Base HD - basic usage

AIS Readers API

Application Programing Interface for Autonomous Identification System Readers

Introduction

This document describes the functions from the AIS Readers dynamic library.

A brand new Base HD device accepts all known NFC cards and store detected NFC UIDs to the reader memory - creating the LOG. This document focus on the basic functionality: connect to the readers, get and set time, change password, listen for the real time events and getting stored events (LOGs).

Library

The library is used to controls and gets information from **A**utonomous **I**dentification **S**ystem (hereinafter referred to as AIS) Readers. AIS Readers are devices that read NFC cards (or QR codes) and autonomous collect events and make decisions about validity of the events.

Supported readers

Library support several AIS Reader device types:

AIS Start

AIS Start is a MiFare card reader used for time and attendance.

It can connect to the PC via USB port.

The AIS Start readers communicates via a USB port.

Base HD

BASE HD is a MiFare card reader used for time and attendance and control access.

BASE HD readers communicate through the RS485 protocol.

Barcode Mifare Reader

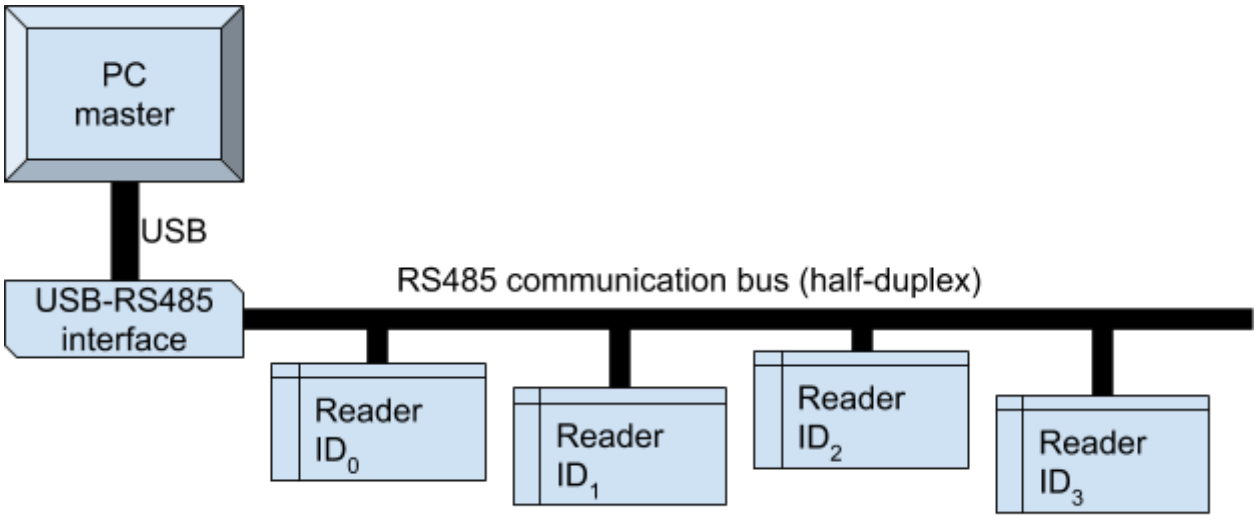
BMR is a MiFare card and QR code reader - used for time and attendance and control access. BMR communicate through the RS485 protocol.

Look at the appendix AIS READERS types for all supported readers.

Communication with reader

All AIS readers can work autonomously, without any additional control *needed*.

To configure the AIS Reader device or collect some information like real-time events or log, connect the reader to the communication bus and use AIS Readers library.



Every reader connected to the communication bus must have unique ID.

Set Reader ID with Base HD configurator

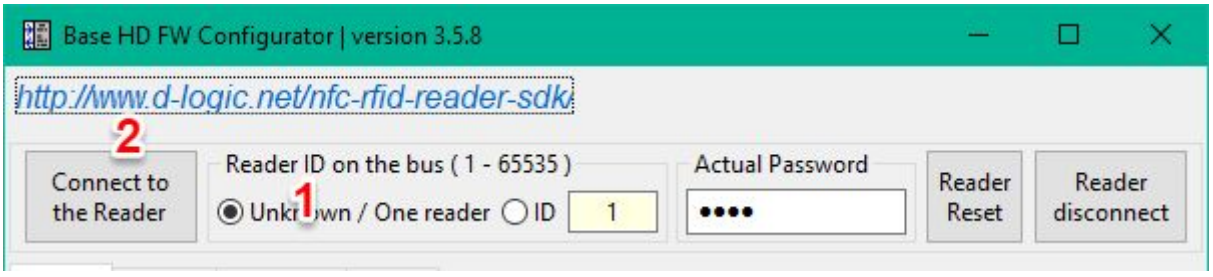
The only way to change Reader ID is from the *Base HD configurator* PC application.

Download the application from link below:

