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

特許第4078360号
(P4078360)

(45)発行日　平成20年4月23日(2008.4.23)　(24)登録日　平成20年2月8日(2008.2.8)

(51)Int.Cl.		F I		
G06F 9/445	(2006.01)	G06F	9/06	610K
G06F 9/54	(2006.01)	G06F	9/06	640B
G06F 12/06	(2006.01)	G06F	12/06	522B
G06F 12/10	(2006.01)	G06F	12/10	505B
		G06F	12/10	507B

請求項の数 7　(全 28 頁)

(21)出願番号　特願2005-126957(P2005-126957)
(22)出願日　平成17年4月25日(2005.4.25)
(65)公開番号　特開2006-202252(P2006-202252A)
(43)公開日　平成18年8月3日(2006.8.3)
審査請求日　平成17年4月25日(2005.4.25)
(31)優先権主張番号　特願2004-374386(P2004-374386)
(32)優先日　平成16年12月24日(2004.12.24)
(33)優先権主張国　日本国(JP)
(31)優先権主張番号　特願2004-374765(P2004-374765)
(32)優先日　平成16年12月24日(2004.12.24)
(33)優先権主張国　日本国(JP)

(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　100090273
弁理士　國分　孝悦
(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　久保　光宏

最終頁に続く

(54)【発明の名称】電子機器、データ処理方法、及びコンピュータプログラム

(57)【特許請求の範囲】

【請求項１】

不揮発性メモリであるところの第１の記録媒体に記録されたプログラムの仮想アドレスと物理アドレスとをページ単位で管理するページテーブルを用いて、前記プログラムを実行する電子機器であって、

前記第１の記録媒体に記録されたプログラムの指定されたエリアに対して物理アドレスが割り当てられない状態となるように前記ページテーブルを設定する設定手段と、

前記プログラムが実行されることによって、前記物理アドレスが割り当てられていないエリアにアクセスされると、そのアクセスされた物理アドレスに対応する部分のプログラムのデータを前記第１の記録媒体から第２の記録媒体に複写する複写手段とを有することを特徴とする電子機器。

【請求項２】

前記プログラムが実行されることによって、前記物理アドレスが割り当てられていないエリアにアクセスされ、ページフォルトが発生すると、ページフォルト割り込み処理を実行する割り込み処理実行手段を有し、

前記複写手段は、前記割り込み処理実行手段により実行されるページフォルト割り込み処理の中で、前記アクセスされた物理アドレスに対応する部分のプログラムのデータを前記第１の記録媒体から第２の記録媒体にコピーすることを特徴とする請求項１に記載の電子機器。

【請求項３】

前記複写手段により複写された部分の複写先の物理アドレスを、前記ページテーブルに割り当てる割り当て手段を有することを特徴とする請求項１又は２に記載の電子機器。
【請求項４】
前記設定手段は、前記第１の記録媒体に記録されたプログラムに記述されている関数で指定されているエリアに対して物理アドレスが割り当てられないように前記ページテーブルを設定することを特徴とする請求項１～３の何れか１項に記載の電子機器。
【請求項５】
前記プログラムは、アプリケーションプログラムであることを特徴とする請求項１～４の何れか１項に記載の電子機器。
【請求項６】
不揮発性メモリであるところの第１の記録媒体に記録されたプログラムの仮想アドレスと物理アドレスとをページ単位で管理するページテーブルを用いて、前記プログラムを実行するデータ処理方法であって、
前記第１の記録媒体に記録されたプログラムの指定されたエリアに対して物理アドレスが割り当てられない状態となるように前記ページテーブルを設定手段が設定する設定ステップと、
前記プログラムが実行されることによって、前記物理アドレスが割り当てられていないエリアにアクセスされると、そのアクセスされた物理アドレスに対応する部分のプログラムのデータを前記第１の記録媒体から第２の記録媒体に複写手段が複写する複写ステップとを有することを特徴とするデータ処理方法。
【請求項７】
不揮発性メモリであるところの第１の記録媒体に記録されたプログラムの仮想アドレスと物理アドレスとをページ単位で管理するページテーブルを用いて、前記プログラムを実行するに際し、
前記第１の記録媒体に記録されたプログラムの指定されたエリアに対して物理アドレスが割り当てられない状態となるように前記ページテーブルを設定する設定ステップと、
前記プログラムが実行されることによって、前記物理アドレスが割り当てられていないエリアにアクセスされると、そのアクセスされた物理アドレスに対応する部分のプログラムのデータを前記第１の記録媒体から第２の記録媒体に複写する複写ステップとをコンピュータに実行させることを特徴とするコンピュータプログラム。
【発明の詳細な説明】
【技術分野】
【０００１】
本発明は、電子機器、データ処理方法、及びコンピュータプログラムに関し、特に、電子機器をスタートアップさせるために用いて好適なものである。
【背景技術】
【０００２】
近年、デジタルカメラなどの分野では、ウインドウシステムを使ったＧＵＩ（Graphical User Interface；グラフィカルユーザーインターフェース）などを含む大規模なソフトウェアをＲＯＭ（Read Only Memory）に組み込んで実行させる事が行われるようになった。このようなシステムでは３２ビットのＲＩＳＣ（Reduced Instruction Set Computer）プロセッサや大容量のＤＲＡＭ（Dynamic Random Access Memory）を小型の電池によって駆動しなければならない。
【０００３】
このため、使用しないときは電源を完全にＯＦＦにしておき、使用する直前に電源を入れるのが普通である。デジタルカメラの場合、撮影したいタイミングを逃さないためには、電源投入から使用可能になるまでの時間ができるだけ短いほうがよい。したがって、システムのスタートアップ時間が短くなるようなコンピュータシステムが望まれている。システムのスタートアップ時間の中には、ＯＳ（Operating System）やドライバーの初期化に加えて、Ｃ言語のような高級言語のデータセグメントの初期値を転送する部分も含まれ

る。これらは、デジタルカメラとしての動作上、初期化以前に必要なものであり、オーバーヘッドと呼ばれるものである。また、ＲＯＭからの高級言語の起動に関する従来技術として特許文献１がある。

【０００４】

このように、Ｃ言語をはじめとする高級言語を使ったＲＯＭ化プログラムを構成する場合、高級言語として動作を可能とするための初期化が必要となる。中でも変数領域に初期値を設定する時間は、プログラムの規模に応じて長くなる傾向にある。Ｃ言語では初期化付き静的変数と呼ばれるものがそれ相当する。変数領域は、データセグメントと呼ばれるエリアで、プログラマによってＲＡＭの所定の番地にアサインされる（割り当てられる）。

【０００５】

初期値はＲＯＭの所定の番地に書き込まれており、Ｃ言語のスタートアップルーチンによってＲＯＭからＲＡＭのデータセグメントへコピーされる。またＣ言語の場合、初期値がアサインされていない静的変数は０に初期化されるという仕様になっており、それらの変数はＢＳＳ領域と呼ばれるＲＡＭ上のエリアに配置される。ＲＯＭからＲＡＭへの初期値の転送に加えて、ＢＳＳ領域に配置されている変数をゼロ（０）にすること（０クリア）もスタートアップの際に時間のかかる処理である。

【０００６】

この他に、アニメーションを多用したＧＵＩの表現や、ソフトウェアによるファイルの圧縮やプリンティングのための画像処理など、ＣＰＵによる高速な処理が要求される部分も多い。

【０００７】

＜ＲＯＭ・ＲＡＭの速度格差＞

ＲＩＳＣプロセッサのクロックは、年々高速化する傾向にあり、１００Ｍｈｚや２００Ｍｈｚといった高速な動作を行うことも珍しくない。しかし、ＲＯＭのアクセススピードは１２０ナノ秒程度とプロセッサスピードに対して非常に遅い。また、カメラを小型化するために３２ビットＢＵＳで接続するのではなく１６ビットＢＵＳを使って接続する場合もあり、１つのマシン命令をフェッチするために３００ナノ秒程度も必要となる事も珍しくない。

【０００８】

一方、ＳＤＲＡＭなどのＲＡＭからの読み出しスピードは、バースト読み出しが可能なため、ＲＯＭからの命令フェッチに対して非常に高速である。このようなシステムではＣＰＵとメモリーとの間にキャッシュメモリーが存在する。このため、ＤＲＡＭとキャッシュとの間は常にバーストモードでの読み書きとなり、平均的に内部スピードの倍から数倍程度（１０～数十ナノ秒程度）のスピードでフェッチできる事になる。つまり、キャッシュがミスヒットしたときにＲＡＭから読み出す場合、ＲＯＭから読み出す場合に比較して１０倍程度高速に読み出す事が出来る事になる。

【０００９】

このような事情からＲＯＭのプログラムを一旦すべてＲＡＭへコピーするという手法が用いられる事が多い。しかし、起動時にＲＯＭのプログラムをすべてＲＡＭへコピーしなければならないため、起動時間が数百ミリ秒から十数秒と非常に長くなってしまうという欠点がある。そのため、デジタルカメラのように起動時間を短くしなければならない装置には向いていなかった。

【００１０】

また、変数の初期値をＲＯＭからＲＡＭへコピーする時間は短縮する事が難しく、スタートアップ時間の非常に大きな要素となっている。概略その時間は数百ミリ秒にも及ぶ。その間、コンピュータシステムは有効なユーザープログラムどころかＯＳの初期化すら実行できなかった。

【００１１】

また、パーソナルコンピュータのＢＩＯＳ（Basic Input/Output System）も同様な理

由でＲＡＭへコピーして実行するという手法が行われている。例えば、特許文献２に記載されているようなＢＩＯＳのシステムが考案されている。かかるシステムは、一般に、シャドウＢＩＯＳとよばれている手法で、ＢＩＯＳに含まれる描画ルーチンなどをＲＡＭに転送して実行する事で実現される。シャドウＢＩＯＳは、ＭＭＵ（Memory Management Unit；メモリーマネージメントユニット）の機能を使って元のＲＯＭのアドレスにＲＡＭを配置して動作させている。しかしながら、このシャドウＢＩＯＳの技術も前述した技術と同様に、スタートアップ時にＲＡＭへの転送を明示的に行う事から、デジタルカメラに向いた技術ではなかった。

【００１２】

【特許文献１】特開２００４－３４８６７７号公報

【特許文献２】特開２００１－５１８５８号公報

【発明の開示】

【発明が解決しようとする課題】

【００１３】

以上のように従来の技術では、ＲＯＭに組み込まれた大規模なソフトウェアを実行するデジタルカメラなどの電子機器のスタートアップ時間を適切に短くすることが困難であるという問題点があった。

本発明は、このような問題点に鑑みてなされたものであり、電子機器のスタートアップ時間を飛躍的に短縮させるようにすることを目的とする。

【課題を解決するための手段】

【００１４】

本発明の電子機器は、不揮発性メモリであるところの第１の記録媒体に記録されたプログラムの仮想アドレスと物理アドレスとをページ単位で管理するページテーブルを用いて、前記プログラムを実行する電子機器であって、前記第１の記録媒体に記録されたプログラムの指定されたエリアに対して物理アドレスが割り当てられない状態となるように前記ページテーブルを設定する設定手段と、前記プログラムが実行されることによって、前記物理アドレスが割り当てられていないエリアにアクセスされると、そのアクセスされた物理アドレスに対応する部分のプログラムのデータを前記第１の記録媒体から第２の記録媒体に複写する複写手段とを有することを特徴とする。

