AP-684

APPLICATION NOTE

Understanding the Flash Translation Layer (FTL) Specification

December 1998

Order Number: 297816-002

CG-041493

CONTENTS

REVISION HISTORY

Number	Description
-001	Original Version
-002	Document formerly referred to as a technical paper; assigned application note number. Legal disclaimer updated.

1.0 INTRODUCTION

This document defines the interfaces and operation behavior of the Flash Translation Layer, or "FTL," as it is more commonly known. For more information about the FTL specification, contact PCMCIA for the full FTL specification contained in the *Media Storage Formats Specification* of the PCMCIA *PC Card Standard* (see Appendix D).

2.0 OVERVIEW OF FLASH TRANSLATION LAYER

Flash memory is valued in many applications as a storage media due to its fast access speeds, low-power, nonvolatile, and rugged operation. FTL is the driver that works in conjunction with an existing operating system (or, in some embedded applications, **as** the operating system) to make linear flash memory appear to the system like a disk drive. It does that be doing a number of things. First, it creates "virtual" small blocks of data, or sectors, out of flash's large erase blocks. Next, it manages data on the flash so that it appears to be "write in place" when in fact it is being stored in different spots in the flash. Finally, FTL manages the flash so there are clean/erased places to store data. How FTL does these is described in detail in the following sections.

2.1 Virtual Block Device

DOS block device drivers perform input and output in structured pieces called blocks. Block devices include all disk drives and other mass-storage devices on the computer. FTL emulates a block device. The flash media appears as a contiguous array of storage blocks numbered from zero to one less than the total number of blocks. FTL is a translation layer between the native DOS BPB/FAT file system and flash. FTL remaps the data to the physical location at which the data is to be written. This allows the DOS file system to treat flash like any other block storage device and remain ignorant of flash device characteristics. FTL appears to simply take the data from the file system and write it at the specified location (sector). In reality, FTL places the data at a free/erased location on the flash media and notes the real location of the data. It also invalidates the block that previously contained the block's data (if any). So when the file system asks for the data that was written out, FTL finds and reads back the proper data.

2.2 Flash Technology

Flash media allows only two states: erased and non-erased. In the erase state, a byte may be either all ones (0xFF) or all zeroes (0x00) depending on the flash device. A given bit of data may only be written when the media is in an erase state. After it is written to, the bit is considered dirty and unusable. In order to return the bit to its erase state, a significantly larger block of flash called an Erase Zone (also known as an erase block) must be erased. Flash technology does not allow the toggling of individual bits or bytes from a non-erased state back to an erased state. FTL shields the file system from these details and remaps the data passed to it by writing to unused data areas in the flash media. This presents the illusion to DOS that a data block is simply overwritten when it is modified—while in reality, the new data has been written somewhere else on the media. FTL also takes care of reclaiming the discarded data blocks for reuse.

Although there are many types and manufacturers of flash memory, the most common type is known as NOR flash. NOR flash, such as that available from Intel Corporation, operates in the following fashion: Erased state is 1, programmed state is 0, a 0 cannot be changed back to a 1 except by an erase, and an erase must occur on a full erase block. Throughout the rest of this document, the examples and bit states will be based on the way NOR flash works.

Figure 2 shows some specific examples of flash commands and the resulting content changes. In the first three program sequences, a bit can be toggled from a "1" to a "0" or left as a "1." However, a "0" may not be set to a "1." The only time a bit can be restored to a "1" from a "0" is during an erase process, as shown at the bottom of the figure.

