)**

Проверяет проведение карты в ридере. Если карта проведена возвращает 0 и копирует первый и второй трек в буферы track1 и track2 и возвращает размеры треков в параметрах tsz1 и tsz2. Если карта не проведена, возвращает STATUS\_NOT\_PRESENT или в случае ошибки ненулевой код.

**4 Коды ошибок**

#define STATUS\_SUCCESS 0

#define STATUS\_TIMEOUT -1

#define STATUS\_INVALID\_PARAMETER -4

#define STATUS\_WRITE\_ERROR -3

#define STATUS\_READ\_ERROR -5

#define STATUS\_MEMORY\_ERROR -6

#define STATUS\_FORMAT\_ERROR -7

#define STATUS\_FAILED -8

#define STATUS\_NOT\_PRESENT -9

#define STATUS\_PROTOCOL\_ERROR -10

#define STATUS\_TRANSMISSION\_ERROR -11

#define STATUS\_STOP -12

#define STATUS\_COLLISION\_ERROR -13

**5. Включение дополнительного логирования по библиотеке L1 в syslog**

**5.1 void contactless\_set\_log\_level (uint8\_t level)**

Переключение уровня логгирования в syslog

##### Аргументы:

|  |  |
| --- | --- |
| *level* | 0 – логирование выключено1 – средний уровень логирования2 – полное логирование |