(12) 特許協力条約に基づいて公開された国際出願

(19) 世界知的所有権機関
国際事務局

(43) 国際公開日
2024年2月1日(01.02.2024)

WIPO | PCT

(10) 国際公開番号
WO 2024/024102 A1

(51) 国際特許分類:
G06F 9/54 (2006.01)

(21) 国際出願番号： PCT/JP2022/029347

(22) 国際出願日： 2022年7月29日(29.07.2022)

(25) 国際出願の言語： 日本語

(26) 国際公開の言語： 日本語

(71) 出願人: 日本電信電話株式会社 (**NIPPON TELEGRAPH AND TELEPHONE CORPORATION**) [JP/JP]; 〒1008116 東京都千代田区大手町一丁目５番１号 Tokyo (JP).

(72) 発明者: 大谷 育生(**OTANI, Ikuo**); 〒1808585 東京都武蔵野市緑町３丁目９－１１ ＮＴＴ 知的財産センタ内 Tokyo (JP). 藤本 圭(**FUJIMOTO, Kei**); 〒1808585 東京都武蔵野市緑町３丁目９－１１ ＮＴＴ 知的財産センタ内 Tokyo (JP).

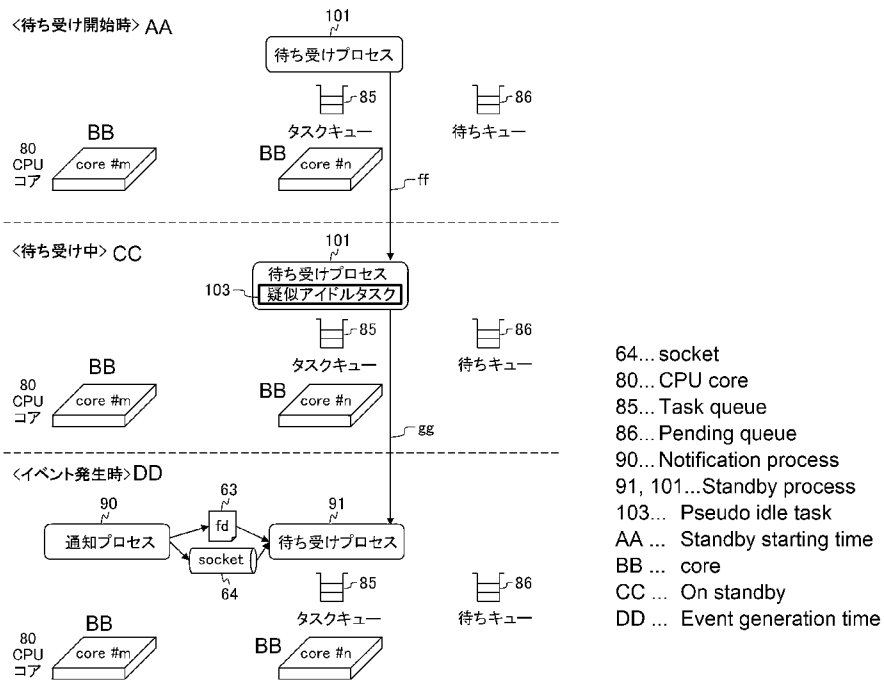
(74) 代理人: 弁理士法人磯野国際特許商標事務所 (**ISONO INTERNATIONAL PATENT OFFICE, P.C.**); 〒1020082 東京都千代田区一番町２１ 一番町東急ビル Tokyo (JP).

(81) 指定国(表示のない限り、全ての種類の国内保護が可能): AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BN, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CV, CZ, DE, DJ, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IQ, IR, IS, IT, JM, JO, JP, KE, KG, KH, KN, KP, KR, KW, KZ, LA, LC, LK, LR, LS, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY,

(54) **Title:** COMPUTER SYSTEM, TASK SCHEDULER DEVICE, PENDING PROCESS AWAKENING METHOD, AND PROGRAM

(54) 発明の名称： 計算機システム、タスクスケジューラ装置、待ちプロセス起床方法およびプログラム

[図5]



(57) **Abstract:** The present invention is a computer system (1000) which has a processor composed of a plurality of cores and in which processes each operate for executing a prescribed command using a time period on the processor. The processes includes: a standby process (101) that is in a standby state until there is a notification and awakens when there is a notification and restarts the operation; and a notification process (90) that transmits a notification for causing awakening from the standby process at an event generation time, wherein the standby process (101) comprises a pseudo idle task

WO 2024/024102 A1

［続葉有］

MZ, NA, NG, NI, NO, NZ, OM, PA, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SA, SC, SD, SE, SG, SK, SL, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, WS, ZA, ZM, ZW.

(84) 指定国(表示のない限り、全ての種類の広域保護が可能): ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, RW, SC, SD, SL, ST, SZ, TZ, UG, ZM, ZW), ユーラシア (AM, AZ, BY, KG, KZ, RU, TJ, TM), ヨーロッパ (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, KM, ML, MR, NE, SN, TD, TG).

添付公開書類:

一 国際調査報告(条約第21条(3))

(103) that continues to exclusively occupy the processor while in the standby state.