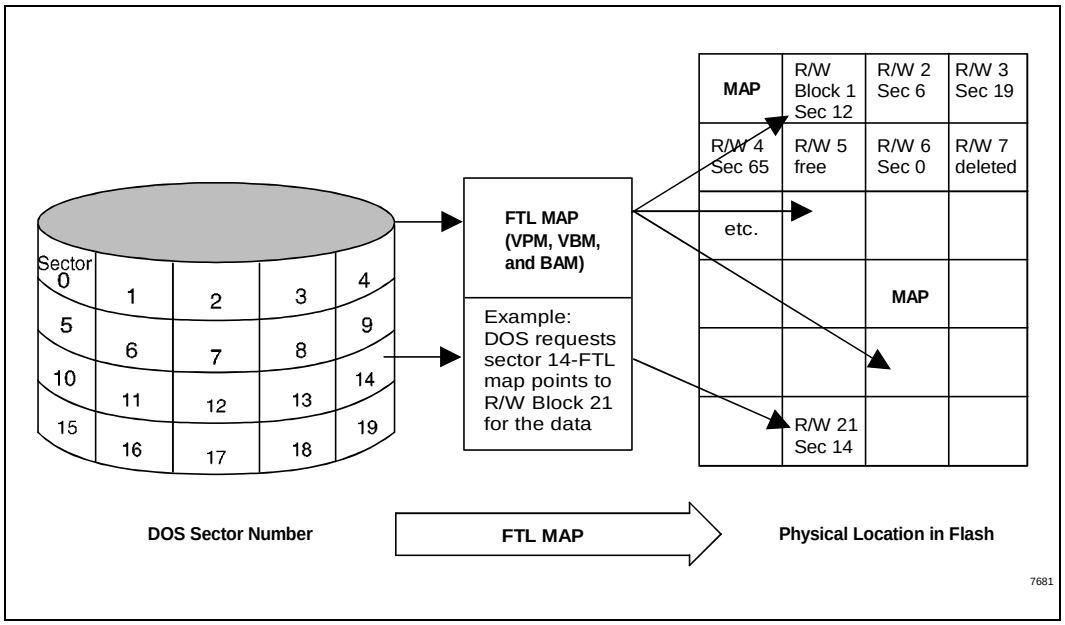

Figure 1. FTL Sector Relocation

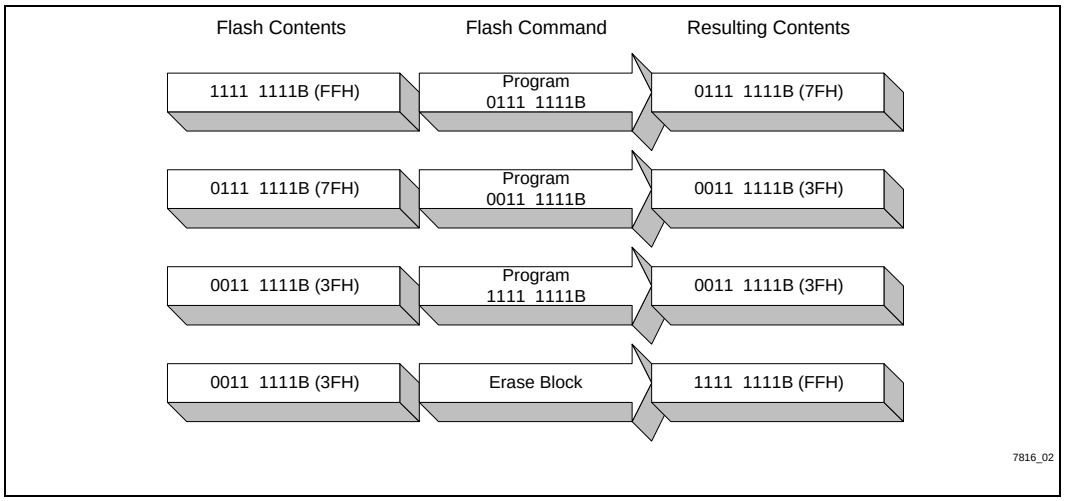

Figure 2. Example of Flash Commands/Content Changes

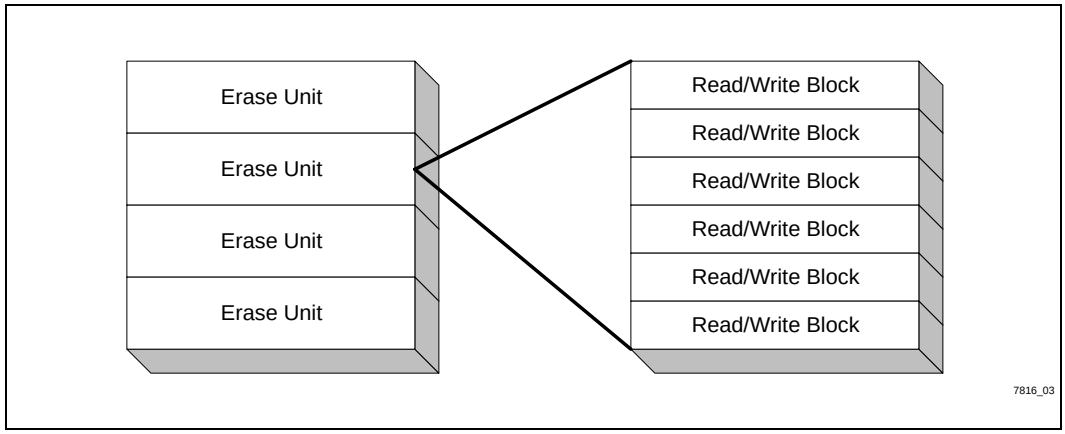

Figure 3. Erase Unit Divided into Read/Write Blocks. Each Read/Write Block Is the Same Size as a Virtual Block (Sector) Used by the Software Layers above FTL.

3.0 ERASE UNITS

FTL divides the flash media into one or more Erase Units of equal size. The size of an Erase Unit is determined during FTL formatting but is also dependent on the size of the device's flash Erase Block as well as component interleaving. An Erase Unit consists of one or more contiguous Erase Zones. An Erase Zone is the smallest contiguous area that can be erased in a single erase operation. If two 8-bit devices are interleaved to form a 16-bit memory, the Erase Unit will consist of an Erase Zone (or Erase Block) on each of the devices. For this example, 1 Erase Unit = 2 Erase Zone (one from each chip in the chip pair). An erase zone is the same as an Erase Block of a flash chip (64-Kbyte typical for current Intel® FlashFile™ memory) devices.

An Erase Unit is evenly divided into one or more equally-sized Read/Write blocks. Each Read/Write block is the same size as a Virtual block used by the host file system. A Virtual Block is the same as a DOS data sector (typically 512 bytes).

