swissbit®

libsbltm – Life Time Monitoring Library

Windows / Linux

Users Guide

BU: Swissbit AG
Date: 23 January 2023
Version: 3.0.0

Contents

1 OVERVIEW 3

1.1 LIBRARY USAGE 4

1.1.1 Windows 4

1.1.2 Linux 4

2 FUNCTION REFERENCE 5

LIBSBLTM_API sbltm_library_version (BYTE * *major*, BYTE * *minor*, BYTE * *build*) 5
LIBSBLTM_API sbltm_library_initialize (char * *name*, UINT32 *key1*, UINT32 *key2*) 5
LIBSBLTM_API sbltm_library_cleanup () 5
LIBSBLTM_API sbltm_enumerate_drives (BYTE * *driveCount, BOOL addNonSupported*) 5
LIBSBLTM_API sbltm_library_event_callback (void(*)(void) sbltm_callback) 6
LIBSBLTM_API sbltm_get_device_type (BYTE *drive*, SB_DEVICE_TYPE * *type*) 6
LIBSBLTM_API sbltm_get_drive_path (BYTE *drive*, char * *buf*, UINT16 *bufsize*) 6
LIBSBLTM_API sbltm_get_filesystem_info (BYTE *drive*, char * *buf*, UINT16 *bufsize*) 7
LIBSBLTM_API sbltm_get_drive_description (BYTE *drive*, char * *buf*, UINT16 *bufsize*) 8
LIBSBLTM_API sbltm_get_model (BYTE *drive*, char * *buf*, UINT16 *bufsize*) 8
LIBSBLTM_API sbltm_get_serial (BYTE *drive*, char * *buf*, UINT16 *bufsize*) 8
LIBSBLTM_API sbltm_get_firmware_revision (BYTE *drive*, char * *buf*, UINT16 *bufsize*) 9
LIBSBLTM_API sbltm_get_capacity (BYTE *drive*, UINT64 * *sectors*) 9
LIBSBLTM_API sbltm_get_interface_mode_supported (BYTE drive, char * buf, UINT16 bufsize) 9
LIBSBLTM_API sbltm_get_interface_mode_active (BYTE drive, char * buf, UINT16 bufsize) 10
LIBSBLTM_API sbltm_get_lifetime_info (BYTE *drive*, SBLTMinfo * *infos*) 10
LIBSBLTM_API sbltm_get_ecc_capability (BYTE *drive*, char * *buf*, UINT16 *bufsize*) 10
LIBSBLTM_API sbltm_ata_get_identify (BYTE *drive*, BYTE * *data*) 10
LIBSBLTM_API sbltm_ata_smart_enable (BYTE *drive*) 11
LIBSBLTM_API sbltm_ata_smart_disable (BYTE *drive*) 11
LIBSBLTM_API sbltm_ata_smart_read_status (BYTE *drive*, BOOL * *pass*) 11
LIBSBLTM_API sbltm_ata_smart_read_values (BYTE drive, BYTE * data) 11
LIBSBLTM_API sbltm_ata_smart_read_thresholds (BYTE *drive*, BYTE * *data*) 12
LIBSBLTM_API sbltm_sd_read_register (BYTE *drive*, SB_REGISTER *reg*, BYTE * *data*, UINT16 *length*) 12
LIBSBLTM_API sbltm_raid_get_count (BYTE* raidCount) 12
LIBSBLTM_API sbltm_raid_get_infos (BYTE *raid*, SBLTM_RAIDinfo * *infos*) 12
LIBSBLTM_API sbltm_raid_index_by_drive (BYTE *drive*, BYTE* *raidIndex*, BYTE* *raidDriveIndex*) 13
LIBSBLTM_API_EX const char* sbltm_error_msg (SBERROR *errnr*) 13

