



embedded-os.de  
a little world of RTOS and data-communication protocols

# *pC/SFS Reference*

## *V1.01b*

### Haftungsausschluß

Der Autor übernimmt keinerlei Haftung für durch diesen Code entstandene oder entstehende Schäden an Hard- und Software. Er versichert lediglich, daß er den Code vielfältigen Test's auf unterschiedlicher Hardware unterzogen hat, um seinerseits keine Fehler bestehen zu wissen. Sollten dennoch Fehler auftauchen oder Vorschläge zur Verbesserung des Codes an den Autor weitergegeben werden, so ist dieser bestrebt, Fehler schnellstmöglich auszumerzen oder Vorschläge einzuarbeiten.

### liability exclusion

The author takes over no liability for through this code originated or emerging damages to hardware and software. He assures merely that he subjected the code of diverse tests on different hardware, about for his part no mistakes to know exists. Mistakes nevertheless should appear or suggestions are passed on at the author to the improvement of the code, so this is striving, mistakes fastest to wipe out or to incorporate suggestions.

The (small) Serial-File-System is based on simple concatenated blocks whose elements (DIR / FILE) are referenced by a 32bit hash of their name and where the number of max. entries per directory must be defined.

By using a 32bit hash as a name, the file system can be massively simplified, but there are therefore no references to the names of the entries. The hash of each entry must be unique within its directory. Since a 32bit hash value can not be guaranteed to be unique in itself - that is, if two names have the same hash value - several 32bit hash algorithms are usable, but the selection must be set at compile time.

This file-system handles all names of directories/files as hash over a STRING, so special characters can be used in the names.

reserved names-elements:

- .. - one directory back
- / - on the beginning of the Path: for from ROOT
- ./ - on the beginning of the Path: for from current DIR (optional)
- / - in middle of the Path: as separation for directory/file-names

The whole file-system works case-sensitive !

Various serial NVM memory devices (SPI & I2C) are tested as hardware (MRAM, FRAM, ReRAM, EEPROM). Additional the file system can run too on parallel memory like RAM, FRAM, MRAM or EEPROM. Ideal are all types that support byte-wise writing by internal buffering of sectors/pages, but this can also be done by the low-level hardware driver.

The use of serial flash memory was not provided due to the sector size of  $\geq 64\text{kB}$ , the necessary buffering of such an update, as well as the long programming time of an entire page. When using EEPROM, it is important to remember that 1.000.000 cycles are already quite a lot, but this also indicates a finite lifetime, In addition, the SerialFileSystem do not respect hot spots (high-update files) or transactions.

### User-Functions:

<b>SFS-Control:</b>	Description
<a href="#">SFS_Init</a>	Initialization of the File-System
<a href="#">SFS_GetRev</a>	It returns a pointer on SFS revision
<a href="#">SFS_Flush</a>	save the SFS as image into a Windows/LINUX-file
<a href="#">SFS_Format</a>	Format the drive

<b>User:</b>	Description
<a href="#">SFS_BecomeUser</a>	As User announce
<a href="#">SFS_CloseUser</a>	As User cancel

<b>Directories:</b>	Description
<a href="#">SFS_CreateDir</a>	Create a directory
<a href="#">SFS_RemoveDir</a>	Remove a directory
<a href="#">SFS_RenameDir</a>	Rename a directory
<a href="#">SFS_ChangeDir</a>	Change current path
<a href="#">SFS_ChangeDirTemp</a>	Change current path temporary
<a href="#">SFS_BackDirTemp</a>	returns from temporary path

<b>Files:</b>	Description
<a href="#">SFS_CreateFile</a>	Create a file
<a href="#">SFS_RemoveFile</a>	Remove a file
<a href="#">SFS_RenameFile</a>	Rename a file

<a href="#">SFS_AttribFile</a>	Change the attributes of a file
<a href="#">SFS_GetFileAttrib</a>	It returns the attributes of a file
<a href="#">SFS_GetFileSize</a>	It returns the current size of a file
<a href="#">SFS_GetMaxFileSize</a>	It returns the max size of a file
<a href="#">SFS_OpenFile</a>	Open a file in "mode"
<a href="#">SFS_CloseFile</a>	Close the opened file
<a href="#">SFS_SeekFile</a>	Place the pointer in opened file absolutely
<a href="#">SFS_TellFile</a>	It returns the pointer in opened file
<a href="#">SFS_SetEOF</a>	Set EndOfFile in opened file to actual R/W-pointer
<a href="#">SFS_ReadFile</a>	Read data from opened file
<a href="#">SFS_WriteFile</a>	Write data into opened file
<a href="#">SFS_GetErrNo</a>	read the error-code from Open, Read, Write ...

