(19)日本国特許庁(JP)　(12)公開特許公報(A)　(11)特許出願公開番号
特開2015-64676
(P2015-64676A)
(43)公開日　平成27年4月9日(2015.4.9)

(51)Int.Cl.			F I			テーマコード（参考）
G06F	1/32	(2006.01)	G06F	1/00	332Z	5B011
G06F	1/04	(2006.01)	G06F	1/04	301C	5B079

審査請求 未請求　請求項の数 32　ＯＬ　（全 54 頁）

(21)出願番号　特願2013-197394 (P2013-197394)
(22)出願日　平成25年9月24日 (2013.9.24)

（特許庁注：以下のものは登録商標）
１．ＢＬＵＥＴＯＯＴＨ
２．ＷＩＮＤＯＷＳ
３．ＡＮＤＲＯＩＤ

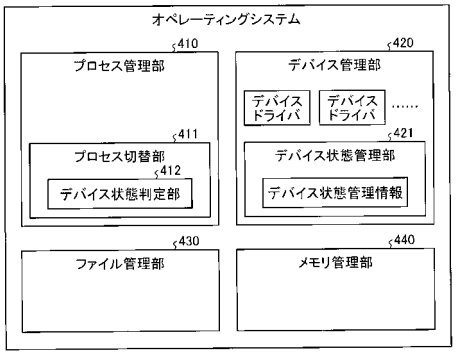
(71)出願人　000003078
株式会社東芝
東京都港区芝浦一丁目１番１号
(74)代理人　100089118
弁理士　酒井　宏明
(74)代理人　100112656
弁理士　宮田　英毅
(72)発明者　瀬川　淳一
東京都港区芝浦一丁目１番１号　株式会社東芝内
(72)発明者　金井　達徳
東京都港区芝浦一丁目１番１号　株式会社東芝内

最終頁に続く

(54)【発明の名称】情報処理装置、半導体装置、情報処理方法およびプログラム

(57)【要約】

【課題】性能を損なうことなく省電力化を適切に行うことが可能な情報処理装置、半導体装置、情報処理方法およびプログラムを提供する。
【解決手段】実施形態の情報処理装置は、記憶装置と、１以上の周辺機器と、命令を実行する第１の状態、または、割込みを待つ第２の状態に遷移可能なプロセッサと、を備える情報処理装置であって、状態制御部を有する。状態制御部は、プロセッサが第２の状態に遷移するときに、少なくとも１つの周辺機器と記憶装置との間でデータ転送のための処理が行われている場合は、プロセッサが第１の状態のときよりも消費電力が小さい第３の状態に情報処理装置を遷移させる制御を行う。一方、何れの周辺機器と記憶装置との間でもデータ転送のための処理が行われていない場合は、状態制御部は、第３の状態よりも消費電力が小さい第４の状態に情報処理装置を遷移させる制御を行う。
【選択図】図１１



【特許請求の範囲】
【請求項１】
記憶装置と、
１以上の周辺機器と、
命令を実行する第１の状態、または、割込みを待つ第２の状態に遷移可能なプロセッサと、を備える情報処理装置であって、
前記プロセッサが前記第２の状態に遷移するときに、少なくとも１つの前記周辺機器と前記記憶装置との間でデータ転送のための処理が行われている場合は、前記プロセッサが前記第１の状態のときよりも消費電力が小さい第３の状態に前記情報処理装置を遷移させる一方、何れの前記周辺機器と前記記憶装置との間でも前記データ転送のための処理が行われていない場合は、前記第３の状態よりも消費電力が小さい第４の状態に前記情報処理装置を遷移させる制御を行う状態制御部を有する、
情報処理装置。
【請求項２】
前記状態制御部は、前記第４の状態では、前記プロセッサへ供給する電圧を下げる制御を行う、
請求項１に記載の情報処理装置。
【請求項３】
前記状態制御部は、前記第４の状態では、前記プロセッサが前記第１の状態のときに用いられるクロックを示す高周波クロックを生成する高周波発振器の発振を停止させる制御を行う、
請求項１に記載の情報処理装置。
【請求項４】
前記状態制御部は、前記第４の状態では、前記プロセッサが前記第１の状態のときに用いられるクロックを示す高周波クロックを生成する高周波発振器への電力供給を停止させる制御を行う、
請求項１に記載の情報処理装置。
【請求項５】
前記高周波発振器は、シリコンオシレータである、
請求項３または請求項４に記載の情報処理装置。
【請求項６】
前記記憶装置は揮発性メモリであり、
前記状態制御部は、前記第４の状態では、前記記憶装置を、前記記憶装置に記憶されたデータを保持可能であって、データの読み出しまたは書き込みを行うことはできないセルフリフレッシュ状態に遷移させる制御を行う、
請求項１に記載の情報処理装置。
【請求項７】
前記記憶装置は不揮発性メモリであり、
前記状態制御部は、前記第４の状態では、前記記憶装置への電力供給を停止させる制御を行う、
請求項１に記載の情報処理装置。
【請求項８】
前記プロセッサはキャッシュメモリを備え、
前記状態制御部は、前記第４の状態に前記情報処理装置を遷移させる場合は、前記記憶装置への書き込みが行われていない前記キャッシュメモリ内のデータを前記記憶装置へ書き出す処理を行った後、前記キャッシュメモリへの電力供給を停止させる制御を行う、
請求項１に記載の情報処理装置。
【請求項９】
前記プロセッサはキャッシュメモリを備え、
前記状態制御部は、前記第４の状態では、前記キャッシュメモリに供給する電圧を、前

記プロセッサが前記第１の状態のときよりも下げる制御を行う、
請求項１に記載の情報処理装置。
【請求項１０】
前記周辺機器は、
ストレージデバイス、コミュニケーションデバイス、ディスプレイデバイス、イメージキャプチャデバイスのうちの何れかである、
請求項１に記載の情報処理装置。
【請求項１１】
前記周辺機器ごとに、前記周辺機器を制御するデバイス制御部をさらに備え、
前記デバイス制御部は、前記プロセッサからの入出力要求を受けると、前記データ転送のための処理を開始し、前記データ転送のための処理が完了すれば前記プロセッサに対して割込みをかけて通知する、
請求項１に記載の情報処理装置。
【請求項１２】
記憶装置と、
１以上の周辺機器と、
命令を実行する第１の状態、または、割込みを待つ第２の状態に遷移可能なプロセッサと、を備える情報処理装置が実行する情報処理方法であって、
前記プロセッサが前記第２の状態に遷移するときに、少なくとも１つの前記周辺機器と前記記憶装置との間でデータ転送のための処理が行われている場合は、前記プロセッサが前記第１の状態のときよりも消費電力が小さい第３の状態に前記情報処理装置を遷移させる一方、何れの前記周辺機器と前記記憶装置との間でも前記データ転送のための処理が行われていない場合は、前記第３の状態よりも消費電力が小さい第４の状態に前記情報処理装置を遷移させる制御を行う状態制御ステップを含む、
情報処理方法。
【請求項１３】
記憶装置と、
１以上の周辺機器と、
命令を実行する第１の状態、または、割込みを待つ第２の状態に遷移可能なプロセッサと、を備える情報処理装置に搭載されたコンピュータに、
前記プロセッサが前記第２の状態に遷移するときに、少なくとも１つの前記周辺機器と前記記憶装置との間でデータ転送のための処理が行われている場合は、前記プロセッサが前記第１の状態のときよりも消費電力が小さい第３の状態に前記情報処理装置を遷移させる一方、何れの前記周辺機器と前記記憶装置との間でも前記データ転送のための処理が行われていない場合は、前記第３の状態よりも消費電力が小さい第４の状態に前記情報処理装置を遷移させる制御を行う状態制御ステップを実行させるためのプログラム。
【請求項１４】
命令を実行する第１の状態、または、割込みを待つ第２の状態に遷移可能なプロセッサを備え、１以上の周辺機器が接続された半導体装置であって、
前記プロセッサが前記第２の状態に遷移するときに、少なくとも１つの前記周辺機器と前記半導体装置との間でデータ転送のための処理が行われている場合は、前記プロセッサが前記第１の状態のときよりも消費電力が小さい第３の状態に前記半導体装置を遷移させる一方、何れの前記周辺機器と前記半導体装置との間でも前記データ転送のための処理が行われていない場合は、前記第３の状態よりも消費電力が小さい第４の状態に前記半導体装置を遷移させる制御を行う状態制御部を有する、
半導体装置。
【請求項１５】
命令を実行する第１の状態、または、割込みを待つ第２の状態に遷移可能なプロセッサを備え、１以上の周辺機器が接続された半導体装置が実行する情報処理方法であって、
前記プロセッサが前記第２の状態に遷移するときに、少なくとも１つの前記周辺機器と

前記半導体装置との間でデータ転送のための処理が行われている場合は、前記プロセッサが前記第１の状態のときよりも消費電力が小さい第３の状態に前記半導体装置を遷移させる一方、何れの前記周辺機器と前記半導体装置との間でも前記データ転送のための処理が行われていない場合は、前記第３の状態よりも消費電力が小さい第４の状態に前記半導体装置を遷移させる制御を行う状態制御ステップを含む、
情報処理方法。
【請求項１６】
命令を実行する第１の状態、または、割込みを待つ第２の状態に遷移可能なプロセッサを備え、１以上の周辺機器が接続された半導体装置に搭載されたコンピュータに、
前記プロセッサが前記第２の状態に遷移するときに、少なくとも１つの前記周辺機器と前記半導体装置との間でデータ転送のための処理が行われている場合は、前記プロセッサが前記第１の状態のときよりも消費電力が小さい第３の状態に前記半導体装置を遷移させる一方、何れの前記周辺機器と前記半導体装置との間でも前記データ転送のための処理が行われていない場合は、前記第３の状態よりも消費電力が小さい第４の状態に前記半導体装置を遷移させる制御を行う状態制御ステップを実行させるためのプログラム。
【請求項１７】
記憶装置と、
１以上の周辺機器と、
命令を実行する第１の状態、または、割込みを待つ第２の状態に遷移可能なプロセッサと、を備える情報処理装置であって、
前記プロセッサが前記第２の状態に遷移するときに、何れの前記周辺機器と前記記憶装置との間でもデータ転送のための処理が行われていない場合は、前記情報処理装置の消費電力を下げる制御を行う状態制御部を有する、
情報処理装置。
【請求項１８】
前記状態制御部は、
前記プロセッサが前記第２の状態に遷移するときに、何れの前記周辺機器と前記記憶装置との間でもデータ転送のための処理が行われていない場合は、前記プロセッサに供給する電圧を下げる制御を行う、
請求項１７に記載の情報処理装置。
【請求項１９】
前記状態制御部は、
前記プロセッサが前記第２の状態に遷移するときに、何れの前記周辺機器と前記記憶装置との間でもデータ転送のための処理が行われていない場合は、前記プロセッサが前記第１の状態のときに用いられるクロックを示す高周波クロックを生成する高周波発振器の発振を停止させる制御を行う、
請求項１７に記載の情報処理装置。
【請求項２０】
前記状態制御部は、
前記プロセッサが前記第２の状態に遷移するときに、何れの前記周辺機器と前記記憶装置との間でもデータ転送のための処理が行われていない場合は、前記プロセッサが記第１の状態のときに用いられるクロックを示す高周波クロックを生成する高周波発振器への電力供給を停止させる制御を行う、
請求項１７に記載の情報処理装置。
【請求項２１】
前記高周波発振器は、シリコンオシレータである、
請求項１９または請求項２０に記載の情報処理装置。
【請求項２２】
前記記憶装置は揮発性メモリであり、
前記状態制御部は、

前記プロセッサが前記第２の状態に遷移するときに、何れの前記周辺機器と前記記憶装置との間でもデータ転送のための処理が行われていない場合は、前記記憶装置を、前記記憶装置に記憶されたデータを保持可能であって、データの読み出しまたは書き込みを行うことはできないセルフリフレッシュ状態に遷移させる制御を行う、
請求項１７に記載の情報処理装置。
【請求項２３】
前記記憶装置は不揮発性メモリであり、
前記状態制御部は、
前記プロセッサが前記第２の状態に遷移するときに、何れの前記周辺機器と前記記憶装置との間でもデータ転送のための処理が行われていない場合は、前記記憶装置に対する電力供給を停止させる制御を行う、
請求項１７に記載の情報処理装置。
【請求項２４】
前記プロセッサはキャッシュメモリを備え、
前記状態制御部は、
前記プロセッサが前記第２の状態に遷移するときに、何れの前記周辺機器と前記記憶装置との間でもデータ転送のための処理が行われていない場合は、前記記憶装置への書き込みが行われていない前記キャッシュメモリ内のデータを前記記憶装置へ書き出す処理を行った後、前記キャッシュメモリへの電力供給を停止させる制御を行う、
請求項１７に記載の情報処理装置。
【請求項２５】
前記プロセッサはキャッシュメモリを備え、
前記状態制御部は、
前記プロセッサが前記第２の状態に遷移するときに、何れの前記周辺機器と前記記憶装置との間でもデータ転送のための処理が行われていない場合は、前記キャッシュメモリに供給する電圧を、前記プロセッサが前記第１の状態のときよりも下げる制御を行う、
請求項１７に記載の情報処理装置。
【請求項２６】
前記周辺機器は、
ストレージデバイス、コミュニケーションデバイス、ディスプレイデバイス、イメージキャプチャデバイスのうちの何れかである、
請求項１７に記載の情報処理装置。
【請求項２７】
前記周辺機器ごとに、前記周辺機器を制御するデバイス制御部をさらに備え、
前記デバイス制御部は、前記プロセッサからの入出力要求を受けると、前記データ転送のための処理を開始し、前記データ転送のための処理が完了すれば前記プロセッサに対して割込みをかけて通知する、
請求項１７に記載の情報処理装置。
【請求項２８】
記憶装置と、
１以上の周辺機器と、
命令を実行する第１の状態、または、割込みを待つ第２の状態に遷移可能なプロセッサと、を備える情報処理装置が実行する情報処理方法であって、
前記プロセッサが前記第２の状態に遷移するときに、何れの前記周辺機器と前記記憶装置との間でもデータ転送のための処理が行われていない場合は、前記情報処理装置の消費電力を下げる制御を行う状態制御ステップを含む、
情報処理方法。
【請求項２９】
記憶装置と、
１以上の周辺機器と、

命令を実行する第１の状態、または、割込みを待つ第２の状態に遷移可能なプロセッサと、を備える情報処理装置に搭載されたコンピュータに、
前記プロセッサが前記第２の状態に遷移するときに、何れの前記周辺機器と前記記憶装置との間でもデータ転送のための処理が行われていない場合は、前記情報処理装置の消費電力を下げる制御を行う状態制御ステップを実行させるためのプログラム。
【請求項３０】
命令を実行する第１の状態、または、割込みを待つ第２の状態に遷移可能なプロセッサを備え、１以上の周辺機器が接続された半導体装置であって、
前記プロセッサが前記第２の状態に遷移するときに、何れの前記周辺機器と前記半導体装置との間でもデータ転送のための処理が行われていない場合は、前記半導体装置の消費電力を下げる制御を行う状態制御部を有する、
半導体装置。
【請求項３１】
命令を実行する第１の状態、または、割込みを待つ第２の状態に遷移可能なプロセッサを備え、１以上の周辺機器が接続された半導体装置が実行する情報処理方法であって、
前記プロセッサが前記第２の状態に遷移するときに、何れの前記周辺機器と前記半導体装置との間でもデータ転送のための処理が行われていない場合は、前記半導体装置の消費電力を下げる制御を行う状態制御ステップを含む、
情報処理方法。
【請求項３２】
命令を実行する第１の状態、または、割込みを待つ第２の状態に遷移可能なプロセッサを備え、１以上の周辺機器が接続された半導体装置に搭載されたコンピュータに、
前記プロセッサが前記第２の状態に遷移するときに、何れの前記周辺機器と前記半導体装置との間でもデータ転送のための処理が行われていない場合は、前記半導体装置の消費電力を下げる制御を行う状態制御ステップを実行させるためのプログラム。
【発明の詳細な説明】
【技術分野】
【０００１】
本発明の実施形態は、情報処理装置、半導体装置、情報処理方法およびプログラムに関する。
【背景技術】
【０００２】
携帯情報機器（例えばタブレットやスマートフォン、メガネ型や腕時計型のウェアラブル端末など）や車載情報機器やセンサシステムなどの、コンピュータシステムを内蔵する情報処理装置の省電力化は重要な技術課題の一つである。このような情報処理装置の主要な構成要素（部品）はＬＳＩチップ上にコンピュータシステムを集積したＳｏＣ(System On Chip)であり、ＳｏＣ内のプロセッサコア（プロセッサ、ＣＰＵ、ＭＰＵなどと呼ばれる場合がある）がプログラム（ソフトウェア）を実行することで様々な機能を実現する。
【０００３】
使用中の情報処理装置のプロセッサコアが取り得る状態（遷移可能な状態）は大きく分けてラン状態（あるいはアクティブ状態とも呼ばれる）とアイドル状態がある。ラン状態では、プロセッサコアはプログラム（命令）の実行を続けている。アイドル状態では、プロセッサコアはプログラムを実行せずに、Ｉ／Ｏデバイス（周辺機器）の処理の完了などのイベント発生を、割込み等の手段によって通知されるのを待っている。
【０００４】
従来の情報処理装置では、プロセッサコアがアイドル状態に入るとＳｏＣはＷＡＩＴモード（あるいはＳＬＥＥＰモードとも呼ばれる）に遷移する。ＷＡＩＴモードのＳｏＣでは、プロセッサコアはプログラムの実行を停止しているが、必要なＩ／Ｏデバイスは動作し続けていて、例えばＩ／Ｏ処理の完了、ユーザ入力、通信データの到着、タイマーなど

のイベントが発生すると、プロセッサコアはすぐにプログラムの実行を再開できる状態にあり、その状態を維持し続けるために必要な電力を消費し続けている。

【０００５】

ここで、多くのＳｏＣは、それが搭載されている情報処理装置をユーザが使用していないときなどに、低消費電力で待機するＤＥＥＰ　ＳＬＥＥＰモード（ＳＴＯＰモードとも呼ばれる）を有する。ＳｏＣをＤＥＥＰ　ＳＬＥＥＰモードにすると、プロセッサコアを停止するだけでなく、動作させる必要のないＩ／Ｏデバイスも停止させ、可能であればそれらへの電源供給を停止したり、供給する電源電圧を状態保持が可能な範囲で下げたり、クロックを停止したりすることで、消費電力を下げることができる。

【０００６】

従来は、ユーザが操作していない状態が長時間にわたって継続していることを検出したときや、バッテリの残量低下を検出したときや、ユーザから情報処理装置を省電力状態に移行する指示を受け付けたときなどに、情報処理装置はサスペンド処理（スタンバイ処理とも呼ばれる）を行い、ＳｏＣ（あるいは情報処理装置全体やプロセッサコア）を消費電力の少ないＤＥＥＰ　ＳＬＥＥＰモードに遷移させることで、消費電力を抑えていた。

【０００７】

従来のサスペンド処理では、ＳｏＣ（あるいは情報処理装置全体）を安定な状態にしてから、ＳｏＣをＤＥＥＰ　ＳＬＥＥＰモードに遷移させることで、サスペンド状態を終えた後にサスペンド前の状態から処理を継続できるようにしている。すなわち、サスペンド処理は、プロセッサコアで実行中のプログラム（あるいはオペレーティングシステムの管理するプロセスやタスク）を停止してＩ／Ｏデバイス（周辺機器）への新たな処理要求の発行を抑制するとともに、ＤＥＥＰ　ＳＬＥＥＰモード中は動作を停止するＩ／Ｏデバイス（ＤＥＥＰ　ＳＬＥＥＰモードのＳｏＣを起床させることができる起床可能デバイスではないＩ／Ｏデバイス）で実行中の処理があればその処理の完了を待ち、ＳｏＣを安定な状態にした後で、ＳｏＣをＤＥＥＰ　ＳＬＥＥＰモードにする。そのため、サスペンド処理の開始直後にＳｏＣをＤＥＥＰ　ＳＬＥＥＰモードに遷移させられるわけではなく、安定な状態にするまでに時間を要する。

【０００８】

一般に、ＷＡＩＴモードよりもＤＥＥＰ　ＳＬＥＥＰモードの方が、消費電力が小さいので、プロセッサコアがアイドル状態の時にＳｏＣをＷＡＩＴモードにする代わりにＤＥＥＰ　ＳＬＥＥＰモードにすることができれば、更に情報処理装置の平均消費電力を抑えることができる。

【０００９】

しかしながら、例えばプロセッサコアがアイドル状態に遷移する時に従来のサスペンド処理の方法を用いてＳｏＣをＤＥＥＰ　ＳＬＥＥＰモードに遷移させようとすると、サスペンド処理のオーバヘッドが無視できない長さとなり、情報処理装置の性能を大きく損なってしまうという問題がある。

【先行技術文献】

【特許文献】

【００１０】

【特許文献１】特許第３７５８４７７号公報

【発明の概要】

【発明が解決しようとする課題】

【００１１】

本発明が解決しようとする課題は、性能を損なうことなく省電力化を適切に行うことが可能な情報処理装置、半導体装置、情報処理方法およびプログラムを提供することである。

【課題を解決するための手段】

【００１２】

実施形態の情報処理装置は、記憶装置と、１以上の周辺機器と、命令を実行する第１の

状態、または、割込みを待つ第２の状態に遷移可能なプロセッサと、を備える情報処理装置であって、状態制御部を有する。状態制御部は、プロセッサが第２の状態に遷移するときに、少なくとも１つの周辺機器と記憶装置との間でデータ転送のための処理が行われている場合は、プロセッサが第１の状態のときよりも消費電力が小さい第３の状態に情報処理装置を遷移させる制御を行う。一方、何れの周辺機器と記憶装置との間でもデータ転送のための処理が行われていない場合は、状態制御部は、第３の状態よりも消費電力が小さい第４の状態に情報処理装置を遷移させる制御を行う。

【図面の簡単な説明】

【００１３】

【図１】第１実施形態の情報処理装置のハードウェア構成例を示す図。

【図２】第１実施形態のＰＭＩＣの構成例を示す図。

【図３】第１実施形態のＳｏＣの電源ドメインを示す図。

【図４】第１実施形態の高周波発振器の起動／停止を説明するための図。

【図５】変形例の情報処理装置のハードウェア構成例を示す図。

【図６】変形例の情報処理装置のハードウェア構成例を示す図。

【図７】変形例の情報処理装置のハードウェア構成例を示す図。

【図８】第１実施形態のクロック制御モジュールの構成例を示す図。

【図９】変形例のクロック制御モジュールの構成例を示す図。

【図１０】実施形態のマイコンの構成例を示す図。

【図１１】第１実施形態のオペレーティングシステムの機能構成例を示す図。

【図１２】第１実施形態のデバイス状態管理情報の一例を示す図。

【図１３】第１実施形態のデバイスドライバの動作例を示すフロー図。

【図１４】第１実施形態のプロセス切替部の動作例を示すフロー図。

【図１５】第１実施形態のＳｏＣの動作例を示すフロー図。

【図１６】第１実施形態のＳｏＣとＰＭＩＣとの連携を説明するための図。

【図１７】第１実施形態のＳｏＣの動作例を示すフロー図。

【図１８】第５実施形態のＩ／Ｏデバイスの状態遷移の様子を示す図。

【図１９】第７実施形態のプロセス切替部の動作例を示すフロー図。

【図２０】第８実施形態の情報処理装置の外観を示す図。

【図２１】第８実施形態の情報処理装置のハードウェア構成例を示す図。

【図２２】第８実施形態の情報処理装置のハードウェア構成例を示す図。

【図２３】変形例の情報処理装置のハードウェア構成例を示す図。

【図２４】第８実施形態のＳｏＣの状態遷移の一例を示す図。

【図２５】第８実施形態のオペレーティングシステムの機能構成例を示す図。

【図２６】第８実施形態のデバイスドライバの動作例を示すフロー図。

【図２７】第８実施形態のプロセス切替部の動作例を示すフロー図。

【図２８】第８実施形態のマイコンとＳｏＣとの接続例を示す図。

【図２９】第８実施形態のマイコンによる電源管理を説明するための図。

【図３０】第８実施形態のマイコンの動作例を示すフロー図。

【図３１】第８実施形態のＳｏＣの動作例を示すフロー図。

【図３２】第９実施形態のハイパーバイザの機能構成例を示す図。

【図３３】第９実施形態のＷＦＩ処理部の動作例を示すフロー図。

【図３４】第９実施形態のＩ／Ｏデバイス制御部の動作例を示すフロー図。

【図３５】第９実施形態の割込み処理の動作例を示すフロー図。

【発明を実施するための形態】

【００１４】

以下、添付図面を参照しながら、本発明に係る情報処理装置、情報処理方法およびプログラムの実施形態を詳細に説明する。

【００１５】

（第１実施形態）

図１は、本実施形態の情報処理装置１のハードウェア構成の一例を示す図である。図１に示すように、情報処理装置１は、電源装置１０と、ＰＭＩＣ(Power Management IC)２０と、メインメモリ３０と、ＳｏＣ(System On Chip)１００と、表示デバイス２００と、ストレージデバイス２１０と、ネットワークデバイス２２０と、ＨＩＤ２３０とを備える。以下の説明では、表示デバイス２００、ストレージデバイス２１０、ネットワークデバイス２２０、および、ＨＩＤ２３０を互いに区別しない場合は、単に「Ｉ／Ｏデバイス」と称する場合がある。このＩ／Ｏデバイスは、ＳｏＣ１００に接続されて利用されるデバイス（情報処理装置１に搭載されたコンピュータシステム（プロセッサコア）と組み合わせて利用されるデバイス）であり、請求項の「周辺機器」に対応していると考えることができる。

【００１６】

電源装置１０は、例えばＡＣアダプタなどのＡＣ電源、アルカリマンガン電池などの一次電池やニッケル水素電池などの二次電池、エナジーハーベスト装置などの発電装置と蓄電装置の組合せ等、さまざまな種類のものを用いることができる。エナジーハーベスト装置とは、光エネルギーを利用する太陽電池や、熱エネルギーや振動エネルギーを利用する環境発電装置である。エナジーハーベスト装置の発電する電力のみでは情報処理装置１の動作のピーク時の消費電力を得ることができない場合には、消費電力が少ない間に余剰電力を蓄電装置に蓄電しておき、ピーク時の電力を賄うことができる。このような使い方をピークアシストとも呼ぶ。蓄電装置は、電気２重層キャパシタやリチウムイオンキャパシタなどの大容量キャパシタ、又はリチウムイオン電池などのバッテリなどを用いることができる。蓄電装置として、大容量キャパシタ及びバッテリの両方を組み合わせて用いてもよい。

【００１７】

ＰＭＩＣ２０は、電源装置１０の供給する電力を、ＳｏＣ１００、メインメモリ３０、Ｉ／Ｏデバイスなどの各部に必要な電圧に変換して各部に供給する。図２にＰＭＩＣ２０の構成の一例を示す。ＰＭＩＣ２０は、複数のスイッチングレギュレータやリニアレギュレータなどの様々な方式のレギュレータを１つ以上内蔵しており、電源装置１０から供給される入力電圧を、それぞれの電源ライン毎の電圧に変換する。ＰＭＩＣ２０は、電源ライン毎に設定すべき出力電圧を示す情報を格納する制御レジスタを内蔵しており、それぞれの電源ラインのレギュレータの出力電圧は、対応する制御レジスタに基づいて設定することができる。また、制御レジスタは、後述のＳＴＢＹ信号を受信した際の各電源ラインのレギュレータの出力電圧および電力供給のＯＮ／ＯＦＦを示す情報も格納することができ、ＰＭＩＣ２０は、後述のＳＴＢＹ信号を受信した際の各電源ラインのレギュレータの出力電圧および電力供給のＯＮ／ＯＦＦも、対応する制御レジスタに基づいて設定することができる。以上より、例えばＳｏＣ１００がＤＥＥＰ　ＳＬＥＥＰモードであることをＳＴＢＹ信号でＰＭＩＣ２０に伝えると、ＰＭＩＣ２０は各電源ラインの出力電圧を指定した電圧に変更し、あるいは電源供給を停止することができる。

