(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号
特許第3593241号
(P3593241)
(45)発行日　平成16年11月24日(2004.11.24)　(24)登録日　平成16年9月3日(2004.9.3)

(51)Int.Cl.[7]	F I		
G06F 9/46	G06F	9/46	330C
G06F 1/00	G06F	1/00	370D
G06F 9/50	G06F	11/14	310K
G06F 11/14	G06F	9/06	640H

請求項の数 6　(全 19 頁)

(21)出願番号　特願平9-191840
(22)出願日　平成9年7月2日(1997.7.2)
(65)公開番号　特開平11-24943
(43)公開日　平成11年1月29日(1999.1.29)
審査請求日　平成13年2月21日(2001.2.21)

(73)特許権者　000005108
株式会社日立製作所
東京都千代田区神田駿河台四丁目６番地
(74)代理人　100099298
弁理士　伊藤　修
(72)発明者　関口　知紀
神奈川県川崎市麻生区王禅寺１０９９番地
株式会社日立製作所　システム開発研究所内
(72)発明者　新井　利明
神奈川県川崎市麻生区王禅寺１０９９番地
株式会社日立製作所　システム開発研究所内

審査官　殿川　雅也

最終頁に続く

(54)【発明の名称】計算機の再起動方法

(57)【特許請求の範囲】

【請求項１】

複数のロードモジュールから構成されるオペレーティングシステムの計算機における再起動方法であって、

あらかじめ、前記複数のロードモジュールの内、前記オペレーティングシステムの再起動中に継続して動作させる一つのロードモジュール（第一のロードモジュール）を特定し、

前記オペレーティングシステムの再起動前に、前記特定に基づいて、

前記第一のロードモジュールの実行状態をメモリに保持し、

前記第一のロードモジュールが処理する割り込みは受け付け可能な状態とし、

前記オペレーティングシステムの再起動時に、

前記第一のロードモジュール以外の前記複数のロードモジュールを前記計算機のメモリのうち前記第一のロードモジュールが保持されている領域以外の領域にロードすることを特徴とする計算機の再起動方法。

【請求項２】

請求項１記載の計算機の再起動方法において、

前記再起動は、前記オペレーティングシステムがソフトウェア障害を受けたときに行われることを特徴とする計算機の再起動方法。

【請求項３】

請求項１記載の計算機の再起動方法において、

前記割り込みは外部割り込みであることを特徴とする計算機の再起動方法。

【請求項４】
請求項１記載の計算機の再起動方法において、
前記オペレーティングシステムの再起動中に、前記第一のロードモジュールによる前記割り込みの受け付けが行われることを特徴とする計算機の再起動方法。
【請求項５】
請求項１記載の計算機の再起動方法において、
前記オペレーティングシステムの再起動中は、前記第一のロードモジュールが存在する前記メモリの領域を書き込み禁止状態とすることを特徴とする計算機の再起動方法。
【請求項６】
第一の計算機と第二の計算機がネットワークを介して接続される計算機システムにおける計算機の再起動方法であって、
前記第一の計算機は、前記第二の計算機にリクエストを送り、
複数のロードモジュールから構成されるオペレーティングシステムの再起動中に前記第二の計算機は、前記オペレーティングシステムの再起動中に前記リクエストを受け付けるための前記複数のロードモジュールの中の一つのロードモジュール（第一のロードモジュール）を特定し、
前記第二の計算機の再起動時に、前記第二の計算機は、前記第一のロードモジュールの実行状態をメモリに保持し、前記第一のロードモジュールが処理する割り込みは受け付け可能な状態とし、前記第一のロードモジュール以外の前記複数のロードモジュールを前記計算機のメモリにロードすることにより前記オペレーティングシステムを再起動し、前記第一の計算機から前記再起動中に送られてきた前記リクエストを、前記割り込みにより前記第一のロードモジュールが受け付けることを特徴とする計算機の再起動方法。
【発明の詳細な説明】
【０００１】
【発明の属する技術分野】
本発明は計算機の起動方法に関し、特に、オペレーティングシステムがソフトウェアフォルトにより停止した時の再起動方法に関する。
【０００２】
【従来の技術】
一部のオペレーティングシステムでは、計算機に接続しているハードウェアについて、それを制御するデバイスドライバモジュールをカーネルから分離して、カーネルが必要なデバイスドライバをロードして利用することが可能になっている。
上記の機能を持つオペレーティングシステムにおいて、クロック割り込みを横取りして管理する実時間処理ドライバを利用して、時間制約の厳しいシステムの制御を計算機に実施させる方式がある。
この方式では、本来オペレーティングシステムが受けとるべきクロック割り込みを実時間処理ドライバが横取りし、オペレーティングシステムの処理より優先して実時間処理を実施し、その後オペレーティングシステムに制御を戻す方法が取られる。
この方式は、Ｗｉｎｄｏｗｓ　ＮＴ等のカーネルモードで動作するモジュールの追加が容易なオペレーティングシステムで多く実現されている。
【０００３】
実時間処理においては信頼性の確保も大きな問題となる。上記の方式においては、実時間処理モジュールをオペレーティングシステムのカーネルに依存しないようにすることにより、オペレーティングシステムがソフトウェア障害により停止してしまっても、実時間処理を継続することができる機能を持つものもある。
また、オペレーティングシステムが障害により停止する時に、実時間処理モジュールにその旨通知を発行して、実時間処理モジュール側でオペレーティングシステムの停止に対処する処理を実施できるようにしているものもある。
【０００４】
しかしながら、これらの方式では、ソフトウェア障害により停止してしまったオペレーテ

ィングシステムを再起動する時、実時間処理ドライバも停止してしまう。つまり、オペレーティングシステムの再起動処理と実時間処理を同時には実施できないという問題点がある。
これは、オペレーティングシステムの再起動時にプロセッサをリセットしてしまうため、実時間モジュールが動作するための仮想記憶、割り込み処理用の情報が失われてしまうからである。
これは、非常に短い周期で停止することなく制御しなければならないハードウェアがある場合には、計算機再起動時にその制御が途絶えてしまうため問題である。
【０００５】
従来技術ではクロック割り込みに限らず、計算機の再起動中はハードウェアの発生する外部割り込みを受け付けることができない。その割り込み処理がオペレーティングシステムの機能に依存していなくてもである。
例えば、複数の計算機からなるクラスタ構成の計算機システムでは、他の計算機が稼働しているかどうかを一定間隔で問い合わせて、一定時間返答がなければその計算機が実行を停止していると判断し、システム構成を変更する処理を実施する。
この場合に、計算機が停止しているという判断を下すためには一定の待ち時間が必要になり、システム再構成を開始するまでの時間が長くなり問題である。
この問合せに反応する外部割り込みをカーネル再起動処理中も受け付けて返答を返すことができれば、再構成を開始するまでの時間を短縮できる。
また、プロセッサをリセットする再起動方法では、メモリチェック、ハードウェア構成認識等の処理が実行されるため、オペレーティングシステムの起動までの時間が長くなるという問題がある。
【０００６】
【発明が解決しようとする課題】
従来のオペレーティングシステムでは、オペレーティングシステムがソフトウェアフォルトにより停止し再起動を実施する場合に、再起動処理の間、外部デバイスからの割り込みを受け付けられず、割り込みに対する処理を実行できない問題がある。
本発明の目的は、オペレーティングシステム再起動時でも、オペレーティングシステムとは独立して処理を実施できる外部割り込み処理について、その割り込みの処理をオペレーティングシステムの再起動の間も実行できるようにすることにある。
【０００７】
【課題を解決するための手段】
上記目的を達成するため、本発明は、
複数のロードモジュールから構成されるオペレーティングシステムの１台の計算機における再起動方法であり、
前記オペレーティングシステムの再起動中に動作させる前記複数のロードモジュールの中の１つのロードモジュールはメモリに保持し、
前記１つのロードモジュールが処理する割り込みは受け付け可能な状態とし、
前記１つ以外のロードモジュールを前記計算機のメモリにロードするようにしている。
また、前記再起動は、前記オペレーティングシステムがソフトウェア障害を受けたときに行われるようにしている。
また、前記割り込みは外部割り込みであるようにしている。
また、前記オペレーティングシステムの再起動中に、前記１つのロードモジュールによる前記割り込みの受け付けが行われるようにしている。
また、前記オペレーティングシステムの再起動中は、前記１つのロードモジュールが存在する前記メモリの領域を書き込み禁止状態とするようにしている。
【０００８】
また、第１の計算機と第２の計算機がネットワークを介して接続される計算機システムにおける計算機の再起動方法であり、
前記第１の計算機は、前記第２の計算機にリクエストを送り、