3.1 Erase Unit Header

The Erase Unit Header (EUH) is present (created by the FTL formatter) in every erase unit on the media. The EUH is located at either offset zero (0) of the erase unit or at an alternative offset specified by the **AltEUHOffset** field of the an EUH located at offset zero (0) of another Erase Unit. The EUH contains various information regarding the FTL partition. Most of the fields apply globally to the partition and a couple are erase-unit specific. All the EUHs on the media are identical except for the EraseCount and **LogicalEUN** fields.

3.1.1 FLAGS

The **Flags** field in the EUH contains bits describing how checksum and block allocation information are stored on the media and the polarity of the media.

Table 1. Erase Unit Header Flags Field

Bit	Description
0	HiddenAreaFlag
1	ReversePolarityFlash
2	DoubleBAI (Double Block Allocation Information)
3–7	Reserved for future use. All of these bits must be set to zero (0).

NOTE:
See Media Storage Formats section of *PC Card Standards* for additional information.

Table 2. Erase Unit Header

Offset	Field	Sz	Description	Example: 4-Meg F008-based Card	
				HEX	Comments
0	LinkTargetTuple	5	PCMCIA Link Target tuple	13 03 43 49 53	(..CIS)
5	DataOrganization Tuple	10	PCMCIA Data Organization tuple	46 39 00 46 54 4C 31 30 30 00	(F9.FTL100.)
15	NumTransferUnits	1	Number of transfer units in partition	01	
16	Reserved	4	Reserved	xx xx	
20	LogicalEUN	2	Logical Erase Unit Number of this block	FF FF	
22	BlockSize	1	Size of a Read/Write Block	09h 2^{9h} = 200h	(512 dec)
23	EraseUnitSize	1	Size of an Erase Unit	11h 2^{11h} = 20000h	(128k dec)
24	FirstPhysicalEUN	2	Physical Erase Unit where FTL partition begins	00 00	
26	NumEraseUnits	2	Total number of Erase Units in partition	20 00	(0020)
28	FormattedSize	4	Total formatted size of partition	00 10 3C 00	(003C1000)
32	FirstVMAddress	4	First virtual address physically mapped in VBM page on media	00 00 01 00	(00100000)
36	NumVMPages	2	Total number of VBM pages	3D 00	(003D)
38	Flags	1	Special bit-mapped flags	00	
39	Code	1	Code designating EDAC type	FF	None
40	SerialNumber	4	Partition serial number	00 00 00 00	
44	AltEUHOffset	4	Offset of alternative EUH	00 00 00 00	
48	BAMOffset	4	Offset of BAM from start of EUH	44 00 00 00	(00000044)
52	Reserved	12	Reserved	xx	

NOTE:
See Media Storage Formats section of *PC Card Standards* for additional information.

3.2 Block Allocation Information

Each Erase Unit keeps allocation information about every Read/Write block in the Erase Unit. Every Read/Write block has a 4-byte value tracking its current state. A Read/Write block in an Erase Unit is either deleted, bad, free, or allocated at any given time. Read/Write blocks store four types of data: FTL Control Ctructures, Virtual Block Data, Virtual Block Map Pages, and Replacement Pages.

Block allocation entries for Virtual Block data, Virtual Block Map Pages, and Replacement Pages contain two codes. The least significant seven bits indicate whether the block contains Virtual Block data, a Virtual Block Map Page, or a Replacement Page. The most significant 25 bits are used to compute the virtual address of the data. FTL assigns a virtual address to each Virtual Block in the contiguous array of blocks presented to the host file system. For Virtual Block Data, the 25 bits are the most significant bits of the virtual address. The least significant bits of the virtual address are assumed to be zeros (0). For Virtual Block Map Pages and Replacement Pages, the most significant 25 bits are used to build the Virtual Page Map. Read/Write blocks containing Virtual Block data have positive BAM entries and positive virtual addresses. Read/Write blocks containing VBM Pages and Replacement Pages have negative BAM entries and negative virtual addresses. Figure 4 contains some example BAM entries and the corresponding block types.

3.3 Block Allocation Map

According to the FTL specification, block allocation information may be stored in two different ways. One method is to store the allocation in hidden areas next to or related to the Read/Write block to which it refers. The other method is to store the allocation information for all the Read/Write blocks in the Erase Unit together in an array called the Block Allocation Map (BAM). This is the most common technique used by the commercially available FTL solutions. One of the bits in the EUH Flags field (see Table 4) indicates which method is used on the media. Figure 4, below, is an example of the kind of entries that may be contained in a BAM. To the right of each BAM entry is its corresponding definition.