【００１５】

本発明のデータ処理方法は、不揮発性メモリであるところの第１の記録媒体に記録されたプログラムの仮想アドレスと物理アドレスとをページ単位で管理するページテーブルを用いて、前記プログラムを実行するデータ処理方法であって、前記第１の記録媒体に記録されたプログラムの指定されたエリアに対して物理アドレスが割り当てられない状態となるように前記ページテーブルを設定手段が設定する設定ステップと、前記プログラムが実行されることによって、前記物理アドレスが割り当てられていないエリアにアクセスされると、そのアクセスされた物理アドレスに対応する部分のプログラムのデータを前記第１の記録媒体から第２の記録媒体に複写手段が複写する複写ステップとを有することを特徴とする。

【００１６】

本発明のコンピュータプログラムは、不揮発性メモリであるところの第１の記録媒体に記録されたプログラムの仮想アドレスと物理アドレスとをページ単位で管理するページテーブルを用いて、前記プログラムを実行するに際し、前記第１の記録媒体に記録されたプログラムの指定されたエリアに対して物理アドレスが割り当てられない状態となるように前記ページテーブルを設定する設定ステップと、前記プログラムが実行されることによって、前記物理アドレスが割り当てられていないエリアにアクセスされると、そのアクセスされた物理アドレスに対応する部分のプログラムのデータを前記第１の記録媒体から第２の記録媒体に複写する複写ステップとをコンピュータに実行させることを特徴とする。

【発明の効果】

【００１７】

本発明によれば、電子機器の動作開始までの時間を大幅に短縮する事が可能となる。

【発明を実施するための最良の形態】

【００１８】

次に、図面を参照しながら、本発明の実施形態について説明する。

（第１の実施形態）

まず、本発明の第１の実施形態について説明する。

＜ＭＭＵ＞

近年、ＬＳＩ（Large Scale Integration）の集積度が上がり、ＭＭＵ（Memory Management Unit；メモリーマネージメントユニット）またはページングハードウェアを内蔵したＳＯＣ（System On Chip；システムオンチップ）を搭載することが可能となった。後述するように、本実施形態は、ページングハードウェアを応用した技術を用いるようにしている。そこで、まずページングハードウェアについて説明する。

【００１９】

＜仮想記憶＞

本来のページングハードウェアの目的は、外部記憶装置を使って主記憶を拡張する仮想記憶ができるようにすることである。ページングハードウェアは、プロセッサとメモリーとの間に介在して動作し、プログラムのアドレス空間を固定サイズ（例えば４Ｋバイトとか８Ｋバイト）のページに分割し、コンピュータの物理アドレスを同じサイズのページフレームに分割する。

【００２０】

ページングハードウェアには、図１のようなアドレス空間のページごとに１つのエントリを持つページテーブルがある。このページテーブルエントリには、そのページに対応した実際のページフレームの物理アドレスか、ページが存在しない（ページフレームが割り当てられていない）事を示すフラグが格納されている。

【００２１】

存在しないページ（割り当てられていないページフレーム）をプロセッサがアドレッシングしようとすると、ページフォルト割り込みが発生する。ページフォルトの割り込みを処理するプログラムは、対応する外部記憶装置の内容を主記憶に読み出し、フレームとして割り当てた後、制御を戻し、元のプログラムを再開させる。ＬＲＵ（最長不使用）のアルゴリズムを用いて外部記憶装置の使用頻度の高い部分を主記憶に割り付ける事によって、アプリケーションから見れば実際に存在する主記憶よりも大きなメモリーが存在しているかのように動作する事が可能となる。詳細は「OSの基礎と応用」A.S.タネンバウム＝著/引地信之、引地美恵子＝訳、トッパンなどで解説されている。

【００２２】

＜ポジションディペンデット＞

マシン語にコンパイルされたプログラムは、一般にポジションディペンデットという特徴を備えている。プログラムの置かれている番地を移動してしまうと、プログラムは正しく動作しない事を意味する。これによって、ＲＯＭに搭載されているプログラムの一部をそのままＲＡＭへ転送して実行するという事は出来ない。ポジションに依存しないがそのまま実行不能な状態のプログラムをリロケータブルなオブジェクトと呼ぶ。

【００２３】

リロケータブルなオブジェクトは、ポジションが決定した時にポジションにあわせてプログラムを実行可能な形態に変更するための冗長な情報を含んでいる。また、リロケータブルなオブジェクトを実行可能な状態にするためにローダー、またはリロケータというプログラムを用いるが、そのようなプログラムによるＲＡＭへのプログラム配置は、非常に時間のかかる処理であるため、デジタルカメラのスタートアップで行うには負荷が重い処理である。

【００２４】

＜ＭＭＵの応用＞

本実施形態では、ＭＭＵの仮想アドレスという特徴に着目し、ＣＰＵからソフトウェア

的に見れば同一のポジションでありながら、物理的にはＲＡＭをアサインする事で、ＲＯＭのプログラムの一部だけを、ＲＡＭに配置して、高速なプログラムの実行を実現する。

【００２５】

＜要求時シャドウアサイン＞

本実施形態では、スタートアップ時にＲＯＭからＲＡＭへプログラムの転送を行う替わりに、アプリケーションの実行中に必要となった部分、すなわち実際に実行された部分だけをＲＡＭへ転送して実行することで、高速なプログラムの実行を実現するようにしている。

【００２６】

＜デジタルカメラ＞

図２は、本発明の第１の実施形態を示し、デジタルカメラのハードウェア構成の一例を示した図である。

図２において、１００１は、プログラムを読み込み実行するプロセッサである。１００２は、仮想アドレスと物理アドレスとを管理するページングハードウェアである。１００３は、デジタルカメラのプログラムを格納しているＲＯＭである。１００４は、画像や変数を格納するＲＡＭである。１００５は、撮像された画像を取り込むためのＡ／Ｄ変換機である。１００６は、レンズ１００７から入った光を受けて電気信号に変換するＣＣＤ（Charge-Coupled Device）である。１００７は、ＣＣＤ１００６に画像を決像させるレンズである。１００８は、メモリーカード１００９を接続するためのインターフェースである。１００９は、画像を格納するメモリーカードである。１０１１は、周辺回路を駆動するためのＩ／Ｏインターフェースである。１０１０は、レンズ１００７を駆動するためのレンズ駆動メカである。

【００２７】

電源が投入されると、プロセッサ１００１は、ページングハードウェア１００２を通してＲＯＭ１００３に格納されたプログラムのリセットベクターと呼ばれる番地の命令を読み出し、実行を開始する。ページングハードウェア１００２は、リセット直後の初期状態となっており、物理アドレスと仮想アドレスとが一致した状態となっている。なお、画像を撮影して、メモリーカード１００９へ記録する過程は本実施形態の本質と関係がないため、説明を省略することとする。

【００２８】

＜初期状態＞

図３は、本実施形態のデジタルカメラにおけるシステムが起動した直後のページングハードウェア１００２と、ＲＯＭ１００３と、ＲＡＭ１００４の状態の一例を示した図である。図３に示すように、システムの起動直後は、すべての仮想アドレスに物理アドレスがアサインされている（割り当てられている）状態である。

【００２９】

＜アプリケーションプログラム＞

図４は、アプリケーションプログラムの一例である（Ｃ言語の）ソースコードを示した図である。図４のプログラム３０１において、１００行目はこのプログラム３０１を呼び出す最初の関数である。プログラム３０１は、システムが起動するときに呼ばれるというものではなく、モジュールが使用される時に呼ばれるものでも良い。例えばプリンティング関連のモジュールであれば、プリンタを接続して初めて呼ばれるようにしてもよい。

【００３０】

そして、１０２行目は、本実施形態の特徴的なプログラムの１つであるシャドウを作成する関数CreateShadowを呼び出している。この関数CreateShadowのパラメータは開始番地とサイズである。開始番地として、１０５行目の関数のアドレスを指定している。そして、サイズとして、１１３行目の関数EndModuleと１０５行目の関数MyFirstFuncとのアドレス差を計算して与えている。

【００３１】

図５は、図４のプログラム３０１の１０２行目で呼び出している関数CreateShadowの動

作の一例を説明するフローチャートである。
図５において、まず、ステップＳ８０１で処理を開始し、ステップＳ８０２で割り込みを禁止する。そして、ステップＳ８０３で、関数CreateShadowのパラメータで指定されたアドレスを含むページであって、物理アドレスと一致しているページがすべて未アサインの状態になるようにページメモリを設定する。そして、ステップＳ８０４で、割り込み禁止を解除し、ステップＳ８０５で処理を終了する。
【００３２】
図５のステップＳ８０５の処理を終えた時点、すなわち図３のプログラム３０１の１０２行目の実行を終えた時点において、ページングハードウェア１００２と、ＲＯＭ１００３と、ＲＡＭ１００４の状態は、図６のような状態となっている。関数MyFirstFuncから関数EndModuleまでの関数を含むページはすべて未アサインの状態となっている。図６では、関数MyFuncの物理アドレスはアサインされていない状態となっている事が分かる。
【００３３】
次に、図４のプログラム３０１の１０９行目の関数MyFuncが呼び出されると、ページフォルト割り込みが発生する。ここで、図７のフローチャートを参照しながら、ページフォルト割り込み処理の一例を説明する。
図７のステップＳ９０１で割り込み処理を開始し、ステップＳ９０２で空きフレームを確保する。そして、ステップＳ９０３でＲＯＭ１００３の該当ページの内容を、ステップＳ９０２で確保した空きフレームにコピーする。
図８は、ＲＯＭ１００３の該当ページの内容を、空きフレームにコピーした時点における、ページングハードウェア１００２と、ＲＯＭ１００３と、ＲＡＭ１００４の状態の一例を示した図である。図８では、関数MyFuncのコピーがＲＡＭ１００４上のフレームに作成されている事が分かる。
【００３４】
このようにして、ＲＯＭ１００３の該当ページの内容を空きフレームにコピーしたら、図７のステップＳ９０４で、ページテーブルにフレームを設定する。図９は、ステップＳ９０４でページテーブルにフレームをアサインした時点における、ページングハードウェア１００２と、ＲＯＭ１００３と、ＲＡＭ１００４の状態の一例を示した図である。なお、ＲＡＭ１００４にコピーされたプログラムは元のアドレスと同じ仮想アドレスから読み出されるため、ポジションディペンドなプログラムでも実行可能である。
そして、ステップＳ９０５でページフォルト割り込みを終了する。
【００３５】
以上のようなページフォルト割り込みから復帰して、プログラム３０１を再開し図３の１０９行目が実行される時には、プログラム３０１そのものはＲＡＭ１００４からフェッチするため、高速な読み出しが可能な状態となっている。図４のプログラム３０１の１０９行目が再び実行される時には、すでにフレームがアサイン済みのためページフォルト割り込みは発生しない。
【００３６】
図１０は、関数EndModuleを含むページについても前述した処理を実行してフレームをアサインした時点における、ページングハードウェア１００２と、ＲＯＭ１００３と、ＲＡＭ１００４の状態の一例を示した図である。
【００３７】
以上のように構成する事によって、システム起動時にＲＯＭ１００３からＲＡＭ１００４へプログラムを転送する待ち時間のロスがなく、且つ、ポジションディペンドなプログラムの一部をＲＡＭ１００４へ展開する事が可能となり、大規模なＲＯＭプログラムの一部を、高速に実行することが可能なＲＡＭ１００４で実行する事を容易に実現するコンピュータシステムを構成する事が出来る。
【００３８】
特に、デジタルカメラの起動時間を構成する要素には、レンズ１００７の初期位置への移動や、メモリーカード１００９からのデータ読み出しなど、プロセッサ（ＣＰＵ）１０

０１の待ち時間が多い。一方、ウインドウシステムを使用したＧＵＩ（グラフィカルユーザーインターフェース）などは、プロセッサ（ＣＰＵ）１００１の処理時間を多く消費し、起動時にはまったく動作しないプログラムである。