(57) 要約:複数のコアからなるプロセッサを有し、当該プロセッサ上の時間を用いて所定の命令を実行するプロセスが動作する計算機システム(１０００)であって、プロセスは、通知があるまで待ち状態となり、通知がある場合に起床して動作を再開する待ち受けプロセス(１０１)と、イベント発生時に待ち受けプロセスを起床させるために通知を送信する通知プロセス(９０)と、を有し、待ち受けプロセス(１０１)は、待ち受け状態に入っているときに、プロセッサを専有し続ける疑似アイドルタスク(１０３)を備える。

明　細　書

発明の名称 ：

計算機システム、タスクスケジューラ装置、待ちプロセス起床方法およびプログラム

技術分野

[0001]　本発明は、計算機システム、タスクスケジューラ装置、待ちプロセス起床方法およびプログラムに関する。

背景技術

[0002]　計算機システムにおいて、計算機（以下、サーバ）上に、汎用プロセッサ（ＣＰＵ）を搭載し、ＣＰＵ上で動作するプロセスを常に動作させる必要がない場合には該当プロセスを停止状態とし、必要なタイミングで別のプロセスから通知を送ることでプロセスを稼働状態に戻すことがある。

[0003]　図１９は、ポーリング方式によるサーバ内のプロセス間通信を示す図である。

図１９に示すように、アプリケーション（以下、適宜「ＡＰＬ」という）［Ａ：待ち側］１から、ＡＰＬ［Ｂ：通知側］１にタスクの処理を依頼する（Ｓ１）。その後、ＡＰＬ［Ａ：待ち側］１は、ポーリングにより処理結果を要求する（Ｓ２）。ＡＰＬ［Ｂ：通知側］１は、ＡＰＬ［Ａ：待ち側］１から依頼されたタスク処理を実行する（Ｓ３）。

[0004]　ＡＰＬ［Ｂ：通知側］１は、依頼されたタスク処理が完了すると、ポーリング要求に応えて処理結果をＡＰＬ［Ａ：待ち側］１に送る（Ｓ４）。ＡＰＬ［Ａ：待ち側］１は、次の処理を実行する（Ｓ５）。

[0005]　図２０は、通知方式によるサーバ内のプロセス間通信を示す図である。図１９と同じ処理には同一ステップ番号を付している。

図２０に示すように、ＡＰＬ［Ａ：待ち側］１から、ＡＰＬ［Ｂ：通知側］１にタスクの処理を依頼する（Ｓ１）。ＡＰＬ［Ａ：待ち側］１は、sleep制御を行い、sleep状態に入る（Ｓ６）。

[0006] ＡＰＬ［Ｂ：通知側］１は、ＡＰＬ［Ａ：待ち側］１から依頼されたタスク処理を実行する（Ｓ３）。

ＡＰＬ［Ｂ：通知側］１は、依頼されたタスク処理が完了すると、ＡＰＬ［Ａ：待ち側］１に処理完了を通知する（Ｓ７）。

[0007] ＡＰＬ［Ａ：待ち側］１は、ＡＰＬ［Ｂ：通知側］１からの処理完了通知を受けて（Ｓ７）、sleep状態から起床し、ＡＰＬ［Ｂ：通知側］１に処理結果を要求する（Ｓ８）。

ＡＰＬ［Ｂ：通知側］１は、ＡＰＬ［Ａ：待ち側］１からの結果要求を受けて、ＡＰＬ［Ａ：待ち側］１に処理結果を送る（Ｓ４）。ＡＰＬ［Ａ：待ち側］１は、次の処理を実行する（Ｓ５）。

[0008] 図１９のポーリング方式では、例えば、あるプロセスＡが別のプロセスＢにタスクの処理を依頼し、プロセスＢにおけるタスク処理が完了するまでの間、完了しているかをポーリングする。プロセスＡが稼働状態であり、消費電力が上がってしまう。

図２０の通知方式では、プロセスＢにタスクの処理を依頼し終わった後に、プロセスＡがsleep状態に入るsleep制御が採られる。プロセスＡかsleep状態に入るので、消費電力を低減することができる。プロセスＡがsleep状態に入ると、プロセスＢが処理完了をプロセスＡに通知し、通知を受けたプロセスＡがsleepから起床し処理完了後のタスクをプロセスＢに対して要求することが行われる。しかし、プロセスＡはsleepから起床するので、復帰のための遅延時間が発生する。

[0009] 同一サーバ内のプロセス間で高速にこのような通知を行うためには、互いに共有したファイルディスクリプタ等のリソースを通じて通知を行うことが一般的である。しかし、プロセスが停止状態から復帰するまではＯＳのスケジューラ（Scheduler）やガバナー（Governor）によって管理されるため、ユーザが予期せぬ遅延時間がかかる可能性がある。

[0010] 既存技術として、Linux OS（登録商標）の中の関数「eventfd+epoll_wait」を用いた通知機構がある。

図２１は、eventfd+epoll_waitを用いた通知システムを示す図である。

図２１に示すように、サーバは、ハードウェア（ＨＷ）１０と、ＯＳ２０と、ユーザ空間上に、待ち受けプロセス[core #n]３０と、通知プロセス［core #m］３２とを有する。待ち受けプロセス[core #n]３０は、図２０では、ＡＰＬ［Ａ：待ち側］１の「プロセスＡ」であり、通知プロセス［core #m］３２は、ＡＰＬ［Ｂ：通知側］１の「プロセスＢ」である。