【００１８】

通常、ＳｏＣ１００とＰＭＩＣ２０はＩ２ＣやＳＰＩなどのシリアルバス（不図示）で接続されており、ＳｏＣ１００からシリアルバスを介してＰＭＩＣ２０内の制御レジスタに値を設定することができる。なお、ＰＭＩＣ２０としてワンチップに集積された部品を用いる代わりに、複数のＤＣ／ＤＣコンバータ（チップまたはモジュール）を組み合わせて用いても良い。また、ＰＭＩＣ２０の一部あるいは全部の機能がＳｏＣ１００に内蔵されている場合もある。すなわち、ＳｏＣ１００内にＤＣ／ＤＣコンバータやレギュレータを内蔵し、内部で電源供給のＯＮ／ＯＦＦや電圧の変更を行うこともできる。

【００１９】

図１に戻って説明を続ける。メインメモリ３０は、揮発性のメモリで構成されてもよいし、不揮発性のメモリで構成されてもよい。揮発性のメモリとしては、例えばＤＲＡＭ（Dynamic Random Access Memory）やＳＲＡＭ（Static Random Access Memory）などを使用することができる。不揮発性のメモリとしては、例えばＭＲＡＭ(Magnetoresist

ive Random Access Memory)やＰＣＭ(Phase Change Memory)、ＲｅＲＡＭ(Resistance Random Access Memory)、ＦｅＲＡＭ(Ferroelectric Random Access Memory)、ＮＯＲ Ｆｌａｓｈなどを使用することができる。例えばプログラムはＮＯＲ Ｆｌａｓｈに記憶し、データはＤＲＡＭに記憶するように、複数の種類のメモリを組み合わせてメインメモリ３０として使ってもよい。メインメモリ３０として用いるメモリは、ＳｏＣ１００内に設けられたメモリコントローラ１０２を介してＳｏＣ１００内のコンピュータシステムに接続される。メインメモリ３０は、請求項の「記憶装置」に対応している。

【００２０】

メモリの中には、プロセッサコアやデバイスコントローラなどがアクセスしない間は、電力消費の少ない省電力モードに遷移させることができるものがある。例えば、広く使われているＤＤＲ３やＬＰＤＤＲ２などの近年のＤＲＡＭの場合はセルフリフレッシュモードに遷移することで消費電力を抑えることができる。セルフリフレッシュモード（セルフリフレッシュ状態）とは、メモリに記憶しているデータを保持可能であって、データの読み出しや書き込みを行うことができない状態を指す。一方、不揮発性メモリの場合は、当該不揮発性メモリへの電力供給が停止しても当該不揮発性メモリが記憶しているデータを消失することはないので、不揮発性メモリへの電力供給を停止させることにより消費電力を抑えることができる。なお、セルフリフレッシュモードに遷移しているＤＲＡＭ（揮発性メモリの一例）や、電力供給が遮断されている不揮発性メモリに対して、再度メモリアクセスを行うには、セルフリフレッシュモードから通常モードに復帰させる、あるいは、遮断していた電力を復帰させるなど、通常モードに戻す処理を行わなければならない。通常モードとは、メモリに記憶しているデータを保持可能であって、かつ、データの読み出しや書き込みを行うことが可能な状態を指す。

【００２１】

次に、ＳｏＣ１００について説明する。ＳｏＣ１００は、請求項の「半導体装置」に対応している。図１に示すように、ＳｏＣ１００は、内部モジュールとして、プロセッサコア１０１、メインメモリ３０を制御するメモリコントローラ１０２、表示デバイス２００を制御する表示デバイスコントローラ１０３、ストレージデバイス２１０を制御するストレージコントローラ１０４、ネットワークデバイス２２０を制御するネットワークコントローラ１０５、ＨＩＤ２３０を制御するＨＩＤコントローラ１０６、内部モジュール間のデータの送受信を行うバス１０７、高周波クロックを生成する高周波発振器１０８、低周波クロックを生成する低周波発振器１０９、クロックを様々な周波数に変換したりクロック供給の可否（ＯＮ／ＯＦＦ）を制御したりするクロック制御モジュール１１０、Ｉ／Ｏデバイスなどからの割込みを受信しプロセッサコア１０１に割込みを送る割込みコントローラ１２０、ＳｏＣ１００内部の記憶領域である内部メモリ１２５（ローカルメモリやスクラッチパッドメモリとも呼ばれる）、計時機能を有するＲＴＣ（Real Time Clock）１３０などを有する。プロセッサコア１０１にはＡＲＭプロセッサをはじめとする様々なアーキテクチャのプロセッサを用いることができる。また、図１の例では、プロセッサコア１０１は、キャッシュメモリ１４０を有している。なお、以下の説明では、表示デバイスコントローラ１０３、ストレージコントローラ１０４、ネットワークコントローラ１０５、および、ＨＩＤコントローラ１０６を互いに区別しない場合は、単に「デバイスコントローラ」と称する場合がある。このデバイスコントローラは、請求項の「デバイス制御部」に対応している。

【００２２】

図３に示すように、ＳｏＣ１００は、複数の電源ドメインに分かれており、各内部モジュールはいずれかの電源ドメインに属している。各電源ドメインに対しては、独立した電源ラインから電力が供給されており、それぞれ電圧値の変更や電力の供給／停止（ＯＮ／ＯＦＦ）が独立に制御できる。図３の例では、プロセッサコア１０１のうち内部のキャッシュメモリ１４０以外の部分と、キャッシュメモリ１４０とが別々の電源ドメインに属し、それぞれ独立に供給電圧の変更や電力供給のＯＮ／ＯＦＦを制御できる。例えばプロセッサコア１０１のうちキャッシュメモリ１４０以外の部分への電力供給を停止しても、キ

ャッシュメモリ１４０への電力供給を続けておけば、キャッシュメモリ１４０上のデータを失うことはない。

【００２３】

キャッシュメモリ１４０が揮発性メモリで構成されている場合は、電力供給を遮断するとデータが失われるため、キャッシュメモリ１４０への電力供給をＯＦＦする前にキャッシュメモリ１４０内のダーティなデータ（プロセッサコア１０１によってキャッシュメモリ１４０には書き込まれたがメインメモリ３０にはまだ書き込まれていないデータ）をメインメモリ３０に書き出すクリーン処理が必要である。また、プロセッサコア１０１がアイドル状態においてキャッシュメモリ１４０への電力供給をＯＦＦしている場合に、ラン状態に復帰する際は、電力遮断により不定値になったキャッシュメモリ１４０のデータを無効化するためのインバリデート処理が必要である。一方、キャッシュメモリ１４０が不揮発性メモリで構成されている場合は、電力供給を遮断しても（ＯＦＦになっても）データは消失しないので、クリーン処理やインバリデート処理は不要である。

【００２４】

ここで、プロセッサコア１０１は、プログラム（命令）を実行しているラン状態、または、Ｉ／Ｏデバイスやタイマー等からのイベント発生を通知する割込みを待っているアイドル状態に遷移可能であり、ラン状態からアイドル状態に遷移する際には、ＳｏＣ１００の制御レジスタ（スペシャルレジスタあるいはコントロールレジスタとも呼ばれる）の設定に従って、プロセッサコア１０１に供給されるクロックを止めたり、レジスタの値を保持可能な範囲で電圧を下げたりすることで、プロセッサコア１０１の消費電力を抑えることができる。アイドル状態のプロセッサコア１０１は、割込み信号を受信すると、ラン状態に遷移してプログラムの実行を再開する。ラン状態は請求項の「第１の状態」に対応し、アイドル状態は請求項の「第２の状態」に対応していると考えることができる。

【００２５】

図１および図３に示すＳｏＣ１００は、ＳｏＣ１００内にプロセッサコア１０１を１つ内蔵するシングルコアの例である。このほかに、ＳｏＣ１００内に複数のプロセッサコアを内蔵するマルチコアの構成をとることもできる。本発明は、シングルコアのＳｏＣに適用することもできるし、マルチコアのＳｏＣに適用することもできる。マルチコアの場合は、プロセッサコアのキャッシュメモリの構成としては、以下のような様々な構成が存在する。

（ａ）Ｌ１キャッシュメモリ、Ｌ２キャッシュメモリを各プロセッサコアがそれぞれ持つ場合。

（ｂ）Ｌ１キャッシュメモリを各プロセッサコアが持ち、Ｌ２キャッシュメモリを複数のプロセッサコアから成るグループで共有する場合。

（ｃ）Ｌ１キャッシュメモリを各プロセッサコアが持ち、Ｌ２キャッシュメモリをすべてのプロセッサコアで共有する場合。

【００２６】

本発明は、いずれの構成のマルチコアを用いても良い。また、上記に限らず、Ｌ３キャッシュのような、さらに多段のキャッシュメモリを用いても良い。

【００２７】

図１の説明を続ける。高周波発振器１０８は、ＳｏＣ１００内の各モジュールが必要とするクロックを生成するための基になるメインクロックを生成する発振回路である。すなわち、高周波発振器１０８は、プロセッサコア１０１がラン状態のときに用いられるクロックを示すメインクロック（以下の説明では、「高周波クロック」と称する場合がある）を生成する。この高周波クロックは、例えば２４ＭＨｚの高い周波数のクロックである。高周波発振器１０８は、低周波発振器１０９と比較して、発振してクロックを生成している時の消費電力が大きい。そのため、ＤＥＥＰ　ＳＬＥＥＰ中などのメインクロックが必要ない時には高周波発振器１０８を停止させることで情報処理装置１全体の消費電力を抑えることができる。高周波発振器１０８の起動／停止を制御する方法としては、図４に示すように、ＳｏＣ１００から、高周波発振器１０８へ供給するイネーブル信号（発振状態

にするかしないかを指示する信号）のＯＮ／ＯＦＦ（active／inactive）を設定する方法や、ＰＭＩＣ２０から高周波発振器１０８への電力供給をＯＮ／ＯＦＦする方法などを用いることができる。

【００２８】

図１に戻って説明を続ける。低周波発振器１０９は、ＤＥＥＰ　ＳＬＥＥＰ中でも動作するモジュール（ＳｏＣ１００を起床可能なＩ／Ｏデバイス）や割込みコントローラ１２０やＲＴＣ１３０などが必要とするサブクロック（以下の説明では、「低周波クロック」と称する場合がある）を生成する発振回路である。低周波発振器が生成するサブクロックの周波数はメインクロックの周波数よりも低く、例えば３２ｋＨｚの周波数のクロックである。低周波発振器１０９は、生成するクロックの周波数が低く消費電力が低いため、常時発振させてクロックを生成し続けても消費電力は小さい。

【００２９】

図１の例では、ＳｏＣ１００は、高周波発振器１０８や低周波発振器１０９を内蔵している。このようなＳｏＣ１００に内蔵された高周波発振器１０８や低周波発振器１０９は、水晶発振回路、ＣＲ発振回路、シリコン発振回路、ＭＥＭＳ発振回路など、さまざまな方式の発振回路によって実現できるが、特に、水晶発振回路を用いられることが多い。水晶発振回路を用いる場合は、例えば図５のように、発振周波数を決定する水晶振動子１３１をＳｏＣ１００に外付けすることもできる。また、例えば図６のように、高周波発振器１０８や低周波発振器１０９をＳｏＣ１００に内蔵せず、ＳｏＣ１００の外部に設けた高周波発振器１０８や低周波発振器１０９からクロック信号を入れる構成であってもよい。このような構成では、外部に水晶発振器やシリコン発振器などの発振回路が接続され、それらが生成したクロックでＳｏＣ１００が動作する。また、例えば図７のように、クロック信号をセレクタ１３２で選択する構成であってもよい。この構成によれば、ＳｏＣ１００内蔵の高周波発振器１０８や低周波発振器１０９でクロックを生成して使う方法（図１、図５）と、外付けの高周波発振器１０８や低周波発振器１０９で生成したクロックをＳｏＣ１００に入力して使う方法（図６）の、両方の使い方が可能で、用途に応じて任意の方法を選べることができる。

【００３０】

また、一般に、発振器が停止した状態（発振器への電源供給が停止されている状態、あるいは電源は供給されているがイネーブル信号がＯＦＦ(inactive)で発振を停止している状態）から発振を開始して安定するまでに要する時間は、水晶発振器の場合は数msec要するが、シリコン発振器（シリコンオシレータ）の場合は数百μsecと短い。そのため，プロセッサコア１０１がラン状態からアイドル状態に入るとＳｏＣ１００をＤＥＥＰ　ＳＬＥＥＰモードにして高周波発振器１０８を停止させて省電力化する場合のように、頻繁に高周波発振器１０８の発振のＯＮ／ＯＦＦを繰り返すような使い方をするときには、高周波発振器１０８にシリコン発振器を用いるとアイドル状態からラン状態への切り替え時に高周波発振器１０８を発振開始させるための時間的なオーバヘッドを小さくできる。したがって、高周波発振器１０８としては、シリコンオシレータを用いることが好ましい。

【００３１】

高周波発振器１０８や低周波発振器１０９で生成されたクロック信号はＳｏＣ１００内のクロック制御モジュール１１０で複数の異なる周波数のクロック信号に変換してＳｏＣ１００内の各モジュールへ分配される。図８は、クロック制御モジュール１１０の構成の一例を示す図である。図８に示すように、クロック制御モジュール１１０は、ＰＬＬ(Phase　Locked　Loop)回路１１１、周波数調整部１１２、クロックゲート部１１３を備える。

【００３２】

ＰＬＬ１１１は、高周波発振器１０８から入力されたクロック信号（メインクロック）を、安定した高い周波数のクロック信号に変更する。ＳｏＣ１００の内部では、複数の異なる周波数のクロック信号が必要になるので、周波数調整部１１２は複数個存在し、各周波数調整部１１２はＰＬＬ１１１から供給されるクロック信号をさらに逓倍や分周するこ

とで、所望の周波数のクロック信号に変更することができる。図８に示すクロック制御モジュール１１０はＰＬＬ１１１を１つしか有していないが、これに限らず、例えば図９のように、複数のＰＬＬ１１１を備える構成であってもよい。クロック制御モジュール１１０が複数のＰＬＬ１１１を備える構成の場合、各ＰＬＬ１１１で異なる周波数のクロック信号を生成し、その中から都合の良いクロック信号をクロック選択部１１４で選んで各周波数調整部１１２へ入力することができる。クロックゲート部１１３は、周波数調整部１１２からＳｏＣ１００の各内部モジュールへのクロック信号のＯＮ／ＯＦＦを制御する。周波数調整部１１２の逓倍や分周の値や、クロックゲート部１１３によるクロック信号の出力のＯＮ／ＯＦＦ、複数のＰＬＬ１１１を備える場合の各周波数調整部１１２に対して周波数を入力するＰＬＬ１１１（入力ＰＬＬ１１１）の選択は、クロック制御モジュール１１０の制御レジスタ１１５を設定（制御レジスタ１１５に各種の制御情報を設定）することで制御できる。また、クロック制御モジュール１１０は、ＰＬＬ１１１などを経由せず直接クロック信号を内部モジュールに出力することもでき、例えば低周波のクロック生成装置で生成されたクロックを、直接、割込みコントローラ１２０やＲＴＣ１３０に送ることもできる。

【００３３】

図８および図９の例では、クロック制御モジュール１１０内のクロックゲート部１１３でクロック信号の出力をＯＦＦ（停止）することによって、その先に接続されている内部モジュールの動作を止めて消費電力を下げる、いわゆるクロックゲーティングを実現できる。また、クロック制御モジュール１１０からのクロック信号の出力をＯＮ／ＯＦＦするのではなく、各内部モジュールがクロック入力のＯＮ／ＯＦＦを制御するレジスタを備え、このレジスタの設定によりクロック入力をＯＦＦに設定することでクロックゲーティングを行うこともできる。

【００３４】

図１に戻って説明を続ける。情報処理装置１は、Ｉ／Ｏデバイスとして、表示デバイス２００、ストレージデバイス２１０、ネットワークデバイス２２０、ヒューマンインタフェイスデバイス（以下、「ＨＩＤ」と称する場合がある）２３０を備える。ただし、Ｉ／Ｏデバイスの種類はこれらに限定されるものではない。本発明が適用されるＳｏＣは、例えばＩ２Ｃ、ＳＰＩ、ＵＡＲＴ、ＵＳＢ、ＳＤＩＯなどの汎用で様々な用途に使えるシリアルインタフェースを備えることもできる。また、カメラなどのイメージキャプチャデバイスから画像データを入力するためのインタフェースを備えることもできる。要するに、Ｉ／Ｏデバイスとしては、ストレージデバイス、コミュニケーションデバイス、ディスプレイデバイス、イメージキャプチャデバイスなどの様々なデバイスを使用することができる。

【００３５】

Ｉ／Ｏデバイスは、そのＩ／Ｏデバイスを制御するデバイスコントローラを介してＳｏＣ１００内のコンピュータシステム（プロセッサコア１０１等）と接続されている。デバイスコントローラは、Ｉ／Ｏデバイスへの処理の開始・停止の指示や、Ｉ／Ｏデバイスとメインメモリ３０との間のデータの転送などの制御を行う。デバイスコントローラは図１のようにＳｏＣ１００の内部モジュールである場合が多いが、これに限らず、例えばＳｏＣ１００の外部にデバイスコントローラの回路（チップあるいはＬＳＩ）が接続される構成であってもよい。なお、例えば無線通信用のデバイスのように、アンテナなどの最小限の外付け部品を除いて、ＳｏＣ１００にＩ／Ｏデバイスを内蔵することも多い。この場合、Ｉ／Ｏデバイスとデバイスコントローラは一体に設けられており、その全体をＩ／Ｏデバイスと呼ぶこともある。また、Ｉ２Ｃ、ＳＰＩ、ＵＳＢ、ＳＡＴＡ、ＳＤＩＯなどのＳｏＣ１００と外部のＩ／Ｏデバイスを接続する汎用の通信インタフェースのデバイスコントローラを、インタフェースと呼ぶこともある。

【００３６】

表示デバイス（ディスプレイデバイス）２００としては、例えば液晶ディスプレイ、ＰＳＲ(Panel Self Refresh)対応液晶ディスプレイ、ＥＰＤ(Electrophoretic Display

、電子ペーパー)などを用いることができるが、これらに限定されない。液晶ディスプレイは、表示内容に変更が無くても、表示を継続するためにメインメモリ３０上あるいはディスプレイコントローラ（表示デバイスコントローラ１０３の一例）内のメモリ上のフレームバッファに記憶している表示データを液晶ディスプレイに送るリフレッシュ処理を、毎秒数十回（例えば３０回あるいは６０回など）の頻度で繰り返している。リフレッシュ処理中は、フレームバッファから液晶ディスプレイへのデータ転送が発生するため、データ転送が中断されないように表示デバイスコントローラ１０３やデータ転送用のバス１０７にクロックを供給し続けなければならず、また、フレームバッファに使っているメモリもアクセス可能な状態にしておかなければならない。

【００３７】

ＰＳＲ対応液晶ディスプレイは、ディスプレイ側に表示データを保存するためのバッファがあり、表示内容に変更が無い場合は、ディスプレイ側のバッファに保存されている表示データを用いてディスプレイ内でリフレッシュ処理を行う。そのため、表示内容に変更が無い場合は、ＳｏＣ１００はメインメモリ３０上やディスプレイコントローラ内のメモリ上にあるフレームバッファ上の表示データをディスプレイ側に転送する必要はない。そのため、液晶ディスプレイと異なり、表示内容に変更のない間は、表示デバイスコントローラ１０３やデータ転送用のバス１０７へのクロックの供給を停止し、フレームバッファのあるメモリをアクセス不可能な省電力状態にして、消費電力を削減することができる。

【００３８】

ＥＰＤは電力供給が無くても表示内容を保持することができる。そのため、表示内容を変更しない期間は、ＥＰＤへの電力供給を停止したり、ＥＰＤコントローラ（表示デバイスコントローラ１０３の一例）へのクロック供給を停止したりすることで消費電力を抑えることができる。表示内容を変更する場合は、表示内容のデータ（メインメモリ３０上に設けたフレームバッファのデータ）をＥＰＤコントローラに転送することや、表示内容のデータを元にＥＰＤを書き換えるための信号を生成するために、ＥＰＤやＥＰＤコントローラへ電源やクロックを供給することは当然必要であり、また、表示内容のデータを記憶しているメインメモリ３０はアクセス可能な状態にしておかなければならない。

【００３９】

ストレージデバイス２１０は、情報処理装置１の２次ストレージとして、データやプログラムを記憶するために使用する。ストレージデバイス２１０は、例えばＮＡＮＤフラッシュメモリ（チップ）、ＮＯＲフラッシュメモリ（チップ）、ＳＤカードなどの各種メモリカード、ハードディスク、ＳＳＤ、ＤＶＤ－ＲＡＭのような記録可能な光ディスクデバイスなどを用いることができるが、これらのデバイスに限定されるものではなく、ストレージデバイス２１０の構成は任意である。ストレージデバイス２１０は、ストレージコントローラ１０４を介してＳｏＣ１００内のコンピュータシステムに接続される。例えば、ストレージデバイス２１０がＳＤカードの場合、ＳＤカードの接続されるストレージコントローラ１０４はＳＤコントローラあるいはＳＤＩＯコントローラである。ストレージデバイス２１０がＳＡＴＡインタフェースのＨＤＤあるいはＳＳＤである場合、それらが接続されるストレージコントローラ１０４はＳＡＴＡコントローラである。ストレージデバイス２１０がＮＡＮＤフラッシュメモリのチップである場合は、それが接続されるストレージコントローラ１０４はＮＡＮＤコントローラである。ストレージデバイス２１０がＳＰＩインタフェースのＮＯＲフラッシュメモリ（チップ）の場合、それが接続されるストレージコントローラ１０４はＳＰＩコントローラである。

【００４０】

ストレージコントローラ１０４は、アプリケーションプログラムあるいはＯＳからの指示を受けて、ストレージデバイス２１０に対するデータの読み書きなどの処理を実行する。ストレージデバイス２１０からデータを読み込む場合は、ストレージコントローラ１０４はストレージデバイス２１０にデータを読み出して送り返すように指示し、送り返されてきたデータは，ストレージコントローラ１０４が内蔵するＤＭＡコントローラ、もしくは、ストレージコントローラ１０４とは別にＳｏＣ１００内に設けられたＤＭＡコントロ

ーラ等により、アプリケーションプログラムがメインメモリ３０上に用意したバッファ領域や、ＯＳがメインメモリ３０上にページキャッシュ（あるいはバッファキャッシュ、ディスクキャッシュなどとも呼ばれる）として管理している領域などに格納される。アプリケーションプログラムがバッファ領域のデータをストレージデバイス２１０に書き込む場合や、ＯＳが定期的にページキャッシュ内の変更された（ダーティな）データをストレージデバイス２１０へ書き戻す場合は，ストレージコントローラ１０４は、ＤＭＡコントローラによって読み出したデータをストレージデバイス２１０に送って書き込ませる。ストレージコントローラ１０４は、データの書き込みや読み出し処理の完了を、割込みコントローラ１２０経由で割込みをかけてプロセッサコア１０１に通知する。ストレージコントローラ１０４がデータの読み出し処理や書き込み処理を行なっている間は、ストレージコントローラ１０４への電力とクロックの供給、およびストレージデバイス２１０への電力の供給は必要であり、メインメモリ３０もアクセス可能な状態にしておかなければならない。

【００４１】

ネットワークデバイス（コミュニケーションデバイス）２２０としては、例えばＥｔｈｅｒｎｅｔ（登録商標）などの有線ＬＡＮデバイス、８０２．１１ａ／８０２．１１ｂ／８０２．１１ｇ／８０２．１１ｎ／８０２．１１ａｃなどの伝送規格の無線ＬＡＮデバイス、近距離無線通信のＢｌｕｅｔｏｏｔｈ、Ｚｉｇｂｅｅ（登録商標）、ＴｒａｎｓｆｅｒＪｅｔなど、様々な通信方式のものを用いることができる。

【００４２】

ＳｏＣ１００に外付けのネットワークデバイス２２０とＳｏＣ１００に内蔵のネットワークコントローラ１０５の間の機能分担にはさまざまなバリエーションがある。例えばＥｔｈｅｒｎｅｔの場合、通信に必要な多くの機能はＳｏＣ１００内のネットワークコントローラ（Ｅｔｈｅｒｎｅｔコントローラ）１０５に実装され、ＳｏＣ１００に外付けのネットワークデバイス２２０としては物理層（ＰＨＹ）のドライバＩＣ（チップ）が接続されることが多い。また無線ＬＡＮやＺｉｇＢｅｅなどの場合、ＳＰＩあるいはＳＤＩＯあるいはＵＳＢあるいはＵＡＲＴなどのインタフェースを持つ通信モジュールをネットワークデバイス２２０として利用することも多い。この場合、通信に必要な多くの機能はネットワークデバイス２２０である通信モジュールに実装されおり、ネットワークコントローラ１０５としてはＳＰＩコントローラあるいはＳＤＩＯコントローラあるいはＵＳＢコントローラあるいはＵＡＲＴコントローラなどを用いる。

【００４３】

ネットワークコントローラ１０５は、ネットワークデバイス２２０を制御してデータの送受信などの処理を行う。ネットワークにデータを送信する際は、メインメモリ３０上の送信したいデータはネットワークコントローラ１０５に内蔵のＤＭＡコントローラ、あるいは、ネットワークコントローラ１０５の外部のＤＭＡコントローラによって読み出され、ネットワークコントローラ１０５からネットワークデバイス２２０に転送され、さらにネットワークデバイス２２０からネットワークへ送り出される。ネットワークからデータを受信する際は、ネットワークデバイス２２０で受信したデータはネットワークコントローラ１０５へ転送され、さらにネットワークコントローラ１０５に内蔵のＤＭＡコントローラあるいはネットワークコントローラ１０５の外部のＤＭＡコントローラを使ってメインメモリ３０上に格納される。ネットワークコントローラ１０５は、データの送受信の完了を、割込みコントローラ１２０経由で割込みをかけることでプロセッサコア１０１に通知する。

【００４４】

ＳＤＩＯインタフェースを持つ無線ＬＡＮ通信モジュールのようなＩ／Ｏデバイスは、通信モジュール内に無線（RF）回路や制御用のプロセッサ、メモリ、ＳＤＩＯコントローラ、それらをつなぐバス、クロック用のオシレータ等を内蔵している。このような通信モジュールをネットワークデバイス２２０として用いる場合は、通信モジュールに電源が供給されていれば、それが接続されているネットワークコントローラ（ＳＤＩＯコントロー

ラ）１０５が停止していても、無線ＬＡＮ通信モジュール内のメモリからネットワークへのデータ送信、ネットワークから無線ＬＡＮ通信モジュール内のメモリへのデータ受信を実行できる。無線ＬＡＮ通信モジュールとＳｏＣ１００の間でデータの送受信を行う時には、ネットワークコントローラ１０５であるＳＤＩＯコントローラが動作していなくてはならない。

【００４５】

ＨＩＤ２３０は、例えばキーボード、タッチパネル、マウスなどユーザが入力を行うデバイスの総称である。ＨＩＤ２３０としてＵＳＢインタフェースのキーボードあるいはマウスを用いる場合は、それが接続されるデバイスコントローラ（ＨＩＤコントローラ１０６）はＵＳＢコントローラになる。ＵＳＢインタフェースのキーボードの場合、押されたキーに対応するコード（キーコード）がキーボードからＵＳＢバスを介してＳｏＣ１００に送られる。例えばＳｏＣ１００は、マトリックス上に配線したキースイッチ（キーパッド）を直接接続できるキーパッドコントローラを持っている。キーパッドコントローラは、接続されているいずれかのキースイッチが押されると、割込みコントローラ１２０を経由してプロセッサコア１０１に割込みを送る。割込みを受信したプロセッサコア１０１は、押されたキースイッチを識別する情報をキーパッドコントローラから読み出し、必要に応じてそのキースイッチに対応するキーコードに変換して、ＯＳやアプリケーションに伝える。

