(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号

特許第5266250号
(P5266250)

(45)発行日　平成25年8月21日(2013.8.21)　(24)登録日　平成25年5月10日(2013.5.10)

(51)Int.Cl.		F I		
G06F 12/00	(2006.01)	G06F	12/00	542J
G06F 12/02	(2006.01)	G06F	12/02	510A
G06F 3/06	(2006.01)	G06F	12/00	597U
		G06F	3/06	301T

請求項の数 16　(全 48 頁)

(21)出願番号	特願2009-544199 (P2009-544199)
(86)(22)出願日	平成19年12月19日 (2007.12.19)
(65)公表番号	特表2010-515162 (P2010-515162A)
(43)公表日	平成22年5月6日 (2010.5.6)
(86)国際出願番号	PCT/US2007/088165
(87)国際公開番号	WO2008/082996
(87)国際公開日	平成20年7月10日 (2008.7.10)
審査請求日	平成22年12月17日 (2010.12.17)
(31)優先権主張番号	11/616,242
(32)優先日	平成18年12月26日 (2006.12.26)
(33)優先権主張国	米国(US)
(31)優先権主張番号	11/616,236
(32)優先日	平成18年12月26日 (2006.12.26)
(33)優先権主張国	米国(US)

(73)特許権者　511226960
サンディスク　テクノロジィース　インコーポレイテッド
アメリカ合衆国、７５０２４、テキサス州、プレーノー、ノース・ダラス・パークウェイ　６９００、ツゥー・レガシー・タウン・センター
(74)代理人　100075144
弁理士　井ノ口　壽
(72)発明者　シンクレア，アラン　ウェルシュ
イギリス連邦共和国、ＦＫ２　ＯＢＵ、フォールカーク、マディストン、キャンディ、ブロードヘッド、ザ　コテージース

最終頁に続く

(54)【発明の名称】連続論理アドレス空間インターフェイスを備えるダイレクトデータファイルシステムの使用

(57)【特許請求の範囲】

【請求項１】

システム内でファイルオブジェクトのデータを識別する方法であって、

論理ブロックに分割されたシステム論理アドレス空間を管理するステップと、

１つ以上の前記論理ブロックの中で前記ファイルオブジェクトの各々に一意なアドレスを割り当てるステップであって、前記論理ブロックは２つ以上のファイルオブジェクトのアドレスを各々収容できる、一意なアドレスを割り当てるステップと、

単一のファイルオブジェクトが割り当てられていて第２のファイルオブジェクトのデータをも収容する論理ブロックの数を制限するステップと、

を含む方法。

【請求項２】

請求項１記載の方法において、

前記ファイルオブジェクトのデータを追加的に提供し、かつメモリシステムとの接続に適した外部インターフェイスで前記システム論理アドレス空間を利用するホストシステムにおいて、前記方法を遂行するステップをさらに含む方法。

【請求項３】

請求項１記載の方法において、

外部供給元から前記ファイルオブジェクトのデータを追加的に受信し、かつ不揮発性データ蓄積媒体とのインターフェイスとしてメモリシステムの中で前記システム論理アドレス空間を利用する前記メモリシステムにおいて、前記方法を遂行するステップをさらに含

む方法。
【請求項４】
請求項１記載の方法において、
第１のインターフェイスを通じてホストに取り外し可能な状態で接続されることに適し、かつメモリシステムに取り外し可能な状態で接続されることに適した処理装置にて、前記方法を遂行するステップをさらに含み、前記メモリシステムは前記処理装置との第２のインターフェイスで前記システム論理アドレス空間を利用する方法。
【請求項５】
請求項１～４のいずれか記載の方法において、
論理ブロックの数を制限するステップは、単一のデータファイルオブジェクトのアドレスが書き込まれていて第２のファイルオブジェクトのアドレスをも収容する論理ブロックの数を所定の最大数に制限するステップを含む方法。
【請求項６】
請求項５記載の方法において、
前記所定の最大数は、２である方法。
【請求項７】
再プログラミングに先立ちまとめて消去できるメモリセルブロックを有するタイプの不揮発性メモリシステムとの接続に適したインターフェイスを通じて、ホストシステムがファイルオブジェクトのデータを転送する方法であって、
論理ブロックに分割された論理アドレス空間をインターフェイスで管理するステップと、
１つ以上の前記論理ブロックの中で前記ファイルオブジェクトの各々に含まれるデータのアドレスを指定するステップであって、前記論理ブロックは２つ以上のファイルオブジェクトを各々収容できるが、単一ファイルオブジェクトのアドレスで部分的にしか満たされていない論理ブロックの数を少なくとも１つの事前設定制限未満に制限し、前記部分的に満たされたブロックに１つ以上の別のファイルオブジェクトのアドレスが書き込まれるようにする、データのアドレスを指定するステップと、
を含む方法。
【請求項８】
請求項７記載の方法において、
前記論理アドレス空間の前記論理ブロックのサイズを、前記ホストシステムとの接続に適したメモリシステムの各メモリセルブロックと同じデータ蓄積容量にそれぞれがなるように構成するステップをさらに含む方法。
【請求項９】
請求項７記載の方法において、
１つ以上の前記論理ブロックにおける各ファイルオブジェクトのデータのアドレスは、前記ホストが前記インターフェイスを通じて前記ファイルオブジェクトのデータを送信するときに指定される方法。
【請求項１０】
データの再プログラミングに先立ちまとめて消去できるメモリセルブロックを有する不揮発性メモリシステムで、ファイルオブジェクトのデータを蓄積する方法であって、
論理アドレス空間は、各メモリセルブロックの特性に一致する少なくとも１つの特性を各々が有する論理ブロックに分割され、
各ファイルオブジェクトのデータのアドレスは、１つ以上の前記論理ブロックの中で割り当てられ、前記論理ブロックは、２つ以上のファイルオブジェクトのデータのアドレスを各々収容できるが、ある特定の単一ファイルオブジェクトのアドレスに加え別のファイルオブジェクトのアドレスをも収容する論理ブロックの数を制限し、かつ
前記論理ブロックのアドレスは、前記メモリシステムの中でメモリセルブロックのアドレスにマップされる方法。
【請求項１１】

請求項１０記載の方法において、
特定のファイルオブジェクトのデータにアドレスを割り当てることは、特定のファイルオブジェクトのアドレスで部分的にしか満たされていない論理ブロックの数を所定の数に制限することを含む方法。
【請求項１２】
請求項１１記載の方法において、
前記所定の数は、２である方法。
【請求項１３】
請求項１０記載の方法において、
少なくとも１つの一致する特性は、各論理ブロックのデータ蓄積容量が、各メモリセルブロックのデータ蓄積容量と同じであることを含む方法。
【請求項１４】
請求項１０記載の方法において、
少なくとも１つの一致する特性は、
各論理ブロックのデータ蓄積容量が、各メモリセルブロックのデータ蓄積容量と同じであることと、
各論理ブロックが、データの書き込みのため、前記メモリセルブロックの複数のページと同じデータ蓄積容量を有する複数のページに分割されることと、
各論理ブロックの最下位ページアドレスが、各メモリセルブロックの第１のページにマップされることと、
を含む方法。
【請求項１５】
再プログラミングに先立ちまとめて消去できるメモリセル単位を有するタイプの不揮発性メモリシステムとの接続に適したインターフェイスを通じて、ホストシステムがファイルオブジェクトのデータを転送する方法であって、
論理ブロックに分割された論理アドレス空間をインターフェイスで管理するステップと、
各ブロック内でアドレスが指定されたファイルデータの構造に基づき、１セットの複数の論理ブロックタイプを指定するステップと、
各ファイルのアドレスを蓄積する１つ以上の論理ブロックのタイプの組み合わせに基づき、１セットの複数の許容ファイル状態を指定するステップと、
前記論理ブロック内でアドレスが指定された各データファイルのファイル状態の記録を管理するステップと、
単一ファイルのデータのアドレスを、前記記録における前記単一ファイルの現在状態に従い選択される１タイプの論理ブロックに指定するステップと、
を含む方法。
【請求項１６】
請求項１５記載の方法において、
複数の指定データブロックタイプは、ただ１つのファイルのアドレスを単独の論理ブロックに蓄積する第１の複数のタイプと、２つ以上のファイルのアドレスを単独の論理ブロックに蓄積する第２の複数のタイプとを含み、前記許容ファイル状態により、単一のファイルのアドレスを指定できる前記第２の複数のタイプの最大ブロック数を制限する方法。
【発明の詳細な説明】
【技術分野】
【０００１】
本願は、一般的にはデータを蓄積し、接続されたホスト装置とデータをやり取りする、再プログラム可能な半導体フラッシュメモリ等の不揮発性メモリシステムの操作に関し、より具体的にはそこでのデータファイルオブジェクトの管理に関する。
【背景技術】
【０００２】

初期世代の商用フラッシュメモリシステムでは、矩形のメモリセルアレイが、標準ディスクドライブセクタのデータ量、すなわち５１２バイトを各々が蓄積する多数のセルグループに分割されていた。さらに通常ならば、誤り訂正符号（ＥＣＣ）を蓄積するため、そしてことによるとユーザデータ、および／またはこれを蓄積するメモリセルグループに、関係する他のオーバーヘッドデータを蓄積するため、一定量のデータ、例えば１６バイトが、各グループに加わる。それぞれのグループには、まとめて消去できる最低数のメモリセルがある。つまり、１データセクタ、さらにオーバーヘッドデータが含まれる場合はオーバーヘッドデータを蓄積するメモリセル数が事実上の消去単位となる。米国特許第５，６０２，９８７号（特許文献１）および第６，４２６，８９３号（特許文献２）には、この種のメモリシステムの例が記載されている。フラッシュメモリの特徴として、メモリセルにデータを再度プログラムするには事前にそのメモリセルを消去する必要がある。

【０００３】

フラッシュメモリシステムは多くの場合、パーソナルコンピュータやカメラ等、様々なホストと取り外し可能な状態で接続するメモリカードやフラッシュドライブの形で提供されるが、ホストシステムの中に埋め込まれることもある。ホストは通例、メモリへデータを書き込む場合に、メモリシステムの連続する仮想アドレス空間の中でセクタ、クラスタ、あるいはその他のデータ単位に一意な論理アドレスを割り当てる。ホストは、ディスクオペレーティングシステム（ＤＯＳ）のように、メモリシステムの論理アドレス空間の中のアドレスでデータを読み書きする。メモリシステム内のコントローラは、ホストから受け取った論理アドレスを、メモリアレイの中でデータを実際に蓄積する物理アドレスに翻訳し、これらのアドレス翻訳の経緯を把握する。メモリシステムのデータ蓄積容量は少なくとも、メモリシステム向けに設定される論理アドレス空間全体にわたってアドレスが指定されるデータの量に相当する。

【０００４】

後続世代のフラッシュメモリシステムでは、消去単位のサイズが複数セクタのデータを十分に蓄積するメモリセルブロックまで拡大した。メモリシステムが接続されたホストシステムでセクタ等の小さな最小単位でデータのプログラミングや読み出しが行われるとしても、フラッシュメモリの１消去単位には多数のセクタが蓄積される。ホストが論理セクタのデータで更新や差し替えを行うときに、ブロックの中で何セクタかのデータが用済みになることは一般的なことである。ブロックに蓄積されたデータに上書きを行うには事前にブロック全体を消去しなければならないため、新規データや更新データは通常、それを受け入れる容量が残っている別の消去済みブロックに蓄積される。この過程で元のブロックには用済みデータが残り、メモリ内の貴重なスペースを取ることになる。しかし、このブロックの中に有効データが残っていると、このブロックを消去するわけにはいかない。

【０００５】

そこでメモリ蓄積容量の有効利用を図るため、部分的に満たされたブロックの有効データを消去済みブロックにコピーすることによってこれを整理統合または回収するのが一般的であり、こうすることでデータのコピー元にあたるブロックは消去でき、その全蓄積容量を再利用できる。データをコピーしてブロック内のデータセクタを論理アドレス順に整理するのも望ましく、こうすることでデータの読み出し速度と読み出しデータをホストへ転送する速度が上がる。そのようなデータコピーがあまりにも頻繁に行われると、メモリシステムの動作性能が低下するおそれがある。これは特に、メモリの蓄積容量が、システムの論理アドレス空間を通じてホストによってアドレスが割り当てられるデータの量と大差ない場合、つまりよくある場合で、メモリシステムの動作に影響する。この場合は、ホストプログラミングコマンドの実行に先立ちデータの整理統合または回収が必要になる。その結果、プログラミングの時間が長引く。

【０００６】

一定の半導体領域に蓄積できるデータのビット数を増やすため、ブロックのサイズはメモリシステムの世代交代を通じて拡大している。２５６以上のデータセクタを蓄積するブロックが一般的になりつつある。加えてデータのプログラミングと読み出しにあたって並

列度を高めるため、異なるアレイまたはサブアレイからなる２つ、４つ、またはそれ以上のブロックがしばしばメタブロックとして論理的にリンクされる。そのような大容量操作単位には、メモリシステムを効率的に操作するという課題がともなう。

【先行技術文献】

【特許文献】

【０００７】

【特許文献１】米国特許第５，６０２，９８７号
【特許文献２】米国特許第６，４２６，８９３号
【特許文献３】米国公開特許出願第２００６／００３１５９３号
【特許文献４】米国特許第５，５７０，３１５号
【特許文献５】米国特許第５，７７４，３９７号
【特許文献６】米国特許第６，０４６，９３５号
【特許文献７】米国特許第６，３７３，７４６号
【特許文献８】米国特許第６，４５６，５２８号
【特許文献９】米国特許第６，５２２，５８０号
【特許文献１０】米国特許第６，７７１，５３６号
【特許文献１１】米国特許第６，７８１，８７７号
【特許文献１２】米国公開特許出願第２００３／０１４７２７８号
【特許文献１３】米国特許第６，９２５，００７号
【特許文献１４】米国特許第６，７６３，４２４号
【特許文献１５】米国公開特許出願第２００５／０１４４３５８号
【特許文献１６】米国特許第７，１３９，８６４号
【特許文献１７】米国公開特許出願第２００５／０１４１３１３号
【特許文献１８】米国公開特許出願第２００５／０１４１３１２号
【特許文献１９】米国公開特許出願第２００５／０１６６０８７号
【特許文献２０】米国公開特許出願第２００５／０１４４３６５号
【特許文献２１】米国公開特許出願第２００６／０１６１７２２号
【特許文献２２】米国公開特許出願第２００６／０１５５９２１号
【特許文献２３】米国公開特許出願第２００６／０１５５９２２号
【特許文献２４】米国公開特許出願第２００６／０１５５９２０号
【特許文献２５】米国公開特許出願第２００５／０１４４３５７号
【特許文献２６】米国公開特許出願第２００５／０１４４３６３号
【特許文献２７】米国公開特許出願第２００５／０１４４３６７号
【特許文献２８】米国特許出願第１０／８９７，０４９号
【特許文献２９】米国特許出願第１１／０２２，３６９号
【特許文献３０】米国特許出願第１１／２５９，４２３号
【特許文献３１】米国特許出願第１１／３１２，９８５号

【発明の概要】

【０００８】

前に相互参照した特許出願では、ホストから提供されるデータファイルオブジェクトをフラッシュメモリに直接蓄積するメモリシステムが説明されている。これは、「背景技術」の欄で前述したホストとメモリシステムとのインターフェイスに連続論理アドレス空間が存在する現在の大部分の商用システムと異なる。「ＬＢＡインターフェイス」の場合は通常、個々のデータファイルオブジェクトのデータが多数のメモリセルブロックに存在する。メモリシステムは、通常ならば多数のデータセクタからなるクラスタでホストから提供されるファイルオブジェクトのデータを、個々のデータファイルオブジェクトに対応付けることをしない。ホストは、ＬＢＡインターフェイスの中で有効データに現在割り当てられていない未使用論理アドレスを、蓄積のためにメモリシステムへ提供されるデータに割り当てるだけである。メモリシステムは、自身が効率よく作動するようにメモリセルブロックを割り当て受信データを蓄積するが、クラスタがどのデータファイルオブジェクト

のものなのかは認識しない。その結果、通常は個々のファイルオブジェクトのデータが断片化され、多数の異なるメモリセルブロックに蓄積される。

【０００９】

他方、前に相互参照した特許出願の多くでは、メモリシステムがＬＢＡインターフェイスを通さずホストから直接データファイルオブジェクトを受け取るため、メモリシステムは自身の性能を上げるように個々のファイルデータをメモリセルブロックに割り振ることができる。例えばデータがどのファイルのものなのかが分かるなら、メモリシステムはいずれか１つのデータファイルの蓄積に使うメモリセルブロックの数を制限できる。具体的に、メモリシステムはある１つのファイルオブジェクトのデータのほかに別のファイルオブジェクトのデータをも収容するメモリセルブロックの数を制限できる。その結果、ファイルデータの断片化は制御できる。このため、共通ブロックに蓄積された別のファイルのデータが削除されたり修正されたりするときに生じる用済みデータ領域の再生にあたって、共通ブロックから別の場所に移すことになる有効ファイルデータの量は最小限に抑えられる。その結果、フラッシュメモリシステムの寿命にわたって性能と耐久性が大幅に向上する。

【００１０】

メモリシステムの代わりにホストでダイレクトデータファイル管理システムを実装する場合も、そのような性能・耐久性の向上を実現することができる。ホストとメモリシステムとの間には引き続きＬＢＡインターフェイスが存在する。しかし、クラスタのファイルデータをこの単一連続論理アドレス空間に割り振るのではなく、メモリシステム内の物理ブロックに対応するこの空間内の論理アドレスブロックにファイルデータを割り振る。フラッシュメモリシステムの中で、物理メモリセルブロックに対して実施される、前に相互参照した特許出願のファイルデータ管理手法は、ホストの中で、ホスト／メモリシステムインターフェイスの論理アドレス空間の中で連続するアドレスの論理ブロックに対して実施される。この場合のメモリシステムは、現在商業的に普及しているＬＢＡインターフェイスを備える従来のメモリシステムでよい。メモリシステムで作動するダイレクトデータファイルシステムによって、２つ以上のファイルのデータを収容する物理メモリセルブロック数が制限されるように、ホストの中で作動するダイレクトデータファイル管理システムによって、２つ以上のファイルのデータを収容する論理ブロック数は制限できる。物理メモリセルブロックにおける個々のファイルオブジェクトデータの断片化も同様に減少するが、これは物理メモリセルブロックにマップされる論理アドレス空間のブロックを管理することによって達成する。

【００１１】

ＬＢＡインターフェイスの論理ブロックは、好ましくはデータ蓄積容量等が共通するメモリシステムの物理ブロックにマップされる。具体的に、ホストのダイレクトデータファイルシステムにとっては、ホストによって構成される論理ブロックが、メモリシステムの中でダイレクトデータファイルシステムが作動した場合の物理ブロックと同様に映る。物理メモリブロックの特性、すなわち通常ホストに提供されない情報は、メモリシステムの初期化のときにメモリシステムからホストに提供できる。そして、ホストは連続論理アドレス空間を、物理メモリのブロックに特性が一致するブロックに構成し、その後、それらの論理ブロックの中のアドレスにデータを書き込む。

【００１２】

代案として、ダイレクトデータファイルシステムをホストで実装する代わりにメモリシステムで操作し、前述したのと同様に、メモリシステムのＬＢＡインターフェイスの連続アドレス空間にわたって論理ブロックを設定することもできる。このダイレクトデータファイル操作は、メモリシステムの一部であっても、前に相互参照した特許出願で説明された例とは異なる。先行出願の例のように、メモリシステムのバックエンドを操作してＬＢＡインターフェイスに代わってメモリシステムでファイルデータを受け付ける代わりに、メモリシステムのＬＢＡインターフェイスの手前にダイレクトデータファイルシステムを追加し、これを前述したのと同じ要領で、あたかもホストのＬＢＡインターフェイスの手

前にあるかのごとく、操作することができる。そのようなメモリシステムにＬＢＡインターフェイスとファイルオブジェクトインターフェイスの両方を設け、両タイプのインターフェイスのうち、いずれか一方のインターフェイスしかないホストと通信することもできる。これは特に、様々なタイプのホスト装置に取り外し可能な状態で接続するように作られたメモリカードの場合に便利である。

【００１３】

さらなる代案として、処理能力を持つ取り外し可能なマザーカードに前述したダイレクトデータファイルシステムを設け、ダイレクトファイル機能はなくともダイレクトデータファイルインターフェイスはあるホストに、ダイレクトファイル機能を追加することもできる。ホストに接続されたマザーカードはその出力でＬＢＡインターフェイスを提供し、カードの出力には、ＬＢＡインターフェイスを備える標準的なメモリカードを取り外し可能な状態で接続できる。

【００１４】

この後に続く本発明の代表的な例の説明には本発明のさらなる態様と利点と特徴が記載されているが、この説明は添付の図面と併せて解釈するべきものである。

【００１５】

ここで参照する特許、特許出願、記事、書籍、仕様書、その他の出版物、文書、事物はどれも、あらゆる目的のためにその全体が本願明細書において参照により援用されている。援用する出版物、文書、または事物のいずれかと本願明細書の本文との間で用語の定義または使用に矛盾や食い違いがある場合は、本願明細書における用語の定義または使用が優先するものとする。

【図面の簡単な説明】

【００１６】

【図１】ホストおよび接続された不揮発性メモリシステムを概略的に示す。

【図２】図１の不揮発性メモリとして使用されるフラッシュメモリシステム例のブロック図である。

【図３】図２のシステムに使用できるメモリセルアレイの代表的な回路図である。

【図４】図２のシステムの物理メモリ編制例を示す。

【図５】図４の物理メモリの一部分の拡大図を示す。

【図６】図４および図５の物理メモリの一部分のさらなる拡大図を示す。

【図７Ａ】再プログラム可能なメモリシステムを操作する３通りの方法の内のひとつを示し、対比する。

【図７Ｂ】再プログラム可能なメモリシステムを操作する３通りの方法の内のひとつを示し、対比する。

【図７Ｃ】再プログラム可能なメモリシステムを操作する３通りの方法の内のひとつを示し、対比する。

【図８Ａ】図７Ａ、図７Ｂ、および図７Ｃにそれぞれ見られる再プログラム可能なメモリシステムを操作する３通りの方法のひとつと、ホストシステムとのインターフェイスを示し、対比する。