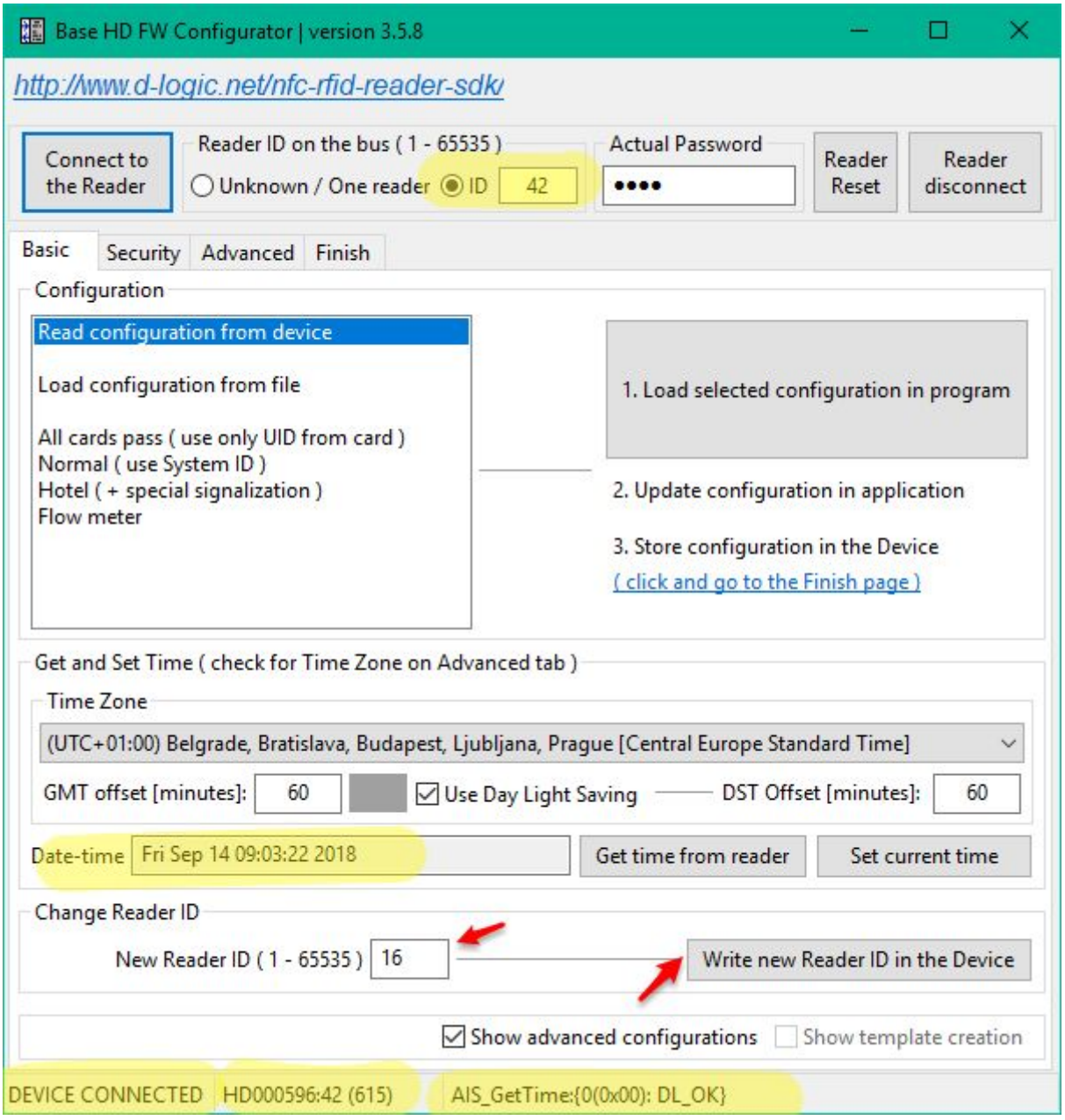
https://www.d-logic.net/code/nfc-rfid-reader-sdk/ais_readers-configurator-executable.git

The reader must be alone on the communication bus and connected to the PC.

First, select “Unknown / One reader” and then click to “Connect to the Reader”



After a successful connection, the form will look like picture (changes is highlighted):



Now, enter new Reader ID in the text box and click to “Write new Reader ID in the Device”

On successful change, the message “Reader ID changed. Reader reopen” will appear in the status bar.

Principle of using the library to communicate with AIS Readers

For successful communication with AIS readers, the host must follow some rules and steps.

1. Enumerate readers

Firstly, the host system need to check how many devices are connected to the communication buses. Then, host gets the device handlers and basic information of every connected reader.

2. Communication

Host communicates with particular reader via API functions. The device handle is the address of a

particular reader in the API calls.

There are three types of API execution:

- Short commands - independent of AIS_MainLoop()

A short command completes its functionality in one call, e.g. AIS_GetVersion(), AIS_GetTime(), AIS_LockOpen(), ...

- Chained commands - depend of AIS_MainLoop()

To execute chained command you have to constantly poll AIS_MainLoop(). At first, start command, then do the AIS_MainLoop() polling until *cmdResponse* parameter become true, and at the end gets the results of the command. Look at Getting events history - LOG for example.

- Real-time events - only AIS_MainLoop()

Getting the real-time events is a specific functionality, and system must constantly polling AIS_MainLoop() for this purpose. Parameter *RealTimeEvents* indicates that *RTE* is arrived, than host calls AIS_GetRTE() for getting results.

3. Close communication

Close communication before exit from application.

Data types

DL_STATUS

Almost every API function returns status of execution. DL_STATUS is an enumeration - 32-bit long number. Value zero (0) or DL_OK indicates that the function execution was successful, all other values indicate an error.

A list of all known statuses is in the Appendix: DL_STATUS - status codes.

c_string

Type for representing null-terminated char array, also known as C-string.

When function returns type c_string, and when we need to get c_string from function: only declare c_string variable, no need to allocate memory space for whole array.

But when we need to send data to the function like password, whitelist string, etc - on the first must allocate and initialize memory space for char array before use.

Array is always one byte longer then string length - for null character.

HND_AIS

HND_AIS is a data type for storing the handle of a reader object.

Library version

AIS_GetLibraryVersionStr

Function description

Returns version of library in string format.

Function declaration (C language)

```
c_string AIS_GetLibraryVersionStr(void);
```

Example

"AIS READERS library version = 4.1.1"

AIS_GetLibraryVersion

Function description

Returns 32 bit unsigned integer with packet major, minor, and build version information.

Function declaration (C language)

```
uint32_t AIS_GetLibraryVersion(void);
```

Example

if the version of library is 4.9.1 then function would return value 0x00040901

Enumerate readers

The purpose of readers enumerating process is to establish communication between host and devices connected to the communication buses.

