



Tel: 1-800-221-6630
Email: sales@datalight.com
www.datalight.com

Optimizing a Flash Media Manager for NAND Flash Imaging

While the process of preparing and writing images to NOR flash is a straight-forward implementation, creating images for NAND is much more complex, making a uniform programming strategy arduous without the proper flash media management tools. Under the crunch of a product development schedule, it may seem practical to use a hardware simulator to capture an image of a file system, and then write that image to new flash by reading only a part of the flash that is thought to include all of the files. Flash experts would tell you that this is not as simple as it sounds, and is not likely to work reliably.

This paper explores the use of tools within the Datalight FlashFX Pro flash media manager for creating a flash image and initializing fresh NAND flash from this image. We also discuss best practices for high volume production programming of NAND flash.

BACKGROUND ON FLASHFX PRO

FlashFX Pro has three distinct levels of formatting used on the flash: (1) BBM (bad block management), (2) VBF (variable block format), and (3) the file system. Each of these has its own function and imposes different constraints.

Bad Block Management (BBM)

BBM is Datalight's method of working around the bad blocks that are generally present in NAND flash. Even a brand new flash chip from the manufacturer may have some erase blocks that do not meet specifications for reliable data storage, and further blocks may go bad after extended use. BBM is responsible for allocating replacement blocks and remapping addresses from bad blocks to their corresponding replacement blocks.

BBM formatting must be done before any use of NAND flash by FlashFX Pro. Formatting consists of allocating space for replacement blocks, scanning for factory bad blocks, and adding them to the bad block map. After this has been done, BBM reports a size that is slightly smaller than the full size of the flash chip. This smaller size is all that is visible to higher levels of software.

For high volume production, the BBM formatting is done by production programming equipment. It is essential that the programming equipment formats the NAND flash device exactly the same as FlashFX Pro, so that the file system will function properly. For this reason, Datalight partners with Data I/O Corporation and ensures that Data I/O's programming equipment correctly formats NAND flash for FlashFX Pro.

Variable Block Format (VBF)

VBF is Datalight's method of managing the allocation of flash to provide two essential features: small block emulation and wear leveling. File systems are designed to work on disk drives that can be written in small increments called sectors (typically 512 bytes), and can be rewritten repeatedly without affecting other data or decreasing the reliability of the disk. Flash memory is not directly usable by most file systems, as it must be erased in large increments called erase blocks (typically tens to hundreds of Kbytes) before being written, and each erase cycle causes wear that ultimately results in the failure of that erase block (typically after more than 100,000 cycles).

VBF overcomes these limitations of flash memory by storing data in flash memory at the next available location and building a map of where each allocation block (simulated disk sector) is located in the flash. A file system may write repeatedly to the same sector address, but VBF will store the data at a different location each time, marking the old copy invalid ("discarding" the allocation) after the new copy is written. Eventually the space occupied by the discarded data is reclaimed and erased for reuse. VBF tracks how many times each erase unit has been erased, and occasionally relocates static data from an infrequently erased unit to a unit that has been erased more frequently.

VBF formatting involves writing an erase unit header (EUH) to each erase unit of the area of flash managed by VBF. Each time FlashFX Pro starts up, it reads the EUHs and associated allocation tables to build its map of what is where in the flash.

In production, it is necessary that the programming equipment does the initial VBF formatting so that the image file can be properly written to the NAND device during the programming process.

The File System

File system formatting creates the initial data structures for a file system. Though FlashFX Pro can effectively communicate with any file system, the Datalight Reliance transactional file system is recommended to be used with FlashFX Pro to achieve optimum device performance. Every file system has its own way of organizing files and directories. Reading and writing only a portion of a file system makes risky assumptions about where the file system stores its data on the disk.

When working with a FAT file system, its File Allocation Table and root directory are fixed data structures that are guaranteed to be stored near the beginning of the disk. However, file data and subdirectories can be stored anywhere. While it is common for FAT file system implementations to start allocating space for new files near the beginning of the disk, as files are created and deleted they have a potential to end up anywhere, not just at the beginning. Thus there is no guarantee that reading only the beginning of the disk up to the total size of the files (plus some safety margin) will actually capture all of the desired data.