3 DATA STRUCTURE DOCUMENTATION 14

3.1 SBLTMINFO STRUCT REFERENCE 14

3.1.1 Data Fields 14

3.2 SBLTM_RAIDINFO STRUCT REFERENCE 16

3.2.1 Data Fields 16

3.3 SBLTM_RAIDDRIVEINFO STRUCT REFERENCE 16

3.3.1 Data Fields 16

3.4 SBLTMERASEINFO STRUCT REFERENCE 17

3.4.1 Data Fields 17

3.5 ENUMERATION TYPE DOCUMENTATION 18

enum SBERROR 18
enum SB_DEVICE_TYPE 18
enum SBLTM_RaidStatus 18
enum SBLTM_RaidDriveStatus 18
enum SBLTM_RaidDriveUsage 19
enum SB_REGISTER 19
enum SB_FLASH_CELL_MODE 19

3.6 DEFINES 19

1 Overview

The Swissbit Life Time Monitoring Library provides a standard C API to enumerate Swissbit devices in a system, and read life time diagnostics data from the device(s).

Initialization:

- sbltm_library_version() validate the correct DLL version
- sbltm_library_initialize() initialize the library, required for all other functions
- sbltm_library_cleanup() clean up library internal structures
- sbltm_library_event_callback() set event callback to get device change notifications

General, device enumeration functions:

- sbltm_enumerate_drives() enumerate all supported drives in a system
- sbltm_get_device_type() get the device type (IDE/SATA drive, USB drive)
- sbltm_get_drive_description() get a short device description
- sbltm_error_msg() get an error description
- sbltm_device_filter_add() add a device filter for enumeration and device change event
- sbltm_device_filter_clear() clears all device filters

Device information access functions:

- sbltm_get_model() get the model number
- sbltm_get_serial() get the serial number
- sbltm_get_firmware_revision() get the firmware revision string
- sbltm_get_capacity() get the device capacity
- sbltm_get_drive_path() get the system device path
- sbltm_get_filesystem_info() get short description of volume filesystem
- sbltm_get_lifetime_info() read and decode all available life time information
- sbltm_get_ecc_capability() get error correction capability information
- sbltm_get_interface_mode_supported() get the supported interface mode as an information string
- sbltm_get_interface_mode_active() get the active interface mode as an information string
- sbltm_get_interface_degraded_status() check if the current interface speed is degraded

ATA device specific convenience functions:

- sbltm_ata_get_identify() get ATA identify data of a device
- sbltm_ata_smart_enable() enable SMART
- sbltm_ata_smart_disable() disable SMART
- sbltm_ata_smart_read_status() read the overall SMART status
- sbltm_ata_smart_read_values() read the raw SMART values
- sbltm_ata_smart_read_thresholds() read the raw SMART threshold values

SD and MMC device specific convenience functions:

- sbltm_sd_read_register() get SD/MMC registers of a device

RAID specific functions:

- sbltm_raid_get_count() get number of RAID's detected
- sbltm_raid_get_infos() get RAID information
- sbltm_raid_index_by_drive() get RAID index by drive index

For an explanation of the low level SMART commands and values please consult the Swissbit datasheet.

1.1 Library Usage

1.1.1 Windows

The SBLTM Library is shipped with the DLL (libsbltm.dll), the header file (libsbltm.h) and the linker file (libsbltm.lib). The DLL is linked statically; no special libraries need to be installed on the target system. There are builds available for both 32bit and for 64bit architectures.

The library supports Windows 7 and up, administrator privileges are required (limitation of Windows).

The include file to use is libsbltm.h. It contains all definitions and functions which are described in this document.

```
#include <libsbltm.h>
```

To link the DLL you will have to include the libsbltm.lib as an additional dependency in your linker settings.
Swissbit provides a small demonstration application to show basic usage.

1.1.2 Linux

The SBLTM Library is shipped with the shared library (libsbltm.so) and the header file (libsbltm.h). The library is dynamically linked to libc.so.6 and libm.so.6. For other configurations please contact Swissbit.

The library supports Linux kernels version 2.6 and up, root privileges are required (limitation of the Linux kernel).