複数のロードモジュールから構成されるオペレーティングシステムの再起動中に前記第２の計算機は、当該オペレーティングシステムの再起動中に前記リクエストを受け付けるための前記複数の中の１つのロードモジュールをメモリに保持し、前記１つのロードモジュールが処理する割り込みは受け付け可能な状態とし、前記１つ以外のロードモジュールを前記計算機のメモリにロードすることにより再起動し、前記第１の計算機から前記再起動中に送られてきた前記リクエストを、前記割り込みにより前記１つのロードモジュールが受け付けるようにしている。

【００１２】

【発明の実施の形態】

以下に、図面を用いて本発明の第一の実施の形態を説明する。

図１は本発明の実施の形態の計算機を示す図である。

計算機１００はプロセッサ１０１、主記憶１０２、読み取り専用メモリ１０３、磁気ディスク１０５、外部デバイス１０６、および、１０７、割り込みコントローラ１０４、各構成要素を接続するバス１０８、割り込み信号バス１０９からなる。

読み取り専用メモリ１０３には、計算機の初期化処理を実行するプログラムが格納されており、プロセッサ１０１がリセット状態になった時に制御が移るアドレスに初期化プログラムが配置されるようにバス１０８に接続されている。

読み取り専用メモリ１０３に格納されているプログラムは、ハードウェア構成を表すデータを主記憶内に構築する。

更に磁気ディスクのあらかじめ定められた領域にあるデータを主記憶にロードし、それをプログラムとみなして制御を渡す。

この例では、ローダ１１４が磁気ディスク１０５の定められた領域にあり、これが主記憶１０２にロードされて実行される。

処理を引き継いだローダ１１４は、カーネル１１１を主記憶にロードし、プロセッサの仮想アドレス変換機構を設定して、プロセッサを仮想アドレスモードに移し、カーネルを実行する。この時、ローダはハードウェア構成データをカーネルに引き渡す。

【００１３】

カーネルは、ハードウェア構成データを参照してカーネルが管理するハードウェアデバイスの初期化処理を実施し、モジュール構成ファイル１１０の内容にしたがってカーネルからは分離されたプログラムモジュールを主記憶１０２にロードし、それぞれのモジュールの初期化処理を実行する。

例えば、モジュール構成ファイルが、再起動ローダ１１３、デバイスドライバ１１２をロードするように書かれていれば、カーネルはこれらを主記憶にロードし、それぞれの初期化ルーチンを実行する。

図１は、主記憶にカーネル、再起動ローダ、および、デバイスドライバがロードされていることを示している。

最後に、カーネルは最初のプロセスを生成して、計算機起動手順は完了する。

【００１４】

このような計算機システムで、オペレーティングシステムカーネルがソフトウェア障害により停止してしまった時に、プロセッサをリセットして上記の手順で計算機を再起動してしまうと、再起動の間、仮想アドレスモードは解除され、外部デバイス１０６、および、１０７からの割り込みを受け付けることができなくなる。

ここでは、外部デバイス１０６が発生する割り込みは、カーネルがソフトウェア障害により停止し、再起動している間も処理しなければならない割り込みで、デバイスドライバ１１２により管理されるものとして説明する。

本発明は、カーネルの再起動処理をしている間も特定の割り込みを受け付け続け、割り込み処理を実行可能な再起動手順を提供するものである。

【００１５】

外部デバイス１０６はデバイスドライバ１１２により管理されるものとする。

デバイスドライバ１１２は、カーネルが提供するサービスを利用せずに外部デバイス１０

６の割り込みを処理するようにプログラムされているモジュールである。
ここで述べる実施の形態では、デバイスドライバ１１２は、無停止モジュールとして再起動ローダ１１３の無停止モジュール管理テーブル５００に登録される。
再起動ローダ１１３は、カーネルがソフトウェア障害により停止してカーネルの再起動を実施する時に実行されるモジュールであり、無停止モジュールとして管理テーブル５００に登録されているモジュールの処理環境を維持したままカーネルの再起動処理を実行する。

【００１６】
無停止モジュールの処理環境とは、無停止モジュールが配置されている仮想アドレス、それに対するアドレス変換テーブル、モジュールが格納されている物理メモリ、無停止モジュールが処理すべき割り込みの割り込みハンドラの設定を指している。
アドレス変換テーブルはページテーブル４１０に設定され、割り込みハンドラは割り込みハンドラテーブル４３０に設定されている。
また、アドレス空間情報とハードウェア資源情報は資源管理データ１１５に格納されている。
再起動ローダ１１３は、無停止モジュール管理テーブル５００を参照して、無停止モジュールが確保しているアドレス領域、割り込みハンドラの設定を保持したまま、カーネルの再ロード、および、実行を行なう。
これにより、無停止モジュールの割り込み処理の連続性を保つことができる。

【００１７】
以下で、本発明の第１の実施の形態の詳細について説明する。
図２は、本発明のオペレーティングシステムのカーネルのアドレス空間を表現するデータ構造を示している。
２００はカーネルのアドレス空間にロードされているモジュールを記録するロードモジュール管理テーブルである。
各モジュールはファイルとして磁気ディスクなどの二次記憶装置に格納されており、オペレーティングシステムローダ、または、カーネルにより主記憶にロードされる。
ロードモジュール管理テーブル２００は、主記憶にロードされている各モジュールのモジュール名称２０１、モジュールのコード領域開始仮想アドレス２０２、コード領域のサイズ２０３、データ領域開始仮想アドレス、データ領域の大きさ２０５、および、モジュールインターフェイス情報２０６を含んでいる。
モジュールインターフェイス情報は、ロードされるモジュールがカーネルに提供するインターフェイスルーチンの開始アドレスである。
インターフェイスルーチン２０６を登録することにより、カーネルはロードされたモジュールの機能を利用できるようになる。

【００１８】
図２の２０７から２１１は、モジュール名称がデバイスドライバ１であるモジュールのコード開始アドレス、サイズ、データ開始アドレス、サイズ、および、インターフェイスルーチン情報を格納している。
この図のロードモジュール管理テーブル２００では、カーネル、再起動ローダ、デバイスドライバ１、およびデバイスドライバ２がカーネル空間にロードされていることを示している。
２２０は、カーネルの仮想アドレス空間の空き領域を表現するカーネル仮想空間空き領域リストを保持している。
２２０は空き領域を表現する空きブロック構造体２３０を指している。空きブロック構造体２３０は、空き領域の開始仮想アドレス２３２、その空き領域のサイズ２３３、および、次の空き領域を表現する空きブロック構造体へのポインタ２３１を含んでいる。２３１は次の空きブロック構造体である２４０のアドレスを格納している。
この図の２２０では、それぞれアドレス２３２、２４２から始まり、大きさが２３３、２４３である２つの連続した空き領域がカーネルの仮想アドレス空間にあることを示してい