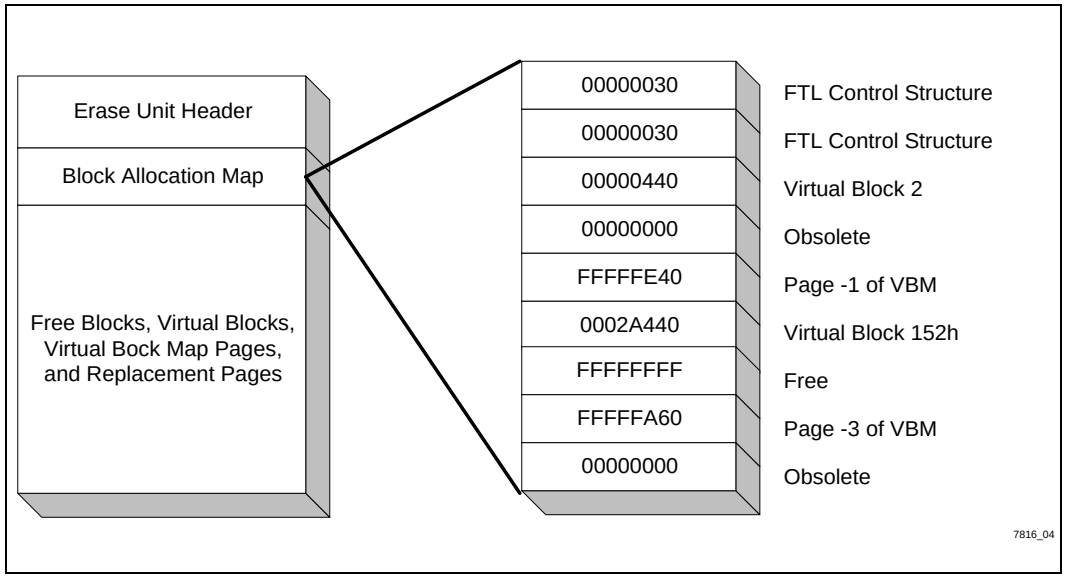

Figure 4. Block Allocation Map Example

Table 3. Block Allocation Information

32-Bit BAM Entry	Meaning	Description
0xFFFFFFFF	Free	Read/Write block is available for writing.
0xFFFFFFE or 0x00000000	Deleted	Data in block is invalid. This Read/Write block must be erased before it may be reused. The value 0xFFFFFFFE indicates that a write operation on this block was interrupted before completion. The value 0x00000000 (all bits programmed) indicates that the data in this block is obsolete.
0x00000070	Bad	This Read/Write block is unusable and may not be written to or read from.
0x00000030	Control	This Read/Write block contains FTL control structures (e.g., EUH, BAM, ECCs, etc.).
0xXXXXXX40	Data or Map Page	Contains data or a Virtual Map Page.
0xXXXXXX60	Replacement Map Page	Replacement Page for a Virtual Map Page.

NOTE:
See Media Storage Formats section of *PC Card Standards* for additional information.

Table 4. Block Allocation Information Example

32-Bit BAM Entry	Page Type	Logical Address	Page Number
0x00003240	Data Page	0x3200	N/A
0xFFFFFE60	Replacement Map Page	N/A	Replacement VBM Page -1
0xFFFFFC40	Map Page	N/A	VBM Page - 2
0x001B2440	Data Page	0x1B2400	N/A
0xFFFFFA40	Map Page	N/A	VBM Page -3
0xFFFFFE40	Map Page	N/A	VBM Page -1

4.0 VERIFY FTL PARTITION

There are two ways to identify a Flash Translation Layer (FTL) partition. An FTL partition may be identified by reading the PC Card's Card Information Structure (CIS) or searching the storage media for FTL data structures. The FTL Data Organization Tuple (located at the beginning of each EUH) is shown in Table 5.

If FTL uses the entire media, the CIS is not required to contain partition information. An FTL partition may be recognized if an FTL Erase Unit Header is found in the first megabyte of the storage media and the information in the header is valid. The search for the FTL Data Organization Tuple entails scanning at offset five of the Erase Unit Header for each Flash Erase Zone in the first megabyte of the storage media. If the tuple is not found within the first megabyte of the partition area, the media is not an FTL data store. If the tuple is found, the rest of the entries in the Erase Unit Header must be validated.

A dynamic map of the logical to physical translation may also be built during this validation process. FTL will read every EUH on the media starting with the unit described by the **FirstPhysicalEUN** field. The EUH must be found at either offset zero (0) or at the **AltEUHOffset**. The **AltEUHOffset** may be found in an EUH located at offset zero (0) of another block. If the EUH is not found, the unit is treated as a transfer unit. If two Erase Units have the same LogicalEUN, either Erase Unit may be treated as a transfer unit. After all the EUHs have been read, the total number of units with distinct and non-negative **LogicalEUN**s must equal **NumEraseUnits** minus **NumTransferUnits**.

The BAM is also scanned to locate VBM Pages and Replacement Pages. The total number of VBM Pages found must equal **NumVMPages**. If a VBM Page is missing and a Replacement Page exists for that particular VBM page, then the Replacement page is used as the original VBM Page. If a VBM Page is missing and a Replacement Page does not exist, recovery operations must be performed. If duplicate VBM Pages are found, only one is used.