【図８Ｂ】図７Ａ、図７Ｂ、および図７Ｃにそれぞれ見られる再プログラム可能なメモリシステムを操作する３通りの方法のひとつと、ホストシステムとのインターフェイスを示し、対比する。

【図８Ｃ】図７Ａ、図７Ｂ、および図７Ｃにそれぞれ見られる再プログラム可能なメモリシステムを操作する３通りの方法のひとつと、ホストシステムとのインターフェイスを示し、対比する。

【図９Ａ】図８Ａ、図８Ｂ、および図８Ｃにそれぞれ見られる再プログラム可能なメモリシステムを操作する３通りの方法のひとつと、ホストとのインターフェイスを示し、対比する。

【図９Ｂ】図８Ａ、図８Ｂ、および図８Ｃにそれぞれ見られる再プログラム可能なメモリシステムを操作する３通りの方法のひとつと、ホストとのインターフェイスを示し、対比

する。
【図９Ｃ】図８Ａ、図８Ｂ、および図８Ｃにそれぞれ見られる再プログラム可能なメモリシステムを操作する３通りの方法のひとつと、ホストとのインターフェイスを示し、対比する。
【図１０】図９Ｃの手法の遂行にあたって使用できる論理－物理ブロックマッピングの一例を示す。
【図１１】図９Ｃおよび図１０に示す手法の遂行にあたってパラメータを設定するためのホストおよびメモリシステム間のやり取りを示す。
【図１２】ダイレクトデータファイルシステムの動作サイクルを示す。
【図１３Ａ】ファイルデータを書き込む４例中のひとつを示す。
【図１３Ｂ】ファイルデータを書き込む４例中のひとつを示す。
【図１３Ｃ】ファイルデータを書き込む４例中のひとつを示す。
【図１３Ｄ】ファイルデータを書き込む４例中のひとつを示す。
【図１４Ａ】一連の単一データファイル書き込みを示す。
【図１４Ｂ】一連の単一データファイル書き込みを示す。
【図１４Ｃ】一連の単一データファイル書き込みを示す。
【図１４Ｄ】一連の単一データファイル書き込みを示す。
【図１４Ｅ】一連の単一データファイル書き込みを示す。
【図１５】図１４Ｅのブロックを再生した結果を示す。
【図１６Ａ】様々なブロックタイプの組み合わせで蓄積されるデータファイルの例を示す。
【図１６Ｂ】様々なブロックタイプの組み合わせで蓄積されるデータファイルの例を示す。
【図１６Ｃ】様々なブロックタイプの組み合わせで蓄積されるデータファイルの例を示す。
【図１６Ｄ】様々なブロックタイプの組み合わせで蓄積されるデータファイルの例を示す。
【図１７】具体例に従い許容ファイル状態を提示する表である。
【図１８】プログラムデータに基づく許容ファイル状態遷移を示す状態図である。
【図１９】図１８に見られるファイル状態遷移を説明する表である。
【図２０】用済みデータに基づく許容ファイル状態遷移を示す状態図である。
【図２１】図２０に見られるファイル状態遷移を説明する表である。
【図２２】再生ブロックに基づく許容ファイル状態遷移を示す状態図である。
【図２３】図２２に見られるファイル状態遷移を説明する表である。
【図２４】データファイルと論理ブロックとの整合の一実施形態を示す。
【図２５】図２４のデータ整合の実施形態で、様々な状況のもとでのアクティブブロック割り当てを示す表である。
【図２６】データファイルと論理ブロックとの整合の代替の実施形態を示す。
【図２７】図２６のデータ整合の実施形態で、様々な状況のもとでのアクティブブロック割り当てを示す表である。
【図２８Ａ】ブロック再生操作の例を示す。
【図２８Ｂ】ブロック再生操作の例を示す。
【図２８Ｃ】ブロック再生操作の例を示す。
【図２８Ｄ】ブロック再生操作の例を示す。
【図２９】再生操作を一般的な用語で説明するフローチャートである。
【図３０】典型的なパーシャルメモリセルブロックに蓄積されるデータのタイプを示す。
【図３１】図２９のフローチャートの１ステップを遂行する具体的な実施形態の詳細を提示する。
【図３２】図２９のフローチャートの同じステップを実行する代替の実施形態の詳細を提示する。

【図３３】別の実施形態で２つのブロックリストに入るブロックタイプを明らかにする表である。

【発明を実施するための形態】

【００１７】

フラッシュメモリシステムの概説

一般的なフラッシュメモリシステムを図１～図６との関係で説明する。そのようなシステムで本発明の様々な態様を実装できる。図１のホストシステム１は、フラッシュメモリ２の中にデータを蓄積し、このフラッシュメモリからデータを引き出す。フラッシュメモリはホストの中に埋め込むこともできるが、メモリ２はより一般的なカードの形で図に示され、このカードは、機械的および電気的コネクタの嵌合部分３および４を通じて取り外し可能な状態でホストへ接続される。例えばコンパクトフラッシュ（ＣＦ）、マルチメディアカード（ＭＭＣ）、セキュアデジタル（ＳＤ）、ミニＳＤ、メモリスティック、スマートメディア、トランスフラッシュカード等、様々なフラッシュメモリカードが現在市販されている。これらのカードはいずれも、それぞれの規格化された仕様に従い特有の機械的および／または電気的インターフェイスを備えているが、それぞれに内蔵されたフラッシュメモリシステムはよく似ている。これらのカードはいずれも、本願の出願人であるサンディスク　コーポレイションから入手できる。また、サンディスク　コーポレイションは、Ｃｒｕｚｅｒという商標のもとで一連のフラッシュドライブを提供し、このフラッシュドライブはユニバーサルシリアルバス（ＵＳＢ）プラグを備える小型の手持ち式メモリシステムで、これをホストのＵＳＢ差込口に差し込むことによりホストと接続する。これらのメモリカードとフラッシュドライブはコントローラを内蔵し、このコントローラがホストと連係し、内蔵されたフラッシュメモリの動作を制御する。

【００１８】

そのようなメモリカードやフラッシュドライブを使用するホストシステムは数多くあり様々である。ここにはパーソナルコンピュータ（ＰＣ）、ラップトップをはじめとするポータブルコンピュータ、携帯電話機、個人用携帯情報端末（ＰＤＡ）、デジタル静止画カメラ、デジタル動画カメラ、ポータブルオーディオプレイヤ等が含まれる。ホストは通常ならば１種類以上のメモリカードまたはフラッシュドライブのための一体化された差込口を内蔵するが、メモリカードを差し込むアダプタが必要なものもある。

【００１９】

図１のホストシステム１は、メモリ２に関する限り、回路とソフトウェアとの組み合わせからなる２つの主要部分を備えるとみなすことができる。それらはアプリケーション部５と、メモリ２と連係するドライバ部６である。例えばパーソナルコンピュータの場合、アプリケーション部５はプロセッサを含み、ワープロ、グラフィック、コントロール等、一般的なアプリケーションソフトウェアを実行する。カメラや携帯電話機等、専ら１組の機能を遂行する専用ホストシステムの場合、アプリケーション部５は、カメラを操作しながら写真を撮影したり蓄積したりするソフトウェアや、携帯電話機を操作しながら電話をかけたり受けたりするソフトウェアを含む。

【００２０】

図１のメモリシステム２はフラッシュメモリ７と回路８とを含み、この回路はカードが接続されるホストと連係しながらデータをやり取りし、メモリ７を制御する。コントローラ８は通常、データのプログラミングと読み出しのときにホスト１によって使用されるデータの論理アドレスとメモリ７の物理アドレスとの変換を行う。

【００２１】

図２を参照すると、図１の不揮発性メモリ２として使用できる典型的なフラッシュメモリシステムの回路を説明する。システムコントローラは通常、システムバス１３沿いに１つ以上の集積回路メモリチップと並列に接続される単一集積回路チップ１１上に実装され、図２にはただひとつのそのようなメモリチップ１５が示されている。図に示されたバス１３は、データを搬送する１セットの導体１７と、メモリアドレスのためのセット１９と、制御および状態信号のためのセット２１とを含む。代わりに、これらの３つの機能で１

セットの導体を時分割共用することもできる。さらに、２００４年８月９日に出願された「Ring Bus Structure and It's Use in Flash Memory Systems 」という米国特許出願第１０／９１５，０３９号（米国公開特許出願第２００６／００３１５９３号）（特許文献３）で説明されているリングバス等、これとは別のシステムバス構成を使用できる。

【００２２】

典型的なコントローラチップ１１は、インターフェイス回路２５を通じてシステムバス１３と連係する独自の内部バス２３を有する。このバスへ通常接続される主要機能には、プロセッサ２７（マイクロプロセッサ、マイクロコントローラ等）と、システムの初期化（「ブート」）コードを収容する読み出し専用メモリ（ＲＯＭ）２９と、主にメモリとホストとの間で転送されるデータをバッファするために使われるランダムアクセスメモリ（ＲＡＭ）３１とがあり、コントローラを通じてメモリとホストの間を行き来するデータで誤り訂正符号（ＥＣＣ）を計算し検査する回路３３をバス２３へ接続することもできる。コントローラを通過するデータを符号化し復号化する専用回路３４を盛り込むこともできる。そのような符号化には圧縮やセキュリティ暗号化等があるが、ほとんどのデータ変換はこのやり方で果たすことができる。専用回路３３および３４を使用する場合は、ファームウェア制御のもとでプロセッサ２７によって実行されるアルゴリズムが、これらの回路によって実行される。コントローラバス２３は回路３５を通じてホストシステムと連係し、メモリカードに内蔵される図２のシステムの場合に、コネクタ４の一部をなすカードの外部接点３７を通じて果たされる。クロック３９はコントローラ１１の他のコンポーネントに接続され、それらのコンポーネントによって利用される。

【００２３】

メモリチップ１５と、システムバス１３に接続される他のメモリチップは通常、複数のサブアレイまたはプレーンに編制されたメモリセルアレイを含み、簡潔を図るために２つのそのようなプレーン４１および４３が図に示されているが、これよりも多いプレーン、例えば４つ、または８つのプレーンを代わりに使用することもできる。あるいは、チップ１５のメモリセルアレイはプレーンに分割しない場合もある。しかし、分割するなら、各プレーンは互いに独立して作動する独自の列制御回路４５および４７を有する。回路４５および４７は、システムバス１３のアドレス部１９からそれぞれのメモリセルアレイのアドレスを受け取り、それらを復号化して１つ以上のビット線４９および５１をアドレスする。ワード線５３は、アドレスバス１９で受け取るアドレスに応じて行制御回路５５によりアドレスされる。ソース電圧制御回路５７および５９もそれぞれのプレーンに接続し、ｐウェル電圧制御回路６１および６３も同様である。メモリチップ１５が単一のメモリセルアレイを有し、２つ以上のそのようなチップがシステムに存在する場合は、前述したマルチプレーンチップにおけるプレーンまたはサブアレイと同様に各チップのアレイを操作することができる。

【００２４】

データは、システムバス１３のデータ部１７に接続されたデータ入出力回路６５および６７を通じてプレーン４１および４３を出入りする。回路６５および６７は、それぞれの列制御回路４５および４７を介してプレーンへ接続する線６９および７１を通じてそれぞれのプレーンのメモリセルにデータをプログラムし、メモリセルからデータを読み出すためにある。

【００２５】

コントローラ１１はデータをプログラムするため、データを読み出すため、消去するため、様々なハウスキーピング作業に対処するため、メモリチップ１５の動作を制御するが、各々のメモリチップもコントローラ１１からのコマンドを実行してそのような機能を遂行する制御回路を内蔵する。インターフェイス回路７３はシステムバス１３の制御・状態部２１へ接続する。コントローラからのコマンドは状態マシン７５へ提供され、この状態マシンは、これらのコマンドを実行するために他の回路を制御する。制御線７７～８１は、状態マシン７５を図２に見られるこれらの他の回路に接続する。状態マシン７５からの状態情報は線８３に沿ってインターフェイス７３へ伝達され、バス部２１に沿ってコント

ローラ１１へ送信される。

【００２６】

現在はメモリセルアレイ４１および４３のＮＡＮＤアーキテクチャが好まれているが、ＮＯＲ等、これとは別のアーキテクチャを代わりに使用することもできる。ＮＡＮＤフラッシュメモリと、メモリシステムの一部としてのこれの動作の例は、米国特許第５，５７０，３１５号（特許文献４）、第５，７７４，３９７号（特許文献５）、第６，０４６，９３５号（特許文献６）、第６，３７３，７４６号（特許文献７）、第６，４５６，５２８号（特許文献８）、第６，５２２，５８０号（特許文献９）、第６，７７１，５３６号（特許文献１０）、および第６，７８１，８７７号（特許文献１１）と米国公開特許出願第２００３／０１４７２７８号（特許文献１２）とで参照できる。

【００２７】

図２のメモリシステムのメモリセルアレイ４１の一部分にあたる図３の回路図に、ＮＡＮＤアレイの例を示す。多数のグローバルビット線が提供されるが、説明を簡潔にするため、図２には４つのそのような線９１～９４だけが示されている。これらのビット線のうちの１ビット線と基準電位との間には、いくつかの直列接続メモリセルストリング９７～１０４が接続される。メモリセルストリング９９を代表としてとりあげ、ストリング両端の選択トランジスタ１１１および１１２には複数の電荷蓄積メモリセル１０７～１１０が直列で接続される。１ストリングの選択トランジスタが通電すると、そのビット線と基準電位との間でこのストリングが接続される。そして、そのストリングの中で一度に１つのメモリセルのプログラミングか読み出しが行われる。

【００２８】

数あるメモリセルストリングの１メモリセルの電荷蓄積素子にわたって図３のワード線１１５～１１８が延在し、ゲート１１９および１２０は、ストリングの末端にある選択トランジスタの状態を制御する。共通のワード線およびコントロールゲート線１１５～１２０を共用するメモリセルストリングが、まとめて消去されるメモリセルのブロック１２３を形成する。このセルからなるブロックは、一度に物理的に消去できる最小数のセルを収容する。ワード線１１５～１１８のうちの１ワード線沿いの１行のメモリセルが一度にプログラムされる。通常、ＮＡＮＤアレイの行は規定の順序でプログラムされ、この場合は、アース等の共通電位へ接続されたストリングの末端に最も近いワード線１１８沿いの行から始まる。次にワード線１１７沿いのメモリセル行がプログラムされ、ブロック１２３の全体を通じて同様に進む。最後にワード線１１５沿いの行がプログラムされる。

【００２９】

第２のブロック１２５も類似し、そのメモリセルストリングは第１のブロック１２３のストリングと同じグローバルビット線へ接続されているが、ワード線とコントロールゲート線は異なる。ワード線とコントロールゲート線は、行制御回路５５によって適切な作動電圧まで駆動される。図２のプレーン１および２等、２つ以上のプレーンまたはサブアレイがシステムに存在する場合は、それらの間に延在する共通のワード線を１つのメモリアーキテクチャで使用する。代わりに、３つ以上のプレーンまたはサブアレイが存在して共通のワード線を共用することもある。これとは別のメモリアーキテクチャでは、各プレーンまたはサブアレイのワード線を別々に駆動する。

【００３０】

前に参照したいくつかのＮＡＮＤ特許および公開特許出願で説明されているように、３つ以上の検出可能な電荷レベルを各電荷蓄積素子または領域に蓄積することにより、２ビット以上のデータを各々蓄積するようメモリシステムを操作できる。メモリセルの電荷蓄積素子は一般的には導電性フローティングゲートだが、米国特許第６，９２５，００７号（米国特許１３）で説明されている非導電性誘電性電荷捕獲材であってもよい。

【００３１】

図４は、以降のさらなる説明で一例として使用するフラッシュメモリセルアレイ７（図１）の編制を概念的に示すものである。４つのメモリセルプレーンまたはサブアレイ１３１～１３４は、１つの集積メモリセルチップ上に存在することもあれば、２つのチップ（

各チップ上に２プレーンずつ）または４つの別々のチップ上に存在することもある。具体的な配置は以降の論述にとって重要ではない。勿論、システムに存在するプレーンの数はこれに限らず、例えば１、２、８、１６またはそれ以上のプレーンが存在することもある。プレーン１３１～１３４に位置するブロック１３７、１３８、１３９、１４０等、プレーンは図４にて矩形で表示されたメモリセルブロックにそれぞれ分割される。各プレーンには何十、何百ものブロックが存在し得る。前述したように、メモリセルのブロックは消去の単位であって、物理的にまとめて消去できる最小数のメモリセルである。しかし、並列性を高めるためには、これよりも大きいメタブロック単位でブロックを操作する。メタブロックは、各プレーンの１ブロックを論理的にリンクすることによって形成される。４つのブロック１３７～１４０によって１つのメタブロック１４１が形成される様子が図に示されている。通常ならば、１メタブロック内の全てのセルをまとめて消去する。ブロック１４５～１４８からなる第２のメタブロック１４３に見られるように、メタブロックを形成するブロックの各プレーンにおける相対位置は同じでなくてもよい。システム性能を高めるため、通常は全てのプレーンにまたがって延在するメタブロックが好ましいが、別々のプレーンにある１ブロック、２ブロック、または３ブロックのいずれかまたは全部から動的にメタブロックを形成しながらメモリシステムを操作することもできる。この場合は、１回のプログラミング操作で蓄積するデータ量にメタブロックのサイズをより近づけることが可能となる。

【００３２】

操作上の目的から、各ブロックはさらに図５に示すようにメモリセルページに分割される。例えばブロック１３１～１３４のメモリセルは、それぞれ８つのページＰ０～Ｐ７に分割されている。代わりに、１６、３２、またはそれ以上のメモリセルページが各ブロックに存在することもある。ページはブロックの中でデータをプログラムし読み出す単位であり、一度にプログラムされる最少量のデータを収容する。図３のＮＡＮＤアーキテクチャでは、ブロックの中でワード線沿いのメモリセルからページが形成される。しかし、メモリシステム動作の並列性を高めるため、２つ以上のブロックにあるページをメタページとして論理的にリンクすることもできる。図５には、４つのブロック１３１～１３４の各ブロックにつき１物理ページからなるメタページ１５１が示されている。例えばメタページ１５１は４つのブロックのページＰ２を含んでいるが、メタページのページは必ずしも各ブロック内で同じ相対位置を占めるとは限らない。システム性能を高めるためには、全４つのプレーンにわたって最大量のデータを並行してプログラムし読み出すのが望ましいが、別々のプレーンにあるブロックの１ページ、２ページ、または３ページのいずれかまたは全部からメタページを形成しながらメモリシステムを操作することもできる。この場合は、並列処理のデータ量に応じたプログラミング操作と読み出し操作が可能になるほか、メタページの一部にデータがプログラムされずに残る事態は少なくなる。

【００３３】

図５に示す複数プレーンの物理ページからなるメタページは、それらの複数プレーンのワード線行沿いのメモリセルを含む。１ワード線行の全セルを同時にプログラムするよりは、２つ以上のインターリーブされたグループでそれらを交互にプログラムするほうが一般的であり、各グループは、（単一ブロック内の）１ページのデータか（複数ブロックにまたがる）１メタページのデータを蓄積する。交互のメモリセルを一度にプログラムすることにより、データレジスタやセンス増幅器を含むひとまとまりの周辺回路をビット線ごとに用意する必要はなくなり、それらの回路は隣接するビット線で時分割共用する。こうすることで周辺回路に要する基板スペースを節約し、行沿いのメモリセル実装密度を増すことができる。さもなくば、行沿いの全セルを同時にプログラムしてメモリシステムから最大限の並列性を引き出すのが望ましい。

【００３４】

図３を参照すると、行沿いの互い違いのメモリセルへのデータの同時プログラミングを最も簡便に果たすには、ＮＡＮＤストリングの少なくとも一端に沿って２行の選択トランジスタ（図示せず）を、図に示された１行の代わりに、提供する。この場合、一方の行の

選択トランジスタは、１つの制御信号に応じてブロック内の互い違いのストリングをそれぞれのビット線へ接続し、他方の行の選択トランジスタは、別の制御信号に応じて介在する互い違いのストリングをそれぞれのビット線へ接続する。その結果、メモリセルの各行には２ページ分のデータが書き込まれる。

【００３５】

各論理ページのデータ量は通常、整数にして１セクタ数以上のデータであり、各セクタは慣例上５１２バイトのデータを収容する。図６は、２セクタ１５３および１５５分のページまたはメタページデータからなる論理データページを示す。各セクタは通常、５１２バイトのユーザまたはシステムデータを収容する部分１５７と、部分１５７のデータに関係する、またはこれを蓄積する物理ページまたはブロックに関係する、オーバーヘッドデータのため、さらなるバイト数１５９とを含む。オーバーヘッドデータのバイト数は通常ならば１６バイトであり、セクタ１５３および１５５の各々につき合計５２８バイトになる。オーバーヘッド部分１５９には、プログラミング中にデータ部分１５７から算出されるＥＣＣ、その論理アドレス、ブロックが消去され再プログラムされた回数の経験カウント、１つ以上の制御フラグ、作動電圧レベル、および／またはその他に加え、そのようなオーバーヘッドデータ１５９から算出されるＥＣＣを収容できる。あるいは、オーバーヘッドデータ１５９またはこれの一部分を別のブロックの異なるページに蓄積することもできる。

【００３６】

メモリの並列性が高まるにつれメタブロックのデータ蓄積容量は増し、その結果、データページおよびメタページのサイズも増す。データページは３セクタ以上のデータを収容することがある。１データページ内に２セクタ、そして各メタページにつき２データページで、１メタページのセクタ数は４セクタになる。したがって、各メタページで２，０４８バイトのデータを蓄積することになる。これは高度な並列性であり、行内のメモリセル数の増加にともないさらに高めることができる。このような理由から、ページおよびメタページ内のデータ量を増やすためにフラッシュメモリの幅は拡大されつつある。