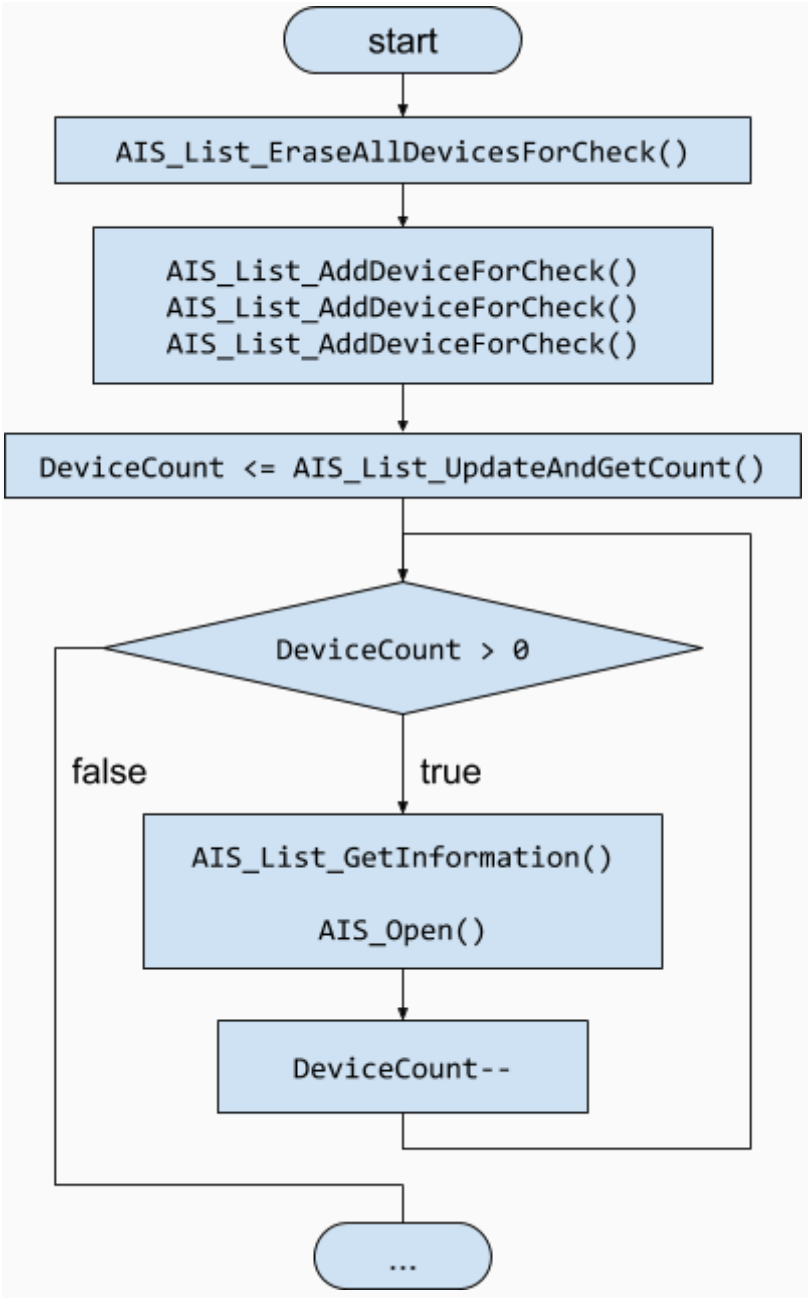
First step is deleting the existing list of device parameters by calling the function AIS_List_EraseAllDevicesForCheck().

The second step is to create a list of devices that are connected to the bus by calling the function AIS_List_AddDeviceForCheck() for each device. In this step, the devices type and their ID are defined.

Third step is an attempt to establish communication with the all devices that are defined in the list by calling the function AIS_List_UpdateAndGetCount().

Fourth step is getting the information about all devices connected to the bus by calling the function AIS_List_GetInformation() for each device.

The last step is opening of communication port for each device by calling the function AIS_Open().



Note:

If you have more than one device on the bus, you'll need to provide all of the readers IDs to the function AIS_List_AddDeviceForCheck().

Opening communication example in C programming language :

```
#include <stdio.h>
#include <ais_readers.h>

void open_readers_demo(void) {
```

```
DL_STATUS status;
int DeviceCount = 0;

// prepare list with device types and IDs for checking
// we expect to open three Base HD devices with ID 16, 18 and 20
AIS_List_EraseAllDevicesForCheck();

status = AIS_List_AddDeviceForCheck(DL_AIS_BASE_HD_SDK, 16);
if (status) {
      // print "warning: already in list"
}
status = AIS_List_AddDeviceForCheck(DL_AIS_BASE_HD_SDK, 18);
if (status) {
      // print "warning: already in list"
}
status = AIS_List_AddDeviceForCheck(DL_AIS_BASE_HD_SDK, 20);
if (status) {
      // print "warning: already in list"
}

// try to communicate with every listed device
status = AIS_List_UpdateAndGetCount(&DeviceCount);
if (status) {
      printf("Error, AIS_List_UpdateAndGetCount() status is %s\n",
                  dl_status2str(status));

      return;
}

for (int DeviceIndex = 0; DeviceIndex < DeviceCount; ++DeviceIndex) {

      HND_AIS Device_HND; //
      c_string Device_Serial; //
      int Device_Type; //
      int Device_ID; //
      int Device_FW_VER; //
      int Device_CommSpeed; //
      c_string Device_FTDI_Serial; //
      int Device_isOpened; //
      int Device_Status; //
      int System_Status;

      status = AIS_List_GetInformation(&Device_HND, &Device_Serial,
                  &Device_Type, &Device_ID, &Device_FW_VER, &Device_CommSpeed,
                  &Device_FTDI_Serial, &Device_isOpened, &Device_Status,
                  &System_Status);
      if (status) {
            printf("Error, AIS_List_GetInformation() status is %s",
                        dl_status2str(status));
            // no more enumerated devices
```

```
                break;
            }

            status = AIS_Open(Device_HND);
            if (status) {
                printf("Error, AIS_Open() status is %s\n", dl_status2str(status));
            } else {
                printf("Device successfully opened!\n");

                // application must take care of device handles (Device_HND)
                // put device handle to some list of opened devices
            }
        }
}
```

WARNING: Pass value 0 as a Ais_List_AddDeviceForCheck() function parameter "Reader ID" only for the purpose of configuring device which is currently the only one connected to the host.

AIS_List_EraseAllDevicesForCheck

Function description

Clear list of all available devices for checking.

Function declaration (C language)

```
void AIS_List_EraseAllDevicesForCheck(void);
```

AIS_List_AddDeviceForCheck

Function description

Set list of available AIS reader device types

Function declaration (C language)

```
DL_STATUS AIS_List_AddDeviceForCheck(device_e device_type,
                                     int device_id);
```

Parameters

device_type	device type by internal specification (enumeration)
device_id	Reader ID - set by Mifare Init Card

WARNING : Don't pass 0 as device_id if you have more than one device on the bus! You must provide all readers IDs.

AIS_List_EraseDeviceForCheck

Function description

Remove specific reader from list for checking.

Function declaration (C language)

```
DL_STATUS AIS_List_EraseDeviceForCheck(device_e device_type,
                                       int device_id);
```

Parameters

device_type	device type by internal specification (enumeration)
device_id	Reader ID - set by Mifare Init Card

AIS_List_GetDevicesForCheck

Function description

Function returns which device will be checked.

Function declaration (C language)

```
c_string AIS_List_GetDevicesForCheck(void);
```

AIS_List_UpdateAndGetCount

Function description

This function builds a internal device information list and returns the number of AIS devices connected to the system. The list contains information about both unopen and open devices.

If the devices connected to the system change, the device info list will not be updated until AIS_List_UpdateAndGetCount is called again.

