(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号
特許第4104746号
(P4104746)
(45)発行日　平成20年6月18日(2008.6.18)　(24)登録日　平成20年4月4日(2008.4.4)

(51)Int.Cl.　　F I
G06F 13/14 (2006.01)　　G06F 13/14 320E
G06F 13/36 (2006.01)　　G06F 13/36 320A

請求項の数 3　(全 42 頁)

(21)出願番号　特願平10-278559
(22)出願日　平成10年9月30日(1998.9.30)
(65)公開番号　特開平11-175454
(43)公開日　平成11年7月2日(1999.7.2)
審査請求日　平成17年8月9日(2005.8.9)
(31)優先権主張番号　940367
(32)優先日　平成9年9月30日(1997.9.30)
(33)優先権主張国　米国(US)

(73)特許権者　591030868
コンパック・コンピューター・コーポレーション
COMPAQ COMPUTER CORPORATION
アメリカ合衆国テキサス州７７０７０，ヒューストン，ステイト・ハイウェイ　２４９，２０５５５
２０５５５ State Highway 249, Houston, Texas 77070, United States of America
(74)代理人　100089705
弁理士　社本　一夫

最終頁に続く

(54)【発明の名称】自動直接メモリ・アクセス機能を備えたコンピュータ・システム

(57)【特許請求の範囲】
【請求項１】
自動直接メモリ・アクセス機能を備えたコンピュータ・システムにおいて、
ホスト・バスと、
ホスト・バスに接続され、パケットの少なくとも１つのプールを記憶する主メモリであって、各プールはパケットのリンク・リストで構成され、各パケットは、リンク・リストを構成するために次のパケットの物理アドレスを含んでいる物理アドレス・フィールドを備えたヘッダを備えている、主メモリと、
ホスト・バスに接続されたプロセッサと、
ローカル・バスと、
ローカル・バスに接続された大容量記憶サブシステムと、
ホスト・バスとローカル・バスとの間に接続され、大容量記憶サブシステムと主メモリとの間でパケットを通信するための分散型バースト・エンジンを含んでいるホスト／ローカル・バス・ブリッジと
からなり、分散型バースト・エンジンは、
リンク・リストの次のパケットの物理アドレスを保持する要求ヘッド・レジスタと、
リンク・リストの最後のパケットの物理アドレスを保持するフリー・キュー・レジスタと、
パケットを保持するための要求キューであって、パケットが保持されているときにキューの充ち具合を示す指示を提供する要求キューと、

処理すべき次のパケットが存在することを表す指示をホストから受け取る要求ドアベル・レジスタと、
完了リストの物理アドレスを保持する完了ヘッド・レジスタと、
パケットを保持する完了キューと、
完了キューが既に提供されかつプロセッサがパケットを受信する準備が完了していることを表す指示をプロセッサから受け取る完了ドアベル・レジスタと、
要求キューに接続され、要求ドアベル・レジスタの出力に応答して、要求ドアベル・レジスタが駆動された時に、要求ヘッド・レジスタに記憶された物理アドレスに応じて、主メモリからパケットを読み出して、要求キューに該パケットを記憶する第１のフロント・エンド・コントローラと、
要求キュー及び完了キューに接続され、キューの充ち具合を示す指示に応答して、要求キューが空ではないことを示しているときに、要求キューからパケットを読み出して大容量記憶サブシステムに提供し、また、大容量記憶サブシステムがパケットの処理を完了したときに、パケットを完了キューに供給する第１バック・エンド・コントローラと、
完了キューに接続され、完了ドアベル・レジスタ及び完了ヘッド・レジスタの出力に応答して、完了キューからのパケットを主メモリに戻す完了コントローラと
を含んでいることを特徴とするコンピュータ・システム。
【請求項２】
請求項１記載のコンピュータ・システムにおいて、分散型バースト・エンジンは、
リンク・リスト中の次のパケットの物理アドレスを保持するポステド・ヘッド・レジスタと、
プロセッサの請求に基づくことなくデータをプロセッサに転送する際に大容量記憶サブシステムにより使用されるパケットを保持するポステド・キューと、
主メモリ中の次のパケットが大容量記憶サブシステムのために取得できることを表すプロセッサからの指示を受け取るポステド・ドアベル・レジスタと、
ポステド・キューに接続され、ポステド・ドアベル・レジスタの出力に応答して、主メモリからポステド・パケットを受け取ってポステド・キューに書き込む第２フロント・エンド・コントローラと、
ポステド・キュー及び完了キューに接続され、かつ大容量記憶サブシステムの出力に応答し、ポステド・キューが空ではないときにポステド・キューからパケットを受け取って該パケットを大容量記憶サブシステムに提供し、大容量記憶サブシステムがパケットの処理を完了したときに、完了キューに該パケットを供給する第２バック・エンド・コントローラと
を備えていることを特徴とするコンピュータ・システム。
【請求項３】
請求項２記載のコンピュータ・システムにおいて、第２バック・エンド・コントローラはさらに、プロセッサがポステド・ドアベル・レジスタを駆動した場合に、主メモリからパケットを取得することを特徴とするコンピュータ・システム。
【発明の詳細な説明】
【発明の属する技術分野】
【０００１】
本発明は、インテリジェント入出力デバイスと通信する方法、及び入出力デバイスとホストとの間で自動的にデータ転送を行なう装置に関するものである。
【０００２】
【従来の技術】
コンピュータ・システムが発展するに連れて、その都度一層強力かつ先進の特徴が取り入れられている。ＰＣＩバス、即ち、周辺要素相互接続バスのような多くの新しい進んだバス構造が開発され、コンピュータ・システムの性能向上をもたらした。加えて、近年のパーソナル・コンピュータが接続及びマルチメディア指向システム(connected and multimedia oriented system)の色合いを強めるに連れて、ユニバーサル・シリアル・バス及びフ

ァイアウエア(Firewire)としても知られているＩ．Ｅ．Ｅ．Ｅ．１３９４（電気電子学会）のような新たなシリアル通信バスが開発されている。しかしながら、これらの投入は、その度毎に、プロセッサに対するデータの管理及び移動の要求を増々増大させることになる。

今日のオペレーティング・システムは、真のマルチタスク・オペレーティング・システムであり、所与のタイム・スライス間で多数のタスクのバランスを保つ役割を担っている。プロセッサに対して入出力動作を処理する要求が増大するに連れて、各タスクに適当なプロセッサ時間量を与えるようにタスクのバランスを適正に保つことが一層難しくなる。更に、割り込みが問題を複雑にする。何故なら、割り込みは予測不可能であり、通常素早くそれに対応しなければならないからである。

【０００３】

【発明が解決しようとする課題】

過去において、ある種の処理をコプロセッサに割り振る(load off)ことによって、プロセッサの負荷配分(loading)の問題を解決しようとする試みがなされている。算術演算用コプロセッサ(math co-processor)又はダイレクト・メモリ・アクセス・コントローラ（ＤＭＡ）が、よく知られている例である。しかしながら、算術演算用コプロセッサは、企業に固有のインターフェース及び命令によってプロセッサに強く結び付けられているので、汎用の入出力への応用には適していない。ＤＭＡは、一旦そのコンフィギュレーションが設定されたならば、データの移動には好適であるが、各データ・ブロック毎にコンフィギュレーションを必要とする。したがって、ＤＭＡコントローラは、単一のデータ・ブロックを超えて独立して機能することができない（そのコンフィギュレーションをデマンド・モード用に設定するのでない場合）。更に、ＤＭＡコントローラは、ハードウエアのレベルで機能する。ＤＭＡコントローラは、処理(service)対照のデータ型又はデバイス型についての理解(comprehension)を有していない。したがって、プロセッサが常に関与している。過去において、ＤＭＡコントローラは、データをあるアドレスから他のアドレスに移動させるには格段に効率的であったが、しかしながら、今日のプロセッサはそれよりもはるかに効率的であるので、そのリード／ライト・サイクル・タイムはＤＭＡコントローラのそれに近づきつつある。したがって、ＤＭＡコントローラの利点は、時とともに減少している。更に、高性能ＤＭＡコントローラによって与えられるいずれの価値(gain)も、典型的に、次世代プロセッサによって相殺されてしまっている。

他にも、インテリジェント入出力（I_2O）処理を開発する試みがなされたが、これらは、Intel i960RPプロセッサのような埋め込み型プロセッサを対象としたものであった。これらの解決策はプロセッサの独立を達成するが、それを行なうためのコストは非常に高い。したがって、インテリジェント入出力処理に対する安価な解決策が必要とされている。

【０００４】

【課題を解決するための手段】

本発明は、ホスト・プロセッサ、メモリ、及び１つ以上の入出力（Ｉ／Ｏ）デバイスを含む。ホスト・プロセッサはパケットを発生し、Ｉ／Ｏデバイスが後に使用するために、これらをメモリに提出する(submit)。パケットは、アドレス及びその他の情報を収容するためのヘッダ部分と、データ又はメッセージを搬送するためのペイロード部分とを含む。パケットは、リスト内において互いにリンクされ、パケット・プールを形成する。パケット・ヘッダは、物理アドレス・フィールド、次パケット物理アドレス・フィールド、仮想アドレス・フィールド、及び次パケット仮想アドレス・フィールドを含む。したがって、一旦パケットが読み込まれたなら、パケットの仮想及び物理アドレス、ならびに次のパケットの仮想及び物理アドレスは知られることになる。第１パケットが、次のパケットの物理アドレス及び仮想アドレスを収容し、リンク・リスト(linked list)を形成する。一旦リンクが確立されたなら、仮想アドレス上又は物理アドレス上のいずれかにおいて、それ以上のアドレス変換を必要とすることなく、パケット・プールを処理することができる。仮想及び物理アドレス・フィールドは、ヘッダの予め規定された位置にあるので、インデックス機能(indexing)を用いて必要なアドレスを読み出すことができる。

【０００５】
パケットの移動を容易に行なうために、分散型バースト・エンジン（ＤＢＥ：distributed burst engine）を開示する。これは、１つ以上の入出力（Ｉ／Ｏ）デバイスと主メモリとを結合するものである。分散型バースト・エンジンは、単一のＩ／Ｏデバイスのバス・インターフェース、ブリッジ・デバイス、又はメモリ・コントローラのいずれにも位置することができる汎用性のある構成物であり、これによって多数のＩ／Ｏデバイスに結合する。
分散型バースト・エンジン・アーキテクチャは、キュー・スキームを定義し、プロセッサにパケットの物理アドレスをデバイス・レジスタに書き込ませることなく、プロセッサが連続的にパケットをＩ／Ｏデバイスに整列(キュー：queue)することを可能にする。パケットが互いにリンクされているので、ＤＢＥは常に、次のパケットがどこで得られるかについて知っている。これにより、ＤＢＥがプロセッサの介入を受けずに、メモリからのデータをバーストすることを可能にする。
【０００６】
ＤＢＥは、フロント・エンドとバック・エンドとの間に、パケットを保持するための３つのデータ・バッファ、即ち、先入れ先出しメモリ（ＦＩＦＯ）を含む。それらは、要求バッファ、ポステッド・バッファ(posted buffer)、及び完了バッファである。ＤＢＥのフロント・エンドは、全体的に、メモリからパケットを引き出し、要求バッファ及びポステッド・バッファを満杯に保持しつつ、パケットをメモリに押し出して、完了バッファを空の状態に保つ役割を担っている。ＤＢＥのバック・エンドは、１つ以上のＩ／Ｏデバイスと双方向処理を行い、パケットからのコマンド、制御、アドレス、及びデータ情報をＩ／Ｏデバイスに提示する。
リンク・リストにおける加入及び削除に対するパケットの移動を容易に行なうために、ＤＢＥ及びプロセッサによって、ヘッド・レジスタ及びテール・レジスタが保持されている。また、ＤＢＥは、新たな要求パケットが要求キューにリンクされたというような、イベントが発生したことの指示をプロセッサから受け取るためのドアベル・レジスタ(doorbell register)も含む。例えば、プロセッサが要求パケットを要求キューに整列した後、プロセッサは要求ドアベルを鳴らす。パケットはリンクされており、ＤＢＥは次のパケットの位置を保持しているので、ＤＢＥは、これに応答して、単にメモリからパケットを引き出し、それをＩ／Ｏデバイスに提示する。コマンド又はデータは、パケット内に収容されている。
【０００７】
単一のＩ／Ｏデバイスが多数の機能を提供することができるので、多数の要求チャネルが利用可能となり、各要求チャネル毎に、要求バッファ、要求ドアベル、ならびに対応するヘッド・レジスタ及びテール・レジスタを有する。
パケットは、当該パケットに関する情報を搬送するためのヘッダを含む。パケットは、非同期、割り込み又はポール・パケット(polled packet)として指定することができる。非同期パケットは、当該パケットが完了したことの通知を必要としない要求に有用である。割り込みパケットは、完了の通知を必要とする要求パケットに有用である。非請求（unsolicited）パケットも、割り込みパケットとして指定される。割り込みパケットが完了した場合、ＤＢＥによってプロセッサに割り込みを発生する。ポール・パケットは、完了の通知を必要とするが、ハードウエア割り込みの形態ではない通知でよい要求に有用である。この形式のパケットは、ハードウエアの割り込みに伴うオーバーヘッドを回避するのに役立つ。
パケット・ヘッダ内に、ポール・パケットが完了したか否かを示すビットが１つある。したがって、プロセッサは、連続的にこのアドレス上で読み出し即ち「スピン(spin)」を行い、通知を受け取らなければならない。プロセッサがパケットをポールしている間メモリ・バス帯域を占領することを防止するために、パケット・ヘッダは、都合よくプロセッサのキャッシュライン内に位置するように設計されている。したがって、プロセッサは、ＤＢＥが前述のビットを変化させたときに、キャッシュ・ミスが発生するまで、そのキャッ

シュ上でポールする。

【０００８】

【発明の実施の形態】

図１は、本発明の好適な実施形態によるコンピュータ・システムＣを示している。コンピュータ・システムＣは、インテル社のペンティウム・プロ・プロセッサ(Intel Pentium Pro processor)等のような、１つ以上のプロセッサ１００を含む。１つのプロセッサ１００のみを示すが、本発明は単一プロセッサ・コンピュータに限定される訳ではない。プロセッサ１００は、ホスト・バス１０２に結合されている。

ホスト・バス１０２には、メモリ・サブシステム１０４、及びIntel 82454KX等のような、ホスト－ＰＣＩ間のブリッジ・デバイス１０６が結合されている。ホスト－ＰＣＩブリッジ・デバイスは、プロセッサ・サイクルをＰＣＩサイクルに、及びその逆に変換し、ＰＣＩバス１１４に接続する。メモリ・サブシステム１０４は、それ自体、 Intel 82453KX等のようなメモリ・コントローラ１０８、 Intel 82452KX等のようなメモリ・データ・パス・デバイス、及び主メモリ・アレイ１１２からなる。メモリ・コントローラ１０８は、主メモリ１１２にアドレス信号及び制御信号を供給する。主メモリ１１２は、複数のダイナミック・ランダム・アクセス・メモリ（ＤＲＡＭ）デバイス（具体的には示さない）で構成されている。メモリ・データ・パス・デバイス１１０は、メモリ・コントローラ１０８と共に動作し、ホスト・バス１０２のデータ部分と複数のＤＲＡＭとの間のバッファ動作(buffering)を行なう。勿論、他の公知の様々なメモリ・サブシステムを利用することも可能である。

【０００９】

また、ＰＣＩバス１１４には、ＰＣＩ／ＩＳＡ（業界標準アーキテクチャ）ブリッジ１１６、小型コンピュータ・システム・インターフェース（ＳＣＳＩ）コントローラ１２０のような１つ以上のＰＣＩバス・マスタ１１８、及びネットワーク・インターフェース・コントローラ１２８も接続されている。更に、ＰＣＩバス１１４には、ビデオ・システム１２２及び１つ以上のＰＣＩスロット１２４が接続されている。ＰＣＩ／ＩＳＡブリッジ１１６は、ＰＣＩサイクルをＩＳＡバス・サイクルに、及びその逆に変換し、ＩＳＡバス１２６に接続する。また、ＰＣＩ／ＩＳＡブリッジ１１６は、強化したダイレクト・メモリ・アクセス（ＤＭＡ）コントローラ、割り込みコントローラ、タイマ／カウンタ、及びノンマスカブル割り込みロジック(non-maskable interrupt logic)ならびに種々の周辺デバイス用デコード・ロジックを統合する。ＳＣＳＩコントローラ１２０は、ハード・ディスク１４４、テープ・ドライブ、及びＣＤ－ＲＯＭのようなＳＣＳＩ周辺機器に接続可能である。ビデオ・システム１２２は、ビデオ・コントローラ、フレーム・バッファ、及びモニタ１３０に接続するための種々のロジックを含む。ネットワーク・インターフェース・コントローラ１２８は、イーサネット・ネットワークに接続するための種々のインターフェース回路、又は代わりにトークン・リング・ネットワークを含む。ＳＣＳＩコントローラ１２０及びネットワーク・コントローラ１２８は、バス・マスタの多くの例の内の２つに過ぎず、これらを総称してＰＣＩバス・マスタ１１８と呼ぶことにする。尚、ここに開示する原理は、ＰＣＩバス１１４以外にも、ＩＳＡ又はＥＩＳＡのようなバス・マスタリング(bus mastering)に対応するその他のバスにも適用可能である。

ＩＳＡバス１２６は、更に、マルチＩ／Ｏデバイス１３２、リード・オンリ・メモリ（ＲＯＭ）１３４、及びキーボード・コントローラ１３６にも接続する。マルチＩ／Ｏデバイスは、フロッピ・ディスク・ドライブ１３８に接続するためのフロッピ・ディスク・コントローラを含む。また、マルチＩ／Ｏデバイス内には、多数のシリアル・ポート回路及びパラレル・ポート回路が収容されている。ＲＯＭは、電力投入時にハードウエアを起動するための低レベル・コードを与える。キーボード・コントローラは、キーボード１４０及びマウス１４２との通信を扱う。

【００１０】

システム・アーキテクチャ

次に、図２のＡのブロック図を参照する。本発明で特に興味深いのは、プロセッサ１００

がＰＣＩバス・マスタ１１８と、及びその逆方向に、通信する方法である。各ＰＣＩバス・マスタ１１８毎に、主メモリ１１２において初期化時に、請求パケット・プール(solicited packet pool)２００及び非請求パケット・プール(unsolicited packet pool)２０２が形成される。パケット・プールを収容するメモリ１１２のエリアは、プロセッサ１００とＰＣＩバス・マスタ１１８との間で共有される。請求パケット・プール２００は、１つ以上のパケット２５０を収容し、これらは、プロセッサ１００によって、目標のＰＣＩバス・マスタ１１８へ／からのデータを請求する際に用いられる。非請求パケット・プール２０２は、２つ以上のパケット２５０を収容し、これらは、ＰＣＩバス・マスタ１１８によって、非請求データ(unsolicited data)をプロセッサ１００に伝達する際に用いられる。非請求データとは、マウス１４２又はネットワーク・コントローラ１２８によって受け取られたデータのように、ＰＣＩバス・マスタ１１８によって、非同期データ源から自発的に受け取られたデータのことである。

パケット・プールを構成するパケット２５０は、初期化時にリンクされ、リストを形成する。各リストは、ヘッドＨポインタとテールＴポインタとを有し、これらに他のパケット２５０をリンクすることができる。請求パケット・プール２００内に位置するパケット２５０は、好ましくは同一の長さで、共に仮想アドレスによってリンクされている。非請求パケット・プール２０２内に位置するパケット２５０は、好ましくは同一の長さで、共に物理アドレスによってリンクされている。

【００１１】

メモリ１１２とＰＣＩバス・マスタ１１８との間のパケットの移動を容易に行なうために、各ＰＣＩバス・マスタ１１８は、分散型バースト・エンジン（ＤＢＥ）２０６を含むことが好ましい。ＤＢＥ２０６は、本質的に、Ｉ／Ｏデバイス２０８のためのインテリジェント・ダイレクト・メモリ・アクセス（ＤＭＡ）コントローラである。ＤＢＥの機能は、プロセッサ１００からの介入なく、パケット２５０を読み出し、送ることである。Ｉ／Ｏデバイス２０８は、本質的に、ＳＣＳＩコントローラのように、ＤＢＥ２０６と通信するように構成された従来からのＩ／Ｏコントローラである。

各ＤＢＥ２０６は、１つ以上のドアベル・レジスタ(doorbell register)２０４を含む。ドアベル２０４は、プロセッサ１００が「鳴らして(ring)」、メモリ１１２において１つ以上のパケット２５０の準備が完了しており、使用可能であることを、目標Ｉ／Ｏデバイス２０８に通知する。その後、Ｉ／Ｏデバイス２０８は、ＤＢＥ２０６を介して、パケットをメモリ１１２から引き出し、即ち、読み出し、パケット２５０の内容に応じてそれらを処理する。

【００１２】

Ｉ／Ｏデバイス２０８が外部ソースからデータを受け取った場合、ＤＢＥ２０６によって非請求パケットが準備され、メモリ１１２内に格納される。ＤＢＥ２０６が未請求データ・パケットをメモリ１１２内に格納した後、プロセッサ１００に割り込みをかけ、パケットが処理可能であることをプロセッサ１００に通知する。