【００３７】

前述した物理的に小さい再プログラム可能な不揮発性メモリカードおよびフラッシュドライブは市販され、データ蓄積容量は５１２メガバイト（ＭＢ）、１ギガバイト（ＧＢ）、２ＧＢ、４ＧＢおよびそれ以上になる。

【００３８】

ファイルオブジェクト操作手法

論理ブロック（ＬＢＡ）メモリ／ホストインターフェイスによる操作

図７Ａ、図８Ａ、および図９Ａには、ホストおよびメモリシステム間の一般的な論理インターフェイスがそれぞれ異なる形式で示されている。ホストによって生成されるデータファイルには、通常ならばマルチセクタデータからなるクラスタの単位で連続システムアドレス空間（ＬＢＡインターフェイス）の論理アドレスがホストによって割り振られる。そして、メモリシステムはこれらの論理アドレスを認識し、データが実際に蓄積されるメモリセルブロックの物理アドレスにマップする。

【００３９】

具体的に図９Ａを参照すると、連続する論理アドレス空間１６１には、メモリシステムに蓄積される全データにアドレスを提供するにあたって十分な大きさがある。通常、ホストアドレス空間はデータクラスタの単位に分割される。ホストシステムではいくつかのデータセクタを収容するように各クラスタを設計でき、４～６４セクタあたりが一般的である。標準的なセクタは５１２バイトのユーザデータに加え、オプションとして何バイトかのオーバーヘッドデータ、通常ならば１６バイトのオーバーヘッドデータを、収容し、全部で５２８バイトになる。

【００４０】

図９Ａは、ホストと、メモリカードやフラッシュドライブに見られる大容量メモリシステムとの間の最も一般的なインターフェイスを示す。ホストで扱うデータファイルは、ホ

ストによって実行されるアプリケーションソフトウェアかファームウェアプログラムによって生成または使用される。「ファイル」または「ファイルオブジェクト」は、何らかの用途または目的のためにホストにより所定の実体として認識されるひとまとまりのデータを意味する。ファイルオブジェクトのデータはひとつの単位として扱われる。ワープロファイルのデータはその一例であり、コンピュータ支援設計（ＣＡＤ）ソフトウェアの描画ファイルのデータもこれにあたり、主にＰＣ、ラップトップコンピュータ等、一般的なコンピュータホストに見られる。ｐｄｆ形式文書のデータもそのようなファイルである。ファイルオブジェクトのデータはアプリケーションプログラムの実行中にホストによって生成されるか、別のところで生成されてホストに提供される。静止画デジタルビデオカメラは写真ごとにデータファイルを生成し、メモリカードに蓄積する。携帯電話機は、電話帳等、内蔵メモリカード上のファイルからデータを利用する。ＰＤＡは、住所ファイル、カレンダーファイル等、数通りのファイルのデータを蓄積し使用する。そのような用途においては、ホストを操作するソフトウェアがメモリカードに内蔵されることもある。

【００４１】

図９Ａの例では、３つのファイル１、２、および３が作成されたものとして示されている。ホストシステムで実行するアプリケーションプログラムは、整頓された１組のデータとして各ファイルを作成し、一意な名前かその他の参照符によってこれを識別する。ファイル１には、別のファイルにまだ割り振られていない十分な使用可能な論理アドレス空間がホストによって割り当てられる。ファイル１は、一連の使用可能な論理アドレス範囲が割り当てられた状態で図に示されている。このほかに、通常ならばホストオペレーティングソフトウェアのための特定のアドレス範囲等、特定の目的のためにアドレス範囲が割り振られ、それらは、たとえホストがデータに論理アドレスを割り当てるときにまだ使われていなくとも、データの蓄積には使われない。

【００４２】

図９Ａに見られるように、ホストは後ほどファイル２が作成されるときにも同様に、論理アドレス空間１６１の中の２つの別々の隣接アドレス範囲を割り当てる。隣接する論理アドレスをファイルに割り当てる必要はなく、既に他のファイルに割り振られているアドレス範囲の間にあるアドレスの断片であってもよい。この例はさらに、ホストによって作成されたもうひとつのファイル３に、ファイル１および２やその他のデータにまだ割り振られていないホストアドレス空間の別の部分を割り振る様子を示している。

【００４３】

ホストは、ファイルアロケーションテーブル（ＦＡＴ）を管理することによってメモリ論理アドレス空間を絶えず把握し、ホストが様々なホストファイルに割り当てる論理アドレスは、このファイルアロケーションテーブルの中で管理する。通常、ＦＡＴテーブルは不揮発性メモリとホストメモリに蓄積され、新しいファイルが蓄積されるとき、他のファイルが削除されるとき、ファイルが修正されるとき等に、ホストによって頻繁に更新される。ホストは、例えばホストファイルが削除されるときにＦＡＴテーブルを更新することによって削除ファイルに割り振られていた論理アドレスを解除し、それらの論理アドレスが別のデータファイルに使用できることを明らかにする。

【００４４】

ホストは、ファイルの蓄積にあたってメモリシステムコントローラが選択する物理位置を考慮しない。典型的なホストは、その論理アドレス空間と、ホストが種々のファイルに割り振った論理アドレスを、認識するにすぎない。他方、メモリシステムは、典型的なＬＢＡホスト／カードインターフェイスを通じて、論理アドレス空間のうちのデータが書き込まれた部分だけを認識し、特定のホストファイルへ割り振られる論理アドレスは認識せず、ホストファイルの数すら認識しない。メモリシステムコントローラは、データの蓄積や引き出しのためにホストから提供される論理アドレスを、ホストデータを蓄積するフラッシュメモリセルアレイの中の一意な物理アドレスに変換する。ブロック１６３は、メモリシステムコントローラによって管理される論理－物理アドレス変換の作業テーブルを表している。

【００４５】
メモリシステムコントローラは、高度なシステム性能を維持しながらメモリアレイ１６５のブロックおよびメタブロックの中でデータファイルを蓄積するようにプログラムされる。この例では４つのプレーンまたはサブアレイが使われている。データは好ましくは、各プレーンのブロックから形成されたメタブロック全体にわたってシステムが許す最大限の並列度でプログラムされ、読み出される。通常は、メモリコントローラによって使用されるオペレーティングファームウェアおよびデータを蓄積する予約ブロックとして、少なくとも１つのメタブロック１６７が割り振られる。ホストオペレーティングソフトウェアやホストＦＡＴテーブル等の蓄積のため、別のメタブロック１６９または複数のメタブロックを割り振ることができる。物理蓄積容量のほとんどはデータファイルの蓄積用として残る。しかし、メモリコントローラは、様々なファイルオブジェクトの中で受信データがホストによってどのように割り振られているかを認識しない。通常、メモリコントローラがホストとのやり取りを通じて知ることは、ホストによって特定の論理アドレスに書き込まれるデータが対応する物理アドレスに蓄積されるということだけであって、これはコントローラの論理－物理アドレステーブル１６３によって管理される。

【００４６】
典型的なメモリシステムにおいて、アドレス空間１６１の中でデータを蓄積するのとは別に数ブロック分の蓄積容量を余分に用意する。メモリの寿命の中で別のブロックが故障した場合に代用される冗長ブロックとして、これらの余分のブロックを１つ以上用意することができる。当初メタブロックに割り当てられていた欠陥ブロックのための冗長ブロックを代用する等、個々のメタブロックにおけるブロックの論理的分類は通常、様々な理由から変化する。消去済みブロックのプールでは通例、メタブロック１７１等、１つ以上の追加ブロックを保守する。コントローラはホストがメモリシステムにデータを書き込むときに、ホストによって割り当てられた論理アドレスを、消去済みブロックプールにあるメタブロック内の物理アドレスに変換する。そして、論理アドレス空間１６１の中でデータ蓄積に使われていないほかのメタブロックは消去され、以降のデータ書き込み操作のときに使用するために消去済みプールのブロックとして指定される。

【００４７】
特定のホスト論理アドレスに蓄積されたデータは、当初の蓄積データが用済みになると新規データによって頻繁に上書きされる。これに応じてメモリシステムコントローラは新規データを消去済みブロックに書き込み、論理アドレスのデータを蓄積する新たな物理ブロックを明らかにするためにこれらの論理アドレスに関し論理－物理アドレステーブルを変更する。そして、それらの論理アドレスのところで当初のデータを収容するブロックは消去され、新規データの蓄積に使えるようになる。書き込みが始まるときに消去ブロックプールの消去済みブロックに十分な蓄積容量がない場合は、データ書き込み操作を完了する前にこのような消去を頻繁に行わなければならない。このため、システムのデータプログラミング速度が損なわれるおそれがある。メモリコントローラは通常、ある特定の論理アドレスにあるデータがホストによって用済みとされていることを、ホストがそれと同じ論理アドレスに新しいデータを書き込むときになって初めて知る。したがって、メモリのブロックの多くは、そのような無効データを暫くの間蓄積することがある。

【００４８】
集積回路メモリチップの領域を効率よく運用するため、商用メモリシステムに使われるブロックとメタブロックのサイズは拡大している。その結果、データ書き込みの大半で蓄積されるデータの量はメタブロックの蓄積容量に満たなく、多くの場合、ブロックの蓄積容量にすら満たない。メモリシステムコントローラは通常、新規データを消去済みプールのメタブロックへ誘導するため、メタブロックには埋まらない部分が生じる。新規データが別のメタブロックに蓄積されたデータの更新にあたる場合は、その別のメタブロックで論理アドレスが新規データメタページの論理アドレスと隣接する残りの有効データメタページもまた、望ましくは論理アドレスの順序で新しいメタブロックにコピーする。古いメタブロックは他の有効データメタページを保持することがある。その結果、メタブロック

のいずれかのメタページのデータはいずれ用済み、無効となり、同じ論理アドレスにより異なるメタブロックに書き込まれる新規データで差し替えられる。

【００４９】

論理アドレス空間１６１の全体にわたってデータ蓄積のための十分な物理メモリ空間を維持するため、データの圧縮または整理統合（ガーベッジコレクション）を定期的に行ってブロックを再生し、消去済みブロックのプールに加える。メタブロックの中でデータセクタをできる限り論理アドレスと同じ順序に保つことも望ましく、こうすることで連続する論理アドレスでデータの読み出し効率が上がる。そこで通常は、この目的のためにもデータの圧縮とガーベッジコレクションが行われる。米国特許第６，７６３，４２４号（特許文献１４）には、部分的ブロックデータ更新を受け取るときのメモリ管理とメタブロック使用の態様がいくつか記載されている。

【００５０】

データ圧縮では通常、メタブロックから有効データメタページを全て読み出して別のブロックに書き込み、その過程で無効データを含むメタページは無視する。また、有効データを含むメタページは、好ましくはそこに蓄積されたデータの論理アドレスの順序に一致する物理アドレス順に配置する。無効データを収容するメタページは新しいメタブロックへコピーされないため、新しいメタブロックに占めるメタページの数は古いメタブロックに占めるメタページの数を下回ることになる。そして、古いブロックを消去し、消去済みブロックプールへ加えて新規データの蓄積に使えるようにする。この整理統合によって得られる追加メタページ容量は、ほかのデータの蓄積に役立てることができる。

【００５１】

ガーベッジコレクションのときには、２つ以上のメタブロックから論理アドレスが隣接するかほぼ隣接する有効データのメタページが回収され、別のメタブロック、通常ならば消去済みブロックプールのメタブロックに書き換えられる。当初の２つ以上のメタブロックから全ての有効データメタページをコピーすると、先々の使用に向けてそれらを消去できる。別々のブロックに蓄積されたファイルの断片化が増すにつれ、データの整理統合やガーベッジコレクションの回数も増える。

【００５２】

データの整理統合とガーベッジコレクションには時間がかかり、ホストからのコマンドの実行に先立ちデータの整理統合やガーベッジコレクションを行う必要がある場合は特にメモリシステムの性能に影響する。メモリシステムコントローラは通常、そのような操作をできるだけバックグラウンドで行うようスケジュールを組むが、これらの操作の実行にあたっては、コントローラは操作が完了するまでビジー状態信号をホストに提供しなければならない。ホストコマンドの実行が遅れる一例として、ホストがメモリに書き込もうとする全データの蓄積にあたって十分な消去済みメタブロックが消去済みブロックプールにない場合は、事前にデータの整理統合かガーベッジコレクションで１つ以上のメタブロックから有効データを片づける必要があり、その後にメタブロックを消去する。これまで、そのような混乱を最小限に抑えるためにメモリ制御の管理に注意が払われてきた。そのような手法が、２００３年１２月３０日に出願された「Management of Non-Volatile Memory Systems Having Large Erase Blocks 」という米国特許出願第１０／７４９，８３１号（米国公開特許出願第２００５／０１４４３５８号）（特許文献１５）、２００３年１２月３０日に出願された「Non-Volatile Memory and Method with Block Management System 」という米国特許出願第１０／７５０，１５５号（米国特許第７，１３９，８６４号）（特許文献１６）、２００４年８月１３日に出願された「Non-Volatile Memory and Method with Memory Planes Alignment 」という米国特許出願第１０／９１７，８８８号（米国公開特許出願第２００５／０１４１３１３号）（特許文献１７）、２００４年８月１３日に出願された「Non-Volatile Memory and Method with Non-Sequential Update Block Management」という米国特許出願第１０／９１７，８６７号（米国公開特許出願第２００５／０１４１３１２号）（特許文献１８）、２００４年８月１３日に出願された「Non-Volatile Memory and Method with Phased Program Failure Handling 」という米国特許出

願第１０／９１７，８８９号（米国公開特許出願第２００５／０１６６０８７号）（特許文献１９）、２００４年８月１３日に出願された「Non-Volatile Memory and Method with Control Data Management 」という米国特許出願第１０／９１７，７２５号（米国公開特許出願第２００５／０１４４３６５号）（特許文献２０）、２００４年１２月１６日に出願された「Scratch Pad Block 」という米国特許出願第１１／０１６，２８５号（米国公開特許出願第２００６／０１６１７２２号）（特許文献２１）、２００５年７月２７日に出願された「Non-Volatile Memory and Method with Multi-Stream Update Tracking」という米国特許出願第１１／１９２，２２０号（米国公開特許出願第２００６／０１５５９２１号）（特許文献２２）、２００５年７月２７日に出願された「Non-Volatile Memory and Method with Improved Indexing for Scratch Pad and Update Blocks 」という米国特許出願第１１／１９２，３８６号（米国公開特許出願第２００６／０１５５９２２号）（特許文献２３）、および２００５年７月２７日に出願された「Non-Volatile Memory and Method with Multi-Stream Updating 」という米国特許出願第１１／１９１，６８６号（米国公開特許出願第２００６／０１５５９２０号）（特許文献２４）で数多く説明されている。

【００５３】

非常に大きな消去ブロックを持つメモリアレイの動作を効率的に制御するには、書き込み操作のときに蓄積されるデータセクタの数をメモリのブロックの容量に一致させ、ブロックの境界に揃えることがひとつの課題となる。それには、ホストからの新規データの蓄積に使うメタブロックを最大ブロック数未満で構成し、メタブロック全体を埋め尽くすまでには至らない一定量のデータを蓄積するようにする方法がある。適応メタブロックの使用は、２００３年１２月３０日に出願された「Adaptive Metablocks 」という米国特許出願第１０／７４９，１８９号（米国公開特許出願第２００５／０１４４３５７号）（特許文献２５）で説明されている。データブロック間の境界とメタブロック間の物理的境界の整合は、２００４年５月７日に出願された「Data Boundary Management」という米国特許出願第１０／８４１，１１８号（米国公開特許出願第２００５／０１４４３６３号）（特許文献２６）と、２００４年１２月１６日に出願された「Data Run Programming」という米国特許出願第１１／０１６，２７１号（米国公開特許出願第２００５／０１４４３６７号）（特許文献２７）で説明されている。

【００５４】

メモリコントローラでは、ホストによって不揮発性メモリに蓄積されるＦＡＴテーブルのデータをメモリシステムの効率的作動に役立てることもできる。例えば、論理アドレスの解除によりホストによってデータが用済みと識別されたことを知るのに役立てる。メモリコントローラは、通常ならばホストがその論理アドレスに新規データを書き込むことから知るデータが用済みと識別されたことを、事前に知ることにより、そのような無効データを収容するブロックの消去スケジュールを組むことができる。これは、２００４年７月２１日に出願された「Method and Apparatus for Maintaining Data in Non-Volatile Memory Systems」という米国特許出願第１０／８９７，０４９号（特許文献２８）で説明されている。このほかの手法として、ホストがメモリに新規データを書き込むパターンを監視することにより、ある特定の書き込み操作が単一のファイルか否かを推定し、複数のファイルである場合にはファイル間の境界がどこにあるかを推定する。２００４年１２月２３日に出願された「FAT Analysis for Optimized Sequential Cluster Management」という米国特許出願第１１／０２２，３６９号（特許文献２９）では、このタイプの手法の使用が説明されている。

【００５５】

メモリシステムを効率よく操作するには、ホストによって各ファイルのデータに割り当てられる論理アドレスについて、コントローラができるだけ多くのことを知るのが望ましい。そうすればコントローラは、ファイルの境界が分からなければ多数のメタブロックに散在することになるデータファイルを、１つのメタブロックまたは１群のメタブロックに蓄積することができる。その結果、データの整理統合操作やガーベッジコレクション操作

の回数と複雑さが抑えられる。結果的にメモリシステムの性能は向上する。しかし、前述したように、ホスト／メモリインターフェイスが論理アドレス空間１６１（図９Ａ）を含む場合に、メモリコントローラがホストデータファイル構造について多くを知ることは困難である。

【００５６】

ダイレクトデータファイル操作

図７Ｂ、図８Ｂ、および図９Ｂに見られるホストと大量データ蓄積用メモリシステムとの各種インターフェイスは、論理アドレス空間の使用を解消する。代わりにホストは、一意なファイルＩＤ（またはその他の一意な参照符）とファイル内でのデータ単位（バイト等）によるオフセットアドレスとによって各ファイルのデータを論理的にアドレスする。これらのアドレスはメモリシステムコントローラへ直接提供され、メモリシステムコントローラは、各ホストファイルのデータの物理的な蓄積位置について独自のテーブルを管理する。これは前に相互参照した特許出願の主題にあたる操作である。このファイルインターフェイスは、図２～図６との関係で前述したものと同じメモリシステムで実装できる。図７Ｂ、図８Ｂ、および図９Ｂのファイル本位インターフェイスと図７Ａ、図８Ａ、および図９ＡのＬＢＡインターフェイスとの主な違いは、メモリシステムがホストシステムと通信しファイルデータを蓄積する方法にある。

【００５７】

図８Ｂのファイル本位インターフェイスと図８ＡのＬＢＡインターフェイスとを比較した場合、図８Ａの論理アドレス空間とホストによって保守されるＦＡＴテーブルは図８Ｂに存在しない。メモリシステムにとっては、ファイル番号とファイル内でのデータオフセットによってホスト生成データファイルが識別される。そして、メモリシステムは、ファイルをメモリセルアレイの物理ブロックに直接マップする。

【００５８】

ダイレクトデータファイル蓄積法で新規データファイルをメモリにプログラムする場合は、メモリセルの消去済みブロックにデータが書き込まれ、この書き込みはこのブロック内の最初の物理位置から始まって残りの位置を順次進んでいく。データは、ファイルにおけるこのデータのオフセット順序にかかわりなく、ホストからの受信順序に沿ってプログラムされる。プログラミングは、ファイルの全データがメモリへ書き込まれるまで続く。ファイル内のデータ量が１メモリブロックの容量を上回る場合は、最初のブロックが一杯になった時点で第２の消去済みブロックでプログラミングが継続する。第２のメモリブロックは第１のメモリブロックと同様にプログラムされ、最初の位置からファイルの全データが蓄積されるか、第２のブロックが一杯になるまで続く。ファイルのデータが残っている場合は、３番目以降のブロックにプログラムする。１ファイルのデータを蓄積する複数のブロックまたはメタブロックが物理的または論理的に隣り合っているとは限らない。説明を平易にするため、ここで用いる用語「ブロック」は別段の断りがない限り、システムでメタブロックを使用するか否かに応じて、ブロック消去単位か多重ブロック「メタブロック」を指すものとする。

【００５９】

図９Ｂを参照すると、ファイル１、２、および３の識別情報とファイル内でのデータオフセットはメモリコントローラへ直接引き渡される。この論理アドレス情報はメモリコントローラ機能１７３によってメモリ１６５のメタブロックおよびメタページの物理アドレスに翻訳される。ファイルデータは図９Ａの論理アドレス空間１６１にマップされない。

【００６０】

フラッシュ最適化ファイルシステムの原理

図７Ｃ、図８Ｃ、および図９Ｃは、図７Ｂ、図８Ｂ、および図９Ｂのダイレクトデータファイル手法に図７Ａ、図８Ａ、および図９Ａに見られるタイプのＬＢＡインターフェイスを組み合わせたオペレーティングシステムを、それぞれ異なる形式で示すものである。図７Ｃの「フラッシュ最適化ファイルシステム」の動作は図７Ｂの「ダイレクトファイル蓄積バックエンドシステム」と基本的に同じだが、ファイルデータは図７Ｃに見られるＬ

ＢＡインターフェイスの連続アドレス空間の中で論理ブロックにマップされ、図７Ｂに見られるＮＡＮＤフラッシュの物理メモリセルブロックにはマップされない。図７ＣのＬＢＡインターフェイスと「ＬＢＡ－物理バックエンドシステム」は、図７Ａのシステムと共通している。図７Ｃのシステムでは、ＬＢＡインターフェイスの前にダイレクトファイル－ブロックアドレス割り当てが行われるが、ＮＡＮＤフラッシュメモリの物理ブロックではなく、ＬＢＡインターフェイスの連続アドレス空間に含まれる論理ブロックアドレスを扱う。