る。
２５０は、物理メモリの空き領域を表現する物理メモリ空き領域リストである。物理メモリ空き領域リストも、カーネル仮想空間空き領域リストと同様に構成される。
空きブロック構造体２６０、および、２７０がそれぞれ物理メモリの空き領域を表現している。この図の２５０も、２２０と同様に２つの連続した空き物理メモリがあることを表現している。
【００１９】
次に、計算機に接続している外部デバイスを管理するデータ構造について説明する。
図３は、本発明の実施の形態の計算機に接続している外部デバイスが占有する資源を表現するデータ構造を示している。
３００は、外部デバイスの制御用のレジスタが存在するアドレス範囲を表現するデバイス占有アドレスリストである。
デバイス占有アドレスリスト３００は、１つのアドレス範囲を表現するデバイス占有アドレス構造体から成り、図３では、３１０、３２０、３３０の３つのデバイス占有アドレス構造体から構成されている。
デバイス占有アドレス構造体３１０は、デバイス制御レジスタ開始アドレス３１２、大きさ３１３、そのアドレス範囲により制御されるデバイスを管理しているモジュールのモジュール番号３１４、および、デバイス占有アドレスリストを構成するためのリンク３１１を含んでいる。
モジュール番号３１４は、図２のロードモジュール管理テーブル２００へのインデックスとなる。
例えば、デバイス占有アドレス構造体３１０がカーネルが管理している外部デバイスの占有資源を表現しているとすると、３１４のモジュール番号には０が格納される。デバイスドライバ１の管理するデバイスがあるならば、そのモジュール番号には２が格納される（ロードモジュール管理テーブル２００へのインデックスは０から数えるとする）。
【００２０】
３４０は、外部デバイスが占有する割り込み番号を記録するデバイス占有割り込みテーブルである。
デバイス占有割り込みテーブル３４０は、プロセッサが認識する割り込み番号が、どのモジュールの管理するハードウェアデバイスに占有されているかを記録している。
例えば、カーネルが管理するクロック割り込み用デバイスが割り込み番号０を占有しているならば、デバイス占有割り込みテーブルの０番目のエントリ３４１に、カーネルのモジュール番号である０が格納される。
【００２１】
次に、プロセッサが直接利用するデータ構造について説明する。
図４は、プロセッサが利用する仮想アドレス変換テーブル、および、割り込み処理用テーブルのデータ構造を示している。
４１０は、プロセッサの仮想アドレス－物理アドレス変換を規定するページテーブルである。
ページテーブル４１０のエントリは、プロセッサの規定するページサイズ毎に存在する。各エントリは、仮想アドレス空間内のそれぞれの仮想ページについて、そのエントリが有効であるかを示すフラグ４１１、そのページが書き込み可能であるかを示すフラグ４１２、そのエントリの仮想ページに対応する物理ページの開始アドレス４１３を含んでいる。
４００は、プロセッサのレジスタで、ページテーブル４１０の開始アドレスを格納している。
プロセッサは、仮想アドレスモードで動作している時、ページテーブルレジスタ４００を参照して、仮想アドレスから実際のメモリアクセスに必要な物理アドレスを生成する。
【００２２】
４３０は、プロセッサに入る割り込みについて、割り込み番号毎の割り込みハンドラを規定する割り込みハンドラテーブルである。

割り込みには要因毎に番号が振られている。割り込みコントローラ１０４は、外部デバイスからの割り込み要求を受け、割り込み番号に変換してプロセッサに通知する。
割り込みハンドラテーブル４３０は、それぞれの割り込み番号毎の割り込みハンドラの開始アドレスを格納している。
例えば、クロック割り込みが０番の割り込みを利用しているならば、割り込みハンドラテーブルの０番目のエントリ４３１には、クロック割り込みハンドラのアドレスが格納されている。
４２０は、プロセッサのレジスタで、割り込みハンドラテーブル４３０の開始アドレスを格納している。
プロセッサは、割り込みハンドラテーブルレジスタ４２０を参照して、割り込みを検出した時に要因毎の割り込みハンドラに制御を移す。
例えば、プロセッサがクロック割り込み、つまり０番の割り込みを検出した時、プロセッサは割り込みハンドラテーブルレジスタ４２０が指すテーブル４３０の０番エントリ４３１に格納されているハンドラに制御を移す。

【００２３】

次に、再起動ローダが管理するデータ構造について説明する。
再起動ローダは、オペレーティングシステムのカーネルがソフトウェア障害により停止し、オペレーティングシステムを再起動する時に、オペレーティングシステムカーネルを主記憶にロードし、カーネルを実行するモジュールである。
図５は、再起動ローダが管理するデータ構造を示している。
５００は無停止モジュール管理テーブルであり、再起動ローダによりカーネルの再ロード、再起動中もハードウェアデバイスからの割り込み受け付けて処理を実行するモジュールである無停止モジュールの情報を管理するデータ構造である。
無停止モジュール管理テーブル５００は、無停止モジュール名５０１、無停止モジュールのコード領域開始アドレス５０２、コード領域の大きさ５０３、データ領域の開始アドレス５０４、データ領域の大きさ５０５、無停止モジュールが管理する外部デバイスが占有する資源の情報５０６、および、無停止モジュールの再初期化ルーチンのアドレスを含んでいる。
利用ハードウェア情報５０６には、無停止モジュールが管理する外部デバイスの制御用占有アドレス、占有割り込み番号が記録されている。
再初期化ルーチン５０７には、カーネルが再起動したときに実行されるルーチンのアドレスを格納する。
再初期化ルーチンが何を実行するかは、モジュールが管理しているハードウェアに依存するが、少なくとも、ロードモジュール管理テーブル２００へのインターフェイスルーチン２０６の登録を実行する。これにより、カーネルは再び無停止モジュールの提供する機能を利用できるようになる。

【００２４】

図５の無停止モジュール管理テーブル５００では、再起動ローダ、および、デバイスドライバ１が無停止モジュールとして登録されている。
更に、５０８ないし５１３には、デバイスドライバ１の占有アドレス、外部デバイス情報、再初期化ルーチンアドレスが格納されている。
再起動ローダは、無停止モジュール管理テーブル５００の情報を利用して、オペレーティングシステムカーネル再起動時のカーネル空間、および、外部デバイス構成情報を構築してカーネルに引き渡す。
カーネルが、これらの情報を参照して起動手順を決定することで、カーネル再起動中でも中断することなく、外部デバイスからの割り込み処理を可能にする。
無停止モジュール管理テーブル５００は、再起動ローダのデータ領域に置かれるデータ構造で、再起動ローダモジュールにより管理される。
再起動ローダモジュールは、モジュールの初期化時に再起動ローダのデータ領域を書き込み禁止とし、ソフトウェア障害による無停止モジュール管理テーブル５００の破壊を防ぐ

