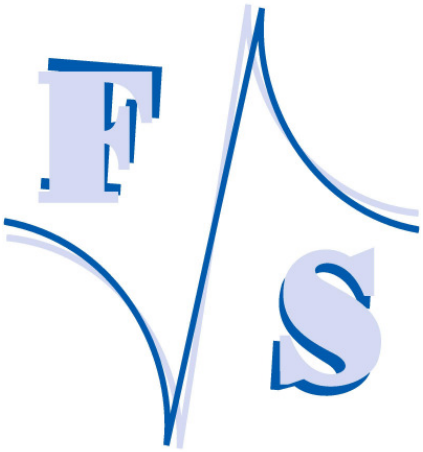
F3S

The transaction-based, power-fail-safe file system for WindowsCE



F & S Elektronik Systeme GmbH
Untere Waldplätze 23
70569 Stuttgart
Phone: +49(0)711/123722-0 Fax: +49(0)711/123722-99

Motivation for the Innovation:

A safe and consistent data management is indispensable in a lot of areas where embedded systems are used, like medical applications for example. The operating system WindowsCE from Microsoft is shipped with the FAT file system for persistent data storage. However it has been shown in a lot of cases, that the FAT file system can't manage this requirement sufficiently. Quite often corrupted data is responsible for malfunctions and sometimes even system failures.

For this reason, F&S has decided to develop a new, transaction-safe file system – the *Failsafe Flash File System (F3S)*. It is especially designed for usage in embedded systems and features an economical and efficient usage of the available system resources. Because of the unique, transaction-based concept, it is fundamentally robust against unexpected electrical power outages. F3S is capable not only to guarantee that meta-information won't be corrupted, like this is possible on FAT file systems when the allocation table gets damaged, it even ensures transaction-safety on file level. Modified file data will be committed not until the operation is finished completely. The point of completion can be defined in an application. In the easiest case by closing the file-handle. If the operation is aborted, the state before the modification was started will be restored automatically. This offers the possibility to save sensible data safely and persistently.

Contrary to conventional file systems for WindowsCE, F3S is targeted to take advantage of NAND flash memory characteristics. Thereby the physically required write operations are minimized. This mainly improves the transfer rate on write operations dramatically. At the same time this reduces wearing of flash memory and thus enhances its lifetime, which is limited anyway. In addition, F3S is equipped with a very efficient Garbage-Collector that resolves principle-occurred fragmentations.

F3S is implemented in C++ using object-oriented methods. As a result of the very modular architecture it is portable to other platforms very easily. Thus it is available for all single board computers of the NetDCU family (WindowsCE 4.2/5/6) from F&S by now.

The F3S file system is maintained and continuously improved. Additionally F&S offers a qualified and customer-oriented support.

Description:

The company F&S Elektronik Systeme GmbH has specialized in developing embedded PCs used to drive graphical displays. These single board computers are equipped with WindowsCE and therefore only include the FAT file system for persistent data storage. Especially in environments where sensible data is used, a safe and consistent data management is indispensable. However it has been shown in a lot of cases, that the FAT file system can't ensure this requirement sufficiently. Quite often corrupted data is responsible for malfunctions and sometimes even system failures. Because of this, F&S has decided to develop a new transactionsafe file system, that is being applied in a lot of systems in the meantime – the *Failsafe Flash File System (F3S).*

The FAT file system uses an allocation table to map file data or contents of directories to their position on the storage medium. This enables a very simple data search. But after a crash that appears while an operation is in progress, there is always the danger of corrupted (meta)-data. Even if writing one data block is atomic, most operations, like creating or deleting a file, are causing several physical write operations. If the electrical power outage occurs while performing a write operation, file data gets corrupted for sure. If modifying the allocation table fails, even complete files or directory structures can get lost. So the goal of development was a file system that ensures the integrity of meta-data and also guarantees an effective protection on file level.