<b>Links:</b>	Description
<a href="#">SFS_CreateLink</a>	Create a Link to a directory or file
<a href="#">SFS_RemoveLink</a>	Remove a Link

<b>Entries:</b>	Description
<a href="#">SFS_GetEntry</a>	It returns all infos of the given entry (dir/file/link) into a struct

**Error-Codes:**

Name	Decimal_Value	Description
SFS_NO_ERR	0	no errors
SFS_USR_OVF	200	no user free
SFS_DBL_USER	201	double user
SFS_NO_USER	202	not a valid user
SFS_NAME_EXIST	210	name of entry exist in this DIR
SFS_NOT_EXIST	211	DIR or FILE not exist
SFS_PATH_ERR	212	error on PATH
SFS_TMP_DIR	213	Temp-Dir is still used / not set
SFS_NO_FILE	214	error on PATH / no FILE-Name given
SFS_FILE_RO	215	file to open for writing is read-only
SFS_FILE_WO	216	file to open for reading is write-only
SFS_FILE_EOF	217	end-of-file while reading or writing
SFS_NOT_EMPTY	218	entry not empty
SFS_FILE_OPEN	219	current user have a opened file
SFS_NO_DATA	220	current file-length is zero
SFS_WRONG_PTR	221	offset into open file is wrong (or size for R/W)
SFS_NO_ENTRY	222	no free entry in directory
SFS_WRONG_A	223	wrong access flags or attributes given
SFS_LINKED	230	entry is linked
SFS_MAX_LINK	231	entry is max count linked
SFS_LINK_ERR	232	error in link-mechanism
SFS_NO_LINK	233	no link to an entry in link-entry
SFS_MEM_ERR	240	error in block-memory manager
SFS_MEM_OVF	241	memory overflow
SFS_FORMAT_ERR	250	error in found format or during formatting
SFS_PORT_ERR	251	error in HW-port

---

## Configuration of the File-System

---

The pC/SFS File-System can be configured in addition to the to-use SPI, I2C or parallel memory type some numbers of ways to configure services as well as to reduce the memory requirements - code-size for the compilers "unused code" may not clearly identify - available. These are in the file "SFS\_cfg.h" together.

<b>user configuration</b>	description
SFS_MAX_USER	max users (tasks)
SFS_HANDLES	max files opened by a user (task)
SFS_AUTO_FORMAT	auto-format during init if no valid ROOT-entry was found
SFS_TempDIR	use the one-level temporary current-dir feature
SFS_LINKS	create & delete of links supported

<b>internal configuration</b>	description
SFS_BLOCK_SIZE	bytes per managed block
SFS_ENTRIES_PER_DIR	entries per dir a 16 byte (one is lost for "..")
SFS_MEM_....	exact type of used SPI, I2C or parallel memory device (to config the LLdriver)

If `SFS_LINKS` is not set, no links can be created or deleted, and the linked entries (DIR/FILE) can not be deleted. However, links contained in the file system can be processed completely otherwise.

If the file system found during the initialization is smaller or differently configured but compatible (HW-compatible prerequisite), its settings are accepted and work can be done with this file system.

## General

---

### SFS\_Init

U08 SFS\_Init(void)

Initialize the File-System and installs on use of a RTOS the required mutex/semaphore. If you using the Windows/Linux-HOST Port, a Windows/LINUX Image-file will be loaded first. If the system should be recognized as unformatted or incompatible, so this is executed if the config-switch `SFS_AUTO_FORMAT` is set. This function must be called before all other file services at the system initialization once.

#### Parameters

none
------

#### Return Value

SFS_NO_ERR	file system initialised
SFS_MEM_ERR	Mistakes in the memory management
SFS_FORMAT_ERR	format unknown or incompatible
SFS_FILE_OPEN	on <code>SFS_AUTO_FORMAT</code> : a file is opened by a user

#### Example

```
void main(void)
{
    U08 returnOk;

    OS_Init();
    .
    .
    returnOk=SFS_Init();
    .
    .
    OS_Start();
}
```

---

## SFS\_GetRev

```
void SFS_GetRev(U08 OS_HUGE **pointer)
```

It returns a pointer on the SFS-revision (NULL-terminated ASCII-array).

### Parameters

**pointer	pointer to pointer will get the address of array
-----------	--------------------------------------------------

### Return Value

none
------

### Example

```
void OS_FAR Task1(void *data)
{
    U08 OS_HUGE *Revision;

    .
    .
    while(1)
    {
        .
        SFS_GetRev(&Revision);
        .
    }
}
```

---

## **SFS\_Flush**

S32 SFS\_Flush(void)

*Only by using the Windows or Linux HOSTs  
Saves the filesystem as IMAGE into a Windows/Linux-file.*

### **Parameters**

<i>none</i>
-------------

### **Return Value**

<i>SFS_NO_ERR</i>	<i>Laufwerk gesichert</i>
<i>from Windows/LINUX</i>	<i>see Windows/LINUX</i>

### **Example**

```
void main(void)
{
    U08  returnOk;

    .
    returnOk=SFS_Init();
    .
    .
    returnOk=SFS_Flush();
}
```

---



## SFS\_Format

U08 SFS\_Format(void)

Formats the SFS-Drive and writes down the ROOT-Entry. At this time no user may be known.