１つ以上の論理通信チャネル２１４が、プロセッサ１００とＩ／Ｏデバイス２０８との間に存在する可能性がある。例えば、Ｉ／Ｏデバイス２０８は、入来データ専用に１つのチャネル２１４、出立データに別のチャネル２１４、ならびにコマンド及び制御のために第３のチャネル２１４を有することができる。

したがって、プロセッサ１００とＰＣＩバス・マスタ１１８との間の直接通信は、このコマンド・パケット・アーキテクチャによって減少即ち制限される。このプロセッサ１００及びＰＣＩバス・マスタ１１８間の、コマンド・パケット・アーキテクチャによる切り離しの結果、多数の利点が得られる。その利点の中には、ハードウエア割り込みの減少、グラフィックスやオーディオのようなプロセッサ１００の集約的アプリケーション(intensive application)に対する処理能力向上、及びプロセッサ１００の目標Ｉ／Ｏデバイス２０８から読み出し不要が含まれる。

【００１３】

次に図２のＢを参照すると、本発明のデバイス・ドライバがどのようにしてメモリ１１２

と通信するかについて示されている。従来技術では、プロセッサ１００は、デバイス・ドライバを介してＩ／Ｏデバイスと直接通信し、デバイス・ドライバが、Ｉ／Ｏデバイス上のメモリ・マップＩ／Ｏレジスタ(memory mapped I/O register)のコンフィギュレーション（環境設定）を行い、Ｉ／Ｏデバイスに処理を行なわせていた。その処理が完了した後、Ｉ／Ｏデバイスはプロセッサ１００に割り込みをかけ、一方プロセッサ１００は、割り込み源に関して、Ｉ／Ｏデバイス上のステータス・レジスタをプロセッサ１００にチェックさせる。

本発明の好適な実施形態によれば、要求は、ホスト・ソフトウエア・アプリケーション２２０から、Windows NTのようなオペレーティング・システム（ＯＳ）２２２に発生する。ＯＳ２２２は、全体として、ある実行サービス２２４、マイクロカーネル２２６，ハードウエア抽象レイヤ（ＨＡＬ: hardware abstraction layer）２２８、及び１つ以上のデバイス・ドライバ２３０を有するＩ／Ｏシステムを含むカーネルで構成されている。デバイス・ドライバ２３０は、メモリ１１２やＰＣＩバス・マスタ１１８のようなハードウエアＣと直接通信することを許可されている。各ＰＣＩバス・マスタ１１８毎に、少なくとも１つのデバイス・ドライバ２３０があることは理解されよう。ＤＢＥデバイス２０６に対するデバイス・ドライバ２３０は、いずれかのプロセッサ１００の特権レベル（インテル社のプロセッサ上における「リング」レベル）で実行するためには不要である。ドライバ２３０は、カーネル・モード又はユーザ・モードで実行することができる。

【００１４】

デバイス・ドライバ２３０とは、ここでは、Windows, Windows NT, OS/2等のようなオペレーティング・システムに関係する技術分野では公知のデバイス・ドライバのクラスを意味する。しかしながら、本発明にしたがって記載されるデバイス・ドライバ２３０は、当技術分野では新規なものであることは理解されよう。ここの開示は、本発明にしたがってデバイス・ドライバを開発するための十分な手引きを当業者に提供する。

デバイス・ドライバ２３０は、ＤＢＥデバイス２０６及びホスト・ソフトウエア２２０との双方向処理に関して、６つの主要な機能を有するものと見ることができる。これらの機能は、初期化／初期化解除(initialization/deinitialization)、パケット提出(packet submission)、パケット完了(packet completion)、同期ダイレクト・アクセス、パケット取り消し、パケット・プールの増減である。

ドライバ２３０は、システムの初期化中に、ＤＢＥデバイス２０６及びＩ／Ｏデバイス２０８を初期化しなければならない。一旦ＤＢＥデバイス２０６及びＩ／Ｏデバイス２０８が初期化され、要求を受け入れ可能となったなら、ドライバ２３０は、デバイス要求を開始することができ、ＤＢＥデバイス２０６はデータ転送を開始することができる。次いで、デバイス・ドライバ２３０は、パケットに対して適切な完了シーケンスを実行することができる。これらの詳細については、以下で論ずる。

尚、デバイス・ドライバ２３０はプロセッサ１００上で実行することが好ましいので、オペレーティング・システム２２２によって実行されるパケット処理に言及する場合、用語は相互交換可能に用いることができることを注記しておく。加えて、「ホスト」という用語は、ときとして、プロセッサ１００、ホスト・ソフトウエア２２０、及びデバイス・ドライバ２３０全体に論理的に言及するために用いることもある。このような場合、番号指定はプロセッサ１００と同一とする。即ち、ホスト１００とする。

【００１５】

パケット

ＤＢＥ２０６の更なる詳細に進む前に、パケット２５０について更に詳しく理解する必要がある。ここで図３を参照すると、好適な実施形態によるパケット２５０が示されている。パケット２５０は、２つの主要部分、即ち、３２バイトのヘッダ２５２及び可変サイズのペイロード２５４からなる。各パケットは、６４バイトの境界、即ち、Pentiumのキャッシュラインに整合されている。このようにすれば、最初のキャッシュラインがヘッダ２５２を収容し、後続のキャッシュラインはペイロード２５４を収容することができる。ペイロード２５４は、デバイスに特定のデータを保持し、４バイト刻みで４～２５６バイト

の範囲を取ることができる。
ヘッダ２５２は、８つの３２ビット・フィールドで構成され、ソフトウエア・コンテクスト(SwContext)フィールド２５６、チャネル・フィールド２５８、仮想アドレス・リンク(VaL ink)フィールド２６０、物理アドレス(Pa)フィールド２６２、ハードウエア・コンテクスト(HwContext)フィールド２６４、物理アドレス・リンクDword (PaLinkDwords)フィールド２６６、物理アドレス・リンク(PaLink)フィールド２６８、及び仮想アドレス(Va)フィールド２７０を含む。リンク・フィールドは、パケットのリンク・リスト形成を可能にする。
【００１６】
SwContextフィールド２５６は、ホスト・ソフトウエア２２０が、適切と見なすいずれかの方法で使用するために予約されている３２ビット・フィールドである。ソフトウエアは、このフィールドを用いて、このパケット２５０に関連するＯＳに特定の情報を埋め込むことができる。その一例をあげるとすれば、パケット２５０内のＩ／Ｏ要求と関連するＩ／Ｏ要求パケット（ＩＲＰ）の仮想アドレスを置くことであろう。これによって、デバイス・ドライバ２３０は、Ｉ／Ｏ完了の間に、Ｉ／Ｏ要求パケットを直接復元することができ、未決要求のリストを管理する必要がなくなる。
チャネル・フィールド２５８は、パケットが関連した最後のチャネル２１４の番号を収容する３２ビット・フィールドである。チャネル２１４は、論理通信リンクである。各Ｉ／Ｏデバイス２０８は、２つ以上のチャネル２１４を有する場合がある。一方、１つ以上のＩ／Ｏデバイスに対応するＤＢＥ２０６は、２つ以上のチャネル２１４を有する場合がある。例えば、１つのチャネル２１４をデータ転送のために割り当て、他方のチャネル２１４をコマンド及びステータスのために割り当てる場合もある。このフィールドは、非請求パケット・プール２０２内のパケットに対しては、「０」を収容する。
【００１７】
Vaフィールド２７０、VaLin kフィールド２６０、Paフィールド２６２、及びPaLinkフィールド２６８は、パケット・ヘッダ２５２内に収容され、物理メモリ及び仮想メモリ間の多数のアドレス変換を不要とする。物理メモリはメモリ１１２である。Pentium及び同等のプロセッサ１００は、物理メモリ１１２を効率的に使用するために、ハードウエア・メモリ管理機構を採用している。典型的に、プロセッサ１００上で実行するプログラムは直接物理メモリ１１２にアドレス指定するのではなく、代わりに、仮想アドレスを用いてメモリ１１２にアクセスする。各パケット２５０内に仮想アドレス・フィールド及び物理アドレス・フィールドの双方を備えることにより、パケット２５０を初期化するときに、変換を１回だけ行えば済むようになる。このように、デバイス・ドライバは仮想アドレスを用いて動作することができ、一方ＤＢＥ２０６は物理アドレスを用いて動作することができ、各々同じパケット２５０を用いて通信する。
【００１８】
Vaフィールド２７０は、以下のフィールドで構成された３２ビット・フィールドである。
【表１】

表　1

ビット	説明
31:6	デバイス・ドライバが見る場合のこのパケットに対する仮想アドレスを収容する。
5	予約
4	０－プロセス・パケット １－キャンセル・パケット 取り消しビット。パケットの処理状態を表す。デバイス・ドライバはこのビットをセットして、パケット２５０を処理せず、直ちに完了しなければならないことを、ＤＢＥデバイス２０６に示す。
3	０－通常の提出 １－ダイレクト・アクセスの開始 ダイレクト・アクセス・ビット。ドライバは、このパケットの完了後ダイレクト・アクセスを行うことを、ＤＢＥデバイス２０６に示す。受け取られた場合、ＤＢＥ２０６は、ダイレクト・アクセスが終了するまで、それ以上のデータを引き出さない。「１」の場合、このパケットを直ちに完了する。
2:1	００－ヌル提出状態 ０１－非同期パケット １０－ポール・パケット １１－割り込みパケット 提出型ビット。このパケットがどのようにＤＢＥデバイスに提出されたかを表す値を収容する。未請求ポール内のパケットは、このフィールドには全て０を有する。
0	０－未請求パケット・プール １－請求パケット・プール パケット型ビット。どのプールにパケットが属するかを示す。

【００１９】

VaLinkフィールドは、リスト内の次のパケットの仮想アドレスを収容する３２ビット・フィールドである。このフィールドは、リンクされているパケットがない場合、ヌルとなる。このフィールドは、ＤＢＥデバイス２０６が完了したときに、空きパケットを再び請求パケット・プール２００にリンクする際にＤＢＥデバイス２０６によって用いられる。空きパケット(free packet)とは、既に完了し、現在は再び「自由に」使用できるパケットのことである。ＤＢＥ２０６デバイスは必ずしも仮想アドレス処理を理解している訳ではないが、単純にパケット２５０内に収容されている仮想アドレスのリード及びライトを行うことによって、仮想アドレスを用いてパケット２５０をリンクすることができる。

Paフィールド２６２は、パケット２５０の物理アドレスを収容する３２ビット・フィールドである。これは、ＤＢＥデバイス２０６が見て使用するアドレスである。

PaLinkフィールド２６８は、リスト内の次のパケット２５０の物理アドレスを収容する３２ビット・フィールドである。

HwContextフィールド２６４は、ＤＢＥデバイス２０６が適切とみなすいずれかの方法で、その特定の実装によって用いるために予約されている３２ビット・フィールドである。例えば、これを用いて、キャッシュされていないメモリ範囲へのアドレスを埋め込むことにより、ペイロード２５４なしでパケット２５０を作成することが可能となる。

PaLinkDwordsフィールド２６２は、次のパケットのペイロード２５４のサイズを収容する３２ビット・フィールドである。そのサイズは、連続する４バイトのチャンク(chunk)の数として表現することが好ましい。

【００２０】
このように、パケット構造及びプロトコルが定義され、これによって、デバイス・ドライバ２３０は、Ｉ／Ｏデバイス２０８とパケットを通信することが可能となる。デバイスの初期化時に、プロセッサ１００は、Ｉ／Ｏデバイス２０８に対して、特定数のパケットを形成する。次に、デバイス・ドライバ２３０は、 Ｉ／Ｏデバイス２０８と通信する際、メモリ１１２に格納されているパケット２５０にコマンド及びデータを書き込み、パケット２５０を処理する準備ができていることをＤＢＥ２０６に通知する。ＤＢＥ２０６は、メモリ１１２からパケット２５０を読み出し、それを解析してドライバ２３０が要求した処理を判定する。どのようにパケット２５０をＤＢＥ２０６に発行したかに応じて、ＤＢＥデバイスがどのように処理を完了し、ドライバ２３０のためにパケット２５０をメモリ１１２に戻すのかが決定される。
パケット２５０は、Vaフィールド２７０に示される３つの方法、即ち、非同期、ポール、又は割り込みの内の１つで提出することができる。デバイス・ドライバ２３０は、完了通知を必要としない又は望まない場合に、非同期パケットを提出する。グラフィックス・コントローラ１２２のビット・ブリット（ＢＬＴ：bit-blit)処理が、その一例である。
【００２１】
デバイス・ドライバ２３０は、完了通知を必要とする場合、Ｉ／Ｏデバイス１１８によってアサートされたハードウエア割り込みによって、割り込みパケットを提出する。
デバイス・ドライバ２３０は、ホスト・ソフトウエア２２０がポーリングを要求するような状況に対して、ポール・パケットを提出する。場合によって、ポール・パケットの方が効率的な手段完了通知(means completion notification)となることもある。例えば、ＳＣＳＩコントローラ１２０に対する典型的な要求は、「セクタ・リード」及び「セクタ・ライト」動作である。従来技術のオペレーティング・システムは、その殆どが、デバイス・ドライバに要求を完了するのを「待つ」ように要求するセクタ・データのリード及びライトを行うためのコール側アプリケーションの目標バッファを表す、物理ページのリストを備えている。要求の完了を示すディスク・デバイスによる通知の際に、ドライバは、コール側のアプリケーションの目標バッファのために、メモリを解放することができる。目標デバイスに基づいて、ドライバにとっては、ハードウエア割り込みの結果としてコンテクストの切り替えを誘発するよりはむしろ、特定の処理上で「ポール」することによって、完了通知を「待つ」方が一層効率的な場合がある。したがって、本発明は、デバイス・ドライバ２３０が、ＤＢＥデバイス２０６に要求を提出し、Ｉ／Ｏデバイス２０８がその要求を終了するまで、パケット２５０上でポールするための方法を提供する。
【００２２】
パケット・キュー
初期化時に、デバイス・ドライバ２３０は、請求パケット・プール２００及び非請求パケット・プール２０２を形成する。図４には、これらのパケットのリンク・リストの形成に関して更に詳細に示されている。デバイス・ドライバ２３０及びメモリ１１２間ならびにＤＢＥデバイス２０６及びメモリ１１２間双方におけるパケットのフローを管理するために、４つのキュー２７２～２７８が形成される。即ち、フリー・キュー（ＦＱ: free queue）２７２、要求キュー（ＲＱ: request queue）２７４、完了キュー（ＣＱ: completion queue）２７６、及びポステッド・キュー（ＰＱ: posted queue）２７８である。フリー・キュー２７２は、請求パケット・プール２００と同一であり、ポステッド・キュー２７８は非請求パケット・プール２０２と同一である。これら４つのキュー２７２～２７８の各々は、ヘッドＨポインタ及びテールＴポインタを有する。
デバイス・ドライバ２３０は、フリー・キュー２７２のヘッドH及び残りの３つのキュー２７４～２７８のテールＴを保持する。ＤＢＥデバイス２０６は、フリー・キュー２７２のテールT及び残りの３つのキュー２７４～２７８のヘッドＨを保持する。
【００２３】
フリー・キュー２７２は、デバイス・ドライバ２３０がＤＢＥデバイス２０６との通信に使用する際に利用可能なパケット２５０を収容する。ホスト・ソフトウエア２２０から

要求が出されると、デバイス・ドライバ２３０は、フリー・キュー２７２のヘッドＨからパケット２５０のリンクを解除し、パケットのペイロード２５４を満たし、パケット２５０を要求キュー２７２のテールＴにリンクする。パケット２５０は、非同期的（Ａ）、ポール（Ｐ）又は割り込み（Ｉ）のいずれかによって提出することができる。

ＤＢＥデバイス２０６は、受け取った要求を処理する際に、パケット２５０のリンクを要求キュー２７４から解除し、ペイロード２５４をＩ／Ｏデバイス２０８に受け渡し、それらの提出方法にしたがってパケット２５０を完了する。更に具体的には、非同期パケットＡの完了は、デバイス・ドライバ２３０に通知せずに行い、フリー・キュー２７２のテールＴで完了する。ポール・パケットＰは、デバイス・ドライバ２３０によって、フリー・キュー２７２のヘッドＨに戻され、割り込みパケットＩの完了は、完了キュー２７６を通じて行う。

【００２４】

割り込みパケットについては、ＤＢＥデバイス２０６が割り込みパケットＩを完了キューのヘッドＨにリンクした後、ハードウエア割り込みを発生すればよい。完了キュー２７６が空の場合、パケット２５０をリンクするときに割り込みを発生する。完了キュー２７６が空でない場合、続いてリンクされるパケットは割り込みを発生しない。デバイス・ドライバ２３０は、完了キュー２７６からのパケット２５０のリンクを解除し、ペイロード２５４を除去し、Vaフィールド２７０のビット０に基づいて、パケットをフリー・キュー２７２又はポステッド・キュー２７８のいずれかに戻す。

ポステッド・キュー２７８は、ホスト側の請求に基づくことなくホスト側にデータを転送することを要求する非請求要求のためにパケット２５０を保持する。非請求要求が発生した場合、ＤＢＥデバイス２０６は、ポステッド・キュー２７８からのパケット２５０のリンクを解除し、ペイロード２５４をデータで満たし、パケット２５０を完了キュー２７６にリンクする。パケットのリストが未だ完了キュー２７６上にない場合、ハードウエア割り込みを発生する。

【００２５】

ＤＢＥに対するソフトウエア・インターフェース

ＤＢＥデバイス２０６は、ＤＢＥデバイス・オブジェクト２７２（図２のＡ）を通じて、デバイス・ドライバ２３０に対して抽象化される。ＤＢＥデバイス・オブジェクト２７２は、関連するデータ構造、及びドライバ２３０とＤＢＥデバイス２０６との間でパケット・プロトコルを管理するために必要なリソースを収容する。ＤＢＥデバイス・オブジェクト２７２は、メモリ１１２に格納され、ＤＢＥデバイス・ドライバ２３０によって管理される。

ＤＢＥデバイス・オブジェクト２７２は、コンピュータ・システムＣ内の特定のＤＢＥデバイス２０６に対応する。ＤＢＥデバイス２０６は、このデータ構造の概念を有していない。オブジェクト２７２は、目標ＤＢＥデバイス２０６のリソースを管理するために、デバイス・ドライバ２３０のみによって作成され使用される。ＤＢＥデバイス・オブジェクトのメモリ・フォーマットを変更しても、新たなファンクショナリティがデバイスによって要求されなければ、デバイスには影響を与えない。ＤＢＥオブジェクト２７２をPentiumのキャッシュ・ラインの境界と整合させ、頻繁に参照される隣接するデータ項目上でキャッシュ・ラインが満杯になる回数を減少させる。ＤＢＥデバイス・オブジェクト２７２は、表２に示したデータ構造からなる。

【００２６】

【表２】

表　2

パラメータ	説　明
DeviceID	このＤＢＥオブジェクトの一意のデバイスＩＤとして使用される３２ビット・フィールド。
FreeQueue	ヘッド・パケットに対するアドレスを含む、フリー・キューを管理するためのフィールドを収容する３２ビット・メモリ位置。
CompletionQueue	テール・パケットに対するアドレスを含む、完了キューを管理するためのフィールドを収容する３２ビット・メモリ位置。
RequestQueue	テール・パケットに対するアドレスを含む、要求キューを管理するためのフィールドを収容する３２ビット・メモリ位置。
PostedQueue	ポステッド・パケットに対するアドレスを含む、フリー・キューを管理するためのフィールドを収容する３２ビット・メモリ位置。
RegistersVa	ＤＢＥデバイス２０６のレジスタの仮想アドレスを保持するためのフィールドを収容する３２ビット・メモリ位置。
RegisterPa	ＤＢＥデバイス２０６のレジスタの物理アドレスを保持するためのフィールドを収容する３２ビット・メモリ位置。
MmioAddressRange	ＤＢＥデバイスのレジスタをマップするために必要なバイトの範囲を収容する３２ビット・メモリ位置。
RequestPacketCount	請求パケット・プールに割り当てられるパケット数を収容する３２ビット・メモリ位置。
RequestPacketSize	請求パケット・プールの各パケット毎に DeviceContext フィールドの Dword サイズを収容する３２ビット・メモリ位置。
PostedPacketCount	非請求パケット・プールに割り当てられるパケット数を収容する３２ビット・メモリ位置。
PostedPacketSize	非請求パケット・プールの各パケット毎に DeviceContext フィールドの Dword サイズを収容する３２ビット・メモリ位置。
DbeFlink	この１つ前のＤＢＥデバイス・オブジェクトへのポインタを収容する３２ビット・メモリ位置。
Dbeblink	この１つ後ろのＤＢＥデバイス・オブジェクトへのポインタを収容する３２ビット・メモリ位置。
DummyPacket	このフィールドは、デバイス・ドライバが、要求キューのテール上にあるパケットがそれ自体を参照しないことを保証するするために用いられるダミー・コマンド・パケットを示す。この状態は、最後に提出したパケットが、別のパケットを割り当ててデバイスに提出する前に、フリー・キューに戻されてきた場合に発生する可能性がある。 DBE_AllocatePacket ルーチンが、パケット・アドレスをコール元に戻す前に、この状態をチェックする。フリー・キューから除去されたパケットが偶然ＤＢＥデバイスに提出された最後のパケットであった場合、DBE_AllocatePacket ルーチンは、除去されたパケット・アドレスを DummyPacket のアドレスと交換する。