Function declaration (C language)

```
DL_STATUS AIS_List_UpdateAndGetCount(int *device_count);
```

Parameters

device_count	Number of attached devices

AIS_List_GetInformation

Function description

With this function you can get all information about device. Function returns DL_STATUS.

Function declaration (C language)

```
DL_STATUS AIS_List_GetInformation(HND_AIS *pDevice_HND,
                                  c_string *pDevice_Serial,
                                  int *pDevice_Type,
                                  int *pDevice_ID,
                                  int *pDevice_FW_VER,
                                  int *pDevice_CommSpeed,
                                  c_string *pDevice_FTDI_Serial,
                                  int *pDevice_isOpened,
                                  int *pDevice_Status,
                                  int *pSystem_Status);
```

Parameters

pDeviceHND	pointer to variable holding information about device handle
pDevice_Serial	pointer to c_string holding information about device serial number
pDevice_Type	pointer to variable holding information about device type
pDevice_ID	pointer to variable holding information about device ID
pDevice_FW_VER	pointer to variable holding information about device firmware version
pDevice_CommSpeed	pointer to variable holding information about device communication speed
pDevice_FTDI_Serial	pointer to c_string holding information about device FTDI Serial
pDevice_isOpened	pointer to variable holding information if communication with device is opened
pDevice_Status	pointer to variable holding information about device status

pSystem_Status	pointer to variable holding information about system status

---

AIS_DeviceTypeInfo

Function description

Function will return information about device based on the provided device type.

Function declaration (C language)

```
DL_STATUS AIS_DeviceTypeInfo(IN device_e dev_type,
                             OUT c_string *name,
                             OUT c_string *description,
                             OUT uint32_t *hw_type,
                             OUT uint32_t *speed,
                             OUT uint32_t *rte_test,
                             OUT uint32_t *is_half_duplex,
                             OUT uint32_t *is_alone_on_the_bus);
```

Parameters

dev_type	Device type
name	Short name
description	Device description - full name
hw_type	Hardware type in D-LOGIC type enumeration
speed	Communication speed of the device
rte_test	How often library test RTE in the device (in milliseconds)
is_half_duplex	Is device half duplex (if 0, device is full duplex)
is_alone_on_the_bus	If only one device is on the bus

AIS_DeviceTypeToEnum

Function description

Translate enumeration E_KNOWN_DEVICE_TYPES from string representation to the integer (enum) e.g. "DL_AIS_BMR" to enum value 9.

Function declaration (C language)

```
DL_STATUS AIS_DeviceTypeToEnum(IN c_string dev_type_str,
                               OUT device_e *dev_type_enum);
```

Parameters

dev_type_enum	Device type
dev_type_str	Short name

AIS_DeviceTypeToString

Function description

Translate enumeration E_KNOWN_DEVICE_TYPES from enumeration (integer) to the string pointer e.g. 9 translate to pointer to string "DL_AIS_BMR".

Function declaration (C language)

```
DL_STATUS AIS_DeviceTypeToString(IN device_e dev_type_enum,
                                 OUT c_string *dev_type_str);
```

Parameters

dev_type_enum	Device type
dev_type_str	Short name

Open and close reader

AIS_Open

Function description

Function opens port for communication with reader.

Function declaration (C language)

```
DL_STATUS AIS_Open(HND_AIS handle);
```

Parameters

handle	Device handle

AIS_Close

Function description

Function closes port for communication with reader.

Function declaration (C language)

```
DL_STATUS AIS_Close(HND_AIS handle);
```

Parameters

handle	Device handle

AIS_Restart

Function description

Function restarts the reader and re-initialize object in library.

Function declaration (C language)

```
DL_STATUS AIS_Restart(HND_AIS handle)
```

Parameters

handle	Device handle

Reader version

AIS_GetVersion

Function description

With this function you can get device's hardware and firmware version.

Function declaration (C language)

```
DL_STATUS AIS_GetVersion(HND_AIS handle,
                         int* hardware_type,
                         int* firmware_version);
```

Parameters

handle	Device handle
hardware_type	Pointer to variable holding information about device's hardware type
firmware_version	Pointer to variable holding information about device's firmware version

Security - password

The AIS readers have security level that only the granted persons, which known the reader password, can change the reader settings, and get log.

AIS_ChangePassword

Function description

Function changes device password.

Default password is “1111” - 5 byte long array of characters where last byte is set to zero.

Function declaration (C language)

```
DL_STATUS AIS_ChangePassword(HND_AIS handle,
                             c_string old_password,
                             c_string new_password);
```

Parameters

handle	Device handle
old_password	Old device password
new_password	New password that will be set

Date and Time

All AIS readers have a real-time clock (RTC) for logging moment of execution of events, and access control decision.

AIS_GetTime

Function description

Function returns time and date data with timezone and offset.

Function declaration (C language)

```
DL_STATUS AIS_GetTime(HND_AIS handle,
                      uint64_t *current_time,
                      int *timezone,
                      int *DST,
                      int *offset,
                      void **additional);
```

Parameters

handle	Device handle
current_time	Pointer to variable holding information about GMT timestamp
timezone	Pointer to variable holding information about how many seconds west of GMT
DST	Pointer to variable holding information if Daylight Saving Time is used
offset	Seconds west of GMT if DST is used
additional	Additional byte in configuration (firmware specific), SET TO 0

GMTstampToString

Function description

Function returns static array of characters (C-string) based on the given timestamp.
e.g. "GMT= 1455378371, Sat Feb 13 15:46:11 2016"

Function declaration (C language)

```
c_string GMTstampToString(uint64_t gm_timestamp);
```

Parameter

gm_timestamp	GMT timestamp

AIS_SetTime

Function description

Function sets GMT time, timezone and offset into reader. You have to provide device's password to set time.

Function declaration (C language)

```
DL_STATUS AIS_SetTime(HND_AIS handle,
                      c_string password,
                      const uint64_t time_to_set,
                      int timezone,
                      int DST,
                      int offset,
                      void *additional);
```

Parameters

handle	Device handle
password	Device password
time_to_set	GMT timestamp
timezone	Seconds west of GMT
DST	If this value is set to 1, Daylight Saving Time is used
offset	Seconds west of GMT if DST is used
additional	additional byte in configuration (firmware specific), SET TO 0

Helper functions for working with time

There are several helper functions to get information about time zone from system.

sys_get_timezone

Function description

Function returns seconds west of GMT.