【００４６】

ＨＩＤ２３０としてＩ２Ｃインタフェースのタッチパネルを用いる場合は、ＨＩＤコントローラ１０６としてＩ２Ｃコントローラを用いる。タッチパネルがタッチされると、タッチパネルからＳｏＣ１００に対してＩ２Ｃインタフェースを介して座標データが送られ、割込みによってプロセッサコア１０１に通知される。プロセッサコア１０１は、必要に応じて座標データを補正（ノイズ除去や座標変換などの処理）した後、ＯＳやアプリケーションに座標データを伝える。ＳｏＣ１００がＡ／Ｄ（Analog to Digital）コンバータを内蔵する場合、タッチパネルをＡ／Ｄコンバータを介して接続する場合もある。この場合は、ＳｏＣ１００内のプロセッサコア１０１が適切なタイミングでＡ／Ｄコンバータを動作させて座標データを検出し、必要に応じて座標データを補正した後、ＯＳやアプリケーションに座標データを伝える。

【００４７】

なお、ＨＩＤ２３０等をＳｏＣ１００に直接接続するのではなく、例えば図１０に示すマイコン３００を介して接続することもできる。マイコンとは、一つの半導体チップにコンピュータシステムを集積したＬＳＩ製品を指す。マイコン３００は、ＳｏＣ１００内のプロセッサコア１０１よりも処理性能が低いが消費電力も小さいので、Ｉ／Ｏデバイスの Ｉ／Ｏ処理の際に頻繁に発生するが短時間で完了するような処理をマイコン３００に代行させる（オフロードする）ことで、情報処理装置１全体の消費電力を削減することができる。例えばＨＩＤ２３０としてキーパッド（キースイッチのマトリックス）を、マイコン３００を介してＳｏＣ１００に接続する場合、チャタリング防止やリピート処理やキーコード変換などをマイコン３００にオフロードできる。タッチパネルの場合は、マイコン３００が内蔵するＡ／Ｄコンバータによって入力したデータを元に、座標変換やノイズ除去やドラッグ処理などをマイコン３００にオフロードできる。タッチパネルやキーボードが押されたらＳｏＣ１００の電源を投入して処理を再開できるように、ＨＩＤ２３０の制御と電源管理の機能を併せてマイコンに持たせることも多い。ＨＩＤ２３０だけでなく、さまざまなセンサーデバイスをマイコン３００に接続し、センサーデバイスから入力したデータのノイズ除去などの計算処理をマイコンにオフロードすることも多い。マイコン３００とＳｏＣ１００は、Ｉ２ＣやＳＰＩなどのシリアルバスやＧＰＩＯあるいはＳｏＣの外部メモリバスなどで接続され、これらの接続を介してデータの送受信を行う。

【００４８】

図１０は、マイコン３００の構成例を示す図である。図１０に示すように、マイコン３００は、プロセッサコア３０１、メモリ３０２、バス３０３、オシレータ３０４、ＳｏＣ

１００とのデータの送受信にも用いるＩ２ＣやＳＰＩやＵＡＲＴなどのシリアルバスコントローラ３０５、および、割込みコントローラ３０６を内蔵している。図１０には図示していないが、ＧＰＩＯやＡ／ＤコンバータやＤ／ＡコンバータやＰＷＭ回路などのＩ／Ｏデバイスを内蔵する場合もある。オシレータを内蔵しているため、ＳｏＣ１００が低消費電力状態になっていて高周波発振器１０８が停止している状態やメインメモリ３０が省電力モードに遷移しておりアクセスできない状態であっても、マイコン３００の処理は実行することができる。シリアルバスを介してＳｏＣ１００との間でデータの送受信を行う場合には、ＳｏＣ１００側のシリアルバスコントローラ３０５は動作できる状態にしておかなければならない。割込みコントローラ３０６は、ＨＩＤ２３０等から割込み要求を受信すると、その割込みの種類を識別する割込みベクタを、割込みコントローラ３０６が有するレジスタ（不図示）に登録し、プロセッサコア３０１に割込み信号を送る（割込みを通知する）。プロセッサコア３０１は、割込み信号を受信すると、割込みコントローラ３０６が有するレジスタに登録された割込みベクタを参照して、割込み要因を特定することができる。

【００４９】

再び図１に戻って説明を続ける。割込みコントローラ１２０は、各デバイスコントローラやＩ／Ｏデバイスなどからの割込み要求を受信すると、その割込みの種類を識別する割込みベクタを、割込みコントローラ１２０内のレジスタに登録し、プロセッサコア１０１に割込み信号を送る（割込みを通知する）。プロセッサコア１０１は、割込み信号を受信すると、割込みコントローラ１２０内のレジスタに登録された割込みベクタを参照して割込み要因を特定し、割込みコントローラ１２０内のレジスタに登録された割込みベクタをクリアし、特定した割込み要因に対応する割込みハンドラを実行する。割込み信号を受信した際のプロセッサコア１０１の状態は、プログラムを実行しているラン状態の場合と、割込み待ちを行なっているアイドル状態の場合がある。プロセッサコア１０１がラン状態の時に割込み信号を受信した場合は、プロセッサコア１０１は実行中のプログラムの実行を中断して割込みハンドラを実行する。プロセッサコア１０１がアイドル状態の時に割込み信号を受信した場合は、プロセッサコア１０１はラン状態に復帰してから割込みハンドラを実行する。割込みコントローラ１２０は、ＳｏＣ１００がＤＥＥＰ　ＳＬＥＥＰモードの場合でも低周波発振器１０９が生成するサブクロックで動作しており、起床可能デバイス（ＤＥＥＰ　ＳＬＥＥＰモードのＳｏＣ１００を起床させることができるデバイス）からの割込みを受信すると、プロセッサコア１０１をラン状態に遷移させて、割込み信号をプロセッサコア１０１に伝えることができる。

【００５０】

ＳｏＣ１００内の各種コントローラは、バス１０７を介してプロセッサコア１０１と接続されている。バス１０７にはさまざまな種類のものがあるが、例えばプロセッサコア１０１がＡＲＭの場合はＡＭＢＡ　ＡＸＩバスなどが広く用いられる。ＳｏＣ１００全体のデータ転送能力を向上させるために複数のバスを組み合わせてもよい。メモリコントローラ１０２や内部メモリ１２５は、プロセッサコア１０１が高速でアクセスできるように専用のバスで接続されてもよい。

【００５１】

ＳｏＣ１００の内部メモリ１２５として、高速なＳＲＡＭを採用することもできる。一般的に内部メモリ１２５はメインメモリ３０よりも高速にアクセスできる。内部メモリ１２５はスクラッチパッドメモリとも呼ばれる。メモリコントローラ１０２の初期化が完了しないとアクセスできないメインメモリ３０と異なり、内部メモリ１２５はブート直後からアクセスできるので、ブート処理などにも用いられる。また、ＲＵＮモード（プロセッサコア１０１がラン状態のときの情報処理装置１（ＳｏＣ１００）の状態）からＤＥＥＰ　ＳＬＥＥＰモードへの遷移処理やＤＥＥＰ　ＳＬＥＥＰモードからＲＵＮモードへの復帰処理は、メインメモリ３０がセルフリフレッシュモードになった状態で実行されるコードも含まれるため、遷移処理や復帰処理のコードの全部、もしくは一部にも用いられる。

【００５２】

以上が情報処理装置１のハードウェア構成の説明である。情報処理装置１を構成する各種Ｉ／Ｏデバイスや、ＳｏＣ１００内の各モジュールは、情報処理装置１が使用されているときに常に有効な動作をし続けているわけではない。ＨＩＤ２３０を介したユーザからの入力や通信デバイス（ネットワークデバイス２２０）へのデータ到着を待っていたり、タイマーにより設定された時間に到達するのを待っていたりして、プロセッサコア１０１がプログラムを実行していないアイドル状態になっている期間は多い。そこで、アイドル状態に使用していないデバイスやモジュールをきめ細かく積極的に停止させることで情報処理装置１全体の消費電力を削減することができる。

【００５３】

しかし、プロセッサコア１０１がアイドル状態であっても、入出力処理を実行しているＩ／Ｏデバイスもあれば、何もしていないＩ／Ｏデバイスもあり、Ｉ／Ｏデバイス毎に状態は異なっている。プロセッサコア１０１が割込み待ちのアイドル状態に入る際は、各Ｉ／Ｏデバイス毎に処理を実行中か否かを判別し、それに応じてＳｏＣ１００をＷＡＩＴモードにするかＤＥＥＰ　ＳＬＥＥＰモードにするかを判断したり、クロック供給を継続するか停止するか決めたり、メインメモリ３０を省電力モードにするかどうかを決めたりして、情報処理装置１の正常な動作を妨げない適切な省電力化処理を行わなければならない。これを実現するのが、本発明の消費電力削減方式である。ここでは、ＷＡＩＴモードとは、プロセッサコア１０１がラン状態のときよりも消費電力が小さい状態（情報処理装置１全体の状態あるいはＳｏＣ１００の状態）であると考えることができ、請求項の「第３の状態」に対応している。また、ＤＥＥＰ　ＳＬＥＥＰモードとは、ＷＡＩＴモードよりも消費電力が小さい状態であると考えることができ、請求項の「第４の状態」に対応している。

【００５４】

本発明の消費電力削減方式は、情報処理装置１内の各種Ｉ／ＯデバイスおよびＳｏＣ１００内の各種モジュールの動作状態を、ＳｏＣ１００内のプロセッサコア１０１が実行するオペレーティングシステム（ＯＳ）によって制御することによって実現する。オペレーティングシステムはＮＡＮＤフラッシュメモリやＳＤカードなどの２次記憶装置に記憶されるプログラムであり、そのカーネル（オペレーティングシステムの中核になる部分）が情報処理装置１の電源が投入された直後にメインメモリ３０に読み出され、それをプロセッサコア１０１が実行する。オペレーティングシステムの一部の機能はメインメモリ３０ではなくＳｏＣ１００の内部メモリ１２５に読み込まれて実行されることも多い。例えば、プロセッサコア１０１のアイドル状態を解除する割込みが発生した時にプロセッサコア１０１が最初に実行するプログラムが、アイドル状態の間は省電力モードになっているメインメモリ３０上に格納されている場合、プロセッサコア１０１はすぐには当該プログラムを実行することはできない。そこで、アイドル状態から復帰後、プロセッサコア１０１がすぐにアクセス可能な内部メモリ１２５に、メインメモリ３０を省電力モードから通常モードに変更するプログラムを記憶しておき、プロセッサコア１０１はまずそのプログラムを実行した後、メインメモリ３０上のプログラムを実行する。

【００５５】

情報処理装置１には、Ｌｉｎｕｘ（登録商標）、Ａｎｄｒｏｉｄ、Ｗｉｎｄｏｗｓ、ｉＯＳ、ｉＴｒｏｎなどの様々なオペレーティングシステム（ＯＳ）を使用することができる。図１１は、本実施形態に係る情報処理装置１に搭載されたＯＳの機能構成の一例を示すブロック図である。情報処理装置１に搭載されたＯＳは、主要な機能として、プロセス管理部４１０、デバイス管理部４２０、ファイル管理部４３０、メモリ管理部４４０を有する。

【００５６】

プロセス管理部４１０は、プロセッサコア１０１で実行するアプリケーションプログラムをプロセス（あるいはタスクまたはスレッドなどのプログラムの実行単位）と呼ぶ単位で管理する。情報処理装置１が提供可能な様々な機能を実現するアプリケーションプログラムひとつひとつが、それぞれ１または複数のプロセスによって実行される。プロセス管

理部４１０は、プロセッサコア１０１が実行すべき複数のプロセスを適時切り替えて（ディスパッチして）多重処理を行う。このプロセスの切り替え（コンテクストスイッチとも呼ばれる）を行う機能を、プロセス切替部（あるいはディスパッチャ）４１１と称する。プロセスの切り替えは、プロセスがＯＳに対して入出力（Ｉ／Ｏ）リクエストを発行してその結果を待つときや、プロセッサコア１０１が一定時間そのプロセスの実行を行ったことをタイマーによって検出したときや、Ｉ／Ｏデバイスからの割込み処理の結果によって優先度の高いプロセスが実行可能になったときなどに行う。そのため、プロセス管理部４１０は割込み管理も行う。

【００５７】

本実施形態では、プロセス切替部４１１は、プロセッサコア１０１がアイドル状態に遷移するときに、少なくとも１つのＩ／Ｏデバイスとメインメモリ３０との間でデータ転送のための処理が行われている場合は、ＷＡＩＴモードに情報処理装置１（ＳｏＣ１００）を遷移させる一方、何れのＩ／Ｏデバイスとメインメモリ３０との間でもデータ転送のための処理が行われていない場合は、ＤＥＥＰ　ＳＬＥＥＰモードに情報処理装置１（ＳｏＣ１００）を遷移させる制御を行う。この具体的な内容は後述するが、ここでは、プロセス切替部４１１は、請求項の「状態制御部」に対応していると考えることができる。

【００５８】

また、図１１に示すように、プロセス切替部４１１は、デバイス状態判定部４１２を含む。デバイス状態判定部４１２は、プロセッサコア１０１が現在実行しているプロセスを停止させて他のプロセスに切換え（ディスパッチ）ようとした時点で実行可能なプロセスが無くアイドル状態に入る場合に、ＳｏＣ１００（情報処理装置１）をＷＡＩＴモードにするかＤＥＥＰ　ＳＬＥＥＰモードにするかを、後述のデバイス状態管理部４２１が管理するデバイス状態管理情報に基づいて判断する。

【００５９】

図１１に示すデバイス管理部４２０は、各種Ｉ／Ｏデバイスやタイマーなど、さまざまなデバイスの管理を行う。デバイス管理部４２０は、デバイスの種類ごとにデバイスドライバと呼ばれるソフトウェアモジュールを備え、アプリケーションプログラムやＯＳからのデバイスへの入出力処理要求（Ｉ／Ｏリクエスト）が来ると、そのデバイスに対応するデバイスドライバに入出力処理を実行させる。デバイス管理部４２０は、ネットワークを介した通信を行うために必要なプロトコル処理も行う。

【００６０】

また、デバイス管理部４２０は、デバイス状態管理情報を管理するデバイス状態管理部４２１を含む。デバイス状態管理部４２１は、各デバイスドライバからの通知に基づいて、デバイス状態管理情報を管理する。デバイス状態管理情報は、少なくとも、入出力処理を実行中または入力イベント待機中のデバイスが１つ以上存在するためＳｏＣ１００をＤＥＥＰ　ＳＬＥＥＰモードにすべきではないことを判断するのに必要な情報を保持している。図１２は、デバイス状態管理部４２１が管理するデバイス状態管理情報の一例を示す図である。この実施形態のデバイス状態管理情報は、Ｉ／Ｏデバイスの種類を識別するデバイス識別子と、Ｉ／Ｏデバイスの状態を示す状態情報（図１２の「実行中」を示す状態情報は、Ｉ／Ｏデバイスが入出力処理を実行していることを表し、「停止中」を示す状態情報は、Ｉ／Ｏデバイスが何もしていないことを表す）とを対応付けたテーブル形式の情報である。

【００６１】

各デバイスドライバは、対応するＩ／Ｏデバイスで入出力処理を実行中であることや入力イベントの待機中であること（すなわち、ＳｏＣ１００をＤＥＥＰ　ＳＬＥＥＰモードに遷移させるべきではないこと）をデバイス状態管理部４２１に知らせる。これによって、ＳｏＣ１００がＤＥＥＰ　ＳＬＥＥＰモードになって実行中の入出力処理や入力イベントの待機を継続できなくなることを防ぐ。本実施形態では、各デバイスドライバは、対応するＩ／Ｏデバイスを動作させて入出力の処理を実行中である場合や、入力イベントの待機中である場合には、入出力処理や入力イベント待機処理の開始と終了を直接デバイス状

態管理情報に登録することで、デバイス状態管理部４２１に知らせる。
【００６２】
図１１に示すファイル管理部４３０は、デバイス管理部４２０が管理する２次記憶デバイス（例えばＮＡＮＤフラッシュメモリ、ＳＤカード、ＳＳＤ、ＨＤＤ等）の記憶領域に記憶するデータを、ファイルと呼ぶ単位で管理する。図１１に示すメモリ管理部４４０は、情報処理装置１が備えるメインメモリ３０を有効に使用するために、メインメモリ３０の空間を小さな領域に分けて管理し、ＯＳや個々のプロセスからの要求に合わせてメインメモリ３０の領域の確保／開放を行う。また、メモリ管理部４４０は、仮想記憶の管理も行う。
【００６３】
以降、本発明の消費電力削減方式を実施した情報処理装置１およびオペレーティングシステム（ＯＳ）の動作を詳しく説明する。
【００６４】
まず、図１３を用いて、本実施形態のデバイスドライバの動作例を説明する。図１３は、本実施形態のデバイスドライバの動作例を示すフローチャートである。図１３に示すように、デバイスドライバは、ＯＳやアプリケーションプログラムからの入出力要求を受け取ると（ステップＳ１）、デバイス状態管理情報に含まれる状態情報のうち、対応するＩ／Ｏデバイスを識別するデバイス識別子に対応付けられた状態情報を「実行中」に変更する（ステップＳ２）。次に、デバイスドライバは、要求された入出力処理に必要なハードウェア機能を起動する処理を行う（ステップＳ３）。次に、デバイスドライバは、要求された入出力処理に必要なデバイスドライバ側の処理（たとえばＤＭＡによるデータ転送に用いるメモリ上のバッファ領域の管理など）を行う（ステップＳ４）。次に、デバイスドライバは、起動した入出力処理の完了（割込みで通知される、または、完了を示すフラグをポーリングして検出する）を待ち（ステップＳ５）、必要に応じてハードウェア機能を停止させる処理を行う（ステップＳ６）。そして、デバイスドライバは、デバイス状態管理情報のうち、対応するＩ／Ｏデバイスを識別するデバイス識別子に対応付けられた状態情報を「停止中」に変更し（ステップＳ７）、入出力処理の結果を入出力要求元にリプライする（ステップＳ８）。
【００６５】
なお、本実施形態では、デバイス状態管理部４２１の管理するデバイス状態管理情報はテーブルで実現し、各デバイスドライバが、テーブルに状態情報を直接書き込むようにしているが、これに限らず、例えばデバイス状態管理部４２１が状態情報を登録するための関数を用意して、各デバイスドライバがそれを呼び出すことで間接的にテーブルを更新するように実施しても良い。
【００６６】
ＮＡＮＤフラッシュメモリやＳＤカードなどのストレージデバイスや、Ｉ２ＣやＳＰＩ等のシリアルインタフェースで通信するＩ／Ｏデバイスなどの場合は、図１３に例示したような流れでデバイスドライバの処理を行う。無線ＬＡＮなどのネットワークデバイスの場合は、図１３のように１回の入出力要求毎にＩ／Ｏデバイスの動作を起動／停止するとオーバヘッドが大きくなるので、短い時間内に何回も入出力要求が発生する場合には、ひとつの入出力要求の処理が終わった後も一定時間は、ハードウェア機能を停止せずに、デバイス状態管理情報の状態情報を「動作中」にしておいて、入出力要求が発生しない状態が継続した場合は、ハードウェア機能を停止して、デバイス状態管理情報の状態情報を「停止中」に変更するように実施しても良い。
【００６７】
また、キーパッド（キーボード）のような起床可能デバイスの場合は、ＳｏＣ１００がＤＥＥＰ　ＳＬＥＥＰモードでも入力を監視して、入力が発生すればＳｏＣ１００を起床させることができる。そのため、起床可能デバイスは、デバイス状態管理部４２１で管理しなくても構わない。
【００６８】

ここで、ＯＳは、プロセス切替部４１１によって複数のプロセスを切り替えながら処理を進めるのが一般的である。ＯＳの管理するプロセスは、大きく分けて実行可能状態とイベント待ち状態の２つの状態をとることができ、優先度に従ってキュー（プロセスキュー）を用いて管理されている。プロセッサコア１０１は実行可能状態のプロセスの中から優先度に基づいて選んだプロセスを実行する。プロセスがイベント待ちになるのは、Ｉ／Ｏデバイスに入出力要求を出してその完了を待っている時や、他のプロセスからの通信データの到着を待っている時などである。プロセス切替部４１１によってプロセッサコア１０１が実行するプロセスの切替えが発生する場合としては、次のような３つの場合が考えられる。第１の場合は、現在実行中のプロセスが入出力要求を出した後、その完了を待つためにイベント待ち状態に入る場合である。第２の場合は、Ｉ／Ｏデバイスからの入出力完了の割り込みを受けて、対応するイベント待ち状態のプロセスが実行可能状態になり、かつその優先度が現在実行中のプロセスよりも高い場合である。第３の場合は、特定のプロセスが長時間プロセッサコア１０１を占有しないようにＯＳが一定時間後に割込みを発生させるように設定したタイマーからの割込みが発生したときである。いずれの場合も、ＯＳに入出力要求を出してカーネルモードに入っているときや、割込み信号を受けてカーネルモードで割り込みハンドラが動いているときなど、ＯＳのカーネル内での処理になる。

【００６９】

次に、図１４を用いて、本実施形態のプロセス切替部４１１の動作例を説明する。図１４は、プロセス切替部４１１の動作例を示すフローチャートである。前述したような３つの場合のうちの何れかが発生してプロセス切替えを行うとき、まずプロセス切替部４１１は、現在実行しているプロセスを停止させて（後で実行再開できるように実行状態を保存して）プロセスキューに入れた後、図１４の処理を開始する。図１４に示すように、まずプロセス切替部４１１は、プロセスキューを参照して実行可能状態のプロセスが存在するか否かを調べる（ステップＳ１１）。実行可能状態のプロセスがプロセスキューに存在すれば（ステップＳ１１：ＹＥＳ）、プロセス切替部４１１は、プロセスキューに存在する実行可能状態のプロセスのうち、最も優先度が高いプロセスを選択し、そのプロセスの実行を再開する（ステップＳ１２）。実行可能状態のプロセスがプロセスキューに存在しなければ（ステップＳ１１：ＮＯ）、プロセス切替部４１１のデバイス状態判定部４１２がデバイス状態管理情報を参照し、プロセッサコア１０１がアイドル状態になったときにＳｏＣ１００をＤＥＥＰ　ＳＬＥＥＰモードにして良いかどうかを判断する（ステップＳ１３）。本実施形態では、デバイス状態判定部４１２は、デバイス情報管理情報に含まれるデバイス識別子の中に、対応する状態情報が「実行中」を示すデバイス識別子が１つ以上存在すればＤＥＥＰ　ＳＬＥＥＰモードにすべきではないと判断する。

【００７０】

上述のステップＳ１３において、ＤＥＥＰ　ＳＬＥＥＰモードにすべきではないと判断した場合（ステップＳ１３：ＮＯ）、ＷＦＩ（Wait for Interrupt）命令を発行してプロセッサコア１０１が割り込み待ちのアイドル状態に遷移する際にＳｏＣ１００をＷＡＩＴモードにするかＤＥＥＰ　ＳＬＥＥＰモードにするかを設定するＳｏＣ１００内の制御レジスタに対して、ＷＡＩＴモードにするように設定する（ステップＳ１４）。例えばプロセス切替部４１１は、上記制御レジスタに対して、ＷＡＩＴモードを指定する情報を書き込む形態であってもよい。要するに、プロセス切替部４１１は、プロセッサコア１０１がアイドル状態（割込み待ち状態）に遷移するときに、少なくとも１つのＩ／Ｏデバイスとメインメモリ３０との間でデータ転送のための処理（データ転送の前処理も含む）が行われている場合は、プロセッサコア１０１がラン状態（命令実行状態）のときよりも消費電力が小さいＷＡＩＴモード（第３の状態）にＳｏＣ１００を遷移させる制御を行っていると考えることができる。

【００７１】

上述のステップＳ１４の後、プロセス切替部４１１は、ＷＦＩ命令を発行することでプロセッサコア１０１を割込み待ちのアイドル状態に遷移させる（ステップＳ１５）。Ｉ／Ｏデバイスからの割込み信号によってプロセッサコア１０１がアイドル状態からラン状態

になると割込みハンドラが割込み処理を実行した後、プロセス切替部４１１は、上述のステップＳ１１以降の処理を繰り返す。

【００７２】

一方、上述のステップＳ１３において、ＤＥＥＰ　ＳＬＥＥＰモードにして良いと判断した場合（ステップＳ１３：ＹＥＳ）、プロセス切替部４１１は、プロセッサコア１０１への割込みを禁止にした（ステップＳ１６）後、ＷＦＩ命令を発行してプロセッサコア１０１がアイドル状態に遷移する際にＳｏＣ１００をＷＡＩＴモードにするかＤＥＥＰ　ＳＬＥＥＰモードにするかを設定するＳｏＣ１００内の制御レジスタに対して、ＤＥＥＰ　ＳＬＥＥＰモードにするように設定する（ステップ１７）。例えばプロセス切替部４１１は、上記制御レジスタに対して、ＤＥＥＰ　ＳＬＥＥＰモードを指定する情報を書き込む形態であってもよい。要するに、プロセス切替部４１１は、プロセッサコア１０１がアイドル状態に遷移するときに、少なくとも１つのＩ／Ｏデバイスとメインメモリ３０との間でデータ転送のための処理が行われていない場合は、ＷＡＩＴモードよりも消費電力が小さいＤＥＥＰ　ＳＬＥＥＰモード（第４の状態）にＳｏＣ１００を遷移させる制御を行っていると考えることができる。

【００７３】

上述のステップＳ１７の後、プロセス切替部４１１は、クロック制御モジュール１１０内の制御レジスタに対して、ＤＥＥＰ　ＳＬＥＥＰモードに入っている間は高周波発振器１０８やＰＬＬ１１１の動作を停止させるように設定する（ステップＳ１８）。つまり、プロセス切替部４１１は、ＤＥＥＰ　ＳＬＥＥＰモードでは、プロセッサコア１０１がラン状態のときに用いられるクロックを示す高周波クロック（メインクロック）を生成する高周波発振器１０８の発振を停止させる制御を行うと考えることができる。また、これに限らず、上述のステップＳ１７の後、プロセス切替部４１１は、ＤＥＥＰ　ＳＬＥＥＰモードに入っている間はＰＭＩＣ２０から高周波発振器１０８への電力供給を停止させるように、ＰＭＩＣ２０の制御レジスタを設定することもできる。つまり、プロセス切替部４１１は、ＤＥＥＰ　ＳＬＥＥＰモードでは、プロセッサコア１０１がラン状態のときに用いられるクロックを示す高周波クロック（メインクロック）を生成する高周波発振器１０８への電力供給を停止させる制御を行うこともできる。

【００７４】

なお、ステップＳ１８で設定する制御レジスタの設定値は、ＳｏＣによっては最初に１回設定しておけば、それ以降設定を変更するまで有効になるものも多いので、その場合にはＯＳの初期化の際に実行しておけば、ステップＳ１８の処理は省略できる。

【００７５】

上述のステップＳ１８の後、プロセス切替部４１１は、ＤＥＥＰ　ＳＬＥＥＰモード中のキャッシュメモリ１４０への電力供給の停止に備えて、プロセッサコア１０１のキャッシュメモリ１４０の動作を停止させて（ステップＳ１９）、キャッシュメモリ１４０中のダーティフラグがアクティブレベルに設定されているデータ（プロセッサコア１０１によってキャッシュメモリ１４０には書き込まれたがメインメモリ３０にはまだ書き込まれていないデータ）をメインメモリ３０に退避させた後、ＤＥＥＰ　ＳＬＥＥＰモードに入っている間はＰＭＩＣ２０からキャッシュメモリ１４０への電力供給を停止させるように、ＰＭＩＣ２０の制御レジスタを設定する。つまり、プロセス切替部４１１は、ＤＥＥＰ　ＳＬＥＥＰモードに遷移させる場合は、メインメモリ３０への書き込みが行われていないキャッシュメモリ１４０内のデータ（ダーティなデータ）をメインメモリ３０へ書き出す処理を行った後、キャッシュメモリ１４０への電力供給を停止させる制御を行うと考えることができる。なお、ＤＥＥＰ　ＳＬＥＥＰモード中もキャッシュメモリ１４０に対して電力供給を続けてデータが消えないようにしてもよい。この場合は、上述のステップＳ１９は省略できる。代わりに、プロセス切替部４１１は、ＤＥＥＰ　ＳＬＥＥＰモードに入っている間はキャッシュメモリ１４０に供給する電源電圧の値が、プロセッサコア１０１がラン状態のときの値（あるいは、ＤＥＥＰ　ＳＬＥＥＰモードのときの値よりも高い値であってもよい）になるように、ＰＭＩＣ２０の制御レジスタを設定することもできる。