【００６１】

図８Ｃには同じ発想が異なる形式で示されている。ホストによって生成されるデータファイルは、蓄積装置の論理アドレス空間に含まれる論理ブロックアドレスに割り当てられる。次に、論理アドレス空間の論理ブロックが従来どおりメモリコントローラによって物理蓄積媒体のブロックにマップされる。図８Ｃには、ホストとメモリシステムとでこれらの機能を分割した場合の２通りの区分が描かれている。図のホスト１から分かるように、第１の実施形態ではホストの中でファイルを論理ブロックアドレスに割り振る。この場合のメモリ１は従来のメモリカードやその他の装置であって、ホストのＬＢＡインターフェイスと接続するＬＢＡインターフェイスを備える。これとは別に、図８Ｃのホスト２は、データファイルの識別情報とファイル内でのデータオフセットをメモリシステムとやり取りする。これらのファイルを論理ブロックアドレスに割り振るダイレクトデータファイル機能は、メモリ２の中で実行される。

【００６２】

図８Ｃのメモリ２は、メモリカードやフラッシュドライブをはじめとする小型ポータブル装置の形が最も一般的で、蓄積装置の論理アドレス空間との外部接点を提供することによってＬＢＡインターフェイスを追加することもできる。さらなる代案として、ファイルを論理ブロックアドレスに割り振る機能はマイクロプロセッサを内蔵するマザーカードで実行することもできる。この場合のマザーカードはホスト２と取り外し可能な状態で接続し、メモリ１はマザーカードと取り外し可能な状態で接続する。

【００６３】

図９Ｃには、ファイルオブジェクトのデータを論理アドレス空間にマップする手法が異なる形式で示されている。機能１７３’は各ファイルのデータを、一意なファイル識別子とファイル内でのデータオフセットアドレスとからなる各論理アドレスと併せて、受信する。これらのファイルアドレスは、機能１７３’によって連続論理アドレス空間１６１の論理ブロックに含まれるアドレスに変換される。物理メモリでブロックとメタブロックのどちらの単位を使用するかに応じ、各論理ブロックのアドレス範囲はメモリアレイ１６５のブロックかメタブロックのデータ蓄積容量と同じになるように設定する。図９Ｃの機能１７３’は図９Ｂの機能１７３と基本的に同じだが、図９Ｃではアドレス空間１６１の論理ブロックにファイルをマップするのに対し、図９Ｂではファイルをメモリセルアレイ１６５に直接マップする。次に、論理アドレスブロックは図９Ｃの機能１６３によってメモリアレイ１６５に翻訳されるが、これは図９Ａと基本的に同じである。機能１６３は、前に述べた米国特許第７，１３９，８６４号（特許文献１６）や米国公開特許出願第２００５／０１４１３１３号（特許文献１７）、第２００５／０１４１３１２号（特許文献１８）、第２００５／０１６６０８７号（特許文献１９）、第２００５／０１４４３６５号（特許文献２０）、および第２００６／０１６１７２２号（特許文献２１）で説明されているもの等、従来のフラッシュメモリオペレーティングシステムであってよい。

【００６４】

図９Ｃでは、アドレス空間１６１の各論理ブロックのアドレスが２つ以上のファイルのデータにまたがることがあることに気づく。各ファイルのデータに２つ以上の論理ブロックにまたがってアドレスが割り当てられることもある。例えば、データファイル２および３には２つ以上の論理ブロックにまたがってアドレスが割り当てられている。論理ブロックは２つの異なるファイルのデータを収容することもあり、図９Ｃの論理ブロック２はその例である。しかし、好ましくは、ある１のファイルのデータと他の何らかのファイルの

データを収容する論理ブロックの数には１つ以上の制限を設ける。制限は状況によって異なる。具体例として、ファイルのデータをアドレス空間１６１のいくつかの論理ブロックに割り振る場合に、いずれか１つのファイルが別のファイルのデータと共有する論理ブロックを２ブロックまでにする。ファイルオブジェクトのデータに論理ブロックアドレスを割り当てるときにこの制約を順守するには、ファイルデータで部分的にしか満たされていない論理ブロックの数を制限する。

【００６５】

この制約により、例えば他のファイルのデータが用済みになった場合に必要となるデータ再配置量は少なくなる。通常、このような場合には別ファイルの用済みデータを含むブロックから別のブロックへファイルの有効データをコピーする。ある特定のファイルが別のファイルのデータと共有するブロックの数を制限することにより、このようなデータコピー操作の頻度は低くなる。その結果、メモリシステムの性能は向上する。

【００６６】

図１０を参照すると、論理ブロックと物理ブロックの両方でファイルデータの割り振りが示されている。例示するため、物理メモリセルブロックの例１９１は４つのページ１９５～１９９に分割されているが、実際のシステムでは通常ブロック当たりのページ数がこれよりも多くなる。各ページは複数セクタのデータを蓄積する。ブロックの中では通常、１９５～１９９の順序で一度に１ページずつデータをプログラムする。メタブロックを使用するメモリシステムの場合はブロック１９１がメタブロックとなり、ページ１９５～１９９はメタページになる。

【００６７】

物理ブロック１９１にマップされるのは、論理アドレス空間１６１の論理ブロック１９３である。論理ブロック１９３のデータ蓄積容量は物理ブロック１９１と同じに設定され、物理ブロック１９１と同数のページ２０１～２０４に分割され、各論理ページのデータ蓄積容量は物理ページ１９５～１９９のそれぞれと同じである。つまり、好ましくは、論理アドレス空間の粒度が物理メモリページまたはメタページのデータ蓄積容量に等しくなるようにする。論理ブロック１９３の中では、物理ブロック１９１にデータページを書き込むのと同じ順序で論理ページのアドレスをデータに割り当てる。論理ブロック１９３の第１のページ２０１の先頭におけるデータの書き込みは、物理ブロック１９１の第１のページ１９５の先頭で始まるようにする。

【００６８】

このような論理・物理機能の連携を保つには、ファイル－論理ブロック翻訳を行うホストが相手方にあたるメモリの物理的特性を知る必要がある。例えばメタブロックを使用するメモリシステムでは、以下のパラメータによってこれらの特性が決まる。

１．蓄積データのセクタ数による物理ページサイズ
２．ともにリンクされ各メタページを形成するページ数によるメタページサイズ
３．１メタブロック当たりのページ数
４．物理メタブロックの第１のページへマップされる最下位論理アドレス

【００６９】

ホストはこの情報をもとに論理アドレス空間１６１の論理ブロック構造を構成し、図１０に示すように作動することができる。ホストに埋め込まれたメモリ等、ある特定のホストで使用するメモリが１種類だけならば、ただひとつのホスト論理アドレス空間構成を管理するだけでよい。しかし、物理的特性がそれぞれ異なるポータブルメモリ装置が取り外し可能な状態でホスト装置に接続される場合のほうが一般的であり、実際にはホスト装置も様々である。そこで、接続先にあたる特定のポータブルメモリ装置の物理ブロック構成に論理ブロック構成を適合させるための機能をホストの中に用意する。それには、前述したメモリパラメータのデータをメモリ装置そのものに蓄積し、ホストで読み出されるようにする。通常、論理ブロックに対応する物理ブロックはメモリシステムのコントローラによって変更されるが、ホストがこれを知ることはなく、論理ブロックへのファイルデータのアドレス割り当てには影響しない。

【００７０】
図１１は、これらのパラメータデータを不揮発性蓄積領域２０９に収容するメモリ装置２０７を示すものであり、ホスト２１１は相互接続バス２１３を通じてこれにアクセスする。ホストがこれらのパラメータを読み出す方法は数多くある。一例として、ベンダー固有のコマンドを設定し、これをメモリ装置２０７の初期化中にホスト２１１からメモリ装置へ発行する。作動したメモリ装置２０７は、蓄積されたパラメータ値をホストへ返す。もうひとつの例として、メモリ装置２０７がホスト２１１からの既存標準コマンドに応じてホストへ返す既存フィールドの未使用部分に、これらのパラメータを盛り込むこともできる。ドライブ識別コマンドはそのようなコマンドの一例である。
【００７１】
代表的なフラッシュ最適化ファイルシステム
この節では、個々のファイルを連続論理アドレス空間の論理ブロックにマップする手法の代表的な実施例をさらに詳しく説明する。この手法のいくつかの態様は、基本的に同じ機能である図７Ｃの「フラッシュ最適化ファイルシステム」と、図８Ｃの「論理ブロックアドレスへのファイルの割り振り」と、図９Ｃの「ファイル／オフセット－論理アドレス変換」１７３’との関係で既に説明している。
【００７２】
論理ブロックアドレスへのファイルマッピングについてこの節で説明する内容の大半は、前に相互参照した特許出願で説明されている物理メモリセルブロックアドレスにファイルをマップする手法と同じ手法を利用する。主な違いは、相互参照した先行特許出願に説明されているように、データファイルをそのまま物理メモリブロックにマップしてＬＢＡインターフェイスを回避する代わりに、例えばホスト装置により、ＬＢＡインターフェイスにまたがってファイルマッピングを行うことにある。先行出願の物理メモリブロックマッピング手法を応用し、ＬＢＡアドレス空間の論理ブロックにデータファイルオブジェクトをマップすることも可能であり、ここではその例をいくつか説明する。
【００７３】
ファイルオブジェクトの論理的マッピングに関するここでの説明では、ＬＢＡインターフェイスのブロックにデータが「書き込まれる」、または「プログラムされる」と言う。当然ながら、これらの論理ブロックは物理メモリブロックと違って実際にはデータを蓄積しないため、これは特定の論理ブロックに対してデータのアドレスを指定することを意味する。同様に、データが割り振られていない論理ブロックのことを「消去済み」と言う。「消去済み」論理ブロックはデータのアドレスがない論理ブロックであり、完全に空いていてデータのアドレスを割り当てることができる。「部分的に消去済み」の論理ブロックもあり、これはその論理ブロックの一部分が空いていて、データのさらなるアドレスを受け付け可能であることを意味する。
【００７４】
フラッシュ最適化ファイルシステムの一般的動作
メモリに新規のデータファイルをプログラムするときには空いている論理ブロックにデータが書き込まれ、この書き込みはこのブロック内の最初の位置から始まって残りの位置を順次進んでいく。データは、ファイルにおけるこのデータのオフセット順序にかかわりなく、ホストからの受信順序に沿って論理ブロック内にプログラムされる。プログラミングは、ファイルの全データが書き込まれるまで続く。ファイル内のデータ量が１論理ブロックの容量を上回る場合は、最初のブロックが一杯になった時点で第２の空の（消去済み）ブロックでプログラミングを継続する。第２の論理ブロックは第１の論理ブロックと同様にプログラムされ、最初の位置からファイルの全データが割り振られるか、第２のブロックが一杯になるまで続く。ファイルのデータが残っている場合は、３番目以降のブロックにプログラムする。１ファイルのデータを蓄積する複数の論理ブロックまたはメタブロックが隣り合っているとは限らない。説明を平易にするため、ここで用いる用語、論理「ブロック」は別段の断りがない限り、システムでメタブロックを使用するか否かに応じて、メモリシステムにおける物理ブロックの最小消去単位と同じ容量を持つ論理ブロックか

、通常まとめて消去される多重ブロック物理メタブロックに対応する多重ブロック論理「メタブロック」を指すものとする。

【００７５】

図１２は、フラッシュ最適化ファイルシステムの全体的な働きを示す。個々の論理ブロックは３つの状態のいずれか１つにあるとみなすことができる。３つの状態とは、消去済みブロック６４１と、有効ファイルデータを蓄積し再生可能な容量がないブロック６４３と、ある程度の有効ファイルデータのほかに再生可能な容量もあるブロック６４５であって、ブロック６４５の場合は、そこにある未プログラム（消去済み）ページおよび／または用済み（無効）データから容量を再生できる。機能６４７によって消去済み論理ブロックへデータが書き込まれると、プログラムされたブロックに再生可能な容量が残るか否かに応じて当該ブロックはカテゴリ６４３か６４５のブロックになる。機能６４９に表示されているようにファイルが削除されると、ファイルデータを収容しているブロック６４３は再生可能な容量を持つブロック６４５に変わる。機能６５０で再生可能なブロックから別のブロックへデータをコピーした後には、ブロック６４５の未使用蓄積容量が機能６５１によって再生され、その結果、ブロックは消去済みブロック６４１の状態に戻り、新たにデータを書き込むことができる。

【００７６】

図１３Ａを参照すると、論理アドレス空間に対するデータファイルの書き込みが示されている。この例のデータファイル１８１は、垂直の実線間に広がる１ブロックまたはメタブロック１８３の蓄積容量より大きい。このため、データファイル１８１の部分１８４は第２のブロック１８５にも書き込まれる。これらの論理ブロックはアドレスが隣接するものとして図に示されているが、必ずしも隣接するとは限らない。ファイル１８１のデータはホストから受信されるにつれ書き込まれ、最終的にはファイルの全データが論理アドレス空間へ書き込まれる。図１３Ａの例で、データ１８１はファイルの最初のデータである。

【００７７】

メモリシステムで蓄積データを管理し追跡するには、可変サイズのデータグループを使用するのが好ましい。つまり、所定の順序につなげて完全なファイルを形成する複数のデータグループとしてファイルのデータを蓄積する。ホストからのデータストリームを書き込むときに、ファイルデータの論理オフセットアドレスやデータを割り振る論理アドレス空間に途切れが生じると、新たなデータグループが始まる。例えば、ファイルのデータによってあるひとつの論理ブロックが一杯になり、別のブロックへの書き込みが始まると、そのような途切れが論理アドレス空間に生じる。これを示す図１３Ａでは、第１のブロック１８３が第１のデータグループで満たされ、ファイルの残りの部分１８４は第２のデータグループとして第２のブロック１８５に蓄積されている。第１のデータグループは（Ｆ０，Ｄ０）で表すことができ、Ｆ０はデータファイルの先頭の論理オフセットであり、Ｄ０は論理ブロック１８３の中でファイルが始まる位置である。第２のデータグループは（Ｆ１，Ｄ１）で表され、Ｆ１は第２のブロック１８５の先頭に蓄積されるデータのファイルオフセットであり、Ｄ１は第２のブロックの先頭の論理アドレスである。

【００７８】

ホスト－メモリインターフェイスを通じて転送されるデータの量は、データのバイト数か、データのセクタ数か、他の何らかの粒度で表すことができる。ホストは通常、ファイルのデータをバイト粒度で定義するが、論理アドレスインターフェイスを通じて大容量メモリシステムと通信するときには、それぞれ５１２バイトのセクタか複数のセクタからなるクラスタにバイトをまとめる。これはメモリシステム動作を簡略化するために通常行われていることである。ここで説明するファイル本位のホスト－メモリインターフェイスでは別のデータ単位を使うこともできるが、通常は本来のホストファイルのバイト粒度が好ましい。つまり、データのオフセットや長さ等は、セクタやクラスタ等ではなく、データの最小分解単位であるバイト単位で表すのが好ましい。そうすれば、ここで説明する手法によりフラッシュメモリの蓄積容量をより有効利用できる。

【００７９】
図１３Ａのやり方で論理アドレス空間に書き込まれた新規ファイルは、ファイルインデックステーブル（ＦＩＴ）の中で（Ｆ０，Ｄ０）、（Ｆ１，Ｄ１）の順に並ぶ一連のデータグループインデックス項目によって表される。つまり、ホストシステムは、ある特定のファイルにアクセスする際にこのファイルのファイルＩＤやその他の識別情報を生成し、次にＦＩＴにアクセスし、このファイルを構成するデータグループを識別する。メモリシステムの操作を簡便にするため、各データグループの長さ＜ｌｅｎｇｔｈ＞をそれぞれの項目に盛り込むこともできる。
【００８０】
図１３Ａのファイルがホストによって開いた状態に保たれている間は、好ましくは書き込みポインタＰも維持し、そのファイルとしてホストから受信するさらなるデータの書き込みにあたって論理アドレスを指定できるようにする。ファイルの新規データは、当該ファイルにおける新規データの論理位置にかかわりなく、論理ブロックの中ではファイルの末尾に書き込まれる。メモリシステムでは一度に複数のファイル、例えば４つのファイルまたは５つのファイルを開いておくことができ、ファイルごとに書き込みポインタＰを維持する。ファイルごとの書き込みポインタは、論理ブロックにおけるファイルの位置を指し示すものである。既にシステムの開放ファイル数制限に達しているときに、ホストシステムで新たなファイルを開く場合は、開いているファイルのいずれか１つを先に閉じ、その後に新たなファイルを開く。
【００８１】
図１３Ｂは、既に図１３Ａで書き込まれ引き続き開いているファイルの末尾にホストがデータを付け加える様子を示している。ホストシステムによってファイルの末尾に追加されたデータ１８７が見られるが、これも第２のブロック１８５の中、このファイルのデータの末尾に、書き込まれる。追加されたデータはデータグループ（Ｆ１，Ｄ１）の一部になり、既存のデータグループ１８４と追加データ１８９との間にファイルや論理アドレスの途切れはないため、データグループ（Ｆ１，Ｄ１）のデータが増えたことになる。したがって、ＦＩＴの中では引き続きファイル全体が一連のインデックス項目（Ｆ０，Ｄ０）、（Ｆ１，Ｄ１）で表される。ポインタＰのアドレスも蓄積された追加データの末尾のアドレスに変更される。
【００８２】
図１３Ｃには、既に図１３Ａで書き込まれたファイルへデータブロック１９１を挿入する例が示されている。ホストはデータ１９１をファイルの中に挿入しているが、フラッシュ最適化ファイルシステムは、書き込み済みファイルデータの末尾にあたる位置１９３に挿入データを付け加える。開いているファイルの中にデータを挿入する場合でもファイルのデータを論理順に書き換える必要はないが、後ほどホストがファイルを閉じた後にバックグラウンドで書き換えを行うこともできる。挿入データは第２の論理ブロック１８５の中に完全に収まるため、新しい単独グループ（Ｆ１，Ｄ３）を形成する。しかし、この挿入により図１３Ａの以前のデータグループ（Ｆ０，Ｄ０）は２つのグループ、すなわち挿入箇所の前のグループ（Ｆ０，Ｄ０）と挿入箇所の後ろのグループ（Ｆ２，Ｄ１）とに分かれる。なぜなら、ファイルデータに途切れが生じると、例えば挿入箇所の先頭Ｆ１と挿入箇所の末端Ｆ２で途切れが生じると、新たなデータグループを形成する必要があるためである。グループ（Ｆ３，Ｄ２）は、論理アドレスＤ２が第２のブロック１８５の先頭になった結果である。グループ（Ｆ１，Ｄ３）とグループ（Ｆ３，Ｄ２）は同じ論理ブロックの中にあるが、そこに蓄積されたデータのファイルオフセットは途切れているために別扱いになる。ＦＩＴの中では挿入が行われたファイルが（Ｆ０，Ｄ０）、（Ｆ１，Ｄ３）、（Ｆ２，Ｄ１）、（Ｆ３，Ｄ２）の順に並んだデータグループインデックス項目によって表される。図１３Ａ、図１３Ｂ、および図１３Ｃの例では、既存ファイルや新規ファイルの新規データ書き込みにともない、論理ブロックアドレスによって表されるデータが用済みになっていないことに気づく。
【００８３】

図１３Ｃに示された既存ファイルへのデータ挿入とは別に、データが挿入されたときにホストによってファイルが別個のファイルとして書き換えられることもある。この別個のファイルはメモリシステムによって新規ファイルとして扱われる。古いファイルがホストによって削除されると、これに応じてシステムは古いファイルに割り当てられた論理アドレス空間を再生し、そのデータは用済みになる。

【００８４】

図１３Ｄに示すもうひとつの例では、図１３Ａのやり方で当初書き込まれたデータの一部分が更新される。データファイルの一部分１９５が更新される様子が見られる。更新によりファイル全体を書き換えるのではなく、書き込み済みのデータにファイルの更新部分１９７を付け加えている。そして、書き込み済みのデータ部分１９９は用済みになる。更新後のファイルはシステムＦＩＴの中で（Ｆ０，Ｄ０）、（Ｆ１，Ｄ３）、（Ｆ２，Ｄ１）、（Ｆ３，Ｄ２）の順序に並んだデータグループインデックス項目によって表される。図１３Ａで１つだったデータグループ（Ｆ０，Ｄ０）は図１３Ｄにおいても更新部分の前と、更新部分と、更新部分の後ろとに分割される。用済みデータで占められたアドレス空間１９９は再生するのが望ましいが、これはファイルデータの書き込みの一部として行うのではなく、後で行ったほうが好ましい。再生を後で行うことにより、通常ならばファイルの蓄積にともなうデータグループ数が少なくなる。

【００８５】

可変長データグループの取り扱いをさらに例示するため、同一ファイルが関係する一連の書き込み操作を図１４Ａ～図１４Ｅに順次示す。図１４Ａに見られるように、当初のファイルデータＷ１はまず、連続アドレス空間の２つの論理ブロックに書き込まれる。ファイルは２つのデータグループによって画定され、第１のグループは論理ブロックの先頭から始まり、第２のグループは論理ブロック境界の後ろになる。ここで図１４Ａのファイルは、一連のデータグループインデックス項目（Ｆ０，Ｄ０）、（Ｆ１，Ｄ１）によって表される。

【００８６】

図１４Ｂでは、図１４Ａで書き込んだファイルデータがホストによって更新される。更新ファイルデータＵ１は以前のグループ（Ｆ１，Ｄ１）のすぐ後ろに書き込まれ、前回の更新データは用済みになる。図１４Ａの以前のグループ（Ｆ０，Ｄ０）は図１４Ｂの修正されたグループ（Ｆ０，Ｄ０）まで短縮し、以前のグループ（Ｆ１，Ｄ１）はグループ（Ｆ４，Ｄ２）まで短縮する。更新データは論理ブロックの境界に重なっているために２つのグループ（Ｆ２，Ｄ３）および（Ｆ３，Ｄ４）に書き込まれる。そのデータの一部は第３の論理ブロックに蓄積される。このときファイルは、一連のデータグループインデックス項目（Ｆ０，Ｄ０）、（Ｆ２，Ｄ３）、（Ｆ３，Ｄ４）、（Ｆ４，Ｄ２）によって表される。

