IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In the *Inter Partes* Review of: Trial Number: To Be Assigned

U.S. Patent No. 5,839,108

Filed: June 30, 1997

Issued: November 17, 1998

Inventor(s): Norbert P. Daberko,
Richard K. Davis

Assignee: e.Digital Corporation

Title: Flash Memory File System In a Handheld Record and Playback Device

Panel: To Be Assigned

Mail Stop *Inter Partes* Review
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

PETITION FOR *INTER PARTES* REVIEW UNDER 35 U.S.C. § 311 AND 37 C.F.R. § 42.100

TABLE OF CONTENTS

Petitioners Micron Technology, Inc. and Micron Consumer Products Group, Inc. (collectively, "Petitioners") hereby request *inter partes* review ("IPR") of claim 1 of U.S. Patent No. 5,839,108 (the "'108 patent") (Ex. 1004).

I. CERTIFICATION PURSUANT TO 37 C.F.R. § 42.104(A) THAT THE '108 PATENT MAY BE CONTESTED BY PETITIONERS

Petitioners hereby certify that the '108 patent is available for IPR and Petitioners are not barred or estopped from requesting IPR of claim 1 on the grounds identified herein. Petitioners are not the owners of the '108 patent. Petitioners have not previously filed a civil action challenging the validity of any claims of the '108 patent. Petitioners submit this petition less than one year after first being served with a complaint alleging infringement of the '108 patent. The estoppel provisions of 35 U.S.C. § 315(e)(1) do not prohibit this IPR.

II. MANDATORY NOTICES – 37 C.F.R. § 42.8(A)(1)

A. 37 C.F.R. § 42.8(b)(1): Real Party-In-Interest

Micron Technology, Inc. and Micron Consumer Products Group, Inc. are the real parties-in-interest for Petitioners.

B. 37 C.F.R. § 42.8(b)(2): Related Matters

e.Digital Corporation ("e.Digital") has asserted the '108 patent in *e.Digital Corp. v. Micron Consumer Products Group, Inc.*, No. 3:13-cv-02907 (S.D. Cal.); *e.Digital Corp. v. Micron Technology, Inc.*, No. 3:13-cv-02944 (S.D. Cal.); *e.Digital Corp. v. Other World Computing, Inc.*, No. 3:13-cv-02915 (S.D. Cal.);

and *e.Digital Corp. v. Mushkin, Inc.*, No. 3:13-cv-02914 (S.D. Cal.). These cases may affect, or be affected by, decisions in these proceedings.

Further, Intel Corporation previously filed petitions for IPR of the '108 patent (IPR2014-01429 and IPR2014-01430). Intel, Petitioners, and others were sued for infringement of the '108 patent in the Southern District of California, but only Intel petitioned for IPR of the '108 patent. Intel has reached a settlement with the Patent Owner, and the Board has recently terminated those proceedings.

C. 37 C.F.R. § 42.8(b)(3) and (4): Lead and Back-Up Counsel and Service Information

Pursuant to 37 C.F.R. § 42.8(b)(3), Petitioners identify the following counsel and submit Powers of Attorney with this Petition. 37 C.F.R. § 42.10(b). Petitioners consent to service by e-mail at the email addresses below.

Lead Counsel for Petitioners	**Backup Counsel for Petitioners**
David M. Maiorana Reg. No.: 41,449 JONES DAY 901 Lakeside Ave. Cleveland, OH 44114 Tel: 216-586-3939 dmaiorana@jonesday.com	Matthew A. Ferry Reg. No.: 63,142 JONES DAY 12265 El Camino Real, Suite 200 San Diego, CA 92130 Tel: 858-314-1120 mferry@jonesday.com

D. PAYMENT OF FEES – 37 C.F.R. § 42.103

The undersigned authorizes the Office to charge the fee set forth in 37 C.F.R. § 42.15(a) for this Petition, or any other required fees, to Deposit Account 501432, ref: 022030-710001. Review of one (1) claim is requested, and thus no

excess claim fees are required. The undersigned further authorizes payment for any additional fees that may be due in connection with this Petition to be charged to the above-referenced Deposit Account.

III. IDENTIFICATION OF CHALLENGE – 37 C.F.R. § 42.104(b)

A. 37 C.F.R. § 42.104(b)(1) and (2)

Pursuant to 37 C.F.R. § 42.104(b)(1), Petitioners request IPR of claim 1 of the '108 patent ("Claim 1"). Pursuant to 37 C.F.R. § 42.104(b)(2), IPR of Claim 1 is requested in view of the following:

- U.S. Patent No. 6,272,610, to Katayama, *et al.* ("Katayama") (Ex. 1005); filed: Mar. 9, 1994; issued: Aug. 7, 2001; prior art under § 102(e);[1]
- U.S. Patent No. 5,696,917 to Mills, *et al.*, ("Mills") (Ex. 1006); filed: Jun. 3, 1994; issued: Dec. 9, 1997; prior art under § 102(e); and
- European Patent Application Publication No. 0 557 736 A2 to Krueger, *et al.* ("Krueger") (Ex. 1007); filed: Jan. 29, 1993; published: Sept. 1, 1993; prior art under § 102(b).

Ground	Proposed Statutory Rejections for the '108 patent
1	Claim 1 is obvious under 35 U.S.C. § 103(a) over Katayama in view of Mills.

[1] Cites to 35 U.S.C. §§ 102 and 103 are to the pre-AIA versions applicable here.

Ground	Proposed Statutory Rejections for the '108 patent
2	Claim 1 is obvious under 35 U.S.C. § 103(a) over Krueger to the extent that "a logical link between the previous logical data segment and the new data segment" is construed to be broader than "a pointer written to the previous logical data segment that points to the physical location of the new data segment." (*See* Section III.B.ii.)

B. 37 C.F.R. § 42.104(b)(3): Claim Construction

Claim 1 must be given its "broadest reasonable construction in light of the specification." 37 C.F.R. § 42.100(b). Petitioners refer to this standard as "BRI."[2]

To determine the meaning of Claim 1, it is appropriate and necessary to consider the '108 patent as well as the specification and prosecution history of its parent—U.S. Patent No. 5,787,445 (the "'445 patent") (Ex. 1008)—to which the '108 patent is a continuation-in-part.[3] *See, e.g.*, *Omega Eng'g, Inc., v. Raytek*

[2] The district court in *e.Digital Corp. v. Other World Computing, Inc*., No. 3:13-cv-02915 (S.D. Cal.); and *e.Digital Corp. v. Mushkin, Inc*., No. 3:13-cv-02914 (S.D. Cal.) rendered a claim construction decision broader than the BRI advocated in this petition. (Ex. 1016.) While Petitioners do not agree with its merits, they attach it to this petition in the interest of full disclosure.

[3] Petitioners reserve their position that Claim 1 is invalid under 35 U.S.C. § 112 for lack of written description and/or lack of enablement because the '108

Corp., 334 F.3d 1314, 1333 (Fed. Cir. 2003); *In re Rambus Inc.*, 694 F.3d 42, 48 (Fed. Cir. 2012) (applying *Omega* in a reexamination appeal); *see also Tempo Lighting, Inc. v. Tivoli, LLC*, 742 F.3d 973, 977 (Fed. Cir. 2014).

First, the '108 patent's specification does not describe Claim 1's subject matter; instead, the disclosures supporting Claim 1 can only be found in the '445 patent (if at all). For instance, the abstract, specification and figures for the '108 patent have no disclosure for key terms in Claim 1 such as "cache memory," "link(s)," "linking," and "logical link(s)."

Second, the claim terms in question here are also found in the claims of the '445 patent. *See Omega*, 334 F.3d at 1333-34. With the exception of one limitation ("industry standard data storage format"), each and every element recited in Claim 1 is found in claim 1 of the '445 patent.

Third, the patentee never evinced a desire to recapture any claim scope disclaimed in the '445 patent. *See, e.g.*, *Hakim v. Cannon Avent Grp., PLC*, 479 F.3d 1313, 1317-18 (Fed. Cir. 2007). To the contrary, the evidence suggests that the patentee intended for the statements in the '445 patent's specification and prosecution history regarding claim scope to carry forward to the '108 patent with equal force. For instance, the '108 patent's specification endorses the

---

patent fails to properly incorporate the '445 patent by reference. *See Application of De Seversky*, 474 F.2d 671, 674 (CCPA 1973).

disparagement of prior art found in the '445 patent's specification: "The ['445 patent] also addressed the drawbacks of other prior art methods of file management designed specifically for use with flash memory such as the system taught in U.S. Pat. No. 5,404,485 issued to Ban." (Ex. 1004 at 1:55-58; *see also id.* at 1:58-2:41 and Ex. 1008 at 2:41-3:21, wherein the '108 patent and '445 patent specifications identify, word-for-word, the same shortcomings in the prior art.) Furthermore, the patentee made no statements during prosecution that Claim 1 should carry a broader, or otherwise different, scope than claim 1 of the '445 patent. Having made clear statements regarding the scope of the '445 patent, the patentee should be held to such statements in the construction of the claim in question here, which includes each and every limitation from claim 1 of the '445 patent. *See Omega Eng'g, Inc. v. Raytek Corp.*, 334 F.3d 1314, 1333 (Fed. Cir. 2003).

In light of the above, Petitioners submit, for purposes of this IPR only, that the BRI of Claim 1 is as follows:

i. "non-volatile, long-term storage medium"

The BRI of the term "non-volatile, long-term storage medium" is "memory that holds its data without the need for ongoing power support." This is consistent with the plain and ordinary meaning of the phrase to one of ordinary skill in the art in view of the intrinsic record. Nothing in the intrinsic record limits this claim term to any specific type of non-volatile, long-term storage medium. (Ex. 1009 at

6 ("[W]hile this invention was motivated by flash memory limitations, ***the principle is applicable to any type of long term memory medium***, and therefore ***claim 1 has not been limited only to flash memory***.").)[4]

ii. "a logical link between the previous logical data segment and the new data segment"

The BRI of the term "a logical link between the previous logical data segment and the new data segment" is "a pointer written to the previous logical data segment that points to the physical location of the new data segment." This is consistent with the plain and ordinary meaning of the phrase to one of ordinary skill in the art in view of the specification,[5] and is supported by the intrinsic record.

The '445 patent's specification supports this construction. For example, it equates a "logical link" to a pointer to the physical location of a data segment by stating that "a path for sequentially accessing the data segments"—which, according to Claim 1, is provided by "logical link[s]" between data segments—is provided by "***pointers to absolute physical locations within flash memory***." (Ex. 1008 at 6:17-18.) The '445 patent's specification further discloses that the

---

[4] For all quotes herein, emphases are added unless otherwise noted.

[5] Pointers were well-known in the art in the mid-1990s as variables containing memory locations or addresses. (*See, e.g.*, Ex. 1010 ("pointer. . . . [A] variable that contains the memory location (address) of some data . . .").)

claimed invention requires the use of pointers to the physical location of data segments to provide logical linkage:

> [I]mplementation of the flash file system ***requires*** that each data segment have written to it a header. Within the header in predetermined fields, ***absolute physical addresses are saved***. These addresses are ***physical locations within flash memory of the next logical data segment***.
>
> * * *
>
> [T]he headers of the present invention are written so as to contain ***pointers which point to files*** which a user deems to be logically related to by subject.

(*Id.* at 6:3-8, 17:49-51.) The patentee also made clear that in a write operation (*i.e.*, writing a data segment to non-volatile memory), a logical link pointing to the subsequent data segment is created:

> In a write operation of a new data segment, a header is placed at the beginning of the segment. . . . ***The header also indicates the location of the next logically related and subsequent data segment.***

(*Id.* at 4:27-31.)

Extrinsic evidence further supports Petitioners' construction, which defines a "linked list" to be a data structure consisting of a group of nodes connected by pointers to locations in memory and a "link" to be a memory location or memory address. (*See* Ex. 1010 at 240 ("**linked list** In programming, a list of nodes or

elements of a data structure connected by pointers."); Ex. 1011 at 277 ("**link . . . 3. (pointer)** A character or group of characters that indicates the storage of an item of data. Thus when a field of an item A in a data structure contains the address of another item B, i.e. of its first word in memory, it contains a link to B.").)