【００２７】

ＤＢＥデバイス・インターフェース

次にＤＢＥデバイス２０６に言及するために、図５に注意を向けると、ＤＢＥデバイス２０６のレジスタ・インターフェースが示されている。ＤＢＥデバイス２０６は、１組のメモリ・マップＩ／Ｏレジスタ２８０～２９２を含み、ＤＢＥプロトコルに便宜を図っている。ＤＢＥドライバ２０６に対応するデバイス・ドライバ２３０は、システムの初期化中に、コンフィギュレーション情報をレジスタ２８０～２９２に書き込む。レジスタ２８０～２９２は、プロセッサ１００（デバイス・ドライバ）及びＤＢＥデバイス２０６双方にアクセス可能である。これらのレジスタは、イベント・イネーブル・レジスタ(EN_REG)２８０、イベント・ディスエーブル・レジスタ(D I_REG)２８２、フリー・キュー・テール

・レジスタ(FQ_REG)２８４、完了キュー・ヘッド・レジスタ(CQ_REG)２８６、ポステッド・バッファ・キュー・ヘッド・レジスタ(PQ_REG)２８８、及び１つ以上の要求キュー・ヘッド・レジスタ(RQ_n_REG) ２９０を含む。
イベント・イネーブル・レジスタ(EN_REG)２８０は、実装に特定の情報を収容するための３２ビット・レジスタである。ビット３１：２は、カスタム化(customization)のために使用可能である。ビット１：０については、以下で定義する。このレジスタは、３２ビット、リード／ライト、ビット・マップ、及びメモリ・マップＩ／Ｏレジスタであることの属性を有する。ビットの定義は以下の通りである。
【００２８】
【表３】

表　３

EN_REG		
ビット	プロセッサ・サイクル	説明
0	W	0－このビットに影響なし 1－ＤＢＥデバイスをRESETに置く
0	R	0－ＤＢＥデバイスはRESETにない 1－ＤＢＥデバイスはRESETにある
1	W	0－このビットに影響なし 1－ホスト・プロセッサへのＤＢＥデバイス割り込みをイネーブルする
1	R	0－ＤＢＥデバイス割り込みはイネーブルされていない 1－ＤＢＥデバイス割り込みはイネーブルされている
31:2	R／W	これらのビットはカスタム使用のために定義される

【００２９】
イベント・ディスエーブル・レジスタ(DI_REG)２８２は、デバイス・ドライバ２３０が、EN_REG２８０によってイネーブルされたイベントをディスエーブルする際に用いられる。EN_REG２８０及びDI_REG２８２レジスタは、同一レジスタに対するセット／クリア動作に対して、個々のビット位置を自動的に反映させる(affect)ために通常必要なリード／修正／ライト動作を不要とするために実装する。このレジスタは、３２ビット、リード／ライト、ビット・マップ及びメモリ・マップＩ／Ｏレジスタであることの属性を有する。ビット定義は以下の通りである。
【表４】

表　4

DI_REG		
ビット	プロセッサ・サイクル	説明
0	W	0－このビットに影響なし。 1－ＤＢＥデバイスを RESET から取りだす。
0	R	0－ＤＢＥデバイスは未だ RESET にある。 1－ＤＢＥデバイスは RESET 外にある。
1	W	0－このビットに影響なし 1－ホスト・プロセッサへのＤＢＥデバイス割り込みをディスエーブルする。
1	R	0－ＤＢＥデバイス割り込みをイネーブルする。 1－ＤＢＥデバイス割り込みをディスエーブルする。
31:2	R／W	これらのビットはカスタム使用のために定義される。

【００３０】
フリー・キュー・テール・レジスタ(FQ_REG)２８４は、請求パケット・プール２００上の最後のパケット２５０、即ち、テール・パケットＴの物理アドレスを格納するための３２ビット・レジスタである。このレジスタは、システムの初期化中に、デバイス・ドライバ２３０によって初期化される。その後、このレジスタは、ＤＢＥデバイス２０６によって維持される。この手順を実装するために、ＤＢＥデバイス２０６は、ＤＢＥデバイス２０６がRESETモードにある場合(EN_REG[0]="1")、このレジスタのビット[31:5]へのプロセッサ１００のライトをラッチする。ＤＢＥデバイス２０６は、物理アドレスのビット[4:0]が"0"を含むことを想定する。
ＤＢＥデバイス２０６は、デバイス・ドライバ２３０に通知を返送することなく、請求パケット・プール２００のテールＴへの非同期パケット（以下で論ずる）を完了する。デバイス・ドライバ２３０は、ＤＢＥデバイス２０６が、完了した非同期パケット２５０にリンクするための場所を有するために、請求パケット・プール２００上に少なくとも１つのフリー・パケット２５０を保持しなければならない。このレジスタは、３２ビット、リード／ライト、ビット・マップ及びメモリ・マップＩ／Ｏレジスタであることの属性を有する。ビット定義は以下の通りである。
【表５】

表　5

FQ_REG		
ビット	プロセッサ・サイクル	説明
31:5	W	Pentium キャッシュラインに整合した請求パケット・プールのテール上のパケットの物理アドレス。ライトは、RESET モードにあるＤＢＥデバイスによってのみラッチされる。
4:0	W	ＤＢＥデバイスは常に無視する。
31:0	R	請求パケット・プール上の最後のパケットの物理アドレスを生成する。

【００３１】
完了キュー・レジスタ(CQ_REG)２８６は、メモリ１１２内に位置する完了リスト・ヘッドの物理アドレスを格納するための３２ビット・レジスタである。CQ_REG２８６は、ＤＢＥデバイス２０６によって構築された完了リストのヘッドを置く位置の４ビット整合物理アドレスを用いて、デバイス・ドライバ２３０によって初期化される。その後、CQ_REG２８６は、ＤＢＥデバイス２０６によって維持される。この手順を実装するために、ＤＢＥデバイス２０６は、ＤＢＥデバイス２０６がRESETモードにある場合(EN_REG[0]="1")、このレジスタのビット[31:2]へのプロセッサ１００の書き込みをラッチする。ＤＢＥデバイス２０６は、物理アドレスのビット[1:0]が"0"を含むことを想定する。
【表６】

表　6

CQ_REG		
ビット	プロセッサ・サイクル	説明
31:2	W	DWORD に整合した、完了キュー・ヘッドの物理アドレス。ライトは RESET モードにあるＤＢＥデバイスによってのみラッチされる。
1	W	０－ＤＢＥデバイスは常に無視する。
0	W	０－このビット位置には影響なし。１－ＤＢＥデバイスが RESET モード以外の場合にのみ、CQ ドアベルをセットする。
31:0	R	このレジスタの現内容を生成する。

【００３２】
ポステッド・キュー・レジスタ(PQ_REG)２８８は、非請求パケット・プール２０２内の最初のパケット２５０の３２ビット整合物理アドレスを格納するための３２ビット・レジスタである。PQ_REG２８８は、システムの初期化中に、デバイス・ドライバ２３０によって初期化される。その後、PQ_REG２８８は、ＤＢＥデバイス２０６によって維持される。この手順を実装するために、ＤＢＥデバイス２０６は、ＤＢＥデバイス２０６がRESETモードにある場合(EN_REG[0]="1")、このレジスタのビット[31:5]へのプロセッサ１００のライトをラッチする。ＤＢＥデバイス２０６は、物理アドレスのビット[4:0]が"0"を含むこ

とを想定する。ＤＢＥデバイスはこのレジスタを読み取り、非請求データをメモリ１１２のどこに書き込むかの決定を行う。

【表７】

図　7

PQ_REG

ビット	プロセッサ・サイクル	説明
31:5	W	Pentiumキャッシュラインに整合した非請求パケット・プール内の最初のパケットの物理アドレス。ライトは、RESETモードにあるＤＢＥデバイスによってのみラッチされる。
4:1	W	００００－ＤＢＥデバイスは常に無視する。
0	W	０－このビット位置には影響なし。１－ＤＢＥデバイスがRESETモード以外の場合にのみ、CQドアベルをセットする。
31:0	R	このレジスタの現内容を生成する。

【００３３】

要求キュー・ヘッド・レジスタ(RQ_n_REG)２９０は、次の要求パケット２５０がデバイス・ドライバ２３０によってどこに置かれたかの、３２ビット整合物理アドレスを格納するための３２ビット・レジスタである。各(RQ_n_REG)２９０は、システムの初期化中に、チャネルｎに対応する「ダミー」パケット２５０の物理アドレスを用いて、デバイス・ドライバ２３０によって初期化される。その後、 RQ_n_REG ２９０は、ＤＢＥデバイス２０６によって維持される。この手順を実装するために、ＤＢＥデバイス２０６は、ＤＢＥデバイス２０６がRESETモードにある場合(EN_REG[0]="1")、このレジスタのビット[31:5]へのプロセッサ１００のライトをラッチする。ＤＢＥデバイス２０６は、物理アドレスのビット[4:0]が"0"を含むことを想定する。ＤＢＥデバイス２０６は、このレジスタを用いてパケット２５０をフェッチする。

【表８】

表　8

RQn _REG		
ビット	プロセッサ・サイクル	説明
31:5	W	Pentium キャッシュラインに整合したチャネル$_n$に対する「ダミー」パケットの物理アドレス。ライトは、RESET モードにあるＤＢＥデバイスによってラッチされる。
31:5	R	ＤＢＥデバイスによって処理された最後の要求パケットの物理アドレスのビット[31:5]を生成する。
4:2	W	０００－ＤＢＥデバイスは常に無視する。
4:2	R	常に０００を返す。
1	W	０－このビットには影響なし。 １－ＤＢＥデバイスが RESET モード以外にある場合のみ、ＤＢＥデバイスにおけるチャネル$_n$に対する同期ダイレクト・アクセスを終了する。
0	R	０－ダイレクト・アクセスのためにチャネル$_n$を同期させる。 １－チャネル$_n$はパケット処理中である。
0	W	RQ_DOORBELL ０－このビット位置に影響なし。 １－デバイスが RESET モード以外にある場合のみ、RQ$_n$ドアベルをセットする。
0	R	RQ_DOORBELL ０－ＤＢＥデバイスによるドアベル・イベントのクリア。 １－ＤＢＥデバイスにおいて保留中のドアベル・イベント。

【００３４】

ソフトウエア・アーキテクチャ

ＤＢＥプロトコルの機構を隠すように、１組のルーチンがホスト・ソフトウエア２２０に使用可能である。各ルーチンは、ホスト・ソフトウエア２２０によってコールされるためのエントリポイントを含む。以下に各エントリポイントを説明する。

DBE_AcknowledgeInterrupt

このルーチンは、完了したパケット２５０のリストを返し、目標ＤＢＥデバイス２０６における割り込みを承認する。このルーチンは、デバイス・ドライバの割り込みサービス・ルーチン（ＩＲＳ：Interrupt Service Routine）又は遅延プロシージャ・コール（ＤＰＣ：Deferred Procdure Call）によってコールすることができる。このルーチンをコールして、ＤＢＥデバイス・オブジェクト・ハンドル内に位置する完了キュー・ヘッドから、完了リストを除去する。コール元は、同期を与える必要はない。

DBE_AllocatePacket

このルーチンは、請求パケット・プール２００のヘッドＨから最初のパケット２５０を原子的に除去する。このルーチンは、ドライバ２３０がある処理を目標ＤＢＥデバイス２０６に請求したい場合に、当該ドライバによってコールされる。ドライバ２３０は、パケット２５０を割り当て、指定された請求要求のためにそれを満たし、ＤＢＥ提出ルーチンの１つを用いてそれを目標ＤＢＥデバイス２０６に発行する。このルーチンは、ドライバ

２３０による同期を必要としない。
【００３５】
DBE_BeginDirectAccess
このルーチンは、ＤＢＥデバイス２０６に対するダイレクト・アクセスを、チャネル・パラメータによって指定された請求要求チャネルに以前に提出されたパケット２５０と同期させる。同期を取るには、ドライバ２３０が指定されたチャネル２１４に対するDBE_EndDirectAccessをコールするまで、指定されたチャネル２１４に対するＤＢＥデバイス２０６上での入力処理を停止する。通常、ドライバのダイレクト・アクセスは、ＤＢＥデバイス２３０において保留となっている請求パケットとの同期は必要としない。ドライバ２３０は、そのダイレクト・アクセスが、指定された請求要求チャネル２１４に以前に提出された未決のパケットの完了に依存する場合に、このルーチンをコールする。このルーチンは、他のスレッドが、指定された請求要求チャネル２１４に要求を提出するのを禁止する。
DBE_CancelPacket
このルーチンは、指定されたパケットに、取り消しのためのマークを付ける。このルーチンは、以前に提出したパケットが目標ＤＢＥデバイスによって処理されるべきでない場合に、ドライバによってコールされる。次に、ＤＢＥドライバ２０６は、パケットのペイロード２５４をバースト・ダウンする(burst down) ことなく、取り消されたパケットを完了する。パケットの完了は、パケット２５０がドライバ２３０によってどのように提出されたかによって判定される。ドライバ２３０は、ＩＯ取り消しプロトコルを実装するために、ＤＢＥデバイス２０６において保留となっているパケット２５０を追跡する役割を担う。このルーチンは、ＩＯ完了のためのマークを、指定されたパケットに付ける方法を与えるに過ぎない。
【００３６】
DBE_CompletionListIsReady
このルーチンは、完了リストのために、完了キュー・ヘッドを「のぞき見る」。このルーチンは、完了リスト・キューから完了リストを除去することはない。このルーチンは、ドライバ２３０が、ＤＢＥデバイス・オブジェクト２７２内の完了リスト・キュー上の完了リストをチェックするためにコールされる。ドライバ２３０が完了キューから完了リストを除去したい場合、DBE_AcknowledgeInterrutpルーチンをコールしなければならない。
DBE_DecreasePool
このルーチンは、指定されたパケット・プールからパケットを除去する。ＤＢＥプロトコルは、２つのパケット・プールを定義する。１つのプールは、ＤＢＥデバイスに対する請求要求のためのものであり、他方のプールは、ＤＢＥデバイスから受け取った非請求データのためのものである。
DBE_Deinitialize
このルーチンは、目標デバイスのためにDBE_Initializeによって作成された、全ての割り当てリソースを削除する。
DBE_FreePacket
このルーチンは、指定されたパケットを、請求パケット・プールのヘッド上に置く。
【００３７】
DBE_EndDirectAccess
このルーチンは、チャネル２１４によって指定された請求要求チャネル２１４に対して、ＤＢＥ状態機械（マシン）を起動する。
DBE_GetDevice
このルーチンは、システム内の全てのＤＢＥデバイスのマスタ・リストを走査し、指定されたデバイスＩＤに対する一致を調べる。このルーチンは、指定されたデバイスＩＤを有するＤＢＥデバイスを作成するのではない。代わりに、このルーチンは、DBE_Initializeルーチンによって既に作成されている、指定されたデバイスＩＤを有するＤＢＥデバイスを検索する。このルーチンは、マルチスレッド及びマルチプロセッサ・セーフ(multith

read and multiprocessor safe)である。

DBE_GetHwContext

このルーチンは、ＤＢＥパケット・ヘッダ内の、特定ハードウエア・データ・フィールドの３２ビットの内容を返す。

DBE_GetNextPacketToComplete

このルーチンは、完了したパケットのリストにおいて、次のパケットを読み出し、現パケットを適切なパケット・プール上に返す。このルーチンは、DBE_AcknowledgeInterruptルーチンによって既に除去されている、完了リスト上の次のパケットの仮想アドレスを返す。このルーチンは、マルチスレッド及びマルチプロセッサ・セーフである。DBE_AcknowledgeInterruptによって完了リストを除去した各実行コンテクスト(execution context)は、当該リストの独占所有権を有する。

【００３８】

DBE_GetSwContext

このルーチンは、ＤＢＥパケット・ヘッダ内の特定ソフトウエア・データ・フィールドの３２ビットの内容を返す。

DBE_IncreasePool

このルーチンは、パケットを、指定されたパケット・プールに追加する。ＤＢＥプロトコルは２つのパケット・プールを定義する。一方のプールは、ＤＢＥデバイスに対する請求要求のためのものであり、他方のプールは、ＤＢＥデバイスから受け取った非請求データのためのものである。

DBE_Initialize

このルーチンは、ＤＢＥデバイス・オブジェクトを作成する。このルーチンは、ドライバがＤＢＥデバイス・オブジェクトを作成し初期化するためにコールされる。このルーチンは、DeviceIDによって指定された同一デバイスＩＤを有する別のＤＢＥデバイスが既に存在する場合は、実行されない(succeed)。

DBE_PacketIsBusy

このルーチンは、指定されたパケットが目標ＤＢＥデバイスにおいて保留となっているか否かについて判定を行う。このルーチンは、パケット・ヘッダ内のVaLinkフィールド２６０のビット２の検査を行うのみである。何故なら、ドライバの完了及び承認を必要とするのは、ポール・パケット及び割り込みパケットだけであるからである。このルーチンは、非同期パケットとして提出されたパケット２５０にはNULL（ヌル）を返す。

【００３９】

DBE_SetHwContext

このルーチンは、ＤＢＥパケット・ヘッダ２５２内の特定ハードウエア・データ・フィールドの３２ビットの内容をセットする。

DBE_SetSwContext

このルーチンは、ＤＢＥパケット・ヘッダ２５２内の特定ソフトウエア・データ・フィールドの３２ビットの内容をセットする。

DBE_SubmitPacketA

このルーチンは、請求パケットを目標ＤＢＥデバイス２０６に発行する。このパケットは、ＤＢＥデバイス２０６が当該処理を完了したときに、ＤＢＥデバイス２０６によって自動的に請求パケット・プール２００のテールＴに再循環される。このルーチンは、プロセッサ割り込みをディスエーブルせず、目標請求要求キューに対するアクセスの同期を取るために、「スピン」ループを必要としない。

DBE_SubmitPacketI

このルーチンは、請求パケットを目標ＤＢＥデバイスに発行する。このパケットを発行して、目標ＤＢＥデバイスからのハードウエア割り込みに、このパケットの完了を承認するように要求する。このルーチンは、この要求を発行後直ちに、コール元に戻る。ドライバは、目標オペレーティング・システム環境によって定義された標準的な割り込み処理を通じて、このパケットの完了を承認しなければならない。このルーチンは、プロセッサ割り

込みをディスエーブルせず、目標請求要求キューに対するアクセスの同期を取るために、「スピン」ループを必要としない。このルーチンは、マルチスレッド及びマルチプロセッサ・セーフである。

【００４０】

DBE_SubmitPacketP

このルーチンは、請求パケットを目標ＤＢＥデバイスに発行する。このパケットは、目標ＤＢＥデバイスに対する「ポール」要求として発行される。このルーチンは、この要求を発行後直ちにコール元に戻る。ドライバは、DBE_PacketIsBusyルーチンを用いて、このパケットの完了を承認しなければならない。ドライバは、何らかのＩＯ完了処理をパケット上で実行した後、DBE_FreePacketを用いて、当該パケットを解放しなければならない。このルーチンは、プロセッサ割り込みをディスエーブルせず、目標請求要求キューに対するアクセスの同期を取るために、「スピン」ループを必要としない。

DBE_SubmitPacketPdeferred

このルーチンは、請求パケットを、目標ＤＢＥデバイスにポール・パケットとして発行する。このルーチンは、直ちにコール元に戻り、ＤＢＥデバイスとプロセッサとの間の処理の重複を最大化する。このルーチンは、目標ＤＢＥデバイスによるこのパケットの提出後直ちにコール元に戻る。ＤＢＥデバイスは、パケット２５０のVaLinkフィールド内の「提出」ビット（ビット[2:1]）をクリアすることによって、完了を指示する。ドライバは、何らかのＩＯ完了をパケット上で実行した後は、DBE_FreePacketを用いて当該パケットを解放しなければならない。このルーチンは、プロセッサ割り込みをディスエーブルせず、目標請求要求キューに対するアクセスの同期を取るために、「スピン」ループを必要としない。

上述のエントリポイントは、概略的に、以下のカテゴリの１つに該当する。即ち、初期化／初期化解除、パケット提出、パケット完了、同期ダイレクト・アクセス、パケット取り消し、及びパケット・プール増減である。

【００４１】

初期化／初期化解除

初期化／初期化解除ルーチンは、ホスト・ソフトウエアが、ＤＢＥハードウエア及びソフトウエア・リソースの初期化及び初期化解除を実行する際に用いられる。このカテゴリは、DBE_Initialize, DBE_Deinitialize及びDBE_GetDeviceという３つのルーチンを含む。ホスト・ソフトウエアは、その初期化フェーズ中にDBE_Initializeをコールし、ＤＢＥデバイス・オブジェクト２７２のハンドルを作成し、適切なアドレス情報を用いてＤＢＥハードウエア・レジスタ２８０～２９２を設定し、自動化ＤＭＡを容易にする。