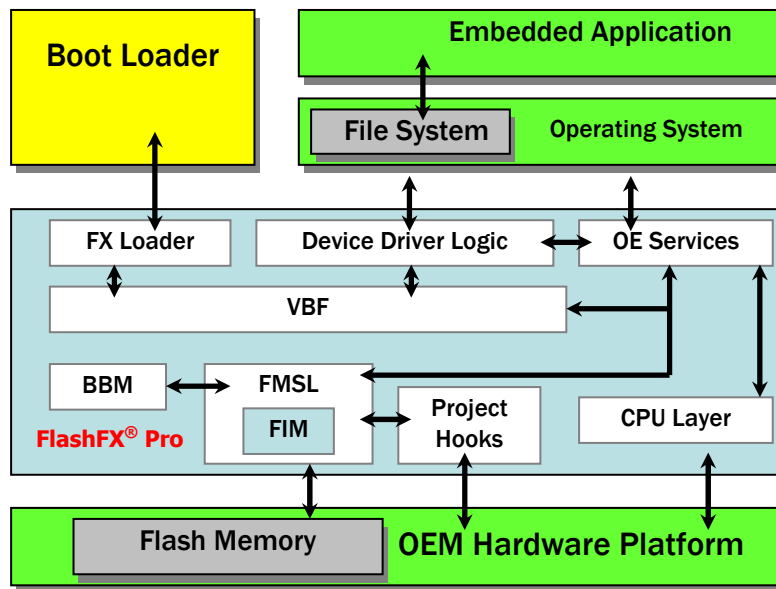


Figure 1: Datalight FlashFX Pro Block Diagram

CREATING AND WRITING AN IMAGE TO FLASH

For low volume applications, FlashFX Pro can be used to write an image to a NAND flash device “in-system”; for example, the image can be written to a NAND flash device in a new phone during development. For high volume applications, the NAND flash device is usually processed by high volume automated programming equipment before being soldered onto the PCB. However, in this case, FlashFX Pro is still needed to create the image by reading from the NAND flash in a development system, for example.

First, we will discuss using FlashFX Pro for low volume reading and writing. FlashFX Pro presents the flash as a simulated hard drive at the block device driver level. Capturing this “hard drive” image, and using it to initialize fresh flash, is unreliable as BBM, VBF and the file system formats are not accounted for in this image. All of these data structures are required as FlashFX Pro does not necessarily do all of its writing near the beginning of the flash, even when it is freshly initialized.

In order to capture the entire FlashFX Pro format, the functions to be used are **oemread** (to read the image) and **oemwrite** (to write it to fresh flash). Using this method, the image is guaranteed to work between devices whose flash arrays are similarly defined. The BBM reserved area will be created at the same location and size on the destination, as it was on the source. Any differences in bad blocks between the two physical flash arrays are handled transparently.

Creating the Image

In summary, creating the image should be performed by linearly reading the raw sectors + the 16-byte spare area. These steps are as follows:

- Format the flash with the flash media manager and create the appropriate file system.
- Populate the file system with the desired files.
- Repeatedly compact the flash until the compaction function (**vbcompact**, in FlashFX Pro) returns **FALSE**, indicating that nothing remains to be compacted.
- Obtain the size of the accessible region of the flash. (*Note that this size does not include the BBM region; this region will be recreated automatically by FlashFX Pro on the new flash.*)
- For each NAND page, call **oemread** (or similar function) and set it to read a total of 516 bytes (512 bytes of user data plus four bytes of VBF allocation information). Write this data to the image file. Do not perform any file system operations on the flash drive while the image is being created.

Writing the Image and Initializing Fresh Flash

Initializing fresh flash from the image file (as created above) should be performed more or less as follows with FlashFX Pro:

- Perform initial BBM formatting. (There is no need to perform VBF formatting, as the image contains the VBF format information).
- Obtain the size of the flash.
- For each NAND page, read 516 bytes from the image file. Examine the data: if it is all ones, simply skip to the next page of flash without writing. Otherwise write it to the next page of flash using **oemwrite**. BBM will handle any bad blocks encountered during this process. Since much of the image file will contain pages that are all ones, it could be compressed using simple run-length encoding or some other scheme to avoid storing data for pages that are in their erased state.
- The size of the file should match the size of the flash.
- After the image has been written, mount the file system.

Next, we will discuss high-volume duplication or programming of the image into new NAND flash devices. Once the application is validated and a final image is created with the process mentioned above, the image is typically checked into configuration management. The process that we describe below assumes that the factory is using Data I/O's RoadRunner programming equipment. The RoadRunner is an "on-line" programmer that mounts directly to an SMT feeder bank and programs the NAND flash devices on the SMT line. (This process avoids risk of having an inventory of pre-programmed flash devices in an application where frequent image (or code) changes occur.) We also assume that the manufacturing floor is using Data I/O's TaskLink[®] software application to create the "task" for the programming equipment. Here, a "task" refers to the binary file that contains instructions for the programming equipment, as well as the master image.