Importantly, the patentee explicitly stated in the '445 patent file history that "the ***only way*** to determine the location of data is to traverse the linked list of data segments." (Ex. 1009 at 5.) Thus, Petitioners' construction of this term is correct under the BRI standard.

iii. "file system"

(1) The preamble of Claim 1 is limiting

The term "file system" appears in the preamble of Claim 1. Here, the preamble is limiting for a number of reasons. *First*, the patentee relied on the preamble during prosecution to distinguish the claimed invention from the prior art. *Catalina Mktg. Int'l, Inc. v. Coolsavings.com, Inc.*, 289 F.3d 801, 808 (Fed. Cir. 2002). During prosecution of the '445 patent, the patentee distinguished prior art references cited by the Examiner because they were applicable to "data structures" and not specifically to "***file*** structures":

> Jeffrey teaches data structures. **In contrast, the present invention teaches an operating system using linked lists *for file structures*.** The Office Action has seemingly failed to notice this distinction, and has **failed to provide any motivation for treating *files structures***

> [sic] **like data structures**.

(Ex. 1009 at 8.) This distinction is captured by the preamble and particularly the preamble term "file system"—indeed, the term "file" only appears in Claim 1 as part of the "file system" term. Thus, the patentee's arguments to overcome prior art demonstrate their "use of the preamble to define . . . the claimed invention" as a method applicable to only file systems, transforming the preamble into a claim limitation. *Catalina*, 289 F.3d at 808.

Consistent with the file history, the '445 patent's specification repeatedly makes clear that aspects of the file system are what purportedly differentiate the patent from the prior art:

> The present invention, however, realizes that data does not have to be contiguous in order to be readable in a logical or relational order. The present invention ***claims being able to manipulate data directly in flash memory because the flash file system of the present invention enables data to be read in a logical order regardless of how many segments the file is comprised of, and where these segments are saved in memory.***

(Ex. 1008 at 5:63-6:3.)

> The present invention takes a very different approach to memory management. ***This new approach, embodied in a method and apparatus, overcomes the significant drawbacks of Ban***. This is accomplished by taking advantage of the properties of flash memory, instead of treating them as a liability.

> To understand the new method, it is necessary to have an understanding of the arrangement of the underlying hardware. ***The apparatus of the present invention is shown in block diagram form in FIG 3A***.
>
> ***FIG. 3A is a block diagram of the components of a preferred embodiment of the present invention which utilizes the file system of the present invention***.

(*Id.* at 7:62-8:4; 4:53-55.)

Second, the patentee repeatedly stressed the importance of the "file system" to the alleged invention, further confirming that the term is limiting. *See Rotatable Techs. LLC v. Motorola Mobility LLC*, No. 2014-1042, 2014 WL 2898532, *1 (Fed. Cir. June 27, 2014). (*See, e.g.*, Ex. 1004 at 3:34-43; Ex. 1008 at 5:33-59.)

The '445 patent's detailed description of the invention describes "[t]he file system of the present invention" (Ex. 1008. at 8:33) and that "[t]he present invention also provides a file system which appears to have significant RAM resources." (*Id.* at 8:61-62.) Thus, the "file system" term "states the framework of the invention," *On Demand Mach. Corp. v. Ingram Indus., Inc.*, 442 F.3d 1331, 1343 (Fed. Cir. 2006), and is thus limiting because it gives life, meaning, and vitality to the claims.[6] *Catalina*, 289 F.3d at 808; *see also C.W. Zumbiel Co. v.*

[6] Notably, the preamble of claim 1 of the '445 patent, which is identical to Claim 1, provides the only antecedent basis for the term "file system" in the bodies of

Kappos, 702 F.3d 1371, 1385 (Fed. Cir. 2012) (applying *Catalina* under BRI to find the preamble limiting). Therefore, the Board should find the preamble of Claim 1, and specifically the "file system" term, limiting.

(2) The patentee claimed to have invented a "file system" that does not use file allocation tables (FATs) or memory maps

The BRI of "file system" is: "system to organize and keep track of files without using file allocation tables (memory maps)."

A "file system" is a term commonly used in the art and generally means a "system [] to organize and keep track of files." (Ex. 1012.) However, in the '108 and '445 patents, the patentee made clear that they are disclosing a "file system" that is different from the prior art by both defining it in the '445 patent's specification and clarifying this definition in statements made during prosecution.

The patentee unambiguously made clear that the file system of the present invention does not use memory maps or file allocation tables (FAT):[7]

---

dependent claims 3, 5, 7, and 8, further confirming this term is a limitation. *See Catalina*, 442 F.3d at 808 (relying on a preamble term "for antecedent basis may limit claim scope because it indicates a reliance on both the preamble and claim body to define the claimed invention").

[7] *In re Abbott Diabetes Care Inc.*, 696 F.3d 1142, 1149 (Fed. Cir. 2012) (patentee disclaimed an "electrochemical sensor" with wires because the "specification

> It is yet another object to provide ***a file system*** which further reduces RAM requirements ***by replacing a memory map with logically linked serial data segments***.
>
> * * *
>
> Still another object is to provide ***a file system*** which uses absolute physical memory addresses ***to avoid the additional overhead created by memory mapping***.

(Ex. 1008 at 3:47-49, 3:57-59.).

Indeed, to illustrate the benefits of avoiding memory maps, the '445 patent's specification repeatedly disparages the teachings of the prior art reference U.S. Patent No. 5,404,485 ("Ban"), which used memory maps. *See* Ex. 1008 at 7:20-25, 51-55, 62-65; *see also* Ex. 1004 at 1:55-2:32.

Furthermore, the patentee confirmed that the invention does not use a FAT (memory map) in statements to the Patent Office during '445 patent's prosecution:

> The present invention enables the <u>elimination</u> of a FAT (memory map) . . . [T]he ***only way*** to determine the location of data is to traverse the linked list of data segments.
>
> * * *

---

contains only disparaging remarks with respect to the external cables and wires of the prior-art sensors" and "every embodiment disclosed in the specification shows an electrochemical sensor without external cables or wires").

> The FAT as described in Jeffrey is ***the same FAT (or virtual memory map) described in Ban*** . . . which ***the present invention has taken great pains from which to distinguish itself***.
>
> * * *
>
> Unlike Ban, the present invention teaches how data can be manipulated directly . . . ***without having to use a FAT***.

(Ex. 1009 at 5 (underline in original), 6, 8.) (*See Tempo Lighting*, 742 F.3d at 977 (looking to prosecution history for the meaning of a disputed claim term on appeal of an *inter partes* reexamination, explaining that "the prosecution history, while not literally within the patent document, serves as intrinsic evidence for purposes of claim construction" and that "[t]his remains true in construing patent claims before the PTO.").) Because the patentee made clear that their "file system" does not use file allocation tables (memory maps), Petitioners' proposed construction of "file system," a "system to organize and keep track of files without using file allocation tables (memory maps)," is the BRI of the term.

iv. "primary memory"

The BRI of the term "primary memory" is "main memory of a computer system, *i.e.*, the main general-purpose storage to which the computer's microprocessor has direct access." This is consistent with the plain and ordinary meaning to one of ordinary skill in the art at the time of the alleged invention. (*See* Ex. 1010 at 250 ("**main memory** *See* primary storage"); 314 ("**primary storage**

Random access memory (RAM); the main general-purpose storage region to which the microprocessor has direct access. A computer's other storage options, such as disks and tape, are called secondary storage or (sometimes) backup storage."); Ex. 1011 at 292 (treating "main memory," "main store," "main storage," "RAM," and "primary memory" as synonyms); 393 ("**primary memory** *Another name for* main memory, specifically the form used as the medium for storing instructions and data that are currently undergoing processing by a CPU.").)

This meaning is also consistent with the patentee's use of term. (*See, e.g.*, Ex. 1004 at 4:5-9 ("The present invention also includes a method of memory management for a primary memory created from non-volatile, long-term storage media, in particular flash memory, which enables direct manipulation of data segments stored therein"); Ex. 1008 at Title ("Operating System Including Improved File Management For Use in Devices Utilizing Flash Memory as Main Memory"), 2:5-8 ("Therefore, it would be advantageous to be able to replace RAM with a long-term storage medium when the substantial benefits of non-volatile data retention are required"), 5:45-6:11 (distinguishing Ban from the "present invention" that "claims being able to manipulate data directly in flash memory").)

v. "a path for sequentially accessing the data segments within the primary memory"

The BRI of the term "a path for sequentially accessing the data segments within the primary memory" is "a linked list used instead of a file allocation table

(memory map) for sequentially accessing data segments within the primary memory."

As an initial matter, throughout the intrinsic record, the patentee made clear that a linked list of data segments is used ***instead of*** a FAT (memory map) and indeed enables the ***elimination*** of a FAT. *See* Section III.B.iii, *supra*. As described above, the patentee explained that "logically linked serial data segments" ***replace*** a memory map, (Ex. 1008 at 3:47-49), and the linked list of data segments is the "***only way*** to determine the location of data" (Ex. 1009 at 5). *See* Section III.B.iii, *supra*. Because the linked list is the "only" way to determine the location of data, it is clearly used instead of a FAT (memory map).

Furthermore, in computer science and engineering, elements, such as data segments, that are linked together are called a linked list. (*See* Ex. 1011 at 277 ("**linked list (chained list)** A list representation in which items are not necessarily sequential in storage. Access is made possible by the use in every item of a link that contains the address of the next item in the list."); *see also* Ex. 1010 at 240 ("**linked list** In programming, a list of nodes or elements of a data structure connected by pointers.").) Indeed, this is how the patentee referred to the path of data segments in the '445 patent file history:

> To increase efficiency, the only way to determine the location of data is to traverse ***the linked list of data segments***. ***The linked list not only tells where the related data segments are located for one file, but it***

> ***also links the first data segment of all files together***.
>
> * * *
>
> The first file contains an address not only of the next logical data segment, but to the address of the first logical data segment of the second file. Accordingly, ***the linked lists not only preserve continuity of discontiguous but logically related data segments, they also preserve continuity to previous and subsequent but unrelated files having their own discontiguous but logically related data segments***."

(Ex. 1009 at 5-6, 9.) In other words, the patentee explicitly describes the logically connected data segments as a "linked list." Thus, the path for sequentially accessing the data segments is a linked list.

vi. "industry standard data storage format"

The BRI of the term "industry standard data storage format" is "format in which data is stored that conforms to an industry standard." This is consistent with the plain and ordinary meaning of the phrase to one of ordinary skill in the art and is supported by the '108 patent's specification. For example, the '108 patent discusses industry standard formats for recording or storing data in flash memory. (*See, e.g.*, Ex. 1004 at Abstract ("a flash memory module which can record data according to industry standard formats"), 3:19-21.) The disclosure specifically points to "industry standards such as MPEG-2," an industry standard format for storing audio and/or video information. (*Id.* at 10:53-56.) Nothing in the intrinsic record limits this claim term to any specific type of hardware or interface for

retrieving the stored data.

C. 37 C.F.R. § 42.104(b)(4): How the Claim is Unpatentable

An explanation of how properlyconstrued Claim 1 is unpatentable is in Section IV.D.

D. 37 C.F.R. § 42.104(b)(5): Evidence Supporting Challenge

An Appendix of Exhibits is attached. Relevance of the evidence, including identification of the specific portions of the evidence that support the challenge, may be found in Section IV.B. Petitioners submit a declaration of Dr. R. Jacob Baker in support of this Petition in accordance with 37 C.F.R. § 1.68. (Ex. 1001.)

IV. THERE IS A REASONABLE LIKELIHOOD THAT CLAIM 1 IS UNPATENTABLE

A. Brief Description of the Technology at Issue

The ’108 patent is directed generally to audio recording devices using non-volatile memory, such as flash. (Ex. 1004 at Abstract.) Claim 1 relates to a specific way for a file system to store data in non-volatile memory. (Ex. 1001 ¶55.)