ホスト・ソフトウエア２２０が、システムのリブート又はＯＳ２２２による動的アンローディング(dynamic unloading)によって再初期化される場合、DBE_Initializeルーチンを再度コールし、目標ＤＢＥデバイス２０６に対する新たな接続を作成し、新たなアドレス情報を用いて、ＤＢＥハードウエア・レジスタ２８０～２９２を再初期化しなければならない。

DBE_Deinitializeルーチンは、ホスト・ソフトウエア２２０が、目標ＤＢＥデバイス２０６のために割り当てられたメモリ・リソースを一掃し、目標ＤＢＥデバイス２０６をRESETモードに設定する際に用いられる。

DBEソフトウエア・アーキテクチャは、システム全体で作成されたＤＢＥデバイス・オブジェクト２７２全てを追跡する。ホスト・ソフトウエア２２０は、DBE_GetDeviceをコールすることによって、他のソフトウエア・エンティティが既に初期化したＤＢＥデバイス２０６に対する接続を確立することができる。ホスト・ソフトウエア２２０は、その初期化コンテクスト内からのこのルーチンのコールに限定されるのではない。DBE_GetDeviceは、いずれの実行コンテクスト内からでもコールすることができる。

【００４２】

パケット提出

パケット提出ルーチンは、ホスト・ソフトウエアが、請求パケットの割り当て、及びＤＢ

Ｅデバイス２０６への提出を実行する際に用いられる。このカテゴリは、DBE_AllocatePacket, DBE_SubmitPacketA, DBE_SubmitPacketP, DBE_SubmitPacketPdeferred、及びDBE_SubmitPacketIの５つのルーチンを含む。
ホスト・ソフトウエア２２０が、請求要求をＤＢＥデバイス２０６に提出可能となる前に、請求パケット・プール２００から請求パケット２５０を割り当てなければならない。ホスト・ソフトウエア２２０は、DBE_AllocatePacketをコールし、使用可能なパケット２５０を請求パケット・プール２００から除去する。ソフトウエアは、パケット２５０が使用可能となるまで、リトライ・カウントを指定するか、あるいはいつまでも待つことができる。DBE_AllocatePacketルーチンは、アプリケーション／デバイスが定義するいずれの方法でも使用可能な、空パケットのペイロード・データ・エリアへのポインタを返す。
一旦空パケット２５０が割り当てられたなら、ホスト・ソフトウエア２２０は、指定された請求要求チャネル２１４への請求トランザクション(solicited transaction)を実行するために必要な関連情報で、パケットのペイロード・データ・エリア２５４を満たす。ホスト・ソフトウエア２２０は、非同期、ポール、又は割り込みの３つの方法の内の１つで、パケット２５０をＤＢＥデバイス２０６に発行する。

【００４３】

非同期パケットは、ＤＢＥデバイス２０６からの完了通知を必要としない。ホスト・ソフトウエア２２０は、ソフトウエアが特定の処理に対して完了通知を必要としない場合、即ち、グラフィック処理が要求されたような場合に、非同期パケット２５０を提出する。ＤＢＥデバイス２０６は、パケットの完了時に、請求パケット・プール２００のテールに、自動的に非同期パケットを置く。ホスト・ソフトウエア２２０は、DBE_SubmitPacketAルーチンを用いて、非同期パケットを発行する。典型的な非同期処理は、グラフィックス・デバイス１２２によって実行される、スクリーン－スクリーン間ＢＬＴ(screen- screen- BLT)である。ポール要求は、ドライバ２３０がＩ／Ｏデバイス１１８に処理の完了を指示するように要求したときに、デバイス・ドライバ２３０によって発行されるが、ハードウエア割り込みの形態ではない。デバイス・ドライバ２３０は、 パケット・ヘッダ２５２内のキャッシュに基づくフィールドにポールすることによって、Ｉ／Ｏデバイス１１８が処理を完了するのを待つ。ポール・パケットは、パケット・ヘッダ２５２内のビットをクリアすることによって、ＤＢＥデバイス２０６からの完了通知を要求する。ホスト・ソフトウエア２２０は、パケット２５０の完了時にＤＢＥデバイス２０６が提出ビット（Va+Flagsフィールドのビット[2:1])のビット位置に"00"を書き込むまで、パケット・ヘッダ２５２内の提出ビット上で「スピン」即ちポールする。ドライバ２３０は、修正されたデータによる「無効化」サイクルがキャッシュ・ラインを更新するまで、そのキャッシュ内においてスピンを行う。これによって、Ｉ／Ｏデバイス１１８に直接ポールし、プロセッサ１００をホスト・バス１０２から分離しておく必要性がなくなる。

【００４４】

ホスト・ソフトウエア２２０は、DBE_SubmitPacketP及びDBE_SubmitPacketPdeferredルーチンを用いて、ポール・パケットを発行する。DBE_SublitPacketPルーチンは、ＤＢＥデバイス２０６がパケット・ヘッダ２５２内の提出ビットをクリアするまで、そのビット上での「スピン」動作を与える。DBE_SubmitPacketPdeferredルーチンは、要求をＤＢＥデバイス２０６に提出した後直ちに戻る。ホスト・ソフトウエア２２０は、パケット２５０がＤＢＥデバイス２０６によって処理されている間、他のタスクの処理を続けることができる。

割り込み要求は、ホスト・ソフトウエア２２０がハードウエア割り込みによる完了通知を要求するときに、デバイス・ドライバ２３０によって発行される。これは、より古典的なドライバ２３０とその目標ＰＣＩバス・マスタ１１８との間の通信方法である。ＤＢＥデバイス２０６は、割り込み要求を完了する際、完了したパケットの仮想アドレスを、システム・メモリ内の既知のメモリ・キューに書き込み、ハードウエア割り込みをアサートする。ＤＢＥデバイス２０６は、ドライバ２３０が未だ直前の完了に対応していない場合、要求を完了し続けることができる。この場合、ＤＢＥデバイス２０６は、システム・メ

モリ内に完了パケットのリストを構築し、完了キューが他の完了リストを受け取る準備ができたことをＩ／Ｏデバイス１１８が示すまで、他のハードウエア割り込みをアサートしない。ホスト・ソフトウエア２２０は、DBE_SubmitPacketIルーチンをコールすることによって、割り込みパケットを発行する。このルーチンは、パケット２５０の提出時に、直ちにコール元に戻る。ホスト・ソフトウエア２２０は、要求された割り込みイベントの到着まで、実行コンテクストの「ポスティング(posting)」を扱う。

【００４５】

パケット完了

パケット完了ルーチンは、ホスト・ソフトウエアが、ＤＢＥデバイス２０６に提出されたパケット２５０上で完了処理を実行する際に用いられる。このカテゴリは、DBE_PacketIsBusy, DBE_FreePacket, DBE_CompletionListIsReady, DBE_AcknowledgeInterrupt, 及びDBE_GetNextPacketToCompleteという５つのルーチンを含む。

非同期で提出された請求パケット２５０は、ホスト・ソフトウエア２２０による完了処理を要求しない。ＤＢＥデバイス２０６は、完了した非同期パケット２５０を自動的に「解放」し、請求パケット・プール２００のテールＴに配する。

ポール・パケット２５０は、何らかの完了処理がホスト・ソフトウエア２２０によって実行された後、請求パケット・プール２００上に返されなければならない。ソフトウエア２２０は、DBE_FreePacketをコールし、完了したパケット２５０を、請求パケット・プール２００のヘッド上に置く。

DBE_SubmitPacketPdeferredルーチンによって発行されたパケット２５０は、DBE_FreePacketを用いて請求パケット・プール２００のヘッド上にパケット２５０を置く前に、最初にDBD_PacketIsBusyルーチンをコールしなければならない。

【００４６】

ＤＢＥデバイス２０６は、割り込みパケット２５０を完了する際に、パケット完了リストを構築し、このリストの先頭を、ＤＢＥデバイス・オブジェクト２７２内に位置する完了キュー・ヘッド内に置く。ＤＢＥデバイス２０６は、完了リストを完了キュー・ヘッド内に置いた後、プロセッサ１００へのハードウエア割り込みをアサートして、ホスト・プロセッサにリストを通知する。

ホスト・ソフトウエアは、完了キュー・ホストを「覗いて」、DDBE_CompletionListIsReadyをコールすることにより、完了キュー・ヘッドからリストを実際に除去することなく、ＢＥデバイス２０６が実際にその割り込みをアサートしているか否か確認することができる。DBE_AcknowledgeInterruptルーチンは、ソフトウエア２２０が、完了キュー・ヘッドから完了リストを除去し、ＤＢＥデバイス割り込みを承認する際にコールされる。一旦このリストが除去されると、完了リストの終端に到達するまで、後続のDBE_GetNextPacketToCompleteへのコールによって、「検索(walk)」することができる。DBE_GetNextPacketToCompleteは、次のパケットを抽出しそれから完了させた後、直ちに、現パケットを請求パケット・プールのヘッド上に置くことによって、それを「解放」することができる。

【００４７】

同期ダイレクト・アクセス

同期ダイレクト・アクセスは、プロセッサ１００が共有ＤＢＥハードウエア・リソースに対するダイレクト・アクセスを要求する場合に有用である。ホスト・ソフトウエア２２０は、DBE_BeginDirectAccessをコールして、ポール・パケット２５０を提出し、指定されたチャネル（群）に対するＤＢＥデバイス２０６に、後続のパケット２５０の処理を停止させる。これによって、以前に提出されたパケットの処理とダイレクト・アクセス処理との間の接続がなくでも、ホスト・ソフトウエア２２０の目標デバイスのハードウエア・リソースに対するダイレクト・アクセスが可能となる。ホスト・ソフトウエア２２０は、DBE_EndDirectAccessルーチンをコールすることによって、指定されたチャネル（群）に対する同期ダイレクト・アクセスを終了する。

【００４８】

パケット取り消し

ＤＢＥデバイス２０６に発行され、ＤＢＥデバイス２０６による完了が未だ保留中のパケット２５０は、ホスト・ソフトウエアがDBE_CancelPacketをコールすることによって取り消すことができる。ＤＢＥデバイス２０６は、パケット２５０がVaフィールド２７０においてＩＯ取り消しのためのマークが付けられている場合、パケットのペイロード２５４をバーストしてはならない。取り消されたパケット２５０は、当該パケット２５０がどのように提出されたかに応じて、直ちにＤＢＥデバイス２０６によって完了される。

【００４９】

パケット・プールの増減

ホスト・ソフトウエア２２０は、より多くのパケット２５０を請求パケット・プール２００及び非請求パケット・プール２０２に追加することによって、それ自体と目標ＤＢＥデバイス２０６との間の切り離し(decoupling)量を増加させることができる。ソフトウエア２２０は、DBE_IncreasePoolをコールし、指定されたプールにパケット２５０を更に追加する。ソフトウエア２２０が、パケット・プール上に過剰に多いパケット２５０があり、いくつかのパケット２５０をＯＳのメモリ・マネージャに戻せることを発見した場合、ホスト・ソフトウエア２２０はDBE_DecreasePoolをコールし、指定されたパケット・プールからパケット２５０を除去することができる。

【００５０】

ＤＢＥアーキテクチャ

図６のＡ及び図６のＢを参照すると、分散型バースト・エンジン（ＤＢＥ）２０６に対する少なくとも２つの選択可能な位置が示されている。図６のＡでは、多数のＰＣＩバス・マスタ１１８がＤＢＥインターフェース２０６を含み、ＰＣＩバス・マスタ１１８とメモリ１１２との間で効率的にパケットをバーストする。図６のＢでは、ホスト－ＰＣＩ間ブリッジ１０６ａがＤＢＥ２０６を含み、ユニバーサル・シリアル・バスのホスト・コントローラ又はファイアウエア・コントローラのようなＩ／Ｏデバイス２０８とメモリ１１２との間で、パケット・アドレスを受け渡しを行う。

図６のＢの更に別の変形として、図６のＣは、ホスト・ブリッジ２１０が、１つ以上のプロセッサ１００と通信するためのホスト・バス１０２，メモリ１１２と通信するためのメモリ・バス２１２、及び周辺機器と通信するためのＰＣＩバス１１４に取り付けられた、コンピュータ・システムを示す。ホスト・ブリッジ２１０内には、ユニバーサル・シリアル・バス（ＵＳＢ）、ファイアウエアとしても知られているＩ．Ｅ．Ｅ．Ｅ．（電気電子学会）１３９４、又は小型コンピュータ・システム・インターフェース（ＳＣＳＩ）というような、Ｉ／Ｏチャネル２０８がある。Ｉ／Ｏチャネル２０８は、分散型バースト・エンジン２０６に接続されている。この実施形態は、図６のＡ及び図６のＢの実施形態とは異なる。何故なら、ＤＢＥ２０６は一層緊密にメモリ１１２に結合されているからである。これによって、ＤＢＥは、Ｉ／Ｏデバイス２０８に対してパケットをバーストする代わりに、パケット・アドレスをＩ／Ｏデバイス２０８に渡すことが可能となる。このようにして、Ｉ／Ｏデバイス２０８は、直接メモリ１１２からのパケット上で処理を行うことができる。ＰＣＩバス１１４を通じてパケットをバーストする際に伴うレイテンシイが排除されるので、これは図６のＡ及び図６のＢの結合が緩いＤＢＥに対する利点である。このように、ＤＢＥデバイス２０６は、種々の状況に対する適合性が非常に高い。

【００５１】

分散型バースト・エンジン（ＤＢＥ）

図７を参照すると、ＤＢＥデバイス２０６及びＩ／Ｏデバイス２０８を含む、ＰＣＩバス・マスタ１１８のブロック図が示されている。ＤＢＥデバイス２０６は、ＤＢＥデバイス２０６とメモリ１１２との間でＰＣＩバス１１４を通じてデータ・パケット２５０を交換するための、バス・マスタ・インターフェース（Ｉ／Ｆ）３００を含む。あるいは、ＤＢＥ２０６は、ＥＩＳＡバス又は他のいずれかの所望のバスに対するバス・マスタ・インターフェースを有することも可能である。また、ＤＢＥデバイス２０６は、あるフロント・エンド状態機械３０２～３０６とあるバック・エンド状態機械３０８～３１０との間に結合された３つのバッファ３１２～３１６も含む。これら３つのバッファは、要求キュー・

バッファ(RQ$_n$_BFR)３１２、完了キュー・バッファ(CQ_BRF)３１４、及びポステッド・キュー・バッファ(PQ_BFR)３１６である。フロント・エンドの目的は、要求キュー・バッファ３１２及びポステッド・キュー・バッファ３１６を満杯に保持し、完了キュー・バッファ３１４を空に保持することである。バック・エンドの目的は、要求キュー・バッファ３１２及びポステッド・キュー・バッファ３１６からパケット・アドレス及びデータ情報を引き出し、パケットを処理し、完了したパケットをプロセッサ１００に戻すことである。

【００５２】

要求キュー・バッファ３１２は、ホスト側から出された要求を整列するための先入れ先出し（ＦＩＦＯ）バッファである。供給キュー・バッファ３１２は、Ｉ／Ｏデバイス２０８が対応するチャネル数２１４に応じた、並列なｎ個のバッファからなる。好ましくは、要求キュー・フロント・エンド状態機械３０２及び要求キュー・バック・エンド状態機械３０８は、要求キュー・バッファ３１２全ての管理を担うが、状態機械の別個の集合を、各要求キュー・バッファ３１２毎に用いることも可能であると考えられる。簡略化のために、要求キュー・バッファ３１２は、ここでは単一のバッファとして扱う。

完了キュー・バッファ３１４は、完了したパケットを整列するためのＦＩＦＯである。ポステッド・キュー・バッファ３１６は、ホスト側の請求に基づくことなくホスト側にデータを転送することを要求する非請求要求を整列するためのＦＩＦＯである。バッファ３１２～３１６は、キューのデータ全体（キュー２７４～２７８）の一部分のみを収容する場合もあるので、バッファ３１２～３１６はそれらの各キュー２７４～２７８のサブセットと見なされる。バッファ３１２～３１６は、典型的にバス調定の結果として起こるＰＣＩバス１１４のレイテンシイから、Ｉ／Ｏデバイス２０８を切り離す。この結果、以前に得られたものよりも、Ｉ／Ｏ性能が向上する。

【００５３】

ここで、図示するＤＢＥデバイス２０６は、単一ストリームＤＢＥデバイスであることを注記しておく。ＤＢＥデバイス２０６が多機能周辺機器に対応することを要求される場合、多数の状態機械の集合を並列に実装し、単一のＩ／Ｏデバイス内に結合されている各機能毎に対応することができる。単一の状態機械の集合が、対応するＩ／Ｏデバイス２０８に対する全てのチャネル２１４を処理する。

フロント・エンド状態機械は、要求キュー・フロント・エンド状態機械(RQSMFEND)３０２、ポステッド・キュー・フロント・エンド状態機械(PQSMFEND)３０６、及び完了キュー・フロント・エンド状態機械(CQSM)３０４を含む。要求キュー・フロント・エンド状態機械３０２は、デバイス・ドライバ２３０によって要求キュー２７４上に置かれたパケット２５０を処理し、要求キュー・バッファ３１２に供給する役割を担う。ポステッド・キュー・フロント・エンド状態機械３０６は、メモリ１１２からポステッド・パケット・アドレスを引き出し、それらをポステッド・キュー・バッファ３１６内に置く役割を担う。

【００５４】

バック・エンド状態機械３０８～３１０は、要求キュー・バック・エンド状態機械(RQSMBEND)３０８及びポステッド・キュー・バック・エンド状態機械(PQSMBEND)３１０を含む。要求キュー・バック・エンド状態機械３０８は、要求キュー・バッファ３１２からパケット情報を引き出し、それをＩ／Ｏデバイス２０８に供給する役割を担う。ポステッド・キュー・バック・エンド状態機械３１０は、ポステッド・キュー・バッファ３１６から非請求パケット・アドレスを引き出し、それらをＩ／Ｏデバイス２０８に提示する役割を担う。

完了キュー状態機械(CQSM)３０４は、フロント・エンド及びバック・エンドと協同してパケットを完了し、完了キュー２７６及びフリー・キュー２７２のリンクを維持する役割を担う。また、ＤＢＥデバイス２０６には、多数のソース、即ち、Ｉ／Ｏデバイス２０８、要求キュー・バック・エンド状態機械３０８、及びポステッド・キュー・バック・エンド状態機械３１０から、データのフローを完了キュー・バッファ３１４に方向付けるための、完了キュー・マルチプレクサ(CQMUX)３１８も含まれる。完了キュー・マルチプレクサ３１８は、ポステッド・キュー・バック・エンド状態機械３１０によって制御される。

【００５５】
次に図８を参照すると、要求キュー・フロント・エンド状態機械３０２及びポステッド・キュー・フロント・エンド状態機械３０６のフロント・エンドの更に詳細なブロック図が示されている。要求キュー・フロント・エンド状態機械３０２について最初に説明する。図９のＡ、図９のＢ及び図９のＣは、要求キュー・フロント・エンド状態機械３０２のフロー・チャート、状態遷移図、及び出力を示す。要求キュー・フロント・エンド状態機械３０２の出力は、図９のＣに示されている。他に特に示さない限り、出力は、それに割り当てられた論理レベルから不変のままであるとする。
要求キュー・フロント・エンド状態機械３０２は、プロセッサ１００によって要求キュー２７４上に置かれたパケットを処理する役割を担う。要求キュー・フロント・エンド状態機械３０２は、Rq_n_REG２９０を用いて、プロセッサ１００が次のパケット・アドレスを置くメモリ・アドレスを得る。プロセッサ１００は、システムの初期化中に１回Rq_n_REG２９０に書き込み、再度それに書き込むことは決してない。要求キュー・フロント・エンド状態機械３０２は、要求キュー・フロント・エンド状態機械３０２が処理した最後のパケットの物理アドレスを用いて、Rq_n_REG２９０の内容を実行時間中保持する。要求キュー・フロント・エンド状態機械３０２は、要求キュー・バッファ３１２を満杯に維持し、要求キュー・バック・エンド状態機械３０８をバス・マスタＩ／Ｆ３００から切り離そうとする。
【００５６】
要求キュー・フロント・エンド状態機械３０２は、PA_LINKフィールド２６８を保持するために、要求キューPA_LINKレジスタ(RQ_PALINK)３２０を制御する。要求キュー・フロント・エンド状態機械３０２内部には、要求キュー・ヘッド・レジスタ(RQ_REG)２９０、現パケットのPALINKDWORDフィールドを保持するためのパケット長レジスタ(RQ_LENGTH)、及び次のパケットのPaLinkDWORDを保持するための次パケット長レジスタ(RQ_NXPLENGTH)がある。RQ_LENGTHレジスタの出力は、バス・マスタＩ／Ｆ３００に供給されるので、パケット２５０の中を適正に読み取ることができる。以下の検討は単一のチャネル要求キューについてのみ言及するが、多数のチャネルが使用可能であることは理解されよう。
パケット２５０が互いにリンクされていることを思い出すのは重要である。したがって、リンク・リスト内のどのパケット２５０を状態機械が処理しているのかを明確にするために、以下の規則を用いる。「現パケット」とは、現在処理又は転送されている最中のパケットを意味する。「前パケット」とは、現パケットに先立ってリンクされ、現パケットよりも前に処理されたパケットを意味する。「次パケット」とは、現パケットの後にリンクされ、現パケットよりも後に処理されるパケットを意味する。したがって、前パケットのPaLinkフィールド２６８は現パケットを示し、現パケットのPaLinkDWORDフィールド２６６は、次パケットに対するDWORDの数を示す。これは全てのパケット型に対して一貫している。
【００５７】
要求キュー・フロント・エンド状態機械３０２は、システム・リセットからIDLE状態に初期化する。RESETモードにある間、デバイス・ドライバ２３０は、請求パケット・プール２００（又はフリー・キュー２７２）上の最後のパケット２５０の物理アドレスを用いて、RQ_REG２９０を初期化する。フリー・キュー２７２は、ＤＢＥデバイス２０６が完了した非同期パケットを整列する場所を有するために、請求パケット・プール２００上に少なくとも１つの空きパケット２５０を維持する必要がある。
IDLE状態３３０において、要求キュー・フロント・エンド状態機械３０２は、RQ_DOORBELLが鳴らされるのを待つ。RQ_DOORBELLを待っている間、要求キュー・フロント・エンド状態機械３０２は、RQ_REG２９０からの値（通常、ＤＢＥによって処理された最後の要求パケットの物理アドレス）を用いて、要求キュー・リード・アドレス・レジスタ(RQ_READADDR_B)を初期化し、RQ_LENGTHレジスタは２ｈに初期化される。ＤＢＥ２０６が丁度デバイス・ドライバ２３０によって初期化されたばかりである場合、RQ_READADDR_Bは、デバイス・ドライバ２３０によってRQ_REG２９０に書き込まれた値によって初期化される。RQ_D