[0011] ハードウェア（ＨＷ）１０は、ＣＰＵ１１を備える。待ち受けプロセス[core #n]３０は、待ち受け関数３１を有する。

ＯＳ２０は、イベント監視インスタンス２１と、通知用ＦＤ（File Descriptor）２２と、を有する。通知用ＦＤ２２は、ＯＳ２０にアクセスを依頼する際にファイルを指定する整数値である。プログラムからファイルを操作する際、操作対象のファイルを識別・（以下、「・」は「および」を表わす）同定するために割り当てられる。

[0012] 既存技術では、図２１中の１．～９．の流れで通知と起床を行う。

１．待ち受けプロセス３０（「プロセスＡ」）が、イベント監視インスタンス２１(eventpoll)を作成する。

２．待ち受けプロセス３０が、通知用に使うＦＤである通知用ＦＤ２２(eventfd)を作成し、上記１．のイベント監視インスタンス２１に登録する。

３．待ち受けプロセス３０が、通知プロセス３２（「プロセスＢ」）に対して通知用ＦＤ２２を転送する。これにより、待ち受けプロセス３０（「プロセスＡ」）と通知プロセス３２（「プロセスＢ」）間で、この通知用ＦＤ２２を使うことが合意される。

[0013] ４．待ち受けプロセス３０が、上記１．のイベント監視インスタンス２１を指定して待ち受け関数３１(epoll_wait)を発行し、ブロッキング（sleep）状態となる。待ち受け関数３１(epoll_wait)を発行すると、待ち受けプロセス３０は、イベントが発行されるまでは、sleep状態となる。待ち受けプロセス３０自体は、sleep状態になるが、該当ＣＰＵコアの上に他になにもプロセスが乗っていない状態、すなわちidle状態に相当する場合には、待ち受けプロ

セス３０がsleepに入ったときには、アイドルタスクが出現する。このアイドルタスクは、ＣＰＵコアが何もしない状態ではあるものの、プロセスがないときには自動的に出現する。

[0014] ５．＜イベント発生＞

６．通知プロセス３２が、イベント発生を検知して、通知用ＦＤ２２に書き込む。

７．通知用ＦＤ２２に書き込みを行うと、イベント監視コールバックが走る（発生する）。

８．イベント監視コールバックが発生すると、イベント監視インスタンス２１を待ち受けている待ち受け関数３１に指示がいく。指定を受けた待ち受け関数３１は、イベントを検知して起床する。

９．待ち受け関数３１を呼び出した待ち受けプロセス３０は、処理を再開する。

なお、上記２．の通知用ＦＤ作成は、通知プロセス３２側で実行するケースもある。

[0015] 既存技術についてさらに説明する。

図２２は、非特許文献１に記載の既存技術の待ち受けプロセスを説明する図である。

図２２に示すように、待ち受けプロセス４０は、待ち受け部４１と、sleep部４２を有し、アイドルプロセス５０（アイドルタスク）は、アイドル部５１を有する。また、ＯＳのkernel６０は、スケジューラ６１と、C-state制御部６２と、を有する（C-stateについては後記）。

アイドルプロセス５０は、当該ＣＰＵコア上に他のプロセス（スレッド）が存在するとなくなる。

[0016] 既存技術の省電力化方法では、待ち受け部４１がsleep部４２を呼び出し、プロセス（待ち受けプロセス４０）が待ち状態に入る。スケジューラ６１は、ほかに実行可能状態のプロセスがない場合、アイドル部５１を持つプロセスを呼び出す。アイドル部５１は、C-state制御部６２を用いてＣＰＵをslee

p状態へ遷移させる、または電力をＯＦＦする。

[0017] 既存技術では、コンテキストスイッチ（context switch）が必ず発生する。コンテキストスイッチは、複数のプロセスが１つのＣＰＵを共有できるように、ＣＰＵの状態（コンテキスト)を保存／（以下、「／」は「または」を表わす）復元する過程をいう。

[0018] 既存技術では、C-state復帰時間は、kernelが決定する。kernelは、C-state制御部６２が推測したアイドル時間をもとにC-stateの深さを決定する。該当C-stateからの復帰時間が発生する（非特許文献２）。

[0019] 図２３は、C-stateの状態の一例を表にして示す図である。なお、ＣＰＵハードウェアに依って状態定義は異なるため、図２３はあくまでも参考例である。

図２３に示すように、CPUidle状態には、グレードＣ０～Ｃ６があり、ＣＰＵの負荷がない時間が長くなるにつれ、深いsleep状態へ遷移する。深いsleep状態の方がＣＰＵ消費電力は小さくなるが、一方で、それだけ復帰までに要する時間が長延化するため、低遅延の観点で課題となる場合がある。