- **Related FTL Functions:** Fill_VBM_PageMap

Table 5. FTL Data Organization Tuple

Byte	D7	D6	D5	D4	D3	D2	D1	D0
0	Tuple Code		CISTPL_ORG, 46H					
1	Tuple Link		Link to next tuple (at least 07H)					
2	TPLORG_TYPE		TPLORGTYPE_FS, 00H					
3–9	TPLORG_DESC		"FTL100\0"		Null terminate string identifying FTL partition.			

NOTE:
See Media Storage Formats section of PC Standards for additional information.

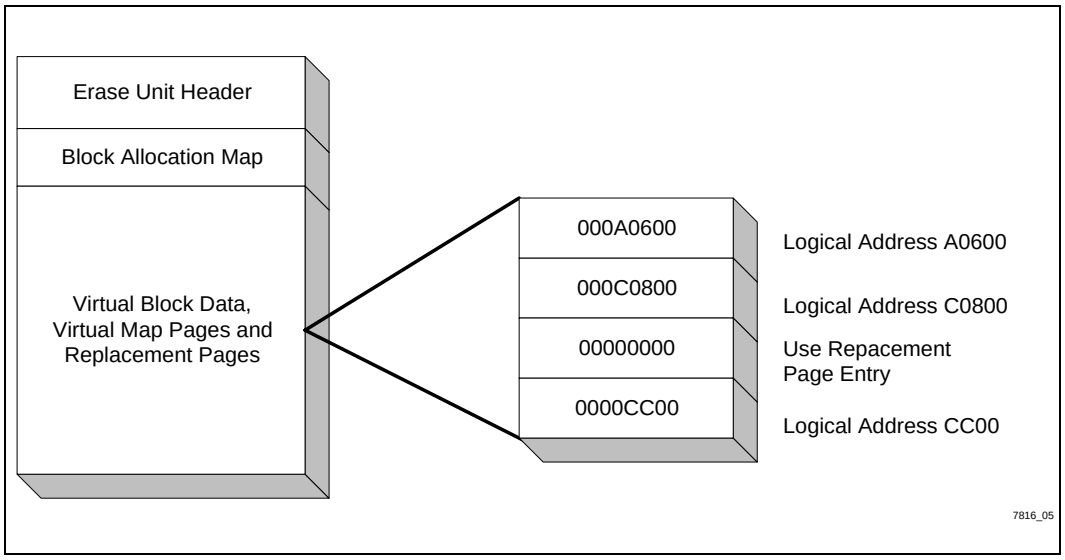

Figure 5. Virtual Block Map Page Example

5.0 VIRTUAL BLOCK MAP– VIRTUAL-TO-LOGICAL MAPPING

FTL performs virtual to logical mapping translations through the use of a Virtual Block Map (VBM). The VBM consists of one or more Pages, each the same size as the Virtual Blocks presented to the host system. Each VBM Page is an array of 32-bit entries; each entry points to a logical address on the media where the corresponding Virtual Data block resides. VBM entries are stored in little-endian order on the media. FTL uses the virtual block number from the host file system as the index into a VBM Page. The VBM is constructed from the virtual addresses of Virtual Data blocks.

The size of the VBM can be calculated by dividing the media's **FormattedSize** by the Virtual Block **BlockSize** and multiplying the result by the size of a VBM entry (32-bits). The number of Pages (**NumVMPages**) required for the VBM is found by taking the size of the VBM and dividing it by **BlockSize** and rounding up to the nearest whole number. Each VBM Page is the size of a Read/Write block. Space for the entire VBM is always reserved on the media whether or not the VBM is maintained on the media itself. At the time of media formatting, FTL indicates in the **FirstVMAddress** field of the Erase Unit Header exactly how much of the VBM is actually maintained on the media.

5.1 First Virtual Mapped Address

The First Virtual Mapped Address (**FirstVMAddress**) refers to the lowest virtual address that is mapped in a VBM on the media. Virtual addresses below the FirstVMAddress are typically not kept on the media so, in that case, the VBM for these addresses are recreated in system memory during card insertion or start-up. The sectors in the lower part of the virtual memory space are modified frequently by FAT. If those VBMs were maintained in flash, the numerous accesses to that memory space would require the frequent updating of BAM entries, VBM entries, and data—thus greatly impacting performance. Keeping this information in system memory increases the Read/Write block usage and decreases the amount of time before reclaim needs to happen.

A larger **FirstVMAddress** may improve performance, but increases the system memory requirements. The largest **FirstVMAddress** FTL supports is 0x10000. If it is set to zero (0), then FTL maintains the entire VBM on the media. If the **FirstVMAddress** exceeds the **FormattedSize** of the media, none of the VBM is maintained on the media. When all or a portion of the VBM is not maintained on the media, FTL must recreate the missing portions of the VBM in host system RAM. A slower way of performing the virtual to logical translation would be for FTL to scan the BAM entries

for the desired Virtual Data block. Exactly how much of the VBM to maintain in RAM is dependent on available system memory and performance needs. In maintaining a portion of the VBM in RAM, FTL may not need to update the media every time a Virtual Data block is written.