つまり、プロセス切替部４１１は、第４の状態では、キャッシュメモリ１４０に供給する電源電圧を、プロセッサコア１０１がラン状態のときと同じ値に設定する制御を行うこともできる。

【００７６】

続いて、プロセス切替部４１１は、メインメモリ３０を省電力モードに遷移させる（ステップＳ２０）。メインメモリ３０がＬＰＤＤＲ２等のＤＲＡＭの場合は、メモリコントローラ１０２の制御レジスタを設定してメモリコントローラ１０２からＤＲＡＭに制御コマンドを送ることで、ＤＲＡＭをセルフリフレッシュモードに遷移させる。つまり、メインメモリ３０が揮発性メモリの場合、プロセス切替部４１１は、ＤＥＥＰ　ＳＬＥＥＰモードでは、メインメモリ３０をセルフリフレッシュ状態に遷移させる制御を行うと考えることができる。また、メインメモリ３０が不揮発性メモリの場合は、プロセス切替部４１１は、ＰＭＩＣ２０からメインメモリ３０への電力供給を停止させるように、ＰＭＩＣ２０の制御レジスタを設定することもできる。つまり、メインメモリ３０が不揮発性メモリの場合、プロセス切替部４１１は、ＤＥＥＰ　ＳＬＥＥＰモードでは、メインメモリ３０への電力供給を停止させる制御を行うと考えることができる。

【００７７】

次に、プロセス切替部４１１は、ＷＦＩ命令を発行することでプロセッサコア１０１を割込み待ちのアイドル状態に遷移させる（ステップＳ２１）。その後、Ｉ／Ｏデバイスからの割込み信号によってプロセッサコア１０１がアイドル状態からラン状態になると、まずキャッシュメモリ１４０のすべてのキャッシュラインに対してデータが無効であることを示すインバリデートフラグを立てるインバリデート処理を行ってキャッシュメモリの初期化を行った後、キャッシュメモリ１４０を動作させる（ステップＳ２２）。次に、メインメモリ３０を省電力モードから通常の動作モードに遷移させる（ステップＳ２３）。メインメモリ３０がＤＲＡＭの場合は、メモリコントローラ１０２の制御レジスタを設定してメモリコントローラ１０２からＤＲＡＭに制御コマンドを送ることで、ＤＲＡＭのセルフリフレッシュモードを解除して通常モードに遷移させる。メインメモリ３０が不揮発性メモリの場合は、ＰＭＩＣ２０からメインメモリへの電力供給を再開するように、ＰＭＩＣ２０の制御レジスタを設定する。その後、プロセッサコア１０１への割込みを可能に設定する（ステップＳ２４）と、割込みハンドラが割り込み処理を実行した後、プロセス切替部４１１は、上述のステップＳ１１以降の処理を繰り返す。

【００７８】

なお、プロセス切替部４１１のプログラムのうち、メインメモリ３０が省電力モードになっている間に実行すべき処理に対応するプログラムは、メインメモリ３０が省電力モードの間もアクセス可能なＳｏＣ１００の内部メモリ１２５や省電力モードに遷移していない第２のメインメモリなどにロードしておかなければならない。なお、図１４のステップＳ１３の結果が否定（ステップＳ１３：ＮＯ）でＳｏＣ１００をＷＡＩＴモードにする場合でも、ＤＥＥＰ　ＳＬＥＥＰモードにする場合と同様にキャッシュメモリ１４０を停止させてもよいし、メインメモリ３０にアクセスするＩ／Ｏデバイスが動作中でないならばメインメモリ３０を省電力モードにするようにしてもよい。

【００７９】

上述したように、ＷＦＩ命令の発行によりプロセッサコア１０１がアイドル状態に遷移するのに伴い、ＳｏＣ１００は、図１５に示す手順に従って、ＷＡＩＴモードあるいはＤＥＥＰ　ＳＬＥＥＰモードに遷移する処理を行う。まず、クロック制御モジュール１１０は、プロセッサコア１０１のクロックを停止（クロックゲーティング）する（ステップＳ３１）。次に、クロック制御モジュール１１０は、ＳｏＣ１００をＷＡＩＴモードにするかＤＥＥＰ　ＳＬＥＥＰモードにするかをクロック制御モジュール１１０内の制御レジスタを参照して判断する（ステップＳ３２）。この例では、ＳｏＣ１００をＷＡＩＴモードにするかＤＥＥＰ　ＳＬＥＥＰモードにするかはクロック制御モジュール１１０内の制御レジスタで設定されている。また、ＤＥＥＰ　ＳＬＥＥＰモードの間における高周波発振器１０８やＰＬＬ１１１の状態も、クロック制御モジュール１１０内の制御レジスタで設

定されている。ただし、これに限られるものではない。要するに、ＳｏＣ１００は、ＷＡＩＴモードあるいはＤＥＥＰ　ＳＬＥＥＰモードに遷移する際のクロックを制御するクロック制御部を有する形態であればよく、この例では、クロック制御モジュール１１０が上記クロック制御部として機能する場合を例に挙げて説明するが、これに限られるものではない。

【００８０】

上述のステップＳ３２において、ＤＥＥＰ　ＳＬＥＥＰモードにすると判断した場合（ステップＳ３２：ＹＥＳ）、クロック制御モジュール１１０は、ＤＥＥＰ　ＳＬＥＥＰモードの間は高周波発振器１０８やＰＬＬ１１１を停止するように制御レジスタが設定されているかどうかを調べ（ステップＳ３３）、設定されている場合（ステップＳ３３：ＹＥＳ）は、高周波発振器１０８やＰＬＬ１１１を停止する（ステップＳ３４）。その後、クロック制御モジュール１１０は、ＳｏＣ１００がＤＥＥＰ　ＳＬＥＥＰモードに入ったことを外部に伝えるための出力信号（ＳＴＢＹ信号などと呼ばれる）をＨＩＧＨ（あるいはＥｎａｂｌｅ）にして（ステップＳ３５）、ＳｏＣ１００がＤＥＥＰ　ＳＬＥＥＰモードに入る処理は完了する。一方、ステップＳ３２において、ＤＥＥＰ　ＳＬＥＥＰモードにしないと判断した場合（ステップＳ３２：ＮＯ）は、その時点でＳｏＣ１００がＷＡＩＴモードに入る処理は完了する。

【００８１】

図１６に示すように、ＳｏＣ１００は、ＳｏＣ１００がＤＥＥＰ　ＳＬＥＥＰモードであるか否かを示すＳＴＢＹ信号（例えばＤＥＥＰ　ＳＬＥＥＰモードならばＳＴＢＹ信号をハイレベル（ＨＩＧＨ）に設定する）をＰＭＩＣ２０へ供給し、ＳｏＣ１００とＰＭＩＣ２０が連携して動作することでＤＥＥＰ　ＳＬＥＥＰ時の電力供給を制御することができる。例えばＰＭＩＣ２０は、内蔵するＤＣ／ＤＣコンバータに対して、ＳＴＢＹ信号がハイレベルのときに出力する電圧と、ローレベル（ＬＯＷ）のときに出力する電圧とを設定することができる。

【００８２】

ＳｏＣ１００がＤＥＥＰ　ＳＬＥＥＰモードに入ってＳＴＢＹ信号がハイレベルのときにＰＭＩＣ２０がプロセッサコア１０１に供給する電圧を０に設定すると、ＤＥＥＰ　ＳＬＥＥＰモード中の電力供給を停止することができる。ＳｏＣ１００内の起床可能でないデバイスや不揮発のメインメモリに電力を供給するＤＣ／ＤＣコンバータは、この設定に応じた電圧供給を行うことで、ＤＥＥＰ　ＳＬＥＥＰモード中の消費電力を下げることができる。また、ＳＴＢＹ信号がハイレベルのときにＰＭＩＣ２０がプロセッサコア１０１に供給する電圧を回路中のレジスタの値を保持できる最低電圧に設定することで、回路の状態を保持したまま消費電力を下げることもできる。プロセッサコア１０１へ電力を供給するＤＣ／ＤＣコンバータは、この設定に応じた電圧供給を行うことで、状態を保持したままＤＥＥＰ　ＳＬＥＥＰ中の消費電力を下げることができる。要するに、ＤＥＥＰ　ＳＬＥＥＰモードでは、プロセッサコア１０１に供給される電源電圧の値は、プロセッサコア１０１がラン状態のときよりも低い値（例えばＷＡＩＴモードのときよりも低い値であってもよい）に設定される形態であればよい。見方を変えれば、上述したように、プロセス切替部４１１は、プロセッサコア１０１がアイドル状態に遷移するときに、少なくとも１つのＩ／Ｏデバイスとメインメモリ３０との間でデータ転送のための処理が行われていない場合は、ＳｏＣ１００内の制御レジスタに対して、ＤＥＥＰ　ＳＬＥＥＰモードにする設定を行うことで、プロセッサコア１０１に供給する電圧を下げる制御を行っていると考えることもできる。なお、キャッシュメモリ１４０に電力を供給するＤＣ／ＤＣコンバータの、ＳＴＢＹ信号がハイレベルのときの電圧は、ＤＥＥＰ　ＳＬＥＥＰモードに入る前にキャッシュメモリ１４０中のダーティなデータをメインメモリ３０に書き込む場合は０に設定する一方、ＤＥＥＰ　ＳＬＥＥＰモード中もデータが消えないように保持したい場合は、データを保持するのに必要な最低電圧に設定する。

【００８３】

続いて、アイドル状態のプロセッサコア１０１が割込み信号によってラン状態に復帰す

る際のＳｏＣ１００の動作例を図１７に示す。この例では、クロック制御モジュール１１０が、ＳｏＣ１００の状態を管理する状態管理部として機能する場合を例に挙げて説明するが、これに限られるものではない。例えばクロック制御モジュール１１０とは別に状態管理部が設けられる形態であってもよい。

【００８４】

図１７に示すように、まず割り込みイベントを待っているＳｏＣ１００がＩ／Ｏデバイスやタイマーなどからの割込み信号を割込みコントローラ１２０で検出すると（ステップＳ４１）、クロック制御モジュール１１０は、制御レジスタを参照して、ＳｏＣ１００がＤＥＥＰ　ＳＬＥＥＰモードであるかＷＡＩＴモードであるかを調べる（ステップＳ４２）。ＤＥＥＰ　ＳＬＥＥＰモードである場合（ステップＳ４２：ＹＥＳ）、クロック制御モジュール１１０は、ＳＴＢＹ信号をローレベルに設定する（ステップＳ４３）。これにより、ＰＭＩＣ２０は、通常よりも低い値にしていたＤＣ／ＤＣコンバータの出力電圧を通常の値に戻す。

【００８５】

次に、クロック制御モジュール１１０は、ＤＥＥＰ　ＳＬＥＥＰモードで高周波発振器１０８やＰＬＬ１１１を停止するように制御レジスタが設定されているかどうかを調べ（ステップＳ４４）、ＤＥＥＰ　ＳＬＥＥＰモードで高周波発振器１０８やＰＬＬ１１１を停止するように制御レジスタが設定されている場合（ステップＳ４４：ＹＥＳ）、クロック制御モジュール１１０は、高周波発振器１０８やＰＬＬ１１１の動作を再開する（ステップＳ４５）。その後、ＳｏＣ１００がＤＥＥＰ　ＳＬＥＥＰモードであった場合でもＷＡＩＴモードであった場合でも、クロック制御モジュール１１０は、プロセッサコア１０１へのクロック供給を再開し（ステップＳ４６）、プロセッサコア１０１をアイドル状態からラン状態に遷移させる（ステップＳ４７）。その後、ラン状態に遷移したプロセッサコア１０１は動作を再開し、プロセッサコア１０１がアイドル状態であった期間においてＳｏＣ１００がＷＡＩＴモードであった場合は、すぐに割込みハンドラが割り込み処理の実行を開始する。一方、プロセッサコア１０１がアイドル状態であった期間においてＳｏＣ１００がＤＥＥＰ　ＳＬＥＥＰモードであった場合は、内部メモリ１２５上のプログラムによってキャッシュメモリ１４０の初期化とメインメモリ３０の動作再開が行われた後に割込み可能状態になって、割込みハンドラが割り込み処理の実行を開始する。

【００８６】

以上に説明したように、本実施形態では、プロセッサコア１０１がアイドル状態に遷移するときに、少なくとも１つのＩ／Ｏデバイスとメインメモリ３０との間でデータ転送のための処理が行われている場合は、ＷＡＩＴモードに遷移する一方、何れのＩ／Ｏデバイスとメインメモリ３０との間でもデータ転送のための処理が行われていない場合は、ＤＥＥＰ　ＳＬＥＥＰモードに遷移する。これにより、ＤＥＥＰ　ＳＬＥＥＰモードに移行するためのサスペンド処理のオーバヘッドを防ぐことができるので、情報処理装置１の性能を損なうことなく省電力化を適切に行うことができる。

【００８７】

（第２実施形態）

第２実施形態では、デバイス状態管理情報として、クロック制御モジュール１１０の制御レジスタ１１５を利用する。図８に例示したように、クロック制御モジュール１１０はクロックゲート部１１３を制御することで、その先に接続される各デバイスコントローラへのクロック供給をＯＮ／ＯＦＦできる。そのため、クロック制御モジュール１１０の制御レジスタ１１５を参照すれば、各デバイスコントローラへのクロック供給が行われているか判別できる。つまり、各デバイスコントローラに対応するＩ／Ｏデバイスが処理を実行中か否かを判定できる。

【００８８】

そこで、クロック制御モジュール１１０の制御レジスタ１１５に設定されているクロック供給のＯＮ／ＯＦＦを決める情報を、デバイス状態管理情報として用いる。この場合、クロック制御モジュール１１０が上述のデバイス状態管理部４２１の役割を果たす。図１

３において、各デバイスドライバは、Ｉ／Ｏデバイスが処理を開始する前のステップＳ２において、クロック制御モジュール１１０からクロックを供給するように制御レジスタを設定し、Ｉ／Ｏデバイスが処理を終えた後のステップＳ７において、クロック制御モジュール１１０からのクロック供給を停止するように制御レジスタを設定する。このように実装することによって、クロック制御モジュール１１０の制御レジスタ１１５を参照すれば、Ｉ／Ｏデバイスが処理を実行中か否かを判定できる。デバイス状態判定部４１２は、クロック制御モジュール１１０の制御レジスタ１１５を参照することで、制御レジスタがクロックを供給する設定になっている場合は、Ｉ／Ｏデバイスが「処理中」であると判定できる一方、制御レジスタがクロックを停止する設定になっている場合は、Ｉ／Ｏデバイスが「停止中」であると判定できる。

【００８９】

（第３実施形態）

第３実施形態では、デバイス状態管理情報として、デバイスコントローラやデバイスのステータスレジスタを利用する。デバイスコントローラやデバイスの中には、Ｉ／Ｏデバイスが処理を実行中であるか否かを示すステータスレジスタを持つものがある。こうしたステータスレジスタは、例えばＩ／Ｏデバイスが処理を実行中である場合は「１」、処理を実行中ではない場合は「０」に設定される。そのため、デバイス状態判定部４１２は、デバイスコントローラやデバイスのステータスレジスタを参照することで、Ｉ／Ｏデバイスが処理を実行中であるか否かを判定できる。第３実施形態では、各デバイスコントローラやデバイスが上述のデバイス状態管理部４２１の役割も兼ねることになる。

【００９０】

（第４実施形態）

第４実施形態では、デバイス状態管理情報としてＰＭＩＣ２０の制御レジスタの設定情報を利用する。ＰＭＩＣ２０は、各電源ラインのＤＣ／ＤＣコンバータの設定用の制御レジスタを有しており、この制御レジスタを参照すれば、どの電源ラインに接続されたＩ／Ｏデバイスに電力を供給しているかを判別できる。そこで、デバイスドライバは、対応するＩ／Ｏデバイスが処理を実行する前にＰＭＩＣ２０の制御レジスタを操作して、Ｉ／Ｏデバイスへの電源供給が行われるように、ＰＭＩＣ２０の制御レジスタを設定し、デバイスが処理を終了した時点で設定用レジスタを操作して、Ｉ／Ｏデバイスへの電源供給が停止するように、ＰＭＩＣ２０の制御レジスタを設定する。デバイス状態判定部４１２は、ＰＭＩＣ２０の設定用の制御レジスタを参照することで、どのＩ／Ｏデバイスが処理を実行中であるかを判別することができる。

【００９１】

（第５実施形態）

第５実施形態では、オペレーティングシステムとしてＬｉｎｕｘを利用し、デバイスドライバがＲｕｎｔｉｍｅＰＭのフレームワークに基づいて実装されている。ＲｕｎｔｉｍｅＰＭのフレームワークに基づいて実装されているデバイスドライバでは、対応するＩ／Ｏデバイスが入出力処理を開始する前に必要な処理（ハードウェアの起動や初期化など）、およびＩ／Ｏデバイスが入出力処理を終了した後に必要な処理（ハードウェアの停止など）を、それぞれ処理前コールバック関数（ｒｕｎｔｉｍｅ＿ｒｅｓｕｍｅ（）コールバック）、および処理後コールバック関数（ｒｕｎｔｉｍｅ＿ｓｕｓｐｅｎｄ（）コールバック）として実装しておくと、Ｉ／Ｏデバイスの入出力処理の実行の前後の適切なタイミングで対応するコールバック関数が呼び出される。Ｉ／Ｏデバイスが入出力処理を開始する前に必要な処理とは、例えばＩ／Ｏデバイスに対応するデバイスコントローラにクロック供給を開始する処理や、Ｉ／Ｏデバイスへの電力供給を開始するためにＰＭＩＣ２０の制御レジスタの値を設定する処理などである。また、Ｉ／Ｏデバイスが入出力処理を終了した後に必要な処理とは、例えばデバイスコントローラへのクロック供給を停止したり電力供給を停止したりする処理である。

【００９２】

図１８に示すように、ＲｕｎｔｉｍｅＰＭでは、Ｉ／Ｏデバイスの状態は、ＡＣＴＩＶ

Ｅ、ＳＵＳＰＥＮＤＩＮＧ、ＳＵＳＰＥＮＤＥＤ、ＲＥＳＵＭＩＮＧの４つの状態のいずれかで表現される。ＡＣＴＩＶＥは、Ｉ／Ｏデバイスが処理を実行中であることを示し、ＳＵＳＰＥＮＤＩＮＧは、デバイスが処理を実行中から停止中に変化させている過渡期であることを示し、ＳＵＳＰＥＮＤＥＤは、デバイスが処理を停止中であることを示し、ＲＥＳＵＭＩＮＧは、Ｉ／Ｏデバイスが処理を停止中から実行中に変化させている過渡期であることを示す。

【００９３】

デバイスドライバは初期化処理を終えると、対応するＩ／Ｏデバイスの状態をＳＵＳＰＥＮＤＥＤにする。そして、処理の実行を開始する前に処理前コールバック関数を呼び出す。処理前コールバック関数が呼び出されると、Ｉ／Ｏデバイスの状態はＲＥＳＵＭＩＮＧに遷移し、処理前コールバック関数を実行中はＩ／Ｏデバイスの状態はＲＥＳＵＭＩＮＧに保持され続ける。そして、処理前コールバック関数の実行が終了するとＩ／Ｏデバイスの状態はＡＣＴＩＶＥに遷移する。続いて、デバイスドライバは、Ｉ／Ｏデバイスの処理の実行が終了すると、処理後コールバック関数を呼び出す。処理後コールバック関数が呼び出されると、Ｉ／Ｏデバイスの状態はＡＣＴＩＶＥからＳＵＳＰＥＮＤＩＮＧに遷移し、処理後コールバック関数を実行中はＩ／Ｏデバイスの状態はＳＵＳＰＥＮＤＩＮＧに保持され続ける。そして、処理後コールバック関数の実行が終了するとＩ／Ｏデバイスの状態はＳＵＳＰＥＮＤＥＤに遷移する。ＲｕｎｔｉｍｅＰＭでは、以上のようなＩ／Ｏデバイスの状態を、各Ｉ／Ｏデバイスを管理する構造体の中のｒｕｎｔｉｍｅ＿ｓｔａｔｕｓという変数で管理している。

【００９４】

本実施形態では、上記処理前コールバック関数および上記処理後コールバック関数が呼び出されるタイミングは、図１３のステップＳ２（デバイス状態管理情報への動作開始登録）およびステップＳ７（デバイス状態管理情報への動作終了登録）と同じである。そこで、例えば上記処理前コールバック関数の中で、図１３のステップＳ２のデバイス状態管理情報への動作開始の登録処理（対応するＩ／Ｏデバイスの状態情報を「実行中」に変更する処理）を行い、上記処理後コールバック関数の中で、図１３のステップＳ７のデバイス状態管理情報への動作終了の登録処理（対応するＩ／Ｏデバイスの状態情報を「停止中」に変更する処理）を行うようにしておくこともできる。こうすると、図１３のステップＳ２とステップＳ７を省略した手順で動くデバイスドライバでも、適切なタイミングでデバイス状態管理情報の更新を行うことができる。

【００９５】

（第６実施形態）

第６実施形態では、オペレーティングシステムとしてＬｉｎｕｘを利用し、デバイスドライバがＲｕｎｔｉｍｅＰＭのフレームワークに基づいて実装されている場合において、ＲｕｎｔｉｍｅＰＭが管理しているＩ／Ｏデバイスの状態（各Ｉ／Ｏデバイスを管理する構造体中のｒｕｎｔｉｍｅ＿ｓｔａｔｕｓ変数に記録している）を、図１２に示すデバイス状態管理情報の状態情報として用いる。この場合、「ＡＣＴＩＶＥ」、「ＳＵＳＰＥＮＤＩＮＧ」、および、「ＲＥＳＵＭＩＮＧ」は、Ｉ／Ｏデバイスが入出力処理を実行していることを表し、「ＳＵＳＰＥＮＤＥＤ」は、Ｉ／Ｏデバイスが何もしていないことを表す。つまり、「ＡＣＴＩＶＥ」、「ＳＵＳＰＥＮＤＩＮＧ」、および、「ＲＥＳＵＭＩＮＧ」は、図１２に示す「実行中」に相当し、「ＳＵＳＰＥＮＤＥＤ」は、図１２に示す「停止中」に相当する。

【００９６】

本実施形態では、デバイスドライバは、ＲｕｎｔｉｍｅＰＭのフレームワークに基づいて上記処理前コールバック関数でハードウェアの起動や初期化を行い、上記処理後コールバック関数でハードウェアの停止を行うように実装されていればよく、図１３のステップＳ２やステップＳ７のようなデバイス状態管理情報の更新処理は不要となる。

【００９７】

本実施形態では、デバイス状態判定部４１２は、ＲｕｎｔｉｍｅＰＭに対応しているデ

バイスドライバの保持しているＩ／Ｏデバイスの状態（ｒｕｎｔｉｍｅ＿ｓｔａｔｕｓ）を参照し、「ＳＵＳＰＥＮＤＥＤ」の場合はＩ／Ｏデバイスが停止中であると判定し、ＳＵ「ＳＰＥＮＤＥＤ」以外の場合はＩ／Ｏデバイスが処理を実行中であると判定する。デバイス状態判定部４１２は、カーネルの初期化時にあらかじめ判定対象とするデバイスドライバを登録しておき、図１４のステップＳ１３にて判定対象とする全てのＩ／ＯデバイスのＲｕｎｔｉｍｅＰＭで管理している状態が「ＳＵＳＰＥＮＤＥＤ」ならば、ＤＥＥＰ　ＳＬＥＥＰモードにして良いと判定し（ステップＳ１３：ＹＥＳ）、そうでなければ（ステップＳ１３：ＮＯ）、ＤＥＥＰ　ＳＬＥＥＰモードにしてはいけないと判定する。

【００９８】

（第７実施形態）

第７実施形態では、電源装置１０内の蓄電装置が十分な電力量を保持している場合は、ＤＥＥＰ　ＳＬＥＥＰモードに遷移できる場合でもＷＡＩＴモードに遷移させることで、応答時間を高めている。第７実施形態では、蓄電装置に蓄積された電力量を検出する電力量検出装置（不図示）をさらに備えている。

【００９９】

第７実施形態では、図１９に示すように、プロセス切替え（ディスパッチ）の手順において、図１４に示すステップＳ１３の前に（ＳｏＣ１００をＷＡＩＴモードにするかＤＥＥＰ　ＳＬＥＥＰモードにするかを判定する前に）、電力量検出装置によって検出された電力量（蓄電装置に蓄積されている電力量）が予め決めておいた値（閾値）以上であるか否かを判断する処理（ステップＳ２５）が追加されている。ステップＳ２５において、プロセス切替部４１１は、電力量検出装置によって検出された電力量が閾値以上であると判断した場合（ステップＳ２５：ＹＥＳ）は、必ずＷＡＩＴモードに遷移するように設定する。

【０１００】

上記電力量検出装置は、電源装置に使われている一次電池や二次電池、およびエナジーハーベスト装置と組み合わせる蓄電装置が蓄積している電力量を検出する装置である。たとえば、一次電池や二次電池、および、エナジーハーベスト装置と組み合わせる電力蓄積装置として用いる電気二重層キャパシタやリチウムイオンキャパシタなどのキャパシタは、その出力電圧を計測することで蓄積している電力量を知ることができるので、Ａ／Ｄコンバータを電力量検出装置として用いることができる。また、蓄電装置としてリチウムイオン電池などのバッテリを用いる場合は、クーロンカウンタを電力量検出装置として用いることができる。つまり、バッテリの充放電量をクーロンカウンタで計測することで蓄積している電力量を知ることができる。

【０１０１】

電力量検出装置は、Ｉ２ＣやＳＰＩやＧＰＩＯなどでＳｏＣ１００と接続してデータを読み出すことができる。また、電力量検出装置を上述のマイコンに接続し、マイコンで読み取った電力量を定期的にＳｏＣ１００に送信するように実施してもよいし、電力量が閾値を下回る場合のみマイコンからＳｏＣ１００に送信するように実施してもよい。また、ＳｏＣ１００が、定期的にマイコンが保持している電力量を読み込むように実施してもよい。

【０１０２】

（第８実施形態）

次に、第８実施形態を説明する。以下の説明では、上述の各実施形態と共通する部分については適宜に説明を省略する。第８実施形態では、本発明が適用される情報処理装置は、太陽電池で発電した電力で動作するタブレット型の情報処理装置である。図２０は、第８実施形態の情報処理装置１の外観を示す図である。このタブレット型の情報処理装置１は、端末表面に低消費電力の反射型液晶ディスプレイあるいは電子ペーパーを表示装置５００として備え、端末表面の表示装置５００以外の部分に太陽電池５０１を備えている。表示装置５００の表面にはポインティングデバイスとして機能するタッチパネル５０２を供えるとともに、端末表面の表示装置５００と重ならない部分にキーボード５０３を備え

る。キーボード５０３は、太陽電池５０１の表面に透明なタッチパネル５０２を重ねることで実現することもできるし、また、透明な素材あるいは遮光性の部分の少ない素材を用いた機械式のキーボードとして実現することもできる。

