DYACON®
RUGGED DATA SYSTEMS

CT605 – USAGE OF THE FILE SYSTEM (TFS)

Doc: 52-6000 Rev B
Date: 30 Jan. 2007

OVERVIEW

This document addresses the way the CT605 File System (TFS) uses the flash so that the user can determine flash life expectancy based on the application's intended use of the file system. Applications use the TFS in different ways, so it is difficult to determine how long the flash will last. Current flash devices typically support 100,000 to one million erases per sector.

TFS API

The TFS (Tiny File System) API presents itself to the programmer in much the same way a standard OS file system would be seen. This may mislead application developers into thinking that the file system can be thought of as a disk that provides the user with the freedom to write/erase at any frequency. This is not the case. The TFS defragmentation mechanism does not use a floating spare sector. This means that the spare sector is the portion of flash that is likely to wear out first simply because it is used as temporary storage for all other sectors when defragmentation is done. Assuming a worst-case defragmentation where all 61 TFS sectors are affected, a defragmentation run once a day on a part having a life expectancy of 100,000 erase cycles will be good for about 4 years (100,000/61/365 = 4). This value of 4 years makes the assumption that a defragmentation will be done daily, and that all sectors are affected by the defragmentation. The frequency of defragmentation and the number of sectors affected are very dependent on the application's use of the file system.

Some applications may wear out flash faster than others based on the way the sectors are utilized. When a file is deleted, it is simply marked as deleted (a bit in the header). A file is deleted by the command *rm* or through the API functions *tfsunlink()*, and *tfsclose()* when the file was opened for writing or appending. If a file that exists is opened for writing, the actual modification steps are deletion of the original file and re-write of the new file after the file that is currently the last file in physical flash space. Eventually this process reaches the end of the flash space used by TFS, so defragmentation must be run. At defragmentation, all of the space wasted by the deleted files is cleaned up and the current list of active files is placed end-to-end in the flash. This means that each sector used by TFS must be copied to the spare sector. That sector is then erased and only the active files in that sector are copied back to the original sector from the spare sector. This is repeated for each sector in TFS. The defragmentation attempts to make this as sector-erase-efficient as possible. If a sector is not affected by the defragmentation, then it is left untouched; hence, the number of sectors actually affected by a defragmentation depends on the amount of flash space currently being used by TFS and the position of the deleted files within the flash space.

In a typical project the files in TFS will be the application itself, an inittab file for system configuration, followed by other application-specific files. The application executable is likely to be the largest file and also the least likely to change at a high frequency. This means that the space taken up by the application executable is fairly static. The number of sectors occupied by the application will not contribute to the wear of the spare sector. This fact in and of itself increases the estimated life expectancy calculated previously. On the other hand, if the application executable is quite large relative to the total TFS space available, and if there are any files that are to be modified on a regular basis, defragmentation will be run more frequently. The point is that the actual wearing rate of the flash is application-specific and must be seriously considered and thought out during application development.