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

特許第3727982号

(P3727982)

(45)発行日　平成17年12月21日(2005.12.21)　(24)登録日　平成17年10月7日(2005.10.7)

(51)Int.Cl.[7]	F I		
G06F 12/14	G06F	12/14	560D
G06F 3/08	G06F	3/08	H
G06F 12/00	G06F	12/00	501H
G06F 12/02	G06F	12/00	537Z
	G06F	12/00	597U

請求項の数 4　(全 41 頁)　最終頁に続く

(21)出願番号　特願平7-251239
(22)出願日　平成7年9月28日(1995.9.28)
(65)公開番号　特開平9-97139
(43)公開日　平成9年4月8日(1997.4.8)
審査請求日　平成14年9月27日(2002.9.27)

(73)特許権者　000001007
キヤノン株式会社
東京都大田区下丸子３丁目３０番２号
(74)代理人　100076428
弁理士　大塚　康徳
(74)代理人　100093908
弁理士　松本　研一
(72)発明者　小川　武志
東京都大田区下丸子３丁目３０番２号　キヤノン株式会社内

審査官　多賀　実

最終頁に続く

(54)【発明の名称】フラッシュＲＯＭ管理方法及び装置

(57)【特許請求の範囲】

【請求項１】

データ領域、及びデータ領域に対応する管理領域とで構成される複数の記憶ブロックをフラッシュＲＯＭに形成し、各記憶ブロックにおいてデータ領域の記憶状態を示す状態情報を管理領域に格納し、該状態情報に基づいてフラッシュＲＯＭのアクセスを管理する管理手段と、

消去が指定されたファイルに含まれる記憶ブロックについて、該記憶ブロックの管理領域に格納されている状態情報を、当該記憶ブロック内のデータが無効であることを示す情報に更新することにより当該記憶を廃棄する廃棄手段と、

ファイルのディレクトリエントリテーブルにおいて、前記消去が指定されたファイルのエントリを、最後に格納されているエントリの内容で上書きし、当該最後に格納されているエントリに最後を表わす情報を上書きすることにより、前記指定されたファイルを、ファイルシステムが有する消去ファイル復元機能によって誤復元されることを防止する防止手段とを備えることを特徴とするフラッシュＲＯＭ管理装置。

【請求項２】

前記フラッシュＲＯＭへのアクセスに基づいて不要となった記憶ブロックを検出する検出手段を更に備え、

前記廃棄手段は、前記検出手段で検出された記憶ブロックの状態情報を、該記憶ブロックのデータが無効であることを示す情報に更新することを特徴とする請求項１に記載のフラッシュＲＯＭ管理装置。

【請求項３】
データ領域、及びデータ領域に対応する管理領域とで構成される複数の記憶ブロックをフラッシュＲＯＭに形成し、各記憶ブロックにおいてデータ領域の記憶状態を示す状態情報を管理領域に格納し、該状態情報に基づいてフラッシュＲＯＭのアクセスを管理する管理工程と、
消去が指定されたファイルに含まれる記憶ブロックについて、該記憶ブロックの管理領域に格納されている状態情報を、当該記憶ブロック内のデータが無効であることを示す情報に更新することにより当該記憶を廃棄する廃棄工程と、
ファイルのディレクトリエントリテーブルにおいて、前記消去が指定されたファイルのエントリを、最後に格納されているエントリの内容で上書きし、当該最後に格納されているエントリに最後を表わす情報を上書きすることにより、前記指定されたファイルを、ファイルシステムが有する消去ファイル復元機能によって誤復元されることを防止する防止工程とを備えることを特徴とするフラッシュＲＯＭ管理方法。
【請求項４】
前記フラッシュＲＯＭへのアクセスに基づいて不要となった記憶ブロックを検出する検出工程を更に備え、
前記廃棄工程は、前記検出工程で検出された記憶ブロックの状態情報を、該記憶ブロックのデータが無効であることを示す情報に更新することを特徴とする請求項３に記載のフラッシュＲＯＭ管理方法。
【発明の詳細な説明】
【０００１】
【発明の属する技術分野】
本発明は、コンピュータ等におけるフラッシュＲＯＭの管理方法及び装置及びコンピュータ制御装置に関する。
【０００２】
【従来の技術】
フラッシュＲＯＭは現在いろいろなタイプのものがあるが大きく分けてフラッシュＤＩＳＫ用に開発されたタイプとパーソナルコンピュータのＢＩＯＳ用に開発されたものがある。
【０００３】
前者は消去単位がハードディスクで一般的な５１２バイトであり、ファイルシステムとの整合性が非常に良い。後者のフラッシュＲＯＭは消去単位が例えば６４Ｋなどといった大きなブロック単位でしか行えない様になっている。また、ＰＲＯＭの様に書き込み電圧として１２Ｖ等の電圧が必要なものもある。後者のフラッシュＲＯＭの方が安価に入手できるがファイルシステムとの整合性が悪い為に特に小容量の記録メディアとしては使う事が出来なかった。
【０００４】
【発明が解決しようとする課題】
以上のように、ＢＩＯＳ用に設計されたフラッシュＲＯＭは、その消去単位が大きく、ファイルシステムとの整合性が悪いが、安価で入手しやすい。従って、そのようなフラッシュＲＯＭをファイルシステムに適用できれば、安価な小容量の記録メディアを提供することができる。
【０００５】
本発明は、消去単位の大きいフラッシュＲＯＭをファイルシステムと適合させることを可能とするフラッシュＲＯＭ管理方法及び装置を提供するものである。
【０００６】
ファイルシステムにより、フラッシュＲＯＭに記憶されているデータの更新を行う場合、当該データ領域へ新たなデータを上書きするか、当該データ領域のデータを無効化して他のデータ領域へデータを書き込むことが考えられる。フラッシュＲＯＭでは、データの上書きができないので、他のデータ領域へのデータ書込みが行われ、この結果、無効データ

が蓄積され、フラッシュＲＯＭの利用効率が低下する。よって、ファイルシステムと適合したフラッシュＲＯＭのアクセスを管理する場合、フラッシュＲＯＭ上の無効データを消去することが不可欠である。

【０００７】

以上のように、フラッシュＲＯＭをファイルシステムに適合させるための記憶管理において、フラッシュＲＯＭ上に格納されているデータが無効データであるか否かを判断することは重要である。

【０００８】

しかしながら、一般のファイルシステムにおいては、ファイルの消去等によりデータが無効となったセクタを積極的に解放するという処理は行われない。フラッシュＲＯＭの管理においては、無効データを消去することで記憶効率を維持するため、ファイルシステム上で無効となったデータをそのまま放置しておくことは好ましくない。

【０００９】

本発明は上記の問題に鑑みてなされたものであり、フラッシュＲＯＭをファイルシステムに適応させることが可能な管理方式において、セクタに相当する記憶ブロックの開放を可能とし、無効となったデータを効率良く消去することを可能とするフラッシュＲＯＭ管理方法及び装置及びコンピュータ制御装置を提供することを目的とする。

【００１０】

また、本発明の他の目的は、ファイルシステムによるフラッシュＲＯＭへのアクセス内容に基づいてデータを無効化すべき記憶ブロックを検出し、当該記憶ブロックのデータを廃棄（無効化）することを可能とすることにある。

【００１１】

【課題を解決するための手段】

上記の目的を達成するための本発明のフラッシュＲＯＭ管理装置は以下の構成を備える。即ち、

データ領域、及びデータ領域に対応する管理領域とで構成される複数の記憶ブロックをフラッシュＲＯＭに形成し、各記憶ブロックにおいてデータ領域の記憶状態を示す状態情報を管理領域に格納し、該状態情報に基づいてフラッシュＲＯＭのアクセスを管理する管理手段と、

消去が指定されたファイルブロックに含まれる記憶ブロックについて、該記憶ブロックの管理領域に格納されている状態情報を、当該記憶ブロック内のデータが無効であることを示す情報に更新することにより当該記憶を廃棄する廃棄手段と、

ファイルのディレクトリエントリテーブルにおいて、前記消去が指定されたファイルのエントリを、最後に格納されているエントリの内容で上書きし、当該最後に格納されているエントリに最後を表わす情報を上書きすることにより、前記指定されたファイルを、ファイルシステムが有する消去ファイル復元機能によって誤復元されることを防止する防止手段とを備える。

【００１２】

また、好ましくは、前記フラッシュＲＯＭへのアクセスに基づいて不要となった記憶ブロックを検出する検出手段を更に備え、前記廃棄手段は、前記検出手段で検出された記憶ブロックの状態情報を、該記憶ブロックのデータが無効であることを示す情報に更新する。廃棄すべき記憶ブロックを自動的に検出し、これを廃棄することが可能となるからである。

【００１４】

【発明の実施の形態】

以下に添付の図面を参照して本発明の実施の形態を説明する。

【００１５】

［実施形態１］

＜カメラシステムの構成＞

図１は実施形態１におけるカメラシステムの構成を表すブロック図である。本カメラシス

テムは、電子カメラと、これに着脱可能な外部記憶媒体１７、ＰＣ通信インターフェース１９、及びＰＣ通信インターフェース１９を介して電子カメラと通信可能に接続されたパーソナルコンピュータ２２から構成される。

【００１６】

１はレンズであり、２はレンズ１を通った光を電気信号として出力するＣＣＤユニットである。３はＡ／Ｄコンバータであり、ＣＣＤユニット２からのアナログ信号をデジタル信号へ変換する。４はＳＳＧユニットであり、ＣＣＤユニット２とＡ／Ｄコンバータ３に同期信号を供給する。５はＣＰＵであり、本カメラシステムにおける各種の制御を実現する。

【００１７】

６は信号処理アクセラレータであり、信号処理を高速に実現する。７は電池であり、８は、電池７よりの電力を電子カメラ全体へ供給するためのＤＣ／ＤＣコンバータである。９は電源コントローラユニットであり、ＤＣ／ＤＣコンバータ８をコントロールする。１０はパネル操作・表示装置・電源のコントロールを行うマイクロコンピュータである。１１はユーザへ各種の情報を表示する表示装置であり、液晶パネル等が用いられる。１２はコントロールパネルであり、ユーザが直接操作するレリーズスイッチを含む。

【００１８】

１３はＲＯＭであり、ＯＳ等のシステムプログラムを格納する。１４はＤＲＡＭであり、本電子カメラの主記憶である。１５はフラッシュＲＯＭであり、内蔵記憶媒体として使用する。１６はＰＣＭＣＩＡカードのインタフェース部、１７はＡＴＡハードディスクなどの外部記憶媒体、１８は拡張バスインタフェースである。１９はＰＣ通信インタフェースであり、パーソナルコンピュータ等を接続してデータの授受を行う。２０はＤＭＡコントローラ、２１はストロボである。また、２２はパーソナルコンピュータであり、ＰＣ通信インターフェース１９を介して、電子カメラとの通信を行う。

【００１９】

＜撮影動作＞

この電子カメラの撮影時の動作を簡単に説明する。コントロールパネル１２のレリーズスイッチをユーザが押すと、ＣＰＵ５がそのことを検出して撮影シーケンスを開始する。以下の動作は全てＣＰＵ５によるコントロールで行われることを前提とする。

【００２０】

さて、レリーズスイッチの押下により、ＳＳＧ４がＣＣＤ２を駆動する。ＣＣＤ２から出力されるアナログ信号は、Ａ／Ｄコンバータ３でデジタル信号へ変換される。Ａ／Ｄコンバータ３の出力は、ＤＭＡコントローラ２０によってＤＲＡＭ１４へＤＭＡ転送される。１フレーム分のＤＭＡ転送が終了した時点でＣＰＵ５は、信号処理シーケンスを開始する。

【００２１】

信号処理シーケンスでは、フラッシュＲＯＭ１５から信号処理プログラムを主記憶（ＤＲＡＭ１４）上に読み出し、主記憶上のデータを信号処理アクセラレータ６へ転送し信号処理を行う。但し、信号処理アクセラレータ６は信号処理の全てを行うわけではなく、ＣＰＵ５で行う処理の特に時間のかかる処理などを助ける演算回路であり、ＣＰＵ５の処理ソフトウェアと連携して動作する。信号処理の一部または全部が終了すると画像ファイルとしてフラッシュＲＯＭ１５へ記録する。この時記録するファイルフォーマットが圧縮処理を必用とするのであれば圧縮も行う。

【００２２】

信号処理プログラムは、フラッシュＲＯＭ１５の中でファイルシステムが管理するファイルの１つである。カメラのプログラムはＯＳやファイルシステムといっしょにＲＯＭ１３に納められている。カメラのプログラムは、特定のファイル名のファイルをプログラムであると認識する。

【００２３】

フラッシュＲＯＭ１５の中でファイルは不連続に配置されている上に、本実施形態のファ

イルシステムが頻繁に再配置を行うため、、フラッシュＲＯＭ１５内の制御プログラムををＣＰＵが直接実行することはできない。従って、主記憶（ＤＲＡＭ１４）に読み出して実行させなければならない。更に、主記憶はメモリマネージャが動的に記憶場所をアロケーションするため、、特定のアドレスに格納されることを想定したソフトウエアであってはならない。そのため、本実施形態で信号処理を行うプログラムのファイルは図４１のような形式となっている。

【００２４】

図４１は本実施形態におけるフラッシュＲＯＭへの制御プログラムの格納状態を説明する図である。図４１において、識別コードはファイルがプログラムであることを確認するためのコードである。ファイルは可変長のレコードの集合として表現されている。レコードには、始めに当該レコードに格納されている情報の種類を識別するＩＤがあり、次にそのレコードの大きさを示す値が格納されている。

【００２５】

そして、プログラムのレコードとリロケーション情報のレコードがファイルに格納されている。プログラムコードは例えば図４２のようなデータである。図４２は相対アドレスで表現されたプログラムコードの一例を表す図である。図４２では、００５０番地にジャンプ命令があるが、ＣＰＵはこの命令を絶対番地へのジャンプ命令と認識する。この命令のオペランドは相対アドレスで表現されている。

【００２６】

図４２のリロケーション情報レコードのデータは図４３のような形式で納められている。即ち、図４２のプログラムの中で、絶対番地へ変換しなければならないデータ（相対アドレス表現になっているデータ）のプログラム番地を示すアドレステーブルがリロケーション情報として格納される。

【００２７】

図４１のファイルを主記憶にロードするための領域を確保すると、ＲＯＭ１３のＯＳのメモリマネージャがアドレスを決定する。メモリマネージャのアロケーションはＣ言語ではalloc関数に相当する機能である。メモリマネージャがプログラム用に８７１０番地を割り当てた場合図４４のようにプログラムがロードされる。図４４は図４１のプログラムを主記憶の８７１０番地へマッピングした場合のプログラムコードを示す図である。ジャンプ命令のオペランドが実際の絶対番地に置き換えられている。この実アドレスへの変換をしながらプログラムを主記憶へ読み出すと言う作業を行うプログラムはＲＯＭ１３に格納されている。

【００２８】

以上のように構成することにより、記憶媒体に信号処理ソフトウエアや圧縮ソフトウエアをファイル形式で格納することができる。その結果、カメラが最終ユーザの元へ届いてから、新しい信号処理アルゴリズムや、Ｗｉｎｄｏｉｗｓ（商標）のBMP形式やTIFF形式、あるいは将来新たに登場する形式等、多種多様なファイル形式への対応が可能となる。

【００２９】

以上の様に、本実施形態１における電子カメラは、撮影画像をフラッシュＲＯＭ１５へファイルするものである。

【００３０】

＜デバイスドライバインタフェース＞

図２は、本実施形態の電子カメラにおけるファイルシステムの階層構造を表す図である。最上位の層がユーザアプリケーション１０１である。ユーザアプリケーション１０１は電子カメラの内部で動くソフトウェアであり、ファイルをファイル名でオープンして読み書きした後クローズする。

【００３１】

ユーザアプリケーション１０１から直接ファンクションコールによって呼び出されるのがファイルシステムＡＰＩ層１０２である。このファイルシステムＡＰＩ層１０２がドライブ名とファイルシステムを関連付けて管理している。各ドライブ毎にファイルシステムア

ーキテクチャ層をマウントする用に構成しているため、複数のファイルシステムアーキテクチャを混在させる事が可能となっている。

【００３２】

ファイルシステムアーキテクチャ層１０３が実際のファイル管理を行う部分である。最下位の層がブロックデバイス層１０４である。ファイルシステムアーキテクチャ層１０３がブロックデバイス層１０４の提供するサービスを利用してファイル入出力を実現している。このブロックデバイス層１０４では、データをセクタという単位で管理しており、１セクタは例えば５１２バイトである。このブロックデバイス層１０４でデバイスごとの入出力制御の違いと、ヘッドやシリンダなどパラメータの違いを吸収している。このように構成しているため、同時に複数の種類のデバイスを混在させることができる。

【００３３】

本実施形態の電子カメラでは、特にブロックデバイス層１０４におけるフラッシュＲＯＭの記憶管理方法に特徴を有する。

【００３４】

図１で示したフラッシュＲＯＭ１１には、現在いろいろなタイプのものがあるが、大きく分けてフラッシュＤＩＳＫ用に開発されたタイプとパーソナルコンピュータのＢＩＯＳ用に開発されたタイプがある。前者は消去単位がハードディスクで一般的な５１２バイトであり、ファイルシステムとの整合性が非常に良い。後者のフラッシュＲＯＭは消去単位が例えば６４ｋなどといった大きなブロック単位でしか行えない様になっている。また、ＰＲＯＭの様に書き込み電圧として１２Ｖ等の電圧が必要なものもある。しかしながら、後者のタイプのフラッシュＲＯＭは安価で入手が容易である。本実施形態では、後者の様な特徴を持つフラッシュＲＯＭでありながらファイルシステムに対してハードディスク同様のサービスを提供する。

