libSD Documentation

Release 0.8.0

ANSSI

Oct 10, 2019

CONTENTS

Contents

Library providing an API for using a SDcard.

CHAPTER

ONE

OVERVIEW

TODO

CHAPTER

TWO

API

2.1 Initializing the stack

Initialize the SD library is made through two main functions:

```
#include "libsd.h"

uint8_t sd_early_init(void);
uint32_t sd_init(void);
```

the early init step is called before the task ends its initialization phase using sys_init(INIT_DONE) syscall. This syscall declare all the requested ressources that can only be declared at initialization time. This include the SDIO device memory mapping.

The init step initialize the SD stack context. At the end of this function call, the SD stack is ready to read or write data from the SDCard or return any information from it (blocksize, number of blocks, etc.).

Caution: Even if the DFU stack internal is ready for handling DFU requests, these requests are executed by the dfu_exec_automaton() function that need to be executed

The task has to declare a buffer and a buffer size that will be used by the DFU stack to hold firmware chunks during the UPLOAD and DOWNLOAD states.

The buffer size depend on the task constraints but **must be a multiple of the control plane USB URB size** (usually 64 bytes length).

2.2 Getting informations from the SDCard

Accessing the SDCard informations is done using the following API

```
#include "libsd.h"

uint32_t sd_get_capacity(void);
uint32_t sd_get_block_size(void);
uint32_t sd_get_block_number(void);
```

FIXME description.

2.3 Accessing SDCard data

Reading from or writing into the SDCard is done using the following API

```
#include "libsd.h"

int8_t sd_read(uint32_t *buffer, uint32_t addr, uint32_t len);
int8_t sd_write(uint32_t *buffer, uint32_t addr, uint32_t len);
```

FIXME description.

CHAPTER

THREE

FAQ