Function declaration (C language)

```
long sys_get_timezone(void);
```

---

sys_get_daylight

The function returns information whether it is currently active day-light saving (DST).

Function description

Function declaration (C language)

```
int sys_get_daylight(void);
```

---

sys_get_dstbias

Function description

Function return GMT offset.

Function declaration (C language)

```
int sys_get_dstbias(void);
```

---

sys_get_timezone_info

Function description

Function returns information about timezone (C - string).

Function declaration (C language)

```
c_string sys_get_timezone_info(void);
```

The Main Loop

The AIS main loop's primary role is to listen (and check) for the real-time events, and for execution chained command like downloading history records (LOG). Executing the main loop will also provide error information in the AIS System.

The proper functioning of the AIS system, like promptly signalling about RTE, is ensured by the constant execution of the main loop. Recommended frequency of execution is several times (100) in seconds per AIS reader.

Chained commands need main loop polling to avoid blocking the application on the host and to get information about command progress. Therefore, chained commands are also called ***non-blocking*** functions.

More about the main loop principles can find at https://en.wikipedia.org/wiki/Event_loop

AIS_MainLoop

Function description

The AIS_MainLoop do lot of things like getting asynchronous incoming data, checking for status of the AIS reader, saving information in certain buffers, keeping records about execution of commands, checking timeouts and information about new events.

Check the function parameters for details.

Be careful when executing more then one AIS_MainLoop() from multiple threads because parameter synchronisation between threads.

However, the main loop will not provide RTE information while a chained command is in progress.

Function declaration (C language)

```
DL_STATUS AIS_MainLoop(HND_AIS handle,
                       int *RealTimeEvents,
                       int *LogAvailable,
                       int *LogUnread,
                       int *cmdResponse,
                       int *cmdPercent,
                       int *DeviceStatus,
                       int *TimeoutOccured,
                       int *Status);
```

Parameters

handle	Device handle
RealTimeEvents	Indicate new Real Time Event
LogAvailable	Indicate new data in log buffer
LogUnread	Indicate unread log from the device
DeviceStatus	Device status flags
cmdResponse	indicate finished chained command execution
cmdPercent	Indicate percent of command execution - progress
TimeoutOccured	Debug only
Status	Additional status, status of the chained command execution

---

Real Time Events

Information that a real-time event happened is provided by the *RealTimeEvents* parameter in AIS_MainLoop. The AIS_MainLoop must be executed several times (100) per seconds (per AIS reader) for promptly signalling about RTE.

AIS_ReadRTE_Count

Function description

Functions returns the number of the Real Time Events (RTE) which is ready (in the library buffer) to read with AIS_ReadRTE for certain reader.

Function declaration (C language)

```
int AIS_ReadRTE_Count(HND_AIS handle);
```

AIS_ReadRTE

Function description

Get information about Real Time Events.

The successfully taken event is deleted from the library buffer, but not from device.

Function declaration (C language)

```
DL_STATUS AIS_ReadRTE(HND_AIS handle,
                      int *log_index,
                      int *log_action,
                      int *log_reader_id,
                      int *log_card_id,
                      int *log_system_id,
                      uint8_t nfc_uid[NFC_UID_MAX_LEN],
                      int *nfc_uid_len, uint64_t *timestamp);
```

Parameters

handle	Device handle
log_index	Pointer to variable holding log index
log_action	Action that happened when card was detected
log_reader_id	Pointer to variable holding device ID
log_card_id	Pointer to variable holding card ID
log_system_id	System ID
nfc_uid	Array of bytes containing UID of card
nfc_uid_len	Pointer to variable holding information about UID length
timestamp	Pointer to variable holding information about time when RTE happened

---

Getting events history - LOG

LOG is the history of the events that are stored in the reader. Maximum number of events in LOG depends of the reader configuration.

Getting log is a chained process. Firstly, getting log starts with either AIS_GetLog or AIS_GetLogByIndex or AIS_GetLogByTime API function. Then, the AIS_MainLoop constantly executed. The progress of execution is in **cmdPercent** parameter of AIS_MainLoop. At the end, when parameter **cmdResponse** become true, number of the events (stored in the library buffer) is in parameter **LogAvailable** of AIS_MainLoop or can get with AIS_ReadLog_Count. Read events from library buffer with multiple calls of AIS_ReadLog, one call per event.

After reading, logs are not deleted from the reader only from the library buffer.

Look at example Reading logs from device.

AIS_GetLog

Function description

Start command for retrieves all logs from the Reader.

Non-blocking function, must execute MainLoop and wait for *cmdResponse* parameter in MainLoop to become true.

Function declaration (C language)

```
DL_STATUS AIS_GetLog(HND_AIS handle,
                     c_string password);
```

Parameters

handle	Device handle
password	Device password

AIS_GetLogByIndex

Function description

Start command for retrieves logs, in provided index range, from the Reader.

Non-blocking function, must execute MainLoop and wait for *cmdResponse* parameter in MainLoop to become true.

Function declaration (C language)

```
DL_STATUS AIS_GetLogByIndex(HND_AIS handle,
                            c_string password,
                            uint32_t start_index,
                            uint32_t end_index);
```

Parameters

handle	Device handle
password	Device password
start_index	Where to start reading
end_index	Where to finish reading

AIS_GetLogByTime

Function description

Start command for retrieves logs, in provided time range, from the Reader.

Non-blocking function, must execute MainLoop and wait for cmdResp parameter in MainLoop to become true.

Function declaration (C language)

```
DL_STATUS AIS_GetLogByTime(HND_AIS handle,
                           c_string password,
                           uint64_t time_from,
                           uint64_t time_to);
```

Parameters

handle	Device handle
password	Device password
time_from	Time where to start reading
time_to	Time where to finish reading

AIS_ReadLog_Count

Function description

Function returns the number of available logs (downloaded from the device) in the library buffer.

Function declaration (C language)

```
int AIS_ReadLog_Count(HND_AIS handle);
```

Parameter

handle	Device handle

AIS_ReadLog

Function description

Call this function multiple times to read all available logs from device (library buffer).

Function declaration (C language)

```
DL_STATUS AIS_ReadLog(HND_AIS handle,
                      int *log_index,
                      int *log_action,
                      int *log_reader_id,
                      int *log_card_id,
                      int *log_system_id,
                      uint8_t nfc_uid[NFC_UID_MAX_LEN],
                      int *nfc_uid_len,
                      uint64_t *timestamp);
```

Parameters

handle	Device handle
log_index	Log index, starts from 0
log_action	Action that happened when card was detected
log_reader_id	Device ID
log_card_ID	Card ID
log_system_id	System ID
nfc_uid	Array of bytes containing card UID
nfc_uid_len	Pointer to variable holding information about UID length
timestamp	Pointer to variable holding information about time when RTE happened

---

AIS_ClearLog

Function description

Function clears all logs received in the library buffer.