The include file to use is libsbltm.h. It contains all definitions and functions which are described in this document.

```
#include <libsbltm.h>
```

Swissbit provides a small demonstration application to show basic usage.

2 Function Reference

LIBSBLTM_API sbltm_library_version (BYTE * *major*, BYTE * *minor*, BYTE * *build*)

Get the library version of the DLL.

At least libraries with the same major and minor number will be binary compatible. This is the only function that does not require the library to be initialized.

Parameters:

major	will be filled with the major version number
minor	will be filled with the minor version number
build	will be filled with the build version number

Returns:

SB_ERROR_NONE on success, error number otherwise (see SBERROR)

LIBSBLTM_API sbltm_library_initialize ()

Initialize the library, allocates the necessary internal memory structures.

Returns:

SB_ERROR_NONE on success, error number otherwise (see SBERROR)

LIBSBLTM_API sbltm_library_cleanup ()

Clean up the library, de-allocates internal memory structures.

Returns:

SB_ERROR_NONE on success, error number otherwise (see SBERROR)

LIBSBLTM_API sbltm_enumerate_drives (BYTE * *driveCount, BOOL addNonSupported*)

Enumerate all drives in the system.

This should function must be called prior to any other drive related functions.

Parameters:

driveCount	be filled with number of supported drives found
addNonSupported	If TRUE, the library will support reading SMART values from Non-Swissbit devices. Those devices will not be able to return data on sbltm_get_lifetime_info

Returns:

SB_ERROR_NONE on success, error number otherwise (see SBERROR)

LIBSBLTM_API sbltm_device_filter_add (const char * *filter*)

Add a device filter for enumeration and device change events. This can be called multiple times to add multiple filters.

If no device filter is set, all Swissbit drives are reported. All set filter can be cleared using sbltm_device_filter_clear()

Parameters

filter	Filter to be set. Can be a path name (i.e. \\.\PHYSICALDRIVE1 for Windows or /dev/sdb for Linux) or a volume name (E: for Windows, /dev/sdb1 for Linux)

Returns

SB_ERROR_NONE on success, error number otherwise (see SBERROR)

LIBSBLTM_API sbltm_device_filter_clear ()

Clears all device filters.

Clears all filters previously added by sbltm_device_filter_add()

Returns

SB_ERROR_NONE on success, error number otherwise (see SBERROR)

LIBSBLTM_API sbltm_library_event_callback (void(*)(void) sbltm_callback)

Set callback method for device change events.

Please note that this function will be called in a separate thread context.

Parameters:

sbltm_callback	Pointer to a function that will get called on device changes

Returns:

SB_ERROR_NONE on success, error number otherwise (see SBERROR)

LIBSBLTM_API sbltm_get_device_type (BYTE *drive*, SB_DEVICE_TYPE * *type*)

Get the device type.

Parameters:

drive	Drive number
type	Will be filled with the device type (see SB_DEVICE_TYPE)

Returns:

SB_ERROR_NONE on success, error number otherwise (see SBERROR)

LIBSBLTM_API sbltm_get_drive_path (BYTE *drive*, char * *buf*, UINT16 *bufsize*)

Get the drive path by drive number.

Parameters:

drive	Drive number
buf	Will be filled with a null-terminated ASCII string holding the drive path. Buffer must be allocated by the application
bufsize	Number of elements allocated in buf

Returns:

SB_ERROR_NONE on success, error number otherwise (see SBERROR)

LIBSBLTM_API sbltm_get_filesystem_info (BYTE *drive*, char * *buf*, UINT16 *bufsize*)

Get short description of volume filesystem by drive number.

Parameters:

drive	Drive number
buf	Will be filled with a null-terminated ASCII string holding a short volume filesystem description. Buffer must be allocated by the application
bufsize	Number of elements allocated in buf

Returns:

SB_ERROR_NONE on success, error number otherwise (see SBERROR)

LIBSBLTM_API sbltm_get_drive_description (BYTE *drive*, char * *buf*, UINT16 *bufsize*)

Get short drive description by drive number.