### Parameters

none
------

### Return Value

SFS_NO_ERR	Drive formatted
SFS_USER	at minimum one user is known
SFS_MEM_ERR	Mistakes in the memory management
SFS_MEM_OVF	File-system too small for ROOT-Entry

### Example

```
void OS_FAR Task1(void *data)
{
    U08 returnOk;
    .
    .
    while(1)
    {
        .
        returnOk=SFS_Format();
        .
        .
    }
}
```

---

## SFS\_BecomeUser

U08 SFS\_BecomeUser (SFS\_USER OS\_HUGE \*SFSUser)

It creates a new User.

This function initializes the User-Control-Block and writes down the new user into the internal list. Every Task need only once to register, after this every User can handle SFS\_HANDLES files. After registration of an user can call this drive and file accesses.

### Parameters

*SFSUser	pointer to user-control-block
----------	-------------------------------

### Return Value

SFS_NO_ERR	User successfully created
SFS_DBL_USER	this User already is announced
SFS_USR_OVF	already SFS_maxUSER are announced

### Example

```
SFS_USER SFS_User1;

void OS_FAR Task1(void *data)
{
    U08 returnOk;

    .
    while(1)
    {
        .
        returnOk=SFS_BecomeUser (&SFS_User1);
        .
    }
}
```

---

## SFS\_CloseUser

U08 SFS\_CloseUser(void)

It deletes a registered user from the internal list. This user must be announced again for it before accesses to the drive become again.

### Parameters

none
------

### Return Value

SFS_NO_ERR	User successfully deleted
SFS_FILE_OPEN	User opened a file currently
SFS_NO_USER	User unknown

### Example

```
SFS_USER  SFS_User1;

void OS_FAR Task1(void *data)
{
    U08  returnOk;

    .
    while(1)
    {
        .
        returnOk=SFS_BecomeUser (&SFS_User1);
        .
        .
        returnOk=SFS_CloseUser ();
        .
    }
}
```

---

## Directory and File-Handlings

---

### SFS\_CreateDir

U08 SFS\_CreateDir(U08 OS\_HUGE \*Name)

Creates a new directory in the current or handed over path. The path statement can absolutely or relatively take place on that occasion.

#### Parameters

*Name	Directory-name [with path]
-------	----------------------------

#### Return Value

SFS_NO_ERR	Directory created
SFS_NO_USER	User unknown
SFS_PATH_ERR	an element of the path statement doesn't exist
SFS_NAME_EXIST	a directory with same name exists already in this directory
SFS_MEM_ERR	Mistakes in the memory management
SFS_NO_ENTRY	directory full

#### Example

```
SFS_USER  SFS_User1;

void OS_FAR Task1(void *data)
{
    U08  returnOk;

    .
    while(1)
    {
        .
        returnOk=SFS_BecomeUser(&SFS_User1);
        .
        .
        returnOk=SFS_CreateDir("/usr");           // from ROOT
        .
        returnOk=SFS_CreateDir("./demo/test");    // from current directory
        .
        returnOk=SFS_CreateDir("../local/test.src"); // from one level back
        .
        returnOk=SFS_CreateDir("config.save");    // in current directory
        .
        .
        returnOk=SFS_CloseUser();
        .
    }
}
```

---

## SFS\_RemoveDir

U08 SFS\_RemoveDir(U08 OS\_HUGE \*Name)

Removes the directory in the current or handed over path. The path statement can absolutely or relatively take place on that occasion. That to deleting directory must be included empty and no link may point this entry.

### Parameters

*Name	Directory-name [with path]
-------	----------------------------

### Return Value

SFS_NO_ERR	Directory deleted
SFS_NO_USER	User unknown
SFS_PATH_ERR	an element of the path statement doesn't exist
SFS_NOT_EMPTY	the directory is not empty
SFS_LINKED	the directory is linked from another entry
SFS_TMP_DIR	the directory is the current directory of a user
SFS_MEM_ERR	Mistakes in the memory management

### Example

```
SFS_USER  SFS_User1;

void OS_FAR Task1(void *data)
{
    U08  returnOk;

    .
    while(1)
    {
        .
        returnOk=SFS_BecomeUser (&SFS_User1);
        .
        .
        returnOk=SFS_RemoveDir ("config.save");    // in actual directory
        .
        .
    }
}
```

---

## SFS\_RenameDir

U08 SFS\_RenameDir(U08 OS\_HUGE \*Name, U08 OS\_HUGE \*NewName)

Changes the name of directory in the current or handed over path. The path statement can absolutely or relatively take place on that occasion. A Path in the new name will be ignored, so the directory can't be moved !

### Parameters

*Name	Directory-name [with path]
*NewName	new Directory-name (a path will ignored)

### Return Value

SFS_NO_ERR	directory renamed
SFS_NO_USER	user unknown
SFS_PATH_ERR	an element of the path statement doesn't exist
SFS_NAME_EXIST	a entry with same name exists already in this directory
SFS_NOT_EXIST	the directory doesn't exist
SFS_MEM_ERR	mistakes in the memory management

### Example

```
SFS_USER  SFS_User1;

void OS_FAR Task1(void *data)
{
    U08  returnOk;

    .
    while(1)
    {
        .
        returnOk=SFS_BecomeUser (&SFS_User1);
        .
        .
        returnOk=SFS_RenameDir ("/usr", "user");    // in ROOT
        .
        .
    }
}
```

---

## SFS\_ChangeDir

U08 SFS\_ChangeDir(U08 OS\_HUGE \*Name)

Changes the current directory. The path statement can absolutely or relatively take place on that occasion.