【００８７】

図１４Ｃでは図１４Ｂのファイルがホストによってさらに修正され、新規ファイルデータＩ１が挿入される。挿入データは論理ブロックの境界に重なっているため、新規データＩ１は、図１４Ｃの新規グループ（Ｆ５，Ｄ６）および（Ｆ６，Ｄ７）として、論理ブロックにおいて図１４Ｂの以前のグループ（Ｆ４，Ｄ２）のすぐ後ろに書き込まれる。第４の論理ブロックが使われている。新規データＩ１の挿入のため、図１４Ｂの以前のグループ（Ｆ０，Ｄ０）は図１４Ｃで短縮グループ（Ｆ０，Ｄ０）および（Ｆ７，Ｄ５）に分割される。このときファイルは一連のデータグループインデックス項目（Ｆ０，Ｄ０）、（Ｆ５，Ｄ６）、（Ｆ６，Ｄ７）、（Ｆ７，Ｄ５）、（Ｆ８，Ｄ３）、（Ｆ９，Ｄ４）、（Ｆ１０，Ｄ２）によって表される。

【００８８】

図１４Ｃのデータファイルに対するさらなる修正を示す図１４Ｄでは、ファイルの末尾に新規データＷ２を付け加えている。新規データＷ２は、図１４Ｄの新規グループ（Ｆ１１，Ｄ８）として図１４Ｃの以前のグループ（Ｆ１０，Ｄ２）のすぐ後ろに書き込まれる。このときファイルは、一連のデータグループインデックス項目（Ｆ０，Ｄ０）、（Ｆ５

，Ｄ６）、（Ｆ６，Ｄ７）、（Ｆ７，Ｄ５）、（Ｆ８，Ｄ３）、（Ｆ９，Ｄ４）、（Ｆ１０，Ｄ２）、（Ｆ１１，Ｄ８）によって表される。

【００８９】

開いたファイルに対する第２の更新を示す図１４Ｅでは、図１４Ｄのファイルに更新ファイルデータＵ２を書き込んでいる。図１４Ｅでは更新データＵ２が図１４Ｄの以前のグループ（Ｆ１１，Ｄ８）のすぐ後ろに書き込まれ、前の更新データは用済みになる。図１４Ｅでは図１４Ｄの以前のグループ（Ｆ９，Ｄ４）が修正されたグループ（Ｆ９，Ｄ４）まで短縮し、以前のグループ（Ｆ１０，Ｄ２）は完全に用済みとなり、以前のグループ（Ｆ１１，Ｄ８）は短縮して新たなグループ（Ｆ１４，Ｄ９）を形成する。更新データは図１４Ｅの新規グループ（Ｆ１２，Ｄ１０）および（Ｆ１３，Ｄ１１）に書き込まれ、論理ブロック境界に重なる。ここで第５の論理ブロックがファイルに必要になる。このときファイルは一連のデータグループインデックス項目（Ｆ０，Ｄ０）、（Ｆ５，Ｄ６）、（Ｆ６，Ｄ７）、（Ｆ７，Ｄ５）、（Ｆ８，Ｄ３）、（Ｆ９，Ｄ４）、（Ｆ１２，Ｄ１０）、（Ｆ１３，Ｄ１１）、（Ｆ１４，Ｄ９）によって表される。

【００９０】

各ファイルのデータのオフセットは、これまで説明したファイルの作成や修正の後に、好ましくは連続する正しい論理順序に保たれる。例えばファイルの中にデータを挿入する操作の一部としてホストによって提供される挿入データのオフセットは、挿入箇所の直前のオフセットから連続し、ファイル内の既存データは、挿入後に挿入データの分だけ増加する。既存ファイルの更新にあたってはほとんどの場合、既存ファイルの特定のアドレス範囲内にあるデータがほぼ同量の更新データで差し替えられるため、通常はファイルのそれ以外のデータのオフセットを差し替える必要はない。

【００９１】

このようにして蓄積されるデータの粒度または分解能は、ホストのものと同じに保つことができる。例えばホストアプリケーションが１バイト粒度でファイルデータを書き込むなら、そのデータは論理ブロックの中でも１バイト粒度で表される。そして、データグループの中でのデータの位置と量はバイト数単位になる。つまり、ホストアプリケーションファイルの中で個別にアドレスされるオフセット単位のデータは、フラッシュメモリに蓄積されたときにもファイルの中で個別にアドレスされる。論理ブロックの中にある同一ファイルのデータグループ境界は、ＦＩＴの中で最も近いバイトオフセット単位かその他のホストオフセット単位まで指定される。同様に、論理ブロックの中にある異なるファイルのデータグループ境界はホストオフセット単位で指定される。

【００９２】

ここで使用する用語「セクタ」は、大きなブロックメモリでＥＣＣが関わる蓄積データの単位を意味する。つまり、セクタは、メモリシステムのコントローラによって誤り訂正符号が生成されデータとともに蓄積される場合に、フラッシュメモリを行き来するデータの最小転送単位である。物理メモリに言及する場合の「ページ」は、ブロック内の１単位のメモリセルを意味する。ページは最小のプログラミング単位である。論理ブロック内の論理「ページ」は、物理ページと同量のデータを収容する。用語「メタページ」は、完全並列メタブロックのページを意味する。メタページは最大のプログラミング単位である。

【００９３】

図１４Ｂおよび図１４Ｅでは、更新コマンドの結果としてファイルによって占められる論理アドレス空間がファイルのデータ量を上回っていることが分かる。これは、更新によって差し替えられたデータの論理アドレスが残っているためである。そこで、用済みになった無効データを取り除いてファイルのデータをより小さい論理アドレス空間に整理統合（ガーベッジコレクション）することが強く望まれる。それによって、ほかのデータに使える論理アドレス空間が増える。

【００９４】

図１４Ｂおよび図１４Ｅのファイルデータ更新に加え、図１４Ｃのデータ挿入によってファイルデータのアドレス順序がずれることにも気づく。つまり、更新や挿入が行われる

とファイルの末尾に追加されるが、ほとんどの場合、更新や挿入の位置はファイルの中である。図１４Ｂ、図１４Ｃ、および図１４Ｅの例がこれに該当する。そこで、ファイル内のオフセット順序に合わせて論理アドレス空間にまたがるファイルデータを並べ替えるのが望ましい。こうすれば、ページやブロックを順次読み出すときにファイルデータがオフセット順に提供され、蓄積データの読み出し速度が向上する。ファイルの断片化も極力抑えることができる。しかし、読み出し効率を上げるファイルデータの並べ替えでも、メモリシステムの性能にとっては、ファイルデータの整理統合ほどには重要ではなく、ファイルデータの整理統合なら、場合によっては別のデータのアドレスのための１つ以上の論理ブロックを開放することができる。ファイルデータの並べ替えを単独で行うとオーバーヘッドの増加に見合うほどのメリットが得られないため、通常は並べ替えを単独で行うことはないが、ガーベッジコレクション操作の一部として行えばオーバーヘッドの増加は皆無かごく僅かですむ。

【００９５】

図１４Ｅのファイルには、２つのデータ更新Ｕ１およびＵ２が行われたことによって用済みのデータグループ（灰色の部分）がある。このファイルに割り振られた論理アドレス空間の量がファイルのサイズを大きく上回っていることは、図１４Ｅから明らかである。したがって、ガーベッジコレクションが妥当である。図１５は、図１４Ｅのデータファイルでガーベッジコレクションを行った結果を示すものである。ガーベッジコレクションの前には５論理ブロック近くのアドレス空間（図１４Ｅ）を占めていたファイルが、ガーベッジコレクションの後には３ブロック強以内（図１５）で収まっている。ガーベッジコレクション操作の一部として、当初の書き込み論理ブロックから別の消去済み論理ブロックへデータをコピーした後に、当初のブロックを消去する。ファイル全体でガーベッジコレクションを行う場合は、ファイル内でのデータの論理オフセット順序と同じ論理順序でデータを別のブロックにコピーできる。例えば更新Ｕ１およびＵ２と挿入Ｉ１はガーベッジコレクション（図１５）の後に、ホストファイル内での順序と同じ順序で蓄積される。

【００９６】

また、ファイル単位のガーベッジコレクションでは通常、別の新しいデータグループが整理統合されているファイルの中で形成される。図１５の場合は、一連の新規データグループインデックス項目（Ｆ０，Ｄ１２）、（Ｆ１，Ｄ１３）、（Ｆ２，Ｄ１４）、（Ｆ３，Ｄ１５）によってファイルが表される。データグループの数は、図１４Ｅに見られるファイルの状態より遥かに少ない。ファイルデータをコピーした各ブロックにつきデータグループが１つずつある。ＦＩＴは、ファイルを形成する新たなデータグループを反映するため、ガーベッジコレクション操作の一部として更新される。

【００９７】

図１４Ｅの状態でファイルデータを保持するブロックの再生は、同じファイルのデータを蓄積する複数のブロックではなくブロックごとに個別に行われる。例えばある一時点で再生操作の候補となるアドレス空間のブロックのうち、有効データ量が最も少ないブロックが図１４Ｅの第２のブロック００２なら、そこにある１データグループを別の消去済みブロックにコピーする。新しいブロックには１つのデータグループ（Ｆ８，Ｄ１６）が入り、ブロックの残りの部分は新規データを書き込める消去済み容量である。この消去済み容量は、図１４Ｅでデータを蓄積していたブロックから再生されたものである。このときファイルは、このファイルを構成するデータグループの一連のインデックス項目（Ｆ０，Ｄ０）、（Ｆ５，Ｄ６）、（Ｆ６，Ｄ７）、（Ｆ７，Ｄ５）、（Ｆ８，Ｄ１６）、（Ｆ９，Ｄ４）、（Ｆ１２，Ｄ１０）、（Ｆ１３，Ｄ１１）、（Ｆ１４，Ｄ９）によって表される。図１４Ｅに見られる他のブロックは再生操作の条件をそれぞれが満たすまで変化しない。

【００９８】

ファイルブロック管理

論理ブロックのタイプは、そこに蓄積されたファイルデータの構造に基づいて認識される。連続アドレス空間の中でアドレスを持つ各ファイルは、数ある状態のいずれか１つを

とり、各ファイルの状態は、ファイルデータを蓄積するブロックの数とタイプによって決まる。ファイルデータの書き込みにあたっては、好ましくはファイルの現在状態とある状態から別の状態に至る許容遷移とを規制することにより、ある特定のファイルのデータのほかに１つ以上の別のファイルのデータも含むブロックの数を制限する。これにより論理ブロックの有効利用を促進し、新規データやコピーデータの受け付けにあたって十分な消去済みブロックを確保するために後ほど行われる再生操作の頻度を抑える。

【００９９】

この例で認識され、ファイルデータを収容する、論理ブロックの主なタイプは次のとおりである。

・「ファイルブロック」は完全にプログラムされ、１ファイルの有効データを意味する。ある程度の用済みデータのアドレスを含むこともある。

・「プログラムブロック」は部分的にプログラムされ、単独ファイルのみの有効データを意味する。このブロックにはある程度の消去済み容量が残っている。ある程度の用済みデータのアドレスを含むこともある。

・「共通ブロック」は部分的にプログラムされ、２ファイル以上の有効データを意味する。ある程度の消去済み容量が残っている。ある程度の用済みデータのアドレスを含むこともある。

・「フル共通ブロック」は完全にプログラムされ、２ファイル以上の有効データを意味する。ある程度の用済みデータを意味することもある。

【０１００】

ブロックタイプにはこのほかに「消去済みブロック」があり、このブロックにはデータアドレスがなく、その全容量をデータの受け付けに使用できる。ＬＢＡインターフェイスの論理アドレス空間がデータアドレスで満たされているかほぼ満たされている場合は通常、使用中の論理ブロックで使われていない容量を絶えず再生しながら所定の最小消去済みブロック数のプールを維持する。

【０１０１】

「フラクタルブロック」は、プログラムブロックと、共通ブロックと、フル共通ブロックとを指す総称である。ファイルのフラクタルブロックの中には、ある１つのファイルの有効データのほかに、プログラムされていない蓄積容量か、別のファイルの有効データか、その両方がある。ここで説明する手法の第１の目的は、ファイルデータを受け付けるよう指定されたアクティブブロックのタイプを管理しながら、アドレス空間の中でフラクタルブロックの数を最小限に抑えることにある。これにより、所定の最小消去済み論理ブロック数を維持するために論理アドレス空間で行われるガーベッジコレクションやデータの整理統合（ブロック再生操作）の回数を抑える。プログラム済みブロックで未使用容量の断片を再生するときにデータの内部コピーにかかる時間が短くなるため、メモリにデータを書き込む速度も向上する。

【０１０２】

ここでは各種ブロックを総称するためにさらに別の用語を使用する。

・「パーシャルブロック」はある程度の未プログラム容量と、１ファイル以上の有効データのアドレスを含み、ある程度の用済みデータを意味することがある。例えば、プログラムブロックや共通ブロックがパーシャルブロックにあたる。

・「用済みブロック」は、ある程度の用済みデータのアドレスを含むファイルブロックまたはフル共通ブロックである。用済みブロックには消去済み容量がなく、有効データと用済みデータの両方を意味する。

・「無効ブロック」の中に有効データはない。無効ブロックは少なくともある程度の用済みデータのアドレスを含むほか、消去済み容量を含むこともあるが、有効データは意味しない。

【０１０３】

図１６Ａ～図１６Ｄは、前に定義したタイプの論理ブロックの使用例を示す。図１６Ａでは、ファイルＡのデータによってブロック６６１および６６３が満たされ、第３のブロ

ック６６５は部分的に満たされている。データは左から右にかけてこの例の各ブロックに書き込まれ、まずはブロック６６１を満たし、次にブロック６６３を満たし、その後ブロック６６５の一部分に書き込まれる。ブロック６６５の残りの部分はプログラムされていない消去済み容量であって、さらなるデータを蓄積できる。前述した定義によるとブロック６６１および６６３はファイルブロックであり、ブロック６６５はプログラムブロックである。ブロック６６５には、プログラムポインタＰのところから新規のデータが書き込まれる。ポインタＰはブロックへのデータの書き込みにともない左から右へ進み、常にブロックの中で次に使用できる蓄積位置を指し示す。プログラムされていない消去済み容量を持つ各ブロックでは、それが現在アクティブであろうとなかろうと、そのようなポインタを管理し、ブロックに書き込まれる他のデータの論理アドレスを常に把握できるようにする。

【０１０４】

図１６Ｂの例にあるブロック６６９は、現在のファイルＡのデータとある程度の未プログラム容量のほかに、別のファイルＢのデータを収容しているため、共通ブロックである。新規のデータは、ファイルＡの末尾、プログラムポインタＰが表示されているところから、ブロック６６９に書き込まれる。ブロック６６９はファイルＡにとってのアクティブブロックである。ファイルＢにとってのアクティブブロックになることもあり、その場合は、ファイルＡまたはＢのいずれかの追加データをプログラムポインタＰのところに書き込むことができる。あるいは、これとは別のブロック（図示せず）がファイルＢにとってのアクティブブロックになることもある。

【０１０５】

消去済みブロックではなく、既に別のファイルのデータを収容しているパーシャルブロックの消去済み容量にファイルデータを直接書き込めば、このような未プログラム容量を上手に利用することができる。これは特に、ブロック全体の容量に満たない既知量のファイルデータを書き込む場合に有益である。それには既存のパーシャルブロックを探査し、既知量の書き込みデータにフィットする消去済み容量を見つける。データのページ（メタブロックを使用する場合はメタページ）数を、パーシャルブロック内の未プログラム容量のページ数に比較する。このやり方でプログラムブロックの未使用消去済み領域がプログラムされると、そのプログラムブロックは共通ブロックに変わる。

【０１０６】

図１６Ｃでは、ファイルブロック６６１と、ブロック６７１の一部分と、ブロック６７３の一部分とにファイルＡが蓄積されている。２つのファイルＡおよびＢのデータで満たされているブロック６７１はフル共通ブロックである。図１６Ａのブロック６６５と同様、ブロック６７３はプログラムブロックである。ファイルにとってのアクティブブロックはブロック６７３であり、ポインタＰは、ブロック６７３の中で未使用容量の位置を指し示し、追加のデータはこの位置から書き込まれることになる。

【０１０７】

図１６Ｄの例では、フル共通ブロック６７１の一部分と共通ブロック６７５にファイルＡが書き込まれている。ブロック６７５の中には第３のファイルＣのデータがある。ポインタＰはアクティブブロック６７５で未使用部分の先頭を指し示し、追加のデータはここに書き込まれることになる。

【０１０８】

図１６Ａ～図１６Ｄの例では、数通りのブロックを例示するために複数のブロックに蓄積されたファイルＡのデータが示されているが、ファイルは多くの場合、これよりも少ないブロックに、ことによると１ブロックに、蓄積できるほど小さい。ここで説明する手法はそのような小さなファイルにも適用できる。大きなファイルが４ブロック以上にわたってページを占めることもある。

【０１０９】

論理ブロック６６５、６６９、６７１、６７３、および６７５がフラクタルブロックであることに気づく。いずれか１ファイルのデータを含むフラクタルブロックがあると、そ

の中にある未使用容量を再生する必要が高まってシステム性能に悪影響がおよぶため、そのようなフラクタルブロックの数は最小限に抑えるのが望ましい。パーシャル論理ブロック６６５、６６９、６７３、および６７５には未使用の消去済み容量があるが、まだ書き込まれていないファイルデータの量が分かっていて、その既知量がこれらのブロックのうちの１つの未使用容量に一致しない限り、ホストからこの領域に新規データをそのまま書き込むのは効率的でない。ほとんどの場合、ホストから到来するファイルデータの量は分からないため、これらの断片的容量が容易に埋まることはない。そこでメモリ容量を有効利用するには、再生操作のときに別のブロックから未使用領域へデータを移す必要がある。ブロック６６９、６７１、および６７５には２ファイル以上のデータが入っているため、それらのファイルのいずれか１つが削除されたり、共通ブロックに蓄積されているデータが用済みになったりする場合はデータ再生を行い、用済みデータのアドレスで占められたブロックの容量を再生することになる。

【０１１０】

そこで、時間のかかるデータ再生操作の回数を減らすには、ある特定のファイルのデータを一度に蓄積するフラクタルブロックの数を１、２等の数にする。許容フラクタルブロック数の決定にあたっては、フラクタルブロックを使用する場合の利点とそれらを使用することの好ましくない影響とでバランスをとる。ここで説明する具体例では、１ファイルのデータを蓄積できるフラクタルブロックを２ブロックまでとする。これに応じて、ファイルデータの蓄積にあたって新規アクティブブロックを指定するプロセスには制約がかかる。各ファイルには１組の許容ファイル状態のいずれか１つを割り当てるが、これはファイルのデータを蓄積するブロックのタイプによって決まる。既存のブロックが一杯になる等、ある特定のファイルのデータを受け付けるための新規のアクティブブロックを指定する必要がある場合、アクティブブロックに指定されるブロックのタイプはファイルの状態に左右され、多くの場合はそれ以外の要因にも左右される。

【０１１１】

図１７の表には、ある特定の実施例でファイルデータを収容するフラクタルブロックの組み合わせによる７通りの許容ファイル状態００～２０の定義が記載されている。どの許容ファイル状態でもデータを蓄積できるのは２フラクタルブロックまでである。ファイルデータを蓄積するファイルブロックの数に制限はない。ファイルのアクティブブロックとして使用するブロックの選択にあたっては、ファイルの状態をプロパティとして使用する。再生ブロックは一時的なものであって、その中にあるファイルデータはファイル状態の判断に寄与しないため、再生ブロックとして選ばれたブロックはフラクタルブロックとして扱わない。装置内に存在する全ファイルの状態を監視し、ファイルデータインデックス情報と併せてＦＩＴに記録する。記録されたファイルの状態は状態遷移のたびに更新される。

【０１１２】

ファイル状態遷移は、それらがプログラムデータに関係する状態遷移か、用済みデータに関係する状態遷移か、再生ブロックの選択に関係する状態遷移かによって、３つの部類に細分される。図１８の状態図には、保留のデータプログラミング操作か完了したデータプログラミング操作に基づく許容ファイル状態遷移が描かれている。図１７の表に記載されたファイル状態識別番号を囲む円により、７通りのファイル状態が示されている。

図１８の状態遷移ラベルの意味は次のとおりである。

Ａ－ファイルのアクティブブロックとして消去済みブロックが割り当てられる。

Ｂ－パーシャルブロックが満杯になった。

Ｃ－ファイルのアクティブブロックとしてパーシャルブロックが割り当てられる。

Ｄ－このファイルのパーシャルブロックが別のファイルのアクティブブロックとして割り当てられる。

Ｅ－アクティブブロックとして割り当てられた消去済みブロックに対しデータ遷移が行われる。

Ｆ－アクティブブロックとして割り当てられたパーシャルブロックに対しデータ遷移が

行われる。

【０１１３】

ほとんどの状態遷移は、ブロックが割り当てられるかブロックが満杯になるときに自動的に生起する。しかし、定義した状態遷移によっては、ある１つのブロックから別のブロックにかけてのデータ再配置をともなうものもある。データの再配置は１回の連続操作で行われ、このデータ再配置が完了した場合に限り状態遷移が生起したとみなされる。このような遷移を「データ遷移」という。図１９の表は、図１８の状態図を参照しながら許容状態遷移の詳細を伝えるものである。

【０１１４】

書き込みデータの長さが分かっている場合は、アクティブブロックとしてパーシャルブロックを割り当てることができる。この場合は、装置内のパーシャルブロック群から「最適」なパーシャルブロックを選ぶ。「最適」とは、既知量の書き込みデータが有効利用できる消去済み容量を有するパーシャルブロックのことである。「最適」パーシャルブロックが存在しなければ、「最大」パーシャルブロックを代わりに選ぶ場合もある。これは、使用可能な未使用容量が最も大きいパーシャルブロックである。

【０１１５】

図２０は状態図であり、用済みデータに基づくファイル状態遷移を示すものである。ファイルの状態遷移は、当該ファイルのデータを収容するフラクタルブロックの中で１ファイルの全データが用済みになるときに起こる。用済みになったデータのファイルは当該ファイルとは限らない。データは４つのイベントのいずれか１つによって用済みとなる。