Parameters:

drive	Drive number
buf	Will be filled with a null-terminated ASCII string holding a short device description. Buffer must be allocated by the application
bufsize	Number of elements allocated in buf

Returns:

SB_ERROR_NONE on success, error number otherwise (see SBERROR)

LIBSBLTM_API sbltm_get_model (BYTE *drive*, char * *buf*, UINT16 *bufsize*)

Get the model of the device.
For ATA devices this is the ATA model (SF...). For USB devices this is the Vendor and Product String.

Parameters:

drive	Drive number
buf	Will be filled with a null-terminated ASCII string holding the model string. Buffer must be allocated by the application.
bufsize	Number of elements allocated in buf

Returns:

SB_ERROR_NONE on success, error number otherwise (see SBERROR)

LIBSBLTM_API sbltm_get_serial (BYTE *drive*, char * *buf*, UINT16 *bufsize*)

Get the serial number of the device.

Parameters:

drive	Drive number
buf	Will be filled with a null-terminated ASCII string holding the serial string. Buffer must be allocated by the application.
bufsize	Number of elements allocated in buf

Returns:

SB_ERROR_NONE on success, error number otherwise (see SBERROR)

LIBSBLTM_API sbltm_get_firmware_revision (BYTE *drive*, char * *buf*, UINT16 *bufsize*)

Get the firmware revision of the device.
May be empty if unknown.

Parameters:

drive	Drive number
buf	Will be filled with a null-terminated ASCII string holding the firmware revision string. Buffer must be allocated by the application.
bufsize	Number of elements allocated in buf

Returns:

SB_ERROR_NONE on success, error number otherwise (see SBERROR)

LIBSBLTM_API sbltm_get_capacity (BYTE *drive*, UINT64 * *sectors*)

Get the full device capacity.

Parameters:

drive	Drive number
sectors	Will be filled with the number of sectors.

Returns:

SB_ERROR_NONE on success, error number otherwise (see SBERROR)

LIBSBLTM_API sbltm_get_interface_mode_supported (BYTE drive, char * buf, UINT16 bufsize)

Get the supported interface mode the device as an information string.
May be empty if unknown.

Parameters:

drive	Drive number
buf	Will be filled with a null-terminated ASCII string holding the supported mode (english). Buffer must be allocated by the application.
bufsize	Number of elements allocated in buf.

Returns:

SB_ERROR_NONE on success, error number otherwise (see SBERROR)

LIBSBLTM_API sbltm_get_interface_mode_active (BYTE drive, char * buf, UINT16 bufsize)

Get the current interface mode the device as an information string.
May be empty if unknown.

Parameters:

drive	Drive number
buf	Will be filled with a null-terminated ASCII string holding the active mode (english). Buffer must be allocated by the application.
bufsize	Number of elements allocated in buf.

Returns:

SB_ERROR_NONE on success, error number otherwise (see SBERROR)

LIBSBLTM_API sbltm_get_interface_degraded_status (BYTE *drive*, BOOL * *isDegraded*)

Check if the current interface speed is degraded (lower than supported)

Parameters:

drive	Drive number
isDegraded	Set to true if interface speed is degraded. Set to false if not degraded or unknown

Returns:

SB_ERROR_NONE on success, error number otherwise (see SBERROR)

LIBSBLTM_API sbltm_get_lifetime_info (BYTE *drive*, SBLTMinfo * *infos*)

Read and decode all available life time information into a SBLTMinfo structure.

Parameters:

drive	Drive number
infos	Pointer to an allocated SBLTMinfo structure, will be filled with information

Returns:

SB_ERROR_NONE on success, error number otherwise (see SBERROR)

LIBSBLTM_API sbltm_get_ecc_capability (BYTE *drive*, char * *buf*, UINT16 *bufsize*)

Get the ECC capability information.
May be empty if unknown.