Function declaration (C language)

```
DL_STATUS AIS_ClearLog(HND_AIS handle);
```

Parameter

handle	Device handle

Signal control

AIS Readers can emit sound (beeper) and light (red and green LED) signals.

AIS_LightControl

Function description

This function turns on or off lights on the Base HD readers.

Function declaration (C language)

```
DL_STATUS AIS_LightControl(HND_AIS handle,
                           uint32_t green_master,
                           uint32_t red_master,
                           uint32_t green_slave,
                           uint32_t red_slave);
```

Parameters

handle	Device handle
green_master	If this value is set to 1, green light on master device will turn ON
red_master	If this value is set to 1, red light on master device will turn ON
green_slave	If this value is set to 1, green light on slave device will turn ON
red_slave	If this value is set to 1, red light on slave device will turn ON

Access Control

AIS Readers, like Base HD and BMR, contain parts like relays to control another device like gate.

AIS_LockOpen

Function description

Start pulse signal on LOCK port (on Base HD) to open locked gate or door.

Function declaration (C language)

```
DL_STATUS AIS_LockOpen(HND_AIS handle, uint32_t pulse_duration);
```

Parameters

handle	Device handle
pulse_duration	duration of active signal (pulse) in milliseconds

---

AIS_RelayStateSet

Function description

Function declaration (C language)

```
DL_STATUS AIS_RelayStateSet(HND_AIS handle, uint32_t state);
```

Parameters

handle	Device handle
state	Look at enumeration relay state

Enumeration - relay state:

0	RELAY_CLOSE_BOTH	Both relays are turned off
1	RELAY_OPEN_A	Relay A is turned ON, and relay B is turned OFF
2	RELAY_OPEN_B	Relay A is turned OFF, and relay B is turned ON
3	RELAY_OPEN_BOTH	Both relay A and relay B are turned ON

Base HD contains only relay A. BMR contains both relays A and B.

AIS_GetIoState

Function description

This function returns states of the signals on Base HD.

Function declaration (C language)

```
DL_STATUS AIS_GetIoState(HND_AIS handle,
                         uint32_t *intercom,
                         uint32_t *door,
```

```
                                    uint32_t *relay_state);
```

Parameters

handle	Device handle
intercom	Is activate signal on *intercom* port
door	Is active signal on *door* port
relay_state	State of relays, look at enumeration relay state

Appendix: AIS READERS types

Data type **device_e** is enumeration (E_KNOWN_DEVICE_TYPES) for known device types.

ID	short name	description
0	DL_UNKNOWN_DEVICE	not valid value
1	DL_AIS_100	AIS GOLD
2	DL_AIS_20	AIS BASE
3	DL_AIS_30	AIS BASE
4	DL_AIS_35	AIS BASE 4K
5	DL_AIS_50	AIS SILVER
6	DL_AIS_110	AIS GOLD
7	DL_AIS_LOYALTY	AIS LOYALTY
8	DL_AIS_37	AIS START (BASE BAT USB) (AIS IrM READER)
9	DL_AIS_BMR	Barcode NFC Reader Half-Duplex
10	DL_AIS_BASE_HD	Base HD AIS Half-Duplex
11	DL_AIS_BASE_HD_SDK	Base HD AIS SDK Half-Duplex
12	DL_AIS_IO_EXTENDER	AIS IO Extender

Appendix: DL_STATUS - status codes

AIS_StatusToString

Function description

Function return string representation of DL_STATUS code.

Function declaration (C language)

```
c_string AIS_StatusToString(DL_STATUS status);
```

Parameter

status	DL_STATUS code

Status codes list:

value [dec]	value [hex]	Name
0	0x00	DL_OK
1	0x01	TIMEOUT_ERROR
2	0x02	NULL_POINTER
3	0x03	PARAMETERS_ERROR
4	0x04	MEMORY_ALLOCATION_ERROR
5	0x05	NOT_INITIALIZED
6	0x06	ALREADY_INITIALIZED
7	0x07	TIMESTAMP_INVALID
8	0x08	EVENT_BUSY
4096	0x1000	ERR_SPECIFIC
4097	0x1001	CMD_BRAKE_RTE
4098	0x1002	TRANSFER_ACK_FAILED
4099	0x1003	NO_RF_PACKET_DATA
4100	0x1004	TRANSFER_WRITING_ERROR
4101	0x1005	EVENT_WAKEUP_BUSY

4102	0x1006	DEVICE_RESET_OCCURRED
8192	0x2000	RESOURCE_NOT_ACQUIRED
8193	0x2001	RESOURCE_ALREADY_ACQUIRED
8194	0x2002	RESOURCE_BUSY
8195	0x2003	RESOURCE_NOT_OWNER
12288	0x3000	FILE_OVERSIZE
12289	0x3001	FILE_EMPTY
12290	0x3002	FILE_LOCKED
12291	0x3003	FILE_NOT_FOUND
12292	0x3004	ERR_NO_FILE_NAME
12293	0x3005	ERR_DIR_CAN_NOT_CREATE
12294	0x3006	ERR_FILE_NOT_CORRECT
16384	0x4000	ERR_DATA
16385	0x4001	ERR_BUFFER_EMPTY
16386	0x4002	ERR_BUFFER_OVERFLOW
16387	0x4003	ERR_CHECKSUM
16388	0x4004	LOG_NOT_CORRECT
28672	0x7000	LIST_ERROR
28673	0x7001	ITEM_IS_ALREADY_IN_LIST
28674	0x7002	ITEM_NOT_IN_LIST
28675	0x7003	ITEM_NOT_VALID