したがって、アプリケーションプログラムの要求時、すなわち実行時にＲＯＭ１００３からＲＡＭ１００４へアプリケーションプログラムの転送を行う事は効率的といえる。そのように構成する事で、スタートアップ時に必要なＲＯＭ１００３からＲＡＭ１００４に転送する時間を節約する事が出来、且つ高速動作が要求されるプログラムをＲＡＭ１００４に配置して動作させる事ができる。

【００３９】

（第２の実施形態）

次に、本発明の第２の実施形態について説明する。なお本実施形態の説明において、前述した第１の実施形態と同一の部分については、図１～図１０に付した符号と同一の符号を付す等して詳細な説明を省略する。

【００４０】

＜要求時データセグメントコピー＞

本実施形態においても、前述した第１の実施形態と同様に、ページングハードウェアを応用している。そして、コンピュータ言語のスタートアップ時の変数初期化に必要な時間を短縮する事で、高速に起動可能な組み込みソフトウェアのフレームワークを提供するようにしている。

【００４１】

パワーオンリセットが解除され、コールドスタートをする最初のプログラムとして、初期化プログラムが動作し、ページングハードウェアのページテーブルを初期化する。その際、データセグメント及びＢＳＳのアドレスとなるエリアに対して、ページフレームをアサインしていない状態として設定を行う。

【００４２】

そして、コンピュータ言語の変数初期化ルーチンをスキップし、変数の初期化を行わない状態のまま、コンピュータ言語の最初のプログラムへのエントリーポイントへジャンプする。コンピュータ言語で書かれたプログラムが、静的変数へアクセスすると、ページフォルトが発生し、ページフォルト割り込みプログラムが呼び出され、そのプログラムが起動する。ページフォルトが発生したページに対してページフレームをアサインする。ページフォルトが発生したページがＢＳＳ領域であればページフレーム内のメモリーの内容をゼロ（０）に初期化し、元のプログラムへ制御を戻す。ページフォルトが発生したページがデータセグメント領域であればページフレーム内のメモリーに対応するＲＯＭから初期値をコピーし、元のプログラムへ制御を戻す。本実施形態は、このようなプログラムを用いて実現される。以下に、本実施形態の詳細を図面と共に説明する。

【００４３】

デジタルカメラのハードウェア構成は、例えば、前述した第１の実施形態と同じである。また、電源が投入されると、プロセッサ１００１が、ＲＯＭ１００３に格納されたプログラムのリセットベクターと呼ばれる番地の命令を読み出して実行を開始することや、ページングハードウェア１３０２が、リセット直後の初期状態となっており、物理アドレスと仮想アドレスとが一致した状態となっていることも前述した第１の実施形態と同じである。なお、画像を撮影して、メモリーカード１００９へ記録する過程は本実施形態の本質と関係がないため、説明を省略することとする。

【００４４】

＜変数＞

図１１は、Ｃ言語で書かれたプログラムの一例を示した図である。また図１２は、図１１のプログラム１１０１をコンパイルした結果のアッセンブラリストの一例を示した図である。図１１のプログラム１１０１において、１００行目は初期化つき変数というもので、プログラム１１０１がスタートするときにはすでに「fFirstAccess」という変数に「TRUE」という値が代入されているという意味である。

【００４５】
「fFirstAccess」という変数は、図１２のアッセンブラリスト１２０１では、０００２行目に記述されている。０００１行目から０００３行目の間がデータセグメントのエリアである。コンパイルの結果、このプログラム３０１のデータセグメントは「fFirstAccess」だけである。ところが、その後リンカーによって全プログラムのデータセグメントが一つのエリアに集められるため、データセグメントには複数のソースによって記述された、モジュールの異なる変数が隣接して存在する事となる。
【００４６】
図１１のプログラム１１０１の１０１行目には、「Ｃｏｕｎｔ」という変数が宣言されているが、初期値は代入されていない。Ｃ言語の仕様で、初期値が宣言されていない変数は０で初期化される事になっているので、「Ｃｏｕｎｔ」の初期値は０となるべきである。図１２のアッセンブラリスト１２０１における０００７行目が「Ｃｏｕｎｔ」のコンパイル結果である。図１２のアッセンブラリスト１２０１における０００６行目から０００８行目は、ＢＳＳ領域であるため、「Ｃｏｕｎｔ」がＢＳＳ領域の変数としてコンパイルされた事が分かる。ＢＳＳ領域もデータセグメントと同様にリンカーによってまとめられ、複数のモジュールの変数が隣接して存在する事になる。
【００４７】
＜従来のスタートアップ＞
本実施形態のスタートアップを説明する前に従来のスタートアップをもう一度説明する事とする。図１３は、従来の技術を示し、スタートアップする際の動作を説明するフローチャートである。
ステップＳ１１０１でリセットが解除されると、ステップＳ１１０２でＢＳＳエリアに０（ゼロ）を書き込み、ＲＯＭからデータセグメントに対して初期値を転送する。そして、ステップＳ１１０３でＣ言語のプログラムを実行しはじめる。
ステップＳ１１０４でＣ言語のプログラムとしてＯＳをスタートする。そして、ＯＳが立ち上がると、ステップＳ１１０５でアプリケーションプログラムがスタートする。デジタルカメラとしての動作を開始できるのは、ステップＳ１１０５からである。ステップＳ１１０１からステップＳ１１０５にくるまでに、数百ミリ秒の時間のロスがある。
【００４８】
＜本実施形態のスタートアップ＞
本実施形態におけるスタートアップする際の動作の一例を図１４のフローチャートを使って説明する。
ステップＳ１００１でリセットが解除されたら、ステップＳ１００２で図２に示したページングハードウェア１００２内のページテーブルの設定を行う。このとき、データセグメントとＢＳＳの両方のエリアに対して物理アドレスを割り当てない状態に設定する。そして、ステップＳ１００３でＣ言語をスタートさせ、ステップＳ１００４でＯＳをスタートさせ、ステップＳ１００５でアプリケーションプログラムをスタートさせる。
【００４９】
＜実行時の動作説明＞
図１４のフローチャートのステップＳ１００５において、アプリケーションプログラムが実行されていく様子を、図１１に示したプログラム１１０１を例に挙げて説明する。
図１１のプログラム１１０１の１０３行目からアプリケーションプログラムを実行していく。最初に実行される有効な行は１０５行目である。１０５行目は、変数「fFirstAccess」が真なら１０６行目を実行し、偽なら１０８行目を実行するという分岐のルーチンである。この行を実行する事によって１００行目で宣言している「fFirstAccess」という変数の値を読む事になる。図１５は、プロセッサ１００１によって変数「fFirstAccess」がアドレッシングされている様子を示した図である。図１４のステップＳ１００２によってページテーブルのデータセグメントエリアの物理空間とＢＳＳエリアの物理空間とがアサインされていない状態となっている。物理空間がアサインされていないアドレスをアドレッシングすると、ページフォルト割り込みが発生する。ページフォルト割り込みが発生す

ると、プログラムが起動する。
【００５０】
＜ページフォルト処理＞
ページフォルト割り込みによって動作するプログラムによる処理動作の一例を、図１６のフローチャートを使って説明する事とする。
まず、ステップＳ１２０１でページフォルト割り込みが発生し、プログラムが起動される。ステップＳ１２０２で、まだアサインされていない物理メモリーを確保する。
ステップＳ１２０３で、ページフォルトが発生したページがデータセグメントなのかＢＳＳなのかデータセグメントとＢＢＳとが混在しているのかを判定する。図１１に示したプログラム１１０１では、１０５行目を実行中なので、変数「fFirstAccess」はデータセグメントであり、ステップＳ１２０４へ分岐する。ステップＳ１２０４では、ＲＯＭ１３０３の対応アドレスから初期値を転送する。
【００５１】
図１７は、ＲＯＭ１００３に格納された変数「fFirstAccess」の初期値である１という値がＲＡＭ１００４の物理アドレスへ転送された様子の一例を示した図である。変数「fFirstAcces」以外にも同一ページ内の４Ｋバイトの初期値はすべて転送される。そして、ステップＳ１２０７でページテーブルに対して物理アドレスを設定する。
図１８は、ページテーブルに物理アドレスが設定された様子の一例を示した図である。そして、ステップＳ１２０８で、元のプログラムへ制御が戻る。図１８に示すように、変数「fFirstAccess」は物理メモリーにアサインされており、値として初期値である１が代入されている状態で、図１１に示したプログラム１１０１の１０５行目に制御が戻る。
【００５２】
変数「fFirstAccess」は真なので、図１１に示したプログラム１１０１の１０６行目を実行し、１０７行目で変数「fFirstAccess」に対して値「ＦＡＬＳＥ」が代入される。１０７行目の代入が行われるときには、変数「fFirstAccess」は物理メモリーへアサインされているため、再びページフォルトが発生することはない。
そして、１０９行目が実行されると、１０１行目で宣言した変数「Ｃｏｕｎｔ」がアクセスされる。変数「Ｃｏｕｎｔ」はＢＳＳエリアにあり、はじめてアクセスされるページのために、再びページフォルト割り込みが発生する。図１９は、ページフォルトが発生した時点の様子の一例を示した図である。以後の動作を再び図１６のフローチャートを用いて説明する。
【００５３】
図１６のステップＳ１２０１でページフォルト割り込みが発生し、ステップＳ１２０２で空きメモリーを確保する。ステップＳ１２０３でページアドレスを判断するが、変数「Ｃｏｕｎｔ」はＢＳＳエリアの変数のため、ステップＳ１２０６へ分岐する。ステップＳ１２０６では、ステップＳ１２０２で確保した物理メモリーをゼロ（０）で埋める。
図２０は、確保した物理メモリーをゼロ（０）に埋めた直後の様子の一例を示した図である。そして、ステップＳ１２０７で、ページテーブルへ物理アドレスをセットする。図２１は、ステップＳ１２０７でページテーブルを更新した直後の様子の一例を示した図である。
【００５４】
プロセッサ１３０１は、物理メモリーの「Ｃｏｕｎｔ」というゼロ（０）に初期化された変数をアクセスする事が出来る。ステップＳ１２０８で元のプログラムに制御が戻ると、図３のプログラム３０１の１０９行目に戻ってくる。この行では、変数「Ｃｏｕｎｔ」の初期値であるゼロ（０）が読まれ、その値がインクリメントされ代入される。代入されるときには物理メモリーがアサインされているので、再びページフォルト割り込みが発生する事はない。
【００５５】
＜バックグラウンド転送＞
上記のように構成する事で、データセグメント転送とＢＳＳの０クリアの待ち時間を省

いてＣ言語をスタートすることができる。そして、ユーザーアプリケーションを高速にスタートする事ができる。しかし、デジタルカメラとしてのスタートアップはユーザーアプリケーションが始まってからスタートする。

【００５６】

デジタルカメラとしては、撮像素子への電源投入や、レンズ１００７の初期位置への駆動、メモリーカード１００９からのファイル情報の読み出しなど、デバイスが反応する待ち時間を含む処理が多い。それらの待ち時間を利用して、未転送のデータセグメントへの初期値の転送や初期化されていないＢＳＳの初期化を行う事で、要求時転送のオーバーヘッドを極力減らす事が可能となる。

【００５７】

バックグラウンドで初期化を行うプログラムの動作の一例を図２２のフローチャートを使って説明する。

ステップＳ１４０１でタスクをスタートする。このタスクは最もプライオリティ（優先順位）の低いタスクで動作するように起動させる。ステップＳ１４０２でページテーブルを検索し、物理アドレスがアサインされていないアドレスを探す。そして、ステップＳ１４０３で物理アドレスがアサインされていないアドレスが存在するか否かを判断する。