【０１０３】

図２１および図２２は第８実施形態の情報処理装置１の構成の一例を示す図である。図２１は、情報処理装置１を構成する部品（モジュール）間の電力供給線の接続関係を示す。図２２は情報処理装置１を構成する部品（モジュール）間の信号線の接続関係を示す。

【０１０４】

図２１および図２２に示すように、情報処理装置１は、太陽電池５０１と、蓄電制御装置５１１と、キャパシタ（蓄電装置）５１２と、ＰＭＩＣ２０と、ＤＣ／ＤＣコンバータ５１３および５１４と、シリコンオシレータ５１５と、ＤＲＡＭ５１６と、ＳｏＣ５１７と、マイコン３００と、ＮＡＮＤフラッシュメモリ５１８と、無線ＬＡＮモジュール５１９と、電子ペーパー（ＥＰＤ）５２０と、タッチパネル５０２と、キーボード５０３とを備える。ここでは、太陽電池５０１、蓄電制御装置５１１、キャパシタ５１２の組み合わせは、前述の電源装置１０の一例であると考えることができる。また、シリコンオシレータ５１５は、前述の高周波発振器１０８の一例であると考えることができる。また、ＤＲＡＭ５１６は、前述のメインメモリ３０の一例であると考えることができる。また、ＮＡＮＤフラッシュメモリ５１８は、前述のストレージデバイス２１０の一例であると考えることができる。また、無線ＬＡＮモジュール５１９は、前述のネットワークデバイス２２０の一例であると考えることができる。また、電子ペーパー５２０は、前述の表示デバイス２００の一例であると考えることができる。さらに、タッチパネル５０２およびキーボード５０３は、前述のＨＩＤ２３０の一例であると考えることができる。

【０１０５】

また、ＳｏＣ５１７は、内部モジュールとして、プロセッサコア１０１、ＤＲＡＭコントローラ５３０、ＥＰＤコントローラ５４０、ＮＡＮＤコントローラ５５０、ＳＤＩＯコントローラ５６０、Ｉ２Ｃコントローラ５７０、ＧＰＩＯコントローラ５８０、クロック制御モジュール１１０、割込みコントローラ１２０、ＲＡＭ５９０、バス１０７などを有する。また、ここでは、図示を省略しているが、第１実施形態と同様に、ＳｏＣ５１７は、内部モジュールとして、計時機能を有するＲＴＣ（Real Time Clock）も有している。この例では、ＤＲＡＭコントローラ５３０は、前述のメモリコントローラ１０２の一例であると考えることができる。ＥＰＤコントローラ５４０は、前述の表示デバイスコントローラ１０３の一例であると考えることができる。ＮＡＮＤコントローラ５５０は、前述のストレージコントローラ１０４の一例であると考えることができる。ＳＤＩＯコントローラ５６０は、前述のネットワークコントローラ１０５の一例であると考えることができる。ＲＡＭ５９０は、前述の内部メモリ１２５の一例であると考えることができる。

【０１０６】

ここでは、詳細な図示を省略しているが、第１実施形態と同様に、割込みコントローラ１２０は、クロック制御モジュール１１０、プロセッサコア１０１、ＤＲＡＭコントローラ５３０、ＥＰＤコントローラ５４０、ＮＡＮＤコントローラ５５０、ＳＤＩＯコントローラ５６０、Ｉ２Ｃコントローラ５７０、ＧＰＩＯコントローラ５８０、ＲＡＭ５９０、および、不図示のＲＴＣの各々と、バス１０７とは異なる専用線で接続されている。同様に、クロック制御モジュール１１０は、割込みコントローラ１２０、プロセッサコア１０１、ＤＲＡＭコントローラ５３０、ＥＰＤコントローラ５４０、ＮＡＮＤコントローラ５５０、ＳＤＩＯコントローラ５６０、Ｉ２Ｃコントローラ５７０、ＧＰＩＯコントローラ５８０、ＲＡＭ５９０、および、不図示のＲＴＣの各々と、バス１０７とは異なる専用線で接続されている。

【０１０７】

まず、図２１を用いて、本実施形態の情報処理装置１の電力供給系統を説明する。本実施形態の情報処理装置１は太陽電池５０１で発電した電力で動作する。しかし太陽電池５０１で発電する電力だけでは動作中の情報処理装置１全体のピークの消費電力を賄えない

。そこで、蓄電制御装置５１１がキャパシタ（蓄電装置）５１２を用いてピークアシスト制御を行う。図２１に示すように、太陽電池５０１の出力は蓄電制御装置５１１に接続されている。蓄電制御装置５１１には、さらにキャパシタ５１２が接続されている。蓄電制御装置５１１は、太陽電池５０１とキャパシタ５１２を使ってピークアシストを行い、必要に応じてその出力電圧を、ＰＭＩＣ２０やＤＣ／ＤＣコンバータ５１３、５１４の入力電圧に合わせて変換するための電源回路である。ＳｏＣ５１７がＷＡＩＴモードやＤＥＥＰ　ＳＬＥＥＰモードなどの低消費電力で待機するモードになっていて、太陽電池５０１で発電している電力だけで情報処理装置１全体の消費電力を賄えている間は、蓄電制御装置５１１は余剰電力をキャパシタ５１２に充電しておく。太陽電池５０１で発電している電力だけで情報処理装置１全体の消費電力を賄えない場合は、蓄電制御装置５１１はキャパシタ５１２に蓄積している電力で補って必要な電力を出力する。

【０１０８】

蓄電装置として用いるキャパシタ５１２としては、電気２重層キャパシタやリチウムイオンキャパシタなどの大容量キャパシタを使用できる。また、蓄電装置として、キャパシタではなくリチウムイオン電池などの電池を用いることもできる。また、電池とキャパシタを組み合わせて用いることもできる。

【０１０９】

図２１に示すように、蓄電制御装置５１１の出力はＰＭＩＣ２０、ＤＣ／ＤＣコンバータ５１３、および、ＤＣ／ＤＣコンバータ５１４の各々に接続されている。ＰＭＩＣ２０は、内部に複数のＤＣ／ＤＣコンバータやＬＤＯレギュレータなどを備えており、入力した電力を様々な電圧に変換して出力する。ＰＭＩＣ２０は、ＰＭＩＣ２０が有する制御レジスタの設定によって、各ＤＣ／ＤＣコンバータやＬＤＯレギュレータ毎に、通常時の出力電圧の設定変更や、スタンバイ時の出力電圧の設定変更などを行えるようになっている。ＰＭＩＣ２０の出力は、シリコンオシレータ５１５、ＤＲＡＭ５１６、ＳｏＣ５１７、ＮＡＮＤフラッシュメモリ５１８、無線ＬＡＮモジュール５１９に接続され、それらの動作に必要な電力を供給している。

【０１１０】

ＰＭＩＣ２０は、ＳｏＣ５１７がスタンバイ状態（ＤＥＥＰ　ＳＬＥＥＰモード）に入るとシリコンオシレータ５１５への電力供給を停止し、ＳｏＣ５１７がスタンバイ状態（ＤＥＥＰ　ＳＬＥＥＰモード）から抜けるとシリコンオシレータ５１５への電力供給を再開することで、シリコンオシレータ５１５の消費電力を削減することができる。ＤＲＡＭ５１６は、メモリアレイとインタフェース回路で異なる電圧の電力を必要とするので、それに合わせてＰＭＩＣ２０からも２種類の電圧の出力を接続する。ＳｏＣ５１７も内部で複数（図２１の例では４種類）の電圧の電力を必要とするので、それに合わせてＰＭＩＣ２０からも複数の電圧の出力を接続する。なお、複数の異なる部品（モジュール）が同じ電圧の電力を必要とするときは、ＰＭＩＣ２０内の１つのＤＣ／ＤＣコンバータあるいはＬＤＯレギュレータの出力をそれら複数の部品（モジュール）に接続しても構わない。

【０１１１】

蓄電制御装置５１１の出力が接続されているＤＣ／ＤＣコンバータ５１４（以下の説明では、「第２のＤＣ／ＤＣコンバータ５１４」と称する場合がある）は、マイコン３００に電力を供給する。マイコン３００はＰＭＩＣ２０を制御して情報処理装置１全体の電源のＯＮ／ＯＦＦを行うため、マイコン３００の動作に必要な電力はＰＭＩＣ２０とは独立したＤＣ／ＤＣコンバータ５１４から供給する。

【０１１２】

蓄電制御装置５１１の出力が接続されているＤＣ／ＤＣコンバータ５１３（以下の説明では、「第１のＤＣ／ＤＣコンバータ５１３」と称する場合がある）は、電子ペーパー５２０に電力を供給する。電子ペーパー５２０の動作に必要な電圧の電力をＰＭＩＣ２０から供給できる場合は、第１のＤＣ／ＤＣコンバータ５１３を用いずに、ＰＭＩＣ２０から電子ペーパー５２０へ電力を供給するように実施しても良い。

【０１１３】

なお、本実施形態では、電力供給源として太陽電池５０１を用いているが、太陽電池５０１以外の振動や熱などのエネルギーハーベストによる発電装置を用いても構わない。また、例えば電力供給源として無線給電を用いることもできる。さらに、例えば電力供給源として乾電池を用い、キャパシタと組み合わせてピークアシストするように実施することもできる。

【０１１４】

次に、図２２を用いて、本実施形態の情報処理装置１を構成する各部品（モジュール）と、それらの間の信号の接続関係を説明する。情報処理装置１の主要構成部品であるＳｏＣ５１７は、内部にコンピュータシステムを集積した半導体チップである。ＳｏＣ５１７を動作させるためのクロック信号として、２４ＭＨｚのメインクロック（上述の高周波クロック）と３２ＫＨｚのサブクロック（上述の低周波クロック）をクロック制御モジュール１１０の入力に接続する。３２ＫＨｚのサブクロックはＰＭＩＣ２０も必要とするため、図２２の構成では、ＰＭＩＣ２０が内蔵する３２ＫＨｚの発振回路で発振したクロック信号を、ＳｏＣ２０のクロック制御モジュール１１０にサブクロックとして接続している。なお、図２２には明示していないが、ＰＭＩＣ２０が内蔵する発振回路には３２ＫＨｚの水晶振動子を外付けする。メインクロックはシリコンオシレータ５１５で生成した２４ＭＨｚのクロック信号を、ＳｏＣ５１７のクロック制御モジュール１１０にメインクロックとして接続する。ＳｏＣ５１７のクロック制御モジュール１１０の構成は、上述の第１実施形態と同様であり、内部に備える複数のＰＬＬや逓倍器や分周器によって、メインクロックから複数の周波数のクロック信号を生成して、ＳｏＣ５１７内の各部に分配する。なお、本実施形態および上述の各実施形態におけるメインクロックとサブクロックの周波数は一例であり、これに限定されるものではない。

【０１１５】

ＳｏＣ５１７の割込みコントローラ１２０は、ＳｏＣ５１７内の各デバイス（Ｉ２Ｃコントローラ５７０、ＧＰＩＯコントローラ５８０、ＥＰＤコントローラ５４０、ＮＡＮＤコントローラ５５０、ＳＤＩＯコントローラ５６０、不図示のタイマーなど）からの割込み要求を受信すると、その割込みの種類を識別する割込みベクタを、割込みコントローラ１２０内のレジスタに登録し、プロセッサコア１０１に割込み信号を送る（割込みを通知する）。プロセッサコア１０１は、割込み信号を受信すると、割込みコントローラ１２０内のレジスタに登録された割込みベクタを参照することで、どのデバイスから割込みがかかっているのかを判断することができる。また、レジスタの設定によって、デバイスごとに割込みを許可するかしないかを設定することもできる。また、割込みコントローラ１２０は、ＳｏＣ５１７がＤＥＥＰ　ＳＬＥＥＰモードのときにデバイスからの割込み要求を受信すると、ＳｏＣ５１７を起床させてＤＥＥＰ　ＳＬＥＥＰモードから抜けさせる機能も持つ。

【０１１６】

ＳｏＣ５１７のプロセッサコア１０１は、アプリケーションプログラムやオペレーティングシステム（ＯＳ）を実行する。図２２は、ＳｏＣ５１７は１つのプロセッサコア１０１を内蔵するシングルコアの構成例を示しているが、これに限らず、例えば２個のプロセッサコアを内蔵するデュアルコア構成や、４個のプロセッサコアを内蔵するクワッドコア構成などの、マルチコア構成にすることもできる。

【０１１７】

ＳｏＣ５１７の内蔵するＲＡＭ５９０は、高速にアクセスが可能なＳＲＡＭである。スクラッチパッドメモリとして一時的な作業領域として使用することもできるし、メインメモリ（この例ではＤＲＡＭ５１６）が使えない時でも常にアクセスできるのでオペレーティングシステムの起動やスタンバイ状態からの起床時の処理に利用することができる。

【０１１８】

本実施形態の情報処理装置１は、メインメモリとしてＤＲＡＭ５１６を用いている。ＤＲＡＭ５１６は、ＳｏＣ５１７内のＤＲＡＭコントローラ５３０に接続される。ＤＲＡＭ５１６は、ＳｏＣ５１７内のプロセッサコア１０１や各種デバイスからアクセスすること

ができる。なお、ここでは、メインメモリとして揮発性メモリを用いる場合を例示しているが、これに限らず、例えばメインメモリとして、ＭＲＡＭやＰＣＭなどの不揮発性メモリを用いることもできる。図２３は、メインメモリとしてＭＲＡＭを用いた場合の情報処理装置１の構成例を示す。図２２の構成との違いは、メインメモリとしてＤＲＡＭ５１６の代わりにＭＲＡＭ６００を用いることと、メインメモリが接続されるＳｏＣ５１７内のメモリコントローラとして、ＤＲＡＭコントローラ５３０の代わりにＭＲＡＭコントローラ６０１を用いることである。メインメモリとして不揮発メモリを用いることで、後述するＨＩＢＥＲＮＡＴＩＯＮモードでメインメモリへの電力供給を止める際に、メインメモリのデータをＮＡＮＤフラッシュメモリ等の不揮発性の２次記憶装置に退避する必要がなくなる。

【０１１９】

図２２に戻って説明を続ける。ＳｏＣ５１７内のＩ２Ｃコントローラ５７０は、Ｉ２Ｃインタフェースを持つさまざまなデバイスを接続できる汎用の通信デバイスである。ＰＭＩＣ２０もＩ２Ｃインタフェースによって ＳｏＣ５１７と接続されており、ＳｏＣ５１７内のプロセッサコア１０１がＰＭＩＣ２０の制御レジスタを読み書きすることができる。ＳｏＣ５１７とＰＭＩＣ２０との間の接続には、Ｉ２Ｃインタフェース以外に、ＳＰＩインタフェースなどを用いることもできる。また、ＳｏＣ５１７とＰＭＩＣ２０との間は、Ｉ２Ｃに加えて、ＳＴＢＹ信号が供給される信号線を介して接続されている。ＳＴＢＹ信号は、ＳｏＣ５１７（あるいはプロセッサコア１０１）がＤＥＥＰ　ＳＬＥＥＰモードであるか否かを示す信号である。例えばＰＭＩＣ２０は、ＳＴＢＹ信号がアクティブレベル（例えばハイレベル）になると、各ＤＣ／ＤＣコンバータやＬＤＯレギュレータの出力をＤＥＥＰ　ＳＬＥＥＰモード用の設定に変更し、ＤＥＥＰ　ＳＬＥＥＰモード時の消費電力を削減する。

【０１２０】

ＧＰＩＯコントローラ５８０は、汎用のデジタルＩ／Ｏポートであり、本実施形態の情報処理装置１では、ＳｏＣ５１７とマイコン３００との間の通信や、電子ペーパー５２０に電力を供給するＤＣ／ＤＣコンバータ５１４の制御に用いられる。電子ペーパー５２０は、表示内容の書き換え時にしか電力を消費しないので、ＳｏＣ５１７内のプロセッサコア１０１からＧＰＩＯコントローラ５８０を介してＤＣ／ＤＣコンバータ５１４のＥＮＡＢＬＥ信号をＯＮ／ＯＦＦすることで、書き換え時にのみ電力を供給するように制御する。本実施形態のように、太陽電池で発電した電力で動作する情報処理装置においては、消費電力がより少ないＩ／Ｏデバイスを用いることが好ましいため、表示内容の書き換え時にしか電力を消費しない電子ペーパーは、太陽電池で発電した電力で動作する情報処理装置の表示デバイスとして特に好適である。

【０１２１】

また、ＳｏＣ５１７には、ＧＰＩＯコントローラ５８０を介してマイコン３００が接続されている。マイコン３００には、タッチパネル５０２とキーボード５０３が接続されていて、タッチパネル５０２のタッチされた位置の検出や、キーボード５０３が押された時のチャタリング除去や押されたキーの判定などをマイコン３００側で処理し、マイコン３００は、タッチ位置や押されたキーのコードを、ＧＰＩＯコントローラ５８０を介してＳｏＣ５１７のプロセッサコア１０１に伝える。マイコン３００は蓄電制御装置５１１にも接続されており、蓄電制御装置５１１の管理するキャパシタ（蓄電装置）５１２の蓄積電力量を検出することができるようになっている。より具体的には、キャパシタ５１２として電気二重層キャパシタを用いる場合はその出力電圧と蓄積電力量には相関関係があるので、電気二重層キャパシタの出力電圧をマイコン３００内蔵のＡ／Ｄコンバータで読み出すように実施することができる。マイコン３００は、さらに、ＰＭＩＣ２０と接続されており、マイコン３００からＰＭＩＣ２０のＯＮ／ＯＦＦを指示することで、情報処理装置１全体の電力供給のＯＮ／ＯＦＦを制御することができる。マイコン３００は、後述するように、ユーザからの入力とキャパシタ（蓄電装置）５１２の蓄積電力量を入力として、ＰＭＩＣ２０のＯＮ／ＯＦＦを制御するシステムコントローラとして機能する。

【０１２２】
ＥＰＤコントローラ５４０は、接続されている電子ペーパー５２０に対する描画処理を行う。ＥＰＤコントローラ５４０は、ＤＲＡＭ（メインメモリ）５１６やＲＡＭ５９０などに記憶しているイメージデータを読み出して、必要に応じて電子ペーパー５２０の書き換えのためにデータ変換を行い、電子ペーパー５２０を書き換える信号を出力する。
【０１２３】
ＮＡＮＤコントローラ５５０は、接続されているＮＡＮＤフラッシュメモリ５０８のチップに対するデータの読み書きを行う。本実施形態の情報処理装置１はＮＡＮＤフラッシュメモリ５０８を２次ストレージとして使用し、その記憶領域は、オペレーティングシステムがファイルシステムによって管理する。
【０１２４】
また、本実施形態の情報処理装置１は、通信手段として無線ＬＡＮモジュール５０９を備える。無線ＬＡＮモジュール５０９は、ＳＤＩＯコントローラ５６０を介してＳｏＣ５１７に接続されている。なお、これに限らず、無線ＬＡＮモジュール５０９とＳｏＣ５１７は、ＳＤＩＯインタフェースではなく、例えばＵＳＢ、ＰＣＩ－ｅｘｐｒｅｓｓ、ＳＰＩ、ＵＡＲＴなどのインタフェースで接続するように実施しても構わない。また、無線ＬＡＮ回路をＳｏＣ５１７に内蔵しても構わない。
【０１２５】
本実施形態の情報処理装置１のＳｏＣ５１７としては、例えばフリースケール・セミコンダクタ社のｉ．ＭＸ５０８を用いることができるが、これに限られるものではなく、様々な種類のＳｏＣを用いることができる。また、本実施形態の情報処理装置１のＰＭＩＣ２０には、例えばフリースケール・セミコンダクタ社のＭＣ３４７０８を用いることができるが、これに限られるものではなく、様々な種類のＰＭＩＣ２０を用いることができる。また、例えばＰＭＩＣ２０を用いずに、複数のＤＣ／ＤＣコンバータやＬＤＯレギュレータを組み合わせて用いるようにしても構わない。
【０１２６】
図２４は、本実施形態の情報処理装置１のＳｏＣ５１７の状態遷移の一例を示す図である。ＳｏＣ５１７はプロセッサコア１０１を中心とするコンピュータシステムであるので、ＳｏＣ５１７が内蔵するプロセッサコア１０１の状態遷移も、図２４と同様であると考えることができる。この例では、プロセッサコア１０１が命令を実行中（プロセッサコア１０１がラン状態）のときのＳｏＣ５１７の状態をＡＣＴＩＶＥモードと呼ぶ。プロセッサコア１０１が命令を実行せず割り込み待ち（プロセッサコア１０１がアイドル状態）のときのＳｏＣ５１７の状態はＷＡＩＴモードまたはＤＥＥＰ　ＳＬＥＥＰモード（ｉ．ＭＸ５０８等のＳｏＣではＳＴＯＰモードとも呼ばれる）のいずれかである。上述の各実施形態と同様に、何れかのＩ／Ｏデバイス（またはコントローラ）がＩ／Ｏ処理を進めている場合はＷＡＩＴモードに、何れのＩ／Ｏデバイス（またはコントローラ）もＩ／Ｏ処理をしていない場合はＤＥＥＰ　ＳＬＥＥＰモードになる。ＷＡＩＴモードもＤＥＥＰ　ＳＬＥＥＰモードも、プロセッサコア１０１の命令実行を停止する待機状態（アイドル状態）であるが、前述したように、ＤＥＥＰ　ＳＬＥＥＰモードでは、ＳｏＣ内のメインクロックが停止される等によりＷＡＩＴモードよりも消費電力が小さくなる。ユーザが情報処理装置１を使用している状態、すなわち稼働状態では、情報処理装置１のＳｏＣ（もしくはプロセッサコア１０１）は、ＡＣＴＩＶＥモードとＷＡＩＴモードとＤＥＥＰ　ＳＬＥＥＰモードとの間を遷移している。
【０１２７】
また、ユーザが情報処理装置１の電源をＯＦＦした場合や、キャパシタ（蓄電装置）５１２の蓄電量が小さく太陽電池５０１の発電量も少ないために稼働状態を続けられない場合には、情報処理装置１は、ＳｏＣ５１７内部の状態をＤＲＡＭ（メインメモリ）５１６に退避してから、さらにＤＲＡＭ５１６のデータを２次ストレージであるＮＡＮＤフラッシュメモリ５０８に退避した後、情報処理装置１への電力供給を停止させる（電源を切る）。この状態を、ＨＩＢＥＲＮＡＴＩＯＮモードと呼ぶ。ＨＩＢＥＲＮＡＴＩＯＮモード

は、ユーザから見て情報処理装置１が停止しているように見える停止状態である。なお、メインメモリとしてＭＲＡＭなどの不揮発性メモリを用いる場合は、ＳｏＣ５１７の内部状態をメインメモリに退避して電源を切ればよく、メインメモリのデータをＮＡＮＤフラッシュメモリ５０８などの２次ストレージに退避する必要はない。

【０１２８】

図２５は、本実施形態の情報処理装置１に搭載されたオペレーティングシステム（ＯＳ）の機能構成の一例を示すブロック図である。オペレーティングシステムとしては、例えばＬｉｎｕｘを用いることができるが、これに限定されるものではなく、さまざまなオペレーティングシステムを用いることができる。オペレーティングシステムは、２次ストレージであるＮＡＮＤフラッシュメモリ５０８のファイルシステム上に記憶され、情報処理装置１の稼働状態にプロセッサコア１０１によってメインメモリ（ＤＲＡＭ５１６）に読み出されて実行される。情報処理装置１に搭載されたオペレーティングシステムは、主要な機能として、プロセス管理部４１０、デバイス管理部４２００、ファイル管理部４３０、メモリ管理部４４０を有する。

【０１２９】

デバイス管理部４２００は、各Ｉ／Ｏデバイスのデバイスドライバ（ＥＰＤデバイスドライバ、ＮＡＮＤデバイスドライバ、Ｉ２Ｃデバイスドライバ、ＳＤＩＯ／ＷＬＡＮデバイスドライバ、不図示のキーボードドライバおよびタッチパネルドライバなど）を有することに加え、本実施形態に特有のものとして、キャパシタ（蓄電装置）５１２に蓄えられた電力量を示す蓄電電力量情報と、動作中デバイスカウンタ（上述のデバイス状態管理情報の一例）とを管理する。本実施形態では、システムコントローラであるマイコン３００は、適切なタイミングでキャパシタ（蓄電装置）５１２の蓄積電力量を検出し、その値をＳｏＣ５１７のプロセッサコア１０１に通知する。プロセッサコア１０１は、通知された値を蓄積電力量情報として記録する。動作中デバイスカウンタは、現在Ｉ／Ｏ処理を実行中のＩ／Ｏデバイス（あるいはコントローラ）の数を記録する領域であり、デバイス状態管理部４２１０によって管理される。

【０１３０】

第１実施形態と同様に、プロセス切替部４１１のデバイス状態判定部４１２は、プロセッサコア１０１がアイドル状態に入るときに、動作中デバイスカウンタの値を参照し、その値が０であればＩ／Ｏ処理を実行中のＩ／Ｏデバイス（あるいはコントローラ）は無いのでＤＥＥＰ　ＳＬＥＥＰモードにして良いと判断する一方、動作中デバイスカウンタの値が０でなければＤＥＥＰ　ＳＬＥＥＰモードにすべきではないと判断する。

【０１３１】

以降、第８実施形態の情報処理装置１およびオペレーティングシステム（ＯＳ）の動作を詳しく説明する。

【０１３２】

図２６は、デバイス管理部４２００が管理するデバイスドライバの典型的な動作フローを示している。図２６に示すように、デバイスドライバは、ＯＳやアプリケーションプログラムからの入出力要求を受け取ると（ステップＳ５１）、デバイス状態管理部４２１０内の動作中デバイスカウンタの値をインクリメントする（ステップＳ５２）。例えばＮＡＮＤデバイスドライバが受け取る入出力要求であればデータの読み出しや書き込みの要求であり、例えばＥＰＤデバイスドライバが受け取る入出力要求であれば画面への表示の要求であり、ＳＤＩＯ／ＷＬＡＮデバイスドライバが受け取る入出力要求であればネットワークへのデータ送信や受信の要求などである。

【０１３３】

次に、デバイスドライバは、要求された入出力処理を実行させるために、該当するデバイスの制御レジスタに所定の値を書き込むなどの方法により、ハードウェア機能を起動する（ステップＳ５３）。次に、デバイスドライバは、入出力処理中にデバイスドライバが関与する必要がある処理（例えばデータ転送に用いる複数のバッファ領域の切り替え指示や、入出力処理途中の例外処理など）があればそれを行い（ステップＳ５４）、起動した

入出力処理が完了するのを待つ（ステップＳ５５）。入出力処理が完了すれば、デバイスドライバは、該当するデバイスの制御レジスタに所定の値を書き込むなどの方法により、デバイスのハードウェア機能を停止させ（ステップＳ５６）、動作中デバイスカウンタの値をデクリメントし（ステップＳ５７）、入出力処理結果（入出力処理の成否や入力したデータなど）を入出力要求元に返す（ステップＳ５８）。なお、オペレーティングシステムはマルチプログラミングで動作しているので、デバイスのハードウェアが入出力処理を行っている間（ステップＳ５３からＳ５５の間）、プロセッサコア１０１は、他の実行可能なプロセス（あるいはスレッドまたはタスク）があればそれを実行する。

【０１３４】

図２６のデバイスドライバのフローでは、入出力要求を実行する度に、ハードウェアを起動して入出力処理を行って、ハードウェアを停止する。一方、無線ＬＡＮなどハードウェアの起動に時間がかかるデバイスの場合は、入出力処理が終わってもすぐには停止せず、一定時間入出力要求が来なければ停止するように制御することで、ハードウェアの起動／停止の頻度を下げて消費電力も削減することができる。このような処理を行うように図２６の動作フローを拡張することもできる。

【０１３５】

次に、図２７を用いて、本実施形態のプロセス切替部４１１の動作例を説明する。図２７は、プロセス切替部４１１の動作例を示すフローチャートである。プロセスの切り替えは、例えばプロセッサコア１０１が実行中のプロセスが、入出力要求を出したり、プロセス間通信やメモリ管理等のＯＳの機能を呼び出したり、Ｉ／Ｏデバイス（コントローラ）やタイマーからの割り込みがかかったりして、ＯＳのカーネルに入ったときなどに実行される。