【００３５】

＜フラッシュＲＯＭドライバインタフェース＞

一般的にブロックデバイスがファイルシステムへ提供するサービスは以下の２つである。即ち、

（１）ロジカルセクタナンバーで指定したセクタからの読み出し

（２）ロジカルセクタナンバーで指定したセクタからの書き込み

である。そして、これに加えて

（３）ロジカルセクタナンバーで指定したセクタの開放

の機能があれば、フラッシュＲＯＭのドライバは必要に応じて不要なセクタを消去することが可能となるため、効率良くフラッシュＲＯＭを消去することができる。

【００３６】

（３）に挙げた機能は、通常のＤＩＳＫでは必要のない機能だが、キャッシュを持ったシステムだと、積極的にキャッシュリストから削除できるので、結果的にキャッシュのヒット率を上げる効果がある。ファイルシステムは、ファイルの消去等で不必要となったセクタを（３）の機能を用いてデバイスドライバへ通知する。フラッシュＲＯＭの消去は非常に時間がかかる処理だが、ＣＰＵ時間をほとんど消費しないためバックグランド処理で行うのが良い。

【００３７】

後述の＜ＦＡＴキャッシュ＞においても説明するが、本実施形態のキャッシュは、新しいデータ（キャッシュ上に無いデータ）をアクセスする場合にキャッシュリストの中で最も古いデータを廃棄する。不要セクタをキャッシュリストの最後へ移動させる（即ち、最も古くアクセスしたデータがキャッシュの後ろへ移動する）ことで有効なデータがキャッシュから廃棄される可能性が低くなる。特にコンパイラー等の中間ファイルを多く生成するシステムでは、消去すべき中間ファイルがキャッシュに残っている可能性が高く、上記のキャッシュ管理はヒット率の向上に非常に有効である。

【００３８】

図３は、デバイスドライバの管理ブロックをＣ言語で記述した宣言文を示す図である。構

造体のＮｅｘｔは、次のデバイスへのリングポインタであり、メモリ中のデバイスを検索する目的で使用される。ＤｅｖＮａｍｅは、デバイスの名前として使用される。ＩｎｉｔＤｅｖは、デバイスの初期化ルーチンへのポインタである。ＳｈｕｔＤｏｗｎは、デバイスのシャットダウンルーチンへのポインタである。ＲｅａｄＳｅｃｔｏｒは、ロジカルセクタを指定して媒体の内容をバッファへ転送するルーチンへのポインタである。ＷｒｉｔｅＳｅｃｔｏｒは、ロジカルセクタを指定してバッファの内容を媒体へ転送する（書き込む）プログラムへのポインタである。ＲｅｌｅａｓｅＳｅｃｔｏｒはロジカルセクタを指定して、セクタを解放するルーチンへのポインタである。

【００３９】

ファイルシステムは、この構造体を仲介してデバイスドライバを利用することになる。固定ディスクやフロッピーディスクの場合、ＲｅｌｅａｓｅＳｅｃｔｏｒには何も仕事をしないプログラムへのポインタが代入されている。または、ディスクキャッシュのキャッシュリストから指定セクタを削除するポインタでもよい。

【００４０】

＜フラッシュＲＯＭ管理方法＞

フラッシュＲＯＭに対するデータ書き込みは、上位層のファイルシステムからセクタ単位で行われる。図４は、フラッシュＲＯＭ上のセクタ構造の例を示す図である。図４において、１５１はイレースブロックである。このイレースブロック１５１は消去の単位であり、フラッシュＲＯＭの技術用語ではセクタと呼ばれるものである。しかしながら、ファイルシステムが扱う単位である“論理セクタ”と区別する為に、ここではイレースブロックと呼ぶことにする。

【００４１】

図４によれば、システム中に複数のフラッシュＲＯＭ１５が搭載されていて、各フラッシュＲＯＭ１５は複数のイレースブロック１５１によって構成される。更に、各イレースブロック１５１は消去回数カウンタ１５２と複数のセクタ１５３によって構成されている様子をあらわしている。消去回数カウンタ１５２は、イレースブロック１５１を消去した回数をカウントする為に用いる部分である。各セクタ１５３は、管理領域とデータ領域とを有する。管理領域は、論理セクタ番号を表すセクタ番号１５４と、セクタが有効利用されているかどうかを表わす使用中フラグ１５５と、セクタとしての利用が終了したことを表わす使用済みフラグ１５６とで構成される。また、データ領域は、５１２バイトのデータ部１５７によって構成されている。

【００４２】

データ領域と管理領域は、隣り合って配置する必要は無く、図５の様にまとめて管理することも考えられる。図５は、管理領域用データと、データ領域とを分離して格納する構成を表す図である。セクタ番号テーブルには、複数のセクタ１５３の各セクタ番号１５４が格納される。また、フラグテーブルには、使用中フラグ１５５、使用済みフラグ１５６が格納される。更に、データテーブルには、データ部１５７の内容が格納される。以上のようなデータ構成をとることも可能であるが、少なくとも管理領域と、これに対応するデータ領域とは同じイレースブロック内に納めるのが好ましい。

【００４３】

なお、システムは「使用中フラグ１５５」より「使用済みフラグ１５６」の方を優先的に評価する。図６は、各フラグの状態に対応した意味を示す図である。図中、ＦＡＬＳＥは、消去後の状態と同じ値をとる。使用中フラグ１５５がＴＲＵＥであっても、使用済みフラグ１５６がＴＲＵＥであれば、当該セクタのデータは無効である。

【００４４】

＜フラッシュＲＯＭの論理セクタ書き換え＞

フラッシュＲＯＭはＰＲＯＭ同様、データを書き換える為に、一度消去してから再書き込みをしなければならない。しかも、消去の最少単位が大きく（例えば６４ｋバイト）消去時間が長い（例えば１秒）。そこで上位層のファイルシステムが、特定のセクタを書き換えようとした場合、消去済みの領域へ論理セクタを移動させることで、消去動作をせずに

、見かけ上で論理セクタのデータ書き換えを実現する。

【００４５】

図７はセクタの書き換え手順を説明する図である。同図を用いて、８番セクタ（論理セクタ番号が８のセクタ）の書き換えを例にして詳しく説明する。図７中、左側が書き換え前の状態であり（（ａ）の状態）、右側が書き換え後の状態（（ｂ）の状態）である。また、図７において、管理領域中の数字は論理セクタ番号を表し、（使用中）は使用中フラグ１５５がＴＲＵＥで使用済みフラグ１５６がＦＡＬＳＥの状態、（使用済）は使用中フラグ１５５と使用済みフラグ１５６が共にＴＲＵＥの状態を示す。

【００４６】

“セクタ番号８（使用中）”の場所に、８番セクタのデータが格納されている。今、８番セクタがＦＡＴやファイルの一部として利用されていて、その内容を変更したい場合に上位層から８番セクタの書き換え要求が発生したとする。書き換え要求が発生すると、フラッシュＲＯＭのデバイスドライバは、フラッシュＲＯＭの未使用セクタを検索し、その場所を新たな８番セクタの場所としてセクタ番号と更新後のデータを格納し、使用中フラグをＴＲＵＥにする。次に、以前８番セクタだったセクタの使用済みフラグをＴＲＵＥにする。このような手順で８番セクタのデータの書き換えが実現される。

【００４７】

＜ガベージコレクション＞

以上の様な方法で論理セクタの書き換えを実行していくと、いずれフラッシュＲＯＭのほとんどの領域を“使用済セクタ”にしてしまうことになる。そこであるタイミングでフラッシュＲＯＭを一旦消去して“使用済セクタ”を“未使用セクタ”へ戻す必要がある。基本的なガベージコレクションの動作を図８を用いて説明する。図８は、本実施形態におけるフラッシュＲＯＭのガベージコレクション動作を説明する図である。

【００４８】

図中（Ａ）は、ガベージコレクション前の状態である。説明を簡単にするために、本例のフラッシュＲＯＭはセクタ６個分の大きさのイレースブロックで構成されているものとする。イレースブロック（１）には使用済セクタが３個と使用中セクタが３個あり、消去回数は５回である（消去回数カウンタ１５２の内容が５である）。イレースブロック（２）には使用中セクタが１個、使用済セクタが１個、未使用セクタが４個あり、消去回数は９回である。この状態からガベージコレクションを開始する。

【００４９】

先ず、調整対象イレースブロックを選定する。ここで調整対象イレースブロックは消去を行う対象としてイレースブロックとなる。整理対象イレースブロックの選定は、使用済セクタをたくさん含むイレースブロックから優先的に選択すると整理効率が良い。しかし使用済セクタを含まない場合を別として消去回数の少ないイレースブロックを優先的に整理対象とする方法を取ればチップ内のイレースブロックを平均的に使用することができ、書き換え耐久を分散させることができる。選択手順の詳細についてはフローチャートを用いて後述する。

【００５０】

今、イレースブロック（１）が整理対象として選定されたとする。次に、整理対照であるイレースブロック（１）の使用中セクタ（使用中フラグがＴＲＵＥで、使用済みフラグがＦＡＬＳＥのセクタ）を他のイレースブロックに移動させる。使用中セクタの移動手順は、セクタの書き換え時と同様に、他のイレースブロック中の未使用セクタを検索して、使用中セクタ内のデータ領域と管理領域の内容をコピーし、移動元の使用中セクタの使用済みフラグをＴＲＵＥにする。なお、未使用セクタが無い場合の処理は、後で述べる。

【００５１】

図８の（Ｂ）は、イレースブロック（１）の使用中セクタをすべてイレースブロック（２）へ移動させた状態である。この結果、イレースブロック（１）には、使用済セクタしか存在しないことになる。

【００５２】

次に、使用済セクタだけで構成されているイレースブロックを検索する。ここで、検索を行うのは、通常の書き換え動作の際に偶然イレースブロック内のセクタが全て使用済セクタとなっている場合があるからである。続いて検索されたイレースブロックに対して消去を行う。消去には時間がかかるが、複数のイレースブロックを同時に消去できるため、できるだけ一度に複数のイレースブロックを消去するのが良い。消去が終了すると消去回数カウンタへ消去前の値＋１したものを書く。これでガベージコレクション完了である。

【００５３】

図８の（Ｃ）がガベージコレクション終了時の状態である。イレースブロックをできるだけ同時に消去した方が効率が良いため、使用済セクタと未使用セクタがある限りたくさんのイレースブロックを同時に調整すると良い。極端に消去回数カウンタの値が他のイレースブロックより少ないものがあれば、使用済セクタを含んでいなくても整理さえすれば、書き換え耐久の分散を図れる。また、一度整理するとデータの配列が変わる為、書き換え耐久分散のきっかけとなる。

【００５４】

＜未使用セクタがない場合＞

次にシステム中に使用済セクタが有るにもかかわらず未使用セクタが全く無くなってしまった場合のガベージコレクション手順を図９を使って説明する。図９は、未使用セクタが存在しない場合のガベージコレクションの動作を説明する図である。

【００５５】

先ず、上述した基本的なガベージコレクション手順に従い、イレースブロック（１）を整理対象として選択する。次にイレースブロック（１）の使用中セクタを移動する為の未使用セクタを検索する。未使用セクタがある場合は、上述の基本的ガベージコレクションと同様にセクタの移動を行う。

【００５６】

一方、検索の結果、未使用セクタが無ければ、ＤＲＡＭ１４のヒープエリアからデータの退避に必要な大きさのメモリブロックをアロケーションする。そして調整対象イレースブロック内の使用中セクタをＤＲＡＭ１４へコピーする。この場合は、フラッシュＲＯＭの別の領域へセクタを移動する場合と違い、元のセクタの使用済フラグをＴＲＵＥにしない。なぜならこの時点で電子カメラの電池７が外れるなどの事故が起こった場合に、ＤＲＡＭ内のデータが消滅してしまい、データの修復ができなくなるからである。図９の（Ｂ）は、ＤＲＡＭ１４の領域へコピーされたセクタを表現している。調整対象のイレースブロックの使用中セクタをすべて退避出来たら、その調整対象のイレースブロックを選択して消去する（図９の（Ｃ）参照）。消去が終わった後は未使用セクタがたくさん出来ているはずである。よって、次に未使用領域を検索してＤＲＡＭ１４へ待避してあったデータを復元する。図９の（Ｄ）は、ガベージコレクションが完了した状態を示している。

【００５７】

上記の手順でガベージコレクションをした場合でも、イレースブロックを消去してからデータを復元するまでの間に電子カメラの電池７が外れるなどの事故が起こったらデータの修復をすることはできない。つまり、セクタのデータをＤＲＡＭ１４に退避する方法はできる限り取らない方が、よりシステムの安全性を保つことができる。一回、ＤＲＡＭを使ってガベージコレクションを行えば未使用セクタができる。したがって、１度ＤＲＡＭ１４を使ったガベージコレクションを行い、その後に通常のガベージコレクションを行えば、消去時間は余分にかかるが安全性を高めることができる。逆にＤＲＡＭ１４への退避を積極的に行う（例えばヒープ領域がある限り退避する）と、同時に整理できるイレースブロックが増える為効率を上げることができる。従って、安全性と効率のどちらを優先するかを指定できるように構成してもよい。またシステムの電源が電池７より供給されている場合は安全性優先、ＡＣアダプタから供給されている場合は効率優先に自動的に切り替わるように構成してもよい。本処理については図３６を参照して後述する。

【００５８】

＜イレースブロックを余分に１つ用意＞

残り容量が極端に少なくなるとガベージコレクションが多発してシステムのパフォーマンスが極端に落ちる。総論理セクタ分を格納できるイレースブロック数よりも１つだけ余分にイレースブロックを使用すれば、そのような事態を避けることが可能である。仮に、１イレースブロックあたり１２７セクタ格納できるとして、全てのセクタが使用中となった場合、同じ１セクタを１０回書換える場合を例にすると、余分イレースブロックがなければ、１０回の消去と１２７０セクタの書き込みが発生する。しかし、イレースブロックを余分に１つ用意しておけば、１０セクタの書き込みしか発生しない。

【００５９】

よって、本実施形態では、イレースブロックの数はチップの構成で決まるので、最低１つのイレースブロックが余る様な総論理セクタ数を設計する。

【００６０】

＜ガベージコレクションのタイミング＞

ガベージコレクションは消去動作を伴うために非常に時間がかかる。そのためガベージコレクションをいつ行うかによってカメラの使い勝ってを左右することとなる。例えば、セルフタイマなどの数秒間撮影しなくても良い時にガベージコレクションを行えばユーザがストレスを感じることがない。

【００６１】

＜ＲＡＭ上の記憶場所管理＞

フラッシュＲＯＭ１５上ではセクタ番号と実際の記憶場所が関連していないために特定のセクタを読み書きする為にフラッシュＲＯＭ１５を検索しなければならない。そこでシステムがリブートする際に、フラッシュＲＯＭ１５における各セクタの格納アドレスを示す記憶場所管理テーブルをＤＲＡＭ１１４上に作成しておくと、フラッシュＲＯＭ１５に対して高速なデータの読み書きを実現できる。一度、記憶場所管理テーブルを作成すれば、フラッシュＲＯＭ１５に対するセクタの書き込みやガベージコレクションによって記憶場所に変更が生じた場合に限って記憶場所管理テーブルの記憶位置を更新するだけで常に正しい記憶場所管理テーブルを維持することが可能である。

【００６２】

図１０は、ＤＲＡＭ上に作成された記憶場所管理テーブルを説明する図である。図中、右側にＲＡＭ上の作成した記憶場所管理テーブル１４０を示した。０セクタと４セクタは記憶場所不在を意味する値（NULL）が入っている。これらは、フォーマット後そのセクタに対する書き込みが全く無かったか、もしくは、ファイルシステムが開放したセクタである。

【００６３】

ファイルシステムがファイルの消去などで不要となったセクタを解放する命令をドライバに出した場合のデバイスドライバの動作は次のようになる。まず、ＤＲＡＭ１４上の記憶場所管理テーブル１４０の指定されたセクタのポインタを参照してフラッシュＲＯＭ１５上の現在使用中の該当するセクタを探し出す。そして、当該セクタの使用済フラグをＴＲＵＥにし、ＤＲＡＭ１４上の記憶場所管理テーブル１４０の指定セクタのポインタへ不在値（NULL）を代入する。

【００６４】

なお、ガベージコレクションの為にセクタの内容をＤＲＡＭ１４へ待避している場合は、記憶場所としてＤＲＡＭへのポインタが代入されている。また、同一論理セクタに対する同時操作を禁止する為のロック変数もテーブルに納めることが望ましい。

【００６５】

＜ＭＳ－ＤＯＳのファイル復元＞

本電子カメラとパーソナルコンピュータ２２で、記憶媒体上のデータ交換が出来ると都合が良い。本実施形態で説明したフラッシュＲＯＭ管理方式を使用して、現在パーソナルコンピュータで普及しているＭＳ－ＤＯＳ（商標）と互換性があるファイルシステムを実装することができる。ＭＳ－ＤＯＳには一度消去したファイルを復元するユーティリティが付属している。ところが、本実施形態ではフラッシュＲＯＭの消去効率を向上させる為に

、消去したセクタのデータ部を失ってしまう様な構成となっている。カメラで消去した媒体をパーソナルコンピュータで復元する事が原理的にできない構成になっているのである。

【００６６】

パーソナルコンピュータでのファイル復元機能を禁止できれば、このような事故を防ぐことができる。本実施形態では、ＭＳ－ＤＯＳがファイル復元の時に使用するデータを破壊することでファイル復元機能を禁止する。これをいかに説明する。

【００６７】

ＭＳ－ＤＯＳ（商標）で、ファイルを消去するとディレクトリに空きスロットができる。ディレクトリにはファイル名／タイムスタンプ／最初のクラスタなどの情報が格納されている。図４５はディレクトリスロットの特徴を表す図である。ディレクトリスロットの最後には、リストの最後であることを示すEndOfDirが格納されている。