VBM entries may range from 0x00000000 to 0xFFFFFFFF. However, all values in between must be a multiple of **BlockSize**. A VBM entry of 0x00000000 indicates that either the logical address of the Virtual Data block is located on a Replacement Page if one exists or that the block simply does not exist on the media. A VBM entry of 0xFFFFFFFF also indicates that the Virtual block does not exist on the media. FTL returns **BlockSize** bytes of "0"s as virtual data if the host system requests a Virtual Data block not present on the media.

- **Related FTL Functions:**

  — Fill_VBM_PageMap

6.0 VIRTUAL PAGE MAP–LOCATING THE VIRTUAL BLOCK MAP PAGES

If any portion of the VBM is maintained on the media, FTL must track and locate those VBM Pages. The Virtual Page Map (VPM) performs a function similar to that of the VBM. Instead of mapping the location of Virtual Data blocks, each entry in the VPM maps the location of a VBM Page. The FTL specification does not define a way to maintain the VPM on the media and therefore typically must be constructed when the media is accessed for the first time. This is accomplished by scanning the media for VBM entries and constructing a table to them in system RAM.

6.1 Page Map Handling

The BAM is scanned to locate VBM Pages and Replacement Pages and the logical locations are stored in the VPM. The total number of VBM Pages found must equal **NumVMPages**. If a VBM Page is missing and a Replacement Page exists for that page, the Replacement page is used as the original VBM Page. If a VBM Page is missing and a Replacement Page does not exist, recovery operations must be performed. If duplicate VBM Pages are found, only one is used.

VBM Pages and Replacement Pages use negative BAM entries. Therefore, the virtual addresses are also negative. A page's virtual address can be calculated by multiplying the page number by the size of a Virtual block. One important item of note is the fact that the page numbers are negative. The VBM Page would be –**NumVMPages** and –1 would be the last page.

6.2 Replacement Pages

Replacement VBM pages are identical to normal VBM pages except for the fact that replacement pages have BAM entries with the least significant seven bits ending in 0x60 while normal VBM pages have BAM entries ending in 0x40. Both types of VBM pages have BAM entries with negative values. A Replacement Page is used only if the original VBM Page has a VBM entry of 0x00000000 for the Virtual Data block being accessed. If the corresponding entry in the Replacement Page is also zero (0), then the Virtual block does not exist on the media. Replacement Pages enhance performance by minimizing the need to supersede VBM Pages when logical addresses on a Page are updated and the existing entry is already used. A replacement page may delay the need to rewrite the VBM Page. If the corresponding entry on a Replacement Page is available, then it is used and the original page need not be rewritten right away. Like normal VBM Pages, Replacement Pages are also allocated from any free Read/Write blocks in any Erase Unit.

6.2.1 REPLACEMENT PAGE MAP

FTL uses a duplicate VPM to keep track of the Replacement Pages. The Replacement VPM is quite similar to the normal VPM except that it points to the Replacement VBM pages. Each VBM Page is capable of having a Replacement Page. Even though the replacement VPM array is allocated large enough to map out a Replacement Page for every VBM Page, only one Replacement Page should be implemented. This is to guarantee compatibility and media interchangability between all FTL solutions. The Replacement Page is located via a scan of the BAM at the time the media is inserted.

NOTE:

The FTL specification in PC Card 95 allows for up to four Replacement Pages at the present time. This will be changed in future revisions to specify only one Replacement Page.

- **Related FTL Functions:**

  — Fill_VBM_PageMap

7.0 LOGICAL TO PHYSICAL MAPPING

All of the addresses stored in the VBM and VPM arrays are logical addresses. The logical address points to a location on the media described when the Erase Units are arranged in logical order. The **LogicalEUN** field indicates the logical number of each block. FTL matches the **LogicalEUN** to a PhysicalEUN by scanning the media and recording the relationship in an array at initialization/card insertion. These relationships are used to translate between physical and logical addresses. A logical address may be treated as having two distinct sections. The first, consisting of the most significant bits, refers to the LogicalEUN. The second part, consisting of the least significant bits, is an offset into that particular Erase Unit. The number of bits in each part is dependent upon the size of the Erase Unit.

- **Related FTL Functions:**
  - GetSectorAddr
  - Logical2Physical

8.0 READ

FTL processes Virtual block data read requests from the host operating system. Virtual block data is accessed with the number of a Virtual block passed from the host. The block number is used as an index into the VBM to obtain the logical address of the Read/Write block. In order to read the data, FTL translates the logical address to a physical address. If the VBM entry for the Read/Write block is 0xFFFFFFFF, then the Virtual Block does not exist and a buffer of "0"s is returned to the host. If the entry is 0x00000000, then the block's logical address may be located on the Replacement Page. If the Replacement Page does not exist or if the entry is 0x00000000, then the Virtual block does not exist and a buffer of "0"s is returned to the host.

- **Related FTL Functions:**
  - FillReadRequest

9.0 WRITE

Virtual Block data writes are the most complex part of FTL. When the host system tells FTL to perform a write operation, FTL must find a free Read/Write block on the media. If one is not found, Erase Unit recovery may take place. (See Section 10 about unit recovery). If a free Read/Write block is found, the free Read/Write block's BAM entry is set to 0xFFFFFFFE to indicate that a write is taking place in that Read/Write block. Should the media write be interrupted, FTL will be able to mark the block as deleted. Upon completion of the write operation, the BAM entry is updated to the Virtual Block's virtual address. The Virtual Block data is then written out to the Read/Write block. The VBM is updated to point to the new area assigned to this Virtual Block. If the new block is replacing an existing block, the old Read/Write block's BAM entry is set to 0x00000000.