### Parameters

*Name	Directory-name [with path]
-------	----------------------------

### Return Value

SFS_NO_ERR	directory changed
SFS_NO_USER	user unknown
SFS_PATH_ERR	an element of the path statement doesn't exist
SFS_NOT_EXIST	the directory doesn't exist

### Example

```
SFS_USER  SFS_User1;

void OS_FAR Task1(void *data)
{
    U08  returnOk;

    .
    while(1)
    {
        .
        returnOk=SFS_BecomeUser(&SFS_User1);
        .
        .
        returnOk=SFS_ChangeDir("../test.src");
        .
        .
        .
    }
}
```

---

## SFS\_ChangeDirTemp

U08 SFS\_ChangeDirTemp(U08 OS\_HUGE \*Name)

Changes the current directory temporary. The path statement can absolutely or relatively take place on that occasion. The current directory up to this call will be registered internally. This feature can only be used one level per user.

### Parameters

*Name	Directory-name [with path]
-------	----------------------------

### Return Value

SFS_NO_ERR	dDirectory changed
SFS_NO_USER	user unknown
SFS_TMP_DIR	Temp-Dir is still used
SFS_PATH_ERR	an element of the path statement doesn't exist
SFS_NOT_EXIST	the directory doesn't exist

### Example

```
SFS_USER  SFS_User1;

void OS_FAR Task1(void *data)
{
    U08  returnOk;

    .
    while(1)
    {
        .
        returnOk=SFS_BecomeUser(&SFS_User1);
        .
        .
        returnOk=SFS_ChangeDirTemp("../userfiles");
        if(returnOk == SFS_NO_ERR) {
            .
            .
            do {
                .
                .
                .
            } while(returnOk == SFS_NO_ERR);
            .
            returnOk=SFS_BackDirTemp();
        }
        .
        .
    }
}
```

---



## SFS\_BackDirTemp

U08 SFS\_BackDirTemp(void)

Returns from the temporary directory to the registered directory from SFS\_ChangeDirTemp().

### Parameters

none
------

### Return Value

SFS_NO_ERR	directory changed
SFS_NO_USER	user unknown
SFS_TMP_DIR	Temp-Dir is not set

### Example

```
SFS_USER  SFS_User1;

void OS_FAR Task1(void *data)
{
    U08  returnOk;

    .
    while(1)
    {
        .
        returnOk=SFS_BecomeUser(&SFS_User1);
        .
        .
        returnOk=SFS_ChangeDirTemp("../userfiles");
        if(returnOk == SFS_NO_ERR) {
            .
            .
            do {
                .
                .
                .
            } while(returnOk == SFS_NO_ERR);
            .
            returnOk=SFS_BackDirTemp();
        }
        .
        .
    }
}
```

---

## Directory and File-Handlings

---

### SFS\_CreateFile

U08 SFS\_CreateFile(U08 OS\_HUGE \*Name, SFS\_ATTR Attr, SFS\_LONG size)

Creates a new file in the current or handed over path in stated size. The path statement can absolutely or relatively take place on that occasion. As attributes, ReadOnly or WriteOnly can be declared. ATTENTION! Files don't possess any type in this system.

#### Parameters

*Name	File-name [with path]
Attr	Attributes of this file
size	max size of file in bytes

#### Return Value

SFS_NO_ERR	File created
SFS_NO_USER	User unknown
SFS_PATH_ERR	an element of the path statement doesn't exist
SFS_NAME_EXIST	a file with same name exists already in this directory
SFS_WRONG_A	wrong file attributes
SFS_MEM_ERR	Mistakes in the memory management
SFS_MEM_OVF	Drive full

#### Example

```
SFS_USER  SFS_User1;

void OS_FAR Task1(void *data)
{
    U08  returnOk;

    .
    while(1)
    {
        .
        returnOk=SFS_BecomeUser(&SFS_User1);
        .
        .
        returnOk=SFS_CreateFile("/file1", SFS_ATTR_RO, 100);
                                     // from ROOT
        .
        returnOk=SFS_CreateFile("../test.src/main.c", SFS_ATTR_RW, 350);
                                     // from one level back
        .
        returnOk=SFS_CreateFile("makefile.mak", SFS_ATTR_WO, 140);
                                     // in actual directory
        .
        .
    }
}
```

---



## SFS\_RemoveFile

U08 SFS\_RemoveFile(U08 OS\_HUGE \*Name)

Deletes the file in the current or handed over path. The path statement can absolutely or relatively take place on that occasion. The to deleting file cannot be opened by any other user on that occasion and no link may point this entry.