【００６８】

今、File Bを削除すると、ファイル名の先頭が削除を表すシンボルに置き換えられ、ＦＡＴのクラスタチェーンが消去される。この様子を図４６に示す。

【００６９】

アンデリートプログラムは、２番目のスロットに残った情報を元に、ファイルの復元を試みる。逆にこの情報がなければ、ファイルの復元を防止できる。

【００７０】

図４７は本実施形態のＤＯＳ互換ファイルシステムでファイルを消去した後の状態を表している。本実施形態では、ディレクトリエントリテーブルの最後に格納されているファイルを消去したいファイルのエントリに上書きし、ディレクトリエントリテーブルの最後のファイルだった部分にEndOfDirを上書きするように構成する。こうすることにより、ファイル復元機能によるファイルの復元を防止できる。

【００７１】

なお、ファイルの消去時にはＭＳ－ＤＯＳと同様にセクタのデータをそのまま残しておき（セクタの開放を行わない）、ガベージコレクション時にまとめてＦＡＴとデータの関係を参照してしながら不要部分を消去する方法もある。

【００７２】

＜バックグランドで前処理＞

あるフラッシュＲＯＭでは、消去前のデータが“０”になっている方が高速に消去処理できる。フラッシュＲＯＭの消去完了の確認は、データ書き込み時と同様にデータポーリングによって行われる。従って、このようなフラッシュＲＯＭを使う場合は、バックグランド処理で“使用済”となったセクタのデータを0に書き換える「前処理」を行うことで性能を向上させることができる。最も低いプライオリティのタスクとして実行するようにしておけば、スループットの低下にはつながらない。

【００７３】

この前処理バックグラウンドでの“前処理済セクタ”管理の為にフラグを用意しておけば前処理の効率を上げることができる。

【００７４】

そのために、セクタのとりうる状態として、「未使用」「使用中」「使用済」に加えて「前処理済」の４つの状態を表示できる管理フラグをフラッシュＲＯＭのセクタ内部へ用意すると効率が良い。

【００７５】

図３８は、本実施形態における消去処理速度向上のための前処理の制御手順を表すフローチャートである。同図において、ステップＳ２５０１にて、使用済でかつ前処理の済んでいないセクタを抽出する。これは、セクタ内の管理フラグが「使用済」となっていて、かつ「前処理済」となっていないセクタを抽出することで実現できる。ステップＳ２５０２において、抽出されたセクタに対してデータ「0」を上書きを開始する。ステップＳ２５０３では当該セクタについて前処理を終了したか否かを判断する。フラッシュＲＯＭへの

書込みは１バイト単位であるので、１セクタ分のバイト数の書込みが必要となる。当該セクタに対する前処理が終了していなければステップＳ２５０４へ進み、他のタスクへ制御を移す。

【００７６】

上述したように本処理は最もプライオリティの低いタスクで行われるので、ＣＰＵ５がアイドル状態となったときに再び本処理が実行される。この場合処理はステップＳ２５０３へ戻る。この時点で、前回の書込みが終了していなければそのまま他のタスクへ処理を移行する。

【００７７】

以上のようにして当該セクタの全バイトに対して「０」の書込みを終えると、ステップＳ２５０３からステップＳ２５０５へ進み、当該セクタの管理フラグを、「前処理済」を示す状態にセットする。そして、引き続き、他のセクタについて前処理を行うために、ステップＳ２５０１へ戻る。

【００７８】

＜ＦＡＴキャッシュ＞

本システムでは、書き込み発生の度に記憶場所を変更し、その度に「未使用セクタ」が発生する。そこで、使用頻度の多い部分を特に優先的にバッファリングするキャッシュが有ればトータルの書き込み頻度が激減する事が予想される。キャッシュとして用意するメモリは多ければ多いほど良いが、システムのメモリには限界がある。

【００７９】

本来使用頻度の高いセクタのデータは、キャッシュ中に存在する確率も高いが、使用頻度の低いセクタを大量に読み書きした場合、当然キャッシュから吐き出されることになる。

【００８０】

そこで、ファイルシステムが管理する管理領域を優先的にキャッシングする様に構成すれば、スループットの向上を期待できる。なぜならファイルシステムの管理領域は頻繁に更新されているからである。

【００８１】

パーソナルコンピュータで普及しているＭＳ－ＤＯＳのＦＡＴシステムの場合、７２０ｋや１．４Ｍといったフォーマット形式では１クラスタが１セクタで構成されている為、シーケンシャルにファイルを読む場合でも２回に１回はＦＡＴを読まなければならない。ファイルを書く場合は、さらにたくさんのＦＡＴアクセスが発生する。このため、システム中にたくさんのファイルがオープンされるとキャッシュのヒット率が落ちてしまう。

【００８２】

アプリケーションソフトウェアにもよるが、ＦＡＴシステムにおいてＦＡＴのみを対処にしたキャッシュは、ＤＩＳＫ全体を対象にしたキャッシュに対して１／２のメモリで同等のヒット率を確保できる。図１１はキャッシュソフトウエアの階層的な位置付けを表す図である。キャッシュのソフトウェアは図１１の様にファイルシステムとフラッシュＲＯＭの中間的な場所となる。

【００８３】

図１２はキャッシュの主記憶上のデータ構造を表わす図である。片方向線形リスト構造でバッファ全体を管理している。検索方向順にデータが古くなっている。論理セクタ番号が１２，１１，６，５の順番でアクセスすれば、図１２に示されるような順番となる。また、各セクタには、変更フラグが設けられており、キャッシュ上でデータの更新があった場合、変更フラグがＦＡＬＳＥからＴＲＵＥに変化する。このようなＦＡＴキャッシュの読み出し手順、及び書き込み手順を図１３、１４を参照して説明する。図１３はＦＡＴキャッシュの読み出し手順を表すフローチャートである。図１４はＦＡＴキャッシュの書込み手順を表すフローチャートである。

【００８４】

図１３において、ステップＳ１５０１でＮセクタの読み出しを開始する。ステップＳ１５０２でＮセクタがＦＡＴかどうかを判断する。ＦＡＴでなければ、ステップＳ１５０９で

フラッシュＲＯＭ１５からデータを読み出す。

【００８５】

一方、ステップＳ１５０２でＮセクタがＦＡＴならステップＳ１５０３へ進み、キャッシュリストを検索する。ここでは、図１２で説明した片方向線形リストを検索することになる。キャッシュ中にＮセクタが存在すればステップＳ１５０７へ進み、Ｎセクタのバッファからデータを読み出す。

【００８６】

また、ステップＳ１５０３でＮセクタがキャッシュリスト中に存在しなかった場合は、ステップＳ１５０４へ分岐し、最も長くアクセスされていないセクタのデータ（図１２ではセクタ番号１２のデータ）の吐き出しを行う。まず、ステップＳ１５０４では、キャッシュリストの最後の項の変更フラグを判断する。もし変更フラグがＴＲＵＥなら、ステップＳ１５０５へ進み、変更内容をフラッシュＲＯＭ１５へ書き込む。変更が無い（変更フラグがＦＡＬＳＥの場合）なら、そのままステップＳ１５０６へ制御を移す。読み出し手順の中で書き込みを行うのは奇妙に思うかもしれないが、バッファがキャッシュの吐き出しが起こるまで極力書き込み動作を行わない方が効率が良い。

【００８７】

ステップＳ１５０６でフラッシュＲＯＭ１５からリスト最後のバッファへＮセクタの内容を読み出す。ステップＳ１５０７でＮセクタのバッファからデータを読み出す。ステップＳ１５０８でＮセクタのバッファをキャッシュリストの先頭へ移動させる。これは、図１２において、各セクタが有する「次のバッファー」（次のバッファーを示すアドレス）の値を変更することで達成される。ＦＡＴキャッシュへのアクセスが行われる度にステップＳ１５０８の動作が繰り返されることで、自然にアクセスされないバッファがリストの先頭から最後に向かってシフトしていく。よって、ステップＳ１５０４でリスト最後のバッファを選ぶのは、最も古いバッファを吐き出す為である。

【００８８】

次に図１４を参照して書込み手順を説明する。

【００８９】

ステップＳ１６００でＮセクタの書き込みを開始する。ステップＳ１６０１では、ＮセクタがＦＡＴかどうかを判断する。ＦＡＴでなければ、ステップＳ１６０８へ進み、フラッシュＲＯＭ１５へのデータの書き込みを実行する。

【００９０】

一方、ステップＳ１６０１でＮセクタがＦＡＴならば、ステップＳ１６０２へ進み、キャッシュリストを検索する。キャッシュ中にＮセクタが存在すればステップＳ１６０６へ進み、Ｎセクタのバッファへデータの書き込みを実行する。

【００９１】

また、ステップＳ１６０２でＮセクタがキャッシュリスト中に存在しなかった場合は、ステップＳ１６０３へ分岐し、バッファから最も長くアクセスされていないセクタの吐き出しを行うとともに、Ｎセクタをキャッシュに登録する。まず、ステップＳ１６０３でキャッシュリストの最後の項の変更フラグを判断する。もし変更フラグがＴＲＵＥなら、ステップＳ１６０４で変更内容をフラッシュＲＯＭへ書き込み、ステップＳ１６０５へ進む。また、変更が無いなら（変更フラグがＦＡＬＳＥなら）そのままステップＳ１６０５へ制御を移す。ステップＳ１６０５では、キャッシュリストの最後の項をＮセクタとする。その後、ステップＳ１６０６で、Ｎセクタのバッファへデータの書込みを実行する。

【００９２】

その後、ステップＳ１６０７でＮ、セクタのバッファをキャッシュリストの先頭へ移動させる。書き込み手順の中でフラッシュＲＯＭへの書き込みを行わないのは奇妙に思うかもしれないが、キャッシュの吐き出しが起こるまで極力書き込み動作を行わない方が効率が良い。

【００９３】

また、ステップＳ１５０１及びステップＳ１６０１における、ＦＡＴの判断であるが、Ｉ

Ｃカード等の完全に上位層（ファイルシステム）の情報を共有できないシステムでも、書き込みデータの内容を解析することでＦＡＴ領域の場所を特定できる。なぜならば、論理セクタＯに相当する部分にＦＡＴの位置等の情報か格納されていることが決まっているからである。

【００９４】

＜フラッシュＲＯＭへの１バイトの書き込み＞

フラッシュＲＯＭ１５に対する全ての（管理領域を含む）読み書きは、最終的に１バイトの読み書き命令によって実行される。フラッシュＲＯＭ１５の書き込みには、通常のＰＲＯＭ同様の時間がかかる。１バイトの書き込みが終了するまでは、同じチップへの書き込みはできない。書き込み終了信号として信号線が用意されているチップと特別な信号が用意されていないチップがある。後者の場合は、データポーリングと言う手法で書き込み終了を確認しなければならない。データポーリングとは、ベリファイに非常によく似た方法で、書き込みデータと読み出しデータが一致するまで待つビジー制御方法である。

【００９５】

信号線によって書き込み終了を知ることが出来る場合は、ＣＰＵ５への割り込みと併用して書き込み待ち中のＣＰＵタイムを別のタスクへ割り当てることができる。

【００９６】

上述のように、信号線が無いチップの場合は、データポーリングを行なわなければならない。データ書き込みの効率をあげる為にはいくつものチップに対してパイプライン的に書き込みを行い、データポーリング時間のロスを押さえなければならない。そのため、１バイトの書き込みが完了する前に次の動作へ制御を移す必要がある。新たな読み書きを行う前に以前の書き込みが完了しているかどうかを確認するのが良い。図１５は、その様子をＣ言語で表現したものである。

【００９７】

図１５の１行目は、データ書き込みを行う関数の入り口である。最初の引数は最後に書き込んだアドレスとデータを保存するための構造体へのポインタ、第２の引数は書き込むアドレス、第２の引数は書き込むデータである。

【００９８】

３行目では、最後に書き込んだアドレスを参照してチップに書かれたデータと最後に書いたデータを比較して、両者が一致するまでループを実行する。これがデータポーリングである。前回の書き込みが完了するとこのループから抜け出す。

【００９９】

４行目で新しいアドレスへＤａｔａを書き込む。５行目と６行目で、今回書いたアドレスとデータを保存する。この情報は、次のデータポーリングで利用される。

【０１００】

リスト７行目のRotateRdyQueueは、自タスクの次に実行されるべき同一プライオリティの実行可能状態のタスクへＣＰＵを譲るオペレーティングシステムのシステムコールである。

【０１０１】

９行目は読み出し関数の入り口である。第１の引数はアドレスとデータを保存するための構造体へのポインタ、第２の引数は読み出すアドレスである。この関数は上位のプログラムに対して第２の引数で指定されたアドレスに格納されたデータを返す。

【０１０２】

１１行目では、もし読み出そうとしたアドレスが最後に書き込んだアドレスなら戻す値は最後に書き込んだデータなので構造体の中に保存された情報を返す。１２行目は３行目と同じようなデータポーリングである。データポーリングに成功しないと同じチップの別のアドレスを読むことができない。データポーリングが終わって１３行目で指定したアドレスの内容を戻している。

【０１０３】

１チップへの書き込みを以上の様な構成にしておけば、チップ数と書き込みタスクを増や

すだけで確実に見かけ上の書き込み速度を向上させることができる。また、全体のスループットを上げる為にわざとチップ数分のセクタバッファを用意（２チップなら２セクタ）して書き込む内容がバッファに溜まるまで処理しないようにすると効果がある。

【０１０４】

図１５のプログラムの特徴的なところは、データポーリングをデータ書き込み直後に行うのではなく、次の書き込みの前に行うことである。そのために前回書いたアドレスとデータを保存しておくＲＡＭ領域をチップごとに確保し構造体「struct DEV」として格納しているのである。

【０１０５】

図３９は、本実施形態におけるフラッシュＲＯＭへの１バイトデータの書込み手順を表すフローチャートである。本フローチャートは、１つのフラッシュＲＯＭチップへの書込みの制御手順を示している。ステップＳ２６０１では、前回の書込み処理が完了したか否かを判断する。前回の書込み処理が終了していなければステップＳ２６０４へ進み、そのまま他のタスクへ制御を移す。

【０１０６】

一方、前回の書込み処理が終了していれば、次の書込みデータを準備し、これをＤＲＡＭ１４へ保存する。上述のステップＳ２６０１における書込み終了の判断は、フラッシュＲＯＭに書き込まれたデータと、このステップＳ２６０２で保持されたデータとの比較によって行われる。

【０１０７】

続いて、ステップＳ２６０３において、データの書込みを開始する。以上のような処理によれば、複数のフラッシュＲＯＭチップに対して、複数のタスクで書き込みを行うような場合に、いわゆるラウンドロビン方式を適用した書き込み処理が可能となり、複数のＲＯＭチップに対して効率良くデータの書き込みが行える。なお、マルチタスクの管理プログラムは、上述のＲＯＭ１３に格納されている。組み込みようのリアルタイムＯＳとしては、VxWorks（商標）やpSOS（商標）等が市販されており、ＲＯＭ１３にこれらのようなリアルタイムＯＳが格納されている。

【０１０８】

＜フラッシュＲＯＭ書き込み電源の共有化＞

データの書き込みや消去の際にＰＲＯＭ同様に１２Ｖ等の特別な書き込み電圧を必要とするチップや、書き込み電圧を与えることで書き込みが高速になるチップがある。この様なチップを使用する場合に専用のＤＣ／ＤＣコンバータ等の電圧発生部を設けると電子カメラのコストアップにつながる。ところが、従来よりカメラにはストロボの充電や、機構部分やＣＣＤの駆動等、特別な電圧が必要な部分がありＤＣ／ＤＣコンバータ等を搭載している。そこで、フラッシュＲＯＭの書き込み電圧とストロボ充電やメカ駆動を時分割多重で行うことで、少容量のＤＣ／ＤＣコンバータでシステムを構築でき、システムのコストを押さえることができる。

【０１０９】

図１６は、ＤＣ／ＤＣコンバータの出力容量を越えない様に電源を管理するプログラムをＣ言語で表現したものである。Ｌｉｎｅ１～６が１ステップのズームアップ関数で、Ｌｉｎｅ７～１３がフラッシュＲＯＭへ１セクタ書き込む書き込み関数である。ズームアップ関数はＬｉｎｅ３でＤＣ／ＤＣコンバータの資源管理用のセマフォ“ＳｅｍＤＣＤＣ”を獲得して、モータを１ステップ動かす関数をＬｉｎｅ４で呼び出す。モータ駆動が終わるとＤＣ／ＤＣコンバータの資源管理用のセマフォ“ＳｅｍＤＣＤＣ”を開放する。セマフォはマルチタスクのオペレーティングシステムで資源を管理する為の一般的な方法であり、多くのオペレーティングシステムがシステムコールとして用意している。

【０１１０】

即ち、Ｌｉｎｅ３で既に“ＳｅｍＤＣＤＣ”が他のタスクによって使用されていたとすると、他のタスクがセマフォ“ＳｅｍＤＣＤＣ”を開放するまでズームアップをしようとしたタスクの実行が保留される。