【００５８】

実行を開始したときに全てのアドレスがアサイン済みという事はないはずなので、このステップＳ１４０３の判断の結果、ステップＳ１４０４へ進む。ステップＳ１４０４では空き物理メモリーを確保する。そして、ステップＳ１４１４でページアドレスを調べ、そのページアドレスがＢＳＳアドレスならステップＳ１４０７に進み、確保した物理メモリーをゼロ（０）で埋める。

【００５９】

また、ステップＳ１４１４でページアドレスを調べた結果、そのページアドレスがデータセグメントならステップＳ１４０６へ分岐し、データセグメントの初期値を、ＲＯＭ１００３から確保した物理メモリーへ転送する。

さらに、ステップＳ１４１４でページアドレスを調べた結果、そのページアドレスがＢＳＳとデータセグメントとが混在したものなら、ステップＳ１４０８へ分岐し、ＢＳＳについては、確保した物理メモリーをゼロ（０）で埋め、データセグメントについては、ＲＯＭ１００３から初期値を転送する。

【００６０】

ステップＳ１４０６またはＳ１４０７またはＳ１４０８を実行し終えた状態では確保した物理メモリーに対して有効な初期データが格納されている。そして、ステップＳ１４０９で割り込みを禁止し、ステップＳ１４１０で、該当ページに物理アドレスがまだアサインされていない事を再び確認する。なぜなら、このプログラムは最もプライオリティの低いタスクとして実行しているため、ステップＳ１４０２からＳ１４０９までを実行する間に他のプライオリティの高いプログラムが動作する事でアクセスが発生したページに関してはすでに物理メモリーがアサインされている可能性があるからである。

【００６１】

もし、該当ページに物理アドレスがすでにアサインされていた場合には、ステップＳ１４１１で物理アドレスを空きエリアとして開放し、ステップＳ１４１３で割り込み禁止を解除してステップＳ１４０２へ戻る。

一方、ステップＳ１４１０で該当ページに物理アドレスがまだアサインされていない場合には、ステップＳ１４１２でページテーブルに物理アドレスをアサインする。そして、ステップＳ１４１３で割り込み禁止を解除し、ステップＳ１４０２へ戻る。ステップＳ１４０９からＳ１４１３までの間は割り込み禁止状態にしてページテーブルへのアクセスを排他的に行う必要がある。ステップＳ１４０２からステップＳ１４１３までを繰り返し実行する事で、すべてのページに物理アドレスがアサインされる事になるため、いずれステップＳ１４０３からステップＳ１４０５へ分岐し、タスクは終了することになる。

【００６２】

以上のように本実施形態では、データセグメントとＢＳＳの両方のエリアに対し物理アドレスを割り当てない状態にして、ページングハードウェア１００２内のページテーブルの設定を行ってアプリケーションプログラムをスタートさせる。その後、物理アドレスが割り当てられていないアドレスをアドレッシングすると、ページフォルト割り込みが発生し、まだ物理アドレスが割り当てられていない物理メモリーを確保する。ページフォルトが発生したページがデータセグメントであれば、ＲＯＭ１００３の対応アドレスから前記確保した物理メモリーに初期値を転送する。一方、ページフォルトが発生したページがＢＳＳであれば、前記確保した物理メモリーをゼロ（０）で埋める。これにより、データセグメントの変数の転送や、ＢＳＳの０クリアのための待ち時間を省くことができ、プログラミング言語の変数領域の初期化を待たずに、アプリケーションプログラムの実行を開始することができる。したがって、例えば、デジタルカメラの電源投入から撮影可能になるまでの時間を大幅に短縮する事が可能となる。

【００６３】

なお、本実施形態では仮想アドレスを持ったページングハードウェア１００２を使用したが、仮想アドレスと物理アドレスとの変換機能をもたない、プロテクションのみを目的としたＭＭＵを使って類似したシステムを構成できる。その場合、データセグメントとＢＳＳとをプロテクション対象として登録した状態で、Ｃ言語をスタートさせ、プロテクションフォルトの割り込みによってデータセグメントへのＲＯＭからの初期値の転送やＢＳＳの初期化を行うように構成すればよい。その他の構成については、上述した第２の実施形態と同じであるので説明を省略する。

【００６４】

（第３の実施形態）

次に、本発明の第３の実施形態について説明する。なお本実施形態の説明において、前述した第１及び第２の実施形態と同一の部分については、図１～図２２に付した符号と同一の符号を付す等して詳細な説明を省略する。

＜要求時データセグメントコピー＞

本実施形態においても、前述した第２の実施形態と同様に、コンピュータ言語のスタートアップ時の変数初期化に必要な時間を短縮する事で、高速に起動可能な組み込みソフトウェアのフレームワークを提供するようにしている。ただし、前述した第２の実施形態では、ページングハードウェア１３０２を用いたが、本実施形態では、後述するデータセグメント管理ユニットを用いるようにしている。

【００６５】

パワーオンリセットが解除され、コールドスタートをする最初のプログラムとして、初期化プログラムが動作し、データセグメント管理ユニットの設定を行う。その際、複写済みエリア記憶レジスタ及び未複写エリアアクセス検出部のカバーするアドレスとして、コンピュータ言語のデータセグメント及びＢＳＳのアドレスとなるエリアを設定し、未複写エリアアクセス検出部を有効化する。そして、本来であればＲＯＭ１００３に存在するデータセグメントの初期値をＲＡＭ１００４へコピーする工程と、ＢＳＳエリアを０で埋める工程とを行わずに、いきなり高級言語で書かれたアプリケーションプログラムに制御を移す。

【００６６】

アプリケーションプログラムの実行に従ってプロセッサ（ＣＰＵ）１００１がデータセグメントをアクセスしようとすると、未複写エリアアクセス検出部によってプログラムが停止させられる。そして、データセグメントにおける部分であって、プロセッサ１００１がアクセスした部分の初期値を、初期データ複写部によって、ＲＯＭ１００３からコピーし、複写済みエリア管理テーブルの管理情報を更新してプログラムに制御を戻す。プロセッサ１００１が再びデータセグメントの同じエリアにアクセスすると、複写済みエリア管理テーブルに複写が既に済んでいる事が記録されているため、未複写エリアアクセス検出部によってプログラムが中断させられる事は無い。本実施形態は、このようなプログラムを用いて実現される。以下、本実施形態の詳細を図面と共に説明する。

【００６７】
＜デジタルカメラ＞
図２３は、本発明の第３の実施形態を示し、デジタルカメラのハードウェア構成の一例を示した図である。
【００６８】
図２３において、２３０２は、ハードウェア処理部分であるデータセグメント管理ユニットである。データセグメント管理ユニット２３０２は、データセグメントとＢＳＳの両方を固定長のページに分割して管理するものであり、バスに接続するタイプの取り外し可能なコンピュータ周辺装置として設計されているものである。その他の部分は、図２に示したものと同じである。また、第１及び第２の実施形態と同様に、電源が投入されると、プロセッサ１００１は、ＲＯＭ１００３に格納されたプログラムのリセットベクターと呼ばれる番地の命令を読み出して実行を開始する。また、データセグメント管理ユニット２３０２は、リセット直後の初期状態となっており、未複写エリアアクセス検出機能停止状態となっている。なお、画像を撮影して、メモリーカード１００９へ記録する過程は本実施形態の本質と関係がないため、説明を省略することとする。
【００６９】
＜データセグメント管理ユニット＞
図２４は、本発明の第３の実施形態を示し、データセグメント管理ユニット２３０２の構成を詳細に示したブロック図である。
図２４において、１０１はデータバス（Data Bus）であり、１０２はアドレスバス（Address Bus）である。これらデータバス１０１とアドレスバス１０２とが、データセグメント管理ユニット２３０２に接続されている。
【００７０】
１０３は、未初期化エリアを検出した場合にそのアドレスを取り込むアドレスラッチ（Address Latch）である。１０４は、データセグメント及びＢＳＳの始まる番地を設定することが可能なプログラマブルアドレスデコーダー（Programmable Address Decoder）である。１０５は、複写済みエリア管理テーブル１０７の記憶済みエリア記憶ビット１０８に対応したアドレスがアクセスされた事を検出するためのＡＮＤ（アンド）回路を複数有するＡＮＤ回路群である。１０６は、アンド回路群１０５のＡＮＤ回路のうち１つからでも信号が出力されたら出力するＯＲ（オア）回路である。１０７は、未複写エリアかどうかを保持するビット群である、複写済みエリア管理テーブルである。なお、本実施形態の複写済みエリア管理テーブル１０７は、例えばプロセッサ１０１により、書き込み及び読み出しがなされるレジスタである。
【００７１】
１０８は、複写済みエリア管理テーブル１０７の１つのエリアに対応した記憶済みエリア記憶ビットである。１０９は、特別なビットでありデータセグメント管理ユニット２３０２の機能を許可、又は不許可にするためのデータセグメント管理ユニットイネーブルビットである。１１０は、ＯＲ回路１０６から信号が出力されるとともに、データセグメント管理ユニットイネーブルビット１０９から、データセグメント管理ユニット２３０２の機能を許可するビットが出力されたときに出力するＡＮＤ回路である。１１１は、データセグメント管理ユニットイネーブルビット１０９から出力されたビットを反転するＮＯＴ回路である。
【００７２】
＜ページ単位＞
図２５は、複写済みエリア管理テーブル１０７の構成の一例を概念的に示した図である。
図２５において、複写済みエリア管理テーブル１０７の各ビットは、ＲＡＭ１００４のエリアに対応している。ＲＡＭ１００４を４Ｋバイト単位のページに分割し、複写済みエリア管理テーブル１０７の各複写済みエリア記憶ビット１０８が、分割されたページに対応する。この事から、例えば８００Ｋバイトのエリアを管理するためには２００ビットの