[0020] C-stateは、ＣＰＵハードウェアに依って状態定義が異なる。例えば、Ｃ４やＣ５が無い機種、Ｃ１の次がＣ１Ｅというステートである機種等のバリエーションがある。

ステートが深くなるにつれ省電力効果は大きくなるが、それだけ、idle状態から復帰に要する時間も大きくなる。

[0021] また、どの深さまでCPUidle状態が遷移するかは、ＣＰＵのハードウェア制御になり、ＣＰＵ製品依存となる（kernel等のソフトウェアから制御できない場合が多い）。

先行技術文献

非特許文献

[0022] 非特許文献1：cpuidle-Do nothing,efficiently,［online］,［令和４年７月６日検索］,インターネット〈URL:http://landley.net/kdocs/ols/2007/ols2007v2-pages-119-126.pdf〉

非特許文献2：CPUの分析，［online］，［令和４年７月６日検索］，インターネット〈URL:https://docs.microsoft.com/ja-jp/windows-hardware/test/wpt/cpu-analysis〉

発明の概要

発明が解決しようとする課題

[0023] 既存技術では、遅延時間増大する課題がある。

（１）コンテキストスイッチ時間による遅延時間増大

図２４は、既存技術の課題を説明する図である。

図２４の下段は、待ち受けプロセス(core #n)のスケジュール、図２４の上段は、通知プロセス(core #m)のスケジュールである。

下記１．～３．を初期化時に実行済みであるとする。

１．イベント監視インスタンス作成

２．通知用ＦＤ作成・イベント監視インスタンス登録

３．通知用ＦＤ転送

[0024] ４．で待ち受けプロセス処理７１がsleep状態になると、該当ＣＰＵコアに対してはアイドルタスクがスケジュールされる可能性がある。図２４では、event receiver (core #n)は、待ち受けプロセス処理７１、および待ち受け関数７２の実行後、コンテキストスイッチ（図２４の符号ａ）によりアイドルタスク７６に移行する。上述したように、待ち受けプロセスがsleep状態になると、該当ＣＰＵコアの上に他になにもプロセスが乗っていないidle状態の場合には、アイドルタスク７６が出現する。アイドルタスク７６が動作することで、ＣＰＵコア（event receiver (core #n)）は段階的に深いスリープ状態（CPU idle）に落ち、消費電力を削減することが可能になる。

[0025] 一方、event sender (core #m)で５．イベント発生すると、６．通知用ＦＤ書き込み７４により、７．イベント監視コールバック７５が実行され、event sender (core #m)は、event receiver (core #n)に対してプロセッサ間割り込み（図２４の符号ｂ）を行う。

[0026] ８．で待ち受け関数が起床する際、再度アイドルタスク７６がスケジュール

され、その後コンテキストスイッチ（図２４の符号ｃ）を経由して待ち受け関数がスケジュールされ、９．待ち受けプロセス処理再開（図２４の番号７８）となる。

コンテキストスイッチは、一般的に、Kernel spaceで動作している状態と、User spaceで動作している状態とを切替える。

このアイドルタスク７６から待ち受け関数起床７７へのコンテキストスイッチ（図２４の符号ｃ）により、大きな遅延が発生する（図２４の☆印ｄ）。すなわち、待ち受け関数に切り替わる際のコンテキストスイッチ時間が発生し、大きな遅延が発生する。

[0027] （２）sleep状態(C-state)復帰による遅延時間増大

CPUidle状態になると、ＯＳのガバナーによって許容されたsleep状態(C-state)になるまで、ＣＰＵは段階的あるいは直接的にsleep状態を遷移させる。

ＯＳのガバナーが、サーバ全体のポリシーとして深いsleep状態を許容する場合（例えば、Ｃ６）、ＣＰＵの稼働状態（Ｃ０）に戻るまでに100ｕｓ以上の長い復帰時間が発生する。

[0028] 図２５は、図２４のスケジュールに、演算に使用するＣＰＵコアのC-state遷移イメージを重ね合わせて示した図である。

図２５に示すように、タスクなし時間が長い場合は、ＣＰＵコアが深いsleep状態（CPUidle状態：グレードＣ６）にあり、タスク発生（図２５の符号ｅ）後、復帰までの遅延時間が長延化する。深いCPU idle stateまで落ちると、復帰するまでに大きな時間を要し、リアルタイム性が損なわれるという課題がある。

[0029] 上記課題は、基地局（ＢＢＵ）のようにリアルタイム性が最優先されるシステムでは、看過できない問題である。

このため、基地局（ＢＢＵ）にあっては、C-stateを無効にする、若しくは、idle stateの遷移をＣ１等の限られた深さに限定する設定を投入する対応が採られる。すなわち、省電力性を犠牲にし、リアルタイム性を指向するチューニングが行われる場合がある。

[0030] このような背景を鑑みて本発明がなされたのであり、本発明は、イベント発生後に待ち受けプロセスが起床するまでにかかる時間を低減することを課題とする。

課題を解決するための手段

[0031] 前記した課題を解決するため、複数のコアからなるプロセッサを有し、当該プロセッサ上の時間を用いて所定の命令を実行するプロセスが動作する計算機システムであって、前記プロセスは、通知があるまで待ち状態となり、通知がある場合に起床して動作を再開する待ち受けプロセスと、イベント発生時に待ち受けプロセスを起床させるために通知を送信する通知プロセスと、を有し、前記待ち受けプロセスは、待ち受け状態に入っているときに、前記プロセッサを専有し続ける疑似アイドルタスクを有することを特徴とする計算機システムとした。

