Application Note

AN102

Using Swissbit SD Card and HCC File System to Implement a Fail-safe Embedded File System

Using Swissbit SD Card and HCC File System to Implement a Fail-safe Embedded File System

At the most fundamental level, for any system to be considered fail-safe, all layers of the system must be predictable and specify the requirements of the adjacent layer. Combining HCC's fail-safe file system with Swissbit's Industrial SD cards is one way to meet that objective. This paper discusses the key requirements of fail-safety and how developers can develop a trusted embedded data storage system.

Each year flash memory costs fall as densities increase and it becomes feasible to store large quantities of data using deeply embedded flash-based systems. However, as the capacity of low-cost flash increases, so does the potentially negative impact of data loss or corruption. Generic SD Cards have no means to manage unexpected system failures. Therefore, when used together with a basic file system, there is a high probability of data loss or irretrievable file system corruption. Many SD cards that are described as 'Industrial' are essentially the same as consumer grade SD Cards but tested to a different temperature range. Most of these devices have no provision for data or file system protection.

To design a system where data is stored in such a way that it is never lost or damaged and is always maintained in a consistent state is a complex task, but well within the reach of all embedded development teams. The fundamental objective is to consider the whole system design and not only the software or hardware in isolation. A reliable system cannot be achieved without the cooperation of both software and hardware.

How to Assess 'Fail-safe' Requirements

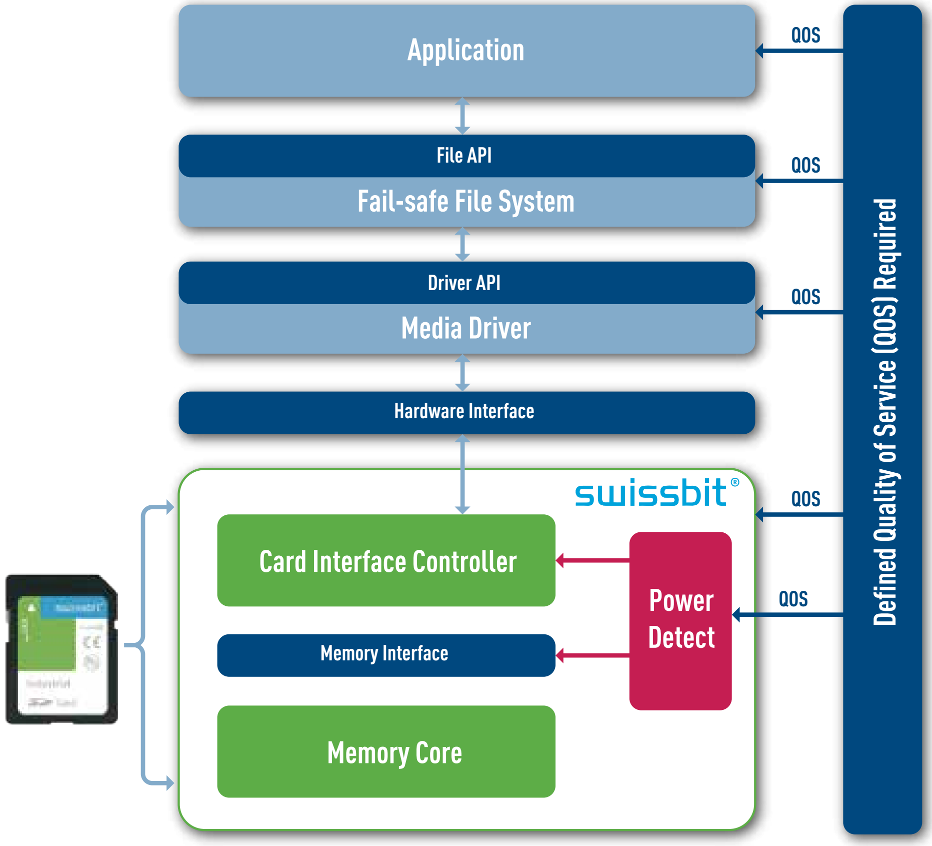
A quick survey of open source and commercial file systems available will reveal many claims about fail-safety and data integrity. In order to make an assessment about the suitability of a file system it is worth outlining the key requirements and what role they play in guaranteeing the integrity of data. As previously stated, each layer of the system needs to understand what the fail-safety requirements are for that level and what services it requires from other levels. File systems cannot claim 'fail-safety' without specifying what Quality of Service (QOS) is required from other parts of the system. It it is important to obtain this information from any potential supplier before commencing design.