ラッチがあれば良い事になる。
【００７３】
＜複写済みエリア管理テーブル＞
図２６は、複写済みエリア管理テーブル１０７の構成の一例を詳細に示した図である。
図２６において、２０１は、プロセッサ（ＣＰＵ）１０１から、ビットを読み書きするためのアドレスをデコードするアドレスデコーダ（Address Decoder）である。２０４は、図２４に示した複写済みエリア記憶ビット１０８に対応するラッチである。２０７はリセット信号（/RESET）、２０８はリードストローブ信号（ＲＤ）、２０９はライトストローブ信号（ＷＲ）である。２０２は、ライトストローブ信号２０９と、アドレスデコーダ２０１との両方が真になった時にライト信号を出力し、特定のアドレスに対応したラッチ２０４が、データバス１０１からデータＤ０～Ｄ３を取り込むようにするためのアンド回路である。２０３は、リードストローブ信号２０８と、アドレスデコーダ２０１との両方が真になった時にリード信号を出力し、特定のアドレスに対応したラッチ２０４のデータＱ０～Ｑ３をデータバス１０１に出力させるアンド回路である。２０５は、ラッチ２０４の出力を反転するＮＯＴ回路である。
【００７４】
プロセッサ１０１が、３２ビットのＣＰＵの場合、データバス１０１は３２ビットのため、一度に読み書き出来るビットは３２個となる。この場合、アドレスデコーダ２０１の出力１つに対してアンド回路２０２、２０３と、３２個のラッチ２０４が接続されることになる。それぞれのラッチ２０４の反転出力は、図２４に示したアンド回路１０５に入力される。また、リセット時のラッチ２０４の値は０であり反転出力はハイになっている。
【００７５】
なお、本実施形態で処理されるプログラムの一例と、そのプログラムをコンパイルした結果のアッセンブラリストの一例は、それぞれ図１１及び図１２に示したものと同じである。
【００７６】
＜本実施形態のスタートアップ＞
本実施形態におけるスタートアップする際の動作の一例を図２７のフローチャートを使って説明する。
ステップＳ２７０１でリセットが解除されたら、ステップＳ２７０２で図２３に示したデータセグメント管理ユニット２３０２の設定を行う。このとき、データセグメントとＢＳＳの両方のエリアに対してデータセグメント管理ユニット２３０２が有効となるように管理開始アドレスを設定し、未複写エリアアクセス検出機能を有効な状態にする。そして、ステップＳ２７０３でＣ言語をスタートさせ、ステップＳ２７０４でＯＳをスタートさせ、ステップＳ２７０５でアプリケーションプログラムをスタートさせる。
【００７７】
＜データセグメントのアクセス＞
図２７のフローチャートのステップＳ２７０５のアプリケーションプログラムが実行されていく様子を、図１１に示したプログラム１１０１を例に挙げて説明する。図１１のプログラム１１０１の１０３行目からアプリケーションプログラムを実行していく。最初に実行される有効な行は１０５行目である。１０５行目は、変数「fFirstAccess」が真なら１０６行目を実行し、偽なら１０８行目を実行するという分岐のルーチンである。この行を実行する事によって１００行目で宣言している「fFirstAccess」という変数の値を読む事になる。図２８は、プロセッサ１００１によって変数「fFirstAccess」がアドレッシングされている様子の一例を示した図である。変数「fFirstAccess」に対応する複写済みエリア管理テーブル１０７のビットが未複写状態となっているため、図２４のＡＮＤ回路１１０の出力がアクティブとなり、アドレスラッチ１０３に変数「fFirstAccess」のアドレスが記録される。そして、図２３に示したプロセッサ１００１にＮＭＩ（ノンマスカバブルインタラプト）信号２８０１が出力され、ＮＭＩ割り込みハンドラが起動する。
【００７８】

＜ＮＭＩ割り込みハンドラの処理（初期データ複写手段）＞
図２９は、ＮＭＩ割り込みハンドラの処理の一例を説明するフローチャートである。
ステップＳ２９０１でＮＭＩ割り込みの処理がスタートすると、ステップＳ２９０２で図２３に示したアドレスラッチ１０３にラッチされたアドレスを調べる。前述した変数「fFirstAccess」はデータセグメント中の変数のため、ステップＳ２９０３に分岐する。ステップＳ２９０３では、変数「fFirstAccess」を含むページの４Ｋバイト分の初期値を、対応するＲＯＭ１００３のアドレスから転送する。図３０は、変数「fFirstAccess」に初期値である「TRUE」がＲＯＭ１００３からＲＡＭ１００４に転送された様子の一例を示した図である。
【００７９】
そして、ステップＳ２９０４で複写済みエリア管理テーブル１０７の情報を更新する。図３１は、複写済みエリア管理テーブル１０７の情報が更新された状態の様子の一例を示した図である。
そして、ステップＳ２９０５で元のプログラムへ制御を戻すが、ＮＭＩ割り込みの原因となった命令からやり直すようにプログラムカウンタを設定してから元のプログラムへ制御を戻す。
【００８０】
前記ステップＳ２９０２において、アドレスラッチ１０３にラッチされたアドレスが、変数「fFirstAccess」を含むページの４Ｋバイトがデータセグメントとｂｓｓの両方を含むページだった場合には、ステップＳ２９０６へ進み、データセグメントの初期値をＲＯＭ１００３からＲＡＭ１００４に転送するとともに、ｂｓｓ領域のデータを０に初期化する。
図１１に示したプログラム１１０１が１０７行目まで進むと、変数「fFirstAccess」を再びアクセスする事となるが、図２４に示した複写済みエリア記憶レジスタの情報が複写済みとなっているため、ＮＭＩ割り込みハンドラが再び発生する事はない。
【００８１】
＜ｂｓｓのアクセス＞
そして、図１１に示したプログラム１１０１が１０９行目まで進み、変数「Count」へアクセスすると、ＮＭＩ割り込みハンドラが再び起動し、図２９のＮＭＩ割り込みハンドらの処理を実行する事となる。この場合、アドレスラッチ１０３にラッチされたアドレスは、ｂｂｓエリアである。このため、ステップＳ２９０２からステップＳ２９０７へ進む。ステップＳ２９０７では、変数「Count」を含むページの４Ｋバイトを全て０で初期化する。そして、ステップＳ２９０４へ進み、複写済みエリア管理テーブル１０７の情報を更新し、ステップＳ２９０５で元のプログラムへ制御を戻す。
【００８２】
＜バックグラウンド転送＞
以上のように構成する事で、データセグメント転送とｂｓｓの０クリアの待ち時間を省いてＣ言語のプログラム１１０１をスタートすることができる。そして、ユーザーアプリケーションを高速にスタートする事ができる。しかし、デジタルカメラとしてのスタートアップはユーザーアプリケーションが始まってからスタートする。
【００８３】
デジタルカメラとしては、撮像素子への電源投入や、レンズ１００７の初期位置への駆動や、メモリーカード１００９からのファイル情報の読み出しなど、デバイスが反応する待ち時間を含む処理が多い。そこで、それらの待ち時間を利用して、未転送のデータセグメントへの初期値の転送や初期化されていないｂｓｓの初期化を行う事で、要求時の転送のオーバーヘッドを極力減らす事が可能となる。
【００８４】
図３２は、本実施形態におけるバックグラウンド処理の一例を説明するフローチャートである。バックグラウンド処理を起動するプログラムは最もプライオリティの低いタスクとして実行する。このプログラムは、他に処理するべきプログラムが存在しない場合のみ

動作するため、デジタルカメラとしての動作に影響する事がない。

【００８５】

ステップＳ３２０１でタスクが開始すると、ステップＳ３２０２でポインタにデータセグメントの開始番地をセットする。次に、ステップＳ３２０３でポインタの値を読む。それにより、図１１に示したプログラム１１０１の１０５行目で、変数「fFirstAccess」をアクセスした場合と同様の事が発生し、複写済みエリア管理テーブル１０７の複写済みエリア記憶ビット１０８が未複写状態であれば、図３１に示したＮＭＩ割り込みハンドラの処理が起動され、初期値がＲＯＭ１００３からＲＡＭ１００４に転送される。

【００８６】

次に、ステップＳ３２０４で、ポインタをページサイズである４Ｋバイト分進める。

次に、ステップＳ３２０５で、ポインタがＢＳＳの最後のアドレスを超えたか否かを判定し、超えている場合にはステップＳ３２０３に戻る。ステップＳ３２０３からステップＳ３２０５の処理を繰り返す事で、データセグメントとＢＳＳの全領域の初期化が完了すると、ステップＳ３２０５からステップＳ３２０６へ進み、バックグラウンドタスクそのものを終了させる。

【００８７】

以上のように本実施形態では、ＲＯＭ１００３のデータセグメントの初期値が格納されたエリアから、データセグメントの対応するエリア（ＲＡＭ１００４）へデータコピーが行われたかどうかを記憶する複写済みエリア管理テーブル１０７を有し、ＲＡＭ１００４のデータセグメント内にあるＲＯＭ１０３からのデータコピーが未だ行われていないエリアに対するアクセスを検出すると、実行中のプログラムを中断して、ＮＭＩ割り込みハンドラの処理の中でＲＯＭ１００３からＲＡＭ１００４へのデータコピーを行った後、複写済みエリア管理テーブル１０７の情報を更新して、前記プログラムの実行を再開することにより、データセグメントの初期値をＲＯＭ１００３からＲＡＭ１００４へコピーする動作を、オンデマンド（要求時）に分散して行うようにした。

【００８８】

これにより、従来はプログラムの実行前に必要だった全てのデータセグメントの初期化を、プログラムの実行の進行に従って必要な部分のみを分散して行う事が可能となり、起動からプログラム実行までのタイムラグを大幅に短縮する事が可能となる。

【００８９】

本実施形態の複写済みエリア管理テーブル１０７は、ＣＰＵであるプロセッサ１００１からの書き込み、読み出しが可能なレジスタであり、各複写済みエリア記憶ビット１０８が、ＲＡＭ１００４の特定のアドレスから始まる一定サイズのブロックに対応している。複写済みエリア管理テーブル１０７の出力とアドレスバス１０２とを監視し、アドレスが複写済みエリア管理テーブル１０７の管理情報における未複写エリアと一致したことを検出すると、ＮＭＩ（ノンマスカバブルインタラプト）などの例外処理を発生させ、検出したアドレスをＣＰＵであるプロセッサ１００１から読み書き可能なレジスタに自動的に保存するようにする。このＮＭＩの例外発生によって実行中のプログラムを中断して、ＲＯＭ１００３からＲＡＭ１００４へのデータコピーを行った後に、複写済みエリア管理テーブル１０７の情報を更新してプログラムの実行を再開することをソフトウェアの例外処理ハンドラで実現するようにしたので、少ない回路規模でシステムを構築する事ができる。

【００９０】

複写済みエリア管理テーブル１０７及び未複写にエリア対するアクセスを検出する未複写エリア部のカバーするアドレスの開始番地を動的に設定可能にするようにしたので、複写済みエリア管理テーブル１０７及び未複写エリアアクセス検出部のカバーするエリアを最適なサイズで構成できる。

【００９１】

前記未複写エリアアクセス検出部は、検出機能を停止させる未複写エリアアクセス検出禁止機能を備え、パワーオンリセット時（コールドスタート時）に自動的に未複写エリアアクセス検出禁止状態となるようにしている。また、複写済みエリア記憶レジスタは、パ

ワーオンリセット時（コールドスタート時）に自動的にすべてのエリアが未複写エリアに設定されるようにしている。このようにする事によって、ソフトウェアによる初期設定にかかる時間を短縮する事が可能となり、より短い時間でコンピュータをスタートアップさせる事が可能となる。

【００９２】

デジタルカメラのスタートアップ時には、レンズ１００７の駆動や、ファイルの読み出しなどの、ＣＰＵであるプロセッサ１００１が待ち状態となる時間が細切れに存在する。そこで、本実施形態では、実行すべき有効なプログラムが存在しないアイドル状態になった場合に、ＲＯＭ１００３からＲＡＭ１００４へのデータコピー及び複写済みエリア管理テーブル１０７の情報更新を行うようにした。これにより、プログラムがデータセグメントをアクセスしたときに、ＲＯＭ１００３からＲＡＭ１００４へのコピーがすでに完了している可能性が向上し、より高速に処理を進める事が可能となる。

【００９３】

（本発明の他の実施形態）

上述した実施形態の機能を実現するべく各種のデバイスを動作させるように、該各種デバイスと接続された装置あるいはシステム内のコンピュータに対し、前記実施形態の機能を実現するためのソフトウェアのプログラムコードを供給し、そのシステムあるいは装置のコンピュータ（ＣＰＵあるいはＭＰＵ）に格納されたプログラムに従って前記各種デバイスを動作させることによって実施したものも、本発明の範疇に含まれる。

【００９４】

また、この場合、前記ソフトウェアのプログラムコード自体が上述した実施形態の機能を実現することになり、そのプログラムコード自体、及びそのプログラムコードをコンピュータに供給するための手段、例えば、かかるプログラムコードを格納した記録媒体は本発明を構成する。かかるプログラムコードを記憶する記録媒体としては、例えばフレキシブルディスク、ハードディスク、光ディスク、光磁気ディスク、ＣＤ－ＲＯＭ、磁気テープ、不揮発性のメモリーカード、ＲＯＭ等を用いることができる。

