smxFFS™

Flash File System

smxFFS is a flash file system for use with board-resident arrays of NAND and NOR flash memory. It has the standard C library file API, it is power fail-safe, and it has a small memory footprint. It has been completely redesigned in v2.

Overview

smxFFS is a simple, power fail-safe flash file system for NAND and NOR flash. It provides the standard C library API (fopen(), fread(), fwrite(), fseek(), fclose(), etc.) to the application. Unlike the smxFS FAT file system, the smxFFS file structure is not DOS- or Windows-compatible. It was designed specifically for flash memory.

Large Flash / Small Footprint

smxFFS was designed to support very large flash devices, up to 256TB, yet have a small footprint (see Features).

Power-Fail-Safe Operation

Unlike the FAT file system, smxFFS has no FAT area, and each file's metadata (like a directory entry in the FAT file system) is located in a separate sector, so if power fails during a file operation, only those files that are not closed may lose data. Other files and the file system itself will not be damaged. smxFFS has a built-in routine to check every file on the flash disk when mounting it to remove any unexpected data.

Porting Layer

Although integrated with the SMX® RTOS, smxFFS can be ported to other RTOSes or run standalone. The porting layer is based on smxBase. You also need to port smxNAND and smxNOR to your hardware platform.

Features

- Supports NAND, NOR, or any block device that can guarantee data consistency within each sector.
- Flash media up to 256TB.
- Standard C library APIs for most common file operations.
- Subdirectory support (limited to 3 levels of nesting and 254 files per directory).
- Power fail safe.
- RAM: 2KB for the file system, plus 2KB for each open file (for 512 byte sector or page size). Additional for flash driver; see Data Size below.
- ROM: 20 KB for the complete API. Additional for flash driver; see Code Size below.
- Shares flash with smxFS, smxFLog, boot code, and application code.

Limitations

In order to achieve the small footprint yet large flash support, it was necessary to put some restrictions on the capabilities of smxFFS.

- Maximum file length is 4GB-2.
- Maximum file name length and number of files per directory is specified at compile time.
- Subdirectories limited to 3 levels of nesting and 254 files per directory.
- Moderate performance.

smxFFS API

sff_init ()	Initialize the file system.
sff_exit ()	Uninitialize the file system.
sff_devreg (dev_if, id)	Register a device driver to the system.
sff_devunreg (id)	Un-Register a device driver from the system.
sff_fopen (filename, mode)	Open a file for read/write access.
sff_fclose (filehandle)	Close an open file and flush all data to the storage media.
sff_fread (buf, size, items, filehandle)	Read data from an open file.
sff_fwrite (buf, size, items, filehandle)	Write data to an open file.
sff_fseek (filehandle, offset, method)	Move the file pointer to the specified location.
sff_fdelete (filename)	Delete a file.
sff_rename (oldname, newname)	Rename a file or directory.
sff_timestamp (filename, datetime)	Set a file's modification timestamp.
sff_filelength (filename)	Return the length of a file, in bytes.
sff_rewind (filehandle)	Move the file pointer to the beginning of the file.
sff_fflush (filehandle)	Flush all data associated with the file handle to the storage media.
sff_ftell (filehandle)	Determine the current file pointer position.
sff_truncate (filehandle)	Truncate a file at the current file pointer.
sff_feof (filehandle)	Test for end-of-file.
sff_findfile (filename)	Test if a file exists.
sff_findfirst (filespec, fileinfo)	Provide information about the first instance of a file whose name matches the name specified by the *filespec* argument.
sff_findnext (id, fileinfo)	Find the next file, if any, whose name matches the *filespec* argument in a previous call to sfs_findfirst(), and return info about it in *fileinfo* structure.
sff_findclose (fileinfo)	Clean up after the findfirst/findnext operation.
s**ff_mkdir** (path)	Create a new directory.
sff_rmdir (path)	Remove a directory.
sff_chkdsk (id)	Check and fix disk problems.
sff_format (id)	Format the storage media.
sff_freekb (id)	Return the number of free KB on the storage media.
sff_totalkb (id)	Return the number of total KB on the storage media.
sff_clustersize (id)	Return the data cluster size.

Size and Performance

Code Size

Code size varies depending upon the CPU, compiler, and optimization level. Below is an example.

CPU and Compiler	smxFFS	smxNAND	smxNOR
ARM & IAR 6.10	20 KB	14 KB	5 KB

Data Size

Data size only depends upon the flash sector size and number of open files. smxNAND data size increases with flash size. 13KB for 256MB flash disk; 37KB for 1GB flash disk. See the smxNAND datasheet for more size information. smxNOR data size is not affected by flash size.

Sector Size and Open Files	smxFFS	smxNAND	smxNOR
512 byte sector size, one open file	4 KB	13 KB	1.5 KB
2048 byte sector size, one open file	10 KB	13 KB	1.5 KB

Performance

Performance highly depends upon the flash chip, bus speed, microprocessor speed, and RAM speed. It is recommended that you do measurements on your hardware before making final design decisions, if performance is critical. The results here are intended only to provide guidance.

Platform	Reading	Writing
AT91SAM9M10G45-EK, 256MB NAND	5600 KB/s	1900 KB/s