Both the ’108 patent and its parent, the ’445 patent, contrast the claimed file system with that disclosed by prior art Ban, which teaches creating a “virtual memory map” for “converting virtual addresses to physical addresses.” (Ex. 1008 at 2:52-61.) According to the ’108 and ’445 patents, the use of this “indirection” in memory such as flash memory causes “severe overhead burdens.” (*Id.* at 2:50; Ex. 1004 at 2:1-13.) Instead, the ’445 patent teaches a file system that logically

links data segments "by creating headers which contain pointers to absolute physical locations within flash memory." (Ex. 1008 at 6:16-17.) Thus, the alleged invention eliminates the use of memory maps and instead uses logical links stored in the headers to provide "a logical path to the data segments." (*Id.* at 6:18.)

B. Description of the Alleged Invention of Claim 1

Independent claim 1 of the '108 patent relates to the previously described memory management for a file system with three main steps. Claim 1 recites:

> 1. A method of memory management for a primary memory created from a non-volatile, long-term storage medium, said method enabling direct manipulation of contiguous and non-contiguous discrete data segments stored therein by a file system, and comprising the steps of:
>
> (a) creating the primary memory from a non-volatile, long-term storage medium, wherein the primary memory comprises a plurality of blocks in which the data segments are to be stored;
>
> (b) coupling a cache memory to the primary memory, said cache memory providing temporary and volatile storage for at least one of the data segments;
>
> (c) writing a new data segment from the cache memory to the primary memory by linking said new data segment to a sequentially previous logical data segment by the following steps:
>
> (1) receiving the new data segment in the cache memory;
>
> (2) moving the new data segment from the cache memory to a next available space within primary memory such that the new data

segment is stored in primary memory in non-used memory space;

(3) identifying the previous logical data segment in primary memory;

(4) creating a logical link between the previous logical data segment and the new data segment such that the logical link provides a path for sequentially accessing the data segments within the primary memory;

(5) creating additional serial and logical links as subsequent new data segments are written to primary memory, said logical links providing the path for serially accessing the data segments regardless of contiguity of the data segments relative to each other within the primary memory; and

(6) storing the data segments to primary memory in a manner consistent with an industry standard data storage format while retaining linking between data segments created in previous steps.

Figure 3A of the ’445 patent illustrates the relationship between the cache memory recited in Claim 1 and the primary memory (*e.g.*, the flash memory).

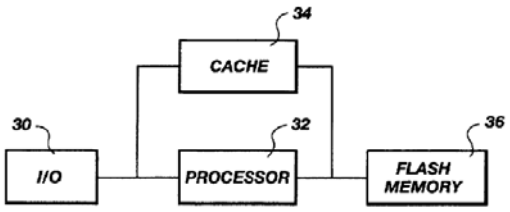
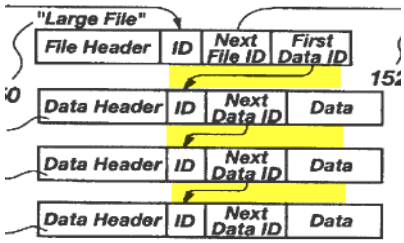


Fig. 3A

(Ex. 1008 at Fig. 3A.) Data segments are received in cache memory and then moved to a next available space in primary memory. A link is then created in

primary memory to link this new data segment to previous data segments.

Figure 7A of the '445 patent, a diagram showing the "data structure linkage" of the alleged invention, further illustrates the claimed subject matter. The following excerpt of Figure 7A highlights the "logical links" created in the writing step that connect and provide an access path to the data segments.



(Ex. 1008 at Fig. 7A (annotated).)

C. Prosecution History

As discussed in Section III.B above, the '445 patent's prosecution history is highly relevant to Claim 1. During prosecution of the '445 patent, the PTO rejected claim 1 as obvious in view of U.S. Patent No. 5,586,291 and J. ESAKOV & T. WEISS, *Data Structures – An Advanced Approach Using C* (1989). (Ex. 1013 at 3-4.) The patentee traversed the rejection, arguing that the combination of cited references used a virtual memory map and thus taught away from the alleged invention. (Ex. 1009 at 6.) The patentee explained that the alleged invention obviated the need for a memory map and instead provided access to data through the use of a linked list file structure. In particular, "[t]o increase efficiency, the ***only way*** [in the alleged invention] to determine the location of data is to traverse

the linked list of data segments." (*Id.* at 5.) The patentee also distinguished the alleged invention from the cited prior art references and Ban because in the invention "data can be manipulated directly in flash memory." (*Id.* at 8.) The patentee further distinguished the linked list of the alleged invention, which is a "linked list for file structures," from prior art linked lists for "data structures." (*Id*.) In other words, the patentee made clear their linked list is used in the specific context of a file system, and not for ***any*** data structure linkage. In light of these arguments, the claim was allowed. (*See* Ex. 1014.)

D. Explanation of Grounds for Unpatentability and Claim Charts

Petitioners provide a detailed discussion of how each asserted prior art reference invalidates Claim 1. For each of the references, Petitioners underline certain portions of the text and add emphasis to figures presented in the following claim charts. Petitioners note, however, that the surrounding text, though not underlined, is also relevant to Petitioners' challenge, as described herein.[8]

i. Claim 1 Is Obvious Over Katayama In View Of Mills

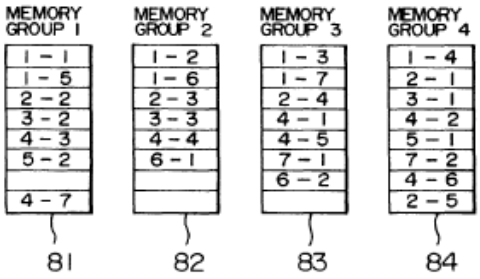
Katayama teaches a semiconductor file memory device that uses flash memory in place of traditional magnetic disk memory in conjunction with "information processing systems." (Ex. 1005 at 3:8-14.) The "information

[8] While the analysis below proceeds with the BRI presented herein, the asserted art invalidates Claim 1 under the district court's construction as well.

processing systems" that can be used with the flash memory storage medium can be personal computers, including portable devices such as PDAs and notebook computers. (*Id.* at 1:16-35.)

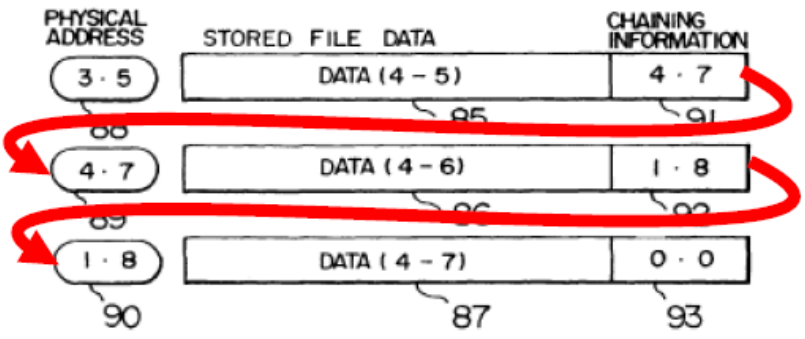
The memory device in Katayama allows for writing concurrently to four "memory groups" comprising flash memory chips. (*Id.* at 4:53-55.) Figure 5B of Katayama, below, shows an example of the memory contents of the memory device. (*Id.* at 10:43-46.)

FIG. 5B



(*Id.* at fig. 5B.) In figure 5B, the symbol m-n indicates a file number m and a sector number n of the file; for example, data 4-7 in memory group 1 is sector number 7 of file number 4. (Ex. 1005 at 10:46-48.) Data is chained, or logically-linked, together, as shown in figure 6:

FIG. 6



(*Id*. at fig. 6 (annotated).) Data 4-5 (representing sector 5 for file 4) has a corresponding chaining information that points to the physical address of the next sector of data, 4-6. (*Id*. at 14:53-57, 14:67-15:9.)

Mills teaches "computer systems that use a large-bock erasable non-volatile semiconductor memory as main memory." (Ex. 1006 at 1:9-11.) Mills teaches that most computer systems include "a main memory made of volatile memory" and a "relatively inexpensive, non-volatile memory store such as a floppy disk or hard disk." (*Id.* at 3:3:13-20.) The computer typically executes a program out of the volatile, main memory "because the non-volatile memory has a relatively slow access speed." (*Id.* at 3:19-20.) To execute a program, the computer must create a "shadow copy" of the program from non-volatile storage into the volatile main memory, and then copy the program back into secondary storage. (*Id.* at 3:18-28.)

To overcome the speed limitation of copying programs and data back and forth from secondary storage to main memory, Mills teaches that "flash memory can serve as the main memory within portable computers, providing user functions similar to those of disk-based systems." (*Id.* at 5:50-52.) Because the computer can execute directly out of flash, users enjoy "virtually instant-on performance and in-place code execution." (Ex. 1006 at 5:67-6:1.) Mills further teaches that

> Flash Memory is exceptionally well-suited to serve as a solid-state disk or a cost-effective and highly reliable replacement for DRAMs and battery-backed static RAMs. Its inherent advantages over these

> technologies make it particularly useful in portable systems that require the utmost in low power, compact size, and ruggedness while maintaining high performance and full functionality.

(*Id.* at 6:21-27.) In addition, "a flash-based nonvolatile main memory . . . reduces or eliminates the lengthy process of obtaining information from disk when power is turned on. Therefore flash main memory based computer system 100 has higher system performance when a program is initially executed than would a volatile main memory based computer system." (*Id.* at 9:10-15.) Mills also teaches using flash as main memory avoids "the duplication of shadowing information on both disk and RAM . . . thereby reducing memory cost by eliminating memory duplication." (*Id.* at 9:17-19.) In addition, "power consumption is reduced because battery backup of volatile memory is eliminated and because disk accesses are minimized or eliminated." (*Id.* at 9:22-24.)

One of ordinary skill in the art would have been motivated to combine Mills' teaching of using flash memory as the computer system's main memory with Katayama's file system for flash memory. Specifically, both references are directed to using flash memory as a way to increase a computer's processing speed. (Ex. 1001 ¶¶ 97, 103-104; *In re Hyon*, 679 F.3d 1363, 1366 (Fed. Cir. 2012) (finding motivation to combine where the prior art references were directed

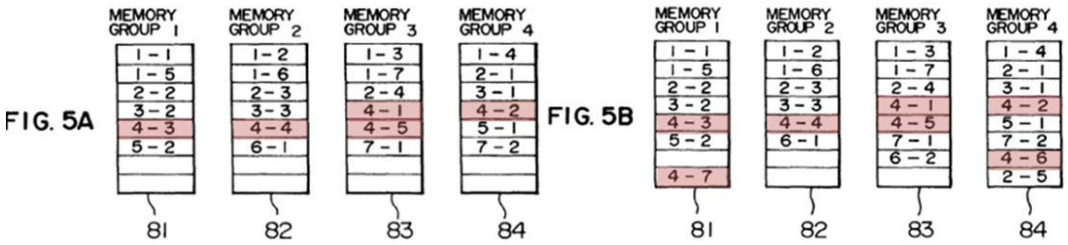
to the "same class of products").)[9] Moreover, one of ordinary skill in the art would recognize Mills's improvement provides for a system that is faster, more powerful, and requires less power than traditional computer systems. (Ex. 1001 ¶ 105; *see, e.g.*, *Sundance, Inc. v. DeMonte Fabricating Ltd*., 550 F.3d 1356, 1367 (Fed. Cir. 2008) (finding a patent invalid as obvious where a combination of two prior art references would have resulted in a design having the same benefit and improvement over the first prior art reference as those disclosed by the second prior art reference).)[10]

Claim 1, Preamble: Katayama in combination with Mills discloses a "method of memory management for a primary memory created from a non-volatile, long-term storage medium, said method enabling direct manipulation of contiguous and noncontiguous discrete data segments stored therein by a file system." (*See* Ex. 1001 ¶¶108-115.) Mills teaches using flash memory as the computer system's main memory. (Ex. 1006 at 1:8-11.) For the reasons above, it

[9] *See, e.g., Randall Mfg. v. Rea*, 733 F.3d 1355, 1362 (Fed. Cir. 2013) ("As *KSR* established, the knowledge of . . . an artisan [of ordinary skill] is part of the store of public knowledge that must be consulted when considering whether a claimed invention would have been obvious.").