【００９５】

また、コンピュータが供給されたプログラムコードを実行することにより、上述の実施形態の機能が実現されるだけでなく、そのプログラムコードがコンピュータにおいて稼働しているＯＳ（オペレーティングシステム）あるいは他のアプリケーションソフト等と共同して上述の実施形態の機能が実現される場合にもかかるプログラムコードは本発明の実施形態に含まれることは言うまでもない。

【００９６】

さらに、供給されたプログラムコードがコンピュータの機能拡張ボードやコンピュータに接続された機能拡張ユニットに備わるメモリーに格納された後、そのプログラムコードの指示に基づいてその機能拡張ボードや機能拡張ユニットに備わるＣＰＵ等が実際の処理の一部または全部を行い、その処理によって上述した実施形態の機能が実現される場合にも本発明に含まれることは言うまでもない。

【図面の簡単な説明】

【００９７】

【図１】本発明の第１の実施形態を示し、ページテーブルの概念の一例を示した図である。

【図２】本発明の第１の実施形態を示し、デジタルカメラのハードウェア構成の一例を示した図である。

【図３】本発明の第１の実施形態を示し、デジタルカメラにおけるシステムが起動した直後のページングハードウェアと、ＲＯＭと、ＲＡＭの状態の一例を示した図である。

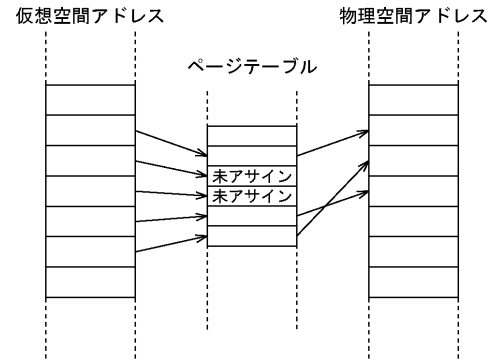
【図４】本発明の第１の実施形態を示し、アプリケーションプログラムの一例であるソースコードを示した図である。

【図５】本発明の第１の実施形態を示し、関数CreateShadowの動作の一例を説明するフローチャートである。

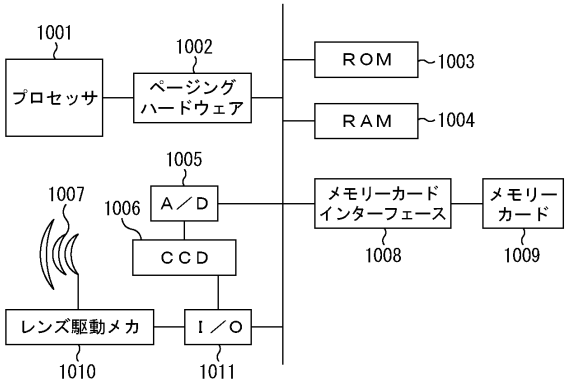
【図６】本発明の第１の実施形態を示し、関数CreateShadowの動作が終了した時点のページングハードウェアと、ＲＯＭと、ＲＡＭの状態の一例を示した図である。
【図７】本発明の第１の実施形態を示し、ページフォルト割り込み処理の一例を説明するフローチャートである。
【図８】本発明の第１の実施形態を示し、ＲＯＭの該当ページの内容を、空きフレームにコピーした時点のページングハードウェアと、ＲＯＭと、ＲＡＭの状態の一例を示した図である。
【図９】本発明の第１の実施形態を示し、ページテーブルにフレームをアサインした時点のページングハードウェアと、ＲＯＭと、ＲＡＭの状態の一例を示した図である。
【図１０】本発明の第１の実施形態を示し、ページメモリにフレームをアサインした時点のページングハードウェアと、ＲＯＭと、ＲＡＭの状態の一例を示した図である。
【図１１】本発明の第２の実施形態を示し、Ｃ言語で書かれたプログラムの一例を示した図である。
【図１２】本発明の第２の実施形態を示し、アッセンブラリストの一例を示した図である。
【図１３】本発明の第２の実施形態を示し、スタートアップする際の従来の動作を説明するフローチャートである。
【図１４】本発明の第２の実施形態を示し、スタートアップする際の動作の一例を説明するフローチャートである。
【図１５】本発明の第２の実施形態を示し、変数領域のページテーブルに物理アドレスがアサインされていない様子を示す図である。
【図１６】本発明の第２の実施形態を示し、ページフォルト割り込みによって動作するプログラムによる処理動作の一例を説明するフローチャートである。
【図１７】本発明の第２の実施形態を示し、ＲＯＭに格納された変数の初期値がＲＡＭの物理アドレスへ転送された様子の一例を示した図である。
【図１８】本発明の第２の実施形態を示し、ページテーブルに物理空間アドレスが設定された様子の一例を示した図である。
【図１９】本発明の第２の実施形態を示し、ページフォルトが発生した時点の様子の一例を示した図である。
【図２０】本発明の第２の実施形態を示し、物理メモリーをゼロ（０）に埋めた直後の様子の一例を示した図である。
【図２１】本発明の第２の実施形態を示し、ページテーブルを更新した直後の様子の一例を示した図である。
【図２２】本発明の第２の実施形態を示し、バックグラウンドで初期化を行うプログラムの動作の一例を説明するフローチャートである。
【図２３】本発明の第３の実施形態を示し、デジタルカメラのハードウェア構成の一例を示した図である。
【図２４】本発明の第３の実施形態を示し、データセグメント管理ユニットの構成を詳細に示したブロック図である。
【図２５】本発明の第３の実施形態を示し、複写済みエリア記憶レジスタの構成の一例を概念的に示した図である。
【図２６】本発明の第３の実施形態を示し、複写済みエリア記憶レジスタの構成の一例を詳細に示した図である。
【図２７】本発明の第３の実施形態を示し、スタートアップする際の動作の一例を説明するフローチャートである。
【図２８】本発明の第３の実施形態を示し、プロセッサによって変数「fFirstAccess」がアドレッシングされている様子の一例を示した図である。
【図２９】本発明の第３の実施形態を示し、ＮＭＩ割り込みハンドラの処理の一例を説明するフローチャートである。
【図３０】本発明の第３の実施形態を示し、変数「fFirstAccess」に初期値である「TRUE

」がＲＯＭからＲＡＭに転送された様子の一例を示した図である。
【図３１】本発明の第３の実施形態を示し、複写済みエリア記憶レジスタの情報が更新された状態の様子の一例を示した図である。
【図３２】本発明の第３の実施形態を示し、バックグラウンド処理の一例を説明するフローチャートである。
【符号の説明】
【００９８】
１０１　データバス
１０２　アドレスバス
１０３　アドレスラッチ
１０４　プログラマブルアドレスデコーダー
１０５　アンド回路
１０６　オア回路
１０７　複写済みエリア管理テーブル
１０８　記憶済みエリア記憶ビット
１０９　データセグメント管理ユニットイネーブルビット
１１０　アンド回路
２０１　アドレスデコーダ
２０２、２０３　アンド回路
２０４　ラッチ
２０７　リセット信号
２０８　リードストローブ信号
２０９　ライトストローブ信号
１００１　プロセッサ
１００２　ページングハードウェア
１００３　ＲＯＭ
１００４　ＲＡＭ
１００５　Ａ／Ｄ変換機
１００６　ＣＣＤ
１００７　レンズ
１００８　インターフェース
１００９　メモリーカード
１０１１　Ｉ／Ｏインターフェース
１０１０　レンズ駆動メカ
２３０２　データセグメント管理ユニット

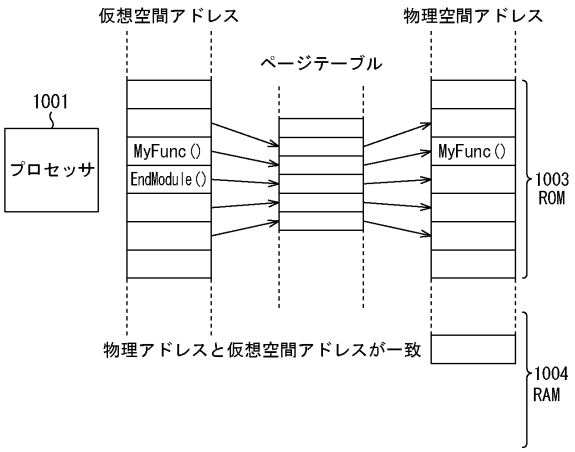
【図１】



【図２】



【図３】

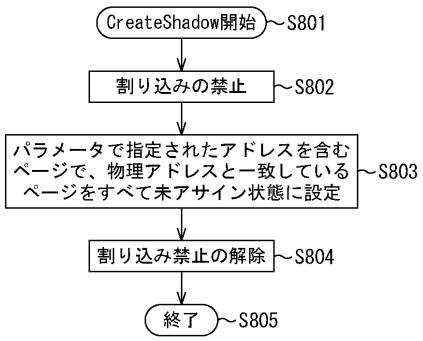


【図４】

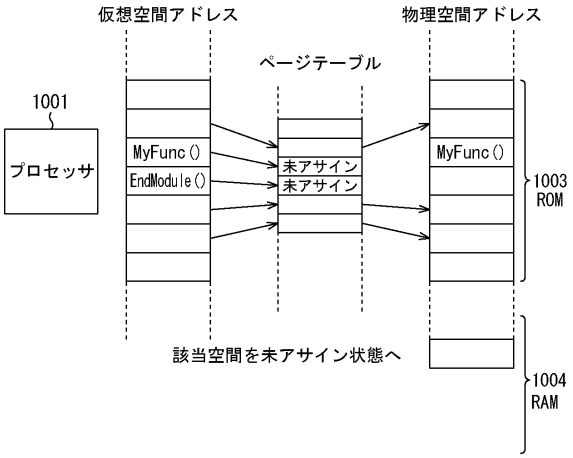
301

```
100: void InitializeMyModule(void)
101: {
102:    CreateShadow(MyFirstFunc,(char*)EndModule-(char*)MyFirstFunc);
103: }
104:
105:void MyFirstFunc(void)
106:{
107:    :
108:}
109:void MyFunc(void)
110:{
111:    :
112:}
113: static void EndModule(void)
114: {
115: }
```

