OmegaESP32Services

v0.1

Generated by Doxygen 1.8.17

Chapter 1

OmegaESP32Services

This is a collection of software abstractions cannot be categorize as drivers (CoreDrivers, PeripheralDrivers) such as Hashing Operations, Fie System managers, System time (NTP servers).

> Speed optimized prebuilt version is available here. feel free to check it out report bug and issues for that repo. License agreement for open source version is available on that repo.

```
document pdf
```

Chapter 2

Class Index

2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 3

File Index

3.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 4

Class Documentation

4.1 OmegaFileData_t Struct Reference

Used to store information about, read bytes, read number of bytes, written bytes and file handle.

`#include <OmegaFileSystemController.h>`

Public Attributes

- uint8_t * in_out_buffer

  pointer to the file content that needs to be written or read
- size_t buffer_size

  size of the in_out_buffer
- size_t read_written_size

  size that was written or read
- FileHandle file_handle

  file handle that is related to the file system operation

4.1.1 Detailed Description

Used to store information about, read bytes, read number of bytes, written bytes and file handle.

The documentation for this struct was generated from the following file:

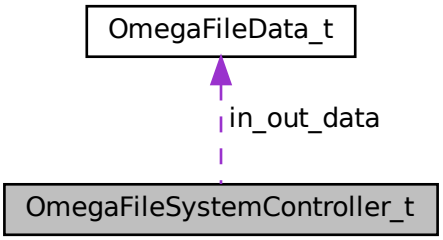
- OmegaFileSystemController.h

4.2 OmegaFileSystemController_t Struct Reference

FileSystemController instance that needs to be provided to use the controller API.

`#include <OmegaFileSystemController.h>`

Collaboration diagram for OmegaFileSystemController_t:



Public Attributes

- OmegaFileData_t in_out_data
  instance of `OmegaFileData_t` which will be used to store results of FileSystem Operations

4.2.1 Detailed Description

FileSystemController instance that needs to be provided to use the controller API.

The documentation for this struct was generated from the following file:

- OmegaFileSystemController.h

4.3 OmegaHashController_t Struct Reference

HashController instance that needs to be provided to use the Controller.

`#include <OmegaHashController.h>`

Public Attributes

- mbedtls_md_context_t ctx
  mbedtls context that will be used by the internal APIs to ingest and digest incoming data and/or data streams

4.3.1 Detailed Description

HashController instance that needs to be provided to use the Controller.

The documentation for this struct was generated from the following file:

- OmegaHashController.h

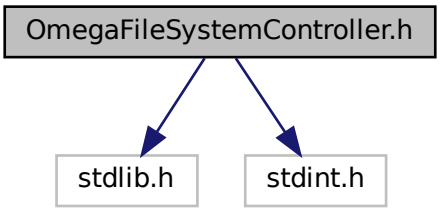
Chapter 5

File Documentation

5.1 OmegaFileSystemController.h File Reference

```
#include <stdlib.h>
#include <stdint.h>
```

Include dependency graph for OmegaFileSystemController.h:



Classes

- struct OmegaFileData_t

  Used to store information about, read bytes, read number of bytes, written bytes and file handle.
- struct OmegaFileSystemController_t

  FileSystemController instance that needs to be provided to use the controller API.

Typedefs

- typedef uint64_t FileHandle

  FileHandle that needs to be used to operate using FileSystemController.

Enumerations

- enum FileSystemControllerStatus {
  FSC_SUCCESS, FSC_FAILED, FSC_NOT_INIT, FSC_INVALID_OPENMODE,
  FSC_INVALID_PARAMETERS, FSC_FILE_NOT_EXIST, FSC_FILE_HANDLE_NOT_EXIST, FSC_NO_MEM,
  FSC_INCOMPLETE_FILE_WRITE, FSC_END_OF_FILE, FSC_FILE_ALREADY_OPENED, FSC_UNKNOWN
  }

  ReturnTypes/StatusCodes for the File System Controller.
- enum FileSystemOpenMode { READING = 1 << 0, WRITING = 1 << 1, APPEND = 1 << 2, OVERWRITE = 1 << 3 }

  FileSystemOpenMode(s) that needs to be specified when calling `OmegaFileSystemController_open_file()`
- enum FileSystemReadMode { READ_LINE, READ_CHUNK, READ_ALL }

