

OmegaESP32Services

v0.1

Generated by Doxygen 1.8.17

1 OmegaESP32Services	1
2 Class Index	3
2.1 Class List	3
3 File Index	5
3.1 File List	5
4 Class Documentation	7
4.1 OmegaFileData_t Struct Reference	7
4.1.1 Detailed Description	7
4.2 OmegaFileSystemController_t Struct Reference	8
4.2.1 Detailed Description	8
4.3 OmegaHashController_t Struct Reference	8
4.3.1 Detailed Description	8
5 File Documentation	9
5.1 OmegaFileSystemController.h File Reference	9
5.1.1 Detailed Description	10
5.1.2 Enumeration Type Documentation	11
5.1.2.1 FileSystemControllerStatus	11
5.1.2.2 FileSystemOpenMode	11
5.1.2.3 FileSystemReadMode	12
5.1.3 Function Documentation	12
5.1.3.1 OmegaFileSystemController_close_file()	12
5.1.3.2 OmegaFileSystemController_deinit()	12
5.1.3.3 OmegaFileSystemController_init()	13
5.1.3.4 OmegaFileSystemController_open_file()	13
5.1.3.5 OmegaFileSystemController_read_file()	14
5.1.3.6 OmegaFileSystemController_write_file()	14
5.2 OmegaHashController.h File Reference	15
5.2.1 Detailed Description	16
5.2.2 Enumeration Type Documentation	16
5.2.2.1 HashAlgorithm	16
5.2.2.2 HashControllerStatus	16
5.2.3 Function Documentation	17
5.2.3.1 OmegaHashController_deinit()	17
5.2.3.2 OmegaHashController_ingest_data_single()	17
5.2.3.3 OmegaHashController_ingest_data_streamed()	18
5.2.3.4 OmegaHashController_init()	18
5.2.3.5 OmegaHashController_reset()	19
Index	21

Chapter 1

OmegaESP32Services

This is a collection of software abstractions cannot be categorize as drivers (CoreDrivers, PeripheralDrivers) such as Hashing Operations, File 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:

OmegaFileData_t	Used to store information about, read bytes, read number of bytes, written bytes and file handle	7
OmegaFileSystemController_t	FileSystemController instance that needs to be provided to use the controller API	8
OmegaHashController_t	HashController instance that needs to be provided to use the Controller	8

Chapter 3

File Index

3.1 File List

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

OmegaFileSystemController.h	9
OmegaHashController.h	15

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:

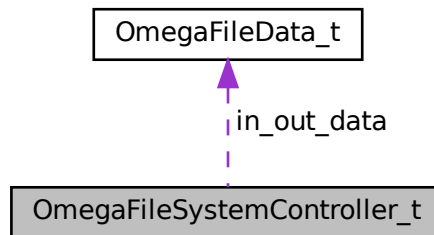
- [OmegaFileSystemController.h](#)

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](#)

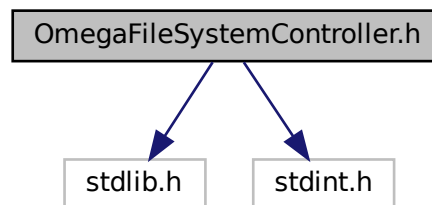
Chapter 5

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 `FileSystemReadMode` 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

Copyright 2024 - 2024 0m3g4ki113r, Xtronic

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 <code>OmegaFileSystemController_init()</code>
FSC_INVALID_OPENMODE	Indicates API misuse of using invalid <code>FileSystemOpenMode</code> in <code>OmegaFileSystemController_open_file()</code>
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 <code>OmegaFileSystemController_open_file()</code> with READING only.
FSC_FILE_HANDLE_NOT_EXIST	Indicates a API misuse of invalid parameter of type <code>FileHandle</code> 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	/// <code>TODO</code> : Implement the usage of this inside <code>OmegaFileSystemController_open_file()</code>
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 WRITING|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

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

<i>in_controller</i>	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

<i>in_controller</i>	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

<i>in_controller</i>	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
<i>in_file_handle</i>	Input parameter. Pointer to an instance of FileHandle . FileHandle needs to be used to do file operations (Read, Write, ...). Cannot be NULL.
<i>in_file_name</i>	Input parameter. Path of the file that needs to be opened with the filename and the extension of the file.
<i>in_open_mode</i>	Input parameter. Enumeration 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 FileSystemReadMode i.e. READ_LINE, READ_CHUNK.