### Parameters

*Name	File-name [with path]
-------	-----------------------

### Return Value

SFS_NO_ERR	File deleted
SFS_NO_USER	User unknown
SFS_FILE_OPEN	User, itself or other, this file opened currently
SFS_PATH_ERR	an element of the path statement doesn't exist
SFS_LINKED	the file is linked from another entry
SFS_MEM_ERR	Mistakes in the memory management

### Example

```
SFS_USER  SFS_User1;

void OS_FAR Task1(void *data)
{
    U08  returnOk;

    .
    while(1)
    {
        .
        returnOk=SFS_BecomeUser (&SFS_User1);
        .
        .
        returnOk=SFS_RemoveFile("makefile.mak");    // in actual directory
        .
        .
    }
}
```

---

## SFS\_RenameFile

```
U08 SFS_RenameFile(U08 OS_HUGE *OldName, U08 OS_HUGE *NewName)
```

Changes the name of file in the current or handed over path. The path statement can absolutely or relatively take place on that occasion. The to changing file cannot be opened by any other user on that occasion. A Path in the new name will be cut, so the file can't be moved !

### Parameters

*OldName	File-name [with path]
*NewName	new File-name (a path will ignored)

### Return Value

SFS_NO_ERR	File name changed
SFS_NO_USER	User unknown
SFS_NO_FILE	no file name in the path or as new name given
SFS_FILE_OPEN	User, itself or other, this file opened currently
SFS_NAME_EXIST	a file with same name exists already in this directory
SFS_NOT_EXIST	File doesn't exist in this directory
SFS_PATH_ERR	an element of the path statement doesn't exist

### Example

```
SFS_USER  SFS_User1;

void OS_FAR Task1(void *data)
{
    U08  returnOk;

    .
    while(1)
    {
        .
        returnOk=SFS_BecomeUser (&SFS_User1);
        .
        .
        returnOk=SFS_RenameFile("/test.src/main.c", "modul.c");
        .
        .
    }
}
```

---

## SFS\_AttribFile

U08 SFS\_AttribFile(U08 OS\_HUGE \*Name, SFS\_ATTR Attribs)

Changes the attributes of the file in the current or handed over path. The path statement can absolutely or relatively take place on that occasion. The to changing file cannot be opened by any other user on that occasion.

### Parameters

*Name	File-name [with path]
Attribs	ew File-attributes

### Return Value

SFS_NO_ERR	File attributes changed
SFS_NO_USER	User unknown
SFS_NO_FILE	no file name in the path given
SFS_FILE_OPEN	User, itself or other, this file opened currently
SFS_NOT_EXIST	File doesn't exist in this directory
SFS_PATH_ERR	an element of the path statement doesn't exist
SFS_WRONG_A	wrong file attributes

### Example

```
SFS_USER  SFS_User1;

void OS_FAR Task1(void *data)
{
    U08  returnOk;

    .
    while(1)
    {
        .
        returnOk=SFS_BecomeUser (&SFS_User1);
        .
        .
        returnOk=SFS_AttribFile("../SFS_err.log", SFS_ATTR_RO);
        .
        .
    }
}
```

---

## SFS\_GetFileAttrib

```
SFS_ATTR SFS_GetFileAttrib(U08 OS_HUGE *Name)
```

It returns the attributes of the file in the current or handed over path. The path statement can absolutely or relatively take place on that occasion.

### Parameters

*Name	File-name [with path]
-------	-----------------------

### Return Value

If the returned attribs -1, so you get the error-codes with a following SFS\_GetErrNo().

SFS_NO_ERR	File attributes readed
SFS_NO_USER	User unknown
SFS_NO_FILE	no file name in the path given
SFS_NOT_EXIST	File doesn't exist in this directory
SFS_PATH_ERR	an element of the path statement doesn't exist

### Example

```
SFS_USER SFS_User1;

void OS_FAR Task1(void *data)
{
    U08      returnOk;
    SFS_ATTR attribs;

    .
    while(1)
    {
        .
        returnOk=SFS_BecomeUser(&SFS_User1);
        .
        .
        attribs=SFS_GetFileAttrib("../SFS_err.log");
        if(attribs==(SFS_ATTR)(-1))
            returnOk=SFS_GetErrNo();
        .
        .
    }
}
```

---

## SFS\_GetFileSize

```
SFS_LONG SFS_GetFileSize(U08 OS_HUGE *Name)
```

It returns the current size of the file in the current or handed over path. The path statement can absolutely or relatively take place on that occasion.

### Parameters

*Name	File-name [with path]
-------	-----------------------

### Return Value

If the returned filesize zero, so you get the error-codes with a following `SFS_GetErrNo()`.

SFS_NO_ERR	filesize returned
SFS_NO_USER	User unknown
SFS_NO_FILE	no file name in the path given
SFS_NOT_EXIST	File doesn't exist in this directory
SFS_PATH_ERR	an element of the path statement doesn't exist
SFS_MEM_ERR	Mistakes in the memory management

### Example

```
SFS_USER SFS_User1;

void OS_FAR Task1(void *data)
{
    U08      returnOk;
    SFS_LONG filesize;

    .
    while(1)
    {
        .
        returnOk=SFS_BecomeUser(&SFS_User1);
        .
        filesize=SFS_GetFileSize("/Dir2/config.sys");
        if(!filesize)
            returnOk=SFS_GetErrNo();
        .
        .
    }
}
```

---



## SFS\_GetMaxFileSize

```
SFS_LONG SFS_GetMaxFileSize(U08 OS_HUGE *Name)
```

It returns the max size of the file in the current or handed over path. The path statement can absolutely or relatively take place on that occasion. This is the size of the file, the file was created.