OORBELL信号がアサートされると、要求キュー・フロント・エンド状態機械３０２は、クリア要求キュー・ドアベル(CLR_RQ_DOORBELL)信号をアサートし、NEXTPKT状態３３２に遷移する。

【００５８】

NEXTPKT状態３３２において、RQ_DOORBELLがクリアされ、パケット２５０がメモリ１１２から読み出され、要求キュー・バッファ３１２に書き込まれる。バス・マスタＩ／Ｆ３００は、PaLinkDWORDフィールド２６６を用いてパケット情報を供給し始め、ペイロードの終端に到達するまで続ける。PaLinkDWORDフィールド２６６の前のフィールド（図３参照）は、要求キュー・フロント・エンド状態機械３０２には必要ではない。

現パケット２５０が書き込まれると、前パケットからの値が要求キュー・バッファ３１２上に押し出される。現パケットのPaLinkDWORDフィールド値２６６がRQ_NXPLENGTHレジスタにラッチされると、RQ_LENGTHレジスタに現在保持されているPaLinkDWORDフィールド値２６６は、要求キュー・バッファ３１２上に押し出される。マルチプレクサ(MUX)３２２は０入力から１入力に切り替えられるので、現在RQ_PALINKレジスタ３２０に保持されているPaLinkフィールド値２６８は、現パケットのPaLinkフィールド値２６８がRQ_PALINKレジスタ３２０にラッチされると、要求キュー・バッファ３１２上に押し出される。その後、マルチプレクサ３２２は、０入力からのデータを受け取るように、再度切り替えられる。

【００５９】

現パケットのVa+Flagsフィールド値２７０は、現パケットの残りのペイロード２５４であるので、要求キュー・バッファ３１２上に押し出される。現パケット２５０が要求キュー・バッファ３１２に書き込まれる際、Va+Flagsフィールド２７０をチェックし、Ｉ／Ｏ取り消しか又はダイレクト・アクセスかを調べる。Va+Flagsフィールド２７０のビット３又はビット４のいずれかがセットされている場合、ペイロードを読み出す前にパケット転送サイクルを中断する。パケット２５０の各DWORDを読み出すに連れて、RQ_READADDR_Bレジスタをそれに応じて増分する。

ペイロード２５４が転送されたとき、RQ_RDDON E信号がバス・マスタＩ／Ｆ３００から受け取られる。RQ_RDDONE信号が受け取られ、RQ_PALINKレジスタが０値を収容している場合、要求キュー・フロント・エンド状態機械はIDLE状態３３０に戻り、別のRQ_DOORBELLを待つ。

【００６０】

RQ_RDDONE信号が受け取られ、更にRQ_PALINKレジスタが０値を収容している場合、別のパケット転送サイクルが必要となる。この場合、状態機械３０２は、RQ_NXPLENGTH内の次パケット長値を、３ｈだけ増分し、この値をRQ_LENGTHレジスタにラッチし、RQ_PALINKレジスタからの値をPQ_READADDR_Bレジスタにラッチする。

RQ_RDDONE信号が受け取られ、RQ_PALINKレジスタが０ではないが、要求キュー・バッファ３１２が満杯である場合(RQFULL)、要求キュー・フロント・エンド状態機械３０２はWAIT状態に遷移する。状態機械３０２は、要求キュー・バッファ３１２が満杯でない場合、再びWAIT状態からNEXTPKT状態に遷移する。RQFULL信号は、要求キュー・バッファ３１２によって供給される。いずれの状態においても、要求キュー・バッファ３１２が満杯でない場合、RQ_PALINKレジスタ内に格納されている値がRQ_REGにラッチされる。RQ_LENGTH, RQ_READADDR_B, 及びRQ_REGレジスタにロードされた後、別のパケット転送サイクルが始まる。

【００６１】

次に、図８及び図１０を参照して、ポステッド・キュー・フロント・エンド状態機械３０６を更に詳細に説明する。図１０のＡ及びＢは、ポステッド・キュー・フロント・エンド状態機械３０６のフロー・チャート及び状態遷移条件を示す。ポステッド・キュー・フロント・エンド状態機械３０６の出力を図１０のＣに示す。他に特に示していない限り、出力は不変のままとする。

ポステッド・キュー・フロント・エンド状態機械(PQSMFEND)３０６は、メモリ１１２から

ポステッド・パケット・アドレスを引き出し、それらをポステッド・キュー・バッファ３１６内に置く役割を担う。ポステッド・キュー・フロント・エンド状態機械３０６は、常に、３つのDWORD（現パケット２５０に対してPALINKDWORDS，PA_LINK 及びVA+FLAGS）を初期リード上で要求し、２つのDWO RD（PA_LINK及びVA+FLAGS)を後続の全てのリード上で要求する。プロセッサ１００は、システムの初期化中にポステッド・パケット２５０のリストを構築し、次に、ポステッド・キュー２７８内の最初のパケット２５０の物理アドレスをPQ_REG２８８に書き込む。ポステッド・キュー・フロント・エンド状態機械３０６は、ポステッド・パケット２５０のリストを見渡し、パケット・アドレスをプリフェッチする。アドレス(PA_LINK及びVA)は、バッファ３１６が満杯になるまで、又はメモリ１１２内の非請求パケット・プール２０２上にもはやポステッド・パケット２５０がなくなるまで、ポステッド・キュー・バッファ３１６内に置かれる。リスト上に残っているパケット２５０が１つのみの場合、非請求パケット・プール２０２は、空と見なされる。

【００６２】

ポステッド・キュー・フロント・エンド状態機械３０６は、ポステッド・キュー・レジスタ(PQ_REG)２８８、ならびに次のパケットのペイロードのPA_LINK及び長さ(PQ_PALINK及びPQ_LENGTH)を保持するための他のレジスタを内蔵する。

ポステッド・キュー・フロント・エンド状態機械３０６は、システム・リセットからREALIDLE状態３３６に初期化する。RESETモードにある間、デバイス・ドライバ２３０は、非請求パケット・プール２０２内の最初のパケットの物理アドレスを用いて、PQ_REG２８８を初期化する。その後、ポステッド・キュー・フロント・エンド状態機械は、非請求パケット・プール２０２のヘッドの物理アドレスを、PQ_REG２８８に保持する。

REALIDLE状態３３６では、ポステッド・キュー・フロント・エンド状態機械３０６は、PQ_DOORBELLが鳴らされるのを待つ。デバイス・ドライバ２３０がPQ_DOORBELLを鳴らすのは、パケット２５０がＤＢＥ２０６に使用可能となったときである。これは、DBE_FreePacket又はDBE_GetNextPacketToCompleteが非請求パケットによってコールされる毎に発生する。PQ_DOORBELLビットがセットされると、ポステッド・キュー・フロント・エンド状態機械３０６は、IDLE状態３３８に遷移する。

【００６３】

IDLE状態３３８では、PQ_DOORBELLがクリアされ、ポステッド・キュー・リード要求信号(PQ_RDREQ)がアサートされる。ポステッド・キュー・フロント・エンド状態機械３０６は自動的にIDLE状態３３８からGETPAVA状態３４０に遷移する。

GETPAVA状態３４０における第１リード・サイクルにおいて、PA_LINKフィールド値２６８はポステッド・キューPaLinkレジスタ(PQ_PALINK)にラッチされ、PaLinkDWORDフィールド値２６６はポステッド・キュー長レジスタ(PQ_LENGTH)にラッチされ、Va+Flagsフィールド値２７０はラッチされる。後続のリード・サイクルでは、PA_LINK及びVa+FLAGSフィールドのみが読み取られる。何故なら、ペイロード２５４のサイズが既にわかっているからである。各リード・サイクル毎に、PA_LINKフィールド及びVA+FLAGSフィールドは、ポステッド・キュー・バッファ３１６に押し出される。現パケット２５０のPA_LINKフィールドが０に等しい場合、ポステッド・キュー・フロント・エンド状態機械３０６は、再びREALIDLE状態に遷移し、PQ_DOORBELLビットが再度セットされるのを待つ。

現パケット２５０のPA_LINKフィールドが０に等しくない場合、現パケットには追加のパケット２５０がリンクされていることになる。この場合、ポステッド・キュー・フロント・エンド状態機械３０６は、QVA状態３４２に遷移し、更にIDEL状態３３８に戻り、次パケット２５０をフェッチする。ポステッド・キュー・バッファ３１６が満杯となっている場合(PWFULL信号）、パケットのフェッチは抑制される。

【００６４】

次に図１１を参照すると、要求キュー・バック・エンド状態機械(RQSMBEND)３０８及びポステッド・キュー・バック・エンド状態機械(PQSM BEND)３１０の更に詳細なブロック図が示されている。要求キュー・バック・エンド状態機械３０８について最初に説明する。図１２及び図１３は、要求キュー・バック・エンド状態機械３０８のフロー・チャート及

び状態遷移条件を示す。要求キュー・バック・エンド状態機械３０８の出力を、図１４に示す。他に特に示されていない限り、出力は、それらに予め割り当てらた論理値から不変のままとする。
要求キュー・バック・エンド状態機械３０８は、要求キュー・バッファ３１２の１つからパケット情報を引き出す役割を担う。要求キュー・バック・エンド状態機械３０８は、要求キュー・バッファ３１２から物理アドレス及び仮想アドレスを引き出し、Ｉ／Ｏデバイス２０８にデータ２５４を引き出させる。パケット２５０が既にＩ／Ｏデバイス２０８によって処理されている場合、Ｉ／Ｏデバイス２０８は、要求キュー・バック・エンド状態機械３０８に通知し、パケット２５０を完了する。パケット２５０には、割り込み、ポール、又は非同期完了とマークが付けられる。要求キュー・バック・エンド状態機械３０８は、完了の間、プロセッサ１００のＤＢＥデバイス２０６へのダイレクト・アクセスをパケット２５０と同期させる役割も担っている。
【００６５】
要求キュー・バック・エンド状態機械３０８は、システム・リセット又はRESETモードからIDLE状態３５０に初期化し、要求キュー・バッファ３１２からの!RQEMPTY信号が、要求キュー・バッファ３１２が空でないことを示すのを待つ。空でない場合、要求キュー・バック・エンド状態機械３０８はGETLENGTH状態３５２に遷移し、要求キュー(dq_RQ)信号をアサートし、最初のDWORDを要求キュー・バッファ３１２からポップする(pop)。
GETLENGT H状態３５２において、状態機械３０８は最初のDWORDを長さレジスタ(RQB_LENGTH)に書き込む。最初のDWORDは、現パケット２５０のPALINKDWORDフィールド値２６６である。要求キュー・バッファ３１２が空でない場合、要求キュー・バック・エンド状態機械３０８はGETPALINK状態３５４に遷移し、要求キュー(dq_RQ)信号をアサートし、他のDWORDを要求キュー・バッファ３１２からポップする。要求キュー・バッファ３１２が空の場合、要求キュー・バック・エンド状態機械３０８は、要求キュー・バッファ３１２が空でなくなるまで、GETLENGTH状態３５２に止まる。
【００６６】
GETPALINK状態３５４において、状態機械３０８は２番目のDWORDを要求キューPA_LINKレジスタ(RQB_PALINK)に書き込む。２番目のDWORDは、現パケット２５０のPA_LINKフィールド値２６８である。要求キュー・バッファ３１２が未だ空でない場合、要求キュー・バック・エンド状態機械３０８はGETVA状態３５６に遷移し、要求キュー(dq_RQ)信号をアサートし、別のDWORDを要求キュー・バッファ３１２からポップする。要求キュー・バッファ３１２が空になった場合、要求キュー・バック・エンド状態機械３０８は、GETPALINK状態３５４に止まる。
GETVA状態３５６において、状態機械３０８は３番目のDWORDを要求キューVA+FLAGSレジスタ(RQB_VA+FLAGS)に書き込む。３番目のDWORDは、現パケット２５０のVA+FLAGSフィールド値２７０である。VA+FLAGSフィールド値２７０を書き込む間、状態機械３０８は、ビット４でＩ／Ｏ取り消しをチェックし、更にビット３でダイレクト・アクセスをチェックする。ビット３は、RQB_LENGTH=0の場合にのみチェックするように制限が設けられる。いずれかのビットがセットされている場合、状態機械３０８はWRITE状態３６０（以下で論ずる）に遷移し、パケットはＩ／Ｏデバイス２０８に提出されない。ビット３及び４がセットされていないが、RQB_LENGTH=0である場合、状態機械３０８は、DONE状態３６２（以下で論ずる）に遷移する。その他の場合、状態機械３０８はNEXTDWORD状態３５８に遷移し、DQRDY信号をアサートして、Ｉ／Ｏデバイス２０８にデータが準備できていることを示す。
【００６７】
NEXTDWORD状態３５８において、DQ_RQ信号をアサートし、DWORDを要求キュー・バッファ３１２からポップする。DWORDがＩ／Ｏデバイス２０８によって読み込まれる毎に、デバイス・レディ(DEVICEDQRDY)信号がＩ／Ｏデバイス２０８から受け取られ、LENGTHレジスタを１だけ減分する。LENGTH値が１ｈに達したとき、DQRDY信号をディアサートし、状態機械３０８はDONE状態３６２に遷移する。

DONE状態３６２において、状態機械３０８は、非同期、割り込み、又はポールの内どの型のパケットが完了したかについて判定を行なう。DQ_RQ信号をディアサートし、パケット情報をキューから取り出す(dequeue)のを停止する。パケット型を判定するために、状態機械３０８はRQB_VALINKレジスタ内に格納されているVALINKフィールド値２７０のビット１及び２を検査する。ビット２：１＝１ｈ(DBE_PKT_SUBMIT_A)の場合、状態機械３０８はWRITE状態３６０に遷移する。ビット２：１＝２ｈ(DBE_ PKT_SUBMIT_P)又は２：１＝３ｈ（DB E_PKT_SUBMIT_I)の場合、状態機械３０８はWAIT状態３６６に遷移し、Ｉ／Ｏデバイス２０８へのDONE信号をアサートし、パケット２５０が処理されたことの承認を要求する。

【００６８】

WAIT状態３６６において、状態機械３０８はＩ／Ｏデバイス２０８からの完了通知を待つ。PROCESS_DONE信号がＩ／Ｏデバイス２０８から受け取られ、パケットが非同期パケットであった場合、DONE信号はディアサートされ、状態機械３０８は再びIDLE状態３５０に遷移する。PROCESS_DONE信号がＩ／Ｏデバイス２０８から受け取られ、パケットが非同期パケットでなかった場合、DONE信号はディアサートされ、状態機械３０８はWRITE状態３６０に遷移する。PROCESS_DONE信号が受け取られない場合、状態機械はWAIT状態３６６において待機する。

WRITE状態３６０において、現パケット２５０のPA_LINK２６８及びVA+FLAGS２７０フィールドの値を、完了キュー２７６に書き込む。パケットの型もキュー上でマークするので、フロント・エンドは、それが処理しようとするパケットの型がわかる。

【００６９】

状態機械３０８は、完了キュー２７６が満杯でなく(CQFULL)、ポステッド・キュー・バック・エンド状態機械３１０が使用中でない(CQBUSY)場合、AGAIN信号に基づいて、WRITE状態３６０を通じて２回のパスを行なう。その他の場合、状態機械３０８は、完了キュー２７６が満杯でなくなり、使用可能となるまで待機する。AGAINは最初にセットされるので、最初のパスではAGAINはセットされず、２回目のパスでセットされる。

最初のパスでは、現パケットの物理アドレス(PA)が、完了キュー２７６に書き込まれる。２回目のパスは、パケット型によって異なる。

パケット型が非同期の場合、現パケット２５０の仮想アドレスが完了キュー２７６に書き込まれ、ビット３３：３４に2h が書き込まれ、非同期パケットであることを意味する。状態機械３０８は、パケット２５０にＩ／Ｏ取り消しのマークが付けられている場合、又はダイレクト・アクセスのマークが付けられていない場合、WAIT状態３６６に遷移する。その他の場合（パケット２５０に、Ｉ／Ｏ取り消しのマークが付けられていなかったが、ダイレクト・アクセスのマークが付けられていた場合）、状態機械３０８はDIRECTACC状態３６４に遷移し、対応する要求レジスタ(Rq_n_REG)内のダイレクト・アクセス・ビットをクリアし、チャネルが使用可能であることあるいは同期されていることを、デバイス・ドライバ２３０に示す。

【００７０】

パケット型がポールの場合、現パケット２５０の仮想アドレスが完了キュー２７６に書き込まれ、ビット３３：３４に1h が書き込まれ、ポール・パケットであることを意味する。状態機械３０８は、パケット２５０にＩ／Ｏ取り消しのマークが付けられていたか、あるいはダイレクト・アクセスのマークが付けられていなかった場合、IDLE状態３５０に遷移する。その他の場合（パケット２５０にはＩ／Ｏ取り消しのマークが付けられていなかったが、ダイレクト・アクセスのマークが付けられていた場合）状態機械３０８はDIRECTACC状態３６４に遷移し、対応する要求レジスタ(Rq_n_REG)内のダイレクト・アクセス・ビットをクリアし、デバイス・ドライバ２３０に、チャネルが使用可能であることあるいは同期されていることを示す。

パケット型が割り込みの場合、現パケット２５０の仮想アドレスが完了キュー２７６に書き込まれ、ビット３３：３４に3h が書き込まれ、割り込みパケットであることを意味する。状態機械３０８は、パケット２５０にＩ／Ｏ取り消しのマークが付けられていた場合

、IDLE状態３５０に遷移する。パケット２５０にダイレクト・アクセスのマークが付けられていた場合、状態機械３０８はDIRECTACC状態３５０に遷移し、対応する要求レジスタ(Rq_n_REG)内のダイレクト・アクセス・ビットをクリアし、デバイス・ドライバ２３０に、チャネルが使用可能であることあるいは同期されていることを示す。その他の場合、状態機械３０８はWAIT状態３５０に遷移する。

DIRECTACC状態３６４において、状態機械３０８は、デバイス・ドライバ２３０がダイレクト・アクセス動作を完了するのを待つ。状態機械３０８は、デバイス・ドライバ２３０が要求レジスタ(Rq_n_REG)内の終了ダイレクト・アクセス・ビット（ビット３）をセットするまで、要求キュー・バッファ３１２からいかなる追加データも引き出さない。

【００７１】

図１５Ａ、及びＢには、図１１のポステッド・キュー・バック・エンド状態機械(PQSMBEND)３１０の対応するフロー・チャート、及び状態遷移条件が示されている。ポステッド・キュー・バック・エンド状態機械３１０の出力を図１５のＣに示す。他に特に示されていない限り、出力は、それらに予め割り当てらた論理値から不変のままとする。

ポステッド・キュー・バック・エンド状態機械３１０は、ポステッド・キュー・バッファ３１６から非請求パケットのアドレスを引き出し、それらをＩ／Ｏデバイス２０８に提示する役割を担う。Ｉ／Ｏデバイス２０８は、これらのアドレスをメモリ１１２内の位置として用い、非請求データを「ダンプ」する。ポステッド・キュー・バック・エンド状態機械３１０は、常に、２つのDWORD (PALINK及びVA+FLAGS)をポステッド・キュー・バッファ３１６から除去する。ＤＢＥデバイス２０６によって受け取られた非請求データは、ポステッド・キュー・フロント・エンド状態機械３０６によって供給された物理アドレス及び仮想アドレスを用いて、メモリ１１２に書き込まれる。