【０１３６】

プロセスを切り替えるとき、まずプロセス切替部４１１は、現在実行中であるプロセスを停止させてプロセスキューに入れる（ステップＳ６１）。プロセスキューには、実行可能なプロセスが入る実行可能プロセスキューと、入出力完了などのイベントを待っているプロセスが入るイベント待ちプロセスキューがある。実行可能プロセスキューは、優先度に基づいてプロセスを管理している。ステップＳ６１では、プロセス切替部４１１は、現在実行中であるプロセスがイベント待ちになる場合は、当該プロセスを停止させてイベント待ちキューに入れる一方、現在実行中であるプロセスがイベント待ちになるのではなく他の優先度の高いプロセスに切り替える場合は、現在実行中のプロセスを停止させて実行可能プロセスキューに入れる。

【０１３７】

次に、プロセス切替部４１１は、実行可能プロセスキューにプロセスが存在するかどうかを調べ（ステップＳ６２）、実行可能プロセスキューにプロセスが存在すれば（ステップＳ６２：ＹＥＳ）、プロセス切替部４１１は、実行可能プロセスキューに存在するプロセスのうち最も優先度が高いプロセスを選択し、プロセッサコア１０１にそのプロセスの実行を再開させる（ステップＳ６３）。実行可能プロセスキューにプロセスが存在しなければ（ステップＳ６２：ＮＯ）、プロセス切替部４１１のデバイス状態判定部４１２が、デバイス状態管理部４２１０が管理する動作中デバイスカウンタの値を参照し、動作中デバイスカウンタの値が０であるかどうかを判断する（ステップＳ６４）。つまり、プロセッサコア１０１がアイドル状態になったときにＳｏＣ５１７をＤＥＥＰ　ＳＬＥＥＰモードにして良いかどうかを判断する。

【０１３８】

上述のステップＳ６４において、動作中デバイスカウンタの値が０でなければ（ステップＳ６４：ＮＯ）、Ｉ／Ｏ処理を実行中のＩ／Ｏデバイス（コントローラ）が存在するので、ＤＥＥＰ　ＳＬＥＥＰモードにすべきではないと判断することができる。この場合、プロセス切替部４１１は、ＷＦＩ命令が発行されたらＳｏＣ５１７はＷＡＩＴモードに遷移するように、ＳｏＣ５１７の制御レジスタを設定する（ステップＳ６５）。そして、プロセス切替部４１１は、ＷＦＩ命令を発行することでプロセッサコア１０１をアイドル状

態に遷移させ、Ｉ／Ｏデバイス（コントローラ）等からの割込みを待つ（ステップＳ６６）。Ｉ／Ｏデバイス（コントローラ）等からの割込みが発生するとＷＡＩＴモードから抜けるので、その後は、上述のステップＳ６２に制御を移す。

【０１３９】

一方、上述のステップＳ６４において、動作中デバイスカウンタの値が０であれば（ステップＳ６４：ＹＥＳ）、Ｉ／Ｏ処理を実行中のＩ／Ｏデバイス（コントローラ）は存在しないので、ＤＥＥＰ　ＳＬＥＥＰモードにしても良いと判断することができる。この場合、プロセス切替部４１１は、プロセッサコア１０１を割込み禁止にし（ステップＳ６７）、ＷＦＩ命令が発行されたらＳｏＣ５１７はＤＥＥＰ　ＳＬＥＥＰモードに遷移するように、ＳｏＣ５１７の制御レジスタを設定する（ステップＳ６８）。次に、プロセス切替部４１１は、ＤＥＥＰ　ＳＬＥＥＰモード中は上述のＰＬＬ１１１を停止するようにＳｏＣ５１７の制御レジスタ（例えばクロック制御モジュール１１０の制御レジスタ１１５）を設定する（ステップＳ６９）。次に、プロセス切替部４１１は、プロセッサコア１０１のキャッシュメモリ１４０の中のダーティなデータをメインメモリであるＤＲＡＭ５１６に書き出してキャッシュメモリ１４０を停止させる（ステップＳ７０）。次に、プロセス切替部４１１は、ＤＲＡＭ５１６にコマンドを送ってセルフリフレッシュモードにする（ステップＳ７１）。次に、プロセス切替部４１１は、ＷＦＩ命令を発行することでプロセッサコア１０１を割込み待ちのアイドル状態に遷移させ（つまり、ＳｏＣ５１７をＤＥＥＰ　ＳＬＥＥＰモードに遷移させ）、Ｉ／Ｏデバイス（コントローラ）等からの割り込みを待つ（ステップＳ７２）。

【０１４０】

その後、Ｉ／Ｏデバイス（コントローラ）等からの起床可能な割込みが発生するとＤＥＥＰ　ＳＬＥＥＰモードから抜けるので、プロセス切替部４１１は、ＤＲＡＭ５１６にコマンドを送ってセルフリフレッシュモードを解除して通常の動作モードにし（ステップＳ７３）、次に、プロセッサコア１０１のキャッシュメモリ１４０を初期化して使用可能な状態にする（ステップＳ７４）。その後、プロセス切替部４１１は、プロセッサコア１０１を割り込み可能にし（ステップＳ７５）、ステップＳ６２に制御を移す。ステップＳ６７からステップＳ７５までの間を割込み禁止にする理由は、割込み処理ルーチンがＤＲＡＭ５１６上にあるので、起床してからＤＲＡＭ５１６がアクセスできるように設定した後に割込みがかかるようにするためである。

【０１４１】

なお、ＤＥＥＰ　ＳＬＥＥＰモード中にプロセッサコア１０１のキャッシュメモリ１４０のデータが保持される場合は、上述のステップＳ７０とステップＳ７４は不要となる。たとえばｉ．ＭＸ５０８では、ＳＴＯＰモード（ＤＥＥＰ　ＳＬＥＥＰモード）に入るときにキャッシュメモリをパワーゲーティングするかどうかを制御レジスタの設定で選べるようになっているので、その設定に合わせて上述のステップＳ７０とステップＳ７４を省略できる。

【０１４２】

図２８は、システムコントローラとして動作するマイコン３００とＳｏＣ５１７のＧＰＩＯコントローラ５８０との接続例を示す図である。図２８の例では、マイコン３００とＳｏＣ５１７との間は、大きく分けて３種類の信号線で接続される。第１の信号線は、図２８ではＣＴＬと示されているハンドシェイク用の２本の信号線である。この２本の信号線は、マイコン３００からＳｏＣ５１７にデータを送るときにＳｏＣ５１７に割込みをかけるための信号線（割込み信号を送るための信号線）と、そのＡｃｋ（応答信号）をＳｏＣ５１７からマイコン３００に返すための信号線とから構成されるが、この方式に限られるものではない。第２の信号線は、図２８ではＣＭＤと示されているデータの種類を識別するための３本の信号線である。この３本の信号線は、マイコン３００からＳｏＣ５１７にデータを送る際に、送るデータ本体が、キーボード５０３のキーコードか、タッチパネル５０２のタッチ開始か、現在のタッチ位置か、タッチ終了か、キャパシタ（蓄電装置）５１２の電力残量か、ＨＩＢＥＲＮＡＴＩＯＮモードに入る指示か、などを識別するため

のコードを伝えるための信号線である。第３の信号線は、図２８ではＤＡＴＡと示されているデータ本体を送るための複数本の信号線である。

【０１４３】

図２９は、システムコントローラとしてのマイコン３００による電源管理を説明するための図である。図２９に示すグラフの縦軸は、キャパシタ（蓄電装置）５１２の蓄積電力量、横軸は時間の推移（経過）を示している。キャパシタの蓄積電力量が０の状態から始まり、太陽電池５０１で発電した電力が蓄積されて時刻Ｔ１に第３の電力量に達すると、まず情報処理装置１内のマイコン３００が動作を開始する。この時点でＰＭＩＣ２０は電力供給を停止しており、ＳｏＣ５１７はＨＩＢＥＲＮＡＴＩＯＮモードで停止している。時刻Ｔ１よりも後の、キャパシタ５１２の蓄積電力量が第１の電力量になった時刻Ｔ２以降に、マイコン３００がキーボード５０３やタッチパネル５０２からのユーザ入力を検出すると、マイコン３００はＰＭＩＣ２０に電力供給開始を指示し、ＳｏＣ５１７の動作を再開させる（つまり、ＨＩＢＥＲＮＡＴＩＯＮモードを抜ける）。その後、太陽電池５０１が発電する電力で情報処理装置１の消費電力が賄えなくなってくると、キャパシタ５１２の蓄積電力量が低下し、時刻Ｔ２よりも後の時刻Ｔ３で蓄積電力量が第２の電力量になったことをマイコン３００が検出すると、マイコン３００は、ＳｏＣ５１７にＨＩＢＥＲＮＡＴＩＯＮモードに入ることを指示する。さらに、キャパシタ５１２の蓄積電力量が０になっても、再び太陽電池５０１で発電した電力がキャパシタ５１２に蓄積されて時刻Ｔ４で再び第３の電力量に達すると、マイコン３００が動作を再開する。そして、蓄積電力量が第１の電力量になった時刻Ｔ５以降に、マイコン３００がユーザ入力を検出すると、マイコン３００は再びＰＭＩＣ２０に電力供給の開始を指示してＳｏＣ５１７の動作を再開させる。動作を再開したＳｏＣ５１７は、時刻Ｔ３の時の状態から処理を継続する。

【０１４４】

ここで、第１の電力量は、ＳｏＣ５１７がＨＩＢＥＲＮＡＴＩＯＮモードから抜け、何らかの処理を行い、再びＨＩＢＥＲＮＡＴＩＯＮモードに入るまでの処理を行うのに必要な電力量である。第２の電力量は、ＳｏＣ５１７がＨＩＢＥＲＮＡＴＩＯＮモードに入るのに必要な電力量である。そのため、第２の電力量は第１の電力量よりも小さい。

【０１４５】

図３０は、図２９に示した電源管理とキーボード５０３およびタッチパネル５０２の制御を行うマイコン３００の動作例を示すフロー図である。マイコン３００のプログラムはマイコン３００が有する各種のＩ／Ｏデバイス（例えばキーボード５０３やタッチパネル５０２などの外付けのデバイスの他、コンパレータなどのマイコン３００に内蔵されたデバイスなど）やタイマーからの割込みイベントによって、イベントドリブンで動作するように作られ、図３０は、各種のＩ／Ｏデバイスやタイマーからの割込みが発生した時にマイコン３００が実行する割り込み処理ルーチンの処理フローになっている。

【０１４６】

マイコン３００に内蔵された割込みコントローラ３０６は、各種Ｉ／Ｏデバイスやタイマー等から割込み要求を受信すると、その要求された割込みの種類を識別する割込みベクタを、割込みコントローラ３０６が有するレジスタ（不図示）に登録し、マイコン３００に内蔵されたプロセッサコア３０１に割込み信号を送る（割込みを通知する）。プロセッサコア３０１は、割込み信号を受信すると、割込みコントローラ３０６が有するレジスタに登録された割込みベクタを参照して、割込み要因を特定することができる。割込みコントローラ３０６からの割込み信号を受信すると、プロセッサコア３０１は、まず割込み要因を調べて、キーボード５０３からの割込みかどうかを判断する（ステップＳ８１）。キーボード５０３からの割込みであれば（ステップＳ８１：ＹＥＳ）、キーボード５０３からデータを入力する（ステップＳ８２）。このとき、必要に応じてキーボードのチャタリング除去などの処理も併せて行う。

【０１４７】

キーボード５０３からの割込みでなければ（ステップＳ８１：ＮＯ）、タッチパネル５０２からの割込みかどうかを調べ（ステップＳ８３）、タッチパネル５０２からの割込み

であれば（ステップＳ８３：ＹＥＳ）、タッチパネル５０２からデータを入力する（ステップＳ８４）。なお、タッチパネル５０２からの割込みは、タッチ開始、タッチ中のタッチ位置の変化、タッチ終了などによって発生する。ステップＳ８２あるいはステップＳ８４でキーボード５０３あるいはタッチパネル５０２からデータを入力した後は、まず、ＳｏＣ５１７が稼働状態（ＨＩＢＥＲＮＡＴＩＯＮモードでない状態）かどうかを調べ（ステップＳ８５）、ＳｏＣ５１７が稼動状態でなければ（ステップＳ８５：ＮＯ）、ＳｏＣ５１７を稼働状態で動作させるのに必要な電力が蓄電装置にあるか（キャパシタ５１２の蓄積電力量が上記第１の電力量よりも大きいかどうか）を調べる（ステップＳ８６）。ＳｏＣ５１７を稼動状態で動作させるのに必要な電力が蓄電装置にあれば（ステップＳ８６：ＹＥＳ）、ＰＭＩＣ２０に電力供給の開始を指示してＳｏＣ５１７の動作を再開させて（ステップＳ８７）、入力したデータをＳｏＣ５１７に送る（ステップＳ８８）。そして、ＳｏＣ５１７に送ったデータに対応する割込みの種類を識別する割込みベクタを、割込みコントローラ３０６のレジスタから削除（クリア）して（ステップＳ８９）、割込み処理を終了する。

【０１４８】

上述のステップＳ８５において、ＳｏＣ５１７が稼働状態であると判断した場合（ステップＳ８５：ＹＥＳ）は、ステップＳ８８に制御を移す。また、上述のステップＳ８６において、ＳｏＣ５１７を稼働状態にするだけの電力が無い場合（ステップＳ８６：ＮＯ）は、入力データを捨てて（ステップＳ８９）、ステップＳ８６に制御を移して割込みをクリアして（ステップＳ９０）、割込み処理を終了する。

【０１４９】

上述のステップＳ８７でＰＭＩＣ２０に電力供給の開始を指示してＳｏＣ５１７の動作を再開させるとき、マイコン３００はＰＭＩＣ２０に指示を出した後、ＳｏＣ５１７が動作を再開したことを知る必要がある。その方法の一例としては、ＳｏＣ５１７がマイコン３００に対して、あらかじめ決めておいたＧＰＩＯの信号線を使って動作再開を伝えるように実施できる。また別の方法としては、ＰＭＩＣ２０に指示を出してからＳｏＣ５１７が動作再開するまでに要する時間をあらかじめ計測しておき、マイコン３００は、ＰＭＩＣ２０に指示を出してから少なくとも上記時間が経過した場合は、ＳｏＣ５１７が動作を再開したと解釈して処理を続けるように実施することもできる。さらに別の方法として、ＳｏＣ５１７とマイコン３００との間をＩ２ＣやＳＰＩなどの通信線で接続しておき、その通信線を使ってＳｏＣ５１７の動作再開をマイコン３００に通知するように実施しても良い。また、ＳｏＣ５１７の状態を示す信号線をマイコン３００に接続し、マイコン３００は、その信号線を介してＳｏＣ５１７の動作再開を知るように実施しても良い。その他、マイコン３００は、ＳｏＣ５１７が動作を再開することによって状態が変化する各種信号線を利用して判断するように実施しても良い。

【０１５０】

上述のステップＳ８３において、タッチパネル５０２からの割込みではなかった場合（ステップＳ８３：ＮＯ）、マイコン３００のプロセッサコア３０１は、タイマーからの割込みかどうかを調べ（ステップＳ９１）、タイマーからの割込みであれば（ステップＳ９１：ＹＥＳ）、ＳｏＣ５１７が稼働状態かどうかを調べる（ステップＳ９２）。ＳｏＣ５１７が稼動状態であれば（ステップＳ９２：ＹＥＳ）、マイコン３００のプロセッサコア３０１は、キャパシタ５１２の蓄積電力量を入力し（ステップＳ９３）、その入力した蓄積電力量をＳｏＣ５１７へ送り（ステップＳ９４）、上述のステップＳ９０に制御を移して割込みをクリアして、割込み処理を終了する。

【０１５１】

上述のステップＳ９２において、ＳｏＣ５１７が稼働状態でない場合（ステップＳ９２：ＮＯ）、マイコン３００のプロセッサコア３０１は、何もせずに上述のステップＳ８９に制御を移して割込みをクリアして、割込み処理を終了する。なお、タイマーからの割込みは、マイコンからＳｏＣ５１７に対してキャパシタ５１２の蓄積電力量を通知するためのものであり、一定の時間間隔で割り込みがかかるように設定しておく。

【０１５２】
また、上述のステップＳ９１において、タイマーからの割込みでなかった場合（ステップＳ９１：ＮＯ）、マイコン３００のプロセッサコア３０１は、コンパレータからの割込みかどうかを調べる（ステップＳ９５）。コンパレータは、マイコン３００に内蔵されたデバイスの一つで、アナログ信号があらかじめ設定した電圧を超えたり下がったりした場合に割込みを発生することができる。この例では、コンパレータを使って、キャパシタ５１２の蓄積電力量が低下して第２の電力量を下回ったことを検出するようにしておく。上述のステップＳ９５において、コンパレータからの割込みであれば（ステップＳ９５：ＹＥＳ）、キャパシタ５１２の蓄積電力量が低下しているので、マイコン３００のプロセッサコア３０１は、ＳｏＣ５１７に対してＨＩＢＥＲＮＡＴＩＯＮモードに入るように指示し（ステップＳ９６）、上述のステップＳ９０に制御を移して割込みをクリアして、割込み処理を終了する。一方、上述のステップＳ９５でコンパレータからの割込みでなかった場合（ステップＳ９５：ＮＯ）、マイコン３００のプロセッサコア３０１は、何もせずに上述のステップＳ９０に制御を移して割込みをクリアして、割り込み処理を終了する。
【０１５３】
なお、図３０は、マイコン３００の割込みコントローラ３０６が、Ｉ／Ｏデバイス（あるいはコントローラ）からの割込みイベントが発生するたびに（割込み要求を受信するたびに）、マイコン３００のプロセッサコア３０１に割込み信号を通知する場合の処理フローを示している。つまり、Ｉ／Ｏデバイス（コントローラ）の割込みイベント発生の数と、プロセッサコアが受ける割込みの数が同じ場合である。一方、複数のＩ／Ｏデバイス（コントローラ）で割込みイベントが発生した時にプロセッサコアにまとめて割込みをかける割込みコントローラもある。この場合、複数の割込み要因によって割込みが発生しても、図３０の処理フローでは一つの割込み要因に対する処理しかできない。そのような場合は、マイコン３００のプロセッサコア３０１は、図３０のステップＳ８９で処理した割込み要因に対応する割込みクリアを行った後、マイコン３００の割込みコントローラ３０６の制御レジスタを参照して、他に処理していない割込みが残っていないかを調べ、残っていれば上述のステップＳ８１に戻って、それを処理するように実施すればよい。
【０１５４】
図３１は、図３０の処理フローで動作するマイコン３００からの割込みに対するＳｏＣ５１７の動作例を示すフロー図である。マイコン３００からの割込みを受けたＳｏＣ５１７のプロセッサコア１０１は、上述の第２の信号線（図２８でＣＭＤと示されている信号線）を見ることで、マイコン３００から送られてきたデータの種類が、キーボード５０３からのデータか否か（ステップＳ１０１）、タッチパネル５０２からのデータか否か（ステップＳ１０３）、キャパシタ（蓄電装置）５１２の蓄積電力量の通知か否か（ステップＳ１０５）、ＨＩＢＥＲＮＡＴＩＯＮモードに入る指示か否か（ステップＳ１０７）を判断する。
【０１５５】
上述のステップＳ１０１において、マイコン３００から送られてきたデータが、キーボード５０３からのデータであれば（ステップＳ１０１：ＹＥＳ）、上述の第３の信号線（図２８でＤＡＴＡと示されている信号線）で送られてきたデータをキーボードドライバに送る（ステップＳ１０２）。上述のステップＳ１０３において、マイコン３００から送られてきたデータが、タッチパネル５０２からのデータであれば（ステップＳ１０３：ＹＥＳ）、上述の第３の信号線（図２８でＤＡＴＡと示されている信号線）で送られてきたデータをタッチパネルドライバに送る（ステップＳ１０４）。上述のステップＳ１０５において、マイコン３００から送られてきたデータが、キャパシタ（蓄電装置）５１２の蓄積電力量の通知であれば（ステップＳ１０５：ＹＥＳ）、上述の第３の信号線（図２８でＤＡＴＡと示されている信号線）で送られてきたデータを、蓄電装置の蓄積電力量情報として記録する（ステップＳ１０６）。上述のステップＳ１０７において、ＨＩＢＥＲＮＡＴＩＯＮモードに入る指示であれば（ステップＳ１０７：ＹＥＳ）、ＨＩＢＥＲＮＡＴＩＯＮモードへ移行する処理を行う（ステップＳ１０８）。

【０１５６】
システムコントローラは、本実施形態のようにマイコン３００で実現してもよいし、専用ハードウェアで実現してもよいし、専用ハードウェアとプロセッサの組み合わせで実現してもよい。また、システムコントローラは、ＳｏＣに内蔵しても良いし、ＰＭＩＣに内蔵しても良い。また、システムコントローラは、キーボードやタッチパネルの制御だけでなく、無線ＬＡＮやＢｌｕｅｔｏｏｔｈやＺｉｇｂｅｅなどの通信デバイスを接続し、それらの制御を行ってもよいし、データの受信に伴ってＳｏＣを起床させるような電源制御を行っても良い。さらに、システムコントローラの機能を複数のマイコンやプロセッサに分散して実装しても良い。例えば、キーボードとタッチパネルはマイコンがシステムコントローラとして管理し、無線ＬＡＮは無線ＬＡＮモジュール内のプロセッサがシステムコントローラとして管理し、両者を協調して動作させてもよい。
【０１５７】
（第９実施形態）
上述の各実施形態では、本発明の消費電力削減方式を、情報処理装置１のプロセッサコア１０１が実行するオペレーティングシステムによって実現する場合を例に挙げて説明したが、第９実施形態では、本発明の消費電力削減方式を、ハイパーバイザによって実現する場合を例に挙げて説明する。ハイパーバイザは、コンピュータを仮想化し、複数のＯＳを並列に実行できるようにするためのソフトウェアであり、仮想計算機（バーチャルマシン）、仮想計算機モニタ、などと呼ばれることもある。
【０１５８】
図３２は、本実施形態のハイパーバイザの機能構成の一例と、ハイパーバイザとオペレーティングシステムとの関係を説明するための図である。良く知られているように、ハイパーバイザはオペレーティングシステムの下の層のシステムソフトウェアで、プラットフォームとなるハードウェアを仮想化してオペレーティングシステムに見せる役割を持つ。オペレーティングシステムは、ハイパーバイザによって仮想化されたハードウェア上で実行される。ハイパーバイザによって仮想化されたハードウェア上で動くオペレーティングシステムを、「ゲストＯＳ」と称する場合がある。図３２ではハイパーバイザの上で実行しているオペレーティングシステム（ゲストＯＳ）はひとつであるが、これに限らず、複数のオペレーティングシステムを同時にハイパーバイザ上で動かすこともできる。
【０１５９】
図３２に示すように、ハイパーバイザは、ＣＰＵ管理部７００によるＣＰＵの仮想化、Ｉ／Ｏデバイス管理部７１０によるＩ／Ｏデバイスの仮想化、メモリ管理部７２０によるメモリの仮想化、の３つの主要なリソース管理機能を持つ。ＣＰＵ管理部７００は、ＷＦＩ処理部７０１と割込み処理部７０２とを有する。Ｉ／Ｏデバイス管理部７１０は、Ｉ／Ｏデバイス制御部７１１とデバイス状態管理部７１２とを有する。
【０１６０】
本実施形態では、現在、入出力処理（Ｉ／Ｏ処理）を実行しているＩ／Ｏデバイスが存在するかどうか判断するための情報（デバイス状態管理情報）を管理する機能（デバイス状態管理部７１２）を、Ｉ／Ｏデバイス管理部７１０に持たせる。デバイス状態管理情報の管理方法の一例として、Ｉ／Ｏデバイス管理部７１０は、例えば入出力処理を実行中のＩ／Ｏデバイスの数をカウントするカウンタを管理するように実施することもできるし、Ｉ／Ｏデバイスを識別する情報ごとに、入出力処理を実行中であるかどうかを示す情報（例えば所定の値で表してもよい）を対応付けたテーブル形式の情報を管理するように実施することができる。
【０１６１】
また、本実施形態では、デバイス状態管理部７１２が管理する情報を適切なタイミングで更新する機能を、ゲストＯＳからの指示に従って実Ｉ／Ｏデバイスを制御するＩ／Ｏデバイス制御部７１１と、Ｉ／Ｏデバイスからの割込みイベントを受けて適切なゲストＯＳにルーティングする割込み処理部７０２とに持たせる。さらに、アイドル状態に入るときにデバイス状態管理部７１２の管理する情報に基づいてＷＡＩＴモードにするかＤＥＥＰ

ＳＬＥＥＰモードにするかを決める機能をＷＦＩ処理部７０１に持たせる。この例では、ＷＦＩ処理部７０１は、上述の各実施形態のプロセス切替部４１１に対応し、請求項の「状態制御部」に対応していると考えることができる。

【０１６２】

図３３は、ＣＰＵ管理部７００のＷＦＩ処理部７０１の動作例を示すフロー図である。ハイパーバイザはゲストＯＳによるＷＦＩ命令の発行をトラップすると、図３３の処理を実行する。ゲストＯＳがＷＦＩ命令をトラップする方法は、プラットフォームのハードウェアが仮想計算機支援機構（Virtual Machine Assist）を備えていれば、そのハードウェアによってＷＦＩ命令の発行を検出できる。別の方法としては、ゲストＯＳの中のＷＦＩ命令を発行する部分を書き換えて、ハイパーバイザのＷＦＩ処理部７０１に制御を移すように実施することもできる。

【０１６３】

ＷＦＩ処理部７０１は、ゲストＯＳのＷＦＩ命令の発行を検出すると、図３３のフローに従って処理を行う。まず、現在実行中であるゲストＯＳを割込み待ち状態にする（ステップＳ１１１）。続いて、他に実行可能なゲストＯＳが存在するかどうかを調べ（ステップＳ１１２）、存在すれば（ステップＳ１１２：ＹＥＳ）、優先度が最も高いゲストＯＳにスイッチしてその処理を再開する（ステップＳ１１３）。一方、他に実行可能なゲストＯＳが存在しない場合は（ステップＳ１１２：ＮＯ）、デバイス状態管理部７１２が管理する情報（デバイス状態管理情報）を参照して、現在入出力処理を実行中のＩ／Ｏデバイス（コントローラ）が存在するかどうかを判断する（ステップＳ１１４）。入出力処理を実行中のＩ／Ｏデバイスが存在する場合（ステップＳ１１４：ＹＥＳ）、ＷＦＩ処理部７０１は、プロセッサコア１０１がアイドル状態に遷移するときにＤＥＥＰ　ＳＬＥＥＰモードにすべきではないと判断し、ＷＦＩ命令が発行されたらＷＡＩＴモードに入るようにＳｏＣの制御レジスタを設定する（ステップＳ１１５）。その後、ＷＦＩ命令を発行することでプロセッサコア１０１をアイドル状態に遷移させ、Ｉ／Ｏデバイス等からの割込みを待つ（ステップＳ１１６）。Ｉ／Ｏデバイス（コントローラ）等からの割込みが発生するとＷＡＩＴモードから抜けるので、その後は、上述のステップＳ１１２に制御を移す。