Parameters:

drive	Drive number
buf	Will be filled with an nul-terminated ASCII string holding the ECC capability information (english). Buffer must be allocated by the application.
bufsize	Number of elements allocated in buf.

Returns

SB_ERROR_NONE on success, error number otherwise (see SBERROR)

LIBSBLTM_API sbltm_ata_get_identify (BYTE *drive*, BYTE * *data*)

Get ATA Identify data.

This function returns the raw ATA identification data, see ATA specification for details. Only supported for device types SB_DEVICE_ATA_DIRECT and SB_DEVICE_ATA_USB.

Parameters:

drive	Drive number
data	Pointer to allocated buffer of size SB_SECTOR_LEN

Returns:

SB_ERROR_NONE on success, error number otherwise (see SBERROR)

LIBSBLTM_API sbltm_ata_smart_enable (BYTE *drive*)

Enable SMART on a device.

Parameters:

drive	Drive number

Returns:

SB_ERROR_NONE on success, error number otherwise (see SBERROR)

LIBSBLTM_API sbltm_ata_smart_disable (BYTE *drive*)

Disable SMART on a device.

Parameters:

drive	Drive number

Returns:

SB_ERROR_NONE on success, error number otherwise (see SBERROR)

LIBSBLTM_API sbltm_ata_smart_read_status (BYTE *drive*, BOOL * *pass*)

Read the SMART status of a device.

Parameters:

drive	Drive number
pass	Will be filled with non-zero on PASS, zero on FAIL status

Returns:

SB_ERROR_NONE on success, error number otherwise (see SBERROR)

LIBSBLTM_API sbltm_ata_smart_read_values (BYTE drive, BYTE * data)

Read the raw SMART values of a device.

This function returns the raw SMART values. Please see Swissbit datasheets for details. If you don't need the raw SMART data, you can use sbltm_smart_read_infos() instead.

Parameters:

drive	Drive number
data	Pointer to allocated buffer of size SB_SECTOR_LEN

Returns:

SB_ERROR_NONE on success, error number otherwise (see SBERROR)

LIBSBLTM_API sbltm_ata_smart_read_thresholds (BYTE *drive*, BYTE * *data*)

Read the raw SMART thresholds of a device.

This function returns the raw SMART thresholds. Please see Swissbit datasheets for details. If you don't need the raw SMART data, you can use sbltm_smart_read_infos() instead.

Parameters:

drive	Drive number
data	Pointer to allocated buffer of size SB_SECTOR_LEN

Returns:

SB_ERROR_NONE on success, error number otherwise (see SBERROR)

LIBSBLTM_API sbltm_sd_read_register (BYTE *drive*, SB_REGISTER *reg*, BYTE * *data*, UINT16 *length*)

Read the raw CID data of a SD or MMC device.

Parameters:

drive	Drive number
reg	Register to read, see SB_REGISTER
data	Pointer to allocated buffer
length	Length of allocated buffer

Returns:

SB_ERROR_NONE on success, error number otherwise (see SBERROR)

LIBSBLTM_API sbltm_raid_get_count (BYTE* raidCount)

Get number of RAIDs.

This function will return the number of RAID drives found in this system. Currently only RAID drivers with CSMI (Common Storage Management Interface), are supported, like Intel Matrix / Rapid Storage Technology.

Information about the RAID can be retrieved using the sbltm_raid_get_information() function.

Parameters:

raidCount	Will be filled with number of RAIDs found

Returns:

SB_ERROR_NONE on success, error number otherwise (see SBERROR)

LIBSBLTM_API sbltm_raid_get_infos (BYTE *raid*, SBLTM_RAIDinfo * *infos*)

Get information about a RAID.

Parameters:

raid	RAID index (zero-based)
infos	Pointer to an allocated SBLTM_RAIDinfo structure, will be filled with information.