### Parameters

*Name	File-name [with path]
-------	-----------------------

### Return Value

If the returned filesize zero, so you get the error-codes with a following SFS\_GetErrNo().

SFS_NO_ERR	filesize returned
SFS_NO_USER	User unknown
SFS_NO_FILE	no file name in the path given
SFS_NOT_EXIST	File doesn't exist in this directory
SFS_PATH_ERR	an element of the path statement doesn't exist
SFS_MEM_ERR	Mistakes in the memory management

### Example

```
SFS_USER SFS_User1;

void OS_FAR Task1(void *data)
{
    U08      returnOk;
    SFS_LONG filesize;

    .
    while(1)
    {
        .
        returnOk=SFS_BecomeUser(&SFS_User1);
        .
        filesize=SFS_GetMaxFileSize("/Dir2/config.sys");
        if(!filesize)
            returnOk=SFS_GetErrNo();
        .
        .
    }
}
```

---

# File-Access

---

## SFS\_OpenFile

SFS\_HANDLE SFS\_OpenFile(U08 OS\_HUGE \*Name, U08 Mode)

Open a file in the current or handed over path. The path statement can absolutely or relatively take place on that occasion. The to opening file cannot be opened by any other user on that occasion. For all folloing accesses a handle will returned, under this this file-data can access. This handle is intern referenced with the User and is checked on every access.

If a NULL-handle returned, so get the Error-Code with SFS\_GetErrNo().

### Access conditions:

If a file opened (more than once) to reading, an other User can't open this file to writing - if a file from one User opened to writing, no other User can open this file for reading or writing.

### Parameters

*Name	File-name [with path]
mode	mode of access (ReadOnly/WriteOnly/ReadWrite)

### Return Value

If a NULL-handle returned, so get the Error-Codes with a following SFS\_GetErrNo().

SFS_NO_ERR	File opened
SFS_NO_USER	User unknown
SFS_NO_FILE	no file name in the path given
SFS_FILE_OPEN	another User opened this file currently (see Access conditions)
SFS_FILE_RO	File is ReadOnly and cannot be opened "write"
SFS_FILE_WO	File is WriteOnly and cannot be opened "read"
SFS_NOT_EXIST	File doesn't exist in this directory
SFS_PATH_ERR	an element of the path statement doesn't exist
SFS_WRONG_A	wrong access attributes

### Example

```
SFS_USER  SFS_User1;

void OS_FAR Task1(void *data)
{
    U08      returnOk;
    SFS_HANDLE  handl;

    .
    while(1)
    {
        .
        returnOk=SFS_BecomeUser(&SFS_User1);
        .
        .
        handl=SFS_OpenFile("/Dir2/config.sys", SFS_ACC_RO);
        if(!handl)
            returnOk=SFS_GetErrNo();
        .
        .
    }
}
```



## SFS\_CloseFile

U08 SFS\_CloseFile(SFS\_HANDLE Handl)

Close the currently opened file.

### Parameters

Handl	File-Handle
-------	-------------

### Return Value

SFS_NO_ERR	File closed
SFS_NO_USER	User unknown
SFS_NO_FILE	Handle invalid

### Example

```
SFS_USER  SFS_User1;

void OS_FAR Task1(void *data)
{
    U08      returnOk;
    SFS_HANDLE  handl;

    .
    while(1)
    {
        .
        returnOk=SFS_BecomeUser(&SFS_User1);
        .
        .
        handl=SFS_OpenFile("/Dir2/config.sys", SFS_ACC_RO);
        if(handl) {
            .
            returnOk=SFS_CloseFile(handl);
            .
        }
        .
    }
}
```

---

## SFS\_SeekFile

U08 SFS\_SeekFile(SFS\_LONG offset, SFS\_HANDLE Handl)

Places the R/W-pointer within the opened file absolutely.

### Parameters

offset	absolut pointer-position (in bytes), 0 for start of file, SFS_SEEK_EOF for end of file
Handl	File-Handle

### Return Value

SFS_NO_ERR	Pointer in file placed
SFS_NO_HAND	handle invalid
SFS_NO_USER	User unknown
SFS_NO_FILE	no file is opened / Handle invalid
SFS_NO_DATA	File has zero-lenght
SFS_WRONG_PTR	offset greater file-size

### Example

```
SFS_USER  SFS_User1;

void OS_FAR Task1(void *data)
{
    U08      returnOk;
    SFS_HANDLE  handl;

    .
    while(1)
    {
        .
        returnOk=SFS_BecomeUser(&SFS_User1);
        .
        .
        handl=SFS_OpenFile("/Dir2/config.sys", SFS_ACC_RO);
        if(handl) {
            .
            returnOk=SFS_SeekFile(100, handl);
        }
        .
    }
}
```

---

## SFS\_TellFile

SFS\_LONG SFS\_TellFile(SFS\_HANDLE Handl)

It returns the R/W-pointer of opened file.