【００７２】

ポステッド・キュー・バック・エンド状態機械３１０は、非請求パケットの物理ペイロード・アドレスを完了キュー・バッファ３１４に書き込む。次いで、ポステッド・キュー・バック・エンド状態機械３１０は、Ｉ／Ｏデバイス２０８に、その非請求データを、現パケット２５０のための非請求パケットのペイロード・データ・エリアの長さだけ、完了キュー・バッファ３１４に「ダンプ」するように要求する。一方、Ｉ／Ｏデバイス２０８は、データの「ダンプ」を完了したときに、ポステッド・キュー・バック・エンド状態機械３１０に通知する。次に、ポステッド・キュー・バック・エンド状態機械３１０は、非請求パケットの仮想及び物理アドレス・ヘッダ情報を、完了キュー・バッファ３１４内に置く。完了キュー状態機械３０４は、このパケットを割り込み型パケット２５０として完了する。

ポステッド・キュー・バック・エンド状態機械３１０は、システム・リセット又はRESETモードからIDLE状態３７０に初期化し、Ｉ／Ｏデバイス２０８からのポステッド・パケット要求(POSTPACKETREQ)信号を待つ。 POSTPACKETREQ信号がアサートされ、ポステッド・キュー・バッファ３１０が空いておらず(PQEMPTY)、更にマルチプレクサ制御信号(MUXVA)がアサートされていない場合、完了キュー・ビジー(CQBUSY)信号がセットされ、現パケットの物理アドレスがポステッド・キュー・バッファ３１０から除去されてラッチされ、ペイロードの物理アドレスが完了キュー上に置かれ、マルチプレクサ制御信号がセットされ(１)、Ｉ／Ｏデバイス２０８からデータを受け取り、状態機械３１０はGETVA状態３７２に遷移する。

【００７３】

GETVA状態３７２において、ポステッド・キュー・バッファ３１６が空でない場合(PQEMPTY)、状態機械は、Ｉ／Ｏデバイス２０８に、ポステッド・パケット２５０が準備できている(POSTPACKETRDY)ことを通知し、データを完了キュー・バッファ３１４に「ダンプ」し、WAIT状態３７４に遷移する。

WAIT状態３７４において、ポステッド・パケット終了(POSTPACKETDONE)信号がＩ／Ｏデバイス２０８から受け取られた場合、マルチプレクサ制御信号を再度セットし(０)、ポステッド・キュー・バック・エンド状態機械３１０からパケット仮想アドレス(VA+FLAGS)を

受け取る。POSTPACKETDONE信号が受け取られ、完了キュー・バッファ３１４は満杯ではなく、POSTPACKETRDY信号がアサートされた場合、状態機械３１０はPOSTPACKETRDY信号をディアサートし、現パケットの物理アドレス(PA_LINK)を完了キュー・バッファ３１４上に押し出す。完了キュー・バッファ３１４が満杯でなく、POSTPACKETRDY信号がディアサートされた場合、仮想アドレス(VA+FLAGS)を完了キュー２７６上に置く。

【００７４】

次に、図１６～図１９を参照すると、完了キュー状態機械３０４を更に詳細に説明する。図１７は、バス・マスタＩ／Ｆ３００の制御のフロー・チャートである。完了キュー状態機械３０４は、バス・マスタＩ／Ｆ３００が完了キュー・バッファ３１４からアドレス及びデータを引き出す際、マルチプレクサ(MUX2)３２４を通じて、データのフローを制御する。

図１７を参照する。バス・マスタＩ／Ｆ３００は、IDLE状態４００において待機している。ステップ４００において、完了キューVAリンク要求(CQ_VALINKREQ)信号が受け取られたか、あるいは完了キュー・バッファ３１４が空でない場合、バス・マスタＩ／Ｆ３００はステップ４０２に遷移し、CQ_VALINKREQ信号を検査する。 CQ_VALINKREQが完了キュー状態機械３０４によってアサートされている場合、バス・マスタＩ／Ｆ３００はステップ４０４に進む。 CQ_VALINKREQがない場合、バス・マスタＩ／Ｆ３００はステップ４０６に進み、完了キュー・バッファ３１４の出力のビット３３：３２を検査する。ビット３３：３２が０に等しい場合、バス・マスタＩ／Ｆ３００はステップ４０８に進む。ビット３３：３２が０でない場合、バス・マスタＩ／Ｆ３００はステップ４１０に進み、ビット３３を検査する。ステップ４１０において、ビット３３が０に等しくない場合、バス・マスタＩ／Ｆ３００はステップ４０２に戻る。ステップ４１０において、ビット３３が０に等しい場合、バス・マスタＩ／Ｆ３００はステップ４１２に進み、パケット・アドレス情報を除去する。ステップ４１２から、バス・マスタＩ／Ｆ３００はステップ４１４に進み、完了キュー・バッファ３１４からパケット・データを除去する。ステップ４１４から、制御はIDLE状態４００に戻る。

【００７５】

ステップ４０４において、バス・マスタＩ／Ｆ３００は、完了キューVAリンク承認(CQ_VALINKACK)信号をアサートする。ステップ４０４から、制御はステップ４１２に進み、パケット・アドレス情報を除去し、CQ_VALINKACK信号をディアサートする。

ステップ４０８において、バス・マスタＩ／Ｆ３００は、パケット・アドレス情報を除去する。ステップ４０８から、制御はステップ４１６に進み、除去したパケット・アドレスのビット３４：３２を検査する。ビット３４：３２が０に等しい場合、制御はステップ４１８に進み、完了キュー・バッファ３１４からパケット・データを除去する。ステップ４１６，４１８は、ビット３４：３２がもはや０に等しくなくなる（パケット・データを除去する）まで繰り返される。ステップ４１６において、ビット３４：３２が０に等しくない場合、制御はIDLE状態４００に戻る。

完了キュー状態機械(CQSM)３０４は、完了及びフリー・キュー・リンクを維持する役割を担う。また、完了リンク・リストをCQ_REG２８６に提示し、 CQ_REG２８６のライト完了時に、バス・マスタＩ／Ｆ３００に割り込みをセットすることを通知する役割も担う（ポステッド・パケット及び割り込みパケットのみ）。完了キュー状態機械３０４は、請求及び非請求イベントから完了パケットを受け取る。要求キュー２７４に発行された割り込みパケット、及びＩ／Ｏデバイス２０８によって非請求データで満たされた非請求パケット・プール２０２から発した全てのパケットには、完了が要求される。

【００７６】

プロセッサ１００は、完了キューが処理され(service)、プロセッサ１００が他の完了パケットのリストの準備ができたときに、CQ_REG２８６のCQ_DOORBELLビットを鳴らす(ring)。メモリ１１２内の完了キュー２７６がプロセッサ１００によって処理されていない場合、完了キュー状態機械３０４は、現完了パケットの仮想アドレスを、直前に完了したパケット２５０のVAフィールド２７０に書き込む。完了キュー状態機械３０４は、次回パケ

ットが完了を要求するときのために、現パケット２５０の物理アドレスをテールＴとしてセーブする。
メモリ１１２内の完了キュー２７６が完了パケット２５０のリストを受け入れる準備ができている場合、完了パケットのリストにおける最初のパケット２５０の仮想アドレスVAが、メモリ１１２の完了キュー２７６に書き込まれる。完了キュー２７６のヘッドの物理アドレスは、CQ_REG２８６内に位置付けられる。その後、完了キュー状態機械２０４の内部ヘッド／テール・レジスタは消去(null）され、新たな完了リストを開始する。
完了キュー状態機械３０４は、ドライバによって非同期要求として提出されたパケット２５０も完了させる。非同期パケット２５０内に指定されている処理が終了したなら、完了キュー状態機械３０４は、パケット２５０の仮想アドレスVAを請求パケット・プール２００のテールＴ上に置く。請求パケット・プール２００のテールＴは、 FQ_REG ２８４を通じて、ＤＢＥデバイス２０６によって維持される。パケットの仮想アドレスVAが直前のパケットのV ALINKフィールド２６０に書き込まれた後、完了キュー状態機械３０４は、パケットの物理アドレスPAをFQ_REG２８４内に置く。
【００７７】
図１８及び図１９を参照し、完了キュー状態機械の動作について説明する。完了キュー状態機械３０４は、システム・リセット時又はRESETモード時に、IDEL状態４３０に初期化する。完了キュー状態機械３０４は、次の３つのイベントの１つが発生するまで、IDLE状態で待機する。１）完了キュー・ドアベル(CQ_DOORBELL)が鳴らされ、完了キュー・バッファ３１４がヌルでない。２）完了した割り込みパケット２０を、完了キュー２７６のテールにリンクする必要がない。又は３）完了した非同期パケットが、完了キュー２７６のテールＴにリンクする必要がある。
完了キュー・ドアベル(CQ_DOORBELL)が鳴らされ、完了キュー・ヘッド・レジスタ(CQ_HEAD)４３８がヌルでない場合、完了キュー状態機械３０４は、CQ_VALINKREG信号及びバス・マスタＩ／Ｆ３００へのセット割り込み(SETINTERRUPT)信号をアサートし、バス・マスタＩ／Ｆ３００が割り込みの準備ができているか否か調べる。CQ_HEAD４３８は、状態機械３０４によって維持され、完了リストを追跡する。バス・マスタＩ／Ｆ３００は、CQ_VALINKACK信号によって応答し、状態機械３０４は、MUX2に、CQ_REG内の物理アドレスを通過させ、次いで、CQ_HEAD内に位置する完了リストのヘッドの仮想アドレスをバス・マスタＩ／Ｆ３００に受け渡す。次に、CQ_HEAD及び完了キュー・ドアベル・ビット(CQ_DOORBELL)をクリアし、状態機械３０４はWAIT状態４３４に遷移する。
【００７８】
CQ_DOORBE LLがセットされていないか、あるいはCQ_HEADが０に等しい場合で、かつバス・マスタＩ／Ｆ３００からのレディ信号(GO)がセットされている場合、状態機械３０４は、完了キュー２７６から最初のDWORDを引き出し（現パケットの物理アドレス）、それを完了キュー・テール・レジスタCQ_TAIL４４０に書き込む。このDWORDは、完了キュー２７６上の最後のパケットの物理アドレスである。次に、状態機械３０４は、BFR_HEAD状態４３６に遷移する。
BFR_HEAD状態において、現パケットの仮想アドレスを格納する。完了キュー・ヘッド・レジスタが０に等しくなく、CQ_DOORBELLがセットされていない場合、状態機械３０４はCQ_VALINKREQ信号をアサートし、直前のパケットの物理アドレス(CQ_TAIL)をバス・マスタＩ／Ｆ３００に受け渡す。バス・マスタＩ／Ｆ３００がCQ_VALINKACK信号で応答すると、完了キュー・バッファ３１４の最初のDWORDがCQ_TAILレジスタ４４０に書き込まれ、状態機械３０４はASYNCHPAK状態４３２に遷移する。（現パケットの仮想アドレスは、直前のパケットの物理アドレス(CQ _TAIL)に書き込まれる。）
【００７９】
CQ_DOORBELLがセットされていないか、あるいはCQ_HEADが０に等しい場合で、かつバス・マスタＩ／Ｆ３００からのレディ信号(GO)がセットされ、現パケットが非同期パケットであることをビット３４：３３が示す場合、状態機械３０４はCQ_VALINKREQ信号をアサートし、次のパケットの仮想アドレスをバス・マスタＩ／Ｆ３００に受け渡す。バス・マスタ

Ｉ／Ｆ３００がCQ_VALINKACK信号で応答すると、FQ_REG２８４の内容がバス・マスタＩ／Ｆ３００に受け渡され、完了キュー・バッファ３１４からの最初のDWORDがFQ_REG２８４に書き込まれ、完了キュー・バッファ３１４からの仮想アドレスがFQ_REG２８４の直前の内容に書き込まれ、状態機械３０４はASYNCHPAK状態４３２に遷移する。
ASYNCHPAK状態４３２において、状態機械３０４は、GO信号がディアサートされ、IDLE状態４３０に遷移するのを待つ。デバイス・ディキュー(DEVICEDQ)信号が受け取られ、AGAIN信号がアサートされていない場合、MUX2がセットされ、完了キュー・バッファ３１４からのアドレスを受け取る。DEVICEDQ信号は、バス・マスタＩ／Ｆ３００によって供給され、完了キュー・バッファ３１４からデータを引き出す。AGAIN信号は、状態機械３０４をASYNCHPAK状態４３２に止めておく。MUX2は、本質的に、完了キュー・バッファ３１４とFQ_REG２８４との間の切り替えを制御する。GO信号がディアサートされると、状態機械はIDEL状態４３０に再び遷移する。
WAIT状態４３４において、状態機械３０４はIDLE状態４３０に戻る前に、割り込みが発生したことの確認を待つ。IN TROUTB信号をバス・マスタＩ／Ｆ３００から受け取った場合、CQ_HEADを消去し、状態機械はIDLE状態４３０に戻る。
BRF_HEAD状態４３６では、完了中の現パケットの仮想アドレスを、完了した直前のパケットのVALINKフィールド２６０に書き込む。
【００８０】
ＤＢＥ／ブリッジ
次に図２０を参照すると、図６のＤのＤＢＥ／ブリッジ２１６の更に詳細なブロック図が示されている。これは、一次ＰＣＩバスＩ／Ｆコントローラ５００及び二次ＰＣＩバスＩ／Ｆコントローラ５０２と共に示されているが、代わりに、ホスト・バス１０２を含むその他のバスに、これらのインターフェースを適合化させることも可能である。各ＰＣＩコントローラ５００，５０２は、マスタ及びスレーブＩ／Ｆを内蔵する。上述した同じパケット・アーキテクチャ及びプロトコルが図１１のハードウエアに適用されるので、ここでは繰り返さないことにする。
ＤＢＥ／ブリッジ２１６は、２つの機能的半部分において見ることができる。即ち、一方の半部分は一次ＰＣＩバス１１４と通信し、他方の半部分は二次ＰＣＩバス１１５と通信する。一次ＰＣＩバス半部分は、一次ＰＣＩバスＩ／Ｆコントローラ５００，一次ＦＩＦＯ５０４，一次ダイレクト・メモリ・アクセス（ＤＭＡ）コントローラ５０６、及び一次バースト・バッファ（ＰＢＢ）５０８を内蔵する。二次ＰＣＩバス半部分は、二次ＰＣＩバスＩ／Ｆコントローラ５０２、二次ＦＩＦＯ５１０、二次ＤＭＡコントローラ５１２、及び二次バースト・バッファ（ＳＢＢ）５１４を内蔵する。バースト・バッファ５０８，５１４は、好ましくは、デュアル・ポート型とし、Ｉ／Ｏプロセッサ５１６及びＤＭＡエンジン（５０６又は５１２）又はＰＣＩＩ／Ｆ（５００又は５１２）のいずれか双方の同時アクセスを可能とすることによって、ＰＣＩデバイス１１８、Ｉ／Ｏプロセッサ５１６、及びプロセッサ１００間の潜在的なボトルネックを全て解消する。しかしながら、従来のＤＲＡＭ又はＳＲＡＭのような他の形式のメモリも、バースト・バッファ５０８，５１４を形成するために使用可能である。
【００８１】
また、ＤＢＥ／ブリッジ２１６には、埋め込みＩ／Ｏプロセッサ５１６も含まれており、プロセッサ１００からの介入を受けずに、データを転送するために必要なインテリジェンス(intelligence)を与える。これは、ＤＢＥ／ブリッジ２１６に接続されているＰＣＩデバイス１１８を効果的にプロセッサ１００から切り離し、プロセッサ１００、ＤＢＥ／ブリッジ２１６、及びＰＣＩデバイス１１８間の同時性及び負荷均衡配分を最大に高める。Ｉ／Ｏプロセッサ５１６は、好ましくは、Advanced Micro Devices486プロセッサであるが、いずれの形式のプロセッサでも使用可能である。データ・キャッシュは、Ｉ／Ｏプロセッサ５１６においてはディスエーブルされる。メモリ・コントローラを含ませ、データ・キャッシュのコヒーレンシを確保することも可能である。
一次バースト・バッファ５０８の一部は、Ｉ／Ｏプロセッサ５１６のためのコードを格納

するために割り当てられる。あるいは、別個のリード・オンリ・メモリ（ＲＯＭ）を用いて、コード又はコンフィギュレーション情報を格納することも可能である。メモリ・コントローラは、Ｉ／Ｏプロセッサ５１６に結合され、プロセッサ・サイクルを、以下の分類にデコードする。ＲＯＭリード／ライト、バースト・バッファ（ローカル・メモリ）ヒット、ＦＩＦＯリード／ライト、ＤＭＡコントローラ・コンフィギュレーション、ローカル・レジスタ・リード／ライト、１６進ステータス・ディスプレイ・ライト、ＰＣＩコンフィギュレーション、及びその他（ＰＣＩサイクル）である。あるサイクルがローカル・アドレス空間に該当しない場合、このサイクルをＰＣＩサイクルに変換する。

【００８２】

異なるＰＣＩデバイス１１８と通信するときに、種々のステータス及びコマンド・フォーマットを、デバイス・ドライバ２３０が理解することのできるパケット２５０にカプセル化するのは、Ｉ／Ｏプロセッサ５１６の役割である。

ＰＣＩバスの最大速度でのバースト処理を容易に行うために、一次（ＰＢＢ）及び二次（ＳＢＢ）バースト・バッファ５０４，５０６が設けられており、これらは、一次及び二次ダイレクト・メモリ・アクセス（ＤＭＡ）コントローラ５０８，５１０によってそれぞれ制御される。一次ＤＭＡコントローラ５０８は、一次バースト・バッファ５０４と一次ＰＣＩバス１１４との間でデータを転送する役割を担う。二次ＤＭＡコントローラ５１０は、二次バースト・バッファ５０６と二次ＰＣＩバス１１５との間でデータを転送する役割を担う。

一次及び二次ＦＩＦＯ５０４，５１０は、ＤＭＡコントローラ５０６，５１２に転送を行わせることに加えて、そしてそれに代わる更に効率的な代替案として、データを転送するために設けられている。ＤＭＡコントローラ５０６又は５１２の一方によってデータが転送される場合、一次及び二次ＰＣＩバス１１４，１１５双方を利用する。例えば、二次ＤＭＡコントローラ５１２がデータを二次バースト・バッファ５１４からメモリ１１２に転送しようとした場合、二次ＰＣＩバス１１５だけでなく、一次ＰＣＩバス１１４も使用不可能となる。データは、一次及び二次バースト・バッファ５０８，５１４において、主にＤＢＥ／ブリッジ２１６内に受け取られる。したがって、ＤＭＡコントローラ５０６，５１２にデータを一次及び二次バースト・バッファ５０８，５１４からＤＢＥ／ブリッジ２１６外部に移動させる代わりに、ＤＭＡコントローラ５０６，５１２は、データをＦＩＦＯ５０４，５１０内に移動させることができる。その後、ＰＣＩバス１１４，１１５双方に影響を与えることなく、ＦＩＦＯ５０４，５１０の一方からデータを移動させることができる。例えば、ＰＣＩデバイス１１８がデータを二次バースト・バッファ５１４に書き込む場合、二次ＤＭＡコントローラ５１２は、データを一次ＦＩＦＯ５０４に移動させることができる。一旦これが行われたなら、二次ＰＣＩＩ／Ｆ５０２が他のデータ・ブロックを受け取っている間、一次ＰＣＩＩ／Ｆ５００からデータを転送することができる。

【００８３】

このバースト・バッファ５０８，５１４と一次及び二次ＦＩＦＯ間のデータの移動を容易に行うために、スイッチ５１８及びマルチプレクサ５２０を設け、図２０に示すようにデータを通過させ、データの経路を決定する。加えて、ＤＢＥ／ブリッジ２１６は、ドアベル・レジスタ５２０を含む多数のレジスタを備えている。

また、一次ＰＣＩバス１１４と二次ＰＣＩバス１１５との間に接続されているＩ／Ｏプロセッサ・ブリッジ・バイパス回路（ＩＯＰブリッジ・バイパス）５３０も示されている。好ましくは、これは、IBM 82352PCI-PCIブリッジのような、従来からのＰＣＩ－ＰＣＩ間ブリッジ・デバイスである。バイパス・ブリッジ５３０は、二次ＰＣＩバス１１５に対する調停を行い、二次ＰＣＩバス１１５及びそれに関連する割り込みを、一次ＰＣＩバス１１４から効果的に分離する。

【００８４】

図２１のＡは、ＰＣＩデバイス１１８及びホスト１００間の転送を示すフロー・チャートである。このプロセスは、典型的に、ＰＣＩデバイス１１８がメッセージ／データを二次バースト・バッファ５１４に転記(post)したときに開始する。あるいは、ＰＣＩデバイス