【０１１１】
書き込み関数はＬｉｎｅ９でセマフォ“ＳｅｍＤＣＤＣ”を獲得し、フラッシュＲＯＭへ１セクタのデータを書き込む。Ｌｉｎｅ１１でデータポーリングを行い最後の書き込みが終了したことを確認したら、Ｌｉｎｅ１２でセマフォ“ＳｅｍＤＣＤＣ”を開放する。このようにプログラムを構成すれば、ズームアップとフラッシュＲＯＭの書き込みを同時に行うことは無くなる。ズームは１ステップ単位であり、書き込みはセクタ単位なので非常に短い保留時間の後に必ず電源を獲得できる。
【０１１２】
図１６について更に説明すると、図１６のＬｉｎｅ１はズームアップする関数の入り口である。本関数には引数はない。Ｌｉｎｅ３で電源の使用権利として宣言したSemDCDCの権利を一つ獲得する。この時、使用権利が１つもなければこの関数を呼び出したタスクの実行は保留される。電源の使用権利を別のタスクが解放すればZoomUpを呼び出したタスクが再び実行可能状態に戻る。そして、Ｌｉｎｅ４のモーターを動かす関数を呼び出すことができる。そして、Ｌｉｎｅ５で、電源使用権利を返却してこの関数の仕事は終了する。Ｌｉｎｅ７は１セクタのデータをＥＥＰＲＯＭに書き込む関数の入り口であり、Ｌｉｎｅ９ｄｅ電源利用権利を獲得してＬｉｎｅ１２で返却している。
【０１１３】
図４０は、上述した電源の共有手順を説明するためのフローチャートである。同図において、ステップＳ１７０１で、電源コントローラ９によるＤＣ／ＤＣコンバータ８の出力電力の供給が解放されたか否かを判断する。ステップＳ１７０２では、電源確保のための指示の内容を解析し、この指示結果に従って、ステップＳ１７０３、１７０５、１７０７、１７０９のいずれかに分岐する。
【０１１４】
指示の内容が、ＣＣＤ駆動電力の供給であれば、ステップＳ１７０３へ進み、ＣＣＤ２に対してＣＣＤ駆動のための電力を供給する。そして、ステップＳ１７０４にて、ＣＣＤ駆動の終了（即ち撮影動作の終了）を検出すると、ステップＳ１７１１へ進み、電源の解放を行う。また、ストロボの充電要求であれば、ステップＳ１７０５へ進み、電源コントローラ９に対してストロボ２１に対する充電電力を提供させる。そして、ステップＳ１７０６でストロボの充電を完了したら、ステップＳ１７１１へ進み、電源の解放を行う。なお、充電の電力供給は、所定時間の充電を行う毎に他の電源供給のために電源を解放する。即ち、ストロボ２１への充電を管理するプログラムは別個に所定のタスクに存在し、充電の完了はそのタスクによって管理される。
【０１１５】
指示の内容が、ズーム機構の駆動であれば、ステップＳ１７０７へ進み、ズーム機構の駆動系（不図示）へ電力供給を行う。そしてステップＳ１７０８で、１ステップのズーム動作を終えたらステップＳ１７１１へ進み、電源を解放する。更に、指示の内容がフラッシュＲＯＭへの書込みであれば、ステップＳ１７０９へ進み、フラッシュＲＯＭ１５への書込み電力を供給する。１セクタ分の書込みが終えたら、ステップＳ１７１０からステップＳ１７１１へ進み、電源を解放する。
【０１１６】
なお、ステップＳ１７０４、１７０６、１７０８、１７１０において、各動作の終了を待つが、この待ちループにおいて、他のタスクへの制御が移り、マルチタスク処理が遂行される。この管理処理は、各タスクから随時起動が可能であり、複数のタスクで同時に起動される可能性もある為、ステップＳ１７０１で電源解放のチェックを行っている。
【０１１７】
以上の図４０のフローチャートによれば時分割で電源を利用することが可能となる。しかしながら、すべてのシステム（ＣＣＤ／ストロボ／ズーム／フラッシュＲＯＭ）が依存しあった１つのプログラムである。このようなソフトウエアを開発すると、開発／デバッグ／メンテナンスのコストが大きくなり、拡張性や柔軟性を保つのが難しくなる。
【０１１８】

そこで、電源を１つの資源に見立ててＯＳの提供する資源管理機能を用いることで開発効率を向上させることができる。そこで上述のセマフォによる資源管理を行う。即ち、ＣＣＤの駆動部、ストロボの駆動部、ズームの駆動部、フラッシュＲＯＭの駆動部のそれぞれの制御プログラムが、電源という資源（セマフォ）を獲得、解放することで、時分割された電源の割当てが行える。

【０１１９】

図４８は本実施形態による電源の時分割利用を説明する図である。同図に示されるように、あるタスクＡ（例えばＣＣＤ）によって電源要求が発生したとき、電源セマフォが解放された状態にあれば、そのセマフォを獲得して、電源を占有する（ステップＳ２００１～Ｓ２００３）。続いてステップＳ２００４において、当該電源よりの電力供給を得て所定の処理を行うと、ステップＳ２００５へ進んでセマフォを解放する。

【０１２０】

一方タスクＡより遅れて電源獲得を要求したタスクＢでは、ステップＳ２０１１における電源要求ではセマフォを獲得できず、ステップＳ２０１２により、セマフォの解放待ちとなる。そして、タスクＡよりセマフォが解放されると、このセマフォをタスクＢが獲得して、電源を占有する（ステップＳ２０１３）。その後タスクＢで所定の処理を実行し（ステップＳ２０１４）、電源を解放する（ステップＳ２０１５）。

【０１２１】

以上のようなセマフォによる電源資源の管理により、電源の時分割利用が可能となる。

【０１２２】

なお、図４８によれば、電源資源の利用権利を示すセマフォが一つしかないが、複数個のセマフォが存在するようにしても良いことは言うまでもない。

【０１２３】

＜実施形態の電子カメラの動作説明＞

図１７は、本実施形態のリブートからサービスの開始までの動作手順を表わすフローチャートである。ステップＳ１０１でシステムがリブートすると、ステップＳ１０２でフラッシュＲＯＭ１５の管理領域をスキャンし、ＤＲＡＭ１４上に記憶場所管理テーブル１４０を作成する。また、この処理と並行して、ＤＲＡＭ１４上の未使用セクタカウンタ、使用済セクタカウンタ、使用中セクタカウンタへ、それぞれの状態に対応するセクタがいくつ有るかを数え、セットする。このカウンタは、後にフラッシュＲＯＭ１５に対して操作を行ったときに更新され、記憶効率を判断するのに用いられる。その後、ステップＳ１０３へ進み、各種のサービスを開始する。

【０１２４】

図１８は、指定セクタの読み出しサービスの手順を表わすフローチャートである。まず、ステップＳ２０１でＮセクタの読み出しを開始する。ステップＳ２０２では、Ｎセクタをロックする。セクタのロックはロック変数を使って行う。このロック変数は、記憶場所管理テーブル１４０で各セクタの記憶場所とともに管理される。ステップＳ２０２では、セクタが既に他のタスクによってロックされている場合、他のタスクによってアンロックされるのを待ち、他のタスクによってアンロックされた後で当該セクタのロックを行う。ロックしたセクタはステップＳ２０６でアンロックするまでの間、自タスクによって占有することが出来る。

【０１２５】

ステップＳ２０２で論理セクタをロックすると、ステップＳ２０３で記憶場所管理テーブルを参照して、当該セクタに有効なデータ記憶されているかどうかを確認する。有効なデータが記録されて無い場合は、ステップＳ２０４へ分岐する。ステップＳ２０４では、ダミーのデータ（例えば全部０など）をセクタの内容として読み出す。ステップＳ２０３で有効なデータが格納されていると判断された場合は、ステップＳ２０５へ分岐する。ステップＳ２０５では記録場所管理テーブルの値を元にフラッシュＲＯＭ（または主記憶）からデータを読み出す。

【０１２６】

ここで、ガベージコレクションを実行中でＮセクタが主記憶（ＤＲＡＭ１４）へ退避されていた場合は、記憶場所管理テーブルのポインタは主記憶をポインティングとしている。また、図中点線で囲んだ部分はＮセクタを占有している期間である。この様なロック機構によって１つのセクタ操作の安全性を保証している為、ガベージコレクションの途中でも操作中でないセクタを自由に読み出すことが可能となっている。

【０１２７】

図１９は論理セクタの書き込みサービスの手順を表わすフローチャートである。ステップＳ３０１でＮセクタの書き込みを開始する。ステップＳ３０２でステップＳ２０２と同様に、論理セクタのロックを行う。

【０１２８】

次に、ステップＳ３０３で、記憶場所管理テーブルを検索して、Ｎセクタに有効なデータが記録されているかどうかを判断する。有効なデータが記録されていればステップＳ３０４へ、記録されてないならばステップＳ３０５へそれぞれ分岐する。ステップＳ３０４では、それまで有効なデータとして記録されていたフラッシュＲＯＭ（または主記憶）のデータを破棄する。ステップＳ３０４におけるデータ破棄の処理は、図２１のフローチャートを用いて詳しく説明を加える。ステップＳ３０４の後ステップＳ３０５へ制御が移る。

【０１２９】

ステップＳ３０５では、フラッシュＲＯＭ１５においてＮセクタを書き込むための記憶領域を獲得する。ステップＳ３０５における記憶領域の獲得手順は図２３を用いて詳しく説明を加える。ステップＳ３０５で正常に記憶領域の獲得に成功すれば、ステップＳ３０８へ制御を移す。ステップＳ３０８では獲得したフラッシュＲＯＭ１５の領域へＮセクタのデータを書き込む。

【０１３０】

一方、ステップＳ３０５でフラッシュＲＯＭに記憶場所が無い場合、即ち記憶領域の獲得に失敗した場合はステップＳ３０６へ分岐する。ステップＳ３０６はデータの退避用に主記憶を獲得する。主記憶の領域確保はオペレーティングシステムが提供するメモリ管理機能によって行う。これはＣ言語でａｌｌｏｃ関数に相当する機能である。そして確保した領域を片方向線形リスト構造によって管理する。

【０１３１】

図２０は主記憶上に獲得した退避データリストの様子である。（ａ）は退避データリストにデータが無い状態であり、リストには、END_OF_LIST が代入されている。（ｂ）は退避データリストに、セクタ番号３，２０，２２１の各セクタの内容が退避されている状態である。

【０１３２】

ステップＳ３０９で、記録場所管理テーブルを更新する。ここで、記録したフラッシュＲＯＭ（または主記憶）へのポインタが代入される。ステップＳ３１０で論理セクタのアンロックを行う。図中点線で囲まれた期間その論理セクタを占有できる。ステップＳ３１１で記憶効率の評価を行う。記憶効率の評価手順については、図２１のフローチャートを用いて詳しく説明を加える。記憶効率の評価の結果、記憶効率が悪化した場合は、ステップＳ３１２へ制御を移す。ステップＳ３１２では上述したガベージコレクションを行う。ガベージコレクションについては、図２４のフローチャートを参照して詳しく説明を加える。ステップＳ３１３でＮセクタの書き込みが終了してメインのルーチンへ復帰する。

【０１３３】

なお、記憶場所管理テーブルに納められるのは、記憶場所のポインタ（バス空間上のアドレス）である。図２０の（ｂ）の主記憶に待避されたデータの「次のデータへのポインタ」の次のフィールド（図中ではすぐ下に示されている）からは、図１０の左側にあるフラッシュＲＯＭ上のデータ構造と互換性がある。記憶場所管理テーブルに納められるのはこの互換部分へのポインタである。このように構成することにより、データの読み出しプログラム側でフラッシュＲＯＭと主記憶を単一のアルゴリズムで扱うことが可能となる。

【０１３４】

次に、指定されたセクタの記憶を破棄する手順（上述のステップＳ３０４）を説明する。図２１は、記憶を破棄する手順を表わすフローチャートである。

【０１３５】

ステップＳ４０１で指定領域の記憶破棄を開始する。ステップＳ４０２では、指定されたセクタを記憶する領域が主記憶上にあるかどうかを判断する。主記憶上にあるならステップＳ４０５へ分岐する。ステップＳ４０５で待避セクタリスト（本例では、図２０で示した片方向線形リスト）から指定領域を削除する。

【０１３６】

片方向線形リストからの指定領域の削除手順は、まずリストの先頭から検索方向順にリストをたどり、ポインタが自分をポインティングしている項を検出する。そして、この検出された項のポインタに現在自分がポインティングとしている値を代入することで実現する。そして、ステップＳ４０６で、リストから削除した主記憶領域をオペレーティングシステムへ返却する。オペレーティングシステムへの記憶領域の返却はＣ言語のｆｒｅｅ関数に相当する機能である。

【０１３７】

一方、ステップＳ４０２で指定された領域が主記憶上でない（すなわちフラッシュＲＯＭ上）ならステップＳ４０３へ分岐する。ステップＳ４０３では、指定されたフラッシュＲＯＭ上のセクタの管理フラグを“使用済”へ変更する。これは、使用済みフラグをＴＲＵＥにセットすることで達成される。ステップＳ４０４では主記憶上の未使用セクタカウンタの値を１つ減少させる。ステップＳ４０７で復帰する。

【０１３８】

次に、記憶効率の評価手順（ステップＳ３１１）について説明する。図２２は、記憶効率の評価手順を表わすフローチャートである。

【０１３９】

ステップＳ５０１で記憶効率の評価を開始する。ステップＳ５０２では、主記憶に設定された未使用セクタカウンタの値と使用済セクタカウンタの値を比較する。ここで、使用済セクタカウンタの値が未使用セクタカウンタの値に対して同じか上回った場合、上位プログラムに対して記憶効率の悪化をレポートする様に構成している（ステップＳ５０２、Ｓ５０４）。また、未使用セクタカウンタの値が使用済セクタカウンタの値よりも大きければ、評価結果を正常とし、正常復帰する（ステップＳ５０３）。

【０１４０】

次に、フラッシュＲＯＭの記憶領域の獲得手順（ステップＳ３０５）について説明する。図２３はフラッシュＲＯＭの記憶領域の獲得手順を表わすフローチャートである。

【０１４１】

ステップＳ６０１でフラッシュＲＯＭの記憶領域の獲得を開始する。ステップＳ６０２で未使用セクタの検索権利を獲得する。ここでは、オペレーティングシステムの提供するセマフォの機能を使用して未使用セクタの検索権利を管理している。ここでは、ステップＳ６０２からステップＳ６０９／ステップＳ６１１までの点線で囲まれた処理期間だけ未使用セクタの検索権利を独占出来る。複数のタスクが同時に同一領域を獲得する様な事態を防ぐ為のしくみである。

【０１４２】

ステップＳ６０３でフラッシュＲＯＭの最初のセクタへポインタを移動する。ステップＳ６０３でそのセクタの管理フラグ（使用中フラグ、使用済みフラグ）を参照して、当該セクタの使用状態を判断する。使用済みか使用中ならステップＳ６０５へ分岐する。ステップＳ６０５で現在ポイントしているセクタが最後のセクタならステップＳ６１１へ分岐する。この場合、使用可能な領域がフラッシュＲＯＭ１５に存在しないことになるので、ステップＳ６１１で未使用セクタの検索権利を開放した後、ステップＳ６１２で異常復帰する。また、ステップＳ６０５で現在ポイントしているセクタが最後のセクタでなければ、ステップＳ６０６へ分岐する。ステップＳ６０６では、ポインタを次のセクタへ移動させてからステップＳ６０４へ戻る。

【０１４３】
ステップＳ６０４ポインタの示すセクタの管理フラグが未使用となっていればステップＳ６０７へ分岐する。ステップＳ６０７では、フラッシュＲＯＭの管理フラグを“使用中”へ変更する（使用中フラグをＴＲＵＥにする）。そして、ステップＳ６０８で、主記憶に設けた未使用セクタカウンタの値を１つ減少させる。この場合は、フラッシュＲＯＭへの記憶領域の獲得に成功しているので、ステップＳ６０９で未使用セクタの検索権利を開放し、ステップＳ６１０で正常復帰する。
【０１４４】
次に、ガベージコレクション（ステップＳ３１２）の手順について説明する。図２４はガベージコレクションの手順を表わすフローチャートである。
【０１４５】
ステップＳ７０１でガベージコレクションを開始する。ステップＳ７０２では、整理対象のイレースブロック（以後、整理対象ブロック）を選出する。整理対象ブロックの選出手順については、図２５のフローチャートを用いて詳しく説明を加える。ステップＳ７０３では、整理対象ブロックの未使用セクタを使用済化する。この使用済化の手順については、図２６のフローチャートを用いて詳しく説明を加える。ここで、最初に整理対象ブロック内の未使用セクタを使用済化させる目的は、ガベージコレクション中であっても、他のタスクが整理対象ブロック内のセクタを含むセクタへの読み書きが可能な構成となっており、ガベージコレクション中に他のタスクによって整理対象ブロック内のセクタへ新たなデータが書き込まれることを防止する為である。
【０１４６】
ステップＳ７０４では、整理対象ブロック中の使用中セクタを他の記憶領域（即ち、他のイレースブロック）へ移動させる。使用中セクタを他の記憶領域へ移動させる処理については、図２７のフローチャートを参照して詳しく説明を加える。
【０１４７】
続くステップＳ７０５では、使用中セクタの移動を終了した整理対象ブロックの消去を実行する。整理対象ブロックを消去する手順については、図２８のフローチャートを参照して詳しく説明を加える。なお、この整理対象ブロックの消去において、消去回数カウンタ１５２の内容を主記憶にコピーしておく。ステップＳ７０５における整理対象ブロックの消去を終えると、ステップＳ７０６で主記憶に退避したデータをフラッシュＲＯＭの当該イレースブロックの消去回数カウンタへ戻す。そして、ステップＳ７０７でガベージコレクションから復帰する。
【０１４８】
次に、ガベージコレクションにおける整理対象ブロックの選出手順（ステップＳ７０２）について説明する。図２５は整理対象ブロック選出する手順を表わすフローチャートである。
【０１４９】
まず、ステップＳ８０１で整理対象ブロックの選出を開始する。ステップＳ８０２で評価ポインタに最初のイレースブロックをセットする。同様に、ステップＳ８０３で、整理対象候補ポインタを最初のイレースブロックにセットする。
【０１５０】
次に、ステップＳ８０４で、評価ポインタの示すイレースブロックに使用済セクタが含まれているかどうかを判断する。使用済セクタが含まれていなければステップＳ８０４，ステップＳ８０５をスキップしてステップＳ８０７へ制御を移す。
【０１５１】
一方、ステップＳ８０４で評価ポインタの示すイレースブロックに使用済セクタが含まれている場合には、ステップＳ８０５へ制御を移す。ステップＳ８０５では、整理対象候補ポインタの示すイレースブロックの消去回数カウンタの値と評価ポインタの示すイレースブロックの消去回数カウンタの値を比較する。もし評価ポインタの示すイレースブロックの消去回数の方が少なければステップＳ８０６へ制御を移す。ステップＳ８０６では、整

理対象候補ポインタへ評価ポインタを代入する。一方、ステップＳ８０５でもし評価ポインタの示すイレースブロックの消去回数のほうが多ければそのままステップＳ８０７へ制御を移す。