[10] Should e.Digital put forth any allegations of secondary considerations of non-obviousness, Petitioners ask for an opportunity to respond.

would have been obvious to one of ordinary skill in the art to combine Mills's use of flash memory as a computer's main memory with Katayama's file system. Katayama teaches a semiconductor file memory device that uses flash memory. (Ex. 1005 at Abstract.) Files are managed and stored in flash memory as sets of file sectors that are chained together. (*Id.* at fig. 5B, fig. 6, 14:53-57, 14:67-15:9.) For example, figures 5A and 5B show how data is stored:

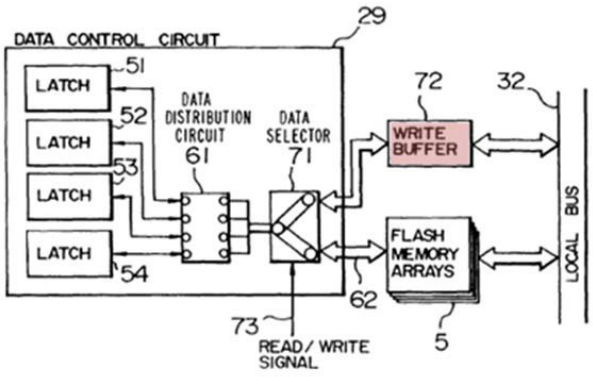


(*Id.* at figs. 5A, 5B (annotated).) Figure 5A shows that file 4 is made up of five sectors, 4-1 through 4-5 (in red). File sectors such as 4-3, 4-4, and 4-5 are stored contiguously. (*Id.* at 13:17-24.) Figure 5B shows that file 4 has been expanded to add sectors 4-6 and 4-7. (*Id.* at 14:32-37.) File sector 4-7 is stored non-contiguously. (*Id.* at 15:14-16.) The file sectors are chained together using the chaining information, which points to the physical address of the next sector of the data file. (*Id.* at fig. 6, 14:53-57, 14:67-15:9.) Thus, the semiconductor file memory device of Katayama allows for "continuous access to a file even if it is not stored in continuous locations." (*Id.* at 15:14-16.) By using the chaining information to link data sectors, Katayama's semiconductor memory device allows a file that is made up of data sectors to be expanded or modified without

modifying every data sector of the file, thus enabling direct manipulation.

Step (a): Katayama in combination with Mills discloses "creating the primary memory from a non-volatile, long-term storage medium, wherein the primary memory comprises a plurality of blocks in which the data segments are to be stored." (*See* Ex. 1001 ¶¶ 116-117.) Mills teaches using non-volatile flash memory as the systems' primary memory. (Ex. 1006 at 1:8-11.) Katayama discloses a non-volatile flash memory (Ex. 1005 at Abstract), which stores data in file sectors in the flash memory, as shown in figures 5A and 5B, above for the preamble. (*Id*. at figs. 5A, 5B; 13:17-19.)

Step (b): Katayama in view of Mills discloses "coupling a cache memory to the primary memory, said cache memory providing temporary and volatile storage for at least one of the data segments." (*See* Ex. 1001 ¶¶ 85-87, 118-121.) Katayama discloses a write buffer that temporarily stores data before it is written to the flash memory. (Ex. 1005 at fig. 4, 12:55-13:4.) As shown in figure 4 below, the memory device in Katayama includes a write buffer that is for "holding write data temporarily" "so that write data from the system is held in the write buffer 72, and thereafter . . . written to the flash memory arrays 5." (*Id*. at 12:63-65, 13:1-4.)



(*Id*. at fig. 4 (annotated to show the write buffer in red).)

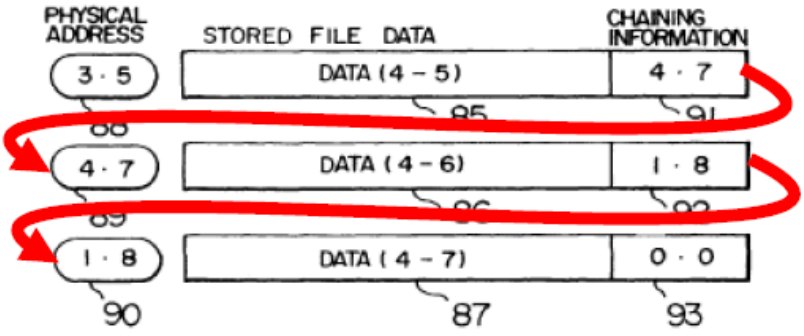
Step (c): Katayama in view of Mills discloses "writing a new data segment from the cache memory to the primary memory by linking said new data segment to a sequentially previous logical data segment." (*See* Ex. 1001 ¶¶ 85-90, 122-123.) As explained above for step (b), Katayama discloses writing data from the write buffer to the flash memory. As explained further in Steps (c)(3) and (c)(4) below, Katayama in view of Mills also discloses linking a new file sector to a sequentially previous logical file sector, using chaining information. (Ex. 1005 at fig. 6, 14:53-57, 14:67-15:9.)

Step (c)(1): Katayama in view of Mills discloses "receiving the new data segment in the cache memory." (*See* Ex. 1001 ¶ 124.) As described in Step (b), Katayama discloses a write buffer for "holding write data temporarily" "so that write data from the system is held in the write buffer 72, and . . . written to the flash memory arrays 5." (Ex. 1005 at 12:63-65, 13:1-4.)

Step (c)(2): Katayama in view of Mills discloses "moving the new data segment from the cache memory to a next available space within primary memory such that the new data segment is stored in primary memory in non-used memory space." (*See* Ex. 1001 ¶¶ 125-28.) Katayama discloses moving the new file sector from the write buffer to the flash memory. (Ex. 1005 at 13:1-4.) The new file sector is written to a vacant sector. (*Id.* at 14:25-41; *see also id.* at figs. 5A, 5B (showing that new sectors 6-2, 4-6, 4-7, and 2-5 are written to vacant sectors).) Mills teaches using non-volatile flash memory as the systems' primary memory. (Ex. 1006 at 1:8-11.)

Step (c)(3): Katayama in view of Mills discloses "identifying the previous logical data segment in primary memory." (*See* Ex. 1001 ¶¶ 85-90, 129.) The flash memory stores files as a chain of file sectors. (Ex. 1005 at fig. 6, 14:53-15:9.) Figure 6 below shows that the sectors of a file are chained together. Data 4-5 has corresponding chaining information that points to the physical address of the next sector of the data file, 4-6. (*Id*. at 14:53-57, 14:67-15:9.)

FIG. 6



(*Id.* at fig. 6 (annotated to show that data 4-5 is chained to data 4-6, which is chained to data 4-7, through the chaining information).) Thus, when a new file sector (*e.g.*, data 4-7) is written to the flash memory, the previous file sector (*e.g.*, data 4-6) must be identified. (Ex. 1001 ¶ 129.)

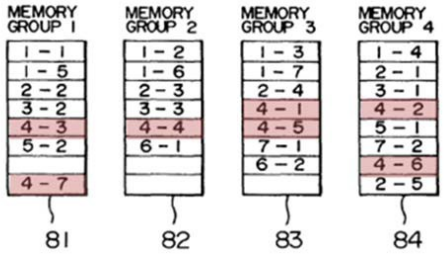
Step (c)(4): Katayama in view of Mills discloses "creating a logical link between the previous logical data segment and the new data segment such that the logical link provides a path for sequentially accessing the data segments within the primary memory." (*See* Ex. 1001 ¶¶ 85-90, 130.) As shown above in Step (c)(3), figure 6 shows that the sectors of a file are chained together. Data 4-5 includes "chaining information that points [to] the physical address of the next sector of the file data." (Ex. 1005 at 14:67-15:5.) The chaining information for data 4-5 is physical address 4 • 7. Figure 6 shows that data 4-6 is stored at that physical address. Thus, Katayama discloses creating a logical link between the new file sector and the previous file sector.

Step (c)(5): Katayama in view of Mills discloses "creating additional serial and logical links as subsequent new data segments are written to primary memory, said logical links providing the path for serially accessing the data segments regardless of contiguity of the data segments relative to each other within the primary memory." (*See* Ex. 1001 ¶¶ 85-90, 131-33.) As shown above in Step(c)(3), figure 6 discloses coupling additional file sectors together,

such as chaining sectors 4-5, 4-6, and 4-7. (Ex. 1005 at 14:53-57, 14:63-15:7.)

Figure 5B shows that the file sectors can be stored non-contiguously:

FIG. 5B



(*Id*. at fig. 5B (annotated); *see also id*. at 15:14-16.) For example, file sector 4-6 is stored non-contiguously to sector 4-5. Figure 6, above in Step (c)(3), shows that sectors 4-5 and 4-6 are nevertheless linked together using the chaining information. Thus, "it becomes possible to make a continuous access to a file even if it is not stored in continuous locations." (*Id*. at 15:14-16.)

Step (c)(6): Katayama in view of Mills discloses "storing the data segments to primary memory in a manner consistent with an industry standard data storage format while retaining linking between data segments created in previous steps." (*See* Ex. 1001 ¶¶ 134-37.) Katayama teaches that its semiconductor memory device is used in notebook and palm-type personal computers to replace magnetic-type disk memory, which are "not reliable against vibrations and consume too much power." (Ex. 1005 at 1:28-38; 2:36-40.) Mills teaches using non-volatile flash memory as the systems' primary memory. (Ex. 1006 at 1:8-11.) Mills also teaches that "applications are executed directly from the random access non-volatile memory." (*Id.* at 6:51-52.) Specifically, Mills teaches that the user can

install "word processing" programs, "spreadsheet or graphics design program[s]," or even "a particular computer game that is the current favorite." (*Id.* at 11:7-20.) One of ordinary skill in the art would understand these programs to create data to be stored in an industry standard data storage format. (Ex. 1001 ¶¶ 136-37.)

If the Board determines the proper construction of "data storage format" is "file storage for data," Katayama in view of Mills also discloses this limitation under this construction. Specifically, Mills teaches the computer system's compatibility with DOS and Windows. (*See* Ex. 1006 at 6:5-20.) Indeed, "the major parts of the DOS, Windows, and word processing programs will be resident in non-volatile flash memory 230 and can be executed in place directly from non-volatile main memory 230 as soon as the system powers up." (*Id.* at 10:18-22.)

Furthermore, for the reasons stated above with respect to Steps (c)(1)-(5), industry standard data format files stored in the combination's semiconductor memory device would have been stored in a manner that "retain[ed] linking between data segments." As a result, Katayama in view of Mills discloses the limitation "storing the data segments to primary memory in a manner consistent with an industry standard data storage format while retaining linking between data segments created in previous steps."

The claim chart below demonstrates how Claim 1 is obvious over Katayama in view of Mills.