Parameters

<i>in_controller</i>	Input parameter. Pointer to an instance of OmegaFileSystemController_t that needs to be used to read the file. Cannot be NULL.
<i>in_file_handle</i>	Input parameter. Instance of FileHandle which were used to open a file using OmegaFileSystemController_open_file() . Cannot be NULL.
<i>in_read_mode</i>	Input parameter. Enumeration 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
<i>in_size_to_read</i>	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

<i>in_controller</i>	Input parameter. Pointer to an instance of OmegaFileSystemController_t that needs to be used to write the file. Cannot be NULL.
<i>in_file_handle</i>	Input parameter. Instance of FileHandle which were used to open a file using OmegaFileSystemController_open_file() . Cannot be NULL.
<i>in_buffer</i>	Input parameter. Content of this will be used to write the file. Cannot be NULL
<i>in_size_to_write</i>	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

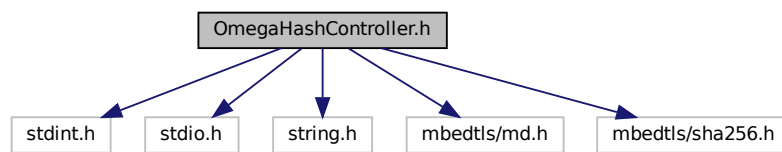
Returns

FileSystemControllerStatus FSC_SUCCESS if the file 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 <code>OmegaHashController_init</code> is not valid.
HSC_HASH_ALGO_NOT_SUPPORTED	Indicates that API is trying to use unsupported hash algorithms. <code>mbedtls</code> supports many hash algorithms [described inside <code>mbedtls_md_type_t</code>]. 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 occurred.

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

<i>in_controller</i>	Input parameter. Instance of <code>OmegaHashController_t</code> 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

<i>in_controller</i>	Input parameter. Instance of the OmegaHashController_t that previously initialized/reset. Therefore this cannot be NULL
<i>in_buffer</i>	Input parameter. buffer that contains the bytes needs to be hashed. Cannot be NULL.
<i>in_buffer_size</i>	Input parameter. Size of the input buffer that was provided in the <i>in_buffer</i> parameter. Cannot be NULL, 0 or negative.
<i>out_buffer</i>	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

<i>in_controller</i>	Input parameter. Instance of the OmegaHashController_t that previously initialized/reset. Therefore this cannot be NULL
<i>in_buffer</i>	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.
<i>in_buffer_size</i>	Input parameter. Size of the input buffer that was provided in the <i>in_buffer</i> parameter. This cannot be NULL during data ingestion and can be NULL, 0 or negative when retireving hashed data.
<i>out_buffer</i>	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

<i>in_controller</i>	Input parameter. Instance of the OmegaHashController_t that needs to be initialized. Cannot be <code>NULL</code>
<i>in_hash_algorithm</i>	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

<i>in_controller</i>	Input parameter. Instance of the OmegaHashController_t that needs to be reset
----------------------	---

Returns

HashControllerStatus HSC_SUCCESS if [OmegaHashController_t](#) reset successfully

Index

APPEND

OmegaFileSystemController.h, [11](#)

FileSystemControllerStatus

OmegaFileSystemController.h, [11](#)

FileSystemOpenMode

OmegaFileSystemController.h, [11](#)

FileSystemReadMode

OmegaFileSystemController.h, [12](#)

FSC_END_OF_FILE

OmegaFileSystemController.h, [11](#)

FSC_FAILED

OmegaFileSystemController.h, [11](#)

FSC_FILE_ALREADY_OPENED

OmegaFileSystemController.h, [11](#)

FSC_FILE_HANDLE_NOT_EXIST

OmegaFileSystemController.h, [11](#)

FSC_FILE_NOT_EXIST

OmegaFileSystemController.h, [11](#)

FSC_INCOMPLETE_FILE_WRITE

OmegaFileSystemController.h, [11](#)

FSC_INVALID_OPENMODE

OmegaFileSystemController.h, [11](#)

FSC_INVALID_PARAMETERS

OmegaFileSystemController.h, [11](#)

FSC_NO_MEM

OmegaFileSystemController.h, [11](#)

FSC_NOT_INIT

OmegaFileSystemController.h, [11](#)

FSC_SUCCESS

OmegaFileSystemController.h, [11](#)