【０１５２】

ステップＳ８０７で評価ポインタが最後のイレースブロックを示しているかどうかを判断する。もし最後のイレースブロックでなければ、ステップＳ８０８で評価ポインタを次のイレースブロックへ移動させた後、ステップＳ８０４へ戻る。以上のように、ステップＳ８０４～Ｓ８０８の処理を繰り返すことで、整理対象候補ポインタは、使用済みセクタを含み、消去回数の少ないイレースブロックを示すようになる。

【０１５３】

ステップＳ８０７で評価ポインタが最後のイレースブロックを示している場合はステップＳ８０９へ分岐する。ステップＳ８０９ではガベージコレクション処理（図２４の処理）に復帰する。この時点の整理対象候補ポインタの示すイレースブロックが整理対象として選出される。

【０１５４】

次に、選択された整理対象ブロック内の未使用セクタを使用済み化する処理（ステップＳ７０３）について説明する。図２６は、整理対象ブロックの未使用セクタを使用済み化する手順を表わすフローチャートである。

【０１５５】

ステップＳ９０１で処理を開始する。ステップＳ９０２で、整理対象ブロックの最初のセクタへポインタを移動させる。次に、ステップＳ９０３で、未使用セクタの検索権利を獲得する。これは、図２３のフローチャートのステップＳ６０２と同様の効果があり、ステップＳ９０８までの点線で囲まれた間、未使用セクタの検索権利を独占する。即ち、整理対象ブロックの全セクタを対象にスキャンして未使用セクタを使用済セクタへ変更するまでの間、他のタスクが未使用セクタの検索をすることを禁止する。しかし、管理フラグのみの操作で未使用セクタを使用済セクタへ変更するので、検索権利の独占時間は短く、全体のスループットが低下することはない。

【０１５６】

ステップＳ９０４で、現在のポインタが示すセクタが未使用セクタかどうかを判断する。もし未使用セクタならステップＳ９０５へ分岐する。ステップＳ９０５でその記憶を廃棄する。ステップＳ９０５の処理手順は図２１のフローチャートで説明した通りである。この処理により、未使用セクタが使用済みセクタに変更される。ステップＳ９０６では、ポインタが整理対象ブロックの最後のセクタを示しているかどうかを判断する。最後のセクタを示していればステップＳ９０８へ、そうでないならステップＳ９０７へ分岐する。ステップＳ９０７ではポインタを次のセクタへ移動させてステップＳ９０４へ制御を戻す。

【０１５７】

また、ステップＳ９０６でポインタが整理対象ブロック最後のセクタならステップＳ９０８で未使用セクタの検索権利を開放し、ステップＳ９０９でガベージコレクション処理（図２４のフローチャート）へ復帰する。

【０１５８】

次に、整理対象ブロックの使用中セクタを他のイレースブロックの未使用セクタへ移動する処理（ステップＳ７０４）について説明する。図２７は、整理対象ブロックの使用中セクタの移動手順を表わすフローチャートである。

【０１５９】

ステップＳ１０００で処理を開始する。ステップＳ１００１で整理対象ブロックの最初のセクタへポインタを移動させる。以下のステップＳ１００２～Ｓ１０１２では、ポインタが指し示すセクタについて処理を行う。

【０１６０】

ステップＳ１００２で当該セクタの管理フラグ（使用中フラグ、使用済みフラグ）を判断する。ステップＳ１００２で管理フラグの値が「使用中」となっていたらステップＳ１０

０３へ制御を移し、「使用済」となっていたらステップＳ１０１２へ制御を移す。ステップＳ１００３で、論理セクタをロックする。ロックしたセクタはステップＳ１０１１でアンロックされるまでの間、自タスクで占有される。

【０１６１】

ステップＳ１００４では記憶領域を獲得する。ステップＳ１００４における記憶領域の確保の手順は、図２３のフローチャートで説明した通りである。ここで、整理対象ブロック内の各セクタは上記ステップＳ７０３の処理で、全て使用済み化されているので、確保される記憶領域は整理対象ブロック以外のイレースブロックとなる。

【０１６２】

記憶領域の獲得に成功すると、処理はステップＳ１００８へ進む。ステップＳ１００８では、獲得した領域へ当該セクタのデータをコピーする。そして、セクタの移動に従って、ステップＳ１００９で記憶場所管理テーブル１４０を更新する。

【０１６３】

一方、ステップＳ１００４で記憶領域の獲得に失敗した場合は、ステップＳ１００５へ分岐する。ステップＳ１００５では、データ退避用の記憶領域を主記憶（ＤＲＡＭ）より獲得する。データ退避用記憶領域の獲得は図１９のフローチャートのステップＳ３０６で説明した通りである。ステップＳ１００６では、獲得した領域へ当該セクタのデータをコピーする。そして、ステップＳ１００７で記憶管理テーブルを更新する。ステップＳ１０１０で元の記憶を廃棄する。即ち、ポインタの指し示すセクタの使用済みフラグをＴＲＵＥにセットする。そして、ステップＳ１０１１で当該論理セクタをアンロックする。

【０１６４】

ステップＳ１０１２で、ポインタの指し示すセクタが、整理対象ブロックの最後のセクタかどうかを判断する。最後のセクタであればステップＳ１０１４へ、最後のセクタでなければステップＳ１０１３へそれぞれ分岐する。ステップＳ１０１３では、ポインタを次のセクタへ移動させて、ステップＳ１００２へ戻り、次のセクタについて上述の処理を繰り返す。また、ステップＳ１０１４では、整理対象ブロック内の全てのセクタについて処理を終えているので、ガベージコレクション処理（図２４のフローチャート）へ復帰する。

【０１６５】

次に、整理対象ブロックの消去処理（ステップＳ７０５）について説明する。図２８は、整理対象となったイレースブロックの消去手順を表わすフローチャートである。

【０１６６】

ステップＳ１１０１で処理を開始する。ステップＳ１１０２で整理対象ブロックの消去回数カウンタを主記憶へコピーする。ステップＳ１１０３では整理対象ブロックの消去を実行する。ステップＳ１１０４では、主記憶へコピーした消去回数カウンタの値を１増加させた値をフラッシュＲＯＭへ書き込む。即ち、当該整理対象ブロックの消去カウンタの値を、消去処理前の値より１増加させる。その後、ステップＳ１１０５でガベージコレクション処理（図２４のフローチャート）へ復帰する。

【０１６７】

上記図２４で示されるガベージコレクション処理は、極力フラッシュＲＯＭを用いた処理であり、退避データの安全性が高い。しかしながら、上述の＜未使用セクタが無い場合＞の項で説明したように、積極的に主記憶（ＤＲＡＭ１４）を用いて使用中セクタのデータを待避し、複数個のイレースブロックを消去すると消去処理の効率がよい。但し、ＤＲＡＭ１４にデータを待避するので、待避中のデータに関して安全性が低下する（例えば電池が外れて電源供給が停止するとＤＲＡＭに待避したデータが失われることになる）。そこで、電源の種別を判断し、供給電源が電池の場合は待避データの安全性を重視し、ＡＣアダプタの場合は電源供給が停止する危険性が少ないので消去処理の効率を重視するように構成してもよい。この場合の処理について図３６を参照して説明する。

【０１６８】

図３６は、電源種別に基づいてガベージコレクション処理を切り換える場合の処理手順を説明するフローチャートである。同図において、図２４のフローチャートで示される処理

と同じ処理を行うステップについては同一のステップ番号を付し、ここでは詳細な説明を省略する。

【０１６９】

ステップＳ１３００においてガベージコレクション処理が起動されると、ステップＳ１３０１へ進み、当該装置への電源供給の形態を判断する。ここでは、図１の電源コントローラ９が、電源の供給元が電池７であるかＡＣアダプタ２３であるかを判断し、ＣＰＵ５に通知する。電源種別が電池７であった場合は、ステップＳ１３０４へ進み、上述の図２４で示したガベージコレクション処理を実行する。

【０１７０】

一方、ステップＳ１３０１において電源種別がＡＣアダプタであった場合は、ステップＳ１３０２へ進む。ステップＳ１３０２では、図２４のステップＳ７０２、Ｓ７０３、Ｓ７０４に相当する処理を実行し、選出した整理対象ブロック内の未使用セクタの使用済み化と使用中セクタの待避を行う。そして、ステップＳ１３０３において、主記憶（ＤＲＡＭ１４）にセクタの待避を行うのに十分な空き領域があるか否かを判断し、十分な空き領域があればステップＳ１３０２へ戻る。ステップＳ１３０２では、前回の整理対象ブロックとは別の整理対象ブロックを選出して、上述の処理を繰り返す。

【０１７１】

ＤＲＡＭ１４上に十分な空き領域が無くなると、ステップＳ１３０３からステップＳ１３０４へ進み、上述の処理で選出された整理対象ブロックの消去を行う。そして、ステップＳ７０６で主記憶に待避したデータをフラッシュＲＯＭ１５に戻して本処理を終了する。

【０１７２】

以上のように、図３６の処理によれば、電源がＡＣアダプタによって供給される場合は、主記憶の空き容量を積極的に利用してデータの待避を行い、複数の整理対象ブロックを選出して、一括して消去処理を行うことができ、消去処理の効率が向上する。

【０１７３】

なお、上記の処理では、電源種別に基づいて自動的にガベージコレクションの形態を切り換えるが、コントロールパネル１２の操作により、マニュアルで切り換えるようにすることもできることはいうまでもない。

【０１７４】

次に、基本サービスの一つである論理セクタの解放手順について説明する。図２９は、論理セクタの解放手順を表すフローチャートである。

【０１７５】

ステップＳ１２０１でＮセクタの解放を開始する。ステップＳ１２０２でＮセクタをロックする。この結果、ステップＳ１２０５でアンロックされるまでの間、自タスクで論理セクタを占有出来る。続いて、ステップＳ１２０３で当該セクタの記憶を廃棄する。この記憶の廃棄処理については、図２１のフローチャートで説明した通りである。ステップＳ１２０４では、ＤＲＡＭ１４の記憶場所管理テーブル１４０へ“不在”値を代入する。ステップＳ１２０５では、論理セクタをアンロックし、ステップＳ１２０６で復帰する。

【０１７６】

例えばＭＳ－ＤＯＳ（商標）等の一般のファイルシステムでは、ファイルの消去に際しては、当該ファイルに属するセクタをＦＡＴにおいて上書き可能とするのみで、各セクタを解放するということは行われない。よって、このようなファイルシステムに本実施形態のフラッシュＲＯＭ管理システムを適用すると、ファイルシステム上では無効となったデータが、有効なセクタとして残されてしまうことになり、ガベージコレクション等の効率を低下させることになる。よって、ファイルシステムの指示（例えばファイル消去）に基づいて不要となったセクタを検出し、これを解放するように構成すれば、ガベージコレクションの効率をより向上させることができる。

【０１７７】

図３７は、ファイルシステムよりファイル消去が指示された場合の、不要セクタの解放手順を表すフローチャートである。同図において、ステップＳ１４０１でファイルシステム

よりファイル消去の指示があったか否かを判断する。ファイル消去の指示があった場合は、ステップＳ１４０２へ進み、消去すべく指示されたファイルに含まれるセクタを抽出する。セクタの抽出は、ＦＡＴを参照することで抽出できる。そして、ステップＳ１４０３で、先のステップＳ１４０２で抽出された各セクタについて、上記図２９のフローチャートで説明したセクタの解放処理を実行する。

【０１７８】

［実施形態２］

次に実施形態２について説明する。

【０１７９】

＜ディスクコントローラエミュレーション＞

上述の実施形態１で説明したフラッシュＲＯＭの記憶管理システムは、上位層から見た特徴がディスク媒体と良く似ている。従って、ディスクコントローラのエミュレーション機能を備えたシステムに組み込むことで、ディスクコントローラとディスク媒体をディスクコントローラエミュレーションと本実施形態の記憶管理システム（あるいは本実施形態の記憶管理システムを組み込んだＩＣカード）へ置き換えることが可能となる。近年ＰＣＭＣＩＡに代表されるＩＣカードが普及しているが、ＩＣカードへディスクコントローラエミュレーション機能と上記実施形態１の記憶管理システムを組込むことにより、リムーバルな記憶媒体として利用することが可能となる。第２の実施形態では、実施形態１の記憶管理システムをＩＣカードへ組み込んだものについて説明する。

【０１８０】

図３０は実施形態２におけるＩＣカードの構成を表すブロック図である。同図において、２００はＩＣカード全体を示す。２０１はマイクロコンピュータであり、ディスクコントローラエミュレーション及び記憶管理を行う。２０２はＲＯＭであり、マイクロコンピュータ２０１のプログラムを格納する。２０３はＲＡＭであり、マイクロコンピュータ２０１の主記憶として機能する。２０４はフラッシュＲＯＭであり、上記実施形態１で説明した記憶管理システムによってデータを蓄積する。即ち、フラッシュＲＯＭ２０４は、図４で説明した管理領域とデータ領域とで管理される。

【０１８１】

２０５はコマンド／データ・ラッチ部であり、ホスト装置より受信した外部バスからのコマンドとシリンダ番号等を保持する。２０６はＦＩＦＯメモリであり、先入れ先出し方式でデータの入出力を行う。２０７はタプルＲＯＭであり、当該カードの特徴等を記憶しており、外部バスからのみ読み出しができる。

【０１８２】

上述の各構成の機能は、以降の動作説明でより明らかとなる。

【０１８３】

図３１は、本実施形態２のＩＣカードを利用する為のホストシステムの簡単なブロック図である。同図において、３０１はホストシステム側のマイクロコンピュータである。３０２はカードインターフェースであり、ホストシステムの内部バスとＩＣカード２００の外部バスを接続する。なお、カードインターフェース３０２は、ＩＣカード２００への電源供給を行うための電源供給線や、ＩＣカード２００からの割り込み要求（ＩＲＱ出力）を受け付けるための信号線も備えている。

【０１８４】

図３２は、図３１のホストシステムがＩＣカードを接続する際の手順を示すフローチャートである。ステップＳ４１００で処理を開始すると、ステップＳ４１０１でＩＣカードへの電源供給を開始する。ステップＳ４１０２では、ＩＣカード２００内のタプルＲＯＭ７から、タプル形式で格納されているデータを解析する。タプルＲＯＭ７の内容を解析することで、接続されているＩＣカードの特徴が分かる。

【０１８５】

ステップＳ４１０３では、ステップＳ４１０２で解析したタプル情報によって、接続されているＩＣカードが内部バスへ接続可能かどうかを判断する。そして、接続可能ならステ

ップＳ４１０４へ、接続不可能ならステップＳ４１０５へとそれぞれ分岐する。ステップＳ４１０４では、ＩＣカード側のバスをホストの内蔵バスのメモリ空間とＩＯ空間へマッピングする。この時点でホスト装置のバスの空間にディスクコントローラが有るのと同じ状態になる。

【０１８６】

図３３はＩＣカード２００内のマイクロコンピュータ１のメインシーケンスを示すフローチャートである。ステップＳ４２０１でＩＣカードの電源が投入されると、ステップＳ４２０２で記憶管理システムの初期化を行う。即ち、フラッシュＲＯＭ２０４の全イレースブロックの論理セクタの状態を一旦読み出し、読み出した情報に従って主記憶用のＲＡＭ２０３へ記憶場所管理テーブルを作成する。ステップＳ４２０３で主記憶上のコマンドバッファとしてリング状のバッファを用意して初期化し、割り込み処理を許可する。この処理以降割り込みルーチンの動作が始まる。

【０１８７】

割り込みルーチンのシーケンスを図３４のフローチャートに示す。割り込みルーチンの動作を理解した方が、図３３のフローチャートの説明が容易となる為、ここで図３４のフローチャートについて説明を行う。

【０１８８】

ホストシステムがコマンド／データ・ラッチ２０５へのコマンドのアドレスへコマンドを書き込むと、コマンド／データ・ラッチ２０５からマイクロコンピュータ２０１へ割り込みが発生する。コマンド／データ・ラッチ２０５は、ホストバスとＩＣカード内部のバスのＩＯアドレス空間にマッピングされていて、コマンド／データはそれぞれ図３５に示すようにＩＯアドレスが割り振られている。図３５は、本実施形態のコマンド／データ・ラッチにおけるＩＯ割り付けを示す図である。本例では、図３５中のCommandのアドレスにコマンド（例えばデータの読出しを指示するReadSector(s)）を書き込むことでマイクロコンピュータ２０１へ割り込みが発生する。

【０１８９】

割り込みが発生すると、マイクロコンピュータ２０１のソフトウェアは、図３４のフローチャートのステップＳ４３０１へ制御を移す。ステップＳ４３０２では、コマンド／データ・ラッチ２０５に書き込まれたデータを読み出して、主記憶上のリングバッファへデータを格納する。ステップＳ４３０３で割り込みルーチンを終了して図３３のフローチャートへ復帰する。

【０１９０】

図３３のフローチャートの説明に戻る。ステップＳ４２０４でマイクロコンピュータ２０１はコマンドバッファの状態を判断する。コマンドバッファへデータが格納されていれば、ステップＳ４２０５へ分岐し、データが格納されていなければステップＳ４２１３へ分岐する。ステップＳ４２１３ではＣＰＵを休止状態にする。多くのワンチップマイクロコンピュータは、命令の実行を休止して消費電流を減らす機能を備えているが、本実施形態のＣＰＵもこの種の機能を備える。そして、ＩＲＱによる割り込み要求信号が入力されると、ＣＰＵ２０１は休止状態から復帰して上述の割り込みルーチンを実行する。割り込みプログラムの実行が済んだ時点でステップＳ４２１３から復帰してステップＳ４２０４へ戻る。