A standard model for organizing storage using a Swissbit Industrial SD Card flash media is shown in Figure 1. This layered approach is important in order to portray the dependencies that exist in a typical embedded system. For the overall system to be reliable, the application requires a defined quality of service from the file system, the file system requires a defined quality of service from the drivers, and the drivers require a defined quality of service from the storage media in order to be reliable.

Table 1: Quality of Service Requirements

LAYER	FUNCTION	REQUIRED QUALITY OF SERVICE (QOS)
Application	User defined	User Defined.
Fail-safe File System	Store data organized files to logical sectors on the media.	Atomic switch of file state.
Media Driver	Transfer logical sector data between the file system and the media.	▪ Reliable data delivery. ▪ Detection of physical events such as card removal.
Media	Storage of data in array of logical sectors.	▪ Sequential write of received sectors. ▪ Atomic switch of state of sectors.
Power management	Detect drop in power to minimum specified level.	Signal power fail event.

Figure 1: Fail-safe Quality of Service Requirements Using SD Card



Application Layer Requirements

What does an application developer require for fail-safe storage and what exactly does fail-safe mean? This can be summarized as follows:

1. Any time a file is written, the action of closing or flushing the file will atomically switch its state from the previous consistent state to the new consistent state. The application developer has full control over over data consistency.

2. If an unexpected system event occurs then, when the system restarts, all elements of the file system must be coherent and all files must be in their pre-restart consistent state.

A file system must determine when new data is valid and then signal that it is time to update the associated meta data. All writes to the file system remain uncommitted until the application issues a close or flush signal that the new state should be made valid. In this way the developer can determine what a consistent state for the data is. This does not mean that the data is not written to the target media – just that the meta-data of the file still refers the previously consistent state. It also means that if you seek back in a file and overwrite something, then an original copy should be maintained. This ensures that in the event of an unexpected occurrence, the original state of the file is valid and consistent.

File System Requirements

HCC-Embedded file systems are designed to achieve 100% fail-safe reliability, but this is only feasible if the underlying drivers and physical media can meet certain requirements:

1. The data must be written in the sequence that it is presented to the driver. In the case of parallel writing to a media, it should be guaranteed that no data be committed unless all data previously received by the media have been committed.

2. In the event of power loss then any sector of data must be either written or the old sector contents must remain valid. There should be no intermediate state where the contents of a sector are effectively random. The switch from the sector's old state to new state must be atomic.

It is possible to design fail-safe systems where the storage media has different characteristics. However, the behavior must be clearly defined and understood to ensure that the system meets the failsafe design criteria.

Media Driver Requirements

The media driver should be designed to guarantee the transfer of data between the file system and the storage media and to properly detect physical events i.e. card removed or changed. Some method of verification that the data has correctly transferred is also required. A CRC check can be used to achieve this.

Power Handling Requirements

The application design must provide a stable power supply and a clean power reset when using a system incorporating flash memory. It is essential to implement 'brown-out' detection that resets the card when power drops below a specified level. This is necessary to guarantee a reliable data write. Without this several types of problems can occur; e.g. a write succeeds when a previous write fails or the write / erase fails creating bad blocks.

During extensive testing by HCC engineers, many generic SD cards, including 'industrial' SD cards experienced complete and permanent failure when the power was slowly reduced during write cycles. Swissbit Industrial SD Cards are suitable for environments where the power supply loses stability since behavior in the event of power-loss and brownout is clearly defined and this provides the basis for a fail-safe system.

Swissbit Industrial SD Card Characteristics

Intensive testing by HCC of many generic and 'industrial' SD cards has demonstrated that very few meet the requirements of sequential writing and atomic switching. As such, they are not suitable when data integrity is required. Swissbit S-200/220/s-200u/s-300u range of SD/SDHC/uSD Cards provide integrated reset protection for power loss and brown out. This logic provides all of the services required by the file system to guarantee 100% system fail-safe operation. Upon a sudden power fail, the controller is reset and the flash is immediately write-protected. Swissbit performs extensive power cycling tests on all products to verify that data corruption due to power failure does not occur. The combination of HCC's fail-safe file system and Swissbit's SD card provides the required level of service and can be determined to be fail-safe by design.