Claim 1	Katayama in view of Mills
[1] A method of memory management for a primary memory created from a non-volatile, long-term storage medium, said method enabling direct manipulation of contiguous and non-contiguous discrete data segments stored therein by a file system, and comprising the steps of:	Katayama in view of Mills discloses a method of memory management for a primary memory created from a non-volatile, long-term storage medium (*e.g.*, flash memory), said method enabling direct manipulation of contiguous and non-contiguous discrete data segments stored therein by a file system (*e.g.*, contiguous and non-contiguous file sectors that are chained together). "[T]he present invention relates to computer systems that use a large-block erasable non-volatile semiconductor memory as main memory." (Ex. 1006 at 1:8-11.) "Therefore, one object of the present invention is to provide an efficient memory hierarchy based on non-volatile memory versus volatile memory wherein both data and applications are stored in random access non-volatile memory and further wherein applications are executed directly from the random access non-volatile memory." (*Id.* at 6:47-52.) *See also id.* at figs. 1, 2. "A semiconductor file memory device, and an information processing system incorporating the device, uses flash memories to achieve fast file access performance." (Ex. 1005 at Abstract.) "Accordingly, file-memory-based semiconductor memory chips have been used in place of magnetic-type disk memory because the latter is not ideally suited to a notebook computer environment, i.e. They are not reliable against vibrations and consume too much power." (*Id*. at 1:30-35.) "For the smooth storing and reading of files, the file management system may be designed such that the system specifies access sectors by setting a starting sector and the number of sectors on a hardware basis, and the file system controls their physical storage locations." (*Id*. at 14:41-46.) "FIG. 5A shows the memory contents after some files have

Claim 1	Katayama in view of Mills
	been stored, and FIG. 5B shows the memory contents, with some files being revised to have increased sizes. In FIG. 5A, files are initially stored closely in the ascending order of the file number, with a memory group being assigned to each sector number sequentially and cyclically. In this manner of storing file sectors closely and sequentially so that the memory groups have no vacant sector, as shown in FIG. 5A, if a file has an increased size due to revision, it could not be stored in physically continuous locations as shown in FIG. 5B. Even in such a case, file sectors are stored such that the assigned memory groups are continuous. For example, a file with file number 4 made up of five sectors 4-1 through 4-5 is initially stored as shown in FIG. 5A, and the file has additional two sectors 4-6 and 4-7 in FIG. 5B." (*Id*. at 14:18-35.) "FIG. 6 shows an example of the chaining information. Indicated by 85 is stored file data having file number 4 and sector number 5. 86 is file data of the next sector having file number 4 and sector number 6, and 87 is file data of the next sector having file number 4 and sector number 7." (*Id*. at 14:53-57.) *See also id*. at figs. 5A, 5B, 6.
[1a] (a) creating the primary memory from a non-volatile, long-term storage medium, wherein the primary memory comprises a plurality of blocks in which the data segments are to be stored;	Katayama in view of Mills discloses creating a primary memory from a non- volatile, long term storage medium (*e.g*., a semiconductor file memory device with a flash memory array that replaces magnetic-type disk memory), wherein the primary memory comprises a plurality of blocks (*e.g*., sectors) in which the data segments are to be stored. "[T]he present invention relates to computer systems that use a large-block erasable non-volatile semiconductor memory as main memory." (Ex. 1006 at 1:8-11.) "Therefore, one object of the present invention is to provide an efficient memory hierarchy based on non-volatile memory versus volatile memory wherein both data and applications

Claim 1	Katayama in view of Mills
	are stored in random access non-volatile memory and further wherein applications are executed directly from the random access non-volatile memory." (Ex. 1006 at 6:47-52.) *See also id.* at figs. 1, 2. "A semiconductor file memory device, and an information processing system incorporating the device, uses flash memories to achieve fast file access performance." (Ex. 1005 at Abstract.) "Accordingly, file-memory-based semiconductor memory chips have been used in place of magnetic-type disk memory because the latter is not ideally suited to a notebook computer environment, i.e. They are not reliable against vibrations and consume too much power." (*Id*. at 1:30-35.) "FIGS. 5A and 5B show the assignment of unit areas (one sector) of each memory group to sectors of files as a result of the software-based file storing operation." (*Id*. at 13:17-19.) *See also id.* at figs 5A, 5B.
[1b] (b) coupling a cache memory to the primary memory, said cache memory providing temporary and volatile storage for at least one of the data segments;	Katayama in view of Mills discloses coupling a cache memory (*e.g*., write buffer) to the primary memory (*e.g*., flash memory), where the cache memory provides temporary and volatile storage for at least one of the data segments. "In effect, a portion of SRAM 240 can be used as a write cache for flash main memory 230." (Ex. 1006 at 11:39-40.) "At a write access, data from the system is written temporarily to the write buffer and thereafter it is written to the flash memory." (Ex. 1005 at 12:55-57.) "In response to a write access, the data distribution circuit 61 is connected to the write buffer 72 so that write data from the system is held in the write buffer 72, and thereafter it is written to the flash memory arrays 5 by the controller." (*Id*. at 13:1-4.) *See also id*. at fig. 4.

Claim 1	Katayama in view of Mills
[1c] (c) writing a new data segment from the cache memory to the primary memory by linking said new data segment to a sequentially previous logical data segment by the following steps:	Katayama in view of Mills discloses writing a new data segment (e.g., file sector) from cache memory (*e.g.*, write buffer) to the primary memory (*e.g.*, flash memory) by linking said new data segment to a sequentially previous logical data segment (*e.g.*, through the chaining information). "In effect, a portion of SRAM 240 can be used as a write cache for flash main memory 230." (Ex. 1006 at 11:39-40.) "At a write access, data from the system is written temporarily to the write buffer and thereafter it is written to the flash memory." (Ex. 1005 at 12:55-57.) "In response to a write access, the data distribution circuit 61 is connected to the write buffer 72 so that write data from the system is held in the write buffer 72, and thereafter it is written to the flash memory arrays 5 by the controller." (*Id*. at 13:1-4.) "FIG. 5A shows the memory contents after some files have been stored, and FIG. 5B shows the memory contents, with some files being revised to have increased sizes." (*Id*. at 10:43-46.) "FIG. 6 shows an example of the chaining information. Indicated by 85 is stored file data having file number 4 and sector number 5. 86 is file data of the next sector having file number 4 and sector number 6, and 87 is file data of the next sector having file number 4 and sector number 7." (*Id*. at 14:53-57.) "Indicated by 88 is the physical address of the file data 85, with its left-hand numeral '3' indicating the memory group number and its right-hand numeral '5' indicating the address within the memory group. . . . 91 is chaining information that points the physical address of the next sector of the file data 85, i.e., the physical address of the file data 86 in this case. . . ." (*Id*. at 14:63-15:9.)

Claim 1	Katayama in view of Mills
	See also id. at figs. 4, 5A, 5B, 6.
[1c1] (1) receiving the new data segment in the cache memory;	Katayama in view of Mills discloses receiving the new data segment in cache memory (*e.g*., write buffer). "In effect, a portion of SRAM 240 can be used as a write cache for flash main memory 230." (Ex. 1006 at 11:39-40.) "At a write access, data from the system is written temporarily to the write buffer and thereafter it is written to the flash memory." (Ex. 1005 at 12:55-57.) "In response to a write access, the data distribution circuit 61 is connected to the write buffer 72 so that write data from the system is held in the write buffer 72, and thereafter it is written to the flash memory arrays 5 by the controller." (*Id*. at 13:1-4.) *See also id.* at fig. 4.
[1c2] (2) moving the new data segment from the cache memory to a next available space within primary memory such that the new data segment is stored in primary memory in non-used memory space;	Katayama in view of Mills discloses moving the new data segment from the cache memory (*e.g*., write buffer) to a next available space within primary memory (*e.g*., moving data to vacant space in the flash memory) such that the new data segment is stored in primary memory in non-used memory space. "In effect, a portion of SRAM 240 can be used as a write cache for flash main memory 230." (Ex. 1006 at 11:39-40.) "At a write access, data from the system is written temporarily to the write buffer and thereafter it is written to the flash memory." (Ex. 1005 at 12:55-57.) "In response to a write access, the data distribution circuit 61 is connected to the write buffer 72 so that write data from the system is held in the write buffer 72, and thereafter it is written to the flash memory arrays 5 by the controller." (*Id*. at 13:1-4.) "FIG. 5A shows the memory contents after some files have

Claim 1	Katayama in view of Mills
	been stored, and FIG. 5B shows the memory contents, with some files being revised to have increased sizes." (*Id*. at 10:43-46.) "In this manner of storing file sectors closely and sequentially so that the memory groups have no vacant sector, as shown in FIG. 5A, if a file has an increased size due to revision, it could not be stored in physically continuous locations as shown in FIG. 5B. Even in such a case, file sectors are stored such that the assigned memory groups are continuous. For example, a file with file number 4 made up of five sectors 4-1 through 4-5 is initially stored as shown in FIG. 5A, and the file has additional two sectors 4-6 and 4-7 in FIG. 5B. Since the file starts with its top sector 4-1 at the memory group 3, the additional sectors 4-6 and 4-7 have their areas reserved in the memory groups 2 and 3. At the writing of the file having the increased sectors, the sectors 4-5 and 4-6 are treated as two continuous sectors and the sector 4-7 is treated as a single sector. Namely, additional sectors are treated to be continuous to the existing sectors." (*Id*. at 14:25-41.) *See also id.* at figs. 4, 5A, 5B.
[1c3] (3) identifying the previous logical data segment in primary memory;	Katayama in view of Mills discloses identifying the previous logical data segment (*e.g*., chaining the previous file sector to the new file sector) in primary memory. "FIG. 6 shows an example of the chaining information. Indicated by 85 is stored file data having file number 4 and sector number 5. 86 is file data of the next sector having file number 4 and sector number 6, and 87 is file data of the next sector having file number 4 and sector number 7." (*Id*. at 14:53-57.) "Indicated by 88 is the physical address of the file data 85, with its left-hand numeral '3' indicating the memory group number and its right-hand numeral '5' indicating the address within the memory group. Similarly, 89 and 90 are physical addresses of the stored file data 86 and 87. 91 is chaining information that points the physical address of the next sector

Claim 1	Katayama in view of Mills
	of the file data 85, i.e., the physical address of the file data 86 in this case, with its left-hand numeral indicating the memory group number and its right-hand numeral indicating the address within the memory group." (*Id*. at 14:63-15:5.) *See also id*. at fig. 6.
[1c4] (4) creating a logical link between the previous logical data segment and the new data segment such that the logical link provides a path for sequentially accessing the data segments within the primary memory;	Katayama in view of Mills discloses creating a logical link (*e.g*., by chaining the previous file sector to the new file sector) between the previous logical data segment and the new data segment such that the logical link provides a path for sequentially accessing the data segments within the primary memory. "FIG. 6 shows an example of the chaining information. Indicated by 85 is stored file data having file number 4 and sector number 5. 86 is file data of the next sector having file number 4 and sector number 6, and 87 is file data of the next sector having file number 4 and sector number 7." (*Id*. at 14:53-57.) "Indicated by 88 is the physical address of the file data 85, with its left-hand numeral '3' indicating the memory group number and its right-hand numeral '5' indicating the address within the memory group. Similarly, 89 and 90 are physical addresses of the stored file data 86 and 87. 91 is chaining information that points the physical address of the next sector of the file data 85, i.e., the physical address of the file data 86 in this case, with its left-hand numeral indicating the memory group number and its right-hand numeral indicating the address within the memory group. Similarly, chaining information 92 indicates the physical address of the file data 87. Chaining information 93 has its content indicating the absence of a successive sector, i.e., this is the last sector of the file of file number 4." (*Id*. at 14:63-15:9.) *See also id.* at fig. 6.
[1c5] (5) creating additional serial and logical links as subsequent new	Katayama in view of Mills discloses creating additional serial and logical links as subsequent new data segments are written to primary memory (*see* limitation [1c4]), said logical links providing the path for serially accessing the data segments

Claim 1	Katayama in view of Mills
data segments are written to primary memory, said logical links providing the path for serially accessing the data segments regardless of contiguity of the data segments relative to each other within the primary memory; and	regardless of contiguity of the data segments relative to each other (*e.g.*, allowing non-contiguous file sectors). "FIG. 6 shows an example of the chaining information. Indicated by 85 is stored file data having file number 4 and sector number 5. 86 is file data of the next sector having file number 4 and sector number 6, and 87 is file data of the next sector having file number 4 and sector number 7." (Ex. 1005 at 14:53-57.) "Indicated by 88 is the physical address of the file data 85, with its left-hand numeral '3' indicating the memory group number and its right-hand numeral '5' indicating the address within the memory group. Similarly, 89 and 90 are physical addresses of the stored file data 86 and 87. 91 is chaining information that points the physical address of the next sector of the file data 85, i.e., the physical address of the file data 86 in this case, with its left-hand numeral indicating the memory group number and its right-hand numeral indicating the address within the memory group. Similarly, chaining information 92 indicates the physical address of the file data 87. Chaining information 93 has its content indicating the absence of a successive sector, i.e., this is the last sector of the file of file number 4." (*Id*. at 14:63-15:9.) "Consequently, it becomes possible to make a continuous access to a file even if it is not stored in continuous locations." (*Id*. at 15:14-16.) *See also id*. at figs. 5A, 5B, 6.
[1c6] (6) storing the data segments to primary memory in a manner consistent with an industry standard data storage format while retaining	Katayama in view of Mills discloses storing data segments to primary memory in a manner consistent with an industry standard storage format (*e.g.*, by storing data in the flash memory, instead of in a magnetic-type disk memory of personal computers) while retaining linking between data segments created in previous steps. "In short, the major parts of the DOS, Windows and word processing programs will be resident in non-volatile flash