32768	0x8000	NO_DEVICES
32769	0x8001	DEVICE_OPENING_ERROR
32770	0x8002	DEVICE_CAN_NOT_OPEN
32771	0x8003	DEVICE_ALREADY_OPENED
32772	0x8004	DEVICE_NOT_OPENED
32773	0x8005	DEVICE_WRONG_HANDLE_ERROR
32774	0x8006	DEVICE_CLOSE_ERROR
32775	0x8007	DEVICE_UNKNOWN
32776	0x8008	DEVICE_NOT_SUPPORTED
36864	0x9000	ERR_COMMAND_RESPONSE
36865	0x9001	CMD_RESPONSE_UNKNOWN_COMMAND
36866	0x9002	CMD_RESPONSE_WRONG_CMD
36867	0x9003	CMD_RESPONSE_COMMAND_FAILED
36868	0x9004	CMD_RESPONSE_UNSUCCESS
36869	0x9005	CMD_RESPONSE_NO_AUTHORIZATION
36870	0x9006	CMD_RESPONSE_SIZE_OVERFLOW
36871	0x9007	CMD_RESPONSE_NO_DATA
40960	0xA000	THREAD_FAILURE
40961	0xA001	ERR_OBJ_NOT_CREATED
40962	0xA002	ERR_CREATE_SEMAPHORE
45056	0xB000	ERR_STATE_MACHINE
45057	0xB001	ERR_SM_IDLE__NO_RESPONSE
45058	0xB002	ERR_SM_COMMAND_IN_PROGRESS

45059	0xB003	ERR_SM_NOT_IDLE
45060	0xB004	ERR_SM_CMD_NOT_STARTED
45061	0xB005	WARN_STOP_POLLING
53248	0xD000	READER_ERRORS_
53249	0xD001	READER_UID_ERROR
53250	0xD002	READER_LOG_ERROR
57344	0xE000	DL_HAMMING_ERROR
57345	0xE001	DL_HAMMING_NOT_ACK
57346	0xE002	DL_HAMMING_WRONG_ACK
57347	0xE003	DL_HAMMING_WRONG_REPLAY
57348	0xE004	ERROR_SOME_REPLAY_FAULT
57349	0xE005	DL_HAMMING_TERR_TIMEOUT
57350	0xE006	DL_HAMMING_TERR_BAD_FRAME
57351	0xE007	DL_HAMMING_TERR_BAD_CODE
57352	0xE008	DL_HAMMING_TERR_TOO_OLD
57353	0xE009	DL_HAMMING_TERR_NOISE
57354	0xE010	DL_HAMMING_TERR_ERROR_MASK
61440	0xF000	NO_FTDI_COMM_DEVICES
61441	0xF001	NO_FTDI_COMM_DEVICES_OPENED
61442	0xF002	ERR_FTDI
61443	0xF003	ERR_FTDI_READ
61444	0xF004	ERR_FTDI_READ_LESS_DATA
61445	0xF005	ERR_FTDI_WRITE

61446	0xF006	ERR_FTDI_WRITE_LESS_DATA
61447	0xF007	DL_FT_ERROR_SET_TIMEOUT
61696	0xF100	DL_FT_
61697	0xF101	DL_FT_INVALID_HANDLE
61698	0xF102	DL_FT_DEVICE_NOT_FOUND
61699	0xF103	DL_FT_DEVICE_NOT_OPENED
61700	0xF104	DL_FT_IO_ERROR
61701	0xF105	DL_FT_INSUFFICIENT_RESOURCES
61702	0xF106	DL_FT_INVALID_PARAMETER
61703	0xF107	DL_FT_INVALID_BAUD_RATE
61704	0xF108	DL_FT_DEVICE_NOT_OPENED_FOR_ERASE
61705	0xF109	DL_FT_DEVICE_NOT_OPENED_FOR_WRITE
61706	0xF110	DL_FT_FAILED_TO_WRITE_DEVICE
61707	0xF111	DL_FT_EEPROM_READ_FAILED
61708	0xF112	DL_FT_EEPROM_WRITE_FAILED
61709	0xF113	DL_FT_EEPROM_ERASE_FAILED
61710	0xF114	DL_FT_EEPROM_NOT_PRESENT
61711	0xF115	DL_FT_EEPROM_NOT_PROGRAMMED
61712	0xF116	DL_FT_INVALID_ARGS
61713	0xF117	DL_FT_NOT_SUPPORTED
61714	0xF118	DL_FT_OTHER_ERROR
61715	0xF119	DL_FT_DEVICE_LIST_NOT_READY

-2	0xFFFF FFFE	NOT_IMPLEMENTED
-1	0xFFFF FFFF	UNKNOWN_ERROR

Appendix: Card action

Data type e_card_action is the enumeration E_CARD_ACTION which provides detailed information of the events.

The values in red rows indicates blocked events.

The values in green rows indicates granted events.

dec	hex	short name	description
0	0x00	CARD FOREIGN	strange card - card from different AIS system
32	0x20	CARD DISCARDED	blocked card - card on blacklist, no valid access right, has no right of passage
64	0x40	CARD HACKED	Bad protective data -- Mifare key OK - CRC OK - but bad user data
80	0x50	CARD BAD DATA	Cards with invalid data -- BAD CRC - Mifare key OK - CRC BAD
96	0x60	CARD NO DATA	unreadable card - card without or unknown Mifare key
112	0x70	QR UNLOCKED	OK QR code
113	0x71	QR BLOCKED	QR was rejected
114	0x72	QR BLACKLISTED	QR was rejected because the blacklist (already evident)
115	0x73	QR BLOCKTIME	QR was rejected because the date-time is out of range
116	0x74	QR FOREIGN	QR was rejected because the foreign job number
117	0x75	QR UNKNOWN	QR was rejected because the unknown QR data format
118	0x76	QR BLOCK GROUP	QR was rejected because the group membership (wrong BMR group)
128	0x80	CARD UNLOCKED	The correct card
129	0x81	CARD UNLOCKED 1	AIS Card Serial Number was in range of ACTION-1
130	0x82	CARD UNLOCKED 2	AIS Card Serial Number was in range of ACTION-2
131	0x83	CARD UNLOCKED 3	AIS Card Serial Number was in range of ACTION-3
132	0x84	CARD UNLOCKED 4	AIS Card Serial Number was in range of ACTION-4
133	0x85	CARD UNLOCKED 5	AIS Card Serial Number was in range of ACTION-5
134	0x86	CARD UNLOCKED 6	AIS Card Serial Number was in range of ACTION-6
135	0x87	CARD UNLOCKED 7	AIS Card Serial Number was in range of ACTION-7
136	0x88	UNLOCKED WHITELIST	The card UID or IMEI were on the white list
137	0x89	UNLOCKED UID	System is open - no AIS job defined - all card can pass

138	0x8A	UNLOCKED INTERCOM	The opening signal came on interfone contact
139	0x8B	UNLOCKED DIGITAL IN	The opening signal came on digital IN contact
144	0x90	IO EXPANDER OK	successfully open target on IO expander
145	0x91	IO EXPANDER FAIL	the target was not open with the IO expander
255	0xFF	UNKNOWN	Unknown card action

AIS_ActionToString

Function description

Function will return string representation of card action. Card action is stored into device for every log, unread log, or real time event.