  ReadMode(s) that needs to be specified when calling `OmegaFileSystemController_read_file()`

Functions

- FileSystemControllerStatus OmegaFileSystemController_init (OmegaFileSystemController_t *in_controller)

  Initialize and allocate required memory for the FileSystemController.
- FileSystemControllerStatus OmegaFileSystemController_deinit (OmegaFileSystemController_t *in_↩ controller)

  Used to free and deallocate all the resources that were allocated by `OmegaFileSystemController_init()`
- FileSystemControllerStatus OmegaFileSystemController_open_file (OmegaFileSystemController_t *in_↩ controller, const FileHandle *in_file_handle, const char *in_file_name, FileSystemOpenMode in_open_↩ mode)

  Used to open a file.
- FileSystemControllerStatus OmegaFileSystemController_close_file (OmegaFileSystemController_t *in_↩ controller, const FileHandle in_file_handle)

  Used to close a previously opened file.
- FileSystemControllerStatus OmegaFileSystemController_read_file (OmegaFileSystemController_t *in_↩ controller, const FileHandle in_file_handle, FileSystemReadMode in_read_mode, size_t in_size_to_read)

  Read from a previously opened file. Can be used on the same file multiple times depending on the `FileSystem↩ ReadMode` i.e. READ_LINE,READ_CHUNK.
- FileSystemControllerStatus OmegaFileSystemController_write_file (OmegaFileSystemController_t *in_↩ controller, const FileHandle in_file_handle, const uint8_t *in_buffer, const size_t in_size_to_write)

  Write a previously opened file.

5.1.1 Detailed Description

Author

Chameera Subasinghe

Date

Friday, 1st March 2024 2:28:17 am

Copyright

5.1.2 Enumeration Type Documentation

5.1.2.1 FileSystemControllerStatus

`enum FileSystemControllerStatus`

ReturnTypes/StatusCodes for the File System Controller.

Enumerator

FSC_SUCCESS	Indicates success in any operation related to FileSystemController.
FSC_FAILED	Indicates failure in any operation related to FileSystemController.
FSC_NOT_INIT	Indicate failure due to FileSystemController functions are being called before `OmegaFileSystemController_init()`
FSC_INVALID_OPENMODE	Indicates API misuse of using invalid `FileSystemOpenMode` in `OmegaFileSystemController_open_file()`
FSC_INVALID_PARAMETERS	Indicates failure fir to invalid parameters.
FSC_FILE_NOT_EXIST	Indicates a failure "File Not existing on the file system" when calling `OmegaFileSystemController_open_file()` with READING only.
FSC_FILE_HANDLE_NOT_EXIST	Indicates a API misuse of invalid parameter of type FileHandle being received to functions.
FSC_NO_MEM	Indicates failure due to not having enough heap memory needed to allocate for the operation.
FSC_INCOMPLETE_FILE_WRITE	Indicates a failure during file writing operation. All the content were no written to the file.
FSC_END_OF_FILE	Indicates the end of file reading a file.
FSC_FILE_ALREADY_OPENED	//!TODO: Implement the usage of this inside `OmegaFileSystemController_open_file()`
FSC_UNKNOWN	Indicates an unknown error.

5.1.2.2 FileSystemOpenMode

`enum FileSystemOpenMode`

FileSystemOpenMode(s) that needs to be specified when calling `OmegaFileSystemController_open_file()`

Possible combinations are, READING|WRITING|OVERWRITE READING|WRITING|APPEND READING WRITI↩
NG|OVERWRITE WRITING|APPEND

Enumerator

READING	dsfs
WRITING	sdfs
APPEND	sdfsd
OVERWRITE	sdfsd

5.1.2.3 FileSystemReadMode

```
enum FileSystemReadMode
```

ReadMode(s) that needs to be specified when calling `OmegaFileSystemController_read_file()`

Enumerator

READ_LINE	Reads set of characters till ' ' including the ' '. If EOF (End Of File) met just the content upto EOF is read.
READ_CHUNK	Read only pre-specified number of bytes.
READ_ALL	Read the whole file in 1 go.

5.1.3 Function Documentation

5.1.3.1 OmegaFileSystemController_close_file()

```
FileSystemControllerStatus OmegaFileSystemController_close_file (
            OmegaFileSystemController_t * in_controller,
            const FileHandle in_file_handle )
```

Used to close a previously opened file.