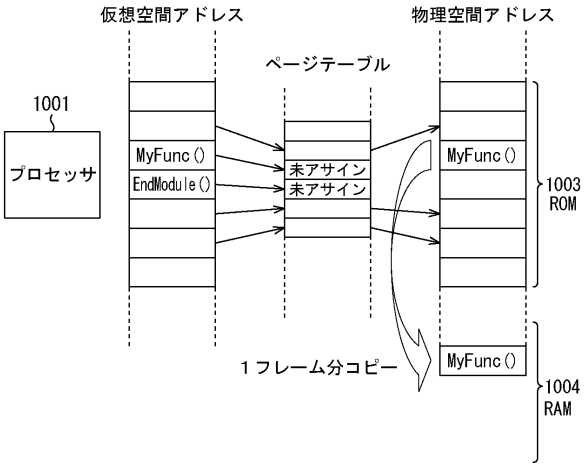
【図５】



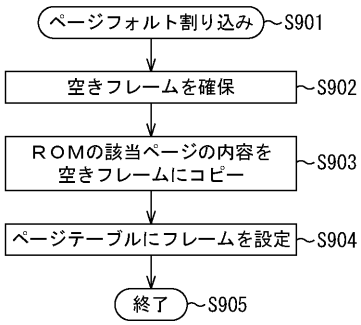
【図６】



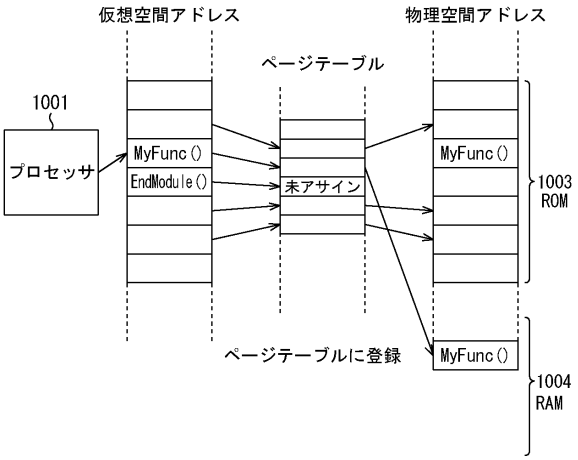
【図８】



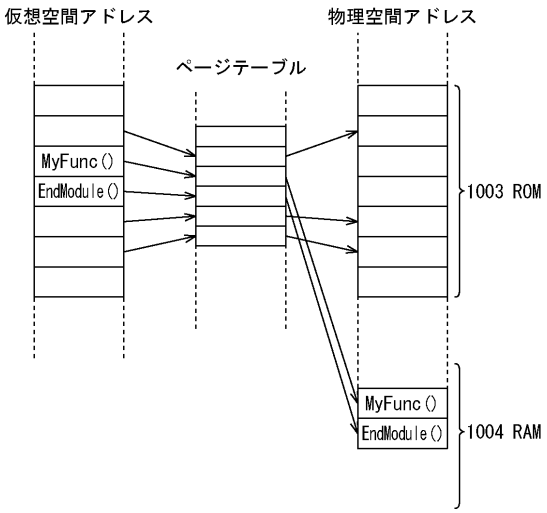
【図７】



【図９】



【図１０】



【図１１】

1101

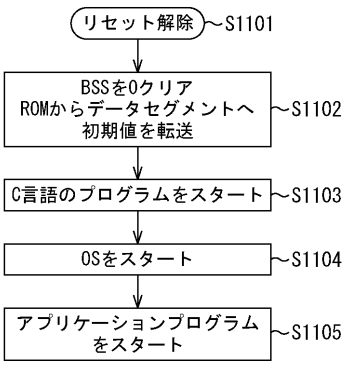
```
100: static int  fFirstAccess = TRUE;
101: static int  Count;
102:
103: void MyFunction( void )
104: {
105:   if( fFirstAccess ){
106:       FirstFunction();
107:       fFirstAccess = FALSE;
108:   }
109:    Count++;
110:    return;
111:}
```

【図１２】

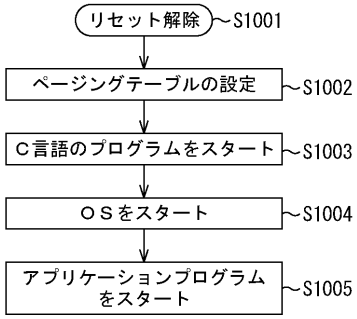
1201

```
0001:_DATA      SEGMENT
0002:  _fFirstAccess DD 01H
0003:_DATA      ENDS
0004:PUBLIC     _MyFunction
0005:EXTRN      _FirstFunction:NEAR
0006:_BSS       SEGMENT
0007:   _Count DD 01H DUP (?)
0008:_BSS       ENDS
0009:_TEXT      SEGMENT
0010:_MyFunction PROC NEAR
0011:  push     ebp
0012:  mov      ebp, esp
0013:  cmp      DWORD PTR _fFirstAccess, 0
0014:  je       SHORT $L34
0015:  call     _FirstFunction
0016:  mov      DWORD PTR _fFirstAccess, 0
0017:$L34:
0018:  mov      eax, DWORD PTR _Count
0019:  add      eax, 1
0020:  mov      DWORD PTR _Count, eax
0021:  mov      eax, DWORD PTR _Count
0022:  pop      ebp
0023:  ret      0
0024:_MyFunction ENDP
0025:_TEXT      ENDS
0026:END
```