The process is as follows:

- Check the image out of configuration management.
- Start the TaskLink application and create a new “task”.
- In the *Task Edit* dialog box, select the NAND flash device from the list of supported devices.
- Selecting any NAND flash device, will automatically invoke the *Special Features* dialog box; select a bad block handling type from the drop down list shown in Figure 2 below. For FlashFX Pro, use the bad-block method called “Skip Customer 1” as shown.

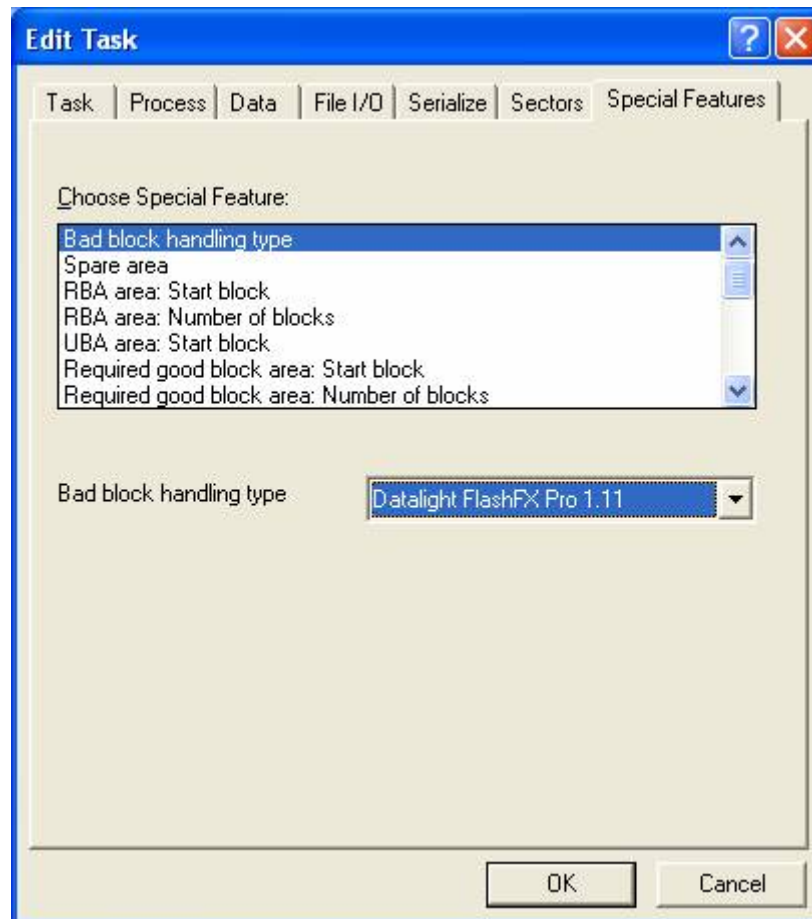


Figure 2: TaskLink Special Feature dialog box.

- In the *Data* dialog box, select the master image (from configuration management).
- The last step is to “Create” a new task and download it to the PCMCIA card for insertion into the RoadRunner on the SMT line.

At this point, you are ready to start high-volume programming of NAND flash devices on your production line, that are guaranteed to work with Datalight's Bad Block Management and Variable block format schemes.

MAXIMIZING WRITE PERFORMANCE

Performance gains can be expected during this image create/write process, with some additional planning:

- Repeatedly compact the data on the flash before creating the image to ensure that all unused storage is erased.
- When writing the image to a new phone, there is no need to program areas that are all ones; such areas may simply be skipped over. This way, the areas that need to be written are written, and all of the areas that can safely be skipped are skipped.

This paper is authored by both Datalight and Data I/O.

About Datalight:

Bill Roman is a Software Architect at Datalight.

Since 1983, Datalight's focus on portable, flexible solutions for embedded systems has enabled OEMs to save money, reduce development time and get to market faster. Datalight has earned a reputation as a provider of reliable, compact and cost-effective data device management software solutions that are backed by a commitment to customer service and satisfaction. Through such features as OS portability, file system reliability and intelligent flash memory utilization, Datalight ensures product quality in the rugged, unstable environments in which many embedded devices operate. For more information, go to www.datalight.com.

About Data I/O:

Kelly Hirsch is the Chief Technologist at Data I/O Corporation.

With more than 32 years of innovative leadership in the device programming industry, Data I/O Corporation® (NASDAQ: [DAIO - News](#)) provides manual and automated device programming systems that specifically address the requirements of engineering and manufacturing operations. FlashCORE™ is the architecture behind a family of Flash programmers that deliver the highest throughput and lowest cost per programmed device. Data I/O leads the industry in technology innovations for management and programming of NAND and NOR flash memories. Data I/O Corporation is headquartered in Redmond, Washington, and has sales and service offices worldwide. For more information, see www.dataio.com or call 800-426-1045.