発明の効果

[0032] 本発明によれば、イベント発生後に待ち受けプロセスが起床するまでにかかる時間を低減することができる。

図面の簡単な説明

[0033] [図1]本発明の実施形態に係る計算機システムの概略構成図である。

[図2]本発明の実施形態に係る計算機システムのコンテキストスイッチ排除（特徴<１>）を説明する図である。

[図3]本発明の実施形態に係る計算機システムの演算に使用するＣＰＵコアのスケジュールに、ＣＰＵコアのC-state遷移イメージを重ね合わせて示した図である。

[図4]本発明の実施形態に係る計算機システムの比較例における<待ち受け開始時>、<待ち受け中>および<イベント発生時>の待ち受けプロセスの挙動を示す図である。

[図5]本発明の実施形態に係る計算機システムの<待ち受け開始時>、<待ち受け中>および<イベント発生時>の待ち受けプロセスの挙動を示す図である。

[図6]本発明の実施形態に係る計算機システムの比較例における<待ち受け開

始時〉、〈待ち受け中〉および〈イベント発生時〉の待ち受けプロセスの挙動を示す図である。

[図7]本発明の実施形態に係る計算機システムの〈待ち受け開始時〉、〈待ち受け中〉および〈イベント発生時〉の待ち受けプロセスの挙動を示す図である。

[図8]本発明の実施形態に係る計算機システムの比較例における実装イメージを示す図である。

[図9]本発明の実施形態に係る計算機システムの実装イメージを示す図である。

[図10]本発明の実施形態に係る計算機システムの適用範囲１を説明する図である。

[図11]本発明の実施形態に係る計算機システムの適用範囲２を説明する図である。

[図12]本発明の実施形態に係る計算機システムのイベント発生前の疑似アイドル機能が必要なケースにおけるシーケンスである。

[図13]本発明の実施形態に係る計算機システムのイベント発生前の疑似アイドル機能が不要なケースにおけるシーケンスである。

[図14]本発明の実施形態に係る計算機システムのイベント発生後の疑似アイドル機能が必要なケースにおけるシーケンスである。

[図15]本発明の実施形態に係る計算機システムの疑似アイドル判断部における疑似アイドル機能を実行するフローチャートである。

[図16]本発明の実施形態に係る計算機システムの復帰時間決定部における最大復帰時間決定のフローチャートである。

[図17]本発明の実施形態に係る計算機システムの最大復帰時間決定処理を実行した場合の動作例を示す図である。

[図18]本発明の実施形態に係る計算機システムのタスクスケジューラ装置の機能を実現するコンピュータの一例を示すハードウェア構成図である。

[図19]ポーリング方式によるサーバ内のプロセス間通信を示す図である。

[図20]通知方式によるサーバ内のプロセス間通信を示す図である。

[図21]eventfd+epoll_waitを用いた通知システムを示す図である。

[図22]既存技術の待ち受けプロセスを説明する図である。

[図23]C-stateの状態の一例を表にして示す図である。

[図24]既存技術の課題を説明する図である。

[図25]図２４のスケジュールに、演算に使用するＣＰＵコアのC-state遷移イメージを重ね合わせて示した図である。

発明を実施するための形態

[0034] 以下、図面を参照して本発明を実施するための形態（以下、「本実施形態」という）における計算機システム等について説明する。

［概要］

図１は、本発明の実施形態に係る計算機システムの概略構成図である。図２２と同一構成部分には、同一符号を付している。

本実施形態は、計算機システムとしてＣＰＵに適用した例である。ＣＰＵ以外にも、ＧＰＵ（Graphic Processing Unit），ＦＰＧＡ（Field Programmable Gate Array），ＡＳＩＣ（Application Specific Integrated Circuit）等のプロセッサに、idle stateの機能がある場合には、同様に適用可能である。

[0035] 図１に示すように、計算機システム１０００は、kernel６０と、ＣＰＵコア（core #m, core #n, core #x）８０と、通知プロセス９０と、アイドルプロセス５０と、タスクスケジューラ装置１００と、を有する。

[0036] kernel６０は、スケジューラ６１と、C-state制御部６２と、eventfd等のfd（file descriptor：ファイル記述子）６３と、Unix Domain Socket等のsocket（通信用ソケット）６４と、を有する。

[0037] アイドルプロセス５０は、アイドル部５１を有する。アイドルプロセス５０は、待ち受け中において、他のプロセス（スレッド）が該当ＣＰＵコアを使用できるようにする。

[0038] 計算機システム１０００は、複数のコアからなるプロセッサを有し、当該プロセッサ上の時間を用いて所定の命令を実行するプロセスが動作し、その

プロセスは、通知があるまで待ち状態となり、通知がある場合に起床して動作を再開する待ち受けプロセス１０１と、イベント発生時に待ち受けプロセスを起床させるために通知を送信する通知プロセス９０と、プロセッサ上にプロセスがないとき発生し何も実行しないアイドルプロセス５０と、を有する計算機システムである。