Returns:

SB_ERROR_NONE on success, error number otherwise (see SBERROR)

LIBSBLTM_API sbltm_raid_index_by_drive (BYTE *drive*, BYTE* *raidIndex*, BYTE* *raidDriveIndex*)

Get RAID index for a given drive.
Will return SB_ERROR_INVALID_DRIVER for non-RAID drives.

Parameters:

drive	Drive number
raidIndex	Will be filled with RAID index, can be NULL if data is not needed
raidDriveIndex	Will be filled with drive index (Drives array in SBLTM_RAIDinfo), can be NULL if data is not needed

Returns:

SB_ERROR_NONE on success, error number otherwise (see SBERROR)

LIBSBLTM_API_EX const char* sbltm_error_msg (SBERROR *errnr*)

Translate error number to a message.

Parameters:

errnr	Error number to translate

Returns:

Pointer to a constant string which holds the error description (ASCII, null-terminated)

3 Data Structure Documentation

3.1 SBLTMinfo Struct Reference

The SBLTMinfo structure holds all available health information values of a device.

SBLTM_Optional_... values are only optionally supported by the device (depending on product). The structs each hold a "Valid" Boolean, if this is TRUE, the "Value" parameter can be used.

3.1.1 Data Fields

- BOOL OverallHealthStatus
  FALSE if failure of device is impending.
- BYTE SpareBlockCountValue
  Spare block count attribute value (percent, for worst unit if global spare block management is not supported)
- BYTE SpareBlockCountThreshold
  Spare block count attribute threshold (percent)
- BOOL SpareBlockCountPrefail
  Spare block count attribute type prefail (TRUE) / advisory (FALSE)
- SBLTMeraseInfo EraseCountInfos [SB_MAX_ERASE_INFOS]
  TRUE if the EraseCount values are valid.*
- BYTE SpareBlockNumberOfUnits
  Number of firmware units, zero if unknown (SpareBlockPerUnit not filled).*
- UINT16 SpareBlockPerUnitCurrent [SB_MAX_UNITS]
  Current number of spare blocks per firmware unit.
- UINT16 SpareBlockPerUnitInitial [SB_MAX_UNITS]
  Initial number of spare blocks per firmware unit.
- UINT32 SpareBlockWorstInitial
  Spare block initial count on worst unit.
- UINT32 SpareBlockWorstCurrent
  Spare block current count on worst unit.
- UINT32 SpareBlockTotalInitial
  Spare block initial count over all units (0xFFFF if not supported)
- UINT32 SpareBlockTotalCurrent
  Spare block current count over all units.
- UINT16 SpareBlockMinimum
  Minimum spare blocks per unit required for the firmware to work.
- char ControllerRevision [8]
  Optional controller revision in format SM122X2, can be empty if not available.
- char FirmwareRevision [11]
  Optional firmware date code in format YYYY-MM-DD, can be empty if not available.
- SBLTM_Optional_UINT64 TotalEraseCount
  total number of flash block erase counts
- SBLTM_Optional_UINT32 TotalECCErrors
  total number of flash ECC errors
- SBLTM_Optional_UINT32 CorrectableECCErrors
  number of correctable flash ECC errors
- SBLTM_Optional_UINT64 TotalFlashReadCommands
  total number of flash read commands issued (relates to ECC errors)
- SBLTM_Optional_UINT32 PowerOnCounter
  number of power on cycles the device has seen

