Treadmill SD Card Framework API Description

Software Version	4.00
Document Version	PDF 2.00
Document Date	2015 04-10

Overview

Certain treadmills can be controlled by the insertion of a properly formatted Secure Digital (SD) memory card. This iOS Universal Framework allows the development, debugging, testing, and release of Apps that create treadmill compatible SD Cards. This single Universal Framework works for iPhone, iTouch, and iPad hardware running iOS 8 or newer. It also works for all Xcode 6 or newer simulators.

Be aware that the file produced by this Framework will correspond exactly to the segments entered into the workout. In other words, the application program should always insure the final segment of every workout specifies a speed and incline of zero.

The minimal correct usage of this Framework is:

```
openSdLibrary()
startSdWorkout()
addSegmentToSdWorkout()
addSegmentToSdWorkout()
addSegmentToSdWorkout()
...
finishSdWorkout()
closeSdLibrary()
```

Notes:

- While this Framework is written for, and used in, iPhone Apps - it is written in standard 'C', not Objective C.
- This Framework is intended to create workouts, not media productions. Hence, it does not support audio in the workout.

Functions In The Framework

uint32_t getSdLibraryVersion(void)

Description

Returns the version of the SD API being used.

Inputs

None

Outputs

None

Returned

The version of the API you are using.

uint32_t openSdLibrary(void)

Description

Initializes the SD Framework for use.

Part of this is malloc() of a buffer. This buffer maintains what will eventually become the workout file that goes on the SD Card. A pointer to this buffer is returned by finishSdWorkout(). The allocated memory is freed by closeSdWorkout(). The returned buffer should not be accessed after calling closeSdWorkout().

Inputs

None

Outputs

None

Returned

SdERR_NONE	No error detected
SdERR_LIBOPEN	The SD Framework is already open
SdERR_MALLOCFAILURE	An error was encountered while allocating memory

uint32_t closeSdLibrary(void)

Description

Closes the Framework and frees allocated memory.

WARNING:
Do not do this until you are finished using the buffer returned by finishSdWorkout()

Inputs

None

Outputs

None

Returned

SdERR_NONE	No error detected
SdERR_LIBCLOSED	The SD Framework is not open

uint32_t startSdWorkout(void)

Description

Starts the first, or perhaps another, SD workout. The SD Framework must be opened prior to starting a workout. The Framework need only be opened once, but each workout must be started (initialized) and finished. Finishing a workout involves some final calculations and adjustments. It is accomplished with a call to finishSdWorkout().

Inputs

None

Outputs

None

Returned

SdERR_NONE	No error detected
SdERR_LIBCLOSED	The SD Framework has not yet been opened, hence no workout can be started.
SdERR_WORKOUTACTIVE	A workout has already been started. It must be finished before another can be started

uint32_t addSegmentToSdWorkout(uint16_t time, float speed, float incline)

Description

Given a started workout, add a segment with the specified speed and incline at the specified time (which need not be monotonic). However, some workout segment, and it need not be the first specified, must start at a running time of zero (0). See SdERR_BADSTARTTIME under finishSdWorkout().

Note that duplicate start times are not allowed and return ERR_BADTIME

Inputs

time	The time at which the speed and incline values are to be applied. This is the running time from the start of the workout.
speed	The speed of this segment in Miles Per Hour
incline	The incline of this segment in degrees

Outputs

None

Returned

SdERR_NONE	No error detected
SdERR_BADTIME	The time was too big or a duplicate of an existing segment
SdERR_BADSPEED	An out of range speed was specified
SdERR_BADINCLINE	An out of range incline was specified
SdERR_LIBCLOSED	The SD API has not yet been opened
SdERR_WORKOUTTOOBIG	The workout has gotten too big for the allocated memory

uint32_t finishSdWorkout(uint32_t* fileSize, uint8_t** fileBuf)

Description

Perform final calculations and adjustments and return the size and data of the workout file that will be written to the SD Card.

Inputs

None

Outputs

fileSize	The caller provides a pointer to a 32 bit unsigned integer. If some error is returned, the 32 bit value is unspecified. If SdERR_NONE is returned, the 32 bit unsigned integer will contain the number of bytes of data in the returned fileBuf buffer
fileBuf	The caller provides a pointer to a "buffer pointer". If some error is returned, the content of the buffer pointer is unspecified. If SdERR_NONE is returned, the buffer pointer will be pointing to the start of the data that constitutes the SD Card workout file. Note on memory management: This function returns a buffer that was allocated when the SD Framework was opened via openSdLibrary(). The buffer will be freed by closeSdLibrary()

Returned

SdERR_NONE	No error detected
SdERR_LIBCLOSED	The SD API has not yet been opened, hence there is no workout to finish
SdERR_BADSTARTTIME	The earliest workout segment must start at time zero and that is not true in this workout.
SdERR_WORKOUTINVALID	There is no workout to finish because no workout has been started

uint32_t countSegmentsInSdWorkout(int* count)

Description

Return a count of the number of segments in a workout.

Inputs

None

Outputs

count	The caller provides a pointer to an integer. If some error is returned, the value is unspecified. If SdERR_NONE is returned, the integer will contain the number segments in the workout.

Returned

SdERR_NONE	No error detected
SdERR_LIBCLOSED	The SD API has not yet been opened.

uint32_t listSegmentsInSdWorkout(int maxSegments, int* count, WoSegment* segmentBuf)

Description

Return an array of structures that define the segments in the current workout
Note - The 16 bit aligned packed structure is defined as:

```
typedef struct
{
        uint16_t        time;
        float           speed;
        float           incline;
} WoSegment;
```

Inputs

maxSegments	Maximum segments for which there is room in the segment buffer. Note: Call countSegmentsInSdWorkout() first to make sure this is big enough.

Outputs

count	The caller provides a pointer to an integer. If some error is returned, the integer value is unspecified. If SdERR_NONE is returned, the integer will contain the number of segments in the returned segmentBuf buffer.
segmentBuf	The caller provides a pointer to an array of workout segments. If some error is returned, the content of the buffer is unspecified. If SdERR_NONE is returned, the array will be filled with "count" WoSegment. Note on memory management: It is the caller's responsibility to allocate and free this memory.

Returned

SdERR_NONE	No error detected
SdERR_LIBCLOSED	The SD API has not yet been opened.
SdERR_BADPOINTER	The segmentBuf pointer was a NULL pointer
SdERR_WORKOUTTOOBIG	The specified maxSegments is too small for this workout

uint32_t deleteSegmentByIndexFromSdWorkout(int segmentIndex)

Description

Delete the segment with the specified index from the workout.

Inputs

segmentIndex	The index of the segment to delete. Here the index is the position in the list of workout segments returned by listSegmentsInSdWorkout()

Outputs

None

Returned

SdERR_NONE	No error detected
SdERR_BADINDEX	No segment with the specified index can be found
SdERR_LIBCLOSED	The SD API has not yet been opened. No segment can be deleted
SdERR_WORKOUTEMPTY	The workout is already empty. No segment can be deleted

uint32_t deleteSegmentByTimeFromSdWorkout(uint16_t time)

Description

Delete the segment with the specified start time from the workout.

Inputs

time	The start time of the segment to delete

Outputs

None

Returned

SdERR_NONE	No error detected
SdERR_BADTIME	No segment with the specified time can be found
SdERR_LIBCLOSED	The SD API has not yet been opened. No segment can be deleted
SdERR_WORKOUTEMPTY	The workout is already empty. No segment can be deleted