[0039] ［タスクスケジューラ装置１００］

タスクスケジューラ装置１００は、待受部１１０と、疑似アイドル部１２０と、疑似アイドル判断部１３０と、復帰時間決定部１４０（復帰時間制御部）と、復帰時間設定部１５０（復帰時間制御部）と、を備える。

上記待受部１１０および疑似アイドル部１２０は、待ち受けプロセス１０１を構成し、アイドルプロセス５０（アイドルタスク）のアイドル部５１と連携する。

上記疑似アイドル判断部１３０、復帰時間決定部１４０、および復帰時間設定部１５０は、管理部１０２を構成する。

[0040] 待ち受けプロセス１０１は、通知があるまで待ち状態となり、通知がある場合に起床して動作を再開する。

待受部１１０は、実行可能状態を維持する。待受部１１０は、疑似アイドル判断部１３０からの疑似アイドル機能要求を受けて、待ち受け設定および実行可能状態設定を行う。具体的には、疑似アイドル判断部１３０の指示により、疑似アイドル部１２０を実行するか、アイドル機能を持つアイドルプロセスにコンテキストスイッチを行う。

[0041] 疑似アイドル部１２０は、何も実行しないsleep（ＮＯＰ）状態を維持する。疑似アイドル部１２０は、アイドル部５１の代わりにＣＰＵの稼働状態を制御するC-state制御部６２を呼び出して実行する。

疑似アイドル部１２０は、待ち受け状態に入っているときに、待ち受けプロセス１０１が、プロセッサを専有し続ける疑似アイドルタスク１０３（図５、図７）を有する。

ここで、Kernelのスケジューラは、イベント発生時に待ち受けプロセス１

０１に実行時間を与える。直前に実行されている疑似アイドルタスク１０３は待ち受けプロセスの延長で実行されているため、プロセス切替に伴うコンテキストスイッチを排除することができる。

[0042] 疑似アイドル判断部１３０は、同一プロセッサ上のプロセス生起状態、および／または、遅延時間の要件をもとに、疑似アイドル部１２０（疑似アイドルタスク）を実行するかを決定し、待受部１１０に指示する。ここで、疑似アイドル部１２０を実行するかの判断が必要な理由は、図１０で後記するように、通知プロセスと待ち受けプロセスが同一ＣＰＵコア上で動作している状況などにおいては、疑似アイドル部を動作させてコンテキストスイッチを排除する効果がない上、他のプロセスにも処理遅延増大などの影響を及ぼす可能性があるためである。

[0043] 復帰時間決定部１４０は、遅延時間の要件や現状のC-stateから、疑似アイドル部１２０が最大で落ちることのできるC-stateに対応する復帰時間を決定する。復帰時間決定部１４０は、最大復帰時間を復帰時間設定部１５０に通知する。

[0044] 復帰時間設定部１５０は、復帰時間決定部１４０からの最大復帰時間を用いて、C-state制御部６２に最大復帰時間を設定する。復帰時間設定部１５０は、疑似アイドル部１２０が終了後に再度、元の最大復帰時間に戻す。

[0045] 上記、復帰時間決定部１４０および復帰時間設定部１５０は、疑似アイドルタスク１０３がタスク発生後に復帰するまでの復帰時間を設定する復帰時間設定部を構成する。この復帰時間制御部は、決められた復帰時間を最大復帰時間で上書きし、最大復帰時間を短くとることで深いsleep状態へ遷移させない機能部である。

[0046] 以下、上述のように構成された計算機システム１０００の動作を説明する。

［タスクスケジューラ装置１００動作の基本的な考え方］

特徴<1>：コンテキストスイッチ排除

待ち受けプロセスにアイドルタスクと同じ振る舞いをさせ、アイドルタス

クからのコンテキストスイッチを排除する。

待ち受け関数は、そのコンテキストの中でアイドルタスクと同等の動作をするため、既存のアイドルタスクへのコンテキストスイッチが発生しない。これにより、〈要件１：通知時間低減〉を満たす。

待ち受け関数が動作する間はアイドルタスクと同様に、ＣＰＵコアは、深いスリープ状態に落ちることが可能である。これにより、〈要件２：省電力性〉を満たす。

[0047] 図２は、コンテキストスイッチ排除（特徴〈１〉）を説明する図である。図２４と同じプロセスには、同一符号を付している。

図２の下段は、event receiver（core #n)のスケジュール、図２の上段は、event sender（core #m)のスケジュールである。

下記１．～３．を初期化時に実行済みであるとする。

１．イベント監視インスタンス作成

２．通知用ＦＤ作成・イベント監視インスタンス登録

３．通知用ＦＤ転送

[0048] event receiver（core #n)は、待ち受けプロセス処理７１後、アイドルタスクへ遷移させずに（コンテキストスイッチを発生させずに）（図２の符号ａａ）、４．待ち受け関数・８．待ち受け関数起床２０１となる。すなわち、待ち受け関数・待ち受け関数起床２０１では、４．待ち受け関数がアイドルタスクと同等にふるまう（図２の符号ｂｂ）。