- SBLTM_Optional_UINT32 PowerOnTotalECCErrors
  power on repairs: total number of flash ECC errors
- SBLTM_Optional_UINT32 PowerOnCorrectableECCErrors
  power on repairs: number of correctable flash ECC errors
- SBLTM_Optional_UINT32 TotalUDMACRCErrors
  total number of UDMA CRC errors that have been seen
- SBLTM_Optional_UINT32 CommitCounter
  internal firmware commit counter
- SBLTM_Optional_UINT32 InitialBadBlockCount
  initial bad block count (at time of production)
- SBLTM_Optional_UINT64 PowerOnHours
  device power on hours (approx)
- SBLTM_Optional_UINT64 TrimCommandCount
  number of TRIM commands counted by the firmware
- SBLTM_Optional_BYTE TrimPercent
  percentage of device that is currently trimmed
- SBLTM_Optional_BOOL GlobalWearLeveling
  non-zero if global wear leveling is active
- SBLTM_Optional_BOOL GlobalBadBlockManagement
  non-zero if global bad block management is active
- SBLTM_Optional_UINT32 WearLevelThresh
  internal firmware wear level threshold
- SBLTM_Optional_UINT32 FlashBlockCount
  number of flash blocks that are available to wear leveling (relates to TotalEraseCount)
- SBLTM_Optional_BYTE CopyBackThresh
  internal firmware copy back threshold
- SBLTM_Optional_UINT64 TotalLBAsWritten
  number of LBA's (sectors) written, in resolution of 65'536 sectors
- SBLTM_Optional_UINT64 TotalLBAsRead
  number of LBA's (sectors) read, in resolution of 65'536 sectors
- SBLTM_Optional_INT16 TemperatureCurrent
  temperature current [Degrees Celcius]
- SBLTM_Optional_INT16 TemperatureMin
  temperature minimum [Degrees Celcius]
- SBLTM_Optional_INT16 TemperatureMax
  temperature maximum [Degrees Celcius]

3.2 SBLTM_RAIDinfo Struct Reference

The SBLTM_RAIDinfo structure holds all available information values about a RAID.

3.2.1 Data Fields

- char DriverDesc [50]
  Driver description string (zero-terminated ASCII)
- char DriverVersion [20]
  Driver version string (zero-terminated ASCII)
- char RaidType [5]
  textual description of RAID type (e.g. "1" for RAID 1)
- SBLTM_RaidStatus Status
  current RAID status
- BYTE DriveCountTotal
  Total (theoretical) drive count in this RAID.
- BYTE DriveCountAvailable
  Drive count in this RAID (currently available drives only). Designates the number of drives that are available in the Drives member below.
- SBLTM_RaidDriveInfo Drives [SBLTM_MAX_RAID_DRIVES]
  For each connected drive in the RAID, the structure is filled with information about its status and SBLTM API drive index.

3.3 SBLTM_RaidDriveInfo Struct Reference

The SBLTM_RaidDriveInfo structure holds all available information values about a RAID drive.

3.3.1 Data Fields

- BYTE Index
  index of drive in the SBLTM API (drive number)
- SBLTM_RaidDriveStatus Status
  current RAID status of drive
- SBLTM_RaidDriveUsage Usage
  current RAID usage of drive

3.4 SBLTMeraseInfo Struct Reference

The SBLTMeraseInfo structure holds all available erase count informations of the flash.

3.4.1 Data Fields

- BOOL Valid
  TRUE if structure values are valid
- BYTE EraseCount
  Erase count attribute value (percent)
- BYTE Threshold
  Erase count attribute threshold (percent)
- BOOL Prefail
  Erase count attribute type prefail (TRUE) / advisory (FALSE)
- SB_FLASH_CELL_MODE FlashCellMode
  Operating flash cell mode
- SBLTM_Optional_UINT32 RatedEraseCount
  Rated erase count value from the flash manufacturer
- SBLTM_Optional_UINT64 MaxEraseCount
  Maximum erase count of all flash blocks
- SBLTM_Optional_UINT64 AverageEraseCount
  Average erase count of flash blocks

3.5 Enumeration Type Documentation

enum SBERROR

This enumeration defines all possible error codes of the API.
Use the function sbltm_error_msg() to get the (english) error description for the error code.