。
データ領域を書き込み禁止にするには、データ領域を含む仮想ページに対応するページテーブル４１０のエントリの書き込み可フラグをリセットすることで実現できる。
【００２５】
次に、カーネルが起動時に読み込むロードモジュールを定義するファイルの形式について説明する。
図６は、本発明の実施の形態のロードモジュールを定義するデータ構造を示している。
１１０は、モジュール構成ファイルの内容を示している。モジュール構成ファイル１１０の各エントリは、カーネルがロードするモジュールの名称６０１、モジュールが格納されているファイル名６０２、および、そのモジュールが無停止モジュールであるかを示すフラグ６０３を含んでいる。
図６の例では、カーネルは再起動ローダ、デバイスドライバ１、デバイスドライバ２の順でモジュールを主記憶に読み込む。デバイスドライバ１が格納されているファイルの名前はｄｒｉｖｅｒ１で、デバイスドライバ１は無停止モジュールであることを表現している。
モジュール構成ファイル１１０は、オペレーティングシステムが予め定めた名前のファイルに格納され、カーネルは容易にこのファイルを発見できる。
【００２６】
図７は、本発明の実施の形態のロードモジュールが格納されるファイルの形式を示している。
ロードモジュールを格納しているファイル７００は、モジュールの実行コードが格納されているコード開始オフセット７０１、実行コードのサイズ７０２、データが格納されているデータ開始オフセット７０３、データ領域のサイズ７０４、モジュールの初期化ルーチンの実行コードが格納されている初期化ルーチンオフセット７０５、モジュールの再配置情報開始オフセット７０６、再配置情報のサイズ７０７、実行コード７０８および、データ７０９を保持している。
モジュールの再配置情報は７１０は、モジュールを主記憶にロードする時に利用するデータで、モジュールのコード、および、データ領域がロードされたアドレスにしたがってモジュールのコードを変更するためのデータである。
これにより、ロードするモジュールの構成が変わってモジュールがロードされるアドレスが変わっても良い。
【００２７】
次に、本発明の実施例のオペレーティングシステム再起動の手順について説明する。
図８は、オペレーティングシステムを再起動する時に実行される再起動ローダの処理手順を示すフローチャートである。
まず、ステップ８０１でカーネル空間に再起動ローダがロードされているか検査する。再起動ローダがロードされていない、あるいは、ロードされているかどうか判別できない場合は、ステップ８０２へ進む。ステップ８０２では、プロセッサをリセットして計算機の再起動を実行する。
多くの計算機ではプロセッサをリセット状態にすると、仮想アドレス変換が禁止になり、割り込みハンドラの設定も無効になり、プロセッサが規定する物理アドレスに制御を移す。通常、この物理アドレスには計算機の起動手順を記憶した読み込み専用メモリ１０３がマップされている。
この起動手順は、計算機に接続しているハードウェアデバイスをリセットする。これは、オペレーティングシステムのカーネル実行時に外部デバイスを既知の状態にするためである。
この起動手順のために、特に、プロセッサがリセットされてしまうために、従来の計算機ではオペレーティングシステム再起動の間、外部デバイスからの割り込み処理を受け付けることができなくなってしまう。
一方、ステップ８０１で再起動ローダがロードされていると判断された場合は、プロセッ

サをリセットすることなく、ステップ８０３に進む。ステップ８０３からが、実際の再起動ローダが実行する処理である。

【００２８】

ステップ８０３では、ロードモジュール管理テーブル２００に登録されているモジュールがハードウェアリセットルーチンをもっているか検査し、登録されていれば、そのリセットルーチンを呼び出す。

特別なハードウェアリセットが必要ないデバイスの場合は、リセットルーチンは登録しなくてもよい。

次のステップ８０４では、無停止モジュール管理テーブル５００の各エントリの利用ハードウェア情報５０６を参照して、無停止モジュールが受け付ける以外の割り込みについて、それらの割り込みの割り込みハンドラを無効にする。

具体的には、割り込みハンドラテーブル４３０のエントリを、割り込みを受け付けるだけの割り込みハンドラのアドレスに設定する。

次のステップ８０５では、この後に続く無停止モジュールのデータ領域へのデータ格納に備えて、無停止モジュールのデータ領域の格納されているページを書き込み可能に設定する。

具体的には、ページテーブル４１０の、無停止モジュールのデータ領域に対応するエントリの書き込み可フラグ４１２をセットする。

【００２９】

続くステップ８０６、８０７、および、８０８では、カーネルが再起動時に参照するカーネル空間空き領域リスト、物理メモリ空き領域リスト、デバイス占有アドレスリスト、および、デバイス占有割り込みテーブルを作成する。

ステップ８０６では、仮想メモリ空き領域リスト、および、物理メモリ空き領域リストを作成する。

ステップ８０６では、無停止モジュール管理テーブル５００を参照して、無停止モジュール、再起動ローダ、およびカーネルスタック１１６が利用している以外のアドレス領域を空き領域とする空き領域リストを、再起動ローダのデータ領域内に作成する。

空き領域リストは、オペレーティングシステムの管理する空き領域リスト２２０、および、２５０と同じデータ構造である。

ステップ８０７では、デバイス占有アドレスリストを作成する。

ステップ８０７では、ステップ８０６と同様、無停止モジュール管理テーブル５００を参照して、無停止モジュールが利用している以外のアドレス領域を空き領域とするデバイス占有アドレスリストを再起動ローダのデータ領域内に作成する。

ここで作成するデバイス占有アドレスリストは、オペレーティングシステムの管理するデバイス占有アドレスリスト３００とほぼ同じデータ構造であるが、デバイス占有アドレス構造体のモジュール番号については、モジュール番号ではなくモジュール名称を格納する。

ステップ８０８では、デバイス占有割り込みテーブルを作成する。

ステップ８０７と同様に、無停止モジュールが利用している以外の割り込み番号を未使用とするデバイス占有割り込みテーブルを、再起動ローダのデータ領域内に作成する。

ここでも、ステップ８０７と同様、デバイス占有割り込みテーブルの各エントリには、モジュール番号ではなくモジュール名称を格納する。

【００３０】

ステップ８０６ないし８０８で作成したデータ構造は、カーネルが再起動した時に、無停止モジュールが管理しているハードウェアデバイス資源を、誤って他のモジュールに割り当てないようにするためのデータ構造である。

最後に、再起動ローダは、主記憶にオペレーティングシステムカーネルをロードし（ステップ８０９）、再起動ローダの再初期化ルーチンのアドレスをパラメータに加えてカーネルを実行する（ステップ８１０）。

【００３１】

次に、本発明の実施例のオペレーティングシステムカーネルの初期化処理について説明する。
図９は、本発明の実施例のオペレーティングシステムカーネルの初期化処理手順を示すフローチャートである。
まず、カーネルは起動時に再起動ローダによる起動か、通常の起動かを判別する（ステップ９０１）。
これは、再起動ローダによる起動では、再起動ローダの再初期化ルーチンのアドレスをパラメータとしてカーネルに渡すので、容易に判別できる。
再起動ローダによる起動でない場合は、通常手順によりカーネルの実行を進める（ステップ９０２）。
ここでの通常手順とは、バス、および、割り込みコントローラの初期化、カーネル管理のハードウェアデバイスの初期化、モジュール構成ファイルに記されたモジュールの主記憶へのロードと初期化、初期プロセスの実行である。
【００３２】
再起動ローダによる起動の場合は、ステップ９０３へ進む。ステップ９０３では、再起動ローダモジュール用にロードモジュール管理テーブル２００のエントリを割り当てる。
続くステップ９０４で、パラメータとして渡された再起動ローダモジュールの再初期化ルーチンのアドレスを基に再初期化ルーチンを実行する。
再起動ローダの再初期化ルーチンは、無停止モジュールの再初期化ルーチンと同様の処理を実施する。
この再初期化ルーチンでは、ロードモジュール管理テーブル２００の設定を実施する。特に、管理テーブル２００のモジュールインターフェイス２０６が設定され、再起動ルーチンが提供するインターフェイスルーチンをカーネルから呼び出せるようになる。
続く、ステップ９０５でハードウェア構成情報のコピー、ステップ９０６と９０７で再起動ローダがカーネル起動前に構築した空き領域リスト、デバイス占有アドレスリスト、デバイス占有割り込みテーブルをカーネルのデータ空間にコピーする。
これらの処理は、再起動ローダが提供するインターフェイスルーチンにより実行される。
【００３３】
通常起動時には、カーネルがロードされている以外のメモリ領域を空き領域とする空き領域リスト、空のデバイス占有アドレスリスト、および、すべての割り込みのエントリが未使用であるデバイス割り込みテーブルを作成して、以降の処理を実行する。
それに対し、再起動ローダによるカーネル再起動時には、再起動ローダが作成した空き領域リスト、デバイス占有アドレスリスト、および、デバイス占有割り込みテーブルを利用する。
これにより、無停止モジュールのデータ領域の初期化、外部デバイスの初期化を避け、無停止モジュールの外部デバイスに関連する処理の連続性を保持することが可能になる。
【００３４】
続くステップ９０８から９１２は、カーネルが管理するバスと割り込みコントローラ以外のすべての外部デバイスについての初期化処理である。
ステップ９０９では、外部デバイスが利用するアドレス領域、および、割り込み番号が、既に利用されているか検査する。
これは、デバイス占有アドレスリスト３００、および、デバイス占有割り込みテーブル３４０を参照することにより検査する。再起動ローダによる再起動時には、これらのデータ構造に無停止モジュールが利用しているデバイス情報が含まれていることになる。
外部デバイスの使用するアドレス領域、および、割り込み番号が未使用であるならば、そのアドレス領域、および、割り込み番号をデバイス占有アドレスリスト３００と、デバイス占有割り込みテーブル３４０に、カーネルのモジュール番号とともに登録する（ステップ９１０）。
カーネルのモジュール番号は、ロードモジュール管理テーブル２００を検索することによって得られる。さらに、対象となっている外部デバイスの初期化を実施し（ステップ９１

