Electronic Wilderness File System (EWFS) Design Document

1. Introduction

The driving force behind this file system is to provide high performance for reading file data on an embedded system. The result is a file system that provides indexed file read access using minimal memory and processing. It is intended to be used with flash memory where the file system predominantly reads data and not update files. This use case greatly simplifies and narrows down the operational and design space resulting in the use with potential applications like storing web pages for a web server. Other file systems for embedded devices require large amounts of memory to support the underlying file management capability which doesn't get fully used or is not needed in most applications.

1.1. Requirements

- Small memory (RAM) footprint.
- Support of directories and files.
- High performance for embedded systems.
- Support generated files by the microcontroller and stored files in flash.

2. Design

The current implementation is done with a PIC32MZEF Starter Kit and was intended to replace the MPFS files. The byte order of values are Least Significant Byte (LSB) first.

A major concept in this file system is the concept of a file and a generated file. A file is data that is saved on flash. A generated file is not saved on flash but is generated by the microcontroller during file reading requests. It is generally used for dynamic data like as json files.

2.1. Addressing

The addressing of files is done using a 32 bit unsigned integer which translates into 4 GB of addressable space with a maximum files size of 4 GB. There is a maximum of 65,535 files supported.

2.2. Composition of File System

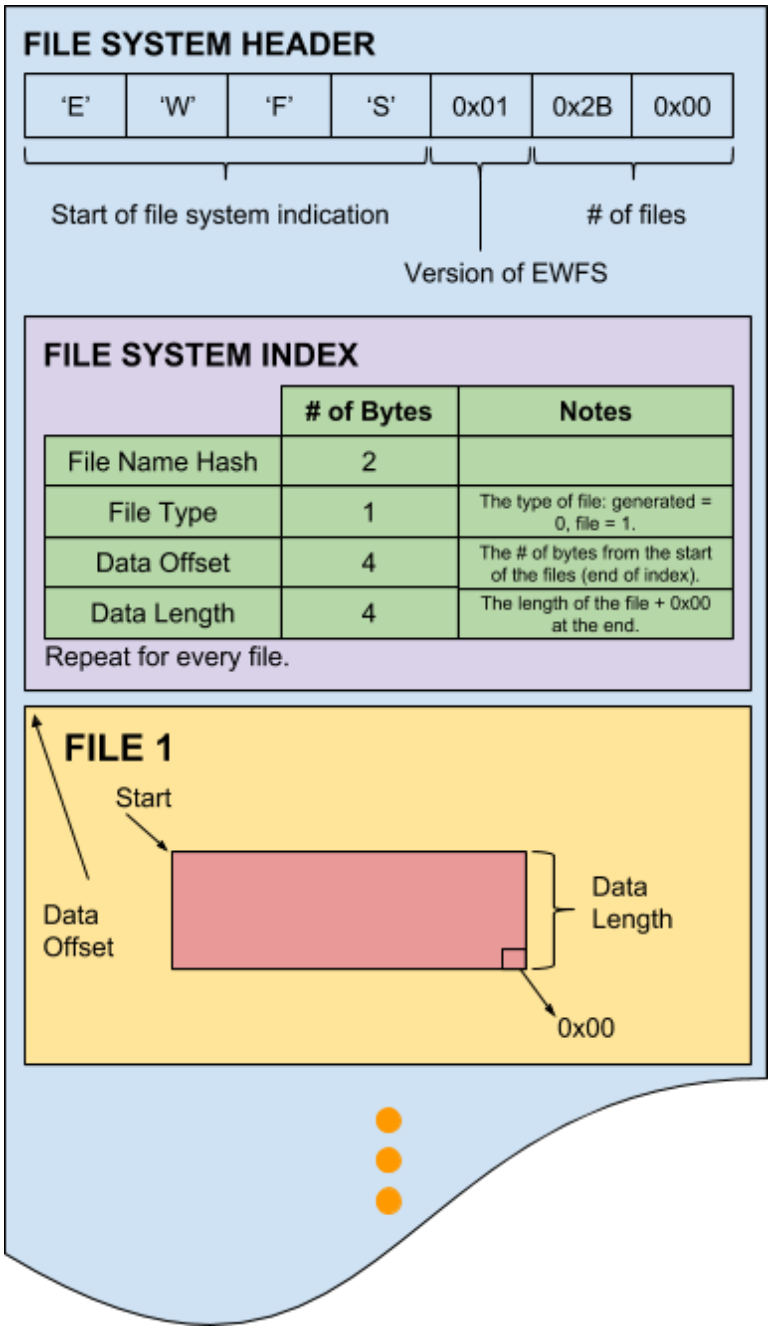


Figure 1: EWFS Memory Representation.