Parameters

in_controller	Input parameter. Pointer to an instance of `OmegaFileSystemController_t` that needs to be used to open up the file. Cannot be NULL.
in_file_handle	Input parameter. Instance of `FileHandle` which were used to open a file using `OmegaFileSystemController_open_file()`. Cannot be NULL.

Returns

FileSystemControllerStatus FSC_SUCCESS if the file closed successfully

5.1.3.2 OmegaFileSystemController_deinit()

```
FileSystemControllerStatus OmegaFileSystemController_deinit (
            OmegaFileSystemController_t * in_controller )
```

Used to free and deallocate all the resources that were allocated by `OmegaFileSystemController_init()`

Parameters

in_controller	Input parameter. Pointer to an instance of `OmegaFileSystemController_t` that needs to be de-initialized. Cannot be NULL.

Returns

FileSystemControllerStatus FSC_SUCCESS if the freeing of resources was successful

5.1.3.3 OmegaFileSystemController_init()

```
FileSystemControllerStatus OmegaFileSystemController_init (
            OmegaFileSystemController_t * in_controller )
```

Initialize and allocate required memory for the FileSystemController.

Parameters

in_controller	Input Parameter. Pointer to an instance of `OmegaFileSystemController_t` that needs to be initialized. Cannot be NULL

Returns

FileSystemControllerStatus FSC_SUCCESS if `OmegaFileSystemController_t` initialized successfully

5.1.3.4 OmegaFileSystemController_open_file()

```
FileSystemControllerStatus OmegaFileSystemController_open_file (
            OmegaFileSystemController_t * in_controller,
            const FileHandle * in_file_handle,
            const char * in_file_name,
            FileSystemOpenMode in_open_mode )
```

Used to open a file.

Parameters

in_controller	Input parameter. Pointer to an instance of `OmegaSystemController_t` that needs to be used to open up the file. Cannot be NULL. Prior to calling this function `OmegaFileSystemController_init()` needs to be called on this parameter
in_file_handle	Input parameter. Pointer to an instance of `FileHandle`. FileHandle needs to be used to do file operations (Read, Write, ...). Cannot be NULL.
in_file_name	Input parameter. Path of the file that needs to be opened with the filename and the extention of the file.
in_open_mode	Input parameter. Enumaration of type `FileSystemOpenMode`. Please see the brief of `FileSystemOpenMode`

Returns

FileSystemControllerStatus FSC_SUCCESS if the file opened successfully

5.1.3.5 OmegaFileSystemController_read_file()

```
FileSystemControllerStatus OmegaFileSystemController_read_file (
            OmegaFileSystemController_t * in_controller,
            const FileHandle in_file_handle,
            FileSystemReadMode in_read_mode,
            size_t in_size_to_read )
```

Read from a previously opened file. Can be used on the same file multiple times depending on the `FileSystem↩ ReadMode` i.e. READ_LINE,READ_CHUNK.

Parameters

in_controller	Input parameter. Pointer to an instance of `OmegaFileSystemController_t` that needs to be used to read the file. Cannot be NULL.
in_file_handle	Input parameter. Instance of `FileHandle` which were used to open a file using `OmegaFileSystemController_open_file()`. Cannot be NULL.
in_read_mode	Input parameter. Enumaration of type `FileSystemReadMode` that will help to use the API flexibly. i.e. read the whole file, read till a new line is found, read a predefined amount of bytes
in_size_to_read	Input parameter. If the `in_read_mode` is `READ_CHUNK` then this parameter is used to read that amount of bytes from the file, else this can be NULL, 0 or negative

Returns

FileSystemControllerStatus FSC_SUCCESS if the file read successfully

5.1.3.6 OmegaFileSystemController_write_file()

```
FileSystemControllerStatus OmegaFileSystemController_write_file (
            OmegaFileSystemController_t * in_controller,
            const FileHandle in_file_handle,
            const uint8_t * in_buffer,
            const size_t in_size_to_write )
```

Write a previously opened file.

