Configuring uC/FS with *fs_cfg.h* (Part I)

Initialization

µC/FS comes with several configuration files. This article will focus on the top level configuration file of uC/FS. The others are specifically geared towards the inner layers of the file system and are not needed for all projects, such as SD Card, NAND device etc. We will discuss the configuration file "*fs_cfg.h"*. The file systems configurable behavior is controlled by how the developer defines the following configuration values to reflect the applications needs.

It helps to ask yourself a few questions about your application:

- What type of file system do I need? Do I need an embedded sized File System or a full sized FS line Linux, Windows, MAC, etc. Do I need a FAT File System? Transactional File System?
- How much data am I looking to store and transfer around? If your embedded device is looking to handle more data than a FAT 32 file system can store you may have some difficulties.
- Do I need to restrict access to the file system? Read-only or Read-Write functionality.
- What device do I intend to run my file system on? SD Card, NAND or NOR Chip, RAMDisk implementation?

When setting up your file system application you will need to configure *fs_cfg.h* to the specifics of your application. We'll cover the first group of enable/disable configurations. The remainder of the configurations will be covered in a later articles.

```
#define  FS_CFG_SYS_DRV_SEL                      FS_SYS_DRV_SEL_FAT
#define  FS_CFG_API_EN                           DEF_ENABLED
#define  FS_CFG_CACHE_EN                         DEF_DISABLED
#define  FS_CFG_DIR_EN                           DEF_ENABLED
#define  FS_CFG_FILE_BUF_EN                      DEF_ENABLED
#define  FS_CFG_FILE_LOCK_EN                     DEF_DISABLED
#define  FS_CFG_PARTITION_EN                     DEF_DISABLED
#define  FS_CFG_WORKING_DIR_EN                   DEF_DISABLED
#define  FS_CFG_UTF8_EN                          DEF_DISABLED
#define  FS_CFG_RD_ONLY_EN                       DEF_DISABLED
#define  FS_CFG_CONCURRENT_ENTRIES_ACCESS_EN     DEF_DISABLED
#define  FS_CFG_64_BITS_LBA_EN                   DEF_DISABLED
#define  FS_CFG_BUF_ALIGN_OCTETS                 sizeof(CPU_DATA)
```

The above values are part of the **File System Configuration** block of the file system configurations. Most of these definitions are purely enable or disable decisions. These configurable values will be further expanded on below.

FS_CFG_SYS_DRV_SEL: Here the included file system driver can be specified. Typically, this is going to only indicate a FAT file system selection. It is, by default, set to **FS_SYS_DRV_SEL_FAT** to indicate the use of our FAT file system driver.

FS_CFG_API_EN: If you wish to utilize any POSIX-compatible API's within your application you would need to enable their inclusion with this configuration. These **Portable Operating System Interface** API's are not something that must be included in an application and if not needed you can disable their inclusion to save of code space. If you are porting an existing application over to use uC/FS and it has POSIX function calls you can enable them here and they would be supported by the stack.

FS_CFG_CACHE_EN: This variable enables or disables the use of built in caching functionality. The FS stack supports the use of caching when handling volumes, creating files and maintaining updated data. If this feature is not needed, as it does have an added overhead cost, you should disable it.

FS_CFG_DIR_EN: The support and management of directories with uC/FS can be enabled/disabled. If you application does not require the use of directories then simply disabling that functionality allows for the reduction of overall code space. If your application is simple and only needs to use a few files and directories

FS_CFG_FILE_BUF_EN: Enable or disable the use of file buffers. This functionality allows you to assign a buffer of a configurable size to a file after it is created in the file system. These data buffers will be lost if power loss occurs and there is no secondary safe-guard setup. Aside from just assigning a buffer to a file this also includes any API's that manage these buffers such as flushing its contents and other buffer functionality.

FS_CFG_FILE_LOCK_EN: This enables/disables a locking functionality for files created within uC/FS. Locking, in this sense, refers to locking or unlocking access to a file to prevent operations being locked up whilst waiting for access to a file. Enabling this functionality allows file access to be passed around to prevent tasks being starved out while a file is accessed by another task.

FS_CFG_PARTITION_EN: If you wish to use partitioning in your application you will need to enable this functionality. Enabling this allows for multiple volumes to be opened on partitions you create on your device. If disabled the creation and use of additional partitions cannot be utilized. This, like other configurations, is made as such to allow for reduced code size if the functionality is not needed.

FS_CFG_WORKING_DIR_EN: If you wish to use relative path referencing within the file system you MUST enable this working directory functionality. With this enabled your file system implementation can perform operations relative to your current working directory. With this disabled you can ONLY use absolute path references within your file system.

FS_CFG_UTF8_EN: To enable support of any UTF-8 characters you'll need to enable that here. Enabling this functionality grants support for UTF-8 file names to be used in the file system. With it disabled you must specify all file names using ASCII.

FS_CFG_RD_ONLY_EN: If you wish for your instance of uC/FS to be read only you can set that up using this configuration variable. Straight-forward use with this one. If you want to restrict any write access to files, directories and volumes enable Read Only mode. Otherwise Read and Write code included in full.

FS_CFG_CONCURRENT_ENTRIES_ACCESS_EN: If you wish to use concurrent access to files and directories then you would want to enable these functionalities. When enabled access to files and directories is more flexible but needs more careful setup to avoid conflicts. With this functionality disabled concurrent access is not allowed thus making these operations a bit more secure and safer.

FS_CFG_64_BITS_LBA_EN: Enabling 64-bit LBA (Logical Block Addressing). If this is enabled your device is able to support storage sectors of a size 2^64. If your application is a bit limited on space this feature can be disabled allowing the file system to only support 2^32 sized storage sectors.

FS_CFG_BUF_ALIGN_OCTETS: This should be used to specify the minimum buffer alignment required by the device. This should be specified in octets. This alignment value will be applied to any internal filesystem buffers. Any application buffers, allocated by the application, are NOT verified for this alignment.

There are many other configurable features of uC/FS that will be covered in subsequent articles. Again, we will cover these other functionalities in the articles to come. It's important to familiarize with these configuration variables in order to best utilize and optimize your implementation of uC/FS with your design.