Claim 1	Katayama in view of Mills
linking between data segments created in previous steps.	memory 230 and can be executed in place directly from non-volatile main memory 230 as soon as the system powers up." (Ex. 1006 at 10:18-22.) "The ability to write to flash main memory 230 further means that the user can install a different program in place of (or in addition to) the word processing program currently stored in flash main memory 230. Therefore, the user can switch to a rival word processing program by installing the rival word processing program in flash main memory 230. Alternately, the user can, for example, install a spreadsheet or graphics design program in flash memory 230 in place of the word processing program that is currently stored there." (*Id.* at 11:7-15.) "Recently, notebook and palm-type personal computers have gained popularity, their appeal lying principally on their usefulness in terms of portability. Accordingly, file-memory-based semiconductor memory chips have been used in place of magnetic-type disk memory because the latter is not ideally suited to a notebook computer environment, i.e. They are not reliable against vibrations and consume too much power." (Ex. 1005 at 1:28-35.) "The information processing system of the present invention is equipped with a built-in semiconductor file memory device in the form of a storage medium, such as a flash memory, having a large unit erasure block size, which memory can therefore realize a level of fast file access performance that is superior compared to magnetic disk-type memories." (*Id*. at 3:9-15.)

ii. Claim 1 Is Obvious Over Krueger

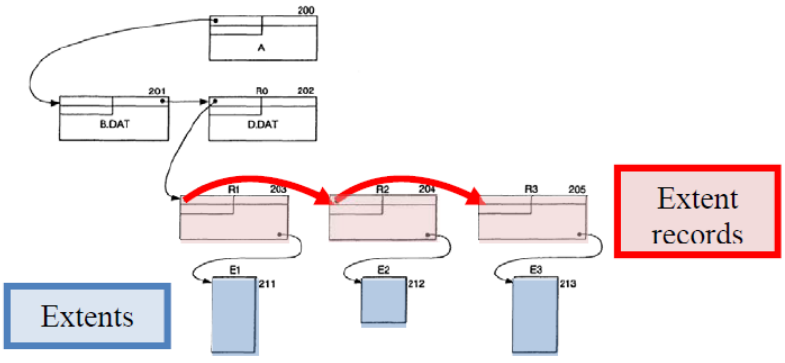
To the extent the Board concludes that "a logical link between the previous logical data segment and the new data segment" should be construed broader than "a pointer written to the previous logical data segment that points to the physical

location of the new data segment," such as with the district court's claim construction, Krueger renders Claim 1 obvious. For example, should the term be construed to include any linkage of data segments, Krueger discloses that term.

Krueger teaches a memory management system in a block-erasable flash-EProm (FEProm) device. (Ex. 1007 at 2:3-5.)[11] The FEProm's file system uses a linked-list structure for both the directory hierarchy and the internal file storage. (*Id*. at 13:46-48.) Each directory and file record can have pointers to the next lower level or to the next directory or file at the same level. (*Id*. at 13:56-14:10.)

Each file is made up of one or more extents, where an extent is a contiguous area of memory. (*Id*. at 14:22-23.) Each extent is associated with an extent record that contains a pointer to the extent data itself (highlighted in blue below) and a pointer to the next extent record (highlighted in red). (*Id*. at 14:23-27.) Figure 2A, below, shows the linked-list structure of the extents for file "D.DAT":

Figure 2A

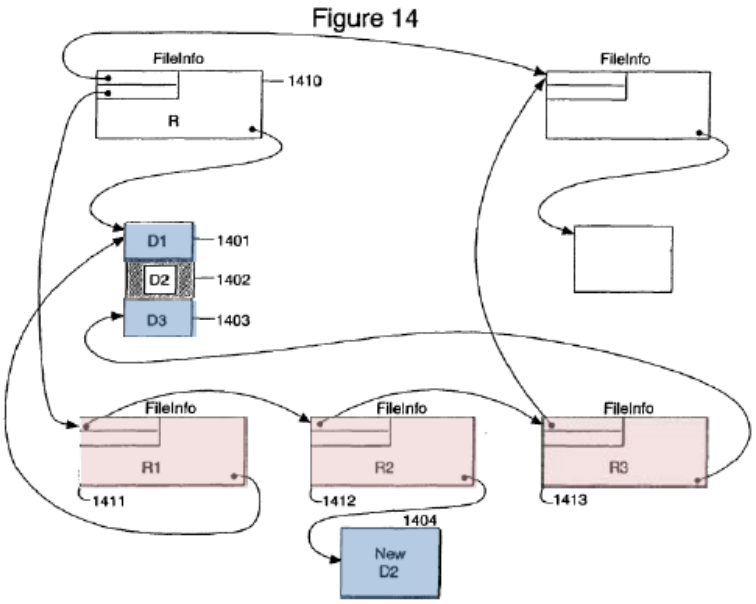


[11] Citations to Krueger are in the format "PAGE:LINE."

(*Id*. at fig. 2A (annotated).) The file record for "D.DAT" points to extent records R1, R2, and R3 (in red), which are linked together. Each extent record contains a pointer to the corresponding extents E1, E2, and E3 (in blue). (*Id*. at 14:25-28.)

Claim 1, Preamble: Krueger discloses a "method of memory management for a primary memory created from a non-volatile, long-term storage medium, said method enabling direct manipulation of contiguous and non-contiguous discrete data segments stored therein by a file system." (*See* Ex. 1001 ¶¶140-51, 153-59.) Krueger teaches a memory management system in a block-erasable flash-EProm (FEProm) device. (Ex. 1007 at 2:3-5.) The FEProm uses a linked-list structure for both the directory hierarchy and the internal file storage. (*Id*. at 13:46-48.) Each directory and file record has pointers to the next lower level or to the next directory or file at the same level. (*Id*. at fig. 1B, 13:56-14:10.) Each file is made up of one or more extents, where each extent is a contiguous area of memory. (*Id*. at 14:22-23.) The extent records for each extent are also chained together with a linked-list structure, as discussed above. (*Id*. at fig. 2A, 14:23-28.) The extents can be non-contiguous from each other. For example, figure 14 below discloses updating a portion of a file; the system divides the old data into three extents, D1, D2, and D3, and creates a new D2 extent that contains the updated portion of the data. (*Id*. at 22:43-55.) The extent record for D1 (1411) points to D1 (1401), which is unchanged; the extent record for D2 (1412) points to the new D2 extent (1404);

and the extent record for D3 (1413) points to the unchanged D3 extent (1403). (*Id*.) Thus, at least extents D1 and D2 are non-contiguous.



(*Id*. at fig. 14 (annotated).) By using pointers to link extent records, Krueger's FEProm allows a file that is made up of extents to be expanded or modified without modifying every extent of the file, thus enabling direct manipulation.

Further, while Krueger does not explicitly disclose using FEProm as the computer system's "main memory," this would have been obvious to one of ordinary skill in the art for several reasons. *First*, Krueger explains the disadvantage of volatile memory is that "when power is disconnected from a volatile storage device the information is lost." (*Id.* at 2:10-11.)

Second, Krueger explains that flash memory possesses the necessary characteristics of speed and byte-addressability that make it a possible substitute

for volatile main memory. Specifically, "[a] storage device known as Flash-EProm (FEProm) has the speed of internal computer memory combined with the nonvolatility of a computer disk." (*Id.* at 2:42-43.) Further, while "the internal memory [of a computer] is byte addressable," (*Id.* at 3:15), "[t]he FEProm . . . is byte addressable" as well. (*Id.* at 3:26.) One of ordinary skill in the art would understand that flash memory's byte-addressability and speed allow it to be used as the computer's main memory. (Ex. 1001 at ¶ 157.)

Third, Krueger is directed to using flash memory as the "computer memory." (*See* Ex. 1007 at 3:39-40 ("It is another object of the present invention to provide a computer memory manager . . . in a block-erasable FEProm."); 26:36-37 ("The method . . . wherein the computer memory device is a block-erasable, programmable, read-only memory.").) Krueger does not, however, relegate "computer memory" to secondary storage, but discloses that the computer memory at least may include main memory. (*See* Ex. 1007 at 3:14-15 ("This may involve the reading of the entire block from disk into the ***computer memory***, changing the one byte (the ***internal memory*** is byte addressable), and writing the entire block to the disk."); Ex. 1001 ¶ 159.)

In sum, Krueger teaches the benefits of flash memory, that flash memory has the necessary characteristics for main memory, and teaches flash memory as "computer memory," where "computer memory" can at least be main memory.

The disclosure of Krueger would therefore motivate one of ordinary skill in the art to use flash memory as primary memory. (*See* Ex. 1001 ¶¶ 156-60.)

Step (a): Krueger discloses "creating the primary memory from a non-volatile, long-term storage medium, wherein the primary memory comprises a plurality of blocks in which the data segments are to be stored." (*See* Ex. 1001 ¶ 161.) Krueger discloses a flash-EProm, (Ex. 1007 at Abstract), which contains "a number of blocks." (Ex. 1007 at 3:28-33.)

Step (b): Krueger discloses "coupling a cache memory to the primary memory, said cache memory providing temporary and volatile storage for at least one of the data segments." (*See* Ex. 1001 ¶¶ 162-64.) A non-FEProm memory is used to temporarily store data during the block reclamation process: "the FEProm manager could copy the allocated regions to non-FEProm memory, then erase the block, and copy the regions back to the block." (Ex. 1007 at 9:6-9.) Krueger explains that the "non-FEProm memory . . . has the potential to lose data should a power failure occur after erasure but before the block is rewritten." (*Id.* at 9:9-11; *see* Ex. 1001 at ¶¶ 163-64.) Thus, the cache memory is volatile.

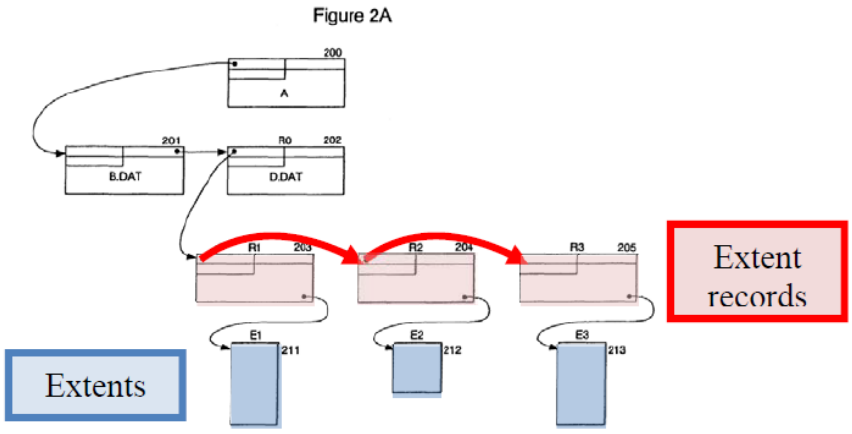
Step (c): Krueger discloses "writing a new data segment from the cache memory to the primary memory by linking said new data segment to a sequentially previous logical data segment." (*See* Ex. 1001 ¶¶ 141-51, 165.) As explained above in Step (b), Krueger discloses temporarily storing data in a non-FEProm

memory. Krueger also discloses linking the records of both new files and new extents to sequentially previous logical file and extent records, as explained further in Steps (c)(3) and (c)(4). (Ex. 1007 at figs. 1B, 2B, 13:56-14:10, 14:23-28.)