１．ファイルがホストによって削除される。

２．ファイル内のデータがホストによって削除される。

３．書き込み済みのファイルデータがホストによって更新される。

４．再生操作中にファイルデータが再配置される。

図２０の状態遷移ラベルの意味は次のとおりである。

Ｇ－パーシャルブロックの中でこのファイルの全データが用済みになった。

Ｈ－フル共通ブロックの中でこのファイルの全データか他の全てのファイルの全データが用済みになった。

Ｉ－パーシャルブロックの中で他の全てのファイルの全データが用済みになった。

【０１１６】

図２１の表は、図２０に描かれた用済みデータに基づくファイル状態遷移の詳細を伝えるものである。これらの経緯でデータが用済みになると、用済みデータが入っているブロックのタイプが変化し、それにともないファイル状態も変化する。

【０１１７】

再生ブロックとして選ばれたブロックは、そこにデータを蓄積するファイルにとってフラクタルブロックでなくなる。その結果、図２２の状態図に示すファイル状態遷移が起こる。図２２の状態遷移ラベルの意味は次のとおりである。

Ｊ－パーシャルブロックが再生ブロックとして選ばれる。

Ｋ－フル共通ブロックが再生ブロックとして選ばれる。

【０１１８】

再生ブロックの選択に基づくファイル状態遷移の詳細を図２３の表に示す。

連続論理アドレス空間の論理ブロックにファイルデータを揃えるには２通りの方法がある。前に相互参照した特許出願で説明されている物理メモリセルブロック上で作動するダイレクトデータファイルシステムの場合は、好ましくは新規ファイルの先頭を消去済みメモリセルブロックの先頭に揃える。図２４に示すように、ダイレクトデータファイルシステムで論理ブロックを扱う場合にもこれは可能である。３つのファイルＡ、Ｂ、およびＣが論理ブロック１～７に蓄積された状態で描かれている。図２４から分かるように、これらのファイルのいずれか１つで全てのデータが書き込まれると、ファイルの最終部分はパーシャルブロックのごく一部分を占めることになる。

【０１１９】

図２５の表には、ファイルデータを蓄積するアクティブブロックとして割り当てる論理ブロックのタイプを判断するための基準が記載されている。記載されているように、これは現行のファイル状態（図１７の表で規定）とプログラムするデータの全般状況に左右される。この基準をもとに割り当てケースのいずれか１つが選ばれても、ブロックの有無に応じて、図２５の右列に記載された一定の候補からブロックタイプを絞りこまなければならない。例えば割り当てケースＢの場合に、長さが既知のデータを受け付けるブロックとして第１候補に登るのはパーシャルブロックである。そこで、まずはこの既知量のデータを蓄積するにあたって丁度いい使用可能（消去済み）な容量を持つパーシャルブロックを探す。これが見つからなければ、未プログラム領域が最も大きいパーシャルブロックの有無を確認する。これがなければ、第３の候補として完全に未使用の（消去済み）ブロックがデータを受け付けるブロックとして指定され、この例の割り当てケースＢの場合、既知量の書き込みデータはブロック全体を満杯にする量には満たないため、このブロックはパーシャルブロックになる。

【０１２０】

図２４のファイルＡ、Ｂ、またはＣ等で、最初に書き込まれたときと同じ状態を保つファイルが削除される場合は、それとは無関係のファイルのデータで再配置を行う必要はない。しかし、再生操作によってパーシャルブロックにある１ファイルのデータを別ファイルのデータと整理統合した場合に当該ファイルが削除されると、ただひとつのブロックで別ファイルのデータの再配置が必要になる。例えば、ブロック２のファイルＡのデータがブロック７にあるファイルＣのデータと整理統合された後に、ファイルＡかファイルＣが削除されると、ただひとつのブロックで、すなわちブロック７で、データの再配置が必要になる。

【０１２１】

ブロック再生は、ファイルデータの書き込みプロセスと交互に行われるプロセスであって、再生の対象となるブロックから別の場所に有効データを移し、このブロック（未使用と指定されたもの全容量）を消去することによってブロック内の未使用容量を再生する。ブロックが再生の対象として選ばれる理由には次の２つがある。

１．ファイルの削除か更新によってブロック内のデータが用済みになる、または

２．ブロックが未プログラム容量を持つパーシャルブロックである。

新規ファイルデータの書き込み速度を一定に保つため、再生プロセスにあてる時間はできるだけ一定であることが好ましい。ファイル書き込みプロセスで発生し再生プロセスで処理しなければならないパーシャルブロックの数は予測できないため、これを果たすのは困難である。

【０１２２】

図２４に見られるファイル－ブロックマッピング方式には、最新の書き込みファイルデータを収容するパーシャルブロックが、再生操作のコピー元ブロックかコピー先ブロックに選択されるまで存続するという利点がある。このため、ファイルが先に削除され、共通ブロックの中にある当該ファイルのデータや無関係ファイルのデータで再配置を行わずにすむ見込みが高くなる。なぜなら、ファイルのデータはそのファイル専用のブロックに収容されるためである。再生操作が不要ならばデータをコピーする時間も不要となり、メモリシステム動作の効率が上がる。

【０１２３】

図２４のマッピング方式の欠点として、通常は書き込みが行われる各ファイルにつきパーシャルブロックが１つずつ作成されるため、使用可能な未プログラム（消去済み）容量の再生にあたっては多数のパーシャルブロックでデータの整理統合が通常必要になる。さらに、図２４のマッピング方式だと、物理メモリを管理するメモリコントローラが新たな消去済みブロック容量を再生するために部分的に書き込まれたブロックのデータを自動的に整理統合する場合に、時間のかかるデータコピーが大量に発生することになる。このため、図２６の代替マッピング方式の実施が望ましい場合もある。この方式の第１の特徴として、新規ファイルの先頭のデータはパーシャルブロック内に既に存在する無関係ファイ

ルのデータと隣接する。ほとんどの場合、１ファイルの全データが書き込まれると、その最終書き込みデータはパーシャルブロックのごく一部分を占めることになるが、これは一時的である。パーシャルブロックの未プログラム領域は、完了したファイルに隣接して書き込まれる新規ファイルデータによって直ちに埋まる。

【０１２４】

図２７の表には、データの書き込みにあたって図２６の方式を実施した場合のアクティブブロック割り当てが記載されている。図２７の表では、新規ファイルの場合と既存ファイルの場合を分けて説明するため、図２５の表に記載された割り当てケースＡが割り当てケースＡ１およびＡ２に差し替えられている。

【０１２５】

図２６では通常、ファイルの先頭とファイルの末尾のデータでブロックが無関係ファイルのデータと共有され、１ファイルが削除されると２つのブロックでデータの再配置が必要になる。例えばファイルＢが削除されると、ブロック２から別の場所にファイルＡのデータを移し、ブロック３から別の場所にファイルＣのデータを移すことになる。次にブロック２および３を消去し、消去済み（未使用）ブロックのプールに加えることができ、プールのブロックには後ほど別のデータを書き込むことができる。

【０１２６】

図２６のファイル－ブロックマッピング方式には、部分的にプログラムされたブロックをためこまずにすむという利点がある。書き込みが行われる新規ファイルのデータの先頭はパーシャルブロックの中に既に存在する無関係ファイルのデータと隣接するため、最新の書き込みファイルデータを含むパーシャルブロックが長時間にわたって存続することはなく、装置に存在するパーシャルブロックの数はごく僅かになる。その結果、再生操作でパーシャルブロックのデータを整理統合する機会が少なくなるほか、一定の再生レートを設定し、新規ファイルデータの書き込みで一定の速度を維持することが可能となる。

【０１２７】

しかし、図２６のマッピング方式には、ファイルが削除されるときに無関係ファイルのデータで再配置が必要になる見込みが高くなるほか、毎回の再配置で移動するデータ量が増えるという欠点がある。図２６の方式には、再生操作でパーシャルブロックのデータを整理統合する頻度を抑えるという図２４の方式を凌ぐ利点があっても、その利点は、ファイルが削除されるときの再配置データ増大という欠点によって帳消しになる。

【０１２８】

ブロック容量の再生

前述したように、新規データ蓄積のためのブロックにおける未使用容量再生はブロック管理の一部として行われる。これは、メモリシステムに蓄積されたデータ量がメモリシステムの容量を遥かに下回る場合は特に問題にならないが、メモリシステムはデータで一杯の状態を想定して設計されることが好ましい。これは、未使用容量を再生しながら、用済みデータだけを含むブロックと、有効データのほかにある程度の用済みデータおよび／または未書き込みページを持つ他のブロックを処理することを意味する。目標は、メモリシステムの蓄積容量をできるだけ余すところなく利用すると同時に、システム性能への悪影響を最小限に抑えることにある。

【０１２９】

再生操作の対象として指定されたブロック（コピー元ブロック）に有効データがある場合は、その有効データを蓄積するにあたって十分な未使用（消去済み）容量を持つ１つ以上のブロック（コピー先ブロック）に有効データをコピーする。このコピー先ブロックは、前述したブロック管理手法に従って選択する。コピー元ブロックに蓄積された各ファイルデータは別のブロックへコピーされ、このコピー先ブロックは、前述したようにファイルの状態とその他の要因に基づいて選択する。再生操作の一部として各種ファイル間で行われるデータコピーの例を図２８Ａ～図２８Ｄに示す。

【０１３０】

図２８Ａには、２つのパーシャルブロック６８１および６８３に対する再生操作が例示

されている。ブロック６８１はファイルＡの有効データを蓄積するほか、データを蓄積しない消去済み容量も持つプログラムブロックである。ファイルＡの状態しだいで決まる再生操作のひとつとして、ブロック６８１にあるファイルＡのデータは別のパーシャルブロック６８５の使用可能な消去済み容量にコピーされ、パーシャルブロック６８５の中には既に別のファイルＢのデータが入っているため、このパーシャルブロックは共通ブロックになる。ＦＩＴの中ではブロック６８１のデータグループが参照されなくなり、このブロックは用済みブロックとして記録される。ブロック６８１に蓄積されていたときのファイルＡは、プログラムブロック状態等、いずれか１つの状態（図１７参照）をとっていた。その後このデータは別のフラクタルブロックへ移されるが、ファイルの書き込みは最大の２フラクタルブロック以内にとどまる。ファイルＡはブロック６８５へコピーされた後、共通ブロックにファイルデータが蓄積されることを含む、いずれか１つの状態（図１７参照）に遷移するが、これは別のファイルデータを蓄積するブロックのタイプしだいで決まる。

【０１３１】

図２８Ａのブロック６８３は共通ブロックであり、そこに蓄積されたファイルＣおよびＤのデータがプログラムブロック６８７の消去済み容量にコピーされることによって再生され、このコピーによってファイルＥのデータが入っているプログラムブロック６８７は共通ブロックになる。そして、ブロック６８３のファイルＣおよびＤのデータは用済みになり、ブロックそのものも用済みになる。データはある１つの共通ブロックから別の共通ブロックへ移されたため、ファイルＣおよびＤの状態は変化していない。しかし、ファイルＥの状態は変化している。これとは別に、ファイルＣおよびＤのデータをそれぞれ別々のブロックへ移すこともでき、必ずしも共通ブロックの空き領域にコピーする必要はない。そこでこれらのファイルの状態が別の状態に遷移する可能性もある。

【０１３２】

図２８Ｂには、ブロックの例６８９および６９１に対する再生操作が示されている。これらのブロックはいずれも有効データと用済みデータの両方で埋まっているため、用済みブロックである。ブロック６８９はファイルＦのデータを収容するファイルブロックであって、その一部分は用済みで、残りの部分は有効である。これは例えばファイルＦの更新のときに、ファイルの既存データと同じ論理オフセットを持つ新規データがファイルの末尾のアドレスに書き込まれると生じ、既存データは用済みになる。この例では、ファイルＧのデータが入っているプログラムブロック６９３の消去済み容量にファイルＦのデータがコピーされることによって、ブロック６９３のタイプは共通ブロックに変化する。これとは別に、ファイルＦの有効データを消去済みブロックに書き込むこともでき、書き込まれたブロックはプログラムブロックになる。

【０１３３】

図２８Ｂのブロック６９１は、ファイルＨの無効データとファイルＩの有効データを収容するフル共通ブロックである。この例では、ブロック６９１からファイルＩの有効データを消去済みブロック６９５にコピーしている。この場合、ブロック６９５はプログラムブロックになる。これとは別に、別のファイルのデータが入っているパーシャルブロックで適切なものが見つかるならば、そのパーシャルブロックにファイルＩのデータを書き込むこともできる。コピー先ブロックは、再生操作のときのファイルＩの状態しだいで決まる。

【０１３４】

図２８Ａおよび図２８Ｂに見られる４つの再生操作の具体例の結果、２つのパーシャルブロックに蓄積されたデータは１つにまとまり、２ブロックのうちの一方には用済みデータだけが残る。それらのブロックは無効ブロックになる。元のブロック６８１、６８３、６８９、および６９１は図２８Ｃのように消去され、全領域が再生される。消去済みブロックは、無効ブロックを再生した結果である。

【０１３５】

図２８Ｄは、ファイルＪのデータを蓄積するファイルブロック６９７の例を示す。ホス

トによってファイルＪが削除されるとブロック６９７にあるファイルＪのデータは用済みになり、場合によっては別のブロックにあるデータも用済みになる。そして、ブロック６９７は無効になる。無効ブロックの再生によって、システム消去済みブロックプールに消去済みブロックが提供される。

【０１３６】

メモリからファイルが削除されると、通常は共通ブロックやフル共通ブロックといった１つ以上のフラクタルブロックにある当該ファイルのデータも用済みになる。ファイルが異なる残りの有効データはブロックの蓄積容量に満たなく少量なので、このブロックは再生操作の対象となる。

【０１３７】

図２９のフローチャートは再生操作を一般的な言葉で説明するものである。ステップ７０１に示すように、具体的な実施形態に応じてパーシャル、用済み、および無効ブロックのリストを１つ以上管理する。このブロックリストは、一手法によると、メモリシステムの起動時、例えば最初に電力が印加されるときに作成される。リストには、各ブロックの有効データ量や各ブロックの消去済み容量等、一度に１つの再生ブロックを選択するためのブロックの他の情報を盛り込むことができる。通常、これらの量はブロックのページ数単位で表され、メタブロックを使用する場合はメタページ数単位で表される。好適な代替手法では、これらのリストを不揮発性メモリで管理し、ブロックの状態が変化するたびにリストにブロックの項目を加えたり、リストのブロックの項目を更新したりする。この手法では、メモリシステムの初期化のときにブロックを走査しリストを作成する必要はない。コピーの必要がある有効データが皆無かごく僅かということが再生ブロックとして選ばれるひとつの特徴となるため、パーシャル、用済み、および無効の全ブロックをリストで管理する代わりに、有効データが少量で所定の限界量を下回るブロックだけをリストに入れる方法もある。多くの再生操作で必要となるのは、ある１つのブロックから別のブロックへのデータのコピーであって、それにはかなりの時間がかかるため、通常はコピーの必要があるデータが少ないブロックから先にコピーを行う。

【０１３８】

そのようなブロックのリストは、データの書き込み、更新、移動、削除等にともない絶えず変化する。パーシャル、用済み、無効の中でブロックタイプの変化を招くような変更があると、図２９のステップ７０１で管理するリストも変化する。そのようなブロックに蓄積された有効データの量や消去済み容量の変化もブロックリストに記録される。

【０１３９】

ステップ７０３では、好ましくは再生する次のブロックとして更新済みリストのブロックから１つの再生ブロックを識別する。パーシャルブロックか用済みブロックならば、これが有効データのコピー元となり、そこからコピー先ブロックと呼ばれる別のブロックに有効データをコピーする。コピー元ブロックの選択に用いるいくつかの具体的な手法は後述する。

【０１４０】

図２９の次のステップ７０５では、ホストのコマンドに応じて実行するメモリ操作を考慮に入れ、今ここで再生操作を実行することが適切か否かを判断する。ホストがアイドルコマンドやこれに類似するものを発行し、ある程度の期間にわたってメモリシステムによる操作の実行を見込まないことを伝えるなら、メモリシステムはフォアグラウンドで再生操作を含むオーバーヘッド操作を自由に実行できる。ホストがメモリシステムへのデータの書き込みやメモリシステムからのデータの読み出しでビジー状態にある場合でも、再生操作、特にそのデータコピーは、インターリーブ方式でデータの読み書き操作の合間に実行できる。２００５年１０月２５日に出願されたAlan Sinclair の米国特許出願第１１／２５９，４２３号（特許文献３０）と２００５年１２月１９日に出願されたAlan Bennettらの米国特許出願第１１／３１２，９８５号（特許文献３１）では、物理メモリセルブロックのためのインターリーブ操作が説明されている。

【０１４１】

再生操作を行ってよいことが図２９のステップ７０５で判明した場合のプロセスは、識別された再生ブロックに有効データが入っているか否かに応じて、さらに有効データが入っている場合には、２つ以上のファイルの有効データが入っているか否かに応じて、異なる。パーシャルブロックか用済みブロックなら、必然的に有効データを収容していることになり、共通ブロックかフル共通ブロックなら、２つ以上のファイルの有効データを収容していることになる。再生ブロックにおける有効データの有無はステップ７０７で判断する。移動の必要がある有効データがある場合は、次のステップ７０９で１ファイルのデータを識別し、さらにそのデータを受け付けるコピー先ブロックを識別する。有効データを含むファイルの全データの蓄積を２フラクタルブロック以下（この例）に保つため、コピー先ブロックは図１７～図１９との関係で前述したプロセスで識別する。そして、ステップ７１１に示すように、コピー元の再生ブロックからコピー先ブロックへの１ファイルの有効データのコピーを開始する。処理はこれらのデータをコピーした後にステップ７０７まで戻り、別のファイルのデータが残っているか否かを判断する。残っているなら、そのデータのためにステップ７０９および７１１のプロセスを繰り返す。そのコピー先ブロックは、別のファイルのデータのために先に選ばれるブロックとは別に選ばれる。これは、移動すべきデータがコピー元ブロックにそれ以上存在しないことがステップ７０７で判明するまで続き、存在しないことが判明する場合は、ステップ７１３に従ってコピー元ブロックを消去できる。次に、消去済みブロックプールの中にこのブロックを入れ、新規データの蓄積に使えるようにする。

【０１４２】

図２９のステップ７０７に戻り、無効ブロックの場合のようにコピー元ブロックの中に有効データがなければ、移動すべき有効データもない。このコピー元ブロックは消去するだけでよい。図２９に見られるように、この場合の処理はステップ７０９および７１１を迂回する。

【０１４３】

図２９のプロセスの第１の実施形態では、ステップ７０１でパーシャル、用済み、および無効ブロックからなる単一のリストを管理する。リストの各項目にはブロック内の有効データ量を盛り込む。ステップ７０３で再生ブロックとしてリストから選ばれるブロックは、有効データが最も少ないブロックである。無効ブロックには有効データがないため、リストに無効ブロックが１つでもあるならば、そのブロックが最初に選ばれる。リストに多数の無効ブロックがあるなら、リストに登録されていた期間が最も長い無効ブロックが選ばれる。リストに無効ブロックがなければ、有効データ量が最も少ないブロックが再生ブロックに選ばれる。リスト上の全ブロックから有効データ量が最少のブロックを選ぶことで、再生操作にかかる時間は、あるブロックから別のブロックへコピーするより多くの有効データがある時よりも短くなる。その結果、メモリにおけるデータの読み書き速度等、再生操作以外のメモリシステム操作は高速に保たれる。メモリ性能への代償を抑えながら新たに消去されたブロックが手に入ることになる。

【０１４４】

単一リスト上でフラクタルブロックの有効データ量に基づきコピー元ブロックを選択する図２９のプロセスの第１の実施形態には、実装が比較的簡単にすむという利点がある。しかし、パーシャルブロックの価値を考慮に入れて、このプロセスを改良することもできる。パーシャルブロックには消去済み容量があり、そこにはデータを書き込むことができるが、用済みブロックと無効ブロックには消去済み容量がない。用済みブロックを新規データの蓄積に使うには、事前にそこから別のブロックへ有効データを移し、用済みブロックを消去し、新規データの蓄積に使える状態にしなければならない。しかし、パーシャルブロックには消去済み容量があって、再生操作のオーバーヘッドを被ることなくデータを書き込むことができる。例えば、有効データ量が最も少ないという理由だけで、データの書き込みが可能な消去済み容量を大量に保有するパーシャルブロックを再生するのは得策でない。

【０１４５】

したがって、図２９のプロセスの別の実施形態では、パーシャルブロックにある有効データの量と消去済み容量の両方を踏まえて、コピー元の再生ブロックの候補となるパーシャルブロックを選ぶ。図３０は、パーシャルブロックの中にあるデータの構成要素を示す。このブロック（またはメタブロック）には、有効データを収容する１つ以上のページ（またはメタページ）のほかに、データを書き込める１つ以上の消去済みページがある。図３０の例に見られるように、パーシャルブロックの１つ以上のページには用済みデータが入ることもある。

【０１４６】

図２９のこれらのプロセスの実施形態では、好ましくは用済みブロックと無効ブロックのリストとは別のリストでパーシャルブロックをステップ７０１で管理する。消去済み容量がごく僅かで（現状ではさほど有用でないことを意味する）、移動を要する有効データが少ないパーシャルブロックは、再生操作リストの先頭寄りに移される。そのようなブロックの主な内容は用済みデータになる。逆に、再生ブロックの候補となる見込みが最も低いのは、大量の消去済み容量があり（データの蓄積に役立つ見込みがあることを意味する）、移動を要する有効データが大量にあるパーシャルブロックである。消去済み容量があるパーシャルブロックを再生しても、用済みブロックを再生した場合と同量の蓄積容量が論理アドレス空間に追加されるわけではない。役に立つ消去済み容量がなく、コピーを要する有効データもない無効ブロックは明らかに、再生の対象として最も魅力的なブロックなのである。

