

DESCRIPTION

µC/FS is a compact, reliable, high-performance file system for microprocessors, microcontrollers and DSPs. µC/FS is based on clean, consistent ANSI C source code, with extensive comments describing most global variables and all functions. It supports the FAT file system for interoperability with all major operating systems. An optional journaling module provides fail-safe operation, while maintaining FAT compatibility.

The memory footprint of µC/FS can be adjusted at compile time based on required features and the desired level of run-time argument checking. For applications with limited RAM, features such as cache and read/write buffering can be disabled; for applications with sufficient RAM, enabling these features improves performance.

Device drivers are available for all common media types. Each of these is written with a layered structure so that it can easily be ported to your hardware. The device driver structure is simple--basically just initialization, read and write functions--so that a new driver can be developed easily for a new medium.

µC/FS can access multiple media simultaneously, including multiple instances of the same type of medium (since all drivers are re-entrant). In addition, a logical device driver is provided so that a single file system can span several (typically identical) devices, such as a bank of NAND flash chips.

µC/FS	
Supported Processors	Any Processor
ROM Footprint	10-50 kB
RAM Requirement	2 kB (minimum)
FAT Support	FAT12/16/32 Long file name (VFAT) can be enabled at compile time. File names can be specified in Unicode (UTF-8) Optional journaling capability
RTOS	Not necessary; when µC/FS is used with an RTOS, a simple port guarantees thread-safe operation
Can be used with...	µC/USB Host Stack Mass Storage Class µC/USB Device Stack Mass Storage Class µC/FTPc, µC/FTPs or µC/HTTPS µC/Shell (also µC/TELNETs)

FEATURES

- POSIX-compatible interface for file access (fopen(), fread(), etc.) and directory access (opendir(), readdir())
- Processor independent
- Easily ported to new platforms
- RAM and ROM requirements scalable
- Full FAT support including FAT12/16/32 and long file names (VFAT)
- Optional journaling module for failsafe FAT operation
- Device format and partition creation

CAPABILITIES

The primary file interface is the familiar POSIX interface, with the following equivalents of standard functions:

fs_clearerr()	fs_fread()	s_mkdir()
fs_closedir()	fs_fseek()	fs_opendir()
fs_fclose()	fs_fsetpos()	fs_readdir()
fs_feof()	fs_ftell()	fs_remove()
fs_ferror()	fs_ftruncate()	fs_rename()
fs_fflush()	fs_ftrylockfile()	fs_rewind()
fs_fgetpos()	fs_funlockfile()	fs_rmdir()
fs_flockfile()	fs_fwrite()	fs_setvbuf()
fs_fopen()		

Device/volume control functions allow direct access to the underlying medium. A device can be formatted, reformatted, or divided into multiple partitions, each of which may be formatted.