１）、ステップ９１２へ進む。
【００３５】
もし、初期化処理中の外部デバイスが利用するアドレス、あるいは、割り込み番号が既にデバイス占有アドレスリスト３００、および、デバイス占有割り込みテーブル３４０に登録されているならば、これは、無停止モジュールが利用しているデバイスであるか、ハードウェア構成に誤りがあるかのどちらかである。
もし、初期化処理中の外部デバイスが利用するアドレス、割り込み番号が既にデバイス占有アドレスリスト３００、および、デバイス占有割り込みテーブル３４０に登録されている場合は、ステップ９１０、および、９１１は実行せずにステップ９１２へ進む。
これにより、無停止モジュールが管理している外部デバイスの初期化を避け、無停止モジュールのデバイスに関連する処理の連続性が保持される。
ステップ９１２では、処理対象デバイスを他のデバイスに設定して、ステップ９０８へ進む。処理対象デバイスが残っていなければ、ステップ９０８の検査からステップ１００１へ処理を進める。
【００３６】
ステップ１００１からの処理のフローチャートは図１０に示す。
図１０のフローチャートは、主に、モジュールのロード処理に関する。
ステップ１００１では、モジュール構成ファイル１１０をメモリに読み込む。
続くステップ１００２ないし１０１１はモジュール構成ファイルに登録されている各モジュールについて実行する。ステップ１００２と１００８により、各モジュール毎に処理を実行するようループを構成する。
ループ内の最初のステップ１００３では、モジュール構成ファイル１１０の無停止フラグ６０３を参照して、処理対象のモジュールが無停止モジュールかどうか検査する。無停止モジュールでない場合は、ステップ１００４へ進む。
【００３７】
ステップ１００４から始まる処理は、モジュールの主記憶へのロードを実施する。
まず、ステップ１００４で必要なメモリ領域を獲得する。これは、空き領域リスト２２０、および、２５０を参照してモジュールを格納するのに十分な空き領域を探し、見つけた領域を空きリストから外すことで実施する。モジュールを格納するのに必要な領域は、モジュールを格納しているファイルの先頭部分７０２、および、７０４を参照して求める。
メモリの割当では、ページテーブル４１０の設定も行なう。
割り当てたアドレス領域に対応するページテーブル４１０のそれぞれのエントリについて、有効フラグ４１１をセットし、割り当てた物理ページの先頭アドレスを４１３に格納する。
さらに、書き込み可能フラグ４１２もセットする。コード領域が格納されているページの書き込み可能フラグ４１２は、コード領域のロード後リセットする。
【００３８】
続くステップ１００５で、モジュール構成ファイル１１０のファイル名６０２で示されるファイルを、ステップ１００４で割り当てた領域にロードする。
ロード後、ファイルに格納されている再配置情報７０６、７０７、および、７１０によりコード領域を修正する。
さらに、ロードモジュール管理テーブル２００に処理中のモジュール用のエントリを割り当て、そのエントリのアドレス、および、サイズ情報を設定する（ステップ１００６）。
次に、モジュールの初期化ルーチンを実行する（ステップ１００７）。
モジュールの初期化ルーチンは、少なくとも、ロードモジュール管理テーブル２００のモジュールインターフェイス２０６を設定する。もし、その他のモジュールが必要とする初期化処理があれば、それも実行する。
モジュールインターフェイス２０６が設定されると、カーネルはモジュールが提供する処理ルーチンのアドレスを知ることができ、モジュールの提供する処理を実行できるようになる。

続くステップ１００８では、処理対象モジュールを次のモジュール構成ファイルのエントリに設定して、ステップ１００２へ戻る。

【００３９】

次に、処理対象モジュールが無停止モジュールである場合について説明する。

ステップ１００３で、処理対象モジュールが無停止モジュールであると判定された場合、ステップ１００９へ進む。

ステップ１００９では、ロードモジュール管理テーブル２００の更新処理をする。

まず、ロードモジュール管理テーブル２００に処理中のモジュールのデータを格納するエントリを割り当てる。

割り当てたエントリに格納すべき２０２、ないし、２０５のアドレス、および、サイズ情報は、再起動ローダのデータ領域内の無停止モジュール管理テーブル５００の５０２、ないし、５０５に格納されており、これを割り当てたエントリにコピーし、ロードモジュール管理テーブル２００を更新する。

【００４０】

次に、無停止モジュールの再初期化ルーチンを実行する（ステップ１０１０）。

再初期化ルーチンのアドレスは、無停止モジュール管理テーブル５００の再初期化ルーチン５０７に格納されており、無停止モジュール名称より再初期化ルーチンを得ることができる。

無停止モジュールの再初期化ルーチンは、少なくとも、ロードモジュール管理テーブル２００のインターフェイスルーチン２０６の設定を実行する。必要であれば、ハードウェアデバイスの設定などその他の処理を行なう。

この時に、無停止モジュールは新しくロードされないため、これまでのデータ領域は保存され、無停止モジュールのデバイスに関連する処理の連続性を保持することができる。

【００４１】

続くステップ１０１１では、デバイス占有アドレスリスト３００、および、デバイス占有割り込みテーブル３４０の内、処理対象の無停止モジュールが管理するアドレス領域、および、割り込み番号を表現しているエントリのモジュール番号の欄に、ステップ１００９で割り当てたロードモジュール管理テーブル２００のエントリ番号を格納する。

無停止モジュールが管理するデバイスアドレス領域と割り込み番号は、無停止モジュール管理テーブル５００の利用ハードウェア情報５０６より得られる。

空き領域リスト２２０と２５０については、再起動ローダが無停止モジュールのあるアドレス領域を空きリストから外しているので、特別な処理をする必要はない。

続いてステップ１００８へ進み、次のロードモジュールの処理を行なう。

ステップ１００２で、すべてのモジュールをロードしたと判定されたら、ステップ１０１２へ進む。ステップ１０１２は初期プロセスの作成と実行を行ない、オペレーティングシステムの起動を完了する。

【００４２】

次に、無停止モジュールの初期化ルーチンについて説明する。

モジュールの初期化ルーチンはすべてのモジュールが持たなければならないルーチンで、カーネルの起動時に実行される。

図１１は、本発明の実施の形態の、無停止モジュールの初期化ルーチンの処理を示すフローチャートである。

ここでは、モジュール構成ファイル１１０に設定されているデバイスドライバ１の初期化ルーチンであるとして説明する。

まず、デバイスドライバ１のカーネルへのインターフェイスとなるモジュールインターフェイスを、ロードモジュール管理テーブル２００のデバイスドライバ１用に割り当てたエントリ２１１に格納する（ステップ１１０１）。