【０１４７】

図２９の再生ブロック識別ステップ７０３の第２の実施形態では、パーシャルブロックと、用済みブロックと、無効ブロックのそれぞれに１つずつ、合わせて３通りのリストをステップ７０１で管理する。無効ブロックがあるなら、無効ブロックのリストでブロックがなくなるまでこのリストから再生ブロックを選択する。無効ブロックのリストに特別な順序はないが、先入れ先出し（ＦＩＦＯ）順にして、リストに登録された期間が最も長い無効ブロックが最初に選ばれるようにすることも可能である。次に無効ブロックがなければ、用済みブロックリストにある全ブロックで有効データ量が最も少ないブロックを選ぶ。

【０１４８】

ステップ７０３で無効ブロックリストか用済みブロックリストにブロックがなければ、パーシャルブロックリストのブロックを再生ブロックに選ぶ。有効データ量が最も少ないパーシャルブロックが選ばれるようにすることもできるが、好ましくは消去済み容量の有用性が分かる形でパーシャルブロックを格付けする。それには、各パーシャルブロックの「再生利得」を次のとおり計算する。

再生利得＝（Ｓ－ｋＥ）／Ｖ　（１）

式中、Ｓは合計データ蓄積ページ数によるブロックサイズであり、Ｅはデータを書き込める消去済み容量のページ数であり、Ｖは別のブロックへ移す必要がある有効データを含むページ数である。定数ｋはブロックの消去済み容量の有用性を重み付けするためのものであるが、１に設定することもできる。ｋＥの値が増すにつれ再生利得は低下する。Ｖの値が増す場合も再生利得は低下する。ステップ７０３では、再生利得の値が最も高いパーシャルブロックが再生ブロックとして選ばれる。これとは別の数式で項ＥおよびＶを使って再生利得を定義し、有効データを保持することのシステム動作にとってのデメリットと消去済み容量のメリットとでバランスをとることもできる。再生利得はブロックに変化があるたびに、例えばブロックの消去済み容量にデータが書き込まれるたびに、計算し、ファイルディレクトリやＦＩＴで管理する情報の一部として蓄積することができる。

【０１４９】

この第２の実施形態を示す図３１は、パーシャルブロックリスト、用済みブロックリスト、および無効ブロックリストの３通りのリスト（図２９のステップ７０１で管理）から再生ブロックを選択（図２９のステップ７０３）する方法を説明するものである。ステップ７２１ではまず、無効ブロックリストに記載されたブロックの有無を判断する。そのよ

うなブロックが多数存在する場合は、リストに登録された期間が最も長いブロックがステップ７２３で再生ブロックに選ばれる。無効ブロックリストにブロックがなければ、ステップ７２５で用済みブロックリストにおける項目の有無を判断する。その際、用済みブロックリストに２つ以上のブロックがある場合は、ステップ７２７で有効データ量が最も少ないブロックが再生ブロックに選ばれる。用済みブロックリストに項目がないことがステップ７２５で判明する場合は、ステップ７２９でパーシャルブロックリストをあたる。パーシャルブロックリストに２つ以上のブロックがある場合は、再生利得が最も高いブロックが再生ブロックに選ばれる。再生利得では、前述した式（１）を使用する等して、ブロック内の有効データ量と消去済み容量を考慮に入れる。パーシャルブロックリストに何もなければステップ７２１まで戻り、リストのいずれか１つでブロックが出現するまでプロセスを繰り返す。再生ブロックが選択された後、処理は図２９のステップ７０５へ進む。

【０１５０】

図３２のフローチャートに第３の実施形態を示す。ステップ７４１でも図２９のステップ７０３の実行が始まり、図２９のステップ７０１で管理する無効ブロックリストで項目を探す。無効ブロックリストに２つ以上の項目がある場合は、図３２のステップ７４３で最も古いブロックが再生ブロックに選ばれる。無効ブロックリストに項目がなければ、次のステップ７４５により用済みブロックリストで項目の有無を判断する。項目がある場合の後続するステップは図３１の実施形態と異なり、パーシャルブロックリストにも１つ以上の項目がある場合は、用済みブロックリストとパーシャルブロックリストのどちらで再生ブロックを選べばよいかを判断する。

【０１５１】

図３２のステップ７４７では、有効データ量が最も少ないブロックを用済みブロックリストで識別する。次に、ステップ７４９でパーシャルブロックリストに１つ以上のブロックが存在するか否かを判断し、存在する場合は、有効データ量が最も少ないブロックをステップ７５１で識別する。次のステップ７５３では、用済みブロックリストで識別した１ブロックとパーシャルブロックリストで識別した１ブロックとで選択を行う。それには、前と同じ項Ｖ、Ｅ、およびｋを使用し、ステップ７５１でパーシャルブロックリストから識別したブロックで量（Ｖ＋ｋＥ）を計算する。この量をステップ７４７で用済みブロックリストから識別したブロックの有効データ量Ｖに比較する。パーシャルブロックの量（Ｖ＋ｋＥ）が用済みブロックのＶを上回るなら、ステップ７５５で用済みブロックが再生ブロックに選ばれる。しかし、用済みブロックのＶが識別されたパーシャルブロックの量（Ｖ＋ｋＥ）を上回るなら、ステップ７５７でパーシャルブロックが再生ブロックに選ばれる。

【０１５２】

識別されたパーシャルブロックの消去済み容量ｋＥをこのブロックの有効データＶに加えたものを、識別された用済みブロックの有効データＶのみに比較すると、そのプロセスは用済みブロックが優先的に選ばれる形に偏る。識別された用済みブロックと同量の有効データを持つ識別されたパーシャルブロックには、その消去済み容量にデータを蓄積するという使い道が残っているため、選ばれずに残る。実際は、有効データ量が用済みブロックの有効データ量よりｋＥ少ないパーシャルブロックが選ばれずに残る。

【０１５３】

図３２のステップ７４５へ戻り、用済みブロックリストに項目がない場合は、パーシャルブロックリストに記載されたブロックの有無をステップ７５９で判断する。ブロックがなければプロセスはステップ７４１まで戻り、３つのリストのいずれか１つにブロックが登録されるまで繰り返される。リストに多数のパーシャルブロックがある場合は、ステップ７６１で有効データ量が最も少ないブロックを再生ブロックとして選択する。あるいは、第２の実施形態（図３１）のステップ７３１との関係で説明したように、再生利得を用いてパーシャルブロックを選択することもできる。

【０１５４】

第３の実施形態では使用するリストを２つに絞ることもできる。第１のリストは用済み

ブロックリストで、このリストには、用済みデータを収容し消去済み容量がないブロックの項目が入る。図３２に見られるように単独の無効ブロックリストを使用するのではなく、無効ブロックと用済みブロックの両方を１つの「用済み」ブロックリストに入れる。これらのブロックは有効データを収容することもある。リストの各項目にはフィールドがあり、このフィールドには該当するブロックの有効データ量を表す値が入る。リストの項目は、このフィールドの値の順に並べる。その結果、用済みデータがあって有効データはないブロック（無効ブロック）は、この第１のリストの先頭に集まる。

【０１５５】

この第３の実施形態の代案で、第２のリストはパーシャルブロックリストであり、このリストには、ある程度の消去済み蓄積容量を持つブロックの項目が入る。これらのブロックは有効データを収容することもある。リストの各項目にはフィールドがあり、このフィールドには該当するブロックの有効データ量を表す値が入る。リストの項目は、このフィールドの値の順に並べる。図３２のステップ７５３の手法により、第１または第２のリストの先頭からブロック（有効データの量が最も少ないブロック）を選ぶことができる。

【０１５６】

図３３の表は、この第３の実施形態の修正版による再生操作でパーシャルブロックリストと用済みブロックリストに入るブロックタイプの詳細を伝えるものである。パーシャルブロックリストに入るブロックは、有効データと消去済み容量の両方を含むブロックである。ブロックに用済みデータがあるかどうかは問題とされない。用済みブロックリストに入るブロックは、用済みデータを含み、かつ有効データと消去済み容量の両方ではなくいずれか一方を含むブロックである。