[0049] 一方、event sender（core #m)で５．イベント発生すると、６．通知用ＦＤ書き込み７４により、７．イベント監視コールバック７５が実行され、event sender（core #m)は、event receiver（core #n)に対してプロセッサ間割り込み（図２の符号ｂ）を行う。

[0050] ４．待ち受け関数がアイドルタスクと同等にふるまうので、８．待ち受け関数起床の際、アイドルタスクのスケジュールが行われず、従ってコンテキストスイッチ（図２の符号ｃｃ）は発生せずに、９．待ち受けプロセス処理再開７８となる。

[0051] 待ち受け関数は、そのコンテキストの中でアイドルタスクと同等の動作をするため、既存のアイドルタスクへのコンテキストスイッチ（図２４の符号ｃ）が発生しない。これにより、〈要件１：通知時間低減〉を満たすことができる。

ここで、図２には示されていないが、待ち受け関数が動作する間はアイドルタスク（図２４の番号７６）と同様に、ＣＰＵコアは、深いスリープ状態に落ちることが可能である。これにより、〈要件２：省電力性〉を満たすことができる。

[0052] 特徴〈２〉：疑似アイドルタスクの復帰時間指定

疑似アイドルタスクの最大の復帰時間を指定し、深いsleep状態に落ちないように制御する。

ＯＳのガバナーによって決められた復帰時間を最大復帰時間で上書きし、最大復帰時間を短くとる。これにより、計算機システムは、深いsleep状態へ遷移しないため、待ち受け関数起床までの遅延時間を短くすることが可能である。これにより、〈要件１：通知時間低減〉を満たす。

一方、最大復帰時間を長くとることで、深いsleep状態へ遷移させ、ＨＷにおいて電力をより削減することが可能である。これにより、〈要件２：省電力性〉を満たす。

[0053] 図３は、図２のスケジュールに、演算に使用するＣＰＵコアのC-state遷移イメージを重ね合わせて示した図である。図２５と同じプロセスには、同一符号を付している。

図３に示すように、タスク発生（図３の符号ｄｄ）から復帰までの復帰時間最大値（図３の符号ｅｅ）を指定する。このような復帰時間最大値指定が可能になるのは、待ち受け関数がアイドルタスクと同等にふるまうようにした待ち受け関数・待ち受け関数起床２０１を設け、この待ち受け関数・待ち受け関数起床２０１のスケジュールに自由度を持たすことができるからである。

上記復帰時間最大値指定について、最大復帰時間を短く設定する場合は、

ＣＰＵコアは浅いsleep状態に落ちるので、電力削減は小さい。また、最大復帰時間を長く設定する場合は、深いsleep状態に落ちるので、電力削減は大きい。ただし、ＣＰＵコアが深いsleep状態まで落ちると、復帰するまで時間を要する。

[0054] ［タスクスケジューラ装置１００動作］

［動作１］

タスクスケジューラ装置１００の動作１について、比較例（既存技術）と対比しながら説明する。

まず、比較例（既存技術）の動作について述べる。

図４は、比較例における〈待ち受け開始時〉、〈待ち受け中〉および〈イベント発生時〉の待ち受けプロセスの挙動を示す図である。図１と同一構成・プロセスには同じ符号を付している。

[0055] 〈待ち受け開始時〉

サーバは、ＣＰＵコア８０（event receiver（core #n））と、ＣＰＵコア８０（event sender（core #m））とを有する。待ち受け開始時、待ち受けプロセス９１は、ＣＰＵコア８０（core #n）上で、タスクキュー８５に入っている。

[0056] 〈待ち受け中〉

いわゆる待ち状態とは、ＣＰＵコア８０（core #n）上で、待ち受けプロセス９１が、待ちキュー８６の中に入ることをいう。

待ち受け中は、待ち受けプロセス９１は、タスクキュー８５から抜け（図４の矢印ｆ）、図４の右側の待ちキュー８６に入り待ち状態となる。

[0057] ここで、ＣＰＵコア８０（core #n）の上に他になにもプロセスが乗っていない状態、すなわちidle状態に相当する場合には、ＣＰＵコア８０（core #n）上にアイドルタスク９２が出現する。このアイドルタスク９２は、タスクキュー８５が空にならないよう、自動で出現するプロセス（タスク）である。

[0058] 〈イベント発生時〉

kernelモードによるevent発行検知時に、通知プロセス９０からkernel６０（図１）のfd６３およびsocket６４を介して通知を受け、待ち受けプロセス９１は、再び、図４の中央のタスクキュー８５に戻る（図４の矢印ｇ）。待ち受けプロセス９１がタスクキュー８５に戻るときに、アイドルタスク９２と待ち受けプロセス９１のコンテキストスイッチが発生する（図２４の符号ｃ）。

[0059] 次に、本実施形態の動作について述べる。

図５は、本実施形態における〈待ち受け開始時〉、〈待ち受け中〉および〈イベント発生時〉の待ち受けプロセスの挙動を示す図である。図１および図４と同一構成・プロセスには同じ符号を付している。