次のステップ１１０２では、デバイスドライバ１用のデバイス占有アドレス構造体を割り当て、開始アドレス、サイズ、モジュール番号を設定し、デバイス占有アドレスリスト３００に追加する。

さらに、デバイス占有割り込みテーブル３４０の、デバイスドライバ１が管理する割り込み番号のエントリにデバイスドライバ１のモジュール番号を格納する。
これにより、デバイスドライバ１の利用するデバイスアドレス領域、および、割り込み番号を、デバイス占有アドレスリスト３００、および、デバイス占有割り込みテーブル３４０に登録する。
続いて、モジュールの無停止モジュール管理テーブル５００への登録を行なう（ステップ１１０３）。
無停止モジュール管理テーブルに現在処理中のモジュール用のエントリを割り当て、アドレス、サイズ情報、利用ハードウェア情報、および、再初期化ルーチンのアドレスを設定する。設定は、再起動ローダの提供するインターフェイスルーチンにより実行する。
無停止モジュール管理テーブル５００を更新する場合、無停止モジュール管理テーブル５００のあるページは書き込み禁止に設定されているので、更新の前後でページテーブル４１０の書き込み可フラグを操作する。

【００４３】

次に、再起動ローダが提供するインターフェイスルーチンについて説明する。再起動ローダはそれ自身が無停止モジュールであり、初期化ルーチン、再初期化ルーチン、ハードウェア構成データコピールーチン、無停止モジュール登録ルーチンを提供する。
まず、再起動ローダの初期化ルーチンの処理について説明する。
図１２は、再起動ローダの処理手順を示すフローチャートである。
再起動ローダは無停止モジュールであり、再起動ローダの初期化ルーチンは、計算機が起動された時にのみ呼ばれる。
まず、再起動ローダの初期化ルーチンでは、以降の再起動処理に備えて、読み込み専用メモリ１０３に格納されている初期プログラムが作成したハードウェア構成情報を、再起動ローダのデータ領域にコピーする（ステップ１２０１）。
次に、無停止モジュール管理テーブル５００の初期化（ステップ１２０２）、再起動ローダ自身の無停止モジュール管理テーブルへの登録（ステップ１２０３）を実行する。
最後に、無停止モジュール管理テーブル、および、ハードウェア構成データが格納されている再起動ローダのデータ領域に割り当てられているページを書き込み禁止に設定して（ステップ１２０４）終了する。

【００４４】

再初期化ルーチンは、他の無停止モジュールと同じであり、ロードモジュール管理テーブルへのインターフェイスルーチンの登録処理をする。
ハードウェア構成データコピールーチンは、再起動ローダのデータ領域に格納されているハードウェア構成データをカーネルのデータ領域にコピーする。
無停止モジュール登録ルーチンは、無停止モジュールに関するデータを無停止モジュール管理テーブル５００に登録する。
通常、再起動ローダのデータ領域は仮想アドレス機構により書き込み禁止に設定されているので、登録ルーチンは書き込み禁止を一旦解除して、無停止モジュールデータをテーブル５００に書き込み、再びデータ領域を書き込み禁止とする。

【００４５】

この実施の形態によれば、オペレーティングシステムカーネルがソフトウェア障害により停止して再起動処理を実施しても、無停止モジュールが利用するとして登録した外部デバイスの割り込みの処理を中断することなく実施することができる。また、プロセッサリセット時に実行されるハードウェア構成認識処理を回避でき、再起動までの時間を短縮できる。

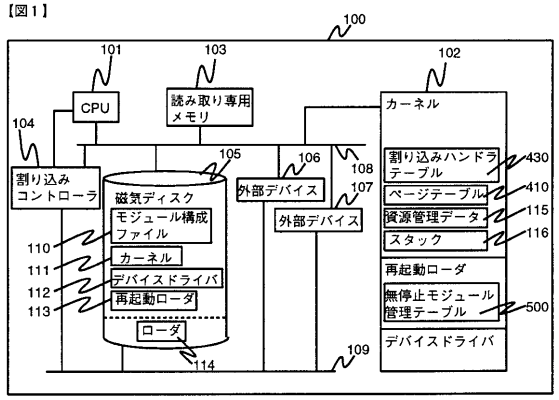
【００４６】

次に、本発明を、クロック割り込みを契機に外部デバイスの制御を実施するモジュールを無停止モジュールとする実施形態について述べる。
クロック割り込みは、通常カーネル本体が管理する割り込みである。時間制約の厳しい処理を実施する場合に、クロック割り込みをカーネルに渡す前に横取りして、時間制約付き

の処理を優先して実施するモジュールを利用する方式がある。割り込みを横取りするとは、割り込みハンドラの設定を変更してしまうことをいう。このような時間制約の厳しい制御においては、時間制約が厳しいと同時に、クロック割り込みに対する処理の中断が許されない場合がある。
本発明によれば、クロック割り込みを横取りするモジュールを無停止モジュールとすることで、カーネルがソフトウェア障害により停止して再起動することになっても、クロック割り込みに対する処理を中断することなく継続することができる。
クロック割り込み処理モジュールを無停止モジュールとして登録する時に、クロック割り込みの割り込み番号、および、クロック割り込みの制御アドレスをモジュールの占有資源としてデバイスとして無停止モジュール管理テーブル５００に登録する。これにより、カーネルの再起動の前後で、クロック割り込みの処理環境は引き継がれる。また、カーネルの再起動時に、カーネルが自身の管理デバイスであるクロック割り込みが他のモジュールに占有されていることを認識し、クロック割り込みに関する設定を変更せずに起動処理を進める。
これにより、オペレーティングシステムカーネルがソフトウェア障害により停止し再起動されても、クロック割り込み処理は中断することなく、信頼性が高く、高可用な制御システムを実現できる。
【００４７】
本発明の第二の実施の形態について説明する。
図１３はこの実施形態における計算機の起動手順に示すフローチャートである。
ステップ１３０１から１３０３にて通常のカーネル初期化処理であるカーネルデータの初期化（ステップ１３０１）、外部デバイスの初期化（ステップ１３０２）、および、カーネルモジュールのロードと初期化（ステップ１３０３）を実施する。
モジュールの初期化は、第一の実施の形態でのように、モジュールが無停止モジュールであるならば、無停止モジュール管理テーブル５００にその利用アドレス情報を登録する。
次のステップ１３０４では、主記憶１０２のうち、カーネル、および、その他のモジュールがロードされている領域で、無停止モジュールのデータ領域以外の領域を磁気ディスク１０５の再起動ファイルに記録する。
さらに、ステップ１３０５では、ページテーブルレジスタ、および、カーネルスタックのスタックポインタ値を磁気ディスク１０５の再起動ファイルに保存し、ステップ１３０６で次のステップ１３０７のアドレスを磁気ディスク１０５の再起動ファイルに保存する。
ここで作成された再起動ファイルは、カーネルがソフトウェア障害のための再起動する時にカーネルのソフトウエア構成を再現するために利用する。
【００４８】
図１４は、本発明の第二の実施形態におけるオペレーティングシステムカーネルの再起動手順を示すフローチャートである。
この再起動手順は、無停止モジュールとして登録されている再起動ローダにより実施される。図にしたがって再起動手順を説明する。
まず、ページテーブルレジスタ４００の指しているページテーブル４１０を、再起動ローダのデータ領域にコピーし（ステップ１４０１）、ページテーブルレジスタ４００がコピーされた方のページテーブルを指すように変更する（ステップ１４０２）。
次に、磁気ディスク１０５の再起動ファイルに記録されたカーネル起動時の主記憶１０２の内容を、主記憶１０２にコピーする（ステップ１４０３）。続いて、ページテーブルレジスタとスタックポインタ、および、ステップ１３０７のアドレスを磁気ディスク１０５の再起動ファイルより取得する（ステップ１４０４、１４０５）。
続くステップ１４０６で、取得したページレジスタ値、および、スタックポインタ値をレジスタに設定し、ステップ１３０７へ処理を移す（ステップ１４０７）。
第２の実施形態では、磁気ディスク１０５の再起動ファイルは、無停止モジュールとして登録されているモジュールのデータ領域を含んでいないため、カーネル再起動の前後で無停止モジュールのデータ領域は主記憶１０２に保存されている。これにより、無停止モジ