【０１５７】

結論

これまで本発明の様々な態様をその代表的な実施形態との関係で説明してきたが、本発明が添付の特許請求の範囲内で保護を受ける権利があることは理解される。

【図１】

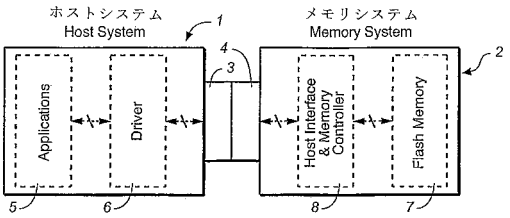

FIG. 1

5 アプリケーション
6 ドライバ
7 フラッシュメモリ
8 ホストインターフェイスとメモリコントローラ

【図２】

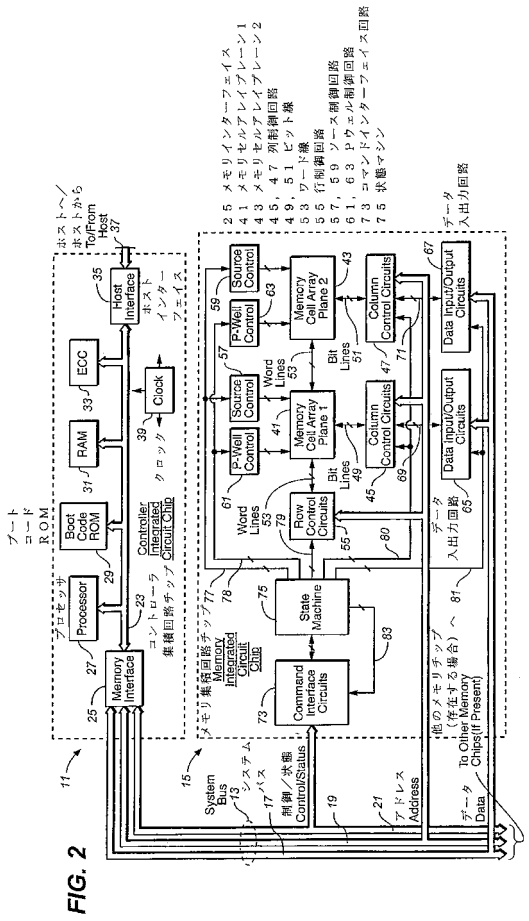

FIG. 2

【図３】

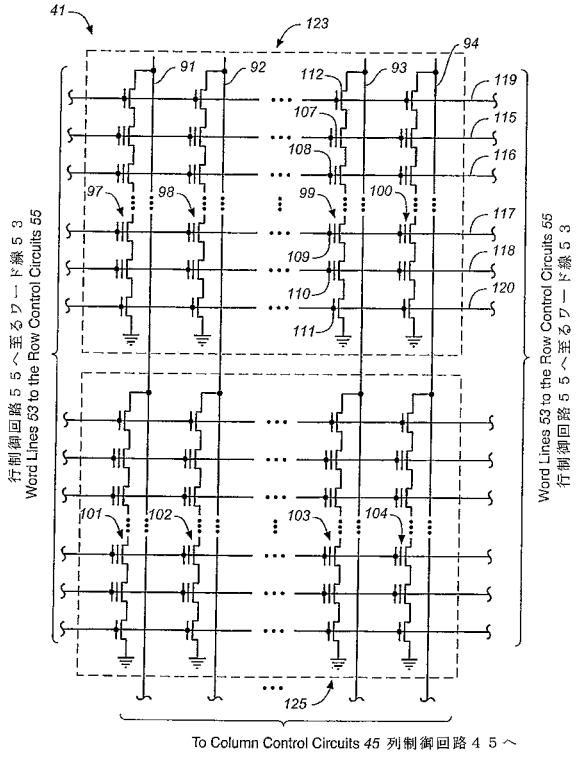

FIG. 3

【図４】

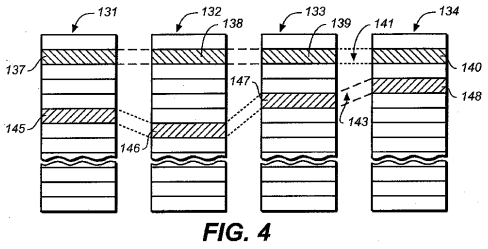

FIG. 4

【図５】

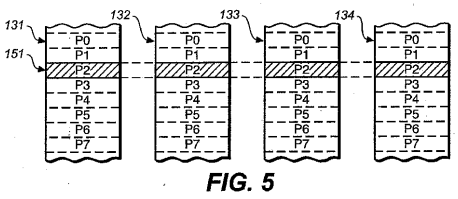

FIG. 5

【図６】

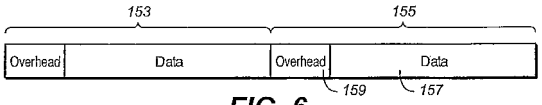

FIG. 6

Overhead オーバーヘッド
Data データ

【図７Ａ】

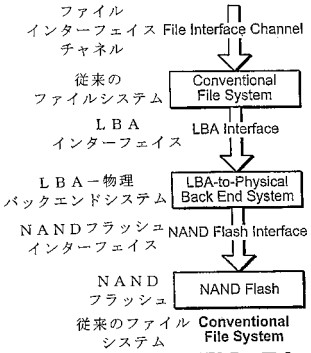

FIG. 7A

【図７Ｂ】

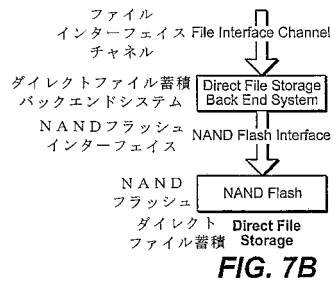

FIG. 7B

【図７Ｃ】

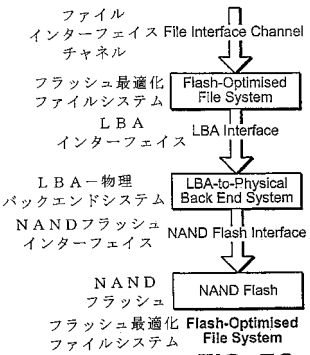

FIG. 7C

【図８Ａ】

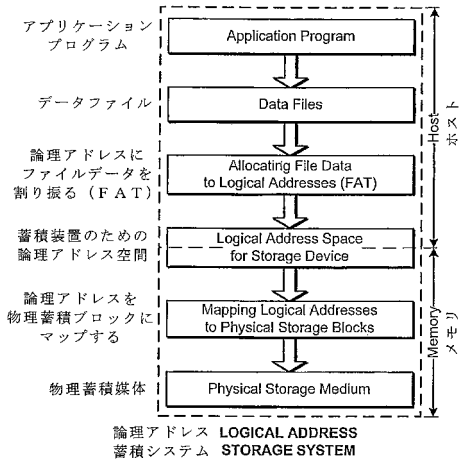

FIG. 8A

【図８Ｂ】

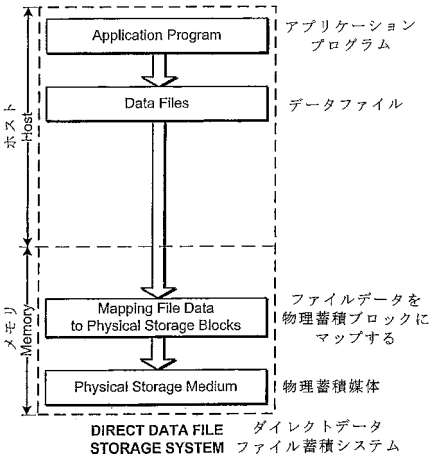

FIG. 8B

【図８Ｃ】

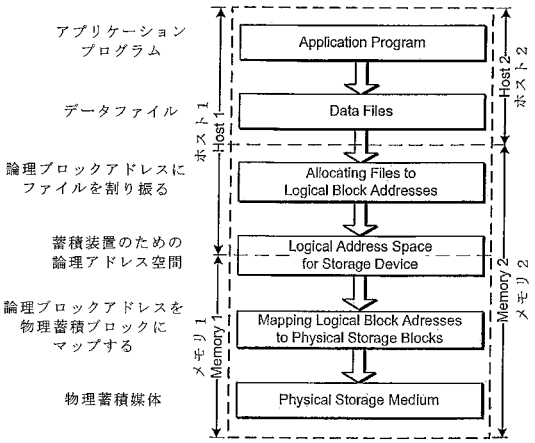

FIG. 8C

【図９Ａ】

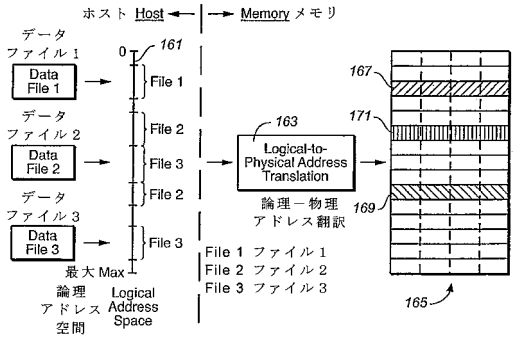

FIG. 9A

【図９Ｂ】

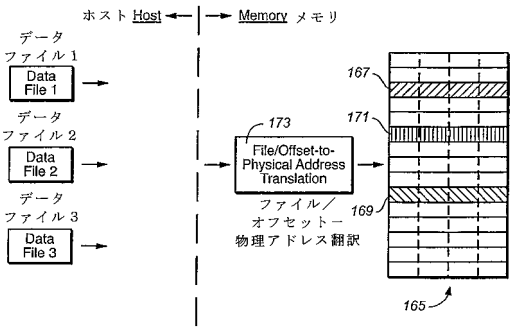

FIG. 9B

【図９Ｃ】

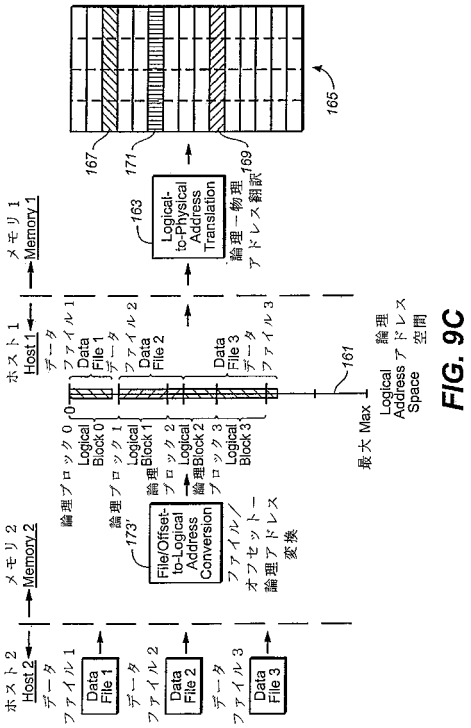

FIG. 9C

【図１０】

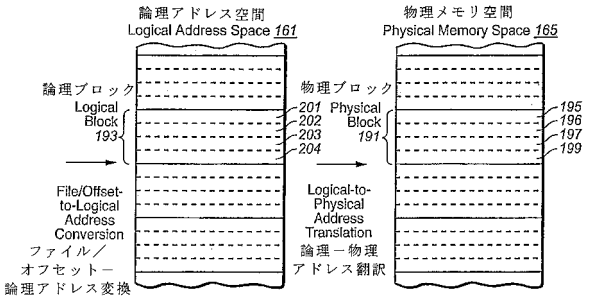

FIG. 10

【図１１】

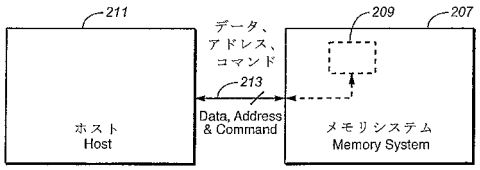

FIG. 11

【図１２】

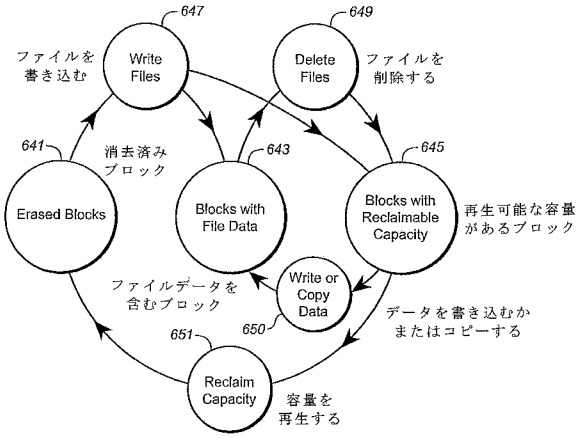

FIG. 12

【図１３Ａ】

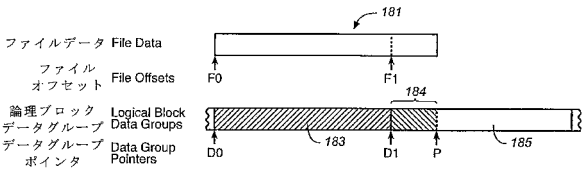

FIG. 13A Write 書き込み

【図１３Ｂ】

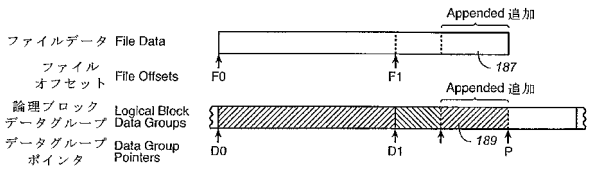

FIG. 13B Write 書き込み

【図１３Ｃ】

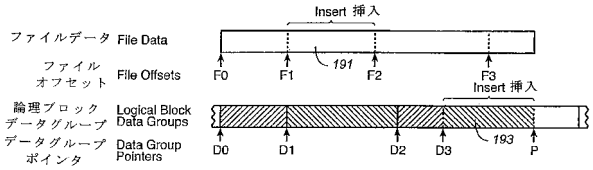

FIG. 13C Insert 挿入

【図１３Ｄ】

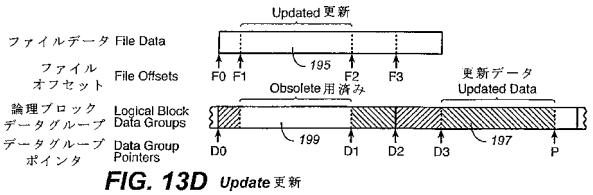

FIG. 13D Update 更新

【図１４Ａ】

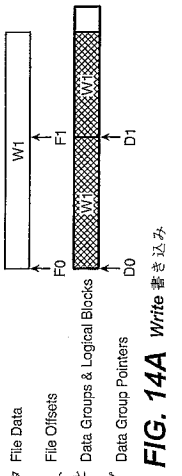

FIG. 14A *Write* 書き込み

【図１４Ｂ】

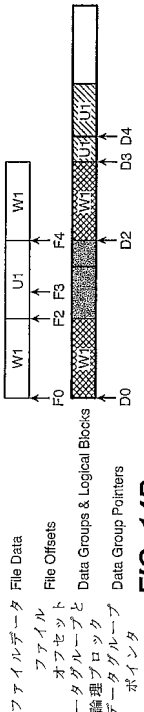

FIG. 14B *Update* 更新

【図１４Ｃ】

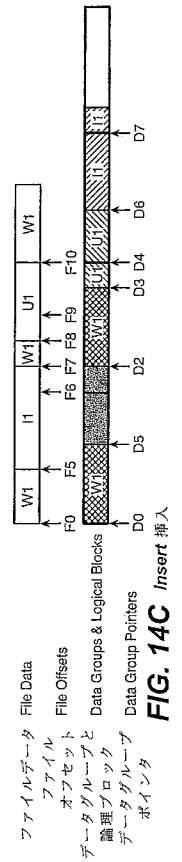

FIG. 14C *Insert* 挿入

【図１４Ｄ】

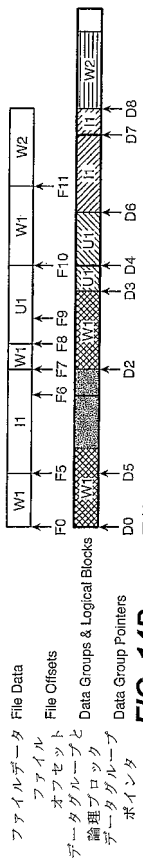

FIG. 14D *Update* 更新

【図１４Ｅ】

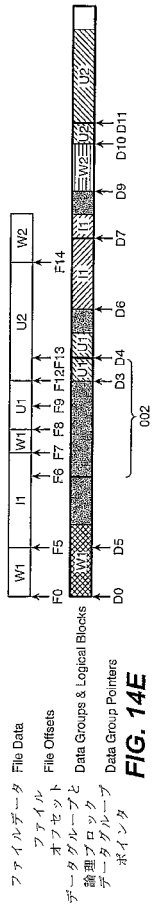

FIG. 14E

【図１５】

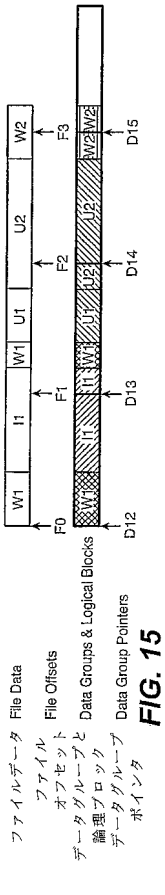

FIG. 15

【図１６Ａ】

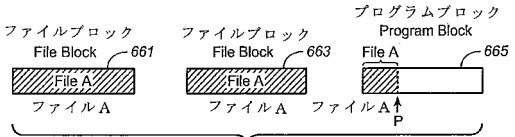

FIG. 16A

【図１６Ｂ】

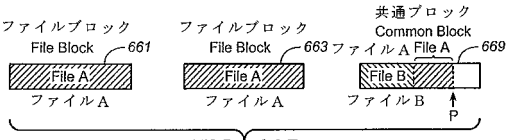

FIG. 16B

【図１６Ｃ】

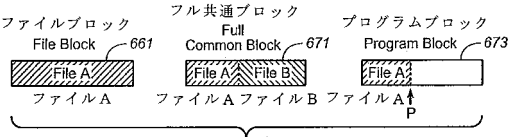

FIG. 16C

【図１６Ｄ】

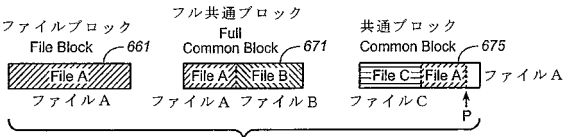

FIG. 16D

【図１７】

許容ファイル状態

ファイル状態	フラクタルブロック	
	フル共通ブロック	パーシャルブロック
００	無	無
０１	無	プログラムブロック
０２	無	共通ブロック
１０	フル共通ブロック	無
１１	フル共通ブロック	プログラムブロック
１２	フル共通ブロック	共通ブロック
２０	２つのフル共通ブロック	無

Permitted File States

File State	Fractal Blocks	
	Full Common Block	Partial Block
00	None	None
01	None	Program Block
02	None	Common Block
10	Full Common Block	None
11	Full Common Block	Program Block
12	Full Common Block	Common Block
20	Two Full Common Blocks	None

FIG. 17

【図１８】

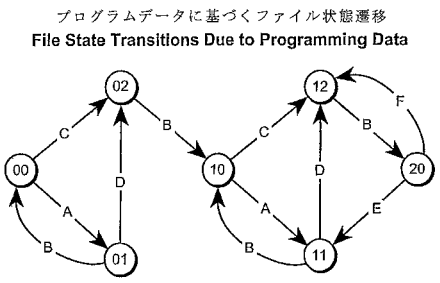

FIG. 18

【図１９】

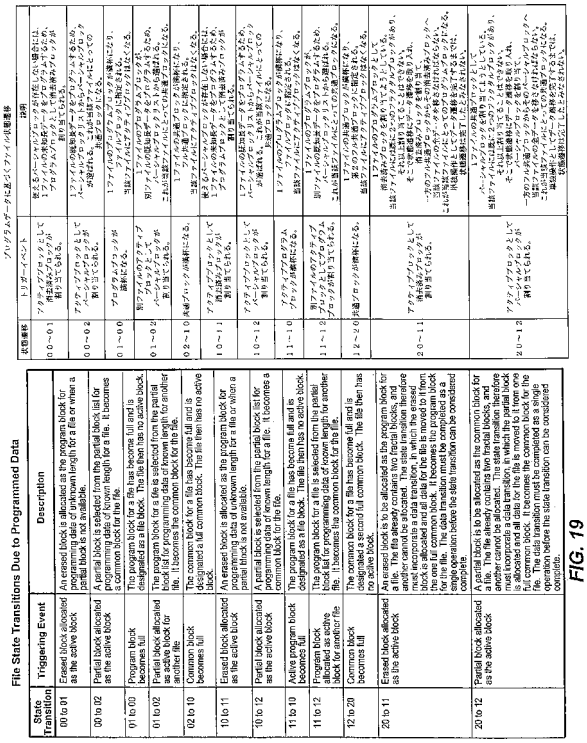

FIG. 19

【図２０】

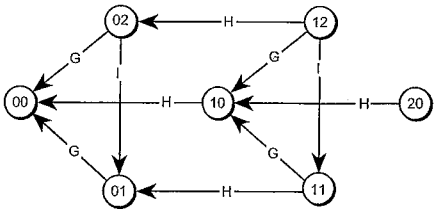

FIG. 20

【図２１】

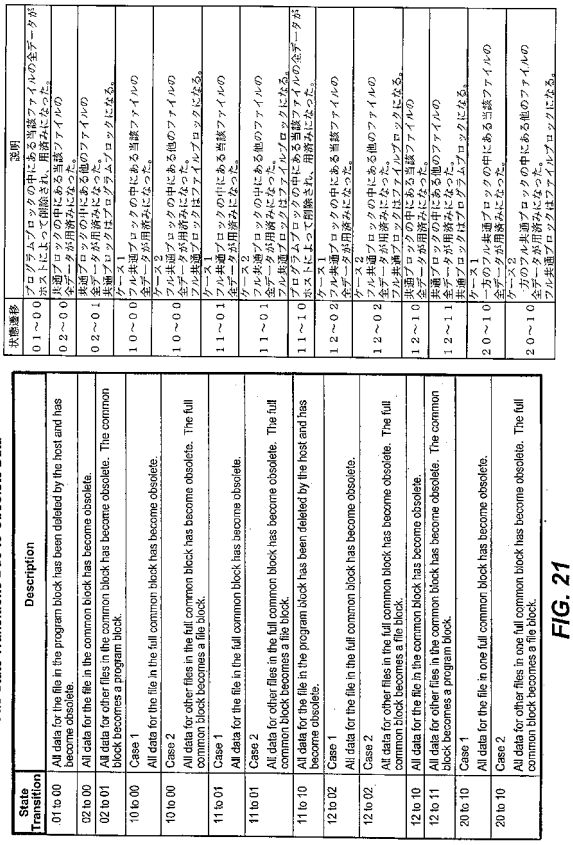

FIG. 21

【図２２】

再生ブロック選択に基づくファイル状態遷移
File State Transitions Due to Reclaim Block Selection

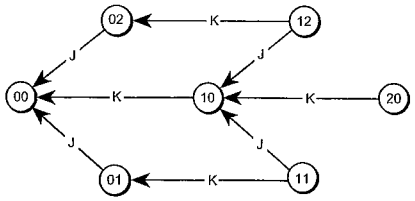

FIG. 22

【図２３】

再生ブロック選択に基づくファイル状態遷移

状態遷移	説明
０１～００	再生ブロックとしてプログラムブロックが選ばれる
０２～００	再生ブロックとして共通ブロックが選ばれる
１０～００	再生ブロックとしてフル共通ブロックが選ばれる
１１～０１	再生ブロックとしてフル共通ブロックが選ばれる
１１～１０	再生ブロックとしてプログラムブロックが選ばれる
１２～０２	再生ブロックとしてフル共通ブロックが選ばれる
１２～１０	再生ブロックとして共通ブロックが選ばれる
２０～１０	再生ブロックとしてフル共通ブロックが選ばれる

File State Transitions Due to Reclaim Block Selection

State Transition	Description
01 to 00	Program block selected as reclaim block.
02 to 00	Common block selected as reclaim block.
10 to 00	Full common block selected as reclaim block.
11 to 01	Full common block selected as reclaim block.
11 to 10	Program block selected as reclaim block.
12 to 02	Full common block selected as reclaim block.
12 to 10	Common block selected as reclaim block.
20 to 10	Full common block selected as reclaim block.

FIG. 23

【図２４】

ブロック先頭へのファイル整合
Alignment of Files to Beginning of Blocks

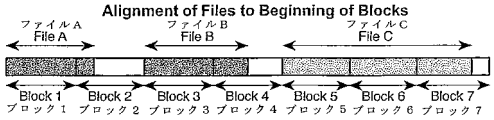

FIG. 24

【図２５】

ブロックの先頭にファイルを揃える場合のアクティブブロック割り当て

割り当てケース	現行ファイル状態	全般状況	ブロックタイプ割り当て優先順位
A	００	プログラム先ブロックを上回る未知長または既知長のデータ	１．消去済みブロック ２．最大パーシャルブロック
B	００	プログラム先ブロックを下回る既知長のデータ	１．最適パーシャルブロック ２．最大パーシャルブロック ３．消去済みブロック
C	１０	プログラム先ブロックを上回る未知長または既知長のデータ	１．消去済みブロック ２．最大パーシャルブロック
D	１０	プログラム先ブロックを下回る既知長のデータ	１．最適パーシャルブロック ２．消去済みブロック
E	２０	プログラム先ブロックを上回る未知長または既知長の集合データ（集合データとは、データ遷移中に再配置されるデータと、別の場所からプログラムされるデータを合わせたもの）	１．消去済みブロック
F	２０	プログラム先ブロックを下回る既知長の集合データ（集合データとは、データ遷移中に再配置されるデータと、別の場所からプログラムされるデータを合わせたもの）	１．最適パーシャルブロック ２．消去済みブロック

Allocation of Active Blocks When Files Aligned to Beginning of Blocks

Allocation Case	Existing File State	Prevailing Condition	Order of Priority for Block Type to be Allocated
A	00	Data of unknown length or known length greater than a block to be programmed	1. Erased block 2. Biggest partial block
B	00	Data of known length less than a block to be programmed	1. Best fit partial block 2. Biggest partial block 3. Erased block
C	10	Data of unknown length or known length greater than a block to be programmed	1. Erased block 2. Biggest partial block
D	10	Data of known length less than a block to be programmed	1. Best fit partial block 2. Erased block
E	20	Aggregate data of unknown length or known length greater than a block to be programmed, where aggregate data is the sum of data to be relocated during data transition and data to be programmed from another source	1. Erased block
F	20	Aggregate data of known length less than a block to be programmed, where aggregate data is the sum of data to be relocated during data transition and data to be programmed from another source	1. Best fit partial block 2. Erased block

FIG. 25

【図２６】

ブロック内でのファイルの隣接
Abutting of Files Within Blocks

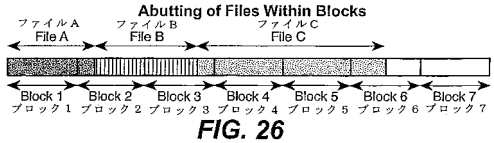

FIG. 26

【図２７】

ブロック内でファイルが隣接する場合のアクティブブロック割り当て

割り当てケース	現行ファイル状態	全般状況	ブロックタイプ割り当て優先順位
A１	００	プログラム先ブロックを上回る未知長または既知長の新規ファイルデータ ファイルの先頭はブロックの中で他のデータと隣接する	１．最大パーシャルブロック ２．消去済みブロック
A２	００	プログラム先ブロックを上回る未知長または既知長の既存ファイルデータ	１．消去済みブロック ２．最大パーシャルブロック
B	００	プログラム先ブロックを下回る既知長のデータ	１．最適パーシャルブロック ２．最大パーシャルブロック ３．消去済みブロック
C	１０	プログラム先ブロックを上回る未知長または既知長のデータ	１．消去済みブロック ２．最大パーシャルブロック
D		プログラム先ブロックを下回る既知長のデータ	１．最適パーシャルブロック ２．消去済みブロック
E	１０	プログラム先ブロックを上回る未知長または既知長の集合データ（集合データとは、データ遷移中に再配置されるデータと、別の場所からプログラムされるデータを合わせたもの）	１．消去済みブロック
F	２０	プログラム先ブロックを下回る既知長の集合データ（集合データとは、データ遷移中に再配置されるデータと、別の場所からプログラムされるデータを合わせたもの）	１．最適パーシャルブロック ２．消去済みブロック

Allocation of Active Blocks for Abutting of Files Within Blocks

Allocation Case	Existing File State	Prevailing Condition	Order of Priority for Block Type to be Allocated
A1	00	Data for a new file of unknown length or known length greater than a block to be programmed. Start of file to abut other data within a block.	1. Biggest partial block 2. Erased block
A2	00	Data for an existing file of unknown length or known length greater than a block to be programmed	1. Erased block 2. Biggest partial block
B	00	Data of known length less than a block to be programmed	1. Best fit partial block 2. Biggest partial block 3. Erased block
C	10	Data of unknown length or known length greater than a block to be programmed	1. Erased block 2. Biggest partial block
D	10	Data of known length less than a block to be programmed	1. Best fit partial block 2. Erased block
E	20	Aggregate data of unknown length or known length greater than a block to be programmed, where aggregate data is the sum of data to be relocated during data transition and data to be programmed from another source	1. Erased block
F	20	Aggregate data of known length less than a block to be programmed, where aggregate data is the sum of data to be relocated during data transition and data to be programmed from another source	1. Best fit partial block 2. Erased block

FIG. 27

【図２８Ａ】

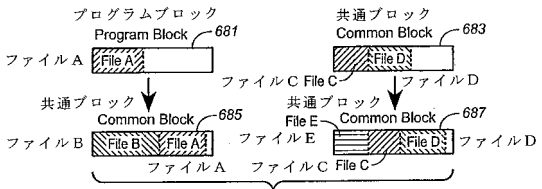

FIG. 28A

【図２８Ｂ】

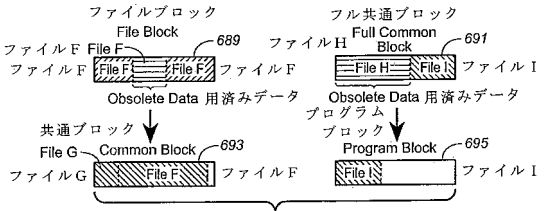

FIG. 28B

【図２８Ｃ】

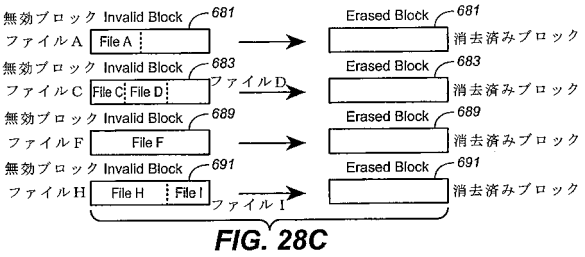

FIG. 28C

【図２８Ｄ】

FIG. 28D

【図２９】

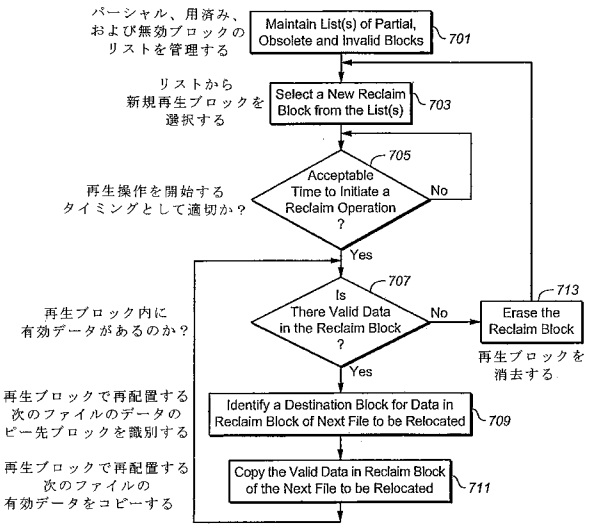

FIG. 29

【図３０】

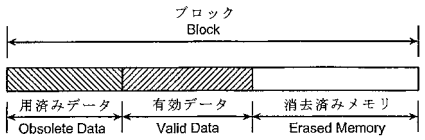

FIG. 30

【図３１】

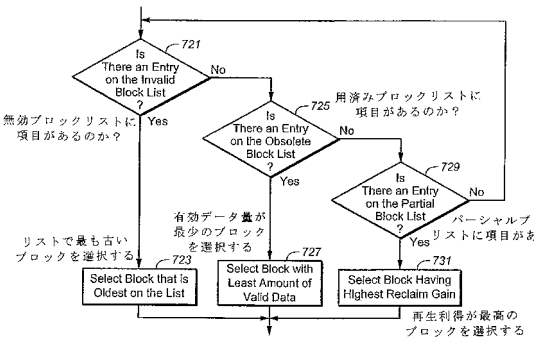

FIG. 31

【図３２】

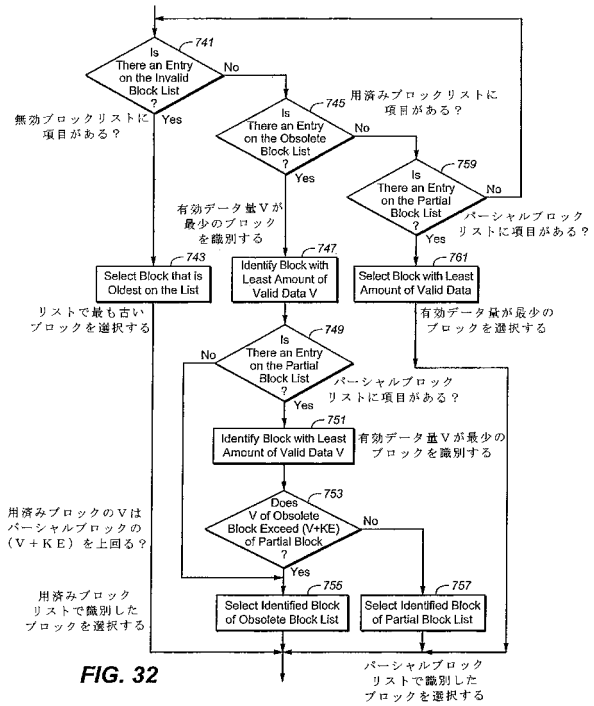

FIG. 32

【図３３】

ブロックタイプ	ブロック内容			ブロックリスト
	有効データ（V）	消去済み容量（E）	用済みデータ（O）	
プログラム	有	有	無視	パーシャル
共通	有	有	無視	パーシャル
フル共通	有	無	無	無
			有	用済み
ファイル	有	無	無	無
			有	用済み
無効	無	有	有	用済み
消去済み	無	有	無	消去済み

Block Type	Block Contents			Block List
	Valid Data (V)	Erased Capacity (E)	Obsolete Data (O)	
Program	Yes	Yes	Don't Care	Partial
Common	Yes	Yes	Don't Care	Partial
Full Common	Yes	No	No	None
			Yes	Obsolete
File	Yes	No	No	None
			Yes	Obsolete
Invalid	No	Yes	Yes	Obsolete
Erased	No	Yes	No	Erased

FIG. 33

フロントページの続き

(72)発明者　ライト，バリー
イギリス連邦共和国、スコットランド、ＥＨ６　６ＤＥ、エディンバラ、ヘンダーソン　ストリート、４０　フラット　４

審査官　田川　泰宏

(56)参考文献　特表２００８－５３０７１０（ＪＰ，Ａ）
特開平１０－３２６２２７（ＪＰ，Ａ）
特表２０１０－５１５１６３（ＪＰ，Ａ）
特表２００９－５０３７４５（ＪＰ，Ａ）

(58)調査した分野(Int.Cl.，ＤＢ名)
Ｇ０６Ｆ　１２／００
Ｇ０６Ｆ　３／０６
Ｇ０６Ｆ　１２／０２