Conventional file systems modify stored data at their current location. If a data block that holds file or meta data is modified, this action will be performed at exactly the same physical location. Due to restricted access of flash memories, this isn't possible directly. There's the need of an additional software layer, implemented between the regular flash driver and the file system – the so called *Flash Abstraction Layer (FAL)* or *Flash Translation Layer (FTL)* (see Figure 1). This enables the file system to access the flash memory like a regular block device. The FTL performs a so called block-mapping. Read and write operation are based on logical block addresses. The size of a data block typically corresponds to the size of a flash page. Writing a modified data block will not change the current physical storage cell. Instead of that, the modified data block will be written to a free physical location. Afterwards, the internal mapping between logical and physical address is adapted.

The fact that each flash page can't be overwritten directly, already supports the development of transaction safety on file level, as old and new file data are both stored separately in flash for a short period of time. Additionally each page of a flash memory includes a special area (spare-page) that offers the possibility to store meta information on flash memory directly. To establish a safety in file level, it is very useful to store the file membership of each page in flash directly. Based on these keynotes the file system F3S was developed.

Generally expressed, transaction safety means that the transition from one stable state to the next stable state only can be performed either completely or not at all. If a transaction isn't completed, the original state will be restored. Related to file contents, a transaction is committed under F3S if one of the following conditions is entered:

- The file is closed: `CloseFile()`
- All used file buffers should be flushed: `FlushFileBuffers()`
- If the `WRITE_THROUGH`-Flag is set and the write operation is finished

If the system crashes before reaching any of these conditions, modified data is lost and the original file content will be recovered.

F3S is a log-structured file system. It carries the meta-journaling, known from NTFS or Ext3/4 to the extreme. The journal is the file system. Data blocks stored in flash memory are not mapped to logical block addresses, instead they are mapped directly to the corresponding file system object (file or directory). This leads to two fundamental benefits compared to conventional file systems:

1. The physical characteristics of flash memory can be utilized directly.
2. Losing data or even directory structures can be avoided by not storing allocation tables or other structures in flash memory.

F3S doesn't store the directory structure in a central structure like file systems for hard discs, as this generally results in inconsistency problems. Instead of this it uses the spare-area of the particular flash page to store all needed meta-information. The exact directory structure is copied to a linked data structure in main memory. There is one object for each file system object, which points to the directory it currently resides in. Several objects on the same hierarchical level are organized as a linked list. With the assistance of this structure, each file system object can be found by giving the pathname. As the relation of an object, or more precisely a reference to the higher ranked directory object, is stored in the spare-page, the correct directory structure can be rebuilt anytime. And because of the atomicity of writing a page, the integrity of the directory structure can be guaranteed in each situation.

To even ensure the consistency of file data, F3S features an additional unique concept. On committing a transaction, a special data block will be written that validates all file modifications before. Thus a transition of modified file data consists of one atomic operation. Any interrupted transaction will be detected next time when mounting the partition, and the last valid state will be recovered automatically.

Using a table that is stored in flash is disclaimed deliberately avoided. Each data block hence is stored in a B-tree like structure within the corresponding file system object in main memory and will be established simultaneously to the directory structure during mount process. In difference to the FAL, mapping of data blocks is not based on logical block addresses. Instead each flash-page is related to file content directly. Thus the mapping is an inherent part of the file system itself. Referring to transaction-safety, this is an essential benefit as it enables a systematic data caching on file level. Cached data blocks are identifiable directly.