Enumerator:

SB_ERROR_NONE No Error.
SB_ERROR_NOT_INITIALIZED Library not initialized.
SB_ERROR_OUT_OF_MEMORY Could not allocate enough memory.
SB_ERROR_NOT_SUPPORTED Function not supported.
SB_ERROR_INTERNAL Internal error.
SB_ERROR_INVALID_ARGUMENT Invalid argument passed to function.
SB_ERROR_ACCESS_DENIED Access to device denied.
SB_ERROR_INVALID_DRIVER Driver not supported for this functionality.
SB_ERROR_UNSUPPORTED_DEVICE Swissbit Device, but not supported, use newer version of library.
SB_ERROR_BUFFER_SIZE Buffer size that was allocated by application was too small.

enum SB_DEVICE_TYPE

This enumeration defines all supported device types.

Enumerator:

SB_DEVICE_ATA_DIRECT SMART capable SATA/IDE device directly connected to host.
SB_DEVICE_ATA_USB SMART capable SATA/IDE device behind USB bridge.
SB_DEVICE_USB USB device.
SB_DEVICE_SD SD device.
SB_DEVICE_MMC MMC device.
SB_DEVICE_NVME NVMe device.
SB_DEVICE_EMMC eMMC device.

enum SBLTM_RaidStatus

RAID status enumeration.

Enumerator:

SBLTM_RaidStatusUnknown
SBLTM_RaidStatusOK
SBLTM_RaidStatusDEGRADED
SBLTM_RaidStatusREBUILDING
SBLTM_RaidStatusFAILED
SBLTM_RaidStatusOFFLINE
SBLTM_RaidStatusTRANSFORMING
SBLTM_RaidStatusQUEUED_FOR_REBUILD
SBLTM_RaidStatusQUEUED_FOR_TRANSFORMATION

enum SBLTM_RaidDriveStatus

RAID drive status enumeration.

Enumerator:

SBLTM_RaidDriveStatus_Unknown
SBLTM_RaidDriveStatus_OK
SBLTM_RaidDriveStatus_REBUILDING
SBLTM_RaidDiveStatus_FAILED
SBLTM_RaidDriveStatus_DEGRADED

SBLTM_RaidDriveStatus_OFFLINE
SBLTM_RaidDriveStatus_QUEUED_FOR_REBUILD

enum SBLTM_RaidDriveUsage

RAID drive usage enumeration.

Enumerator:

SBLTM_RaidDriveUsage_Unknown
SBLTM_RaidDriveUsage_NOT_USED
SBLTM_RaidDriveUsage_MEMBER
SBLTM_RaidDriveUsage_SPARE
SBLTM_RaidDriveUsage_SPARE_ACTIVE

enum SB_REGISTER

This enumeration defines all supported SD/MMC registers.

Enumerator

SB_REGISTER_CID CID:	Card Identification.
SB_REGISTER_CSD CSD:	Card-Specific Data.
SB_REGISTER_SSR SSR:	SD Card Configuration.
SB_REGISTER_SCR SCR:	SD Status.
SB_REGISTER_EXT_CSD: Extended CSD:	MMC Extended Card-Specific Data.

enum SB_FLASH_CELL_MODE

This enumeration defines all supported flash cell modes.

Enumerator

SB_FLASH_STANDARD:	Native flash cell mode (SLC, MLC or TLC).
SB_FLASH_PSLC:	Pseudo SLC flash cell mode.
SB_FLASH_PMLC:	Pseudo MLC flash cell mode.

3.6 Defines

- #define SB_OK(x) ((x) == SB_ERROR_NONE)
  You can use this convenience macro to test for a non-error result.
- #define SB_SECTOR_LEN 512
  Sector length definition.
- #define SB_MAX_UNITS 32
  Maximum number of managed units on a device.
- #define SB_MAX_ERASE_INFOS 3

  Maximum number of different erase informations (depending on flash cell mode).
- #define LIBSBLTM_API LIBSBLTM_API_EX SBERROR
- #define LIBSBLTM_DEF_VERSION_MAJOR 3 Major version.
- #define LIBSBLTM_DEF_VERSION_MINOR 0 Minor version.
- #define LIBSBLTM_DEF_VERSION_BUILD 0 Build version.