【０１９１】

ステップＳ４２０４でコマンドバッファへデータが格納されていると、ステップＳ４２０５へ移行する。ステップＳ４２０６では、リングバッファからデータを読み出す。ステップＳ４２０６でコマンドを解釈する。Seekコマンドの場合はステップＳ４２０７、ReadSector(s)コマンドの場合はステップＳ４２０８へ、WriteSector(s)の場合はステップＳ４２０９へ、IdentifyDrvコマンドの場合はステップＳ４２１０へそれぞれ分岐する。他にもコマンドがあるが本実施形態の説明上重要でないものは省き、フローチャートを簡略化している。ステップＳ４２０７～４２１０までのコマンドの実行を終了したらステップＳ４２０４まで戻り、上記の処理を繰り返す。

【０１９２】
ステップＳ４２０７では、Seekコマンドを実行する。Seekといってもフラッシュ ＲＯＭには、ディスクデバイスと違ってヘッドが無いので、次のコマンドに備えての妥当性等をチェックするだけである。ＩＣカードのサポートするヘッド数を超えるヘッド位置などを指定された場合は、ディスク装置同様にエラーが発生する。
【０１９３】
ステップＳ４２０８はReadSector(s)コマンドに対する処理を行う。ReadSector(s)コマンドは、読み出すべきセクターの個数が図３５のSectorCountで指定される。よって、ステップＳ４２０８では、指定された場所のセクタをSectroCount個読み出す行為を行う。本実施形態の記憶管理システムでは、リニアな論理セクタ番号を使って管理を行っているので、シリンダ／ヘッド／セクタ番号を元にリニアな論理セクタ番号を計算し、論理セクタの内容をＦＩＦＯメモリ２０６へ転送し、コマンド／データ・ラッチ２０５のSectorNumberのインクリメントも行う。ＦＩＦＯメモリ２０６は、ＩＣカード２００の内部バスから書き込んだデータを外部バスから読み出すことができ、また外部バスから書き込んだデータをＩＣカード内部バスから読み出す構成となったＦＩＦＯメモリである。
【０１９４】
ここで、上述のリニアな論理セクタ番号について説明する。一般にハードディスクに対して指定する番号は、セクタ、シリンダ、ヘッドのパラメータで決まる３次元の不連続な番号である。例えば、シリンダ数が１０２４個、ヘッド数が１６個、セクタ数が６３個のハードディスクの場合、セクタ数は１０２４×１６×６３＝１０３２１９２個となる。
【０１９５】
このセクタを０番から１０３２１９２番としてアクセスできると良いのであるが、上記の３つのパラメータをすべて指定してアクセスするように設計されている。例えば、シリンダ５００・ヘッド１６・セクタ６３の次は、シリンダ５０１・ヘッド０・セクタ１をアクセスするといった具合である。なお、これら３つのパラメータをそれぞれの頭文字をとってＣＨＳパラメータと呼ぶ。
【０１９６】
ＭＳ－ＤＯＳ（商標）のようなオペレーティングシステムでは、内部ではリニア（連続的）なセクタ番号を用いるが、デバイスドライバがこれをＣＨＳパラメータに変換する。本実施形態のシステムでは、リニアなセクタ番号を用いるのでＣＨＳパラメータの値を元にリニアなセクタ番号を求める。上記で挙げたハードディスクの場合は、
シリンダ番号×（１６×６３）＋ヘッド番号×（６３）＋セクタ番号
を計算することで、リニアな論理セクタ番号が求まる。
【０１９７】
ステップＳ４２０９はデータラッチで指定された場所のセクタへデータを書き込む処理を行う。データは、ＦＩＦＯメモリ２０６経由でホストシステムから受け取る。
【０１９８】
ステップＳ４２１０は、ＩＣカード２００がどのようなハードディスクをエミュレーションしているかという情報を返す処理を行う。すなわちシリンダ数やModelNumberなどハードディスクとしてのスペックを含むデータをＦＩＦＯメモリ２０６へ書き込む処理を行う。
【０１９９】
＜ファイルシステムの解析＞
以上説明した様に実施形態１で説明した記憶管理システムをＩＣカードに組み込むことで、ＡＴＡハードディスク等の置き換え用途に使用できる。しかし、ＡＴＡコマンド等のＦＡＴキャッシュやファイル消去によって生じた不要セクタの開放といった処理を行う為の情報を上位システムからもらう手法が無い。ＡＴＡコマンドの空き部分を利用してセクタの開放コマンドとキャッシュするセクタ番号指定コマンドを追加実装することで、ＦＡＴキャッシュと不要セクタ解放の機能が実現できる。そして、この様な機能があることを想定していない現状のＭＳ－ＤＯＳ等のシステムでも、ＦＡＴキャッシュやセクタの開放を

実現できた方が良いことは言うまでもない。

【０２００】

ＦＡＴシステムは論理セクタ番号０に相当する部分にＦＡＴの場所やサイズといった情報を格納している。本実施形態では、このセクタを読むことでＦＡＴの場所やサイズを取得し、ＦＡＴキャッシュの処理に利用する。同様に本来書き込みデータの内容を理解しないはずのＩＣカードであるが、ファイルシステムの為の情報（ディレクトリエントリやＦＡＴ）を解析することでＩＣカードが自立的に不要セクタを判断して開放する等の処理に役立てることが出来る。もちろんＦＡＴに限った話では無く、ＨＰＦＳやマッキントッシュ（商標）のファイルシステムでも書き込むデータの内容を解析すれば、不要セクタの検出が可能である。この様に構成することでＡＴＡハードディスクのインタフェースでも、ファイルシステムの動作に合わせた最適化処理を行うことを可能にする。

【０２０１】

上記装置の機能もしくは方法の機能によって達成される本発明の目的は、前述の実施形態のプログラムを記憶させた記憶媒体によっても達成できる。例えば、パーソナルコンピュータに、その記憶媒体を装着し、その記憶媒体から読み出した以下に説明するようなフラッシュＲＯＭ管理プログラムを実行することにより、フラッシュＲＯＭをディスクシステムと同等に使用できるようになるとともに、効率的なガベージコレクションが可能となる。このための本発明にかかるプログラムの構造的特徴は、図４９に示す通りである。

【０２０２】

図４９は本実施形態における記憶媒体に格納される制御プログラムの制御手順、及び本記憶媒体のメモリマップを示す図である。

【０２０３】

図４９（ａ）において、３５０は管理処理であり、データ領域、及びデータ領域に対応する管理領域とで構成される複数のセクタをフラッシュＲＯＭに形成し、各セクタにおいてデータ領域の記憶状態を示す状態情報（セクタ番号１５４、使用中フラグ１５５、使用済フラグ１５６）を管理領域に格納し、該状態情報に基づいてフラッシュＲＯＭ１５のアクセスを管理する。例えば、図１８や図１９のフローチャートで示したように、セクタ単位でのフラッシュＲＯＭへのデータの書込みや読出しを制御する。

【０２０４】

また、３５１は検出処理であり、ファイルシステムよりの指示に基づいて不要とすべきセクタを検出する。例えば、ファイル消去の指示であれば、ＦＡＴを参照して、当該ファイルに属するセクタを検出する（図３７のステップＳ１４０２に相当する）。

【０２０５】

３５２は廃棄処理であり、指定されたセクタについて、該セクタの管理領域における使用済フラグをＴＲＵＥにし、記憶場所管理テーブルの更新等を行い、指定されたセクタの廃棄（解放）を行う（図２９のフローチャート）。なお、廃棄処理３５２は、上記検出処理３５１で検出されたセクタを指定されたセクタとして、廃棄することもできるし（図３７のステップＳ１４０３）、ファイルシステムから指定されたセクタを廃棄することもできる。

【０２０６】

上記制御手順を実現するための制御プログラムは、フロッピーディスクやハードディスク、あるいはＣＤ－ＲＯＭ等の記憶媒体に、例えば図４９の（ｂ）のメモリマップに示すような構成で格納される。上記制御プログラムは、例えばパーソナルコンピュータ等の情報処理装置によって読み出され、主記憶（ＲＡＭ）上にロードされて、ＣＰＵにより実行される。なお、主記憶上への上記制御プログラムのロードは、ＬＡＮを介して行われてもよい。

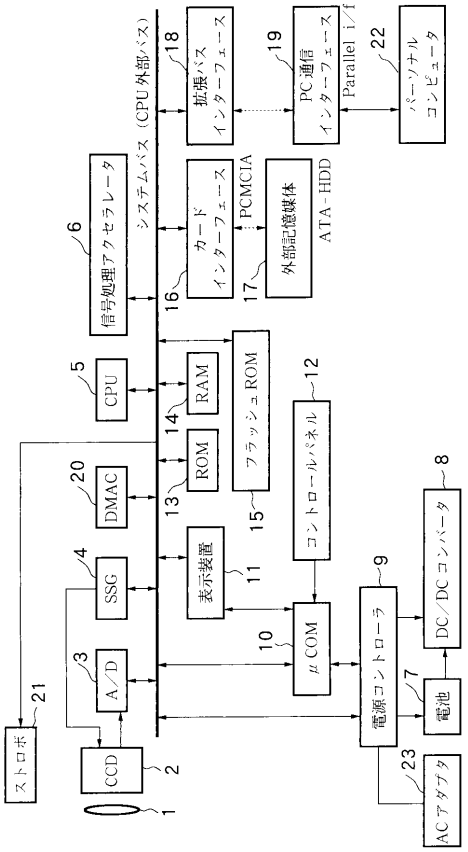
【０２０７】

なお、図４９（ｂ）において、管理処理モジュール３５０’、検出処理モジュール３５１’、廃棄処理モジュール３５２’は、それぞれ制御手順で示した管理処理３５０、検出処理３５１、廃棄処理３５２の各処理を実行するプログラムモジュールである。

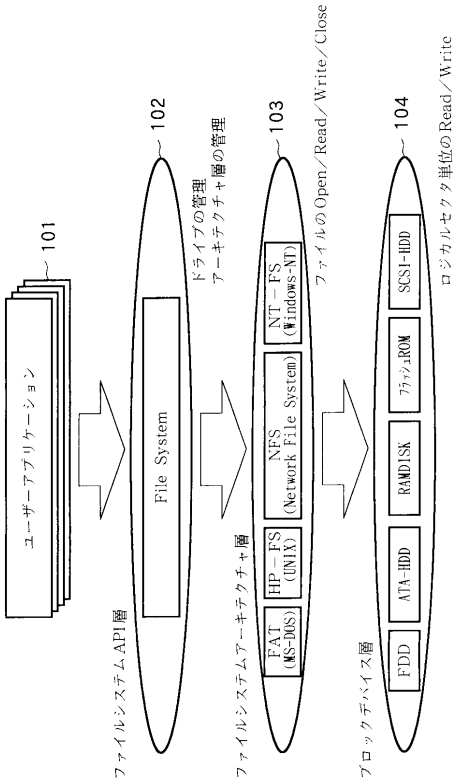
【０２０８】
また、本発明は、複数の機器から構成されるシステムに適用しても、１つの機器からなる装置に適用してもよい。また、本発明はシステム或は装置にプログラムを供給することによって達成される場合にも適用できることは言うまでもない。この場合、本発明に係るプログラムを格納した記憶媒体が、本発明を構成することになる。そして、該記憶媒体からそのプログラムをシステム或は装置に読み出すことによって、そのシステム或は装置が、予め定められた仕方で動作する。
【０２０９】
【発明の効果】
以上説明したように、本発明によれば、フラッシュＲＯＭをファイルシステムに適応させることが可能な管理方式において、セクタに相当する記憶ブロックの開放を可能とし、無効となったデータを効率良く消去することが可能となる。
【０２１０】
また、本発明によれば、ファイルシステムによるフラッシュＲＯＭへのアクセス内容に基づいてデータを無効化すべき記憶ブロックを検出し、当該記憶ブロックのデータを廃棄（無効化）することが可能となり、記憶ブロックのデータの有効・無効の管理が容易となる。
【０２１１】
【図面の簡単な説明】
【図１】実施形態１におけるカメラシステムの構成を表すブロック図である。
【図２】本実施形態の電子カメラにおけるファイルシステムの階層構造を表す図である。
【図３】デバイスドライバの管理ブロックをＣ言語で記述した宣言文を示す図である。
【図４】フラッシュＲＯＭ上のセクタ構造の例を示す図である。
【図５】管理領域用データと、データ領域とを分離して格納する構成を表す図である。
【図６】各フラグの状態に対応した意味を示す図である。
【図７】フラッシュＲＯＭにおけるセクタの書き換え手順を説明する図である。
【図８】本実施形態におけるフラッシュＲＯＭのガベージコレクション動作を説明する図である。
【図９】未使用セクタが存在しない場合のガベージコレクションの動作を説明する図である。
【図１０】ＤＲＡＭ上に作成された記憶場所管理テーブルを説明する図である。
【図１１】キャッシュソフトウエアの階層的な位置付けを表す図である。
【図１２】キャッシュの主記憶上のデータ構造を表わす図である。
【図１３】ＦＡＴキャッシュの読み出し手順を表すフローチャートである。
【図１４】ＦＡＴキャッシュの書込み手順を表すフローチャートである。
【図１５】データ書き込み完了を確認するための動作手順をＣ言語で表現した図である。
【図１６】ＤＣ／ＤＣコンバータの出力容量を越えない様に電源を管理するプログラムをＣ言語で表現した図である。
【図１７】本実施形態のリブートからサービスの開始までの動作手順を表わすフローチャートである。
【図１８】指定セクタの読み出しサービスの手順を表わすフローチャートである。
【図１９】論理セクタの書き込みサービスの手順を表わすフローチャートである。
【図２０】主記憶上に獲得した退避データリストの様子である。
【図２１】記憶を破棄する手順を表わすフローチャートである。
【図２２】記憶効率の評価手順を表わすフローチャートである。
【図２３】フラッシュＲＯＭの記憶領域の獲得手順を表わすフローチャートである。
【図２４】ガベージコレクションの手順を表わすフローチャートである。
【図２５】整理対象ブロック選出する手順を表わすフローチャートである。
【図２６】整理対象ブロックの未使用セクタを使用済み化する手順を表わすフローチャートである。

【図２７】整理対象ブロックの使用中セクタの移動手順を表わすフローチャートである。
【図２８】整理対象となったイレースブロックの消去手順を表わすフローチャートである。
【図２９】論理セクタの解放手順を表すフローチャートである。
【図３０】実施形態２におけるＩＣカードの構成を表すブロック図である。
【図３１】本実施形態２のＩＣカードを利用する為のホストシステムの簡単なブロック図である。
【図３２】図３１のホストシステムがＩＣカードを接続する際の手順を示すフローチャートである。
【図３３】ＩＣカード内のマイクロコンピュータのメインシーケンスを示すフローチャートである。
【図３４】ＩＣカード内のマイクロコンピュータの割り込み処理の手順を表すフローチャートである。
【図３５】ＩＯアドレスの割り付け状態を表す図である。
【図３６】電源種別に基づいてガベージコレクション処理を切り換える場合の処理手順を説明するフローチャートである。
【図３７】ファイルシステムよりファイル消去が指示された場合の、不要セクタの解放手順を表すフローチャートである。
【図３８】本実施形態における消去処理速度向上のための前処理の制御手順を表すフローチャートである。
【図３９】本実施形態におけるフラッシュＲＯＭへの１バイトデータの書込み手順を表すフローチャートである。
【図４０】電源の共有手順を説明するためのフローチャートである。
【図４１】本実施形態におけるフラッシュＲＯＭへの制御プログラムの格納状態を説明する図である。
【図４２】相対アドレスで表現されたプログラムコードの一例を表す図である。
【図４３】図４２のリロケーション情報レコードのデータを格納するテーブルを表す図である。
【図４４】図４１のプログラムを主記憶の８７１０番地へマッピングした場合のプログラムコードを示す図である。
【図４５】ディレクトリスロットの特徴を表す図である。
【図４６】図４５のディレクトリスロットにおいて、FileB が削除された状態を示す図である。
【図４７】本実施形態のＤＯＳ互換ファイルシステムでファイルを消去した後の状態を表す図である。
【図４８】本実施形態による電源資源（セマフォ）の時分割利用を説明するフローチャートである。
【図４９】本実施形態の制御を実現する制御プログラムを提供する記憶媒体の内容を説明する図である。

【図１】



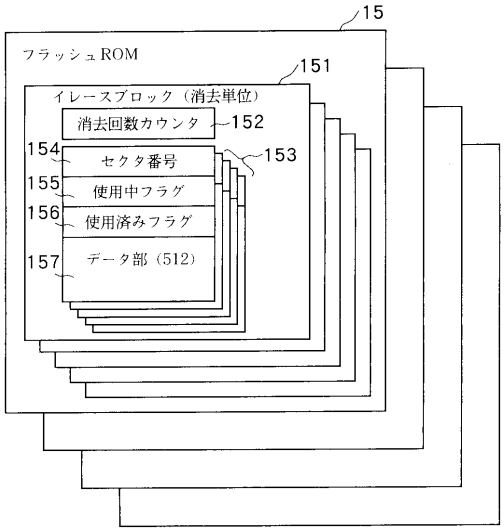
【図２】



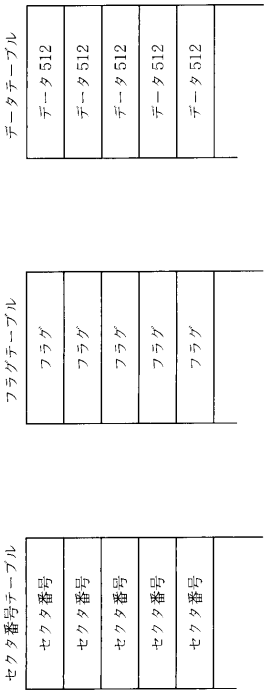
【図３】

```
1 : typedef struct DeviceTag {
2 :     struct DeviceTag * Next ;
3 :     char   DevName [32] ;
4 :     int    (* InitDev) (void) ;
5 :     int    (* ShutDown) (void) ;
6 :     int    (* ReadSector) (long lsect, long nsect, char * buffer) ;
7 :     int    (* WriteSector) (long lsect, long nsect, char * buffer) ;
8 :     int    (* ReleaseSector) (long lsect, long nsect) ;
9 : } Device ;
```