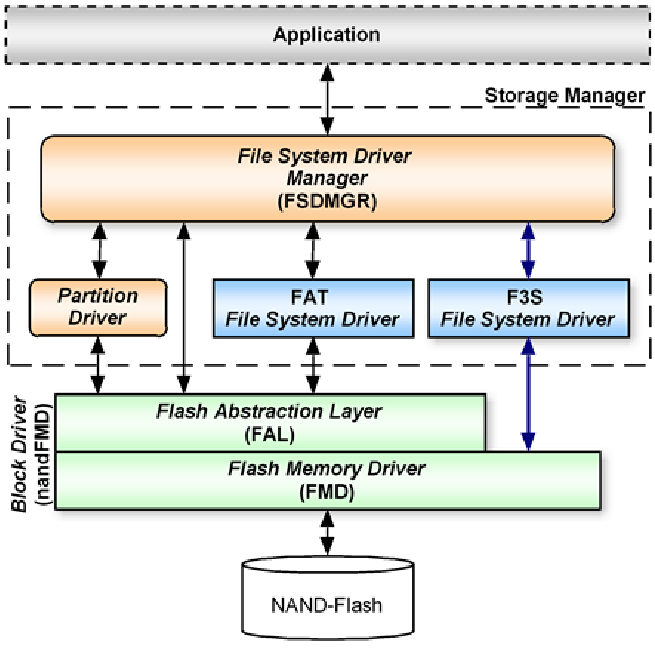


Figure 1: Accessing flash memory on WindowsCE

As a result of the WindowsCE architecture, file systems that are especially developed for flash memories can't be adopted directly. The FAL is an inherent part of the block driver for flash memory restricting the physical access to the memory. Some slight adaptations are inevitable. Running the FAL besides a specific flash file

system is achieved by a special flag in the spare-page, which causes the FAL to ignore the concerning pages from usage. When declaring the data stored in this area, this compatibility must be taken into account.

In the course of time and as a consequence of the characteristics of NAND flash memories, there arise some fragmentation effects within several flash blocks. To work against these effects, a so called Garbage-Collector (GC) is used that performs a defragmentation if needed. In this process, valid data is copied to other free locations to set up deletable blocks that are free for rewriting afterwards. The GC implemented in F3S has two operation modes. On the one hand it scans through flash periodically to detect fragmentation. On the other hand the GC will be initiated if the free available space isn't sufficient to finish the current write operation. There is only freed as much storage space as is actually needed in this situation to keep latency as short as possible. This process doesn't affect transaction safety as data wouldn't be changed.

Because of the ageing process of flash memory cells, lifetime of these memories is limited. It is specified by the manufacturer as a maximum number of delete cycles of a flash block. To use the lifetime efficiently, so called Wear-Leveling mechanisms are applied. Thus F3S for example selects the one free block for writing that has been deleted the least so far.

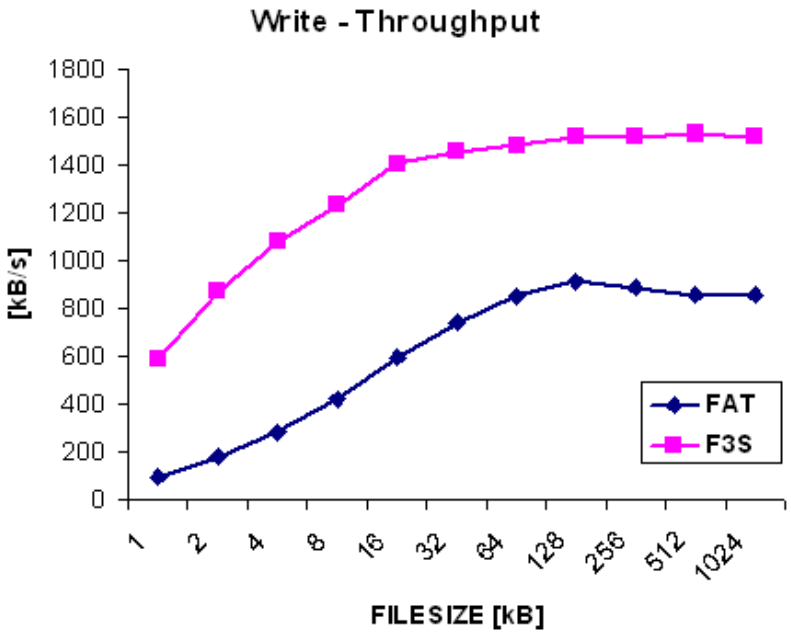


Figure 2: Throuput on write operations

As mentioned before, the specific characteristics of flash memory are addressed. In contrast to conventional file systems, the number of physical write accesses are reduced to a minimum, which mainly improves the transfer rate on read and write operations considerably. Additionally this reduces wearing of flash memory and thereby enhances the lifetime of the flash memory.

F3S is implemented in C++ using object-oriented methods. As a result of the very modular architecture, it is portable to other platforms very easily. Thus it is available for all single board computers of the NetDCU family (WindowsCE 4.2/5/6) from F&S in the meantime. The F3S file system is maintained and continuously improved. Additionally F&S offers a qualified and customer-oriented support.