- **Related FTL Functions:**
  - FillWriteRequest
  - Find_BAM_Entry
  - DiscardSector

10.0 UNIT RECOVERY/RECLAIM

As the media is used, it will eventually fill up. Read/Write blocks marked as deleted or superseded may only be reused after being returned to the erased state. Due to the nature of flash technology, all the Read/Write blocks in the same Erase Unit must be erased at the same time. However, it is not very often that all the Read/Write blocks in the Erase Unit are marked as deleted or superseded at the same time. The unit recovery (reclaim) process preserves the data in valid Read/Write blocks while recovering the deleted blocks in the same Erase Unit.

Reclaim requires the media to have at least one usable Transfer Unit. A Transfer Unit is an Erase Unit completely in a erased state except for its EUH. Transfer Units are used to store valid data from an Erase Unit being erased. Upon locating a properly prepared transfer unit, FTL sets the unit's **LogicalEUN** to 0x7FFF to indicate that a unit recovery is in progress. A properly prepared transfer unit is one which has the global areas of the EUH initialized and which has erased regions for Virtual Block data, VBM Pages, Replacement Pages and the BAM. Upon completion of the unit recovery procedure, the **LogicalEUN** will be changed to the **LogicalEUN** of the recovered Erase Unit. The old Erase Unit is erased and reformatted as a Transfer Unit.

- **Related FTL Functions:**
  - DoReclaim

APPENDIX A
GLOSSARY

Block Allocation Map (BAM)	This is a method of storing Block Allocation Information about the status of Read/Write Blocks.
BIOS Parameter Block (BPB)	This DOS FAT structure provides the Operating System with information about the media and the partition.
Control Units	Read/Write blocks used by FTL for storing EUH, BAM, ECC, etc.
ECC	Error Correction Code.
EDAC	Error Detection and Correction.
Erase Unit Header (EUH)	This header contains information specific to the Erase Unit and global information about the entire FTL partition.
Erase Unit	The area of the flash media handled as a single erasable unit by FTL. It may contain one or more erase zones. This area is determined during media formatting time.
Erase Zone	An area of flash which must be erased as a single unit due to the characteristics of the media.
FAT	File Allocation Table.
FTL	File Translation Layer.
Little - Endian	Method of storing data where the lowest byte address contains the least significant byte.
Logical Address	This address is based on accessing the media in a Logical Erase Unit order.
Logical Erase Unit Number (LogicalEUN)	Logical Unit Number. This is a logical number assigned to a Erase Unit by FTL. This field is contained in the EUH.
LUN	See Logical Erase Unit Number.
Memory Technology Driver (MTD)	This driver contain device specific algorithms.
Partition	A region of the flash media dedicated for use by a single file system.
Partition Boot Record (PBR)	This structure contains partition information and the BPB.

PCMCIA	Personal Computer Memory Card International Association
Physical Address	This address is based on accessing the media in a physical erase unit order.
Physical Erase Unit Number (PhysicalEUN)	This is a number assigned to an Erase Unit based on its physical location on the media. This number never changes for a Erase Unit.
Reclaim	This is the procedure for recovering deleted/superseded Read/Write Blocks for reuse. This process includes the preservation of valid data and erase the entire Erase Unit.
Read/Write Block	A subdivision of an Erase Unit. FTL tracks the allocation of each piece.
Replacement Page	VBM pages that contain superseded entries from original VBM page.
Spare Block	A erase unit reserved for reclaim usage.
Transfer Unit	See spare block.

APPENDIX B
STRUCTURES

EUH struct – Erase Unit Header Structure

Offset	Size	Field	Description
0	BYTE	LinkTargetTuple[5]	PCMCIA Link Target tuple TPL_CODE CISTPL_LINKTARGET (13H) TPL_LINK 3 TPLTG_TAG 'C', 'I', 'S'
5	BYTE	DataOrgTuple[10]	PCMCIA Data Oranization tuple TPL_CODE CISTPL_ORG (46H) TPL_LINK At least fifty-seven (57) TPLORG_TYPE TPLORGTYPE_FS (0) TPLORG_DESC "FTL100", 0
15	BYTE	NumTransferUnits	Number of transfer units in partition. Must be at least one or else media is a WORM.
16	DWORD	Reserved	Reserved
20	WORD	LogicalEUN	Logical Erase Unit Number of this block. The **LogicalEUN** of a formatted Transfer Unit is the media's erase state. (i.e., 0xFFFF).
21	BYTE	BlockSize	Size of a Read/Write Block. This value is expressed as a $\log_2$ value.
22	BYTE	EraseUnitSize	Size of a Erase Unit. This value is expressed as a $\log_2$ value. Erase Units must be a multiple of the flash device's Erase Zone size.
24	WORD	FirstPhysicalEUN	Physical Erase Unit where FTL partition begins.
26	WORD	NumEraseUnits	Total number of Erase Units in partition. This value includes Erase Units used to store data, block allocation information, checksums, transfer units, replacement pages, spare blocks, and Virtual Block Map pages.
28	DWORD	FormattedSize	Total formatted size of partition. This is the total about of free space available to the host system for file storage. This valued does not include areas marked as format blocks, transfer units, replacement pages, and Virtual Map pages. This value is in bytes and must be a multiple of BlockSize.
32	DWORD	FirstVMAddress	First virtual address physically mapped in VBM page on media. If the value in this field is zero (0), then the entire VBM is maintained on the media. If the value exceeds the FormattedSize, none of the VBM is maintained on the media. However, space must be reserved on the media for the entire VBM at all times.