【０１６４】

上述のステップＳ１１４において、入出力処理を実行中のＩ／Ｏデバイスが存在しない場合（ステップＳ１１４：ＮＯ）、ＷＦＩ処理部７０１は、プロセッサコア１０１がアイドル状態に遷移するときにＤＥＥＰ　ＳＬＥＥＰモードにしてよいと判断し、まずプロセッサコア１０１を割込み禁止にし（ステップＳ１１７）、ＷＦＩ命令が発行されたらＤＥＥＰ　ＳＬＥＥＰモードに入るようにＳｏＣの制御レジスタを設定する（ステップＳ１１８）。さらに、ＤＥＥＰ　ＳＬＥＥＰモード中は上述のＰＬＬ１１１を停止するようにＳｏＣの制御レジスタを設定する（ステップＳ１１９）。次に、ＷＦＩ処理部７０１は、プロセッサコア１０１のキャッシュメモリ１４０の中のダーティなデータをメインメモリ（この例ではＤＲＡＭであるとする）に書き出してキャッシュメモリ１４０を停止させる（ステップＳ１２０）。次に、ＷＦＩ処理部７０１は、ＤＲＡＭにコマンドを送ってセルフリフレッシュモードにする（ステップＳ１２１）。次に、ＷＦＩ処理部７０１は、ＷＦＩ命令を発行することでプロセッサコア１０１を割込み待ちのアイドル状態に遷移させ、Ｉ／Ｏデバイス（コントローラ）等からの割り込みを待つ（ステップＳ１２２）。

【０１６５】

その後、Ｉ／Ｏデバイス（コントローラ）等からの起床可能な割込みが発生するとＤＥＥＰ　ＳＬＥＥＰモードから抜けるので、ＷＦＩ処理部７０１は、ＤＲＡＭにコマンドを送ってセルフリフレッシュモードを解除して通常の動作モードにし（ステップＳ１２３）、次に、プロセッサコア１０１のキャッシュメモリ１４０を初期化して使用可能な状態にする（ステップＳ１２４）。その後、ＷＦＩ処理部７０１は、プロセッサコア１０１を割り込み可能にし（ステップＳ１２５）、上述のステップＳ１１２に制御を移す。

【０１６６】

なお、ＤＥＥＰ　ＳＬＥＥＰモード中にプロセッサコア１０１のキャッシュメモリ１４

０のデータが保持される場合は、上述のステップＳ１２０とステップＳ１２４は不要となる。たとえばｉ．ＭＸ５０８では、ＳＴＯＰモード（ＤＥＥＰ　ＳＬＥＥＰモード）に入るときにキャッシュメモリをパワーゲーティングするかどうかを制御レジスタの設定で選べるようになっているので、その設定に合わせて上述のステップＳ１２０とステップＳ１２４を省略できる。

【０１６７】

図３４は、ハイパーバイザのＩ／Ｏデバイス制御部７１１が、ゲストＯＳによるＩ／Ｏデバイスの制御レジスタのアクセスをトラップして、実ハードウェアのＩ／Ｏデバイスの制御レジスタをアクセスする際の動作例を示すフロー図である。ゲストＯＳによるＩ／Ｏデバイスの制御レジスタのアクセスをトラップする方法は、プラットフォームのハードウェアが仮想計算機支援機構（Virtual Machine Assist）を備えていればハードウェアによって検出できる。別の方法としては、ゲストＯＳの中のＩ／Ｏデバイスの制御レジスタのアクセスする部分を書き換えて、ハイパーバイザのＩ／Ｏデバイス制御部７１１に制御を移すように実施することができる。

【０１６８】

Ｉ／Ｏデバイス制御部７１１がゲストＯＳによるＩ／Ｏデバイスの制御レジスタのアクセスを検出すると、図３４に示すように、まずアクセスしようとしている制御レジスタに対応するＩ／Ｏデバイスは、入出力処理を実行中のときはＳｏＣをＤＥＥＰ　ＳＬＥＥＰモードにすべきでないと判断されるデバイスであって、検出したアクセスは、当該デバイスの入出力動作を開始するアクセスであるかどうかを判断する（ステップＳ１３１）。ステップＳ１３１の結果が肯定であれば（ステップＳ１３１：ＹＥＳ）、Ｉ／Ｏデバイス制御部７１１は、そのＩ／Ｏデバイスが入出力処理を実行中であることをデバイス状態管理部７１２に登録する（ステップＳ１３２）。その後、Ｉ／Ｏデバイス制御部７１１は、ゲストＯＳに代わって、実Ｉ／Ｏデバイスの制御レジスタをアクセスする（ステップ１３３）。

【０１６９】

図３５は、ハイパーバイザの割込み処理部７０２が、実ハードウェアのＩ／Ｏデバイスからの割込みを受けて、その割込みを適切なゲストＯＳにルーティングする際の動作例を示すフロー図である。割込み処理部７０２は、割込み発生を検出すると、まず、その割込みを発生したＩ／Ｏデバイスは、入出力処理を実行中のときはＳｏＣをＤＥＥＰ　ＳＬＥＥＰモードにすべきではないと判断されるデバイスであって、検出した割込みは、当該デバイスが実行していた入出力処理の完了を知らせる割込みであるかどうかを判断する（ステップＳ１４１）。ステップＳ１４１の結果が肯定であれば（ステップＳ１４１：ＹＥＳ）、割込み処理部７０２は、そのＩ／Ｏデバイスが入出力処理を完了したことをデバイス状態管理部７１２に登録する（ステップＳ１４２）。その後、割込み処理部７０２は、その割込みを受け取るべきゲストＯＳに対して割込みを転送する（ステップ１４３）。

【０１７０】

以上のように、本発明の消費電力削減方式をハイパーバイザによって実施する場合、ハイパーバイザへの不正なアクセス（リバースエンジニアリングや改ざん）を防止するために、ＳｏＣの持つセキュアブート機能を使ってハイパーバイザを起動するのに加え、ハイパーバイザのコードの全部あるいは一部（重要な部分）をＳｏＣの内蔵ＲＡＭにロードして実行するのが好ましい。ＳｏＣのセキュアブートは、プログラムをＳｏＣ固有のカギで暗号化して署名（ハッシュ値）を割り当てておき、ブート時に復号するとともに、署名（ハッシュ値）が正しいかどうかを確認することで、不正なプログラムでブートすることを防ぐ。さらに、復号してメモリにロードしたプログラムをＳｏＣの内蔵ＲＡＭに記憶することで、ＳｏＣの外部から観測できる信号線（メインメモリを接続するバスなど）のプローブなどの手段による不正なアクセスを困難にすることができる。また、上述のＨＩＢＥＲＮＡＴＩＯＮモードに入ると内蔵ＲＡＭへの電力供給も停止して内蔵ＲＡＭに記憶されたデータは消失するので、不正なアクセスにさらされる機会を少なくすることができる。

【０１７１】

（変形例）
以上、本発明の実施形態を説明したが、上述の各実施形態は、例として提示したものであり、発明の範囲を限定することは意図していない。これら新規な実施形態は、その他の様々な形態で実施されることが可能であり、発明の要旨を逸脱しない範囲で、種々の省略、置き換え、変更を行うことができる。これら実施形態やその変形は、発明の範囲や要旨に含まれるとともに、特許請求の範囲に記載された発明とその均等の範囲に含まれる。
【０１７２】
上述の各実施形態では、情報処理装置１中の各Ｉ／Ｏデバイス（あるいはデバイスコントローラ）が動作中か否かをデバイス状態管理情報で管理しているが、情報処理装置１内の全てのＩ／Ｏデバイスの状態をデバイス状態管理情報に管理する必要はない。少なくとも、入出力処理の実行や外部からのデータ到着待ちのような動作中のときにＤＥＥＰ　ＳＬＥＥＰモードにすべきでないと判断されるデバイスの状態をデバイス状態管理情報で管理していればよい。ＤＥＥＰ　ＳＬＥＥＰモードでも入力を検出してＳｏＣを起床可能なデバイス（例えばキーパッドなど）は、いつＤＥＥＰ　ＳＬＥＥＰモードに入っても構わないので、このような起床可能デバイスの状態をデバイス状態管理情報で管理する必要はない。
【０１７３】
上述の各実施形態で説明したデバイス状態管理情報は、様々な方式で実施することができる。例えばひとつの情報処理装置１の中で複数の異なる方式のデバイス状態管理情報を混在させて利用しても構わない。例えば、ＲｕｎｔｉｍｅＰＭのフレームワークに基づいて実装されているデバイスドライバを持つＩ／Ｏデバイスは、デバイス状態管理情報として、ＲｕｎｔｉｍｅＰＭで管理する情報（ｒｕｎｔｉｍｅ＿ｓｔａｔｕｓ）を使用し、ＲｕｎｔｉｍｅＰＭのフレームワークに基づいて実装されていない従来型のデバイスドライバを持つデバイスに関しては、図１２に示すようなデバイス状態管理情報を、図１３に示すような手順で更新するように実施することもできる。この場合、上述のデバイス状態判定部は、それぞれのデバイス状態管理情報を見て、どのデバイス状態管理情報においても動作中のＩ／ＯデバイスがなければＤＥＥＰ　ＳＬＥＥＰモードに遷移するように、総合的に判断を行うようにする。
【０１７４】
上述の各実施形態においては、上述のデバイス状態判定部（４１２）は、デバイス状態管理情報を参照して、ＳｏＣをＷＡＩＴモードにするかＤＥＥＰ　ＳＬＥＥＰモードにするかを判断する。このとき、デバイス状態管理情報に加えて、他の情報も加味して判断するように実施することもできる。例えば、デバイス状態管理情報からはＤＥＥＰ　ＳＬＥＥＰモードに遷移して構わない場合でも、タイマーの管理情報からは短い時間内にタイマー割込みが発生することが分かれば、ＷＡＩＴモードに遷移するように実施することができる。ＷＡＩＴモードに遷移するよりもＤＥＥＰ　ＳＬＥＥＰモードに遷移する方が、遷移時の消費電力が大きく遷移時間も長いので、すぐにタイマー割込みが発生することが予測できれば、ＤＥＥＰ　ＳＬＥＥＰモードに遷移できる場合でもＷＡＩＴモードに遷移した方が総合的な消費電力を小さくすることができる。
【０１７５】
上述の各実施形態においては、上述のデバイス状態判定部はＳｏＣをＷＡＩＴモードおよびＤＥＥＰ　ＳＬＥＥＰモードの２つのモードのどちらに遷移させるかを判断しているが、例えばデバイス状態判定部は、２つのモードではなく、３つ以上のモードの中から遷移させるべき一つのモードを判断するように実施することもできる。上述の各実施形態では、ＤＥＥＰ　ＳＬＥＥＰモードではメインクロックの生成（および供給）を停止したり、メインメモリをセルフリフレッシュモードにしたりして、消費電力を削減している。一方、ＷＡＩＴモードでは動作しているＩ／Ｏデバイスが一つ以上存在するので、動作中のデバイスがバスやメインメモリにアクセスする可能性があるため、バスやメインメモリは動作を継続している。しかし、全てのＩ／Ｏデバイスがバスやメインメモリをアクセスするわけではない。そこで、ＷＡＩＴモードと同等だがバスを停止させてメインメモリをセ

ルフリフレッシュモードにする、ＷＡＩＴモードとＤＥＥＰ　ＳＬＥＥＰモードの中間の新たなモード（ここではＷＡＩＴ２モードと呼ぶ）を設けることもできる。例えばデバイス状態判定部は、バスやメインメモリをアクセスするＩ／Ｏデバイスがひとつ以上動作している場合はＷＡＩＴモードに、バスやメインメモリをアクセスしないＩ／Ｏデバイスのみが動作している場合はＷＡＩＴ２モードに、全てのＩ／Ｏデバイスが動作していない場合はＤＥＥＰ　ＳＬＥＥＰモードに遷移させると判断する。このように実施することで、ＷＡＩＴ２モードの消費電力は、ＷＡＩＴモードより小さくＤＥＥＰ　ＳＬＥＥＰモードより大きいので、省電力の効果をより高めることができる。なお、ここでは３つのモードを選択する例を示したが、例えばメインメモリはセルフリフレッシュモードにするがバスは停止しないモードなど、他にも様々なモードを用意して選択するように拡張するのは容易である。

【０１７６】

以上のように、デバイス状態判定部が、複数のモードの中から遷移させるべき一つのモードを判断する場合、デバイス状態判定部は、各Ｉ／Ｏデバイスが動作するときにバスやメインメモリなどの他のデバイスを使用するかどうかを考慮する必要がある。これを実現する方法の一例としては、Ｉ／Ｏデバイスを識別する情報ごとに、当該Ｉ／Ｏデバイスが動作するときに使用する他のデバイスを示す情報を対応付けた対応情報を表などの形式でデバイス状態判定部が記憶しておき、デバイス状態判定部はその対応情報とデバイス状態管理情報を参照して、遷移すべきモードを選択するように実施できる。

【０１７７】

また、別の実施方法としては、上記対応情報を上述のデバイス状態管理部（４２１、４２１０、７１２）が記憶しておき、デバイス状態管理部は、対応情報を参照して、あるＩ／Ｏデバイスの動作開始を登録する際に、そのＩ／Ｏデバイスの動作に必要な他のデバイスについても動作開始を登録するようにする。Ｉ／Ｏデバイスの動作終了を登録する際にも、そのＩ／Ｏデバイスの動作に必要な他のデバイスについても動作終了を登録する。なお、バスやメインメモリのようなデバイスは複数のＩ／Ｏデバイスが共有して使用するので、どのＩ／Ｏデバイスも使用しない場合に動作終了するように管理する必要がある。

【０１７８】

以上の各実施形態および各変形例は任意に組み合わせることが可能である。

【０１７９】

また、上述の情報処理装置１で実行されるプログラムを、インターネット等のネットワークに接続されたコンピュータ上に格納し、ネットワーク経由でダウンロードさせることにより提供するようにしてもよい。また、上述の情報処理装置１で実行されるプログラムを、インターネット等のネットワーク経由で提供または配布するようにしてもよい。また、上述の情報処理装置１で実行されるプログラムを、ＲＯＭ等の不揮発性の記録媒体に予め組み込んで提供するようにしてもよい。

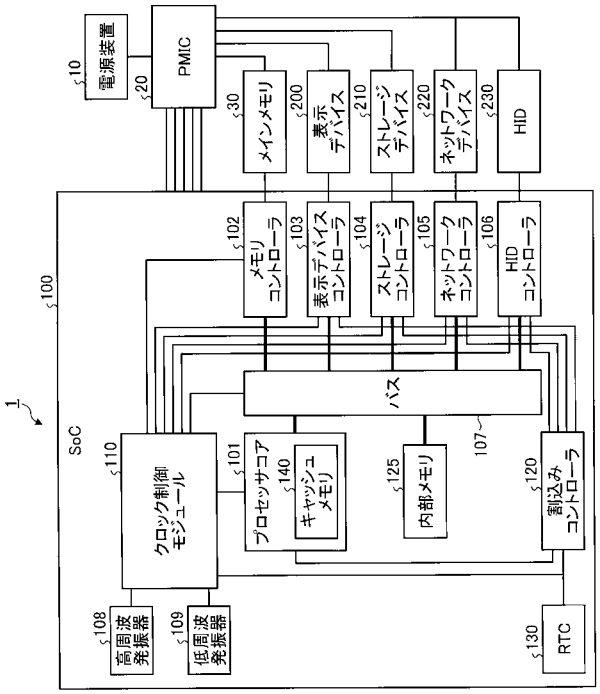
【符号の説明】

【０１８０】

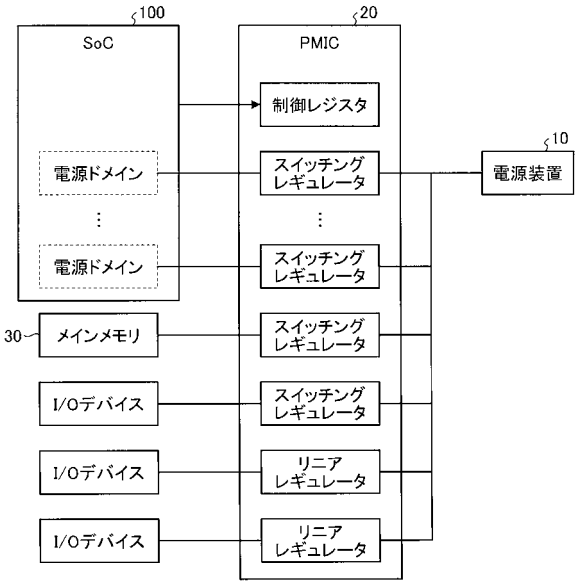
１　　情報処理装置
１０　　電源装置
２０　　ＰＭＩＣ
３０　　メインメモリ
１００　ＳｏＣ
１０１　プロセッサコア
１０８　高周波発振器
１０９　低周波発振器
１１０　クロック制御モジュール
１２０　割込みコントローラ
１２５　内部メモリ
１４０　キャッシュメモリ

３００　マイコン
３０６　割込みコントローラ
４１０　プロセス管理部
４１１　プロセス切替部
４１２　デバイス状態判定部
４２０　デバイス管理部
４２１　デバイス状態管理部
４３０　ファイル管理部
４４０　メモリ管理部

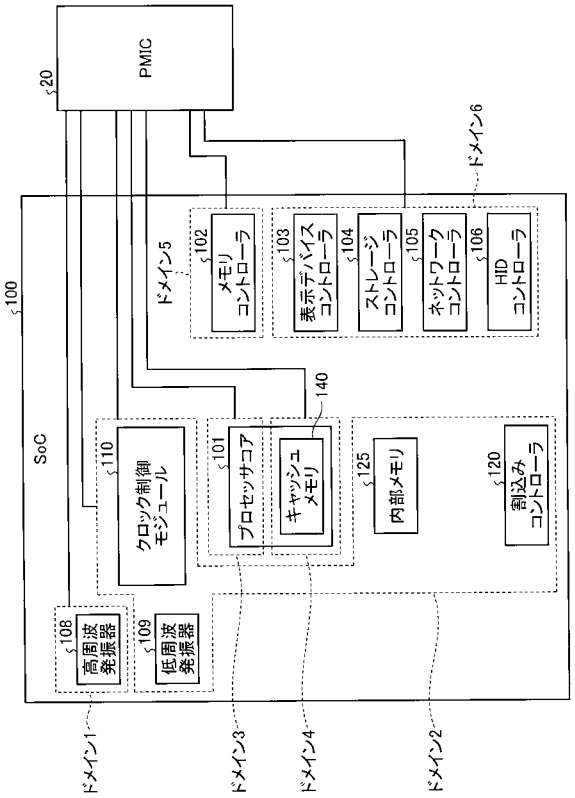
【図１】



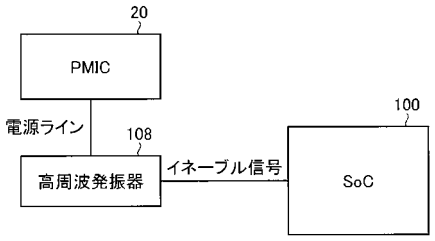
【図２】



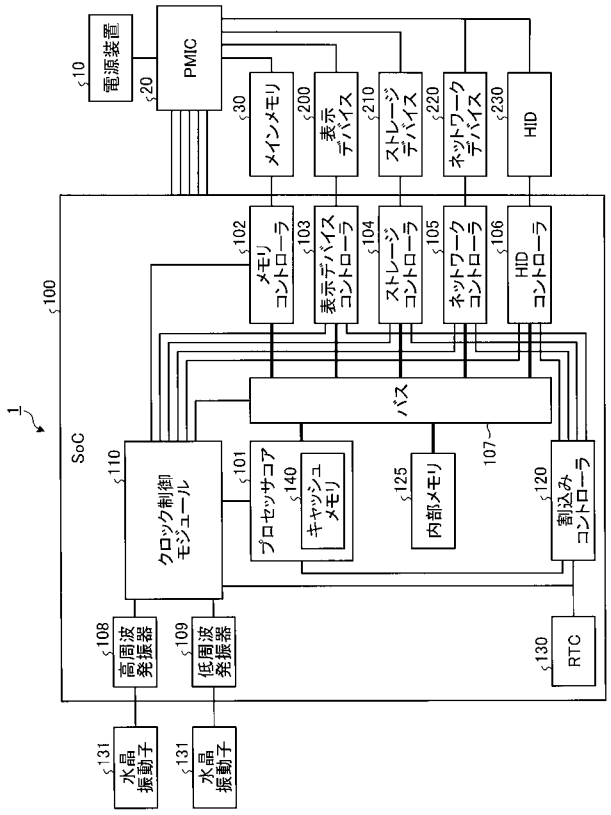
【図３】


PMIC
20
ドメイン6
ドメイン5
100
SoC
102
メモリ
コントローラ
103
表示デバイス
コントローラ
104
ストレージ
コントローラ
105
ネットワーク
コントローラ
106
HID
コントローラ
140
110
クロック制御
モジュール
101
プロセッサコア
キャッシュ
メモリ
125
内部メモリ
120
割込み
コントローラ
108
高周波
発振器
109
低周波
発振器
ドメイン1
ドメイン3
ドメイン4
ドメイン2

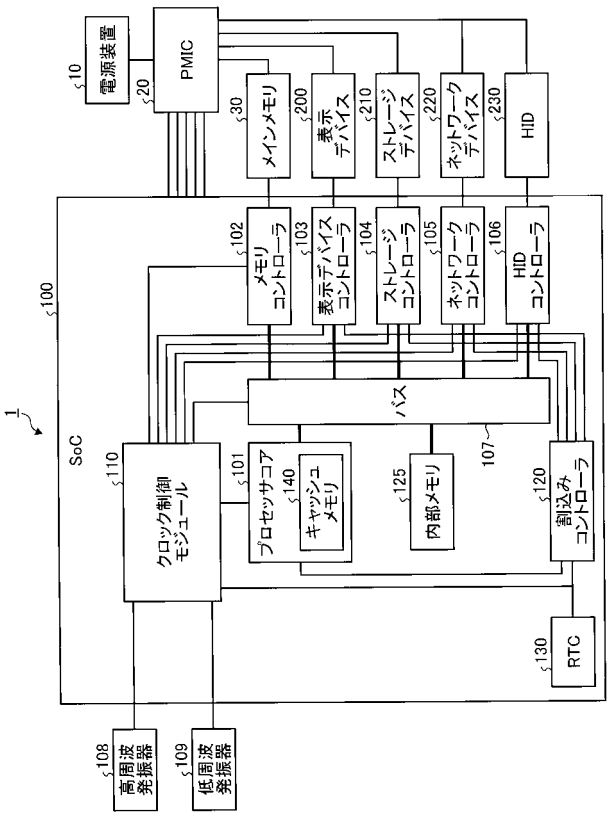
【図４】


20
PMIC
電源ライン
108
高周波発振器
イネーブル信号
100
SoC

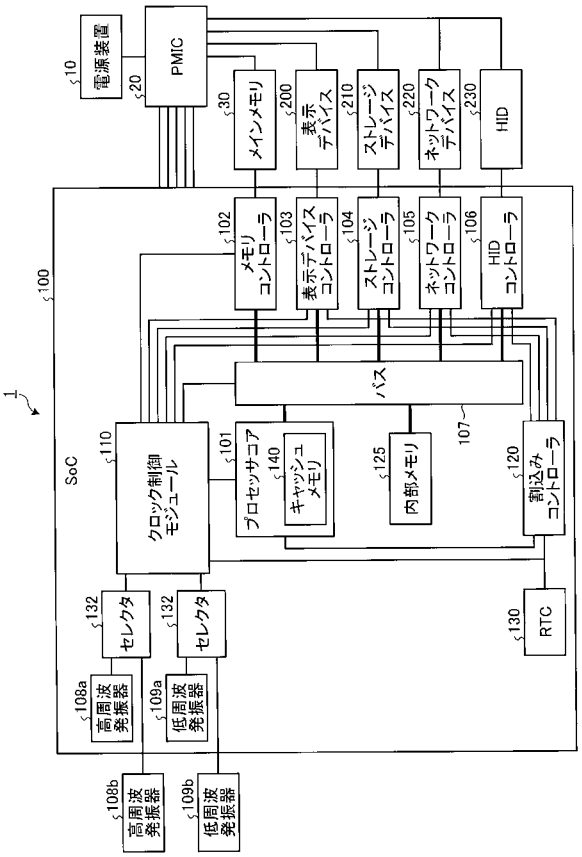
【図５】


1
10
電源装置
20
PMIC
30
メインメモリ
200
表示
デバイス
210
ストレージ
デバイス
220
ネットワーク
デバイス
230
HID
100
SoC
102
メモリ
コントローラ
103
表示デバイス
コントローラ
104
ストレージ
コントローラ
105
ネットワーク
コントローラ
106
HID
コントローラ
バス
107
110
クロック制御
モジュール
101
プロセッサコア
140
キャッシュ
メモリ
125
内部メモリ
120
割込み
コントローラ
130
RTC
108
高周波
発振器
109
低周波
発振器
131
水晶
振動子
131
水晶
振動子

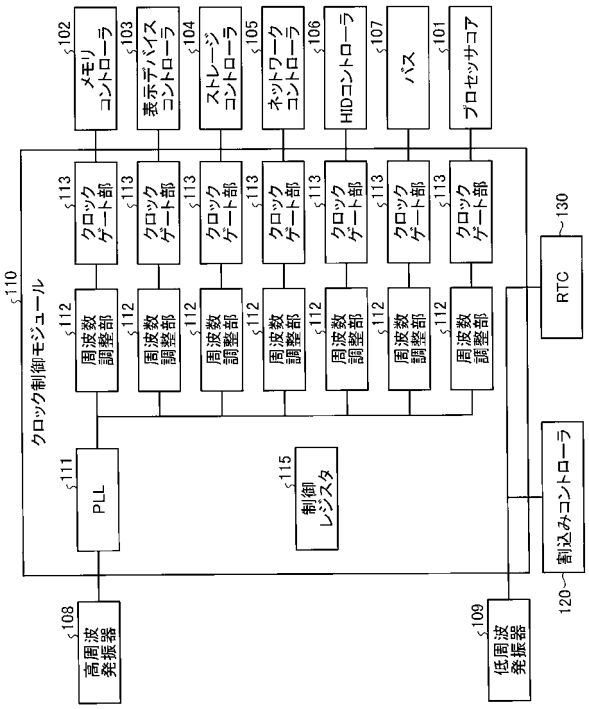
【図６】


1
10
電源装置
20
PMIC
30
メインメモリ
200
表示
デバイス
210
ストレージ
デバイス
220
ネットワーク
デバイス
230
HID
100
SoC
102
メモリ
コントローラ
103
表示デバイス
コントローラ
104
ストレージ
コントローラ
105
ネットワーク
コントローラ
106
HID
コントローラ
バス
107
110
クロック制御
モジュール
101
プロセッサコア
140
キャッシュ
メモリ
125
内部メモリ
120
割込み
コントローラ
130
RTC
108
高周波
発振器
109
低周波
発振器

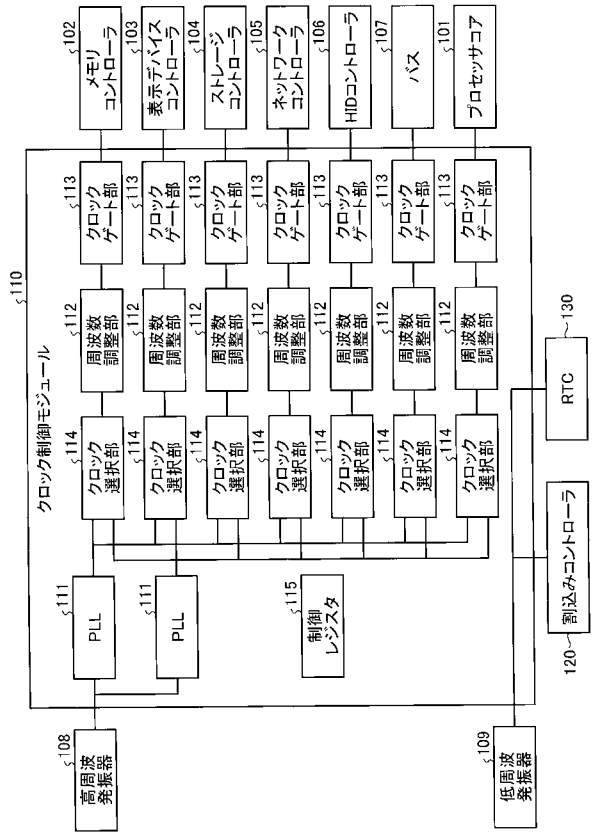
【図７】


1
10 電源装置
20 PMIC
30 メインメモリ
200 表示デバイス
210 ストレージデバイス
220 ネットワークデバイス
230 HID
100 SoC
102 メモリコントローラ
103 表示デバイスコントローラ
104 ストレージコントローラ
105 ネットワークコントローラ
106 HIDコントローラ
バス
107
110 クロック制御モジュール
101 プロセッサコア
140 キャッシュメモリ
125 内部メモリ
120 割込みコントローラ
130 RTC
132 セレクタ
108a 高周波発振器
109a 低周波発振器
108b 高周波発振器
109b 低周波発振器