【図４】



【図５】



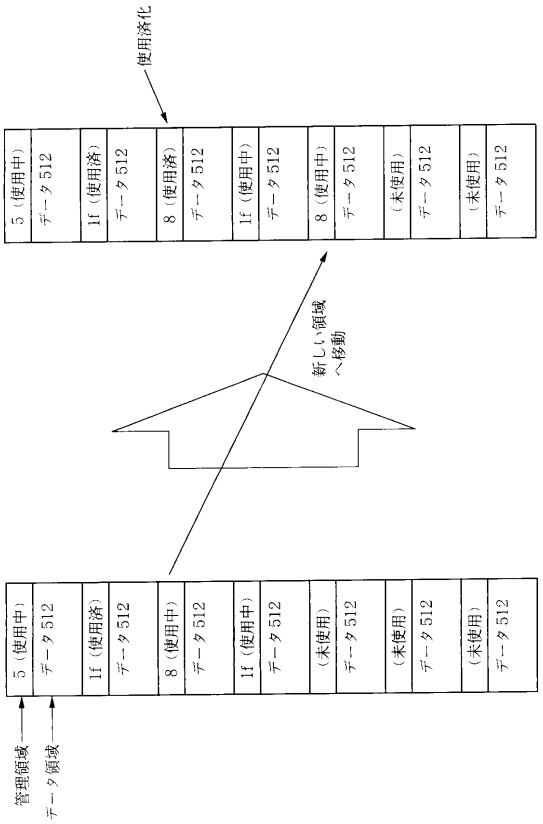
【図６】

使用中フラグ	使用済みフラグ	意味
FALSE	FALSE	「未使用セクタ」…消去後の状態
TRUE	FALSE	「使用中セクタ」…セクタ番号が示すセクタをデータ部に格納している
TRUE	TRUE	「使用済みセクタ」…データ部に格納されているデータは無効であり消去するまでこのセクタを利用できない

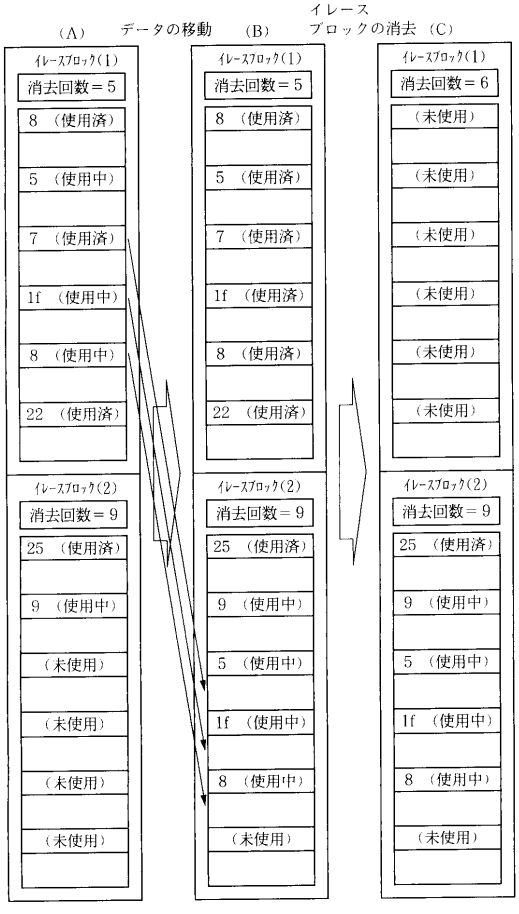
155　156

FALSE：消去後の状態　TRUE：FALSEのビット反転値

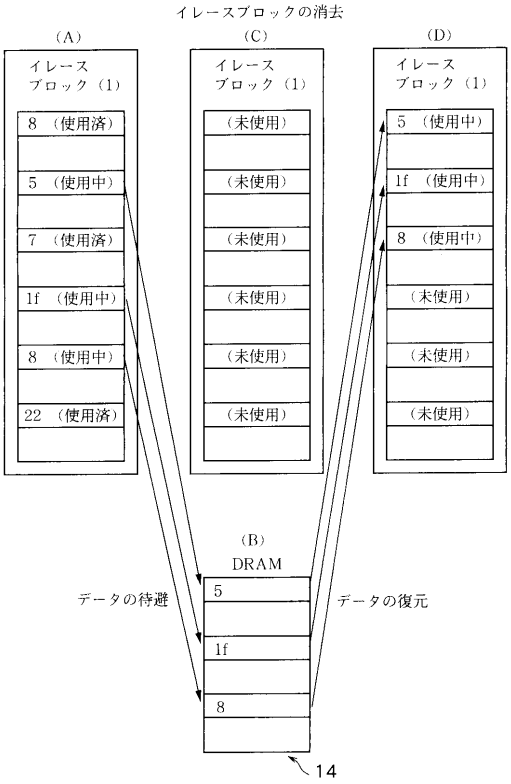
【図７】



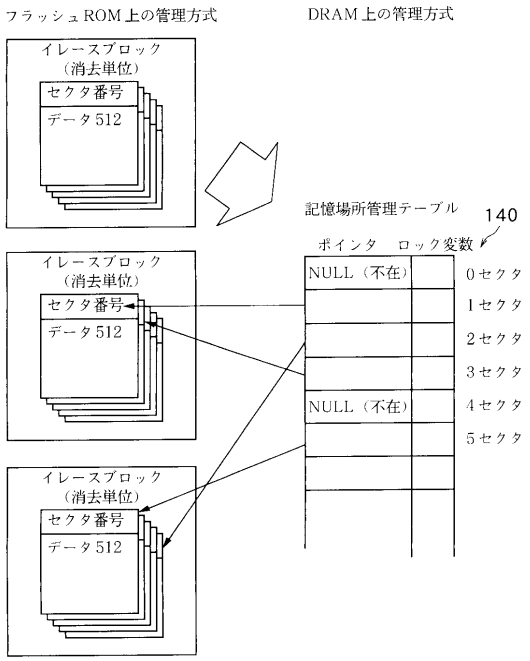
【図８】



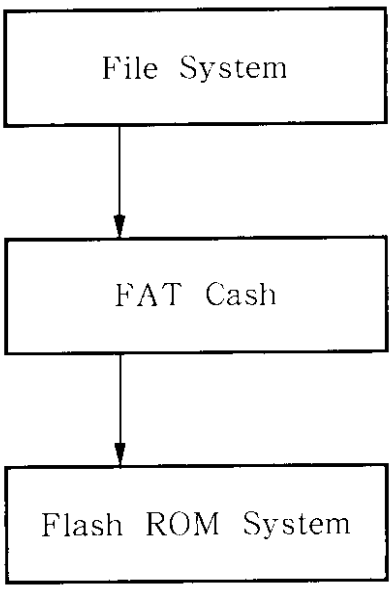
【図９】



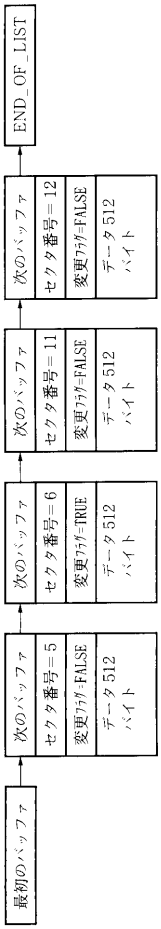
【図１０】



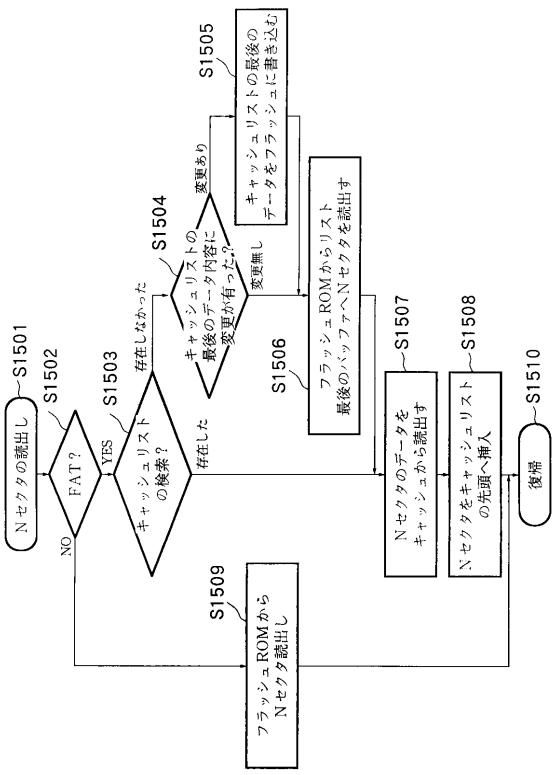
【図１１】



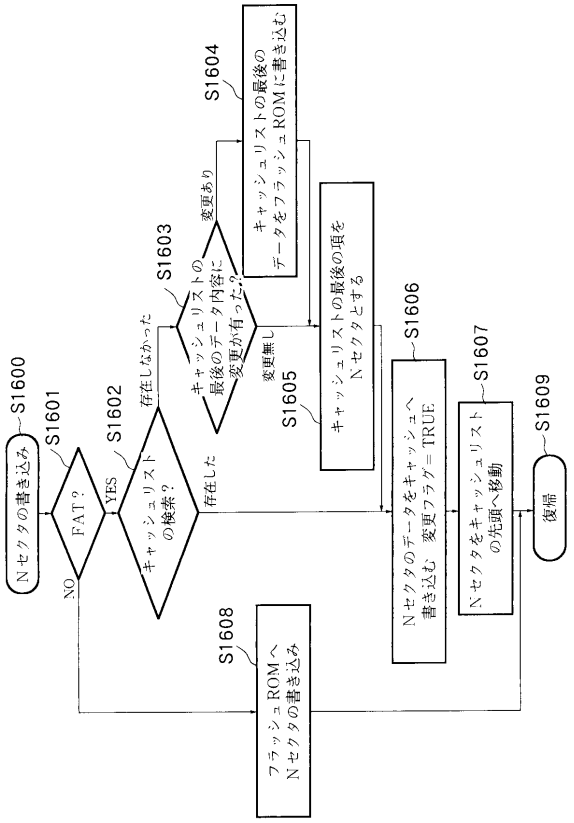
【図１２】



【図１３】



【図１４】



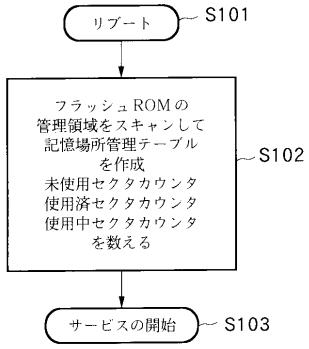
【図１５】

```
1 : void  WriteByteEEP (struct  DEV  * Dev,char  * Address,char  Data)
2 : {
3 :     while ( * Dev - > Address  ! = Dev - > Data) :
4 :     * Address = Data :
5 :     Dev - > Address = Address :
6 :     Dev - > Data    = Data :
7 :     RotateRdyQueue0) :
8 : }
9 : char  ReadByteEEP (struct  DEV  * Dev,char  * Address)
10 : {
11 :     if (Dev - > Address = = Address) return  Dev - > Data :
12 :     while ( * Dev - > Address  ! = Dev - > Data) :
13 :     return  * Address :
14 : }
```

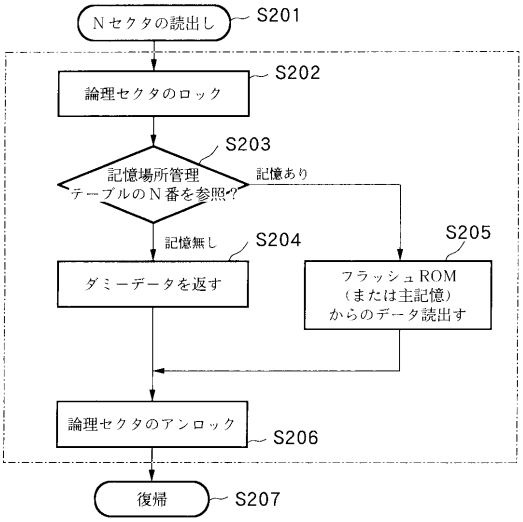
【図１６】

```
1 : void  ZoomUp (void)
2 : {
3 :     WaitSemapho (SemDCDC) :
4 :     MotorlStepUp (void) :
5 :     SignalSemapho (SemDCDC) :
6 : }
7 : char  XWriteSectorEEP (struct * Dev.int  Sector ,char  * Buffer)
8 : {
9 :     WaitSemapho (SemDCDC) :
10 :    WriteSectorEEP (Dev, Sector, Buffer)
11 :    While ( * Dev - > Address  ! = Dev - > Data) :
12 :    SignalSemapho (SemDCDC) :
13 : }
```