Parameters

in_controller	Input parameter. Pointer to an instance of `OmegaFileSystemController_t` that needs to be used to write the file. Cannot be NULL.
in_file_handle	Input parameter. Instance of `FileHandle` which were used to open a file using `OmegaFileSystemController_open_file()`. Cannot be NULL.
in_buffer	Input parameter. Content of this will be used to write the file. Cannot be NULL
in_size_to_write	Input parameter. Amount of bytes that needs to be written from the `in_buffer`. Should be less or equal to the size of `in_buffer`

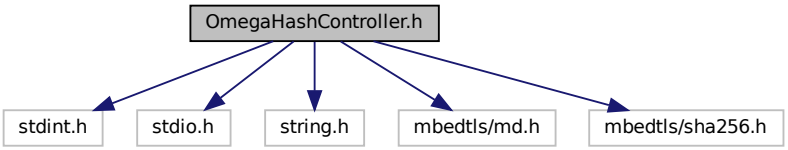
Returns

FileSystemControllerStatus FSC_SUCCESS if the filw written successfully

5.2 OmegaHashController.h File Reference

```
#include <stdint.h>
#include <stdio.h>
#include <string.h>
#include <mbedtls/md.h>
#include <mbedtls/sha256.h>
```

Include dependency graph for OmegaHashController.h:



Classes

- struct OmegaHashController_t

  HashController instance that needs to be provided to use the Controller.

Enumerations

- enum HashControllerStatus { HSC_SUCCESS, HSC_FAILED, HSC_INVALID_PARAMETERS, HSC_HASH_ALGO_NOT_FOUND, HSC_HASH_ALGO_NOT_SUPPORTED, HSC_NO_MEM, HSC_UNKNOWN }

  ReturnTypes/StatusCodes for the Hash Controller.
- enum HashAlgorithm { HASH256 }

  Hash Algorithms that are supported by Hash Controller.

Functions

- HashControllerStatus OmegaHashController_init (OmegaHashController_t *in_controller, HashAlgorithm in_hash_algorithm)

  Initialize and allocate required memory for the specified hash algorithm for the HashController instance.
- HashControllerStatus OmegaHashController_reset (OmegaHashController_t *in_controller)

  After a successful `OmegaHashController_ingest_data_streamed()` hashing operation, `OmegaHashController_t` needs to be reset before doing another hash operation. Purpose of this function is to reset all the internal variables, free old memory and allocate new memory. This function isn't required to be called if the hash operation was `OmegaHashController_ingest_data_single()`. This function will be called internally in `OmegaHashController_ingest_data_single()`
- HashControllerStatus OmegaHashController_ingest_data_single (OmegaHashController_t *in_controller, const uint8_t *in_buffer, const size_t in_buffer_size, const uint8_t *out_buffer)

If all the bytes needs to be hashed is known before hashing and/or system has enough heap/stack memory to allocate to all the bytes, This function can be called.

- HashControllerStatus OmegaHashController_ingest_data_streamed (OmegaHashController_t *in_controller, const uint8_t *in_buffer, const size_t in_buffer_size, const uint8_t *out_buffer)

  If all the bytes needs to be hashed is not known before hashing and/or system doesn't have enough heap/stack memory to allocate to all the bytes, This function can be called. To ingest the input data as well as retireve the final hash output this function is being used with providing some parameters as NULL.

- HashControllerStatus OmegaHashController_deinit (OmegaHashController_t *in_controller)

  used to free all the allocated resources

5.2.1 Detailed Description

Author

Chameera Subasinghe

Date

Friday, 1st March 2024 2:28:17 am

Copyright

Copyright 2024 - 2024 0m3g4ki113r, Xtronic

5.2.2 Enumeration Type Documentation

5.2.2.1 HashAlgorithm

`enum HashAlgorithm`

Hash Algorithms that are supported by Hash Controller.

Enumerator

HASH256	abstraction for `MBEDTLS_MD_SHA256` inside md.h `mbedtls_md_type_t`

5.2.2.2 HashControllerStatus

`enum HashControllerStatus`

ReturnTypes/StatusCodes for the Hash Controller.

Enumerator

HSC_SUCCESS	Indicates success in any operation related to HashController.
HSC_FAILED	Indicates failure in any operation related to HashController.
HSC_INVALID_PARAMETERS	Indicates failure due to invalid parameters provided to functions/sub-routines and/or structures.
HSC_HASH_ALGO_NOT_FOUND	Indicates that provided HashAlgorithm parameter in `OmegaHashController_init` is not valid.
HSC_HASH_ALGO_NOT_SUPPORTED	Indicates that API is trying to use unsupported hash algorithms. mbedtls supports many hash algorithms [described inside `mbedtls_md_type_t`]. But this controller only supports SHA256 as of 2024-03-03.
HSC_NO_MEM	Indicate that there is not enough Heap/Stack memory to allocate for the necessary operations.
HSC_UNKNOWN	Indicates Unknown error has occured.