The intent of the file system header and the index are to reside in the flash or the index can be in RAM for faster accessing. The motivation for this is to allow changes to the file system to be independent to the microcontroller code. A good example of file system dependencies on

microcontroller code would be a linked list array in the microcontroller program memory that will point to the address of the file in external flash.

2.2.1. File System Header

File system header information contains the following information:

- “EWFS” is the first 4 bytes of the file system, this indicates the file system type.
- A byte indicates the version of the file system
- 2 bytes are indicating the number of files within the file system (LSB format)

2.2.2. File System Index

An index of the file system provides a fixed index memory size for each file to facilitate searching in the file system. The file system index includes normal files and generated files. The file name is not included in the index, instead a hash is used of the file name (including directory) to keep the RAM usage size small and increase operational speed.

An optional feature of the file system index is to allow for the index to be cacheable on the microcontroller for faster access and searching. This is possible because the file system index has a small size even for a large number of files.

2.2.2.1. File Name Hash

This includes only the text of the file names specified in the directory that the image generator was run on. For example, if the directory specified to the image generator was “/web_page”, the files within the folder will have the hash run on {web_page_sub_dir}/{file_name}.{extension}.

The pseudo code for the hash generation is as follows:

Pseudo Code	Comments/Notes
`hash = 0`	`Initialize the hash to 0`
`For each {character} in the string`	`Loop through all the characters in the file name string`
`    hash = hash << 1`	`Shifting first allows the last bit to change with the new character for this iteration. It also adds additional data that a checksum doesn’t.`
`    hash = hash + {character}`	`Adding character to hash is similar to checksum.`

2.2.2.2. File Type

In the file system index a file type of 1 represents a file stored in memory and 0 represents a generated file. This file type can be set using a special file name of ewfslist.txt and listing the files that are generated in it, where each file is on its own line and ends with a carriage return. A

generated file will have the hash of the file name but the data offset and length fields will be set to 0.

2.2.2.3. Data Offset

The data offset starts counting from 0, which is the first byte after the file system index. So for example, an offset of 17 would mean that the 18th byte is the start of the data for the referenced file.

2.2.2.4. Data Length

The length of the file in bytes and Includes the 0x00 at the end of each file.

2.2.3. File {Data}

The bytes of the file are referenced from the index where the offset is referenced from the start of the files section in memory.

2.3. Image Generation

The generation of the file system inputs a directory with the included files and outputs a binary image of the files in the EWFS. It can be generated using the command:

```
ewfs_generator.exe -f -o "output.bin" -i "folder_in"
```

Where the following parameters represent:

- `-f` Force file overwriting.
- `-i` Input directory with reference to current directory the tool is run in.
- `-o` Output file name.

2.4. Not Supported Features

- Multiple partitions or disks - it was only intended to work across one flash memory chip.
- No wear leaving
- No encryption
- Fail-safe operation
- Timestamp information for files
- Bad block management
- Error correction codes (ECC)

2.5. Future Additions

- The file system index can be sorted based on hash to speed up searching.
- When generating the file system image, the generator can check that each file is a unique hash number of the file name. If not, can alert the user the change the file name of the current file. Currently this is not handled but will only result in the wrong file being

found and read. To fix it, simply change the file name so that a different hash is generated.

- Could add high reliability or fail-safe operation of writing to flash by verifying what was written.
- Add a encryption layer for security of the saved data in external flash to the microcontroller.
- Option to include the file system header and index in the microcontroller flash for a lower RAM footprint when using the index in the cacheable way.