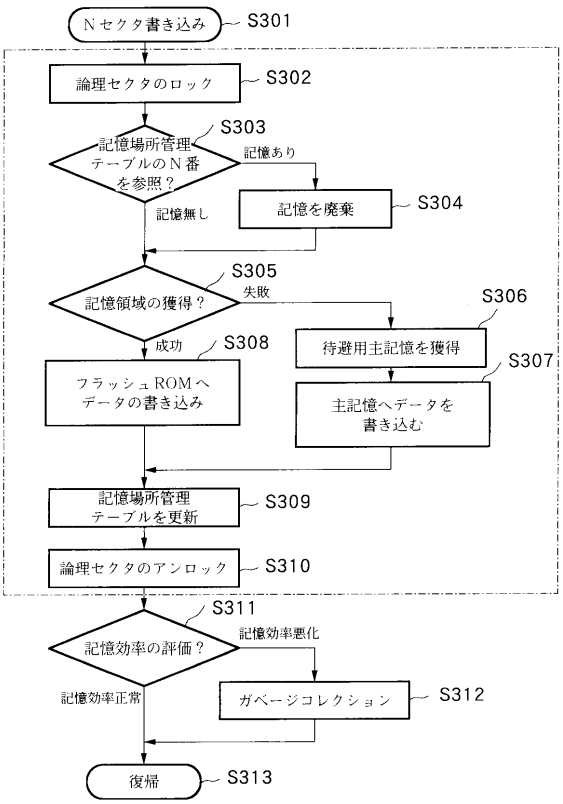
【図１７】



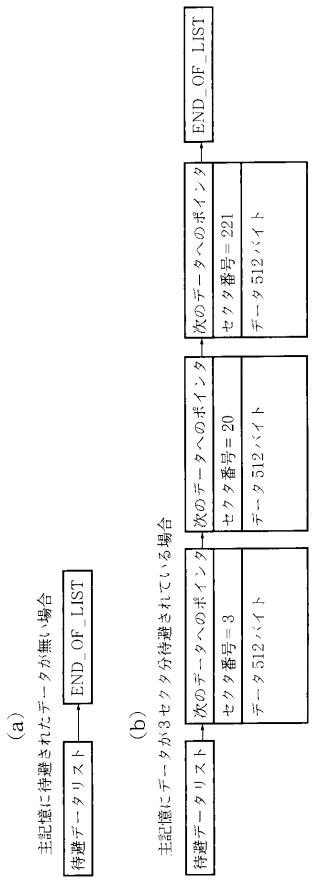
【図１８】



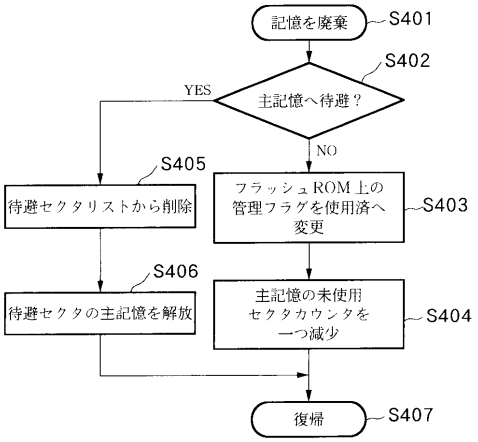
【図１９】



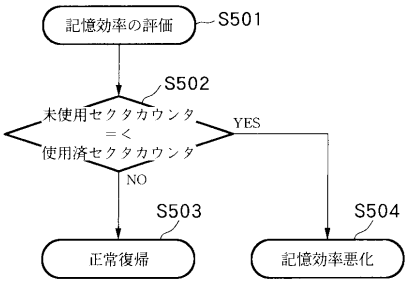
【図２０】



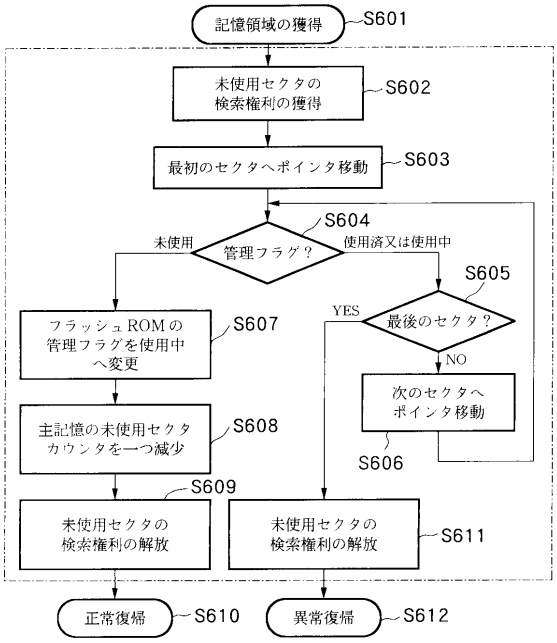
【図２１】



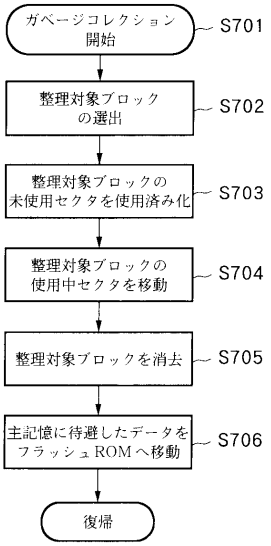
【図２２】



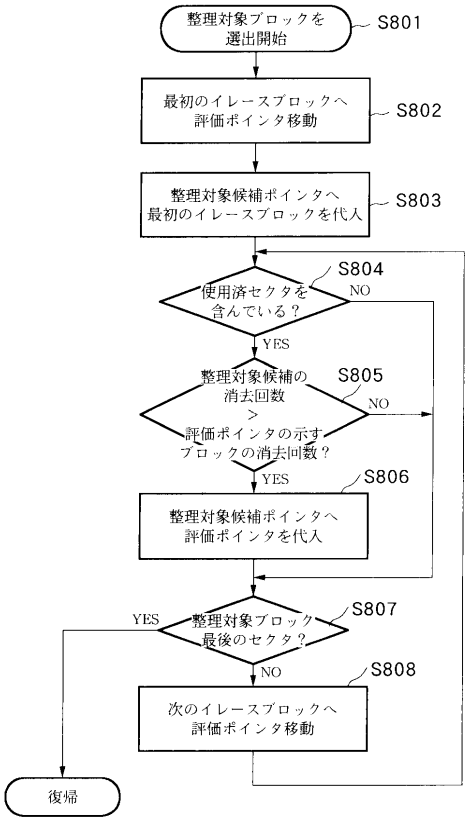
【図２３】



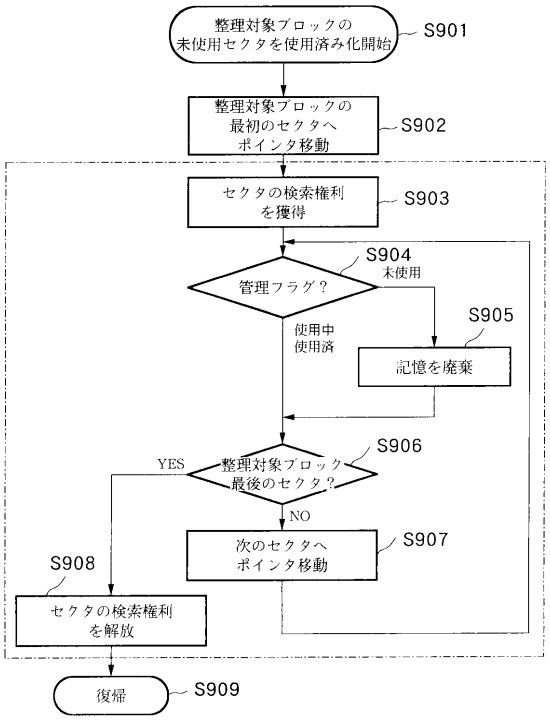
【図２４】



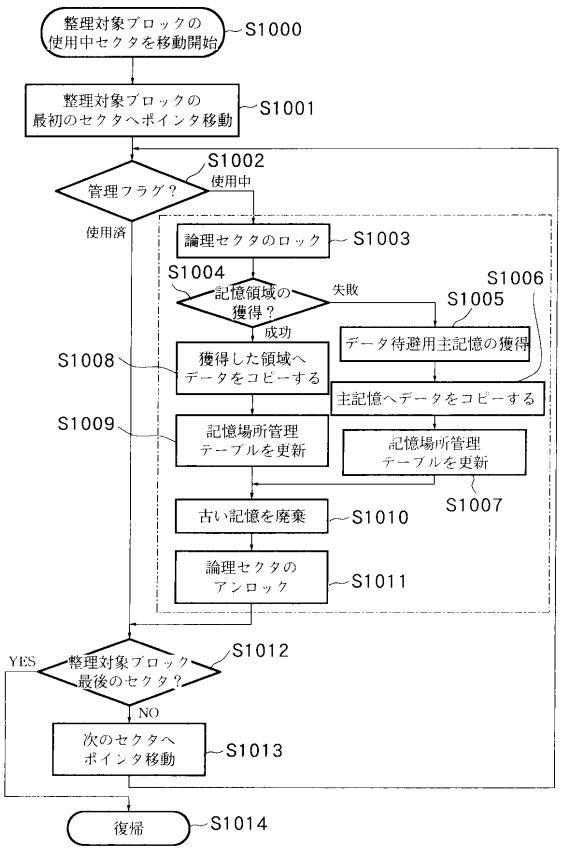
【図２５】



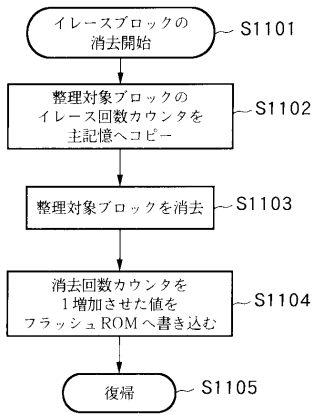
【図２６】



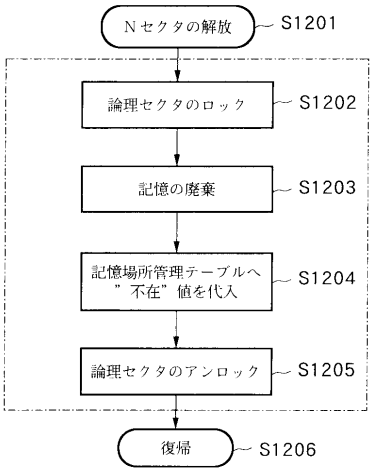
【図２７】



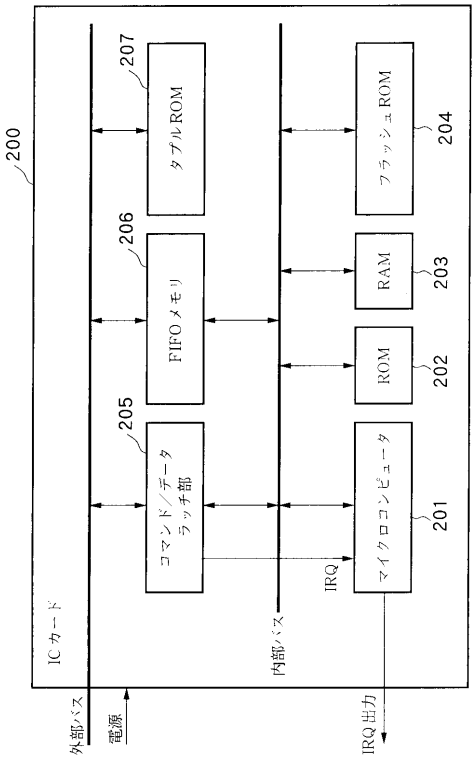
【図２８】



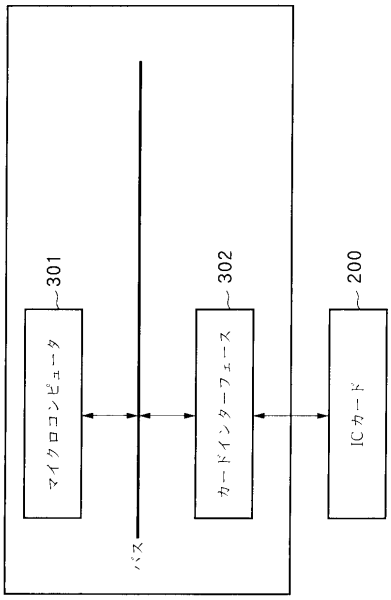
【図２９】



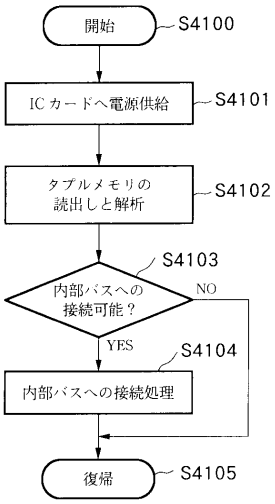
【図３０】



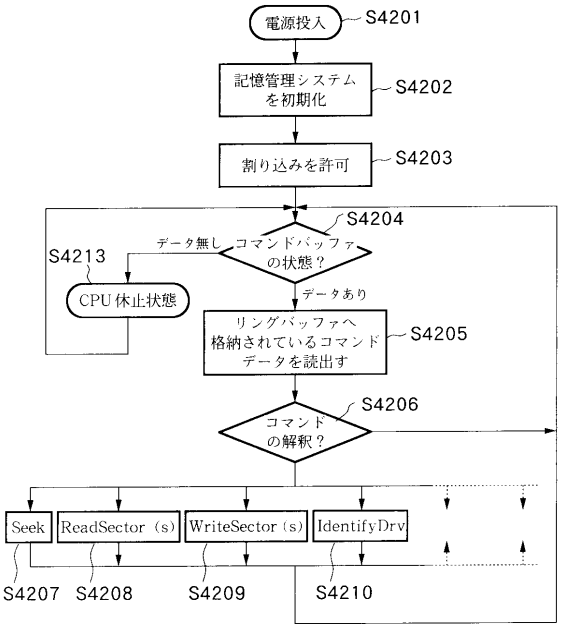
【図３１】



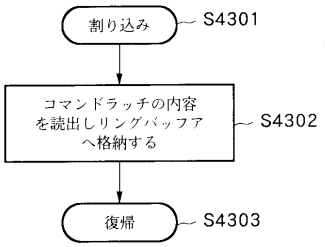
【図３２】



【図３３】



【図３４】



【図３５】

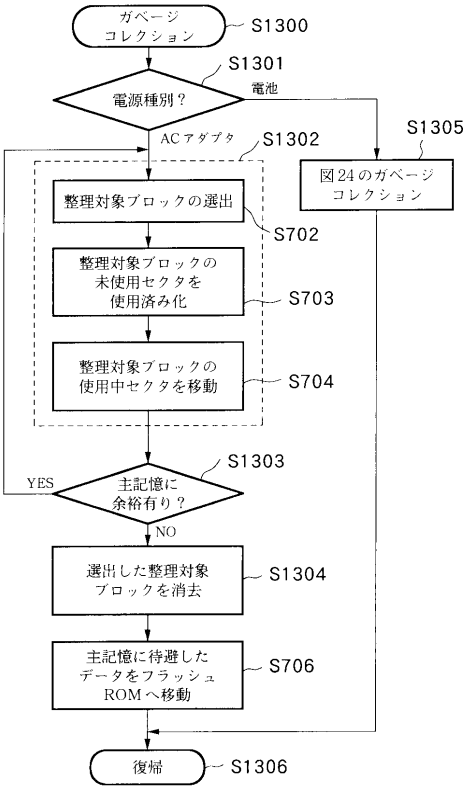
コマンド

読出し（In）	書き込み（Out）
Error	Features
SectorCount	SectorCount
SectorNumber	SectorNumber
CylinderLow	CylinderLow
CylinderHigh	CylinderHigh
Drive／Head	Drive／Head
Status	Command ←割り込み発生

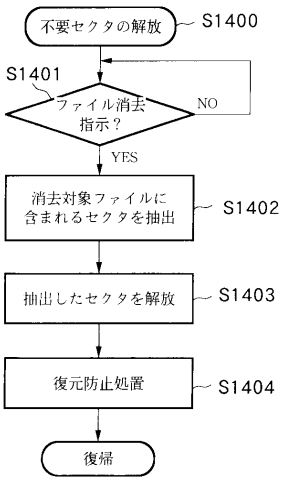
コントロール

読出し（In）	書き込み（Out）
Alt.Status	Device Ctl
DriveAddr	

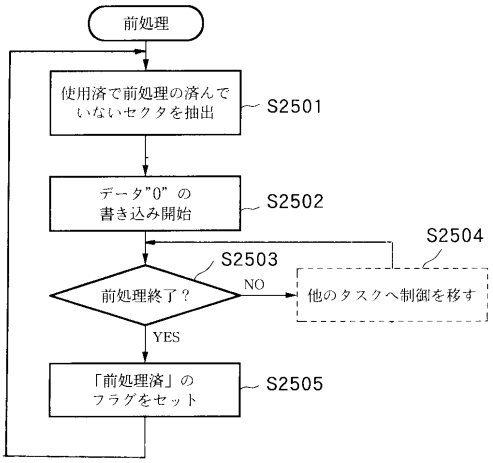
【図３６】



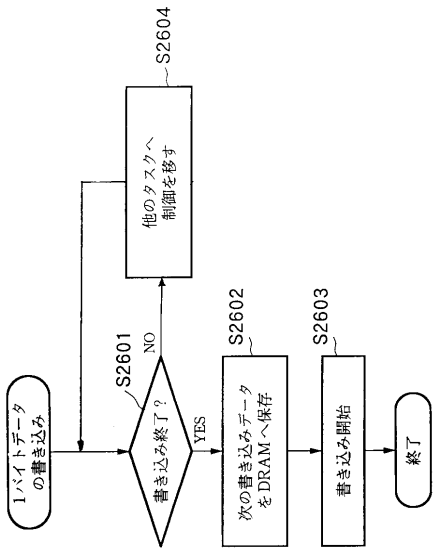
【図３７】



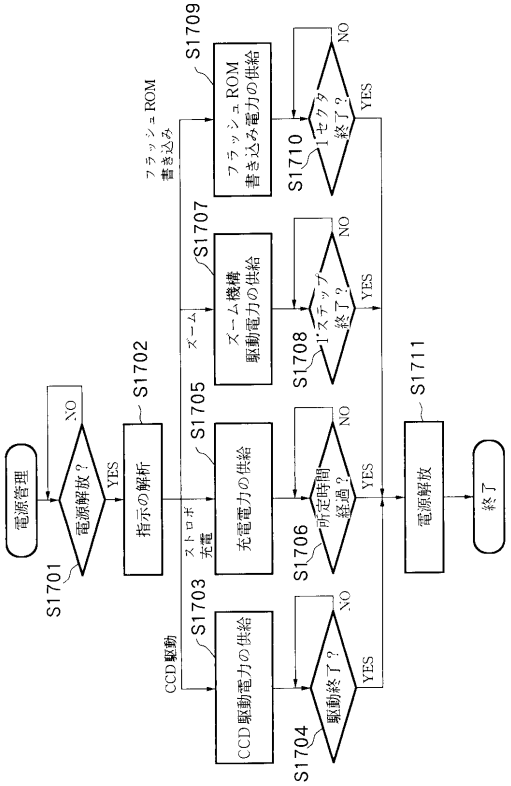
【図３８】



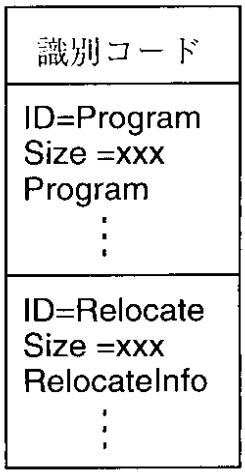
【図３９】



【図４０】



【図４１】



【図４２】

相対アドレスで表現されたプログラムコード

番地	データ	ニーモニック
0040	1234	DB 1234
0050	85,0010	JMP 0010
⋮		
0060	86,-20	MOV A.@(-20)
⋮		

【図４３】

プログラム中で絶対アドレスに変換しなければならない番地のテーブル

0052
0062
⋮

【図４４】

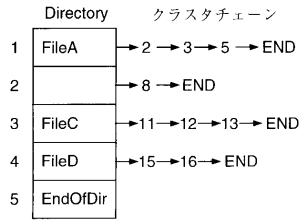
主記憶の8710番地へマッピングされたプログラムコード

番地	データ	ニーモニック
8750	1234	DB 1234
8760	85,8770	JMP 8770
⋮		
8770	86,8750	MOV A.@(8750)
⋮		

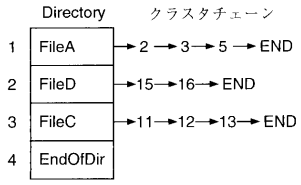
【図４５】

	Directory	クラスタチェーン
1	FileA	→2→3→5→END
2	FileB	→8→9→10→END
3	FileC	→11→12→13→END
4	FileD	→15→16→END
5	EndOfDir	

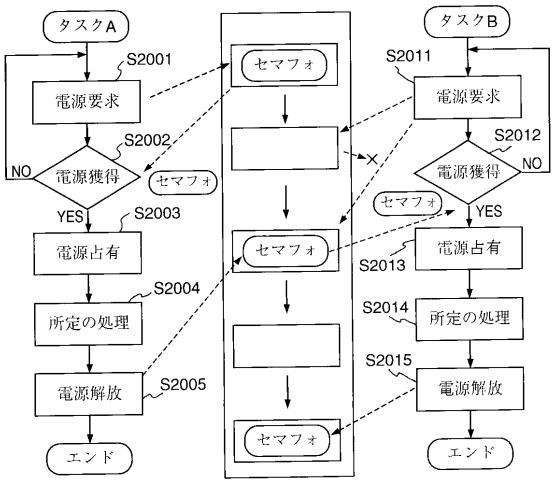
【図４６】



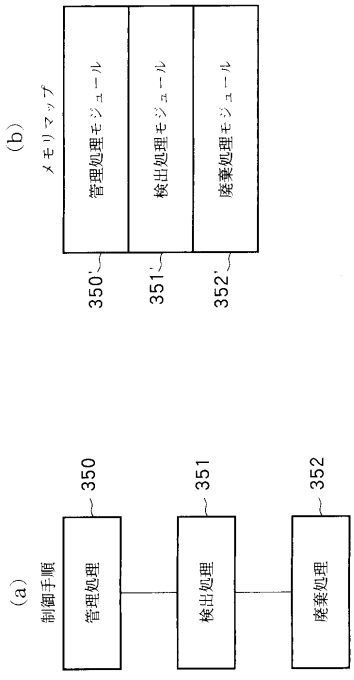
【図４７】



【図４８】



【図４９】



フロントページの続き

(51)Int.Cl.[7] FI
G06F 12/02 510A

(56)参考文献 特開平05－027924（JP，A）
特開昭64－053241（JP，A）
特開平07－013872（JP，A）
特開平06－348573（JP，A）
特開平06－139140（JP，A）

(58)調査した分野(Int.Cl.[7]，DB名)
G06F 3/06-08,
12/00-08,
12/14-16,
G11C16/02