１１８がＤＢＥプロトコルに準拠する場合、ＰＣＩデバイス１１８はデータを直接メモリ１１２に転記することができる。ステップ５５０において、ＰＣＩデバイス１１８が適正なフォーマットでデータを供給可能か否かについて判定を行う。可能であれば、処理はステップ５５２に進み、ＰＣＩデバイス１１８は、適正にフォーマットされたデータ／メッセージを直接メモリ１１２に転記する。この場合、Ｉ／Ｏプロセッサ５１６は、ＰＣＩデバイス１１８に対してアドミニストレータ(administrator)となり、バイパス・ブリッジ５３０を用いて、データ／メッセージをホスト１００に移動させる。可能でない場合、処理はステップ５５４に進み、ＰＣＩデバイス１１８はデータ／メッセージを二次バースト・バッファ５１４に転記する。ステップ５５６において、Ｉ／Ｏプロセッサ５１６は、このデータ／メッセージをＤＢＥと互換性のあるプロトコルに変換し、データ／メッセージをパケット２５０にカプセル化する。データ／メッセージをフォーマットした後、ステップ５５８において、データ／メッセージは、ＤＭＡ転送によってホスト１００に送出するか、あるいは一次ＦＩＦＯ５５４に転記することができる。データ／メッセージを一次ＦＩＦＯ５５４に転記することによって、データの連続的な順序が保証される。

【００８５】

図２１の（Ｂ）に示すホスト－ＰＣＩデバイス１１８間の転送では、データ／メッセージをＰＣＩデバイス１１８に送るプロセスは、 ＰＣＩデバイス－ホスト間転送（図２１のＡ）と同様である。ステップ５６０において、ホスト１００がＰＣＩデバイス１１８と直接通信可能か否かについて判定を行う。可能であれば、ステップ５６２において、ホストはデータ／メッセージを直接バイパス・ブリッジ５３０を通じてＰＣＩデバイス１１８に送る。可能でない場合、ステップ５６４において、ホスト１００は要求をメモリ１１２に書き込み、対応するドアベル５２０を鳴らす。ドアベル５２０は、Ｉ／Ｏプロセッサ５１６に割り込みを発生させる。ドアベルが鳴らされた場合、ステップ５６６においてＩ／Ｏプロセッサ５１６は、割り込み源を判定し、その割り込みを処理し、ドアベルをクリアする。Ｉ／Ｏプロセッサは、請求パケット及び非請求パケットを、ＤＢＥデバイス２０６と同様に取り扱う。

以上、ホスト１００とＩ／Ｏデバイス２０８との間でパケット２５０を移動するためのパケット・プロトコル及び複数の選択可能なハードウエア・エンジンを説明した。

【００８６】

本発明の上述の開示及び説明は、本発明の例示及び説明のためのものであり、サイズ、形状、材料、構成部品、回路素子、配線接続及びコンタクト、ならびに図示の回路及び構造や、動作方法の詳細において、本発明の精神から逸脱することなく、様々な変更が可能である。

【図面の簡単な説明】

【図１】本発明によるコンピュータ・システムＣのブロック図である。

【図２】Ａは、好適な実施形態による本発明のシステム・アーキテクチャを示すブロック図である。

Ｂは、好適な実施形態による本発明のソフトウエア・システム・アーキテクチャを示すブロック図である。

【図３】好適な実施形態によるパケット・アーキテクチャを示すブロック図である。

【図４】好適な実施形態によるパケット・キューを示すブロック図である。

【図５】好適な実施形態によるＤＢＥデバイスのハードウエア・レジスタを示すブロック図である。

【図６】Ａは、PCIバスに沿ったＤＢＥデバイスの第１の配置を示すブロック図である。

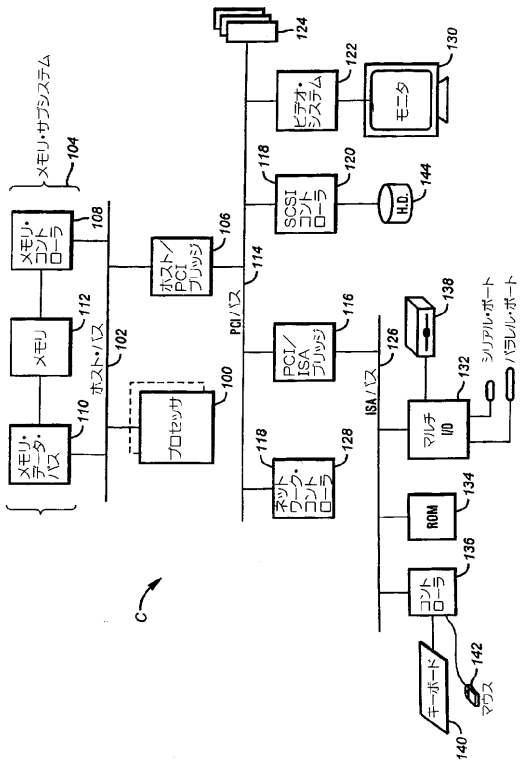
Ｂは、図１のホスト／PCIブリッジにおけるＤＢＥデバイスの第２の実施形態を示すブロック図である。

Ｃは、ブリッジ／メモリ・コントローラ・デバイスにおけるＤＢＥデバイスの第３の実施形態を示すブロック図である。

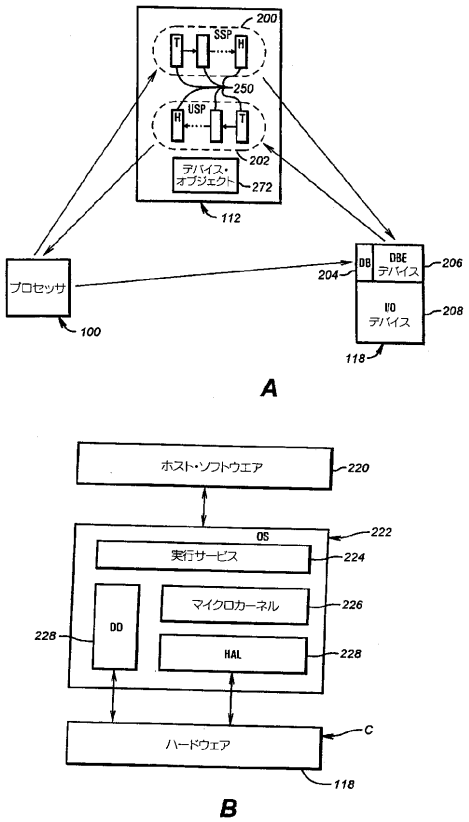
Ｄは、PCI／PCIブリッジにおけるＤＢＥデバイスの第４の実施形態を示すブロック図である。

【図７】好適な実施形態によるＤＢＥデバイスの更なる詳細を示すブロック図である。
【図８】好適な実施形態によるＤＢＥデバイスの要求キュー・フロント・エンド状態機械及びポステッド・キュー・フロント・エンド状態機械のブロック図である。
【図９】要求キュー・フロント・エンド状態機械の状態遷移図、状態遷移条件、及び出力を示す図である。
【図１０】ポステド・キュー・フロント・エンド状態機械の状態遷移図、状態遷移条件、及び出力を示す図である。
【図１１】好適な実施形態によるＤＢＥデバイスの要求キュー・バック・エンド状態機械及びポステッド・キュー・バック・エンド状態機械のブロック図である。
【図１２】要求キュー・フロント・エンド状態機械の状態遷移図である。
【図１３】要求キュー・フロント・エンド状態機械の状態遷移条件、及び出力を示す図である。
【図１４】要求キュー・フロント・エンド状態機械の出力を示す図である。
【図１５】ポステド・キュー・フロント・エンド状態機械の状態遷移図、状態遷移条件、及び出力を示す図である。
【図１６】好適な実施形態によるＤＢＥデバイスの完了キュー状態機械のブロック図である。
【図１７】好適な実施形態による図７のバス・マスタ・インターフェースのプロセスを示すフロー・チャートである。
【図１８】完了キュー状態機械の状態遷移図、及び状態遷移条件を示す図である。
【図１９】完了キュー状態機械の出力を示す図である。
【図２０】ＤＢＥデバイスの代替実施例を示すブロック図である。
【図２１】デバイス－ホスト間転送及びホスト－デバイス間転送を示すフロー・チャートである。

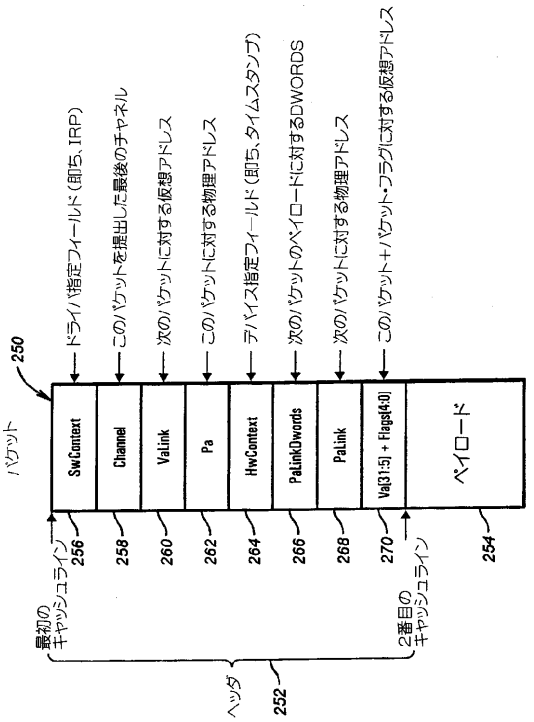
【図１】



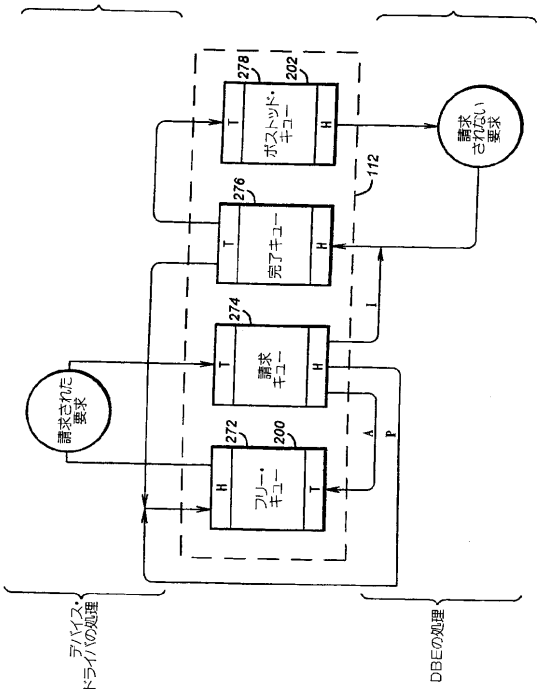
【図２】



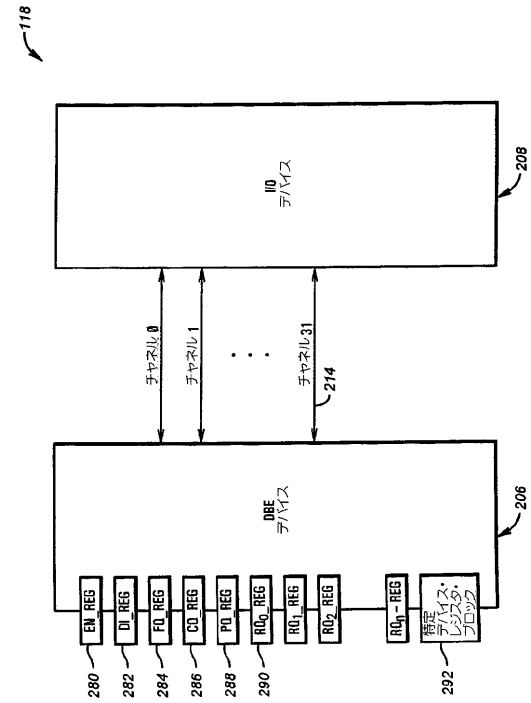
【図３】


250
パケット
ドライバ指定フィールド（即ち、IRP）
このパケットを提出した最後のチャネル
次のパケットに対する仮想アドレス
このパケットに対する物理アドレス
デバイス指定フィールド（即ち、タイムスタンプ）
次のパケットのペイロードに対するDWORDS
次のパケットに対する物理アドレス
このパケット＋パケット・フラグに対する仮想アドレス
SwContext
Channel
ValLink
Pa
HwContext
PaLinkDwords
PaLink
Val[31:5] + Flags[4:0]
ペイロード
256
258
260
262
264
266
268
270
254
最初のキャッシュライン
2番目のキャッシュライン
ヘッダ
252

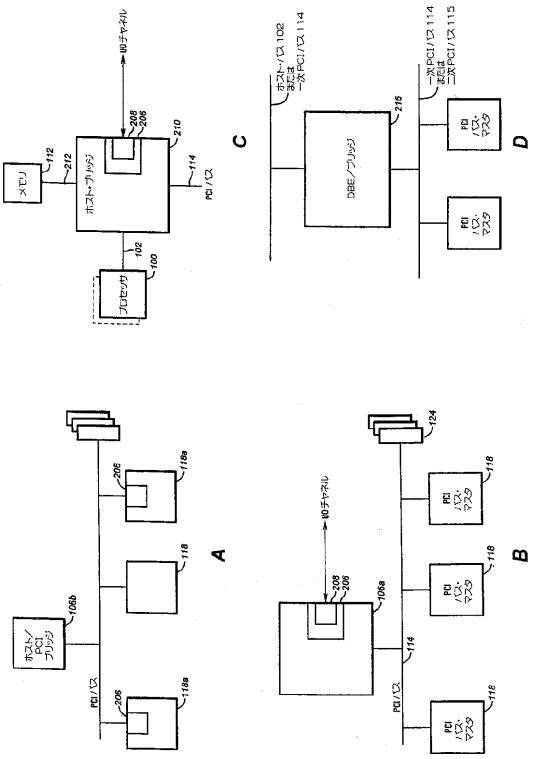
【図４】


278
202
ポストド・キュー
276
完了キュー
274
請求キュー
272
200
フリー・キュー
請求された要求
請求されない要求
112
デバイス・ドライバの処理
DBEの処理

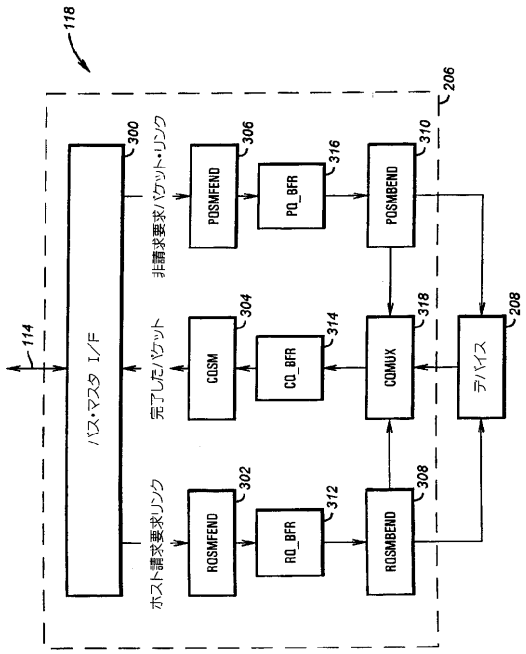
【図５】


118
IIO デバイス
208
チャネル 0
チャネル 1
チャネル 31
214
DBE デバイス
206
EN_REG
DI_REG
FD_REG
CD_REG
PD_REG
RQ0_REG
RQ1_REG
RQ2_REG
RQn_REG
特定デバイス・ロジック・ブロック
280
282
284
286
288
290
292

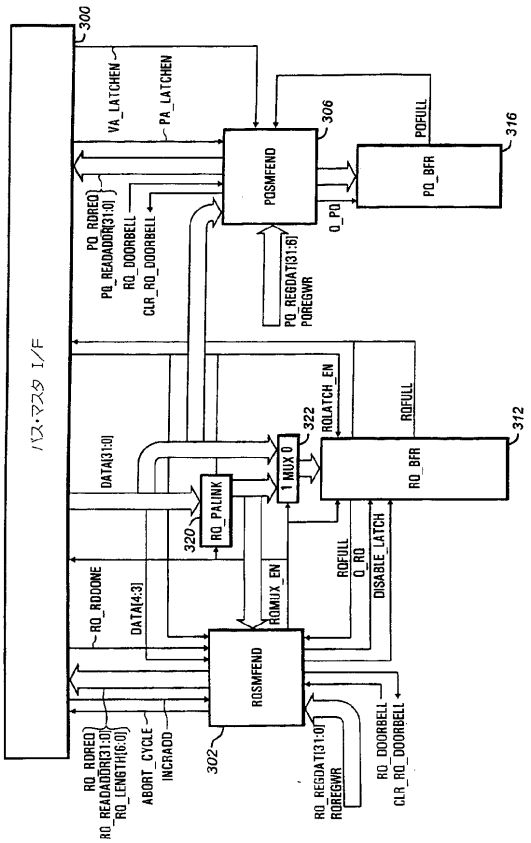
【図６】


A
B
C
D
IIOチャネル
メモリ
ホスト・ブリッジ
プロセッサ
PCI バス
DBE ブリッジ
PCI バス・マスタ
ホスト/PCI ブリッジ
ホスト・バス 102
第1次PCI バス 114
第2次PCI バス 115

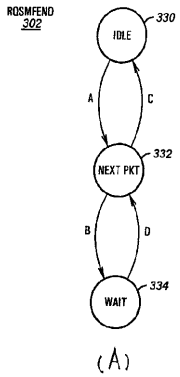
【図７】



【図８】



【図９】



(A)

(B)

ROSMFEND 状態遷移条件

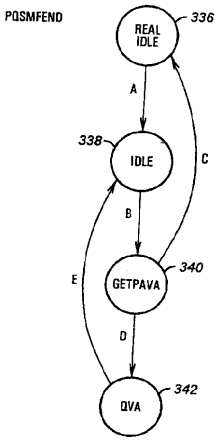
A: IDLE * RQ_DOORBELL;
B: NEXTPKT * RQ_RDDONE * RQ_PALINK[31:5]! = 0h * ROFULL;
C: NEXTPKT * RQ_RDDONE * RQ_PALINK[31:5] = 0h;
D: WAIT * !RQ_FULL;

(C)

ROSMFEND 出力

```
ABORT_CYCLE = 1 if    NEXTPKT * RQLATCH_EN * CHECKVA * (RDDATA[4] + RDDATA[3]);
ABORT_CYCLE = 0 if    NEXTPKT * RQ_RDDONE;
CLR_RQ_DOORBELL = 1 if      IDLE * RQ_DOORBELL
DISABLE_LATCH = 1 if  IDLE +
                NEXTPKT * RQLATCH_EN * CHECKVA * RDDATA[4](INCLUDING rddata4 + RDDATA[3])
DISABLE_LATCH = 0 if  NEXTPKT * RQ_RDDONE;
O_RQ = 0 if   IDLE + WAIT;
O_RQ = 1 if   NEXTPKT * RQLATCH_EN * !DISABLE_LATCH;
RQ_LEN [9:0] = 2h if IDLE;
RQ_LEN [9:0] = NEXTPKT * RQ_RDDONE * RQ_PALINK [31:5] ! = 0h;
RQ_MUX_EN = 0 if NEXTPKT * RQLATCH_EN * RQMUX_EN;
RQ_RDREQ = 1 if   (IDLE * RQ_DOORBELL) +
                (WAIT * !RQFULL)
RQ_RDREQ = 1 if   NEXTPKT * RQ_RDDONE * RQ_PALINK[31:5]! = 0h * !ROFULL;
RQ_RDREQ = 0 if   NEXTPKT * RQLATCH_EN * RQMUX_EN;
RQ_RDREQ = 0 if   NEXTPKT * RQLATCH_EN * RQMUX_EN;
RQ_READADDR_B[31:5] <= RQ_REG [31:5] if     IDLE;
RQ_READADDR_B[31:5] <= RQ_READADDR_B [31:0] + 4h if     NEXTPKT * INCRADD;
RQ_READADDR_B[31:5] <= RQ_PALINK [31:5] if     NEXTPKT * RQ_RDDONE * RQ_PALINK[31:5]! = 0h;
RQ_READADDR_B[31:5] = RQ_PALINK[31:5] if     NEXTPKT * RQ_RDDONE;
RQ_REG[31:5] <= RQREGDAT [31:5] if IDLE * RQREGWR;
RQ_REG[31:5] <= RQ_PALINK [31:5] if WAIT * !ROFULL;
RQ_REQ[31:5] = RQ_PALINK[31:5] if     NEXTPKT * RQ_RDDONE * RQ_PALINK[31:5]! = 0h * !RQFULL;
RQPALINK[31:5] <= RDDDRDDAT[31:5] if     NEXTPKT * RQLATCH_EN * RQMUX_EN;
```

【図１０】



(A)

(B)