FSC_UNKNOWN

OmegaFileSystemController.h, [11](#)

HASH256

OmegaHashController.h, [16](#)

HashAlgorithm

OmegaHashController.h, [16](#)

HashControllerStatus

OmegaHashController.h, [16](#)

HSC_FAILED

OmegaHashController.h, [17](#)

HSC_HASH_ALGO_NOT_FOUND

OmegaHashController.h, [17](#)

HSC_HASH_ALGO_NOT_SUPPORTED

OmegaHashController.h, [17](#)

HSC_INVALID_PARAMETERS

OmegaHashController.h, [17](#)

HSC_NO_MEM

OmegaHashController.h, [17](#)

HSC_SUCCESS

OmegaHashController.h, [17](#)

HSC_UNKNOWN

OmegaHashController.h, [17](#)

OmegaFileData_t, [7](#)

OmegaFileSystemController.h, [9](#)

APPEND, [11](#)

FileSystemControllerStatus, [11](#)

FileSystemOpenMode, [11](#)

FileSystemReadMode, [12](#)

FSC_END_OF_FILE, [11](#)

FSC_FAILED, [11](#)

FSC_FILE_ALREADY_OPENED, [11](#)

FSC_FILE_HANDLE_NOT_EXIST, [11](#)

FSC_FILE_NOT_EXIST, [11](#)

FSC_INCOMPLETE_FILE_WRITE, [11](#)

FSC_INVALID_OPENMODE, [11](#)

FSC_INVALID_PARAMETERS, [11](#)

FSC_NO_MEM, [11](#)

FSC_NOT_INIT, [11](#)

FSC_SUCCESS, [11](#)

FSC_UNKNOWN, [11](#)

OmegaFileSystemController_close_file, [12](#)

OmegaFileSystemController_deinit, [12](#)

OmegaFileSystemController_init, [13](#)

OmegaFileSystemController_open_file, [13](#)

OmegaFileSystemController_read_file, [14](#)

OmegaFileSystemController_write_file, [14](#)

OVERWRITE, [11](#)

READ_ALL, [12](#)

READ_CHUNK, [12](#)

READ_LINE, [12](#)

READING, [11](#)

WRITING, [11](#)

OmegaFileSystemController_close_file

OmegaFileSystemController.h, [12](#)

OmegaFileSystemController_deinit

OmegaFileSystemController.h, [12](#)

OmegaFileSystemController_init

OmegaFileSystemController.h, [13](#)

OmegaFileSystemController_open_file

OmegaFileSystemController.h, [13](#)

OmegaFileSystemController_read_file

OmegaFileSystemController.h, [14](#)

OmegaFileSystemController_t, [8](#)

OmegaFileSystemController_write_file

OmegaFileSystemController.h, [14](#)

OmegaHashController.h, [15](#)

HASH256, [16](#)

- HashAlgorithm, [16](#)
- HashControllerStatus, [16](#)
- HSC_FAILED, [17](#)
- HSC_HASH_ALGO_NOT_FOUND, [17](#)
- HSC_HASH_ALGO_NOT_SUPPORTED, [17](#)
- HSC_INVALID_PARAMETERS, [17](#)
- HSC_NO_MEM, [17](#)
- HSC_SUCCESS, [17](#)
- HSC_UNKNOWN, [17](#)
- OmegaHashController_deinit, [17](#)
- OmegaHashController_ingest_data_single, [17](#)
- OmegaHashController_ingest_data_streamed, [18](#)
- OmegaHashController_init, [18](#)
- OmegaHashController_reset, [19](#)
- OmegaHashController_deinit
 - OmegaHashController.h, [17](#)
- OmegaHashController_ingest_data_single
 - OmegaHashController.h, [17](#)
- OmegaHashController_ingest_data_streamed
 - OmegaHashController.h, [18](#)
- OmegaHashController_init
 - OmegaHashController.h, [18](#)
- OmegaHashController_reset
 - OmegaHashController.h, [19](#)
- OmegaHashController_t, [8](#)
- OVERWRITE
 - OmegaFileSystemController.h, [11](#)
- READ_ALL
 - OmegaFileSystemController.h, [12](#)
- READ_CHUNK
 - OmegaFileSystemController.h, [12](#)
- READ_LINE
 - OmegaFileSystemController.h, [12](#)
- READING
 - OmegaFileSystemController.h, [11](#)
- WRITING
 - OmegaFileSystemController.h, [11](#)