### Parameters

Handl	File-Handle
-------	-------------

### Return Value

If the returned position is zero, so get the following Error-Codes with SFS\_GetErrNo().

SFS_NO_ERR	pointer returned (start of file)
SFS_NO_HAND	handle invalid
SFS_NO_USER	User unknown
SFS_NO_FILE	no file is opened / Handle invalid
SFS_NO_DATA	File has zero-lenght

### Example

```
SFS_USER  SFS_User1;

void OS_FAR Task1(void *data)
{
    U08      returnOk;
    SFS_HANDLE  handl;
    SFS_LONG  posit;

    .
    while(1)
    {
        .
        returnOk=SFS_BecomeUser(&SFS_User1);
        .
        .
        handl=SFS_OpenFile("/Dir2/config.sys", SFS_ACC_RO);
        if(handl) {
            .
            posit=SFS_TellFile(handl);
            if(!posit)
                returnOk=SFS_GetErrNo();

            }
        .
    }
}
```

---

## SFS\_SetEOF

U08 SFS\_SetEOF(SFS\_HANDLE Handl)

Set EndOfFile in opened file to actual R/W-pointer.

### Parameters

Handl	File-Handle
-------	-------------

### Return Value

SFS_NO_ERR	EndOfFile set
SFS_NO_HAND	handle invalid
SFS_NO_USER	User unknown
SFS_NO_FILE	no file is open
SFS_NO_DATA	File has zero-length
SFS_FILE_RO	File or Open-mode is Read-Only

### Example

```
SFS_USER  SFS_User1;

void OS_FAR Task1(void *data)
{
    U08      returnOk;
    SFS_HANDLE  handl;
    U08      writebuffer[]={"SFS_Test_File R/W"};
    SFS_LONG  written;

    .
    while(1)
    {
        .
        returnOk=SFS_BecomeUser(&SFS_User1);
        .
        handl=SFS_OpenFile("/Dir2/config.sys", SFS_ACC_RW);
        if(handl) {
            written=SFS_WriteFile(writebuffer, strlen(writebuffer)-1, handl);
            if(written == strlen(writebuffer)-1)
                returnOk=SFS_SetEOF(handl);
            .
        }
        .
    }
}
```

---

## SFS\_ReadFile

```
SFS_LONG SFS_ReadFile(U08 OS_HUGE *dest, SFS_LONG size, SFS_HANDLE Handl)
```

Reads number of bytes from currently opened file from current position. After successful reading, the R/W-pointer stands behind the readed block.

The readed number of bytes will returned. If this not the same from the call, so call `SFS_GetErrNo()` to get the error-code.

### Parameters

*dest	Pointer to buffer where the bytes must written in
size	Bytes to read
Handl	File-Handle

### Return Value

If the returned number of readed bytes not the same from the call, so you get the following error-codes from `SFS_GetErrNo()`.

SFS_NO_ERR	Bytes from file readed
SFS_NO_HAND	handle invalid
SFS_NO_USER	User unknown
SFS_NO_FILE	no file is open
SFS_NO_DATA	File has zero-lenght
SFS_FILE_EOF	End-Of-File
SFS_FILE_WO	File or Open-mode is Write-Only
SFS_WRONG_PTR	offset and/or size greater file-size

### Example

```
SFS_USER  SFS_User1;

void OS_FAR Task1(void *data)
{
    U08      returnOk;
    SFS_HANDLE  handl;
    U08      readbuffer[100];
    SFS_LONG  readed;

    .
    ...
    {
        .
        returnOk=SFS_BecomeUser(&SFS_User1);
        .
        .
        handl=SFS_OpenFile("/Dir2/config.sys", SFS_ACC_RO);
        if(handl) {
            returnOk=SFS_SeekFile(50, handl);
            .
            readed=SFS_ReadFile(&readbuffer[0], 80, handl);
            if(readed != 80)
                returnOk=SFS_GetErrNo();
            .
        }
        .
        .
    }
}
```





## SFS\_WriteFile

```
SFS_LONG SFS_WriteFile(U08 OS_HUGE *src, SFS_LONG size, SFS_HANDLE Handl)
```

Writes number of byte in currently opened file beginning on current position. After successful writing, the R/W-pointer stands behind the written block.

The written number of bytes will returned. If this not the same from the call, so call `SFS_GetErrNo()` to get the error-code.

### Parameters

*src	Pointer to source-buffer
size	Bytes to write
Handl	File-Handle

### Return Value

If the returned number of written bytes not the same from the call, so you get the following error-codes from `SFS_GetErrNo()`.