POSMFEND 状態遷移条件

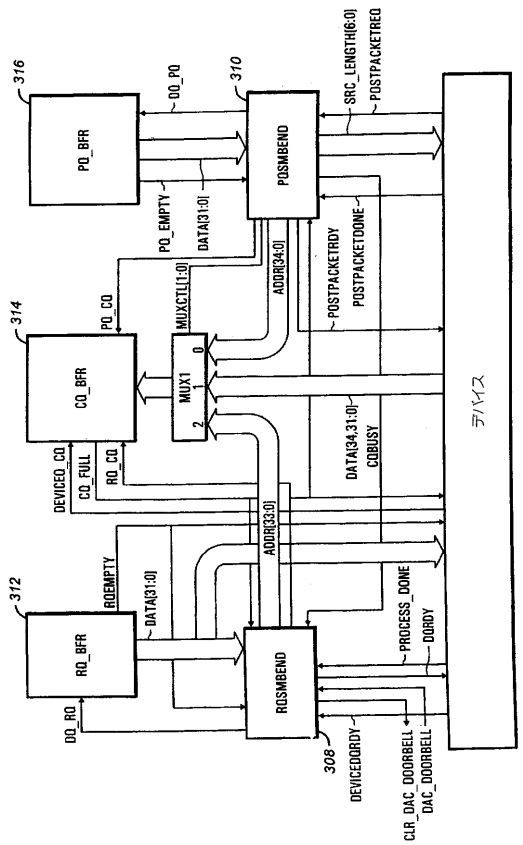
A: REALIDLE * PQ_DOORBELL;
B: don't care;
C: GETPAVA * VA_LATCHEN * PQ_PALINK[31:5] = 0h;
D: GETPAVA * VA_LATCHEN * !PQ_PALINK[31:5] = 0h;
E: don't care;

(C)

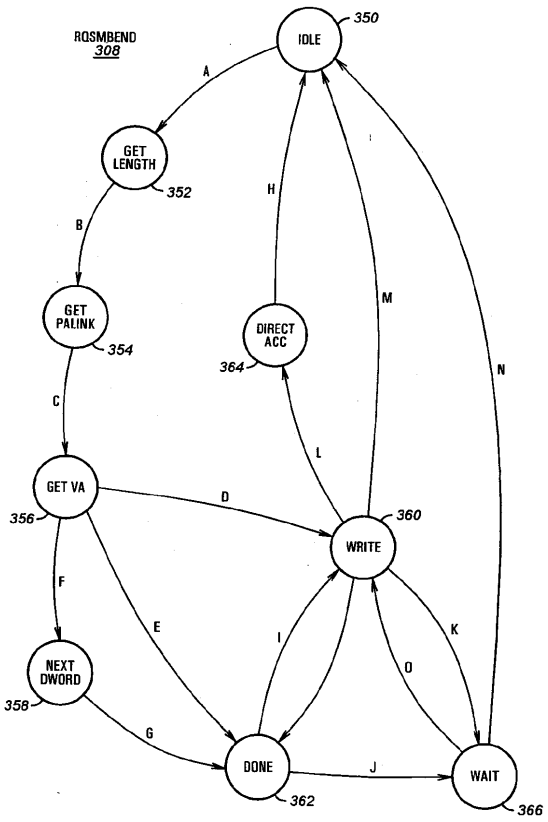
POSMFEND 出力

```
PQ_REG[31:5] <= PQ_PALINK[31:5] if GETPAVA * VA_LATCHEN * !(PQ_PALINK[31:5] = 0h);
PQ_PALINK[31:5] <= RDDATA[31:5 if GETPAVA * VA_LATCHEN * !(PQ_PALINK[31:5] = 0h);
O_PQ <= 0 if      REALIDLE + IDLE + GETPAVA;
O_PQ <= 1 if      GETPAVA * VA_LATCHEN * !(PQ_PALINK[31:5] = 0h);
VALINK <= 1 if    GETPAVA * VA_LATCHEN * !(PQ_PALINK[31:5] = 0h);
```

【図１１】



【図１２】



【図１３】

RQSMBEND 状態遷移条件

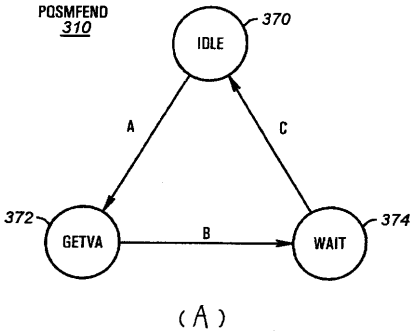
A: IDLE * ! RQ RQEMPTY;
B: GETLENGTH * ! RQEMPTY;
C: GETPALINK * ! RQEMPTY;
D: GETVA * (RQ_BFR[4] + (RQ_BFR[3] * LENGTH[9:0] = 0h));
E: GETVA * ! LENGTH [9:0] = 0h; (RQ_BFR[4] + (RQ_BFR[3] * LENGTH[9:0] = 0h));
F: GETVA * (RQ_BFR[4] + (RQ_BFR[3] * LENGTH[9:0] = 0h));
G: NEXTDWRD * DEVICEODRDY * LENGTH[9:0] = 0h;
H: DIRECTACC * DAC_DOORBELL * ! CLR_DAC_DOORBELL;
I: DONE * RQ_VALINK[2:1] = 1h;
J: !DONE * RQ_VALINK[2:1] = 2h +;
!DONE * RQ_VALINK[2:1] = 3h;
K: [WRITE * !COFULL * !CDBUSY * RQ_VALINK[2:1]=DBE_PKT_SUBMIT_A * AGAIN * (RQ_VALINK[4] + !RQ_VALINK[3])] +
[WRITE * !COFULL * !CDBUSY * RQ_VALINK[2:1]=DBE_PKT_SUBMIT_1 * AGAIN * !RQ_VALINK[4] * !RQ_VALINK[3]];
L: [WRITE * !COFULL * !CDBUSY * RQ_VALINK[2:1]=DBE_PKT_SUBMIT_A * AGAIN !(RQ_VALINK[4] + !RQ_VALINK[3])] +
[WRITE * !COFULL * !CDBUSY * RQ_VALINK[2:1]=DBE_PKT_SUBMIT_P * AGAIN * !(RQ_VALINK[4] + !RQ_VALINK[3])];
M: [WRITE * !COFULL * !CDBUSY * RQ_VALINK[2:1]=DBE_PKT_SUBMIT_P * AGAIN * (RQ_VALINK[4] + !RQ_VALINK[3])] +
[WRITE * !COFULL * !CDBUSY * RQ_VALINK[2:1]=DBE_PKT_SUBMIT_1 * AGAIN * RQ_VALINK[4];
N: WAIT * PROCESS_DONE * RQ_VALINK[2:1]=DBE_PKT_SUBMIT_A;
O: WAIT * PROCESS_DONE * !RQ_VALINK[2:1]=DBE_PKT_SUBMIT_A;

【図１４】

RQSMBEND出力

CLR_DAC_DOORBELL < = 0 if IDLE + DIRECTACC;
CLR_DAC_DOORBELL < = 1 if [WRITE * !COFULL & !CDBUSY * RQ_VALINK[2:1]=DBE_PKT_SUBMIT_A * AGAIN * !(RQ_VALINK[4] + !RQ_VALINK[3])] +
[WRITE * !COFULL & !CDBUSY * RQ_VALINK[2:1]=DBE_PKT_SUBMIT_P * AGAIN * !(RQ_VALINK[4] + !RQ_VALINK[3])] +
WRITE * !COFULL & !CDBUSY * RQ_VALINK[2:1]=DBE_PKT_SUBMIT_1 * AGAIN * !RQ_VALINK[4] * RQ_VALINK[3]);
DO_RQ < = 1 if IDLE * !RQEMPTY +
GETLENGTH * !RQEMPTY +
GETPALINK * !RQEMPTY;
otherwise DO_RQ < = 0;
DORDY < = 1 if GETVA * !(RQ_BFR[4] + (RQ_BFR[3] * LENGTH[9:0] = 0h)) * !LENGTH[9:0] = 0h;
DORDY < = 0 if NEXTDWRD * DEVICEODRDY * LENGTH[9:0] = 1h;
RQ_CO < = 1 if WRITE * !COFULL * !CDBUSY;
RQ_CO < = 0 if IDLE + DIRECTACC + WAIT;

【図１５】



(A)

POSMBEND 状態遷移条件

A: IDLE * CQBUSY;
B: GETVA * !PQEMPTY;
C: WAIT * !CQFULL * !POSTPACKETRDY;

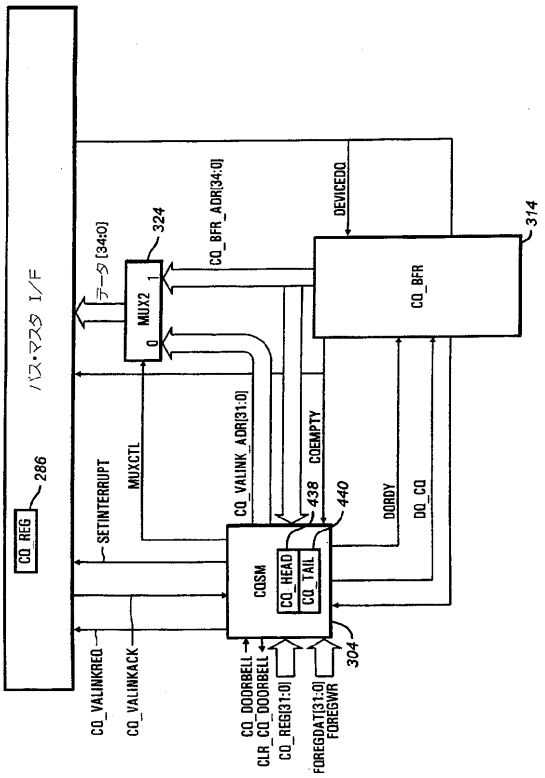
(B)

POSMBEND 出力

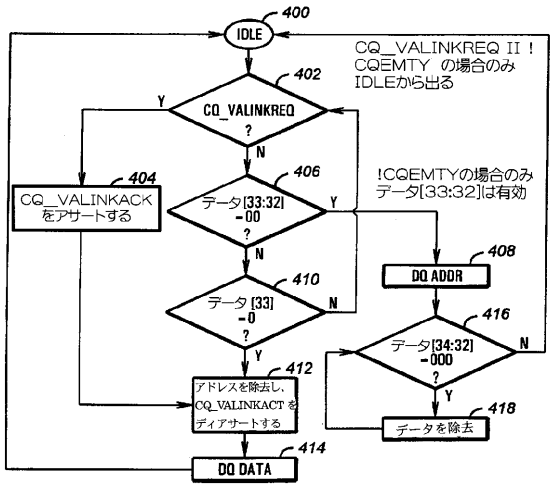
```
CQBUSY <= 0 if   WAIT * !CQFULL * !POSTPACKETRDY;
CQBUSY <= 1 if   IDLE * POSTPACKETREQ * !PQEMPTY * !MUXVA;
DQ_CQ <= 1 if    (IDLE * CQBUSY) +
                 (WAIT * POSTPACKETDONE * !CQFULL * POSTPACKETRDY) +
                 (WAIT * !CQFULL * !POSTPACKETRDY);
DQ_CQ <= 0 if    (IDLE * !CQBUSY) + GETVA + WAIT;
DQ_PQ <= 1 if    IDLE * (MUXVA + CQBUSY);
otherwise DQ_PQ = 0;
POSTPACKETRDY <= 0 if   WAIT * POSTPACKETDONE * !CQFULL * POSTPACKETRDY;
POSTPACKETRDY <= 1 if   GETVA * !PQEMPTY;
```

(C)

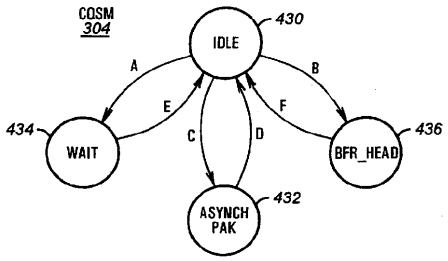
【図１６】



【図１７】



【図１８】



(A)

CQSM状態遷移条件

A: IDLE * CQ_DOORBELL * (CQ_HEAD[31:2] != 0h) * CQ_VALINKACK;
B: IDLE * !(CQ_DOORBELL * (CQ_HEAD[31:2] != 0h)) * GO * !CQ_BFR_ADR[34] * (CQ_BFR_ADR[33:32]=3h) * (CQ_HEAD[31:2]=0);
C: (IDLE * !(CQ_DOORBELL * (CQ_HEAD[31:2] != 0h)) * GO * !CQ_BFR_ADR[34] * (CQ_BFR_ADR[33:32]=3h) * !(CQ_HEAD[31:2]=0) * CQ_VALINKACK) + (IDLE * !(CQ_DOORBELL * (CQ_HEAD[31:2] != 0h)) * GO * !CQ_BFR_ADR[34] * (CQ_BFR_ADR[33:32]=2h) * CQ_VALINKACK);
D: ASYNCH * !GO;
E: WAIT * INTROUTB;
F: don't care;

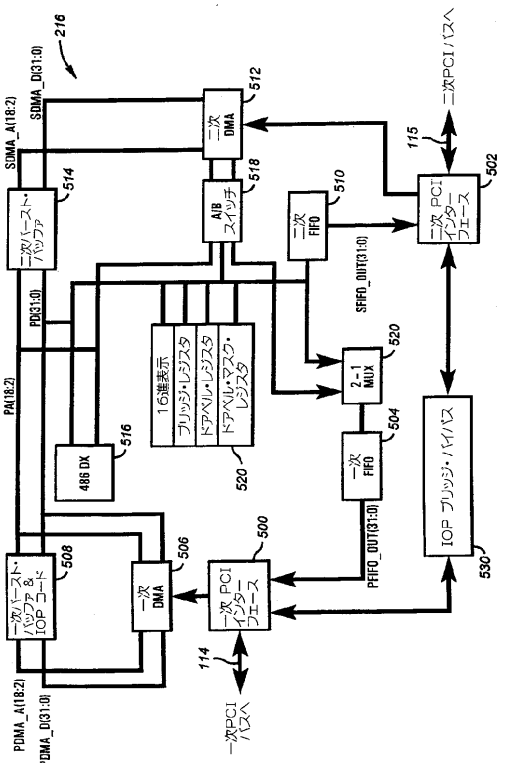
(B)

【図１９】

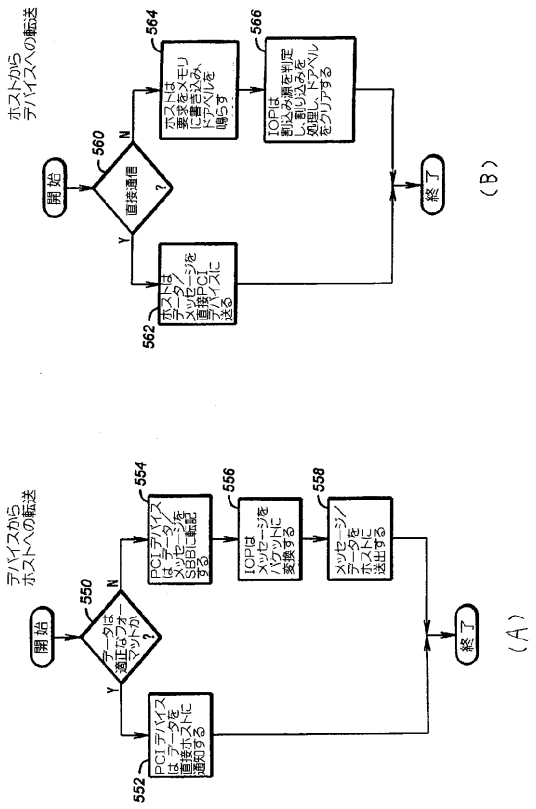
```
CQSM 出力

CLR_CQ_DOORBELL <= 1 if   IDLE * CQ_DOORBELL * !CQ_HEAD[31:2] = 0h) * CQ_VALINKACK;
CLR_CQ_DOORBELL <= 0 if   WAIT * INTROUTB;
CQ_HEAD[31:2] <= CQ_BFR_ADR[31:2] if   BFR_HEAD;
CQ_HEAD[31:2] <= 0h if   WAIT * INTROUB;
CQ_TAIL[31:2] <= CQ_BFR_ADR[31:2] if   [(IDLE * !(CQ_DOORBELL * (CQ_HEAD[31:2] = 0h)) * GO * !CQ_BFR_ADR[34] *
                                        (CQ_BFR_ADR[33:32] = 3h) * CQ_HEAD[31:2] = 0h) +
                                       [(IDLE * !(CQ_DOORBELL * (CQ_HEAD[31:2] = 0h)) * GO * !CQ_BFR_ADR[34] *
                                        (CQ_BFR_ADR[33:32] = 3h) * !CQ_HEAD[31:2] = 0h * CQ_VALINKACK];
CQ_VALINKREQ <= 1 if   [(IDLE * CQ_DOORBELL * (CQ_HEAD[31:2] = 0h)) +
                       [(IDLE * !(CQ_DOORBELL * (CQ_HEAD[31:2] = 0h)) * GO * !CQ_BFR_ADR[34] * (CQ_BFR_ADR[33:32] = 3h) *
                       !(CQ_HEAD[31:2] = 0h)) +
                       [(IDLE * !(CQ_DOORBELL * (CQ_HEAD[31:2] = 0h)) * GO * !CQ_BFR_ADR[34] * (CQ_BFR_ADR[33:32] = 2h)];
CQ_VALINKREQ <= 0 if   [(IDLE * CQ_DOORBELL * (CQ_HEAD[31:2] = 0h)) * GO * !CQ_BFR_ADR[34] * (CQ_BFR_ADR[33:32] = 2h) +
                       [(IDLE * !(CQ_DOORBELL * (CQ_HEAD[31:2] = 0h)) * CQ_VALINKACK] +
                       !CQ_HEAD[31:2] = 0h * CQ_VALINKACK] +
                       [(IDLE * !(CQ_DOORBELL * (CQ_HEAD[31:2] = 0h)) * GO * !CQ_BFR_ADR[34] * (CQ_BFR_ADR[33:32] = 3h) *
                       CQ_VALINKACK];
D_AGAIN <= 0 if   IDLE;
D_AGAIN <= 1 if   (ASYNCH * DEVICEDQ * !AGAIN) +
                  (WAIT * DEVICEDQ);
D_MUXDAT = 0 if   IDLE *
                  IDLE * CQ_DOORBELL * (CQ_HEAD[31:2] = 0h * CQ_VALINKACK) *
                  IDLE * ! (CQ_DOORBELL * (CQ_HEAD[31:2] ! = 0h)
DQ_CQ <= 1 if     (IDLE * !(CQ_DOORBELL * (CQ_HEAD[31:2] = 0h)) *
                  GO * !CQ_BFR_ADR[34] * (CQ_BFR_ADR[33:32] = 3h) * CQ_HEAD[31:2] = 0h) +
                  (WAIT * INTROUTB);
Otherwise DQ_CQ <= 0;
DQRDY <= 0 if     IDLE * CQ_DOORBELL * (CQ_HEAD[31:2] = 0h) * CQ_VALINKACK;
FQ_REQ[31:5] <= FOREQDAT[31:5] if   [IDLE * FOREQWR];
FQ_REQ[31:5] <= CQ_BFR[31:5] if     [(IDLE * !(CQ_DOORBELL * (CQ_HEAD[31:2] = 0h)) * GO * !CQ_BFR_ADR[34] *
                                    CQ_BFR_ADR[33:32] = 2h * CQ_VALINKACK];
MUXCTL <= 0 if    [(IDLE * CQ_DOORBELL * (CQ_HEAD[31:2] = 0h)) * CQ_VALINKACK] +
                  [(IDLE * !(CQ_DOORBELL * (CQ_HEAD[31:2] = 0h)) * GO * !CQ_BFR_ADR[34] * (CQ_BFR_ADR[33:32] = 3h) *
                  !CQ_HEAD[31:2] = 0h * CQ_VALINKACK] +
                  [(IDLE * !(CQ_DOORBELL * (CQ_HEAD[31:2] = 0h)) * GO * !CQ_BFR_ADR[34] *
                  CQ_BFR_ADR[33:32] = 2h * CQ_VALINKACK];
MUXCTL <= 1 if    (ASYNCH * DEVICEDQ * !AGAIN) +
                  (WAIT * INTROUTB);
SETINTERRUPT <= 1 if   IDLE * CQ_DOORBELL * (CQ_HEAD[31:2] = 0h);
SETINTERRUPT <= 0 if   WAIT * INTROUTB;
```

【図２０】



【図２１】



フロントページの続き

(74)代理人 100071124
弁理士 今井 庄亮
(74)代理人 100076691
弁理士 増井 忠弐
(74)代理人 100075236
弁理士 栗田 忠彦
(74)代理人 100075270
弁理士 小林 泰
(74)代理人 100096068
弁理士 大塚 住江
(72)発明者 マイケル・ピー・モリアーティ
アメリカ合衆国テキサス州７７３７９，スプリング，サニー・ポイント・ドライブ ８８３０
(72)発明者 トーマス・ジェイ・ボノラ
アメリカ合衆国テキサス州７７３７５，トンボール，クレアーショルム １２３３１
(72)発明者 ブライアン・ティー・パーセル
アメリカ合衆国テキサス州７７３７５，トンボール，エガンビル・サークル １８１３０

審査官 石井 茂和

(56)参考文献 特開平１０－０４９４８１（ＪＰ，Ａ）

(58)調査した分野(Int.Cl.，ＤＢ名)
G06F 13/14
G06F 13/36
WPI(DIALOG)