Step (c)(1): Krueger discloses "receiving the new data segment in the cache memory." (*See* Ex. 1001 ¶¶ 166-67.) Krueger discloses that the FEProm manager copies data to the non-FEProm memory. (Ex. 1007 at 9:6-9.)

Step (c)(2): Krueger discloses "moving the new data segment from the cache memory to a next available space within primary memory such that the new data segment is stored in primary memory in non-used memory space." (*See* Ex. 1001 ¶¶ 168-70.) The FEProm manager copies data from the non-FEProm memory to the flash FEProm. (Ex. 1007 at 9:6-9.) The system also "allocates a region for new FileInfo record in the FEProm." (Ex. 1007 at 22:10-13.)

Step (c)(3): Krueger discloses "identifying the previous logical data segment in primary memory." (*See* Ex. 1001 ¶¶ 171-72.) As explained above, each extent has an extent record (also called a FileInfo record). (Ex. 1007 at 14:23-28.) The extent record contains a pointer to the extent data itself and a pointer to the next extent record. (*Id*.) For example, figure 2A below shows the extents of file "D.DAT"; the file record for "D.DAT" points to extent records R1, R2, and R3 (in red), which are linked together. Each extent record contains a pointer to the corresponding extents E1, E2, and E3 (in blue).



(*Id*. at fig. 2A (annotated).) To extend a file, the system traverses the linked list of extent records and identifies the last extent record in the file to link it to the new extent record. (Ex. 1007 at 22:20-21 ("[T]he system locates the last FileInfo record (if one exists) for the file to be extended.").)

Krueger does not meet this limitation under Petitioners' proposed construction of this term as "a pointer written to the previous logical data segment that points to the physical location of the new data segment"; Petitioners' proposed construction would require a pointer written to the extent itself, rather than to the extent record. Krueger discloses this limitation if the term is construed to be broader (by not requiring that the pointer be written directly to the data segment).

Step (c)(4): Krueger discloses "creating a logical link between the previous logical data segment and the new data segment such that the logical link provides a path for sequentially accessing the data segments within the primary memory." (*See* Ex. 1001 ¶¶ 141-151, 173.) Krueger discloses creating a pointer between the

last extent record in the file and the new extent record. Each extent record contains a PrimaryPtr field, which points to the next extent record. (Ex. 1007 at 14:27-28, 19:26-20:36.) After the system identifies the last extent in a file to be extended, it sets the PrimaryPtr field of the corresponding extent record to point to the extent record of the new extent. (*Id*. at 22:33-35.) In other words, Krueger discloses creating a logical link between extent records, rather than the extents themselves. Krueger does not meet this limitation under Petitioners' proposed construction of "a logical link between the previous logical data segment and the new data segment," which requires "a pointer written to the previous logical data segment" itself. Krueger meets this limitation if the term is construed to be broader (by not requiring that the pointer be written directly to the previous logical data segment).

Step (c)(5): Krueger discloses "creating additional serial and logical links as subsequent new data segments are written to primary memory, said logical links providing the path for serially accessing the data segments regardless of contiguity of the data segments relative to each other within the primary memory." (*See* Ex. 1001 ¶¶ 141-51, 174-75.) As shown above in Step (c)(3), figure 2A discloses coupling additional extent records together, such as linking the extent records for extents E1, E2, and E3. (Ex. 1007 at fig. 2A, 14:22-27.) As discussed above for the preamble, the extents can also be non-contiguous.

Step (c)(6): Krueger discloses "storing the data segments to primary

memory in a manner consistent with an industry standard data storage format while retaining linking between data segments created in previous steps." (*See* Ex. 1001 ¶¶ 176-77.) The file system can store files that comport with an industry standard format, such as .DOC files. (Ex. 1007 at fig. 1A, 13:46-55.)

Further, to the extent the Board determines the proper construction of "data storage format" is "file storage for data" per the district court's construction, Krueger discloses this limitation, such as compatibility with MS-DOS. (Ex. 1007 at 13:49-14:10; 20:39-57; and 21:8-11.).

The claim chart below demonstrates how Krueger renders Claim 1 obvious.

Claim 1	Krueger
[1] A method of memory management for a primary memory created from a non-volatile, long-term storage medium, said method enabling direct manipulation of contiguous and non-contiguous discrete data segments stored therein by a file system, and comprising the steps of:	Krueger discloses a method of memory management for a primary memory created from a non-volatile, long-term storage medium (*e.g.*, FEProm), said method enabling direct manipulation of contiguous and non-contiguous discrete data segments (*e.g.*, by linking extents) stored therein by a file system. "This invention relates generally to a computer system for managing files and, more specifically, to a method and system for managing files stored on a flash-erasable, programmable, read-only memory (FEProm)." (Ex. 1007 at 2:3-5.) "A storage device known as a Flash-EProm (FEProm) has the speed of internal computer memory combined with the nonvolatility of a computer disk." (Ex. 1007 at 2:42-43.) "Each file has a file record associated with it that contains, among other data, the name of the file and that is linked into the directory hierarchy as described above. An extent is a contiguous area of memory that contains data for the file.

Claim 1	Krueger
	Each file comprises one or more extents, which contain the file data. Each extent has an extent record associated with it. The extent record contains, among other data, a pointer to the extent and the length of the extent. FIG. 2A shows the extents of the file '\A\D.DAT' 202. The extent records R1 203, R2 204, and R3 205 are linked and contain a pointer to the corresponding extents E1 211, E2 212, and E3 213. The file is the logical concatenation of extents E1 211, E2 212, and E3 213. In a preferred embodiment, the extent records are FileInfo structures as described below." (*Id*. at 14:21-28.) "FIG. 13 shows a typical portion of the linked list of the FileInfo records for a file. The Update_File routine will replace the data represented by the shaded area 1301. FIG. 14 shows the structure of the linked list after the modified data has been written to the FEProm. Three FileInfo records R1 1411, R2 1412, and R3 1413, have been inserted into the linked list. The entire extent is not rewritten, rather only the portion that actually changed is rewritten. The routine divides the extent into three sections, D1 1401, D2 1402, and D3 1403. Sections D1 1401 and D3 1403 contain data that is not changed by the update, and section D2 1402 contains the data that will change. Each section will have a corresponding FileInfo record. The FileInfo records R1 1411, R2 1412, and R3 1413 are linked through their PrimaryPtr fields. Also, the ExtentPtr field in R1 1411 and R3 1413, are set to point to their corresponding extent sections, and the ExtentLen fields are set. A new extent is allocated for the new data corresponding to the section new D2 1404, which is pointed to by record R2 1412. The SecondaryPtr of record R 1410 points to FileInfo R1 1411 to indicate that the PrimaryPtr of R 1410 is superseded. The PrimaryPtr of FileInfo record R3 1413 is set to the value contained in the PrimaryPtr of FileInfo record R 1410 to complete the link." (*Id*. at 22:43-55.) *See also id*. at figs. 1B, 2A, 13, 14.
[1a] (a) creating the primary	Krueger discloses creating the primary memory from a non-volatile, long-term storage medium (*e.g*., FEProm), wherein

Claim 1	Krueger
memory from a non-volatile, long-term storage medium, wherein the primary memory comprises a plurality of blocks in which the data segments are to be stored;	the primary memory comprises a plurality of blocks in which the data segments are to be stored (*e.g.*, blocks). "This invention relates generally to a computer system for managing files and, more specifically, to a method and system for managing files stored on a flash-erasable, programmable, read-only memory (FEProm)." (Ex. 1007 at 2:3-5.) "A storage device known as a Flash-EProm (FEProm) has the speed of internal computer memory combined with the nonvolatility of a computer disk." (Ex. 1007 at 2:42-43.) "An FEProm can also be organized in a block-erasable format. A block-erasable FEProm contains a number of blocks, typically 16, that can be independently erased." (*Id*. at 3:28-29.)
[1b] (b) coupling a cache memory to the primary memory, said cache memory providing temporary and volatile storage for at least one of the data segments;	Krueger discloses coupling a cache memory (*e.g.*, non-FEProm memory) to the primary memory (*e.g.*, FEProm), where the cache memory provides temporary and volatile storage for at least one of the data segments. "The FEProm manager reclaims a block by copying the allocated regions to a spare block, a block that has been erased. By copying only the allocated regions, the deallocated regions are reclaimed. Alternatively, the FEProm manager could copy the allocated regions to non-FEProm memory, then erase the block, and copy the regions back to the block. However, this method requires enough non-FEProm memory to store the allocated regions and has the potential to lose data should a power failure occur after erasure but before the block is rewritten. In the preferred method, the FEProm manager copies the allocated regions in the block to be reclaimed to the spare block and copies the Block Allocation Structure adjusting the variable Offset to reflect the new region locations in the spare block." (Ex. 1007 at 9:6-13.)
[1c] (c) writing a new data segment from the cache	Krueger discloses writing a new data segment from the cache memory to the primary memory by linking the new data segment to a sequentially previous logical data segment.

Claim 1	Krueger
memory to the primary memory by linking said new data segment to a sequentially previous logical data segment by the following steps:	"Each file has a file record associated with it that contains, among other data, the name of the file and that is linked into the directory hierarchy as described above. An extent is a contiguous area of memory that contains data for the file. Each file comprises one or more extents, which contain the file data. Each extent has an extent record associated with it. The extent record contains, among other data, a pointer to the extent and the length of the extent. FIG. 2A shows the extents of the file '\A\D.DAT' 202. The extent records R1 203, R2 204, and R3 205 are linked and contain a pointer to the corresponding extents E1 211, E2 212, and E3 213. The file is the logical concatenation of extents E1 211, E2 212, and E3 213. In a preferred embodiment, the extent records are FileInfo structures as described below." (Ex. 1007 at 14:21-28.) "Referring to FIG. 10 in block 1001, the system allocates a region for new FileInfo record in the FEProm and sets the variable FI to point to that record. In block 1002, the system allocates a region for the data extent and sets the variable D to point to the extent. In block 1003, the system writes the data to the allocated block." (Id. at 22:10-13.) "In blocks 1007 through 1012, the system locates the last FileInfo record (if one exists) for the file to be extended. . . . In block 1013, the system sets PrimaryPtr of the record pointed to by prev_ptr equal to the pointer to FI to effect the extending of the file. In block 1014, the system sets Status of the record pointed to by prev_ptr equal to PrimaryPtrValid and the routine is done." (*Id*. at 22:20-36.) *See also* figs. 1B, 2A.
[1c1] (1) receiving the new data segment in the cache memory;	Krueger discloses receiving the new data segment in cache memory (e.g., non-FEProm memory). "The FEProm manager reclaims a block by copying the allocated regions to a spare block, a block that has been erased. By copying only the allocated regions, the deallocated

Claim 1	Krueger
	regions are reclaimed. Alternatively, the FEProm manager could copy the allocated regions to non-FEProm memory, then erase the block, and copy the regions back to the block. However, this method requires enough non-FEProm memory to store the allocated regions and has the potential to lose data should a power failure occur after erasure but before the block is rewritten. In the preferred method, the FEProm manager copies the allocated regions in the block to be reclaimed to the spare block and copies the Block Allocation Structure adjusting the variable Offset to reflect the new region locations in the spare block." (Ex. 1007 at 9:6-13.)
[1c2] (2) moving the new data segment from the cache memory to a next available space within primary memory such that the new data segment is stored in primary memory in non-used memory space;	Krueger discloses moving the new data segment from the cache memory (*e.g.*, non-FEProm memory) to a next available space within primary memory (*e.g.*, through data allocation) such that the new data segment is stored in primary memory in non-used memory space. "The FEProm manager reclaims a block by copying the allocated regions to a spare block, a block that has been erased. By copying only the allocated regions, the deallocated regions are reclaimed. Alternatively, the FEProm manager could copy the allocated regions to non-FEProm memory, then erase the block, and copy the regions back to the block. However, this method requires enough non-FEProm memory to store the allocated regions and has the potential to lose data should a power failure occur after erasure but before the block is rewritten. In the preferred method, the FEProm manager copies the allocated regions in the block to be reclaimed to the spare block and copies the Block Allocation Structure adjusting the variable Offset to reflect the new region locations in the spare block." (Ex. 1007 at 9:6-13.) "Referring to FIG. 10 in block 1001, the system allocates a region for new FileInfo record in the FEProm and sets the variable FI to point to that record. In block 1002, the system allocates a region for the data extent and sets the variable D to point to the extent. In block 1003, the system writes the data to the allocated block." (Ex. 1007 at 22:10-13.)

