

# **TFS4 FAST LOOKUP METHOD**

## ***Application Note***

---

November-2006, Version 1.0



# Copyright Notice

---

Copyright © 2006, Flash Software Group, Samsung Electronics Co., Ltd

All rights reserved.

## Trademarks

TFS4 is a trademark of Memory Division, Samsung Electronics Co., Ltd in Korea and other countries.

## Restrictions on Use and Transfer

No part of this guide may be reproduced in any form or by any means (including electronic storage and retrieval or translation into a foreign language) without prior agreement and written consent from Samsung Electronics Co., Ltd. as governed by Republic of Korea and international copyright laws.

Document confidentiality status: Restricted Distribution

Readers of this document are not granted the right to distribute this document to any third party without prior written agreement from Samsung Electronics Co., Ltd.

Samsung Electronics Co., Ltd may make changes to the contents of this document any time without notice.

THIS DOCUMENT IS INTENDED ONLY TO ASSIST THE READER IN THE USE OF THE PRODUCT. SAMSUNG ELECTRONICS Co., Ltd. SHALL NOT BE LIABLE FOR ANY LOSS OR DAMAGE ARISING FROM THE USE OF ANY INFORMATION IN THIS DOCUMENT. SAMSUNG ELECTRONICS Co., Ltd. ASSUMES NO RESPONSIBILITY FOR ANY ERROR OR OMISSION IN INFORMATION CONTAINED IN THIS DOCUMENT, OR ANY INCORRECT USE OF THE PRODUCT.

## Contact Information

Flash Software Group  
Memory Division  
Samsung Electronics Co., Ltd

**Address:** San #16 Banwol-Dong, Taeon-Eup  
Hwasung-City, Gyeonggi-Do, Korea, 445-701

## Directory Entries

Every file system data object is pointed to by a directory entry of 32 bytes in length, which contain information about the file; name, address of the first cluster (if any), length of the file in bytes, time and date stamps, and a set of attribute bits that determine whether the file was recently backed up, hidden, Read-only, long file name, directory, etc.

But not every data object within the file system is a file - some of the attribute bits are used to signify subdirectories or volume labels. There should only be one volume label per disk volume, found in the root; however, subdirectories may abound. A subdirectory is stored like a file, except that the data clusters of this "file" are interpreted as further lists of file system objects.

Each directory entry is 32 bytes long, so that a 512 byte sector can contain 16 directory entries. The number of sectors in the directory is fixed for the root directory on FAT12 and FAT16 partitions, and sectors are consecutive on disk. For non-root directories, as well as for the root directory on the FAT32 partitions, the number of sectors is not fixed, and the directory is stored according to the normal cluster chain.

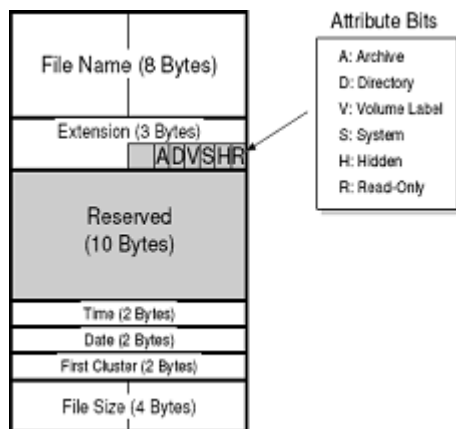


Figure 1. FAT 16 Directory Entry

There are 4 type of directory entry.

- File pointers
- Subdirectory pointers
- Volume labels
- Long File Name components

and these are distinguished by the "directory" and "volume label" attribute bits. If both of these bits are reset, the entry is a file pointer; if the "directory" bit is set, it is a subdirectory pointer; if the "volume label" is set, it is a volume label (one or none of which is present in the root directory), and if both "directory" and "volume label" bits are set, it is a Long File Name component. Long File Name and volume label entries do not point to data clusters and have a "file length" of zero bytes.

There are 11 bytes set aside for the object name in a directory entry. All 11 can be used for volume labels, but in the case of files and (sub)directories, the 11 bytes are interpreted as 8 bytes of name, and 3 bytes of extension (the "three letters after the dot" that determine what should be done with the file). This pattern of 8 name characters plus a 3 character extension is referred to as the 8.3 naming convention, which also excludes spaces, certain other characters, and uses only upper case letters internally. It's quite a restrictive convention to live with, if you want to use meaningful names; hence the challenge to add long file name support while remaining compatible with programs bound to 8.3 names and conventions.

## Fast Lookup Operation

Beyond a single file stored as a stream of bytes, a file system must provide a way to name and organize multiple files. File systems use the term directory to describe a container that organizes files by name. The primary purpose of a directory is to manage a list of files and to connect the name in the directory with the associated file.

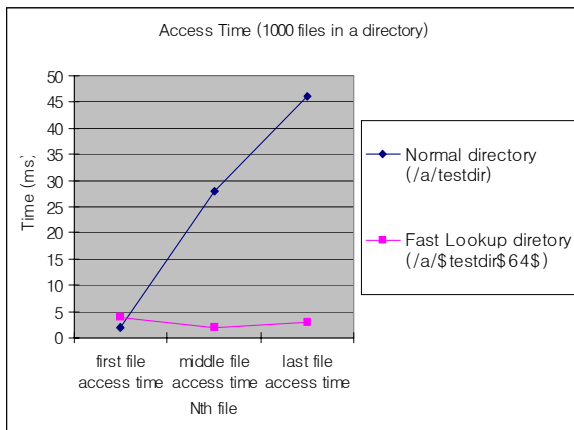
A directory contains the list of names. Associated with each name is a handle that refers to the contents of that name (which may be a file or directory). When a file is first created, internally the checksum value is evaluated for that file. A directory is created with that value and the corresponding file is placed in that directory. There can be multiple files in that directory.

The checksum value is the key that the directory searches on when looking for a file and is a reference that allows the file system to access the directory containing the file. Once the directory is located, the search is done sequentially with the file name for accessing the contents of the file and other metadata about the file.

## Fast Lookup Performance

Consider a test environment with target as Reindeer plus December, Arm9 (202.8 MHz), 128 MB SDRAM, OneNAND KFG1G16, XSR-1.5.0 and FAT 16 with cluster size 8.

The graph below shows the performance comparison for the access time in a normal directory and fastlookup directory.



**Figure 2. Performance Comparison**

The access time in case of fast lookup directory is approx. 15 times faster than that of the normal directory.