EUH struct – Erase Unit Header Structure (Continued)

Offset	Size	Field	Description
36	WORD	NumVMPages	Total number of VBM pages.
38	BYTE	Flags	Special bit-mapped flags indicating checksum, block allocation information, and flash polarity.
39	BYTE	Code	Binary value designating type of EDAC, ECC, CRC, or checksum maintained for Virtual block data. No such information is present if the field is in the erase state. In the non-erase state, such information was present at a time, but is no longer maintained. For a value of one, a two- byte checksum is computed and maintained for each Virtual block.
40	BYTE	SerialNumber[4]	Partition serial number.
44	DWORD	AltEUHOffset	Offset of alternative EUH.
48	DWORD	BAMOffset	Offset of BAM from start of EUH. This value is expressed in bytes and is only valid if the Flags - **HiddenAreaFlag** is set to zero (0). The BAM need not be aligned on a Virtual Block boundary. If the Flags - **DoubleBAI** bit is set to one (1), two copies of the BAM are maintained on the media. If checksums, CRCs, or ECCs are used in the Erase Unit, these codes follow the block allocation information.
52	BYTE	Reserved[12]	Reserved for future use.

RootEntries struct–Root Directory Entry Structure

Offset	Size	Field	Description
0	BYTE	Name[8]	Name of file/directory
8	BYTE	Ext[3]	File/directory extension
11	BYTE	Attribute	File/directory attributes
12	BYTE	Reserved[10]	Reserved (not used)
22	WORD	Time	Time file/directory was created or last updated
24	WORD	Date	Date file/directory was created or last updated
26	WORD	StartCluster	Starting cluster of file/directory
28	DWORD	Size	Size of file/directory

APPENDIX C
FTL AVAILABILITY

M-System TrueFFS* *(v3.2 and up are FTL)*

M-Systems, Inc.
4655 Old Ironsides Drive
Suite 290
Santa Clara, CA 95054
Tel: (408) 654-5820
Fax: (408) 654-9107
email: info@ccm.msyscal.com
WWW: http://www.m-sys.com

M-Systems, Ltd
Atidim Industrial Park
P.O. Box 58036
Tel Aviv, Israel 61580
Tel: 972-3-647-7776
Fax: 972-3-647-6668
email: info@msys.co.il
WWW: http://www.m-sys.com

SCM SwapFTL* *(v3.0 and up are FTL)*

SCM Microsystems, Inc.
4655 Old Ironsides Drive
131 Albright Way.
Los Gatos, CA 95030
Tel: (408) 370-4888
FAX: (408) 370-4880
email: pccard@scmmicro.com
WWW: http://www.scmmicro.com

SCM Microsystems, GmbH
Muhlhauser Str. 2
Erfurt D-99092, Germany

Tel: (49) 361-66-4870
FAX: (49) 361-211-3515
email: pccard@scmmicro.com
WWW: http://www.scmmicro.com

SystemSoft FTL

SystemSoft, Inc. (Main)
2 Vision Dr.
Natick, MA 01760

Tel: (508) 651-0088
FAX: (508) 651-8188
email: wizard@systemsoft.com
WWW: http://www.systemsoft.com

SystemSoft, Inc. (West)
2350 Mission College Blvd
Suite 450
Santa Clara, CA 95054-1534
Tel: (408) 988-6756
FAX: (408) 988-6758
email: wizard@systemsoft.com
WWW: http://www.systemsoft.com

APPENDIX D
ADDITIONAL INFORMATION

Related Documents

PC Card Standard, Media Storage Formats Specification and the Flash Translation Layer Specification are available through PCMCIA:

PCMCIA
2635 North First Street
San Jose, CA 95134

Telephone: (408) 433-2273
Fax: (408) 433-9558

Related Information

FTL is based on Intellectual Property and patent(s) from M-Systems, Ltd. All rights are reserved by M-Systems. M-Systems does grant a royalty-free, non-exclusive license for the design and development of FTL-compatible drivers, file systems, and utilities using the data formats with PCMCIA PC Cards as described in the FTL Specification. Please see the PC Card standard, available from PCMCIA, for detailed information.

M-Systems provides software which incorporates the FTL standards and algorithms. For information on contacting them, or other FTL providers, please refer to Appendix C, *FTL Availability*.