Function declaration (C language)

```
c_string AIS_ActionToString(e_card_action action);
```

Parameter

`action`	Action that happened when card was detected

Example in c programming language :

Reading logs from device :

```
int log_index;
  int log_action;
  int log_reader_id;
  int log_card_id;
  int log_system_id;
  uint8_t nfc_uid[10];
  int nfc_uid_len;
  uint64_t timestamp;
  DL_STATUS status, status_last;
  int DeviceStatus_last;
  c_string device_password = "";
  bool cmd_finish;
```

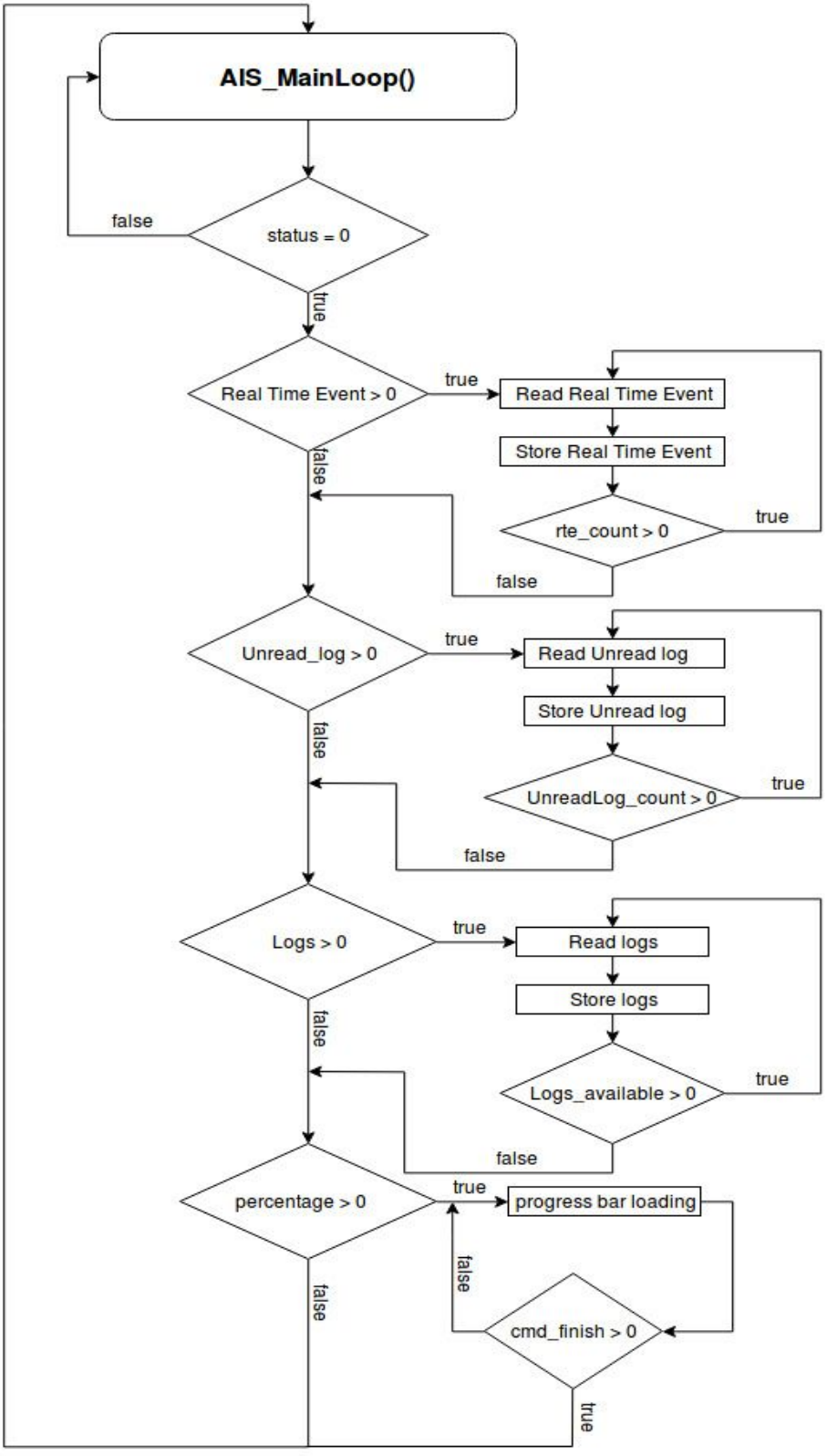
```
       int RealTimeEvents;
       int LogAvailable;
       int LogUnread;
       int cmdResponse;
       int cmdPercent;
       int DeviceStatus;
       int TimeoutOccurred;
       int Status;

       for(int i = 0; i < device_count; i++)
     {
             AIS_ClearLog(handle[i]);
             status = AIS_GetLog(handle[i], device_password);
          if (status)
              return;
         do
         {
              status = AIS_MainLoop(handle[i], &RealTimeEvents,       &LogAvailable,
&LogUnread, &cmdResponse, &cmdPercent, &DeviceStatus, &TimeoutOccurred, &Status);
             printf("\rDownloading logs... %d",cmdPercent);
             if (status)
             {
                  if (status_last != status)
                  {
                       status_last = status;
                  }
                  return;
             }

             if (cmdResponse)
             {
                cmd_finish = true;
             }

             if (DeviceStatus_last != DeviceStatus)
             {
                  DeviceStatus_last = DeviceStatus;
             }
       }
       while (!cmd_finish);
       cmd_finish = false;
        do
       {
            status = AIS_ReadLog(handle[i], &log_index, &log_action, &log_reader_id,
&log_card_id, &log_system_id, nfc_uid, &nfc_uid_len, &timestamp);
            //Here you can print your results
            if (status)
                break;
       }
       while(true);
```

```
}
```


AIS_MainLoop()
status = 0
false
true
Real Time Event > 0
true
Read Real Time Event
Store Real Time Event
rte_count > 0
true
false
false
Unread_log > 0
true
Read Unread log
Store Unread log
UnreadLog_count > 0
true
false
false
Logs > 0
true
Read logs
Store logs
Logs_available > 0
true
false
false
percentage > 0
true
progress bar loading
false
cmd_finish > 0
false
true