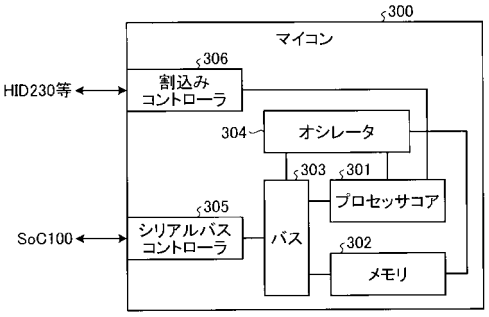
【図８】


102 メモリコントローラ
103 表示デバイスコントローラ
104 ストレージコントローラ
105 ネットワークコントローラ
106 HIDコントローラ
107 バス
101 プロセッサコア
110 クロック制御モジュール
113 クロックゲート部
112 周波数調整部
111 PLL
115 制御レジスタ
130 RTC
120 割込みコントローラ
108 高周波発振器
109 低周波発振器

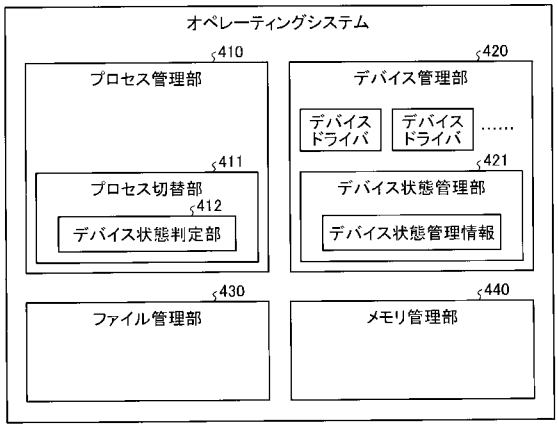
【図９】


102 メモリコントローラ
103 表示デバイスコントローラ
104 ストレージコントローラ
105 ネットワークコントローラ
106 HIDコントローラ
107 バス
101 プロセッサコア
110 クロック制御モジュール
113 クロックゲート部
112 周波数調整部
114 クロック選択部
111 PLL
115 制御レジスタ
130 RTC
120 割込みコントローラ
108 高周波発振器
109 低周波発振器

【図１０】


300 マイコン
HID230等
306 割込みコントローラ
304 オシレータ
303
301 プロセッサコア
SoC100
305 シリアルバスコントローラ
バス
302 メモリ

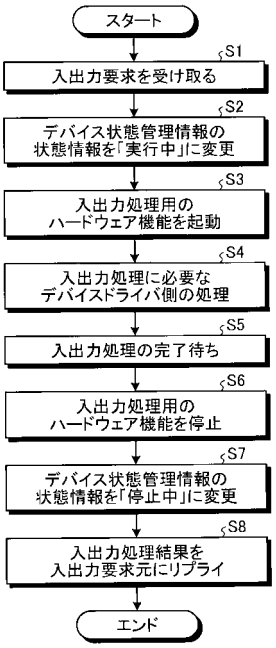
【図１１】



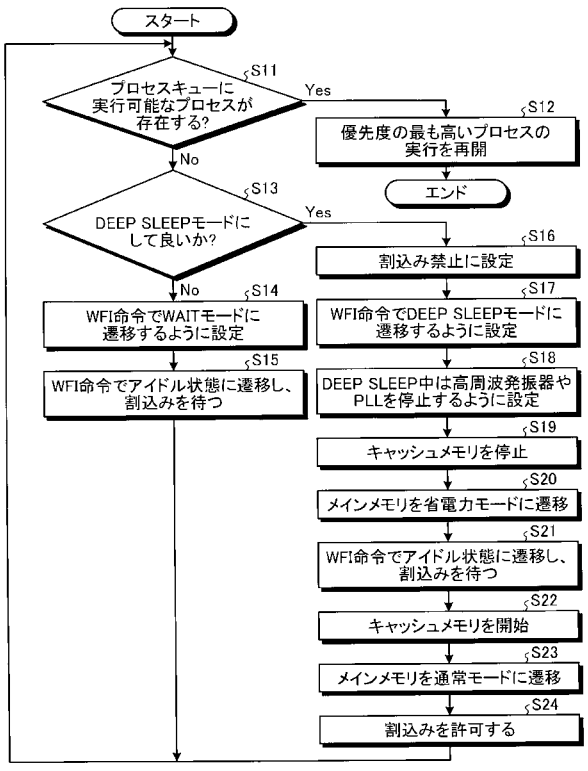
【図１２】

デバイス識別子	状態情報
EPD	実行中
NANDデバイス	停止中
ネットワークデバイス	停止中
キーボード	停止中

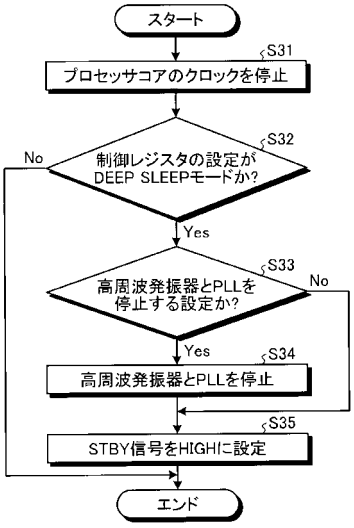
【図１３】



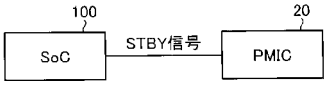
【図１４】



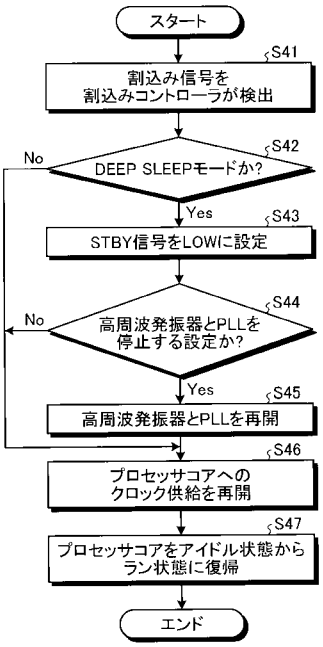
【図１５】



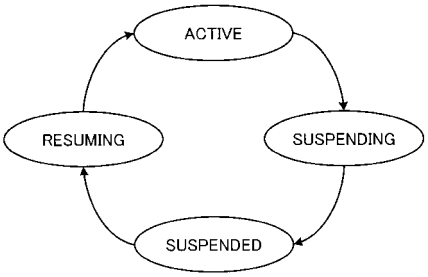
【図１６】



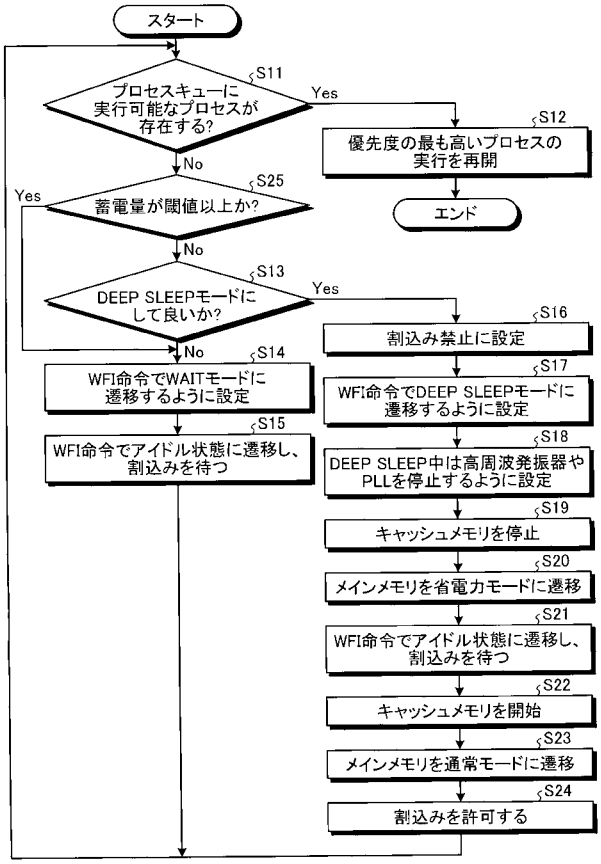
【図１７】



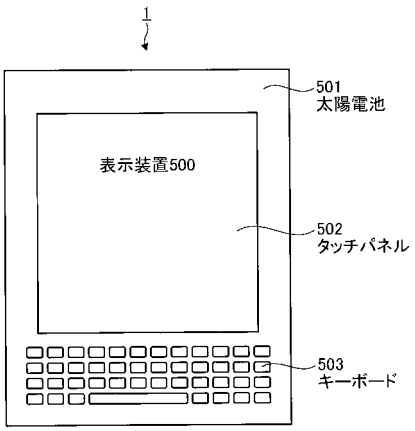
【図１８】



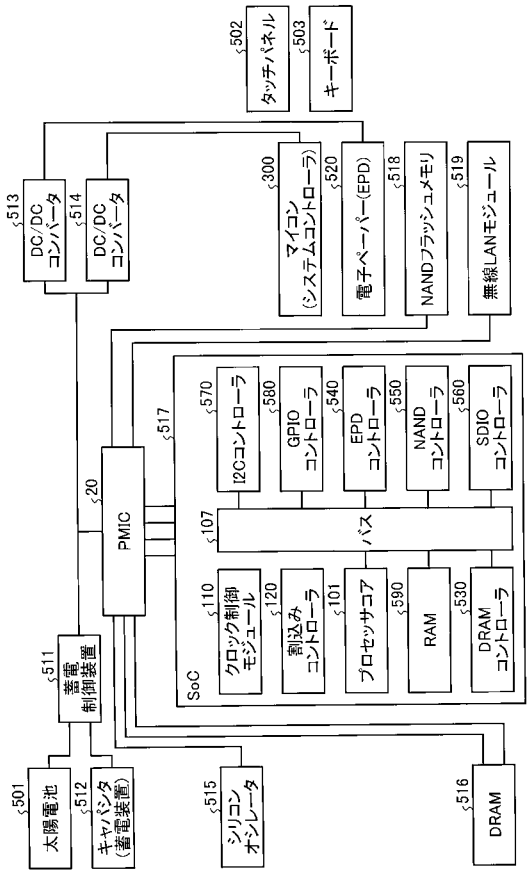
【図１９】



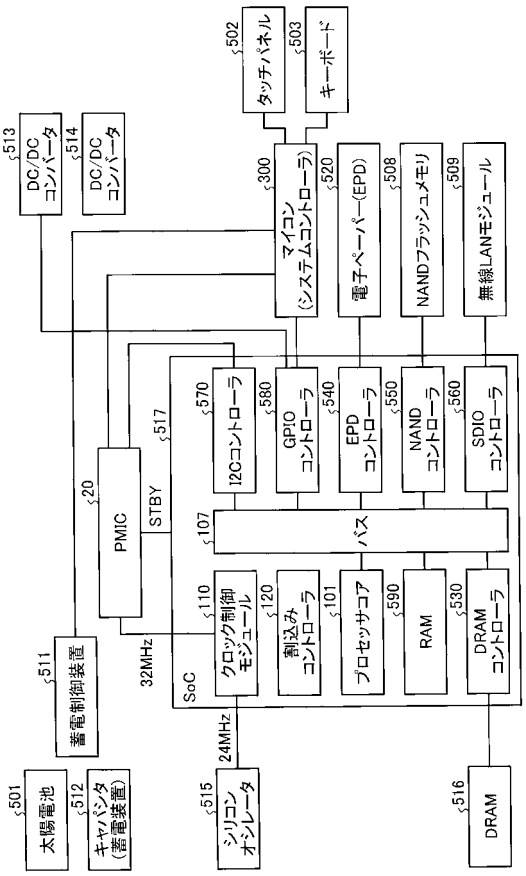
【図２０】



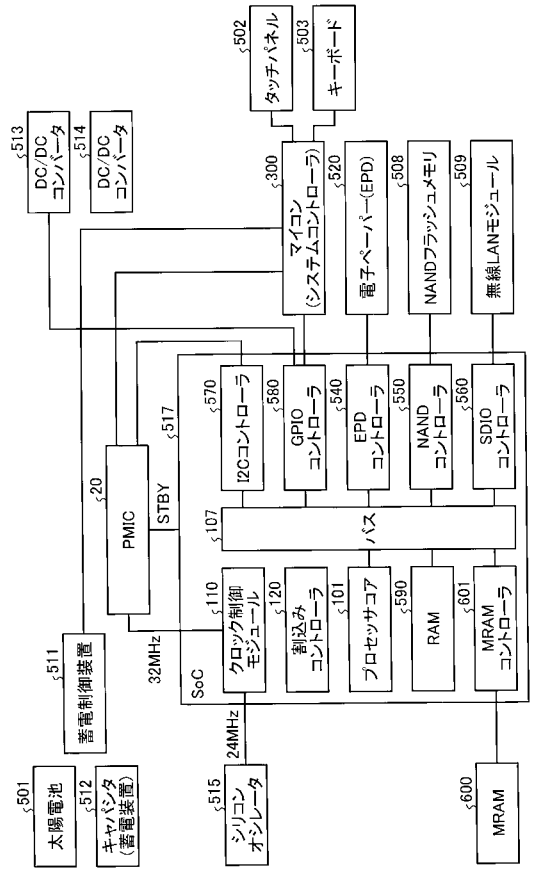
【図２１】



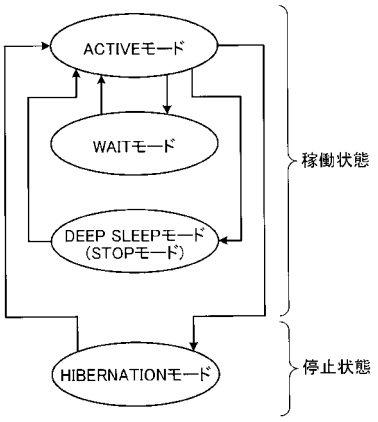
【図２２】



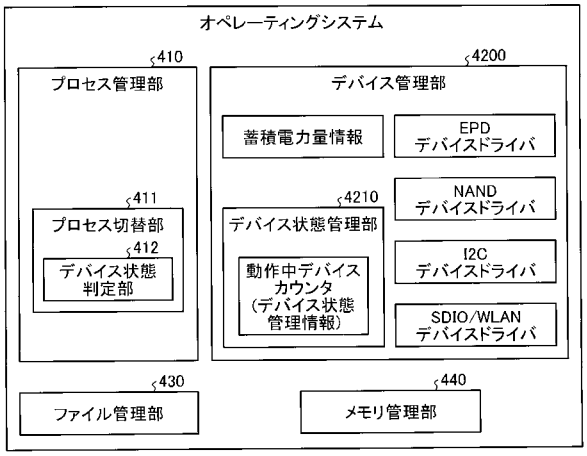
【図２３】



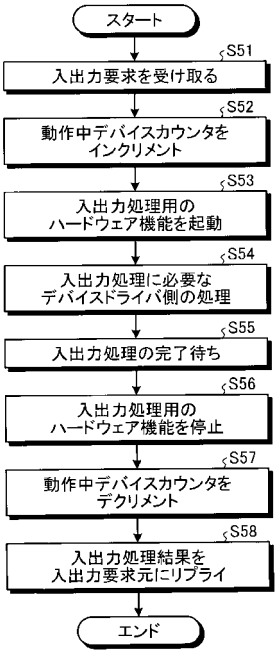
【図２４】



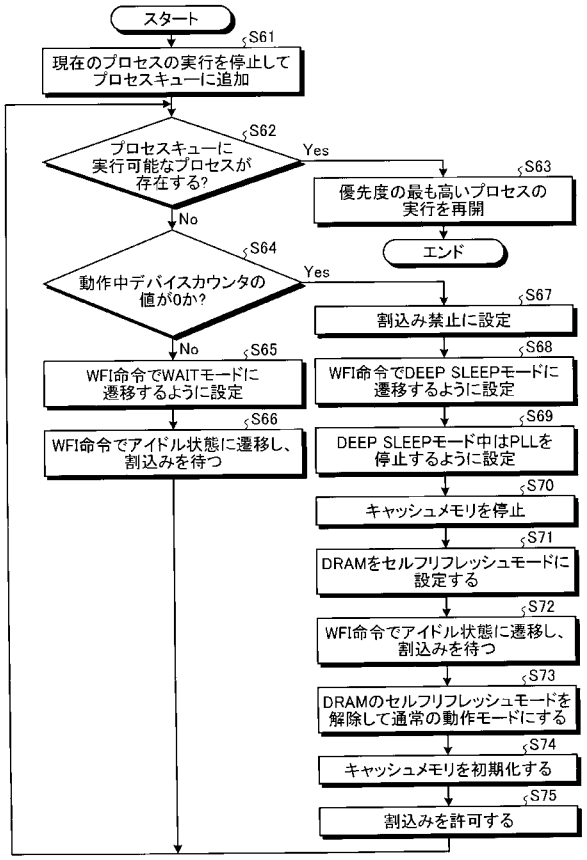
【図２５】



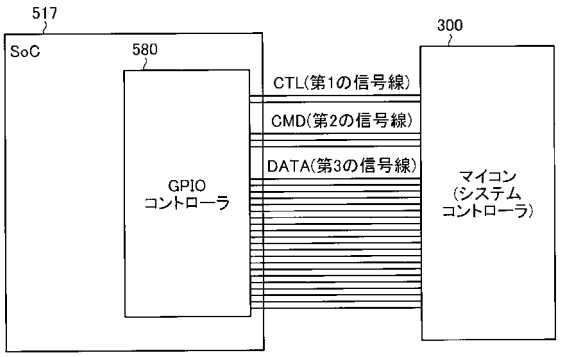
【図２６】



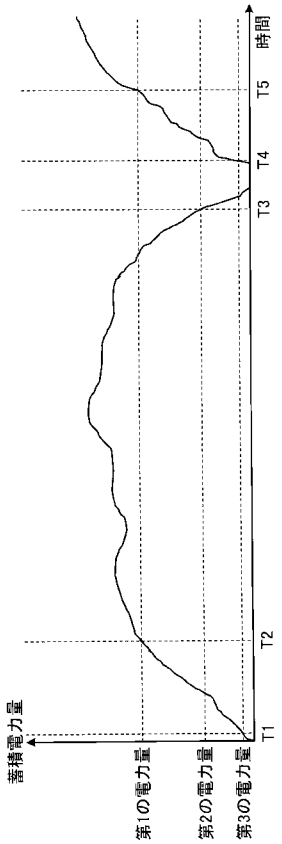
【図２７】



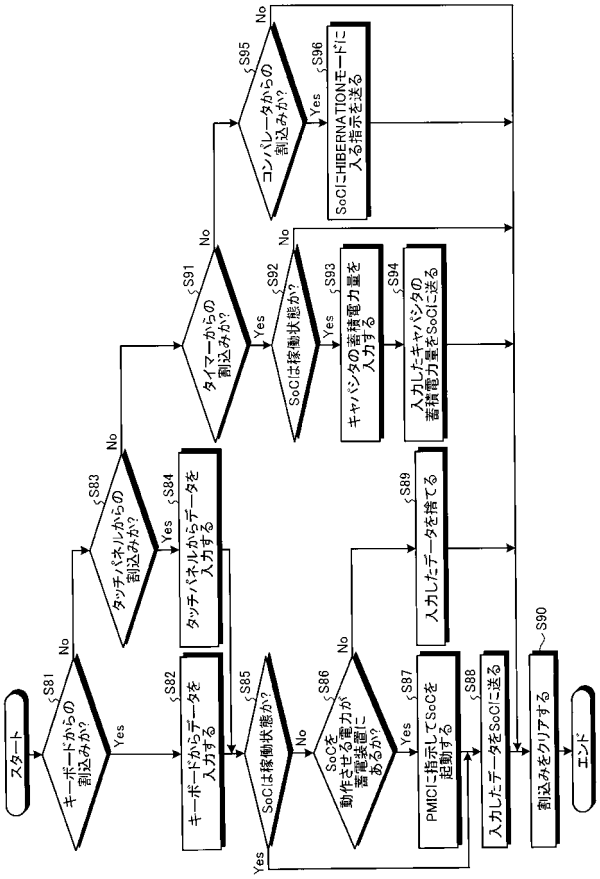
【図２８】



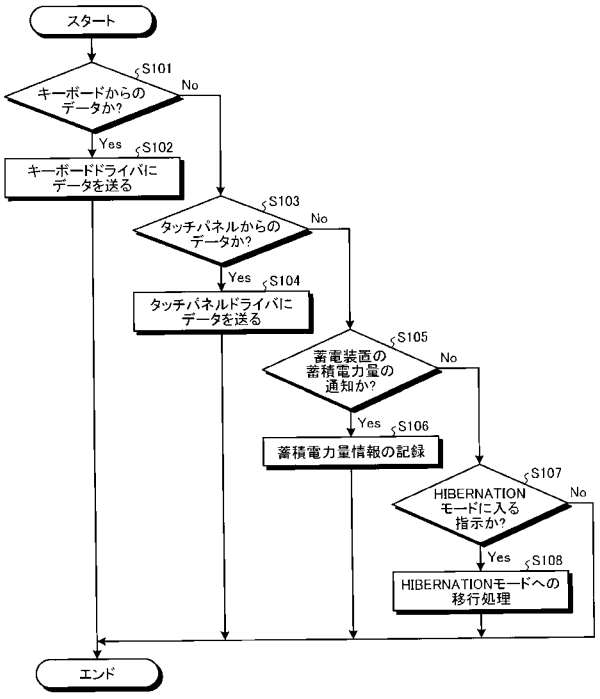
【図２９】



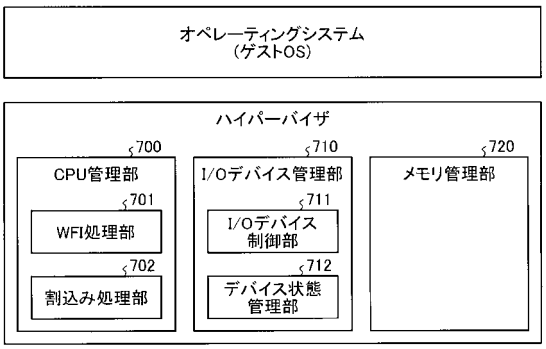
【図３０】



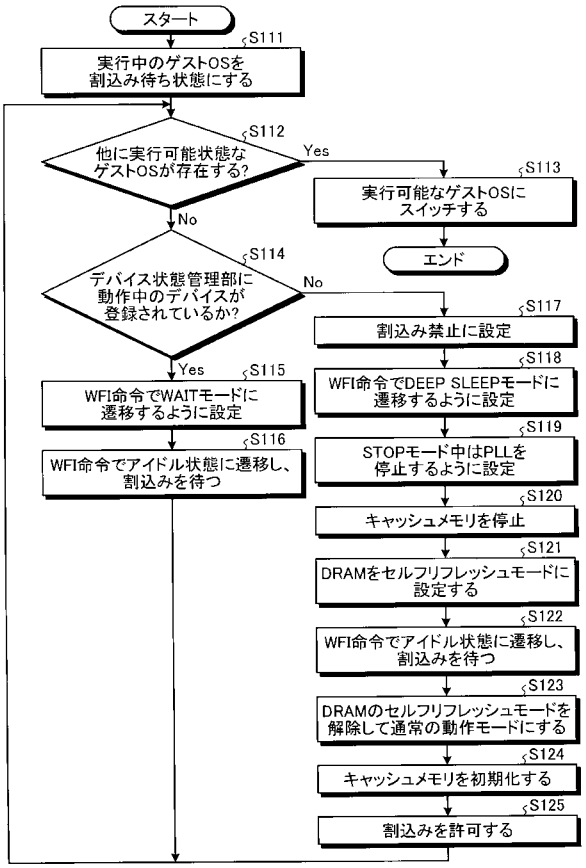
【図３１】



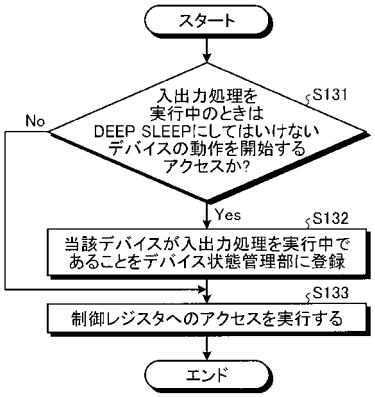
【図３２】



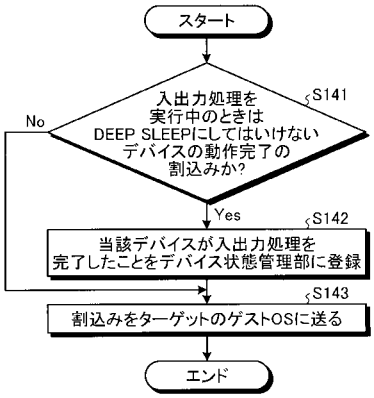
【図３３】


スタート
S111
実行中のゲストOSを
割込み待ち状態にする
S112
他に実行可能状態な
ゲストOSが存在する?
Yes
S113
実行可能なゲストOSに
スイッチする
エンド
No
S114
デバイス状態管理部に
動作中のデバイスが
登録されているか?
No
S117
割込み禁止に設定
Yes
S115
WFI命令でWAITモードに
遷移するように設定
S116
WFI命令でアイドル状態に遷移し、
割込みを待つ
S118
WFI命令でDEEP SLEEPモードに
遷移するように設定
S119
STOPモード中はPLLを
停止するように設定
S120
キャッシュメモリを停止
S121
DRAMをセルフリフレッシュモードに
設定する
S122
WFI命令でアイドル状態に遷移し、
割込みを待つ
S123
DRAMのセルフリフレッシュモードを
解除して通常の動作モードにする
S124
キャッシュメモリを初期化する
S125
割込みを許可する

【図３４】


スタート
S131
入出力処理を
実行中のときは
DEEP SLEEPにしてはいけない
デバイスの動作を開始する
アクセスか?
No
Yes
S132
当該デバイスが入出力処理を実行中で
あることをデバイス状態管理部に登録
S133
制御レジスタへのアクセスを実行する
エンド

【図３５】


スタート
S141
入出力処理を
実行中のときは
DEEP SLEEPにしてはいけない
デバイスの動作完了の
割込みか?
No
Yes
S142
当該デバイスが入出力処理を
完了したことをデバイス状態管理部に登録
S143
割込みをターゲットのゲストOSに送る
エンド

フロントページの続き

(72)発明者　木村　哲郎
東京都港区芝浦一丁目１番１号　株式会社東芝内
(72)発明者　藤崎　浩一
東京都港区芝浦一丁目１番１号　株式会社東芝内
(72)発明者　樽家　昌也
東京都港区芝浦一丁目１番１号　株式会社東芝内
(72)発明者　白井　智
東京都港区芝浦一丁目１番１号　株式会社東芝内
(72)発明者　春木　洋美
東京都港区芝浦一丁目１番１号　株式会社東芝内
(72)発明者　城田　祐介
東京都港区芝浦一丁目１番１号　株式会社東芝内
(72)発明者　柴田　章博
東京都港区芝浦一丁目１番１号　株式会社東芝内
(72)発明者　吉村　礎
東京都港区芝浦一丁目１番１号　株式会社東芝内
(72)発明者　外山　春彦
東京都港区芝浦一丁目１番１号　株式会社東芝内

Ｆターム(参考) 5B011 EA02　KK11　LL11
5B079 BA11　BC01