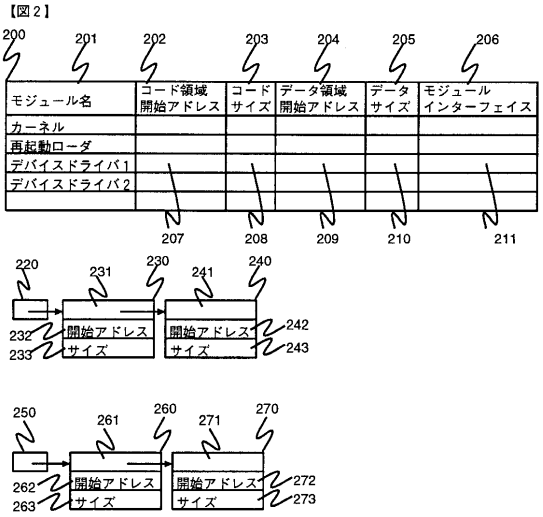
ュールの外部割り込み処理の連続性を維持することができる。
【００４９】
また、カーネルがソフトウェア障害を検出したときに、プロセッサのリセットを行わないようにし、カーネル再起動中も特定の割り込みに対する処理ができるように割り込み処理に利用される主記憶の内容を主記憶中に維持したまま、かつ割り込みハンドラの設定を主記憶中に維持したままカーネルの処理を停止するようにしてもよい。
そして、カーネルを再起動するとき、主記憶中に格納された割り込み処理用の主記憶の内容、および割り込みハンドラの設定を、そのまま引き継ぐようにしている。
【００５０】
【発明の効果】
オペレーティングシステムのカーネルがソフトウェア障害を検出し再起動する時に、無停止モジュール管理テーブル５００を参照して、カーネル再起動時の外部デバイスの初期化手順、および、ロードするモジュールの構成を決定する手順を設けることにより、特定の外部デバイスが発生する割り込みに対する処理をカーネルの再起動処理中も中断することなく実施し続けることができる。
プロセッサをリセットすることなくオペレーティングシステムを再起動することにより、プロセッサリセット時に実行されるハードウェア構成検査等の処理を回避することができ、オペレーティングシステムが再起動するまでの時間を短縮できる。
【図面の簡単な説明】
【図１】本発明の実施の形態の適用される計算機の構成を示す図である。
【図２】ロードモジュール、および、空きメモリ空間を管理するデータ構造を示す図である。
【図３】ハードウェア占有資源を管理するデータ構造を示す図である。
【図４】ページテーブル、および、割り込みハンドラテーブルのデータ構造を示す図である。
【図５】無停止モジュールを管理するデータ構造を示す図である。
【図６】モジュール構成ファイルの形式を示す図である。
【図７】モジュールを格納しているファイルの形式を示す図である。
【図８】ソフトウェア障害検出時の処理フローチャートを示す図である。
【図９】カーネル再起動処理のフローチャートを示す図である。
【図１０】カーネル再起動処理の図９のフローチャートに続くフローチャートを示す図である。
【図１１】無停止モジュールの初期化手順のフローチャートを示す図である。
【図１２】再起同ローダの処理のフローチャートを示す図である。
【図１３】本発明の第二の実施の形態の計算機の起動手順の処理のフローチャートを示す図である。
【図１４】本発明の第二の実施の形態のカーネル再起動処理のフローチャートを示す図である。
【符合の説明】
１００　計算機
１０１　プロセッサ
１０２　主記憶装置
１０３　読み取り専用メモリ
１０４　割り込みコントローラ
１０５　磁気ディスク
１０６、１０７　外部デバイス
１０８　バス
１０９　割り込み信号バス
１１０～１１４　ファイル
２００～７０９　データ構造

【図１】



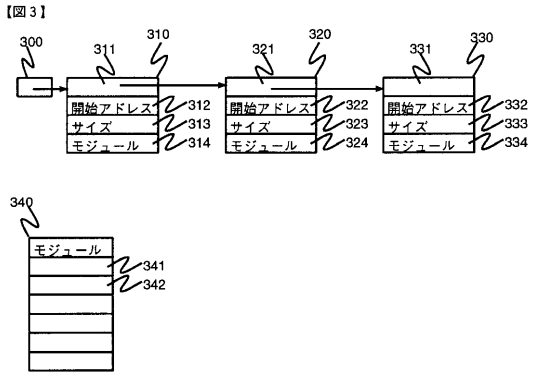
100: 計算機
102: 主記憶装置
108: バス
109: 割り込み信号バス

【図２】



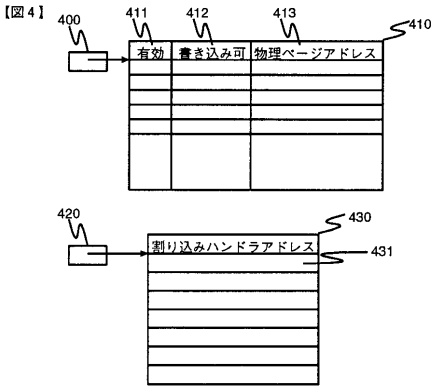
200: ロードモジュール管理テーブル
220: カーネル仮想空間空き領域リスト
230, 240: 空きブロック構造体
250: 物理メモリ空き領域リスト
260, 270: 空きブロック構造体

【図３】



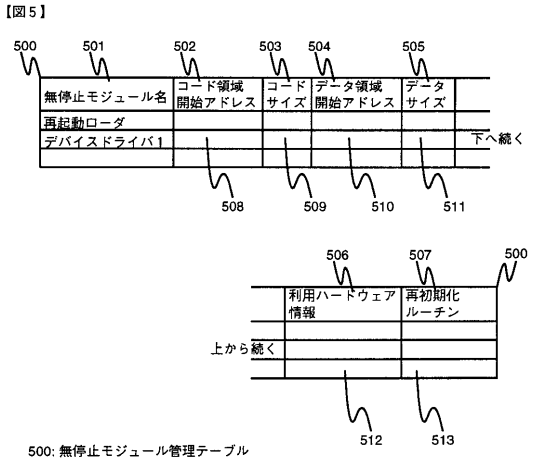
300: デバイス占有アドレスリスト
310, 320, 330: デバイス占有アドレス構造体
340: デバイス占有割り込みテーブル

【図４】

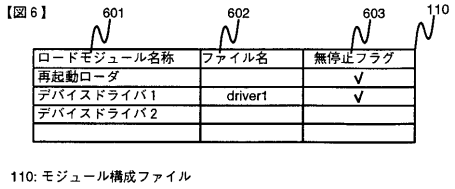


400: ページテーブルレジスタ　410: ページテーブル
420: 割り込みハンドラテーブルレジスタ　430: 割り込みハンドラテーブル

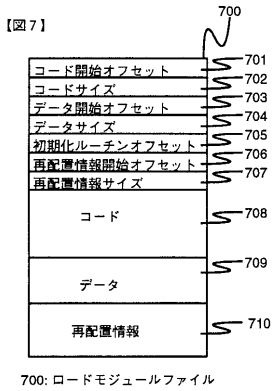
【図５】


【図5】
500
501
502
503
504
505
無停止モジュール名
コード領域開始アドレス
コードサイズ
データ領域開始アドレス
データサイズ
再起動ローダ
デバイスドライバ1
下へ続く
508
509
510
511
506
507
500
利用ハードウェア情報
再初期化ルーチン
上から続く
512
513
500: 無停止モジュール管理テーブル