5.2.3 Function Documentation

5.2.3.1 OmegaHashController_deinit()

```
HashControllerStatus OmegaHashController_deinit (
            OmegaHashController_t * in_controller )
```

used to free all the allocated resources

Parameters

in_controller	Input parameter. Instance of `OmegaHashController_t` that previously initialized.

Returns

HashControllerStatus HSC_SUCCESS if the freeing of resources was successful

5.2.3.2 OmegaHashController_ingest_data_single()

```
HashControllerStatus OmegaHashController_ingest_data_single (
            OmegaHashController_t * in_controller,
            const uint8_t * in_buffer,
            const size_t in_buffer_size,
            const uint8_t * out_buffer )
```

If all the bytes needs to be hashed is known before hashing and/or system has enough heap/stack memory to allocate to all the bytes, This function can be called.

Parameters

in_controller	Input parameter. Instance of the `OmegaHashController_t` that previously initialized/reset. Therefore this cannot be NULL
in_buffer	Input parameter. buffer that contains the bytes needs to be hashed. Cannot be NULL.
in_buffer_size	Input parameter. Size of the input buffer that was provided in the `in_buffer` parameter. Cannot be NULL, 0 or negative.
out_buffer	Output parameter. Result of the hash operation will be set in this byte buffer. This needs to be in the correct size.

Returns

HashControllerStatus HSC_SUCCESS if hash operation was successful.

5.2.3.3 OmegaHashController_ingest_data_streamed()

```
HashControllerStatus OmegaHashController_ingest_data_streamed (
            OmegaHashController_t * in_controller,
            const uint8_t * in_buffer,
            const size_t in_buffer_size,
            const uint8_t * out_buffer )
```

If all the bytes needs to be hashed is not known before hashing and/or system doesn't have enough heap/stack memory to allocate to all the bytes, This function can be called. To ingest the input data as well as retireve the final hash output this function is being used with providing some parameters as NULL.

Parameters

in_controller	Input parameter. Instance of the `OmegaHashController_t` that previously initialized/reset. Therefore this cannot be NULL
in_buffer	Input parameter. buffer that contains the bytes needs to be hashed. This cannot be NULL during data ingestion and can be NULL when retireving hashed data.
in_buffer_size	Input parameter. Size of the input buffer that was provided in the `in_buffer` parameter. This cannot be NULL during data ingestion and can be NULL, 0 or negative when retireving hashed data.
out_buffer	Output parameter. Result of the hash operation will be set in this byte buffer. This can be NULL during ingestion of data. Cannot be NULL when retireving hashed data.

Returns

HashControllerStatus HSC_SUCCESS if the hash operations [ingestion and digestion] was successful.

5.2.3.4 OmegaHashController_init()

```
HashControllerStatus OmegaHashController_init (
            OmegaHashController_t * in_controller,
            HashAlgorithm in_hash_algorithm )
```

Initialize and allocate required memory for the specified hash algorithm for the HashController instance.

Parameters

in_controller	Input parameter. Instance of the OmegaHashController_t that needs to be initialized. Cannot be NULL
in_hash_algorithm	Input parameter. Indicates the hash algorithm that is going to be used by this instance of HashController

Returns

HashControllerStatus HSC_SUCCESS if OmegaHashController_t initialized successfully.

5.2.3.5 OmegaHashController_reset()

```
HashControllerStatus OmegaHashController_reset (
            OmegaHashController_t * in_controller )
```

After a successful OmegaHashController_ingest_data_streamed() hashing operation, OmegaHashController_t needs to be reset before doing another hash operation. Purpose of this function is to reset all the internal variables, free old memory and allocate new memory. This function isn't required to be called if the hash operation was OmegaHashController_ingest_data_single(). This function will be called internally in OmegaHashController_ingest_data_single()

Parameters

in_controller	Input parameter. Instance of the OmegaHashController_t that needs to be reset

Returns

HashControllerStatus HSC_SUCCESS if OmegaHashController_t reset succesfully

Index