SFS_NO_ERR	Bytes in file written
SFS_NO_HAND	Handle invalid
SFS_NO_USER	User unknown
SFS_NO_FILE	no file is open / handle invalid
SFS_NO_DATA	File has zero-lenght
SFS_FILE_RO	File or Open-mode is Read-Only
SFS_WRONG_PTR	actual offset plus size greater file-size

### Example

```
SFS_USER  SFS_User1;

void OS_FAR Task1(void *data)
{
    U08      returnOk;
    SFS_HANDLE  handl;
    U08      writebuffer[]={"SFS_Test_File R/W"};
    SFS_LONG  written;

    .
    ...
    {
        .
        returnOk=SFS_BecomeUser(&SFS_User1);
        .
        .
        handl=SFS_OpenFile("/Dir2/config.sys", SFS_ACC_WO);
        if(handl) {
            written=SFS_WriteFile(writebuffer, strlen(writebuffer)-1, handl);
            if(written != strlen(writebuffer)-1)
                returnOk=SFS_GetErrNo();
            .
            .
            .
        }
        .
    }
}
```

---

## SFS\_GetErrNo

U08 SFS\_GetErrNo(void)

returns the error-code from Open, Read, Write ...

### Parameters

none
------

### Return Value

error-code	error-code from last called and failed function-call without error-code return in API
------------	---------------------------------------------------------------------------------------

### Example

```
SFS_USER  SFS_User1;

void OS_FAR Task1(void *data)
{
    U08      returnOk;
    SFS_HANDLE  handl;

    .
    while(1)
    {
        .
        returnOk=SFS_BecomeUser(&SFS_User1);
        .
        .
        handl=SFS_OpenFile("/Dir2/config.sys", SFS_ACC_RO);
        if(!handl)
            returnOk=SFS_GetErrNo();
        .
        .
    }
}
```

---

## Link-Handling

---

### SFS\_CreateLink

U08 SFS\_CreateLink(U08 OS\_HUGE \*OrgName, U08 OS\_HUGE \*Name)

Creates a new link to a file or directory. The original entry can be located in another tree-part. As attributes of this link are the attributes of the linked entry valid.

#### Parameters

*OrgName	File/Directory-name to link [with path]
*Name	Link-name [with path]

#### Return Value

SFS_NO_ERR	Link created
SFS_NO_USER	User unknown
SFS_PATH_ERR	an element of the path statement doesn't exist
SFS_NAME_EXIST	a entry with same name exists already in this directory
SFS_NO_ENTRY	directory full
SFS_MEM_ERR	Mistakes in the memory management
SFS_MAX_LINK	*OrgName is SFS_MAX_Links linked

#### Example

```
SFS_USER  SFS_User1;

void OS_FAR Task1(void *data)
{
    U08  returnOk;

    .
    ...
    {
        .
        returnOk=SFS_BecomeUser(&SFS_User1);
        .
        returnOk=SFS_CreateLink("/Dir2/config.sys", "../test.dir/linked.conf");
        .
        .
    }
}
```

---

## SFS\_RemoveLink

U08 SFS\_RemoveLink(U08 OS\_HUGE \*LinkName)

Deletes the link in the current or handed over path. The path statement can absolutely or relatively take place on that occasion. The link to be deleted cannot be linked by another entry.

--- TO DELETING DIR / FILE: ---

Linked Dir / Files can't be removed while one link to this entry is valid !

### Parameters

*Name	Link-name [with path]
-------	-----------------------

### Return Value

SFS_NO_ERR	Link removed
SFS_NO_USER	User unknown
SFS_PATH_ERR	an element of the path statement doesn't exist
SFS_MEM_ERR	Mistakes in the memory management
SFS_LINKED	*Name is oneself linked
SFS_NO_LINK	the entry isn't a link / the entry is a link but points to no entry or this linked entry doesn't know this

### Example

```
SFS_USER  SFS_User1;

void OS_FAR Task1(void *data)
{
    U08  returnOk;

    .
    ...
    {
        .
        returnOk=SFS_BecomeUser(&SFS_User1);
        .
        returnOk=SFS_CreateLink("/Dir2/config.sys", "/test.dir/linked.conf");
        .
        .
        returnOk=SFS_RemoveLink("../test.dir/linked.conf");
    }
}
```

---

## Entries

---

### SFS\_GetEntry

U08 SFS\_GetEntry(U08 OS\_HUGE \*Name, SFS\_GET OS\_HUGE \*get)

Returns in the struct all relevant information of the given entry.

#### Parameters

*Name	Entry-name (dir/file/link) [with path]
*get	pointer to GET struct

#### Return Value

SFS_NO_ERR	entry read
SFS_NOT_EXIST	this entry doesn't exist
SFS_NO_USER	User unknown

#### Example

```
SFS_USER  SFS_User1;

void OS_FAR Task1(void *data)
{
    U08      returnOk;
    SFS_GET  get;

    .
    while(1)
    {
        .
        returnOk=SFS_BecomeUser(&SFS_User1);
        .
        returnOk=SFS_GetEntry("/Dir2/element", &get);
        .
        if (get.Attr & SFS_ATTR_DIR) {
            .
            .
        }
        .
        .
    }
}
```

---

## Comments

---



## Comments

---

## Comments

---