

# **TFS4 Power-Off Recovery**

## ***Technical Paper***

---

August-2007, Version 1.0





# Copyright Notice

---

Copyright © 2007, 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



## Summary

A power failure can detrimentally affect the data (e.g. files) integrity of a file system, in a computer that uses flash media to store the data. For instance, suppose that a user of a computer is attempting to store data and has just performed a task that issues a data request. In this scenario the data request is to write data to the flash medium, but unfortunately the user accidentally drops the computer disconnecting the power source. When the user reconnects the battery, will the file system know that a power failure event occurred? Did the data get stored on the medium? How will the file system know whether data was corrupted or not? How will the file system recover from the power failure event and still preserve data integrity for the computer?

## Description

To ensure that TFS4 file system maintains consistency during a power-failure event, the Log manager is implemented to record and maintain log of each transaction. Each set of operations for performing a specific task is a transaction. These transactions include creating files, deleting files, copying files, reading files, creating directories, creating subdirectories, renaming file/directory, and other related file tasks. In particular, the log manager module can be configured to maintain logs for transactions performed by the file system. Fundamentally, all metadata changes related to a transaction are written sequentially to a log file. Once the changes are recorded to this log, the actual operation is executed. A log file generally serves as a history journal of transactions performed by the file system over a period of time that can be read back in the event there is a power failure to ensure integrity of metadata stored on the flash medium.

## Log Writing Process

In the course of performing its various operations such as file write, or file delete, the file system writes log for use in a potential recovery operation. Therefore, the information needed for data recovery is stored in log file in the flash medium. Accumulation of this information occurs in a step by step process during normal execution by the file system. When the log manager receives requests to perform a transaction, the log module stores transaction information associated with performing the operation at the log slot of log file in a circular way. A circular log file writes to the end of its space and then continues at the beginning, overwriting older values as it goes.

## Log Recovery Process

When a system crash occurs, the last few transactions performed on the flash disk may have left it in an inconsistent state (for example, a new file may have been written without modifying its directory entry information); during reboot the operating system must review these operations in order to correct any inconsistencies. In traditional FAT file systems without logs, like Microsoft FAT 32 file system, the system cannot determine where the last changes were made, so it must scan all of the metadata structures on disk to restore consistency. The cost of these scans is already high (tens of minutes in typical configurations), and it is getting higher as storage systems expand. In case of TFS4 file system, the locations of the last transactions are easy to determine: they are at the end of the log. Thus it should be possible to recover very quickly after crashes. During log recovery process, the metadata change information recorded in the log file is read and transactions are recovered from log.

Following steps are implemented in a log based recovery process:

1. Read the log slots in the log file.
2. For the last transaction in the log file, it's possible that it might be executed only partially. So based on the type of the operation, log recovery manager 'undo' or 'redo' the transaction to recover last transaction.
3. For the transactions before the last transaction, they must be "finished" but its execution result may be stored in cache and lost after system crashes. Therefore these finished transactions should be "redo" to recover them.

## Benefits

Following are the main benefits of Samsung's TFS4 FAT based file system:

- Consistency of management of data storage in the file system.
- Prevention of the corruption of metadata when a computer system shuts down abnormally.
- Recovery of data processed in the file system

## Conclusion

This paper describes about the Log manager features in the Samsung's TFS4 FAT based file system where the transactions are performed in an atomic way in order to prevent the data corruption and maintain consistency of file system.