Claim 1	Krueger
[1c3] (3) identifying the previous logical data segment in primary memory;	Krueger discloses identifying the previous logical data segment in primary memory *(e.g.*, the previous extent record). "In blocks 1007 through 1012, the system locates the last FileInfo record (if one exists) for the file to be extended. The system follows the PrimaryPtr or the SecondaryPtr of the FileEntry record and the FileInfo records. . . . When the last FileInfo record in the file is located, the pointer prev_ptr will contain the pointer to that record. . . . In block 1013, the system sets PrimaryPtr of the record pointed to by prev_ptr equal to the pointer to FI to effect the extending of the file. In block 1014, the system sets Status of the record pointed to by prev_ptr equal to PrimaryPtrValid and the routine is done." (Ex. 1007 at 22:20-36.) *See also id*. at fig. 2A.
[1c4] (4) creating a logical link between the previous logical data segment and the new data segment such that the logical link provides a path for sequentially accessing the data segments within the primary memory;	Krueger discloses creating a logical link between the previous logical data segment and the new data segment (*e.g.*, by setting the PrimaryPtr in the extent record for the previous extent to the extent record for the new extent) such that the logical link provides a path for sequentially accessing the data segments within the primary memory. "Each file has a file record associated with it that contains, among other data, the name of the file and that is linked into the directory hierarchy as described above. An extent is a contiguous area of memory that contains data for the file. Each file comprises one or more extents, which contain the file data. Each extent has an extent record associated with it. The extent record contains, among other data, a pointer to the extent and the length of the extent. FIG. 2A shows the extents of the file '\A\D.DAT' 202. The extent records R1 203, R2 204, and R3 205 are linked and contain a pointer to the corresponding extents E1 211, E2 212, and E3 213. The file is the logical concatenation of extents E1 211, E2 212, and E3 213. In a preferred embodiment, the extent records are FileInfo structures as described below." (*Id*. at 14:21-28.) "PrimaryPtr [in] FileInfo: points to the next FileInfo entry for

Claim 1	Krueger
	the file" (*Id*. at 19:26-45.) "In blocks 1007 through 1012, the system locates the last FileInfo record (if one exists) for the file to be extended. The system follows the PrimaryPtr or the SecondaryPtr of the FileEntry record and the FileInfo records. . . . When the last FileInfo record in the file is located, the pointer prev_ptr will contain the pointer to that record. . . . In block 1013, the system sets PrimaryPtr of the record pointed to by prev_ptr equal to the pointer to FI to effect the extending of the file. In block 1014, the system sets Status of the record pointed to by prev_ptr equal to PrimaryPtrValid and the routine is done." (*Id*. at 22:20-36.) *See also id*. at fig. 2A.
[1c5] (5) creating additional serial and logical links as subsequent new data segments are written to primary memory, said logical links providing the path for serially accessing the data segments regardless of contiguity of the data segments relative to each other within the primary memory; and	Krueger discloses creating additional serial and logical links as subsequent new data segments are written to primary memory (see limitation [1c4]), said logical links providing the path for serially accessing the data segments (*e.g*., extents) regardless of contiguity of the data segments relative to each other. "An extent is a contiguous area of memory that contains data for the file. Each file comprises one or more extents, which contain the file data. Each extent has an extent record associated with it. The extent record contains, among other data, a pointer to the extent and the length of the extent. FIG. 2A shows the extents of the file '\A\D.DAT' 202. The extent records R1 203, R2 204, and R3 205 are linked and contain a pointer to the corresponding extents E1 211, E2 212, and E3 213. The file is the logical concatenation of extents E1 211, E2 212, and E3 213." (*Id*. at 14:22-27.) "FIG. 13 shows a typical portion of the linked list of the FileInfo records for a file. The Update_File routine will replace the data represented by the shaded area 1301. FIG. 14 shows the structure of the linked list after the modified data has been written to the FEProm. Three FileInfo records R1 1411, R2 1412, and R3 1413, have been inserted into the

Claim 1	Krueger
	linked list. The entire extent is not rewritten, rather only the portion that actually changed is rewritten. The routine divides the extent into three sections, D1 1401, D2 1402, and D3 1403. Sections D1 1401 and D3 1403 contain data that is not changed by the update, and section D2 1402 contains the data that will change. Each section will have a corresponding FileInfo record. The FileInfo records R1 1411, R2 1412, and R3 1413 are linked through their PrimaryPtr fields. Also, the ExtentPtr field in R1 1411 and R3 1413, are set to point to their corresponding extent sections, and the ExtentLen fields are set. A new extent is allocated for the new data corresponding to the section new D2 1404, which is pointed to by record R2 1412. The SecondaryPtr of record R 1410 points to FileInfo R1 1411 to indicate that the PrimaryPtr of R 1410 is superseded. The PrimaryPtr of FileInfo record R3 1413 is set to the value contained in the PrimaryPtr of FileInfo record R 1410 to complete the link." (*Id*. at 22:43-55.) *See also id*. at figs. 2A, 13, 14.
[1c6] (6) storing the data segments to primary memory in a manner consistent with an industry standard data storage format while retaining linking between data segments created in previous steps.	Krueger discloses storing data segments to primary memory in a manner consistent with an industry standard storage format (*e.g*., storing files in the .DOC Microsoft Word format) while retaining linking between data segments created in previous steps. "The present invention provides a directory-based hierarchical file system for an FEProm device. . . . A preferred embodiment uses a linked-list data structure to implement both the directory hierarchy and the internal file storage. FIG. 1A shows a typical hierarchical directory structure. The MS-DOS operating system, which is available from Microsoft Corporation of Redmond, Washington, implements a file system with a hierarchical directory structure. As shown in FIG. 1A . . . . [t]he directory DAVID 107 contains one file LETTERI.DOC 109. The directory MARY 108 contains three files LETTER1.DOC 110, LETTER2.DOC 111, and LETTER3.DOC 112." (Ex. 1007 at 13:46-55.)

Claim 1	Krueger
	"The portion shown in Figure 26 comprises the DirEntry and FileEntry records for the directory ROOT, directory DOS, directory WORD, file AUTOEXEC.BAT, and file COMMAND.COM. Block 0 contains directory DOS, and directory WORD; block 12 contains directory ROOT and file COMMAND.COM; and block 14 contains file AUTOEXEC.BAT." (*Id.* at 20:43-46.) "The PrimaryPtr 2413 of directory ROOT points to Alloc[O] entry 2421 corresponding to directory DOS. Alloc[O] entry 2421 contains the 10 variable Offset 2422, which contains the offset of region 2420. Region 2420 contains the DirEntry for directory DOS." (*Id.* at 21:8-11.)

V. MEANINGFUL BENEFIT TO INSTITUTING ON BOTH GROUNDS

Petitioners respectfully submit that the Board should institute IPR on both Grounds 1 and 2 because of the meaningful benefit from doing so. For example, Katayama in view of Mills requires combining two references, whereas Krueger is only one reference. As compared with Krueger, Katayama in view of Mills more robustly meets the BRI of the term "a logical link between the previous logical data segment and the new data segment."

Further, Krueger, unlike Katayama or Mills, was published more than one year before the earliest priority date of the '108 patent and qualifies as prior art under § 102(b). The patent owner, therefore, cannot swear behind Krueger. Katayama and Mills, however, both qualify as prior art only under § 102(e) and could be sworn behind.

In sum, the two grounds presented in this petition do not impede "the just, speedy and inexpensive resolution of [this] proceeding" as required by 37 C.F.R. § 42.1(b). Petitioners respectfully request that the Board institute IPR on both grounds presented, as each of Grounds 1 and 2 have meaningful benefits (and differences) relative to each other for purposes of challenging Claim 1.

VI. CONCLUSION

The cited prior art references provide non-cumulative technological teachings which indicate a reasonable likelihood of success as to Petitioners' assertion that claim 1 of the '108 patent is not patentable pursuant to the grounds presented in this Petition. Accordingly, Petitioners request IPR of Claim 1 of U.S. Patent No. 5,839,108.

Respectfully submitted,

Date: December 31, 2014

/David M. Maiorana/
David M. Maiorana, Reg. No. 41,449
JONES DAY
North Point, 901 Lakeside Avenue
Cleveland, Ohio 44114
(216) 586-3939

/Matthew A. Ferry/
Matthew A. Ferry, Reg. No. 63,142
JONES DAY
12265 El Camino Real, Suite 200
San Diego, CA 92130
(858) 314-1200

Attorneys for Petitioners

Appendix of Exhibits

Exhibit No.	Description
1001	Declaration of R. Jacob Baker, Ph.D. under 37 C.F.R. § 1.68 in Support of Petition for *Inter Partes* Review of U.S. Patent No. 5,839,108
1002	Appendix A to the Declaration of R. Jacob Baker, Ph.D.: R. Jacob Baker's current *curriculum vitae*
1003	Declaration of Matthew A. Ferry
1004	U.S. Patent No. 5,839,108 to Daberko, *et al*.
1005	U.S. Patent No. 6,272,610 to Katayama, *et al.*
1006	U.S. Patent No. 5,696,917 to Mills, *et al.*
1007	European Patent Application Publication No. 0 557 736 A2 to Krueger, *et al.*
1008	U.S. Patent No. 5,787,445 to Daberko
1009	Excerpt from File History of U.S. Patent No. 5,787,445: November 10, 1997 Amendment
1010	Excerpt from Microsoft Press Computer Dictionary (2nd ed.): EEPROM, flash memory, linked list, main memory, microprocessor, primary storage, pointer, RAM
1011	Excerpt from Dictionary of Computing (4th ed.): link, linked list, main memory, primary memory
1012	Excerpt from Random House Personal Computer Dictionary (2nd ed.): file system
1013	Excerpt from File History of U.S. Patent No. 5,787,445: July 1, 1997 Office Action

Exhibit No.	Description
1014	Excerpt from File History of U.S. Patent No. 5,787,445: January 5, 1998 Notice of Allowability
1015	USPTO Assignment Information for U.S. Patent No. 5,839,108
1016	Claim Construction Order in Cases Nos. 13-cv-2897-H-BGS, 13-cv-2899-H-BGS, 13-cv-2914-H-BGS, 13-cv-2915-H-BGS, and 13-cv-2946-H-BGS

Certificate of Service

The undersigned hereby certifies that a copy of the foregoing Petition for *Inter Partes* Review Under 37 C.F.R. § 42.100 [U.S. Patent No. 5,839,108], along with all exhibits and other supporting documents, was served on December 31, 2014 by UPS overnight delivery directed to the attorney of record for the patent at the following address:

Attorneys Associated with Attorney Docket No. T2614CIP
THORPE NORTH & WESTERN
8180 South 700 East, Suite 350
Sandy, Utah 84070

A courtesy copy was also served by email on the attorney of record for the plaintiff in related matters e.*Digital Corp. v. Micron Consumer Products Group, Inc.*, No. 3:13-cv-02907 (S.D. Cal.); *e.Digital Corp. v. Micron Technology, Inc.*, No. 3:13-cv-02944 (S.D. Cal.); *e.Digital Corp. v. Other World Computing, Inc.*, No. 3:13-cv-02915 (S.D. Cal.); and *e.Digital Corp. v. Mushkin, Inc.*, No. 3:13-cv-02914 (S.D. Cal.):

Anthony Handal
Handal & Associates
750 B Street
Suite 2510
San Diego, CA 92101
anh@handal-law.com

/s/ Matthew A. Ferry
Matthew A. Ferry