

# µC/FS

## Embedded File System

µC/FS is a FAT file system that can be used on any media, for which you can provide basic hardware access functions. µC/FS is a high performance library that has been optimized for speed, versatility and memory footprint.

µC/FS has been designed to cooperate with any kind of hardware. To use a specific hardware with µC/FS, a device driver for that hardware is required. The device driver consists of basic I/O functions for accessing the hardware and a global table, which holds pointers to these functions.

## FEATURES AND BENEFITS

µC/FS is written in ANSI C and can be used on virtually any CPU. Some of the features of µC/FS are:

**MS-DOS/MS-Windows compatible** FAT12, FAT16 and FAT32 support.

**Multiple device driver support.** You can use different device drivers with µC/FS, which allows you to access different types of hardware with the file system at the same time.

**Multiple media support.** A device driver allows you to access different medias at the same time.

**OS support.** µC/FS is suitable for usage with just about any multithreaded environment. To ensure that different tasks can access the file system concurrently, you have to implement a few operating system dependent functions. µC/OS-II functions are included with µC/FS.

**ANSI C stdio.h like API** for user applications. An application using standard C I/O library can easily be ported to use µC/FS.

**Very simple device driver structure.** µC/FS device drivers need only very basic functions for reading and writing blocks. Therefore it is very simple to support your custom hardware.

**Generic device drivers for:**

- SMC (SmartCards)
- SD (Secure Digital)
- MMC (MultiMedia Card)
- CF (CompactFlash)
- IDE (Integrated Device Electronics)
- RAMdisk
- Windows (to allow you to simulate your software in a Windows environment) and others.

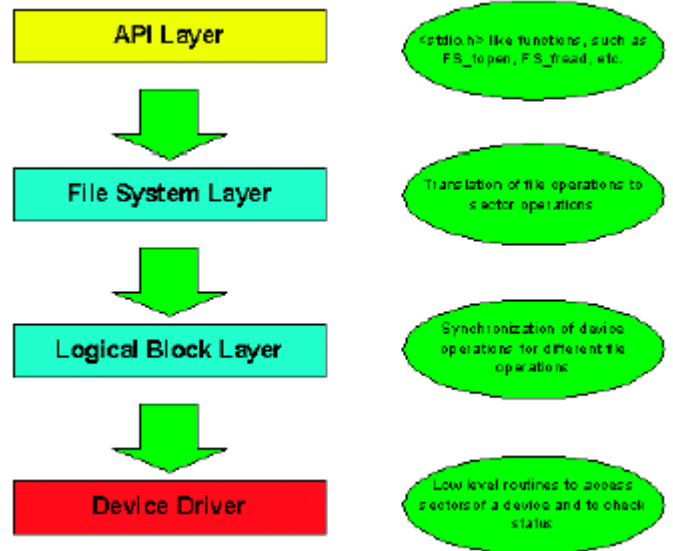
## MEMORY FOOTPRINT

These numbers have been achieved using an ARM 7 target in ARM mode with the IAR compiler. The configuration does allow only 1 open file. Additional open files require 1064 RAM bytes (2 sector buffers + one file handle).

**Using a Ramdisk**

- ROM: 15896
- RAM: 1584

## LAYERED ARCHITECTURE



**API Layer:** The API layer is the interface between µC/FS and the user application. It contains a library of ANSI C oriented file functions, such as FS\_FOpen(), FS\_FWrite() etc. The API layer transfers these calls to the file system layer. Currently we only provide a FAT file system layer for µC/FS, but the API layer can deal with different file system layers at the same time. So it is possible to use FAT and any other file system concurrently with µC/FS.

**File System Layer:** This layer translates file operations to logical block operations. After such a translation, the file system calls the logical block layer and specifies the corresponding device driver for a device.

**Logical Block Layer:** The main purpose of the logical block layer is to synchronize accesses to a device driver and to have an easy interface for the file system layer. The logical block layer calls a device driver to make a block operation.

**Device Driver Layer:** Device Drivers are low level routines that are used to access your hardware. The structure of the device driver is simple to allow easy integration of your hardware.