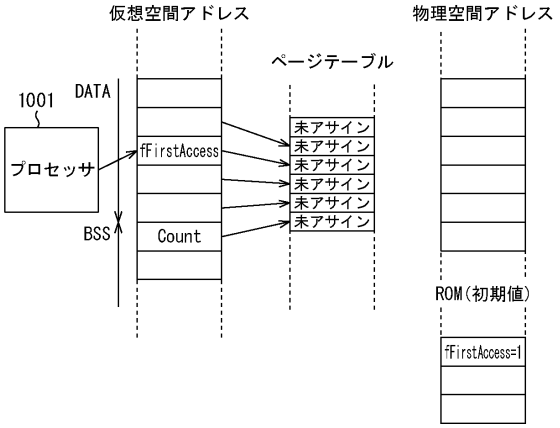
【図１３】



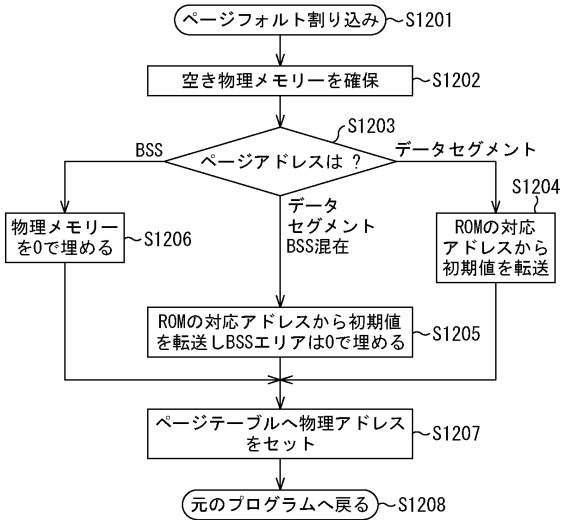
【図１４】



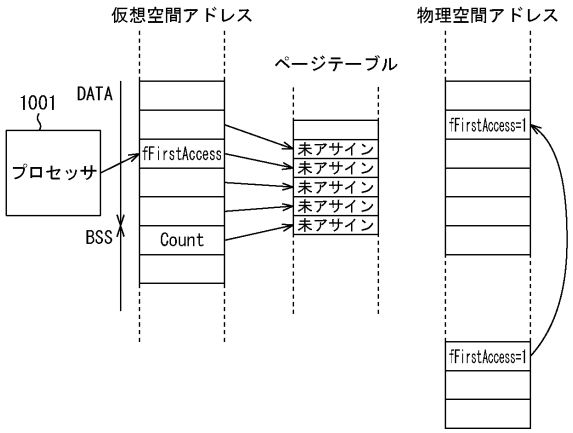
【図１５】



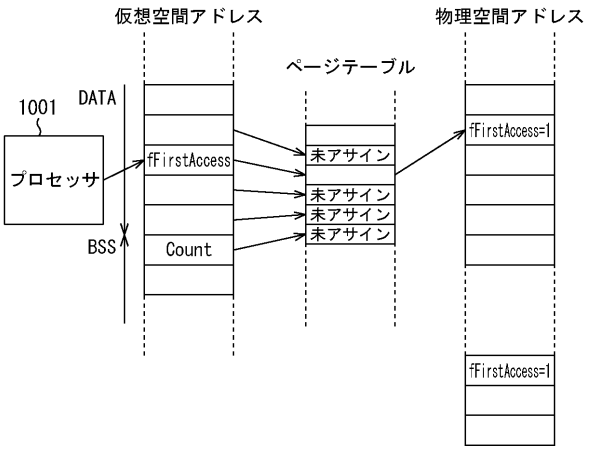
【図１６】



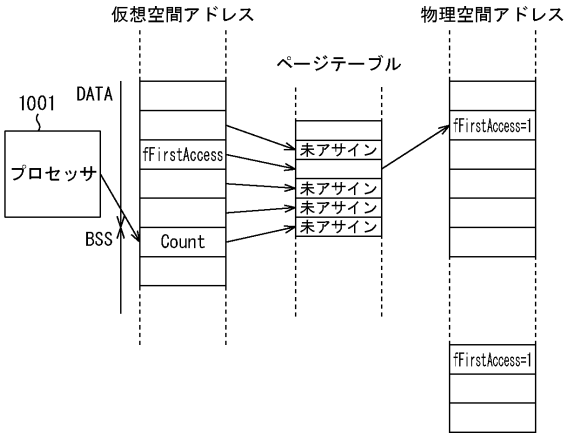
【図１７】



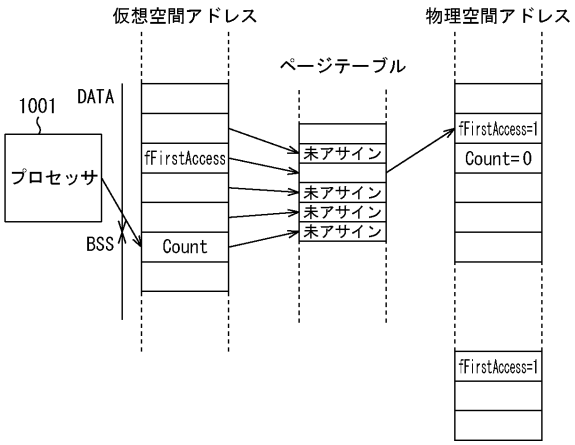
【図１８】



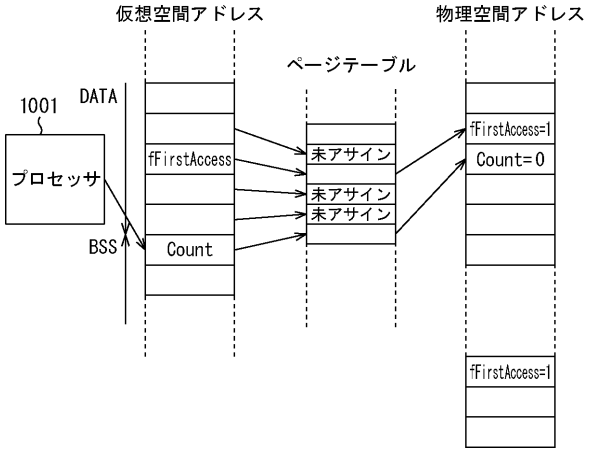
【図１９】



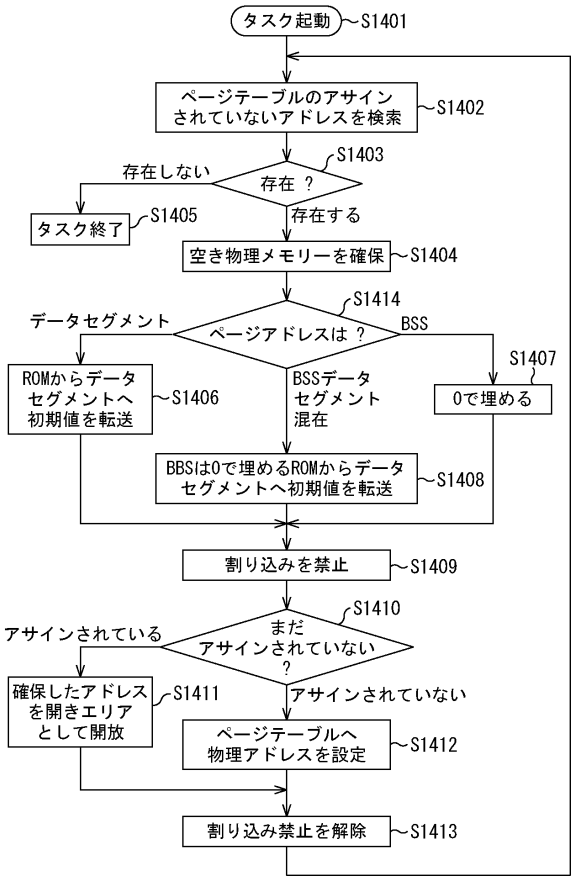
【図２０】



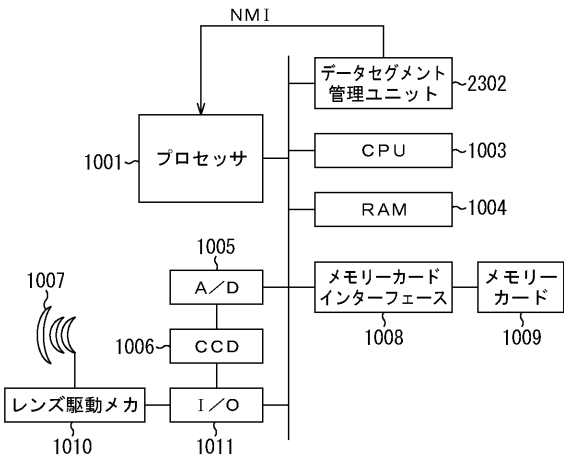
【図２１】



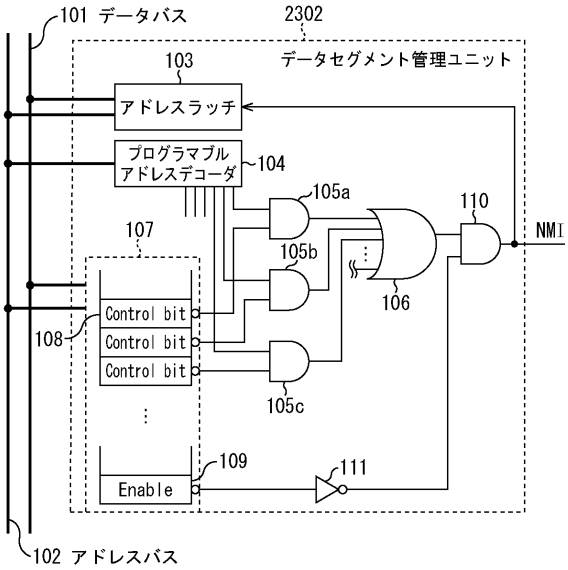
【図２２】



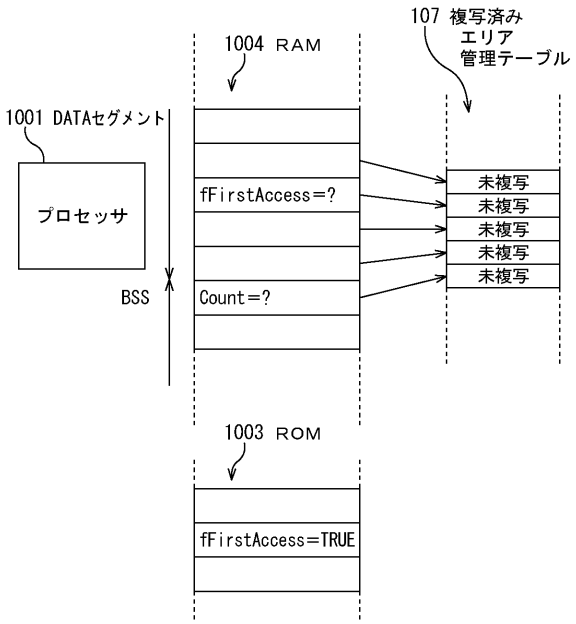
【図２３】



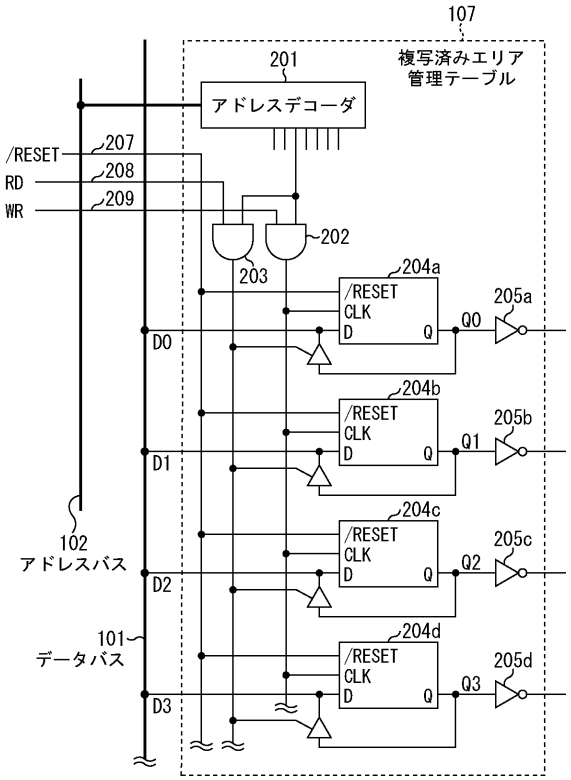
【図２４】



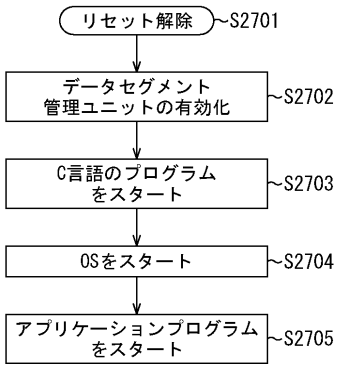
【図２５】



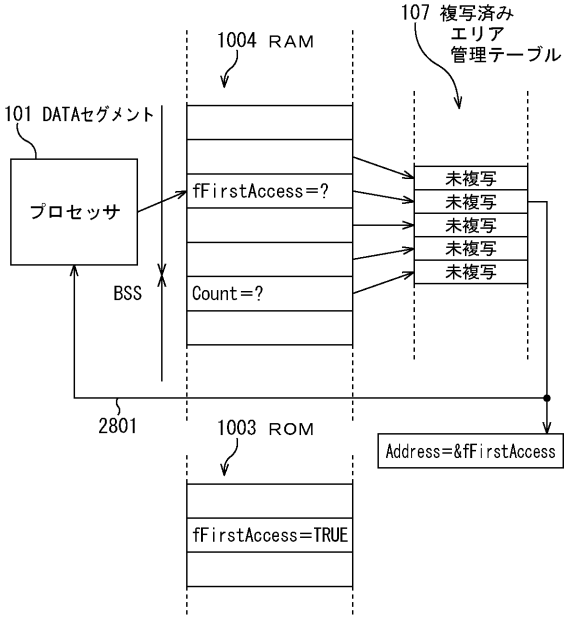
【図２６】



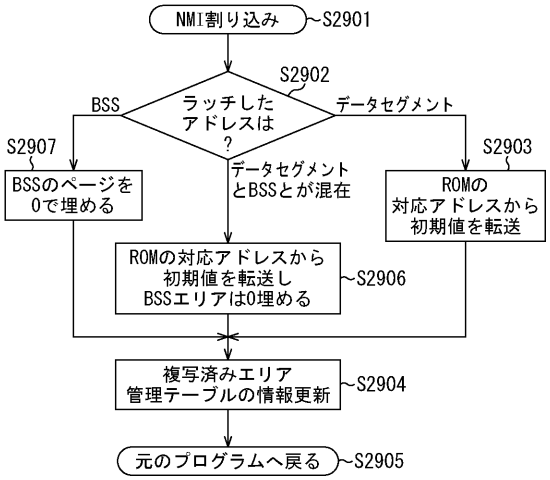
【図２７】



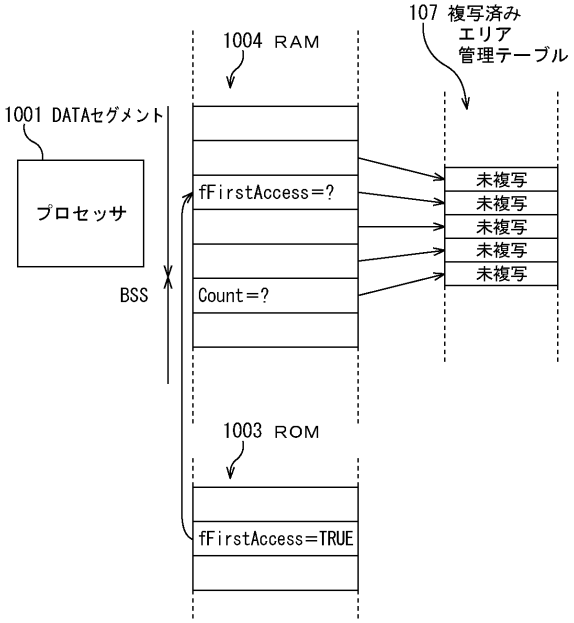
【図２８】



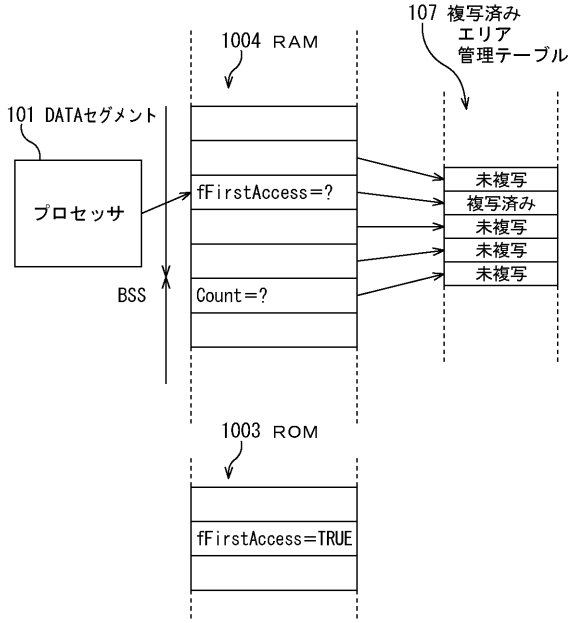
【図２９】



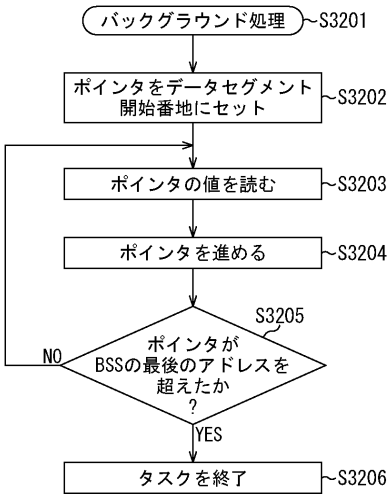
【図３０】



【図３１】



【図３２】


バックグラウンド処理
S3201
ポインタをデータセグメント開始番地にセット
S3202
ポインタの値を読む
S3203
ポインタを進める
S3204
S3205
ポインタがBSSの最後のアドレスを超えたか？
NO
YES
タスクを終了
S3206

フロントページの続き

(56)参考文献 特開平１０－１８７５２９（ＪＰ，Ａ）
特開昭５７－１２７９９４（ＪＰ，Ａ）
特開平７－１１４４９９（ＪＰ，Ａ）
特開２００５－７８４１９（ＪＰ，Ａ）
特開昭６３－１０９５５７（ＪＰ，Ａ）

(58)調査した分野(Int.Cl.，ＤＢ名)
Ｇ０６Ｆ９／４４５，
Ｇ０６Ｆ９／０６，
Ｇ０６Ｆ１２／００－１２／１２，
Ｇ０６Ｆ９／４６－９／５４，
Ｇ０６Ｆ９／３０－９／３６