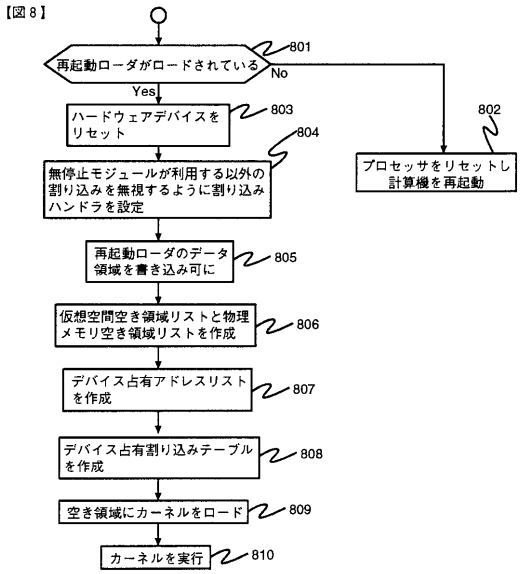
【図６】


【図6】
601
602
603
110
ロードモジュール名称
ファイル名
無停止フラグ
再起動ローダ
√
デバイスドライバ1
driver1
√
デバイスドライバ2
110: モジュール構成ファイル

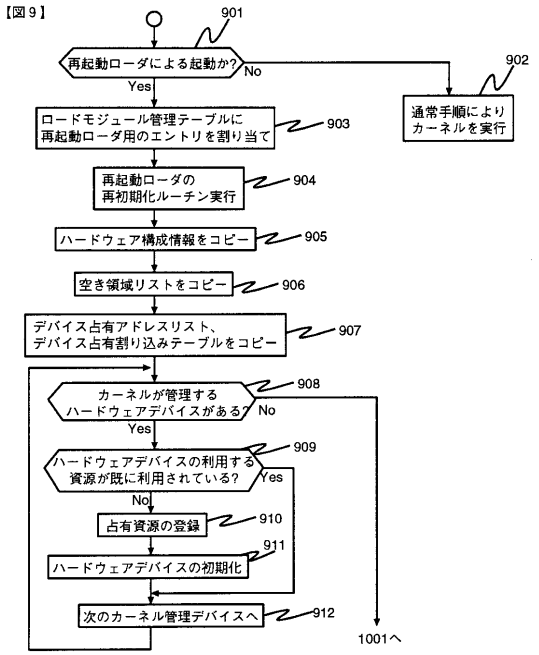
【図７】


【図7】
700
コード開始オフセット
701
コードサイズ
702
データ開始オフセット
703
データサイズ
704
初期化ルーチンオフセット
705
再配置情報開始オフセット
706
再配置情報サイズ
707
コード
708
データ
709
再配置情報
710
700: ロードモジュールファイル

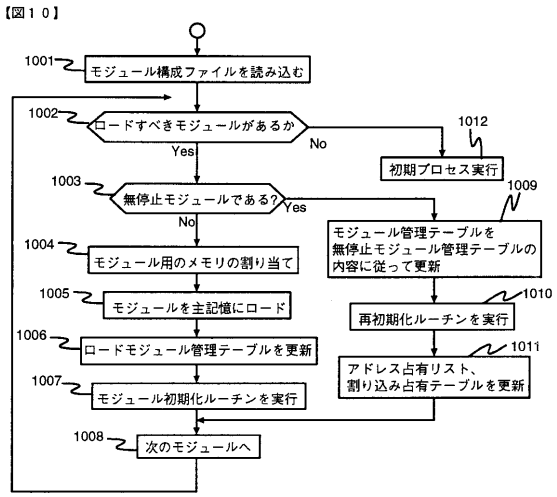
【図８】


【図8】
再起動ローダがロードされている
801
No
Yes
ハードウェアデバイスをリセット
803
804
802
プロセッサをリセットし計算機を再起動
無停止モジュールが利用する以外の割り込みを無視するように割り込みハンドラを設定
再起動ローダのデータ領域を書き込み可に
805
仮想空間空き領域リストと物理メモリ空き領域リストを作成
806
デバイス占有アドレスリストを作成
807
デバイス占有割り込みテーブルを作成
808
空き領域にカーネルをロード
809
カーネルを実行
810

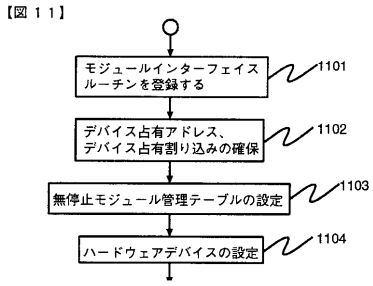
【図９】


【図9】
901
再起動ローダによる起動か?
No
902
Yes
通常手順によりカーネルを実行
ロードモジュール管理テーブルに再起動ローダ用のエントリを割り当て
903
再起動ローダの再初期化ルーチン実行
904
ハードウェア構成情報をコピー
905
空き領域リストをコピー
906
デバイス占有アドレスリスト、デバイス占有割り込みテーブルをコピー
907
カーネルが管理するハードウェアデバイスがある?
908
No
Yes
ハードウェアデバイスの利用する資源が既に利用されている?
909
Yes
No
占有資源の登録
910
ハードウェアデバイスの初期化
911
次のカーネル管理デバイスへ
912
1001へ

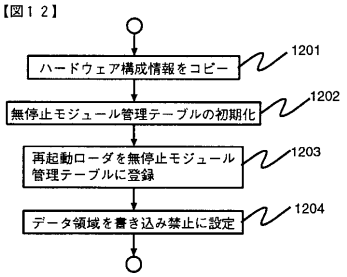
【図１０】



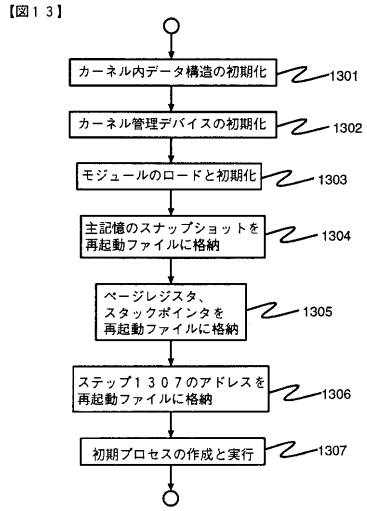
【図１１】



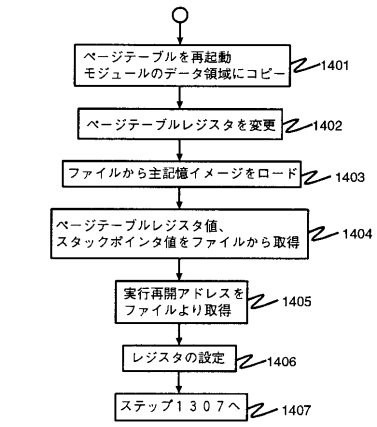
【図１２】



【図１３】



【図１４】



フロントページの続き

(56)参考文献 特開昭５９－１４０５６８（ＪＰ，Ａ）
特開平０６－２０２８９３（ＪＰ，Ａ）
特開平０３－１３０８３５（ＪＰ，Ａ）

(58)調査した分野(Int.Cl.[7]，ＤＢ名)
G06F 9/06
G06F 1/00
G06F 9/46 - 9/54
G06F 11/14