[0060] 〈待ち受け開始時〉

サーバは、ＣＰＵコア８０（event receiver（core #n））と、ＣＰＵコア８０（event sender（core #m））とを有する。待ち受けプロセス１０１は、ＣＰＵコア８０（core #n)上でタスクキュー８５に入っている。〈待ち受け開始時〉では、待ち受けプロセスの挙動は、本実施形態と比較例で同じである。

[0061] 〈待ち受け中〉

図４の比較例で述べたように、待ち受けプロセス９１がタスクキュー８５に戻るときに、アイドルタスク９２と待ち受けプロセス９１のコンテキストスイッチが発生していた（図２４の符号ｃ）。これを解決するには、アイドルタスク９２が発生しないようにすればよい。

[0062] そこで、本実施形態では、待ち受けプロセス１０１は、待受部１１０および疑似アイドル部１２０を有する（図１）。すなわち、待ち受けプロセス１０１は、図４の待ち受けプロセス９１にさらに、疑似アイドル部１２０（疑似アイドルタスク１０３）を導入する。疑似アイドルタスク１０３は、アイドルタスク９２を発生させないため、待ち受けプロセス１０１をタスクキュー８５に入らせないように、ＣＰＵコア８０（core #n)を専有し続ける。すなわち、疑似アイドルタスク１０３の導入により、他のプロセスに該当ＣＰＵコアを明け渡さずに、該当ＣＰＵコアを専有し続ける。ＣＰＵコアの専有

は、より詳細には、ＣＰＵタイムを譲らずに専有し続けることである。具体的には、疑似アイドルタスク１０３は、epollを発行するが、他プロセス（スレッド）にＣＰＵタイムを譲らずに専有し続ける。

[0063] 図５の符号ｆｆに示すように、待ち受けプロセス１０１は、タスクキュー８５に残り続ける。すなわち、図４の比較例では、待ち受け中は、待ち受けプロセス９１が、タスクキュー８５から抜け、待ちキュー８６に入り待ち状態となっていた。これに対し、本実施形態では、疑似アイドルタスク１０３は、epollを発行するものの、他プロセス（スレッド）にＣＰＵタイムを譲らずに専有し続けることで、待ち受けプロセス１０１は、タスクキューに残り続ける。

[0064] これにより、疑似アイドルタスク１０３が、ＣＰＵコア８０（core #n）を専有し続けるので、待ち受けプロセス１０１は、タスクキュー８５に残り続けることができ、コンテキストスイッチは発生しない。

[0065] また、図４の比較例で述べたように、アイドルタスク９２がタスクキュー８５にあることで、ＣＰＵコア８０（core #n）は、idle状態に移行できていた。

本実施形態の疑似アイドルタスク１０３は、アイドルタスク９２と同じような動きをする。疑似アイドルタスク１０３は、アイドルタスク９２と同じように基本的には何もしないタスクであるが、アイドルタスク９２と同じ動作する疑似アイドルタスク１０３があることで、疑似アイドルタスク１０３は、ＣＰＵにsleep（ＮＯＰ）命令を実行させることができる。これにより、CPU cycleを削減し、該当ＣＰＵコアの消費電力を抑制することが可能になる。

[0066] 〈イベント発生時〉

kernelモードによるevent発行検知時に、待ち受けプロセス１０１は、通知プロセス９０からkernel６０（図１）のfd６３およびsocket６４を介して通知を受ける。待ち受けプロセス１０１は、タスクキュー８５に残り続けているので（図５の符号ｇｇ）、kernelモードによるevent発行検知時に通知を受

けた際、すでにタスクキュー８５にある。このため、コンテキストスイッチの発生はない（図２の符号ｃｃ）。

[0067] ［動作２］

タスクスケジューラ装置１００の動作２について、比較例（既存技術）と対比しながら説明する。

まず、比較例（既存技術）の動作について述べる。

図６は、比較例における〈待ち受け開始時〉、〈待ち受け中〉および〈イベント発生時〉の待ち受けプロセスの挙動を示す図である。図４と同一構成・プロセスには同じ符号を付している。

タスクスケジューラ装置１００は、タスクキュー８５にある実行可能状態(TASK_RUNNING)のプロセスに対してＣＰＵ実行時間を分け与えるものとする。

[0068] 〈待ち受け開始時〉

タスクスケジューラ装置１００は、タスクキュー８５に待ち受けプロセスのみ存在するため、待ち受けプロセスに実行時間を与える（図６の符号ｈ）。

[0069] 〈待ち受け中〉

タスクスケジューラ装置１００は、タスクキュー８５が空になるとアイドルタスク９２が生成され、アイドルタスク９２に実行時間を与える。このとき、コンテキストスイッチが発生する（図６の符号ｉ）。また、待ち受け中は、待ち受けプロセス９１は、タスクキュー８５から抜け（図６の矢印ｊ）、待ちキュー８６に入り待ち状態となる。

[0070] 〈イベント発生時〉

タスクスケジューラ装置１００は、イベント発生時に待ち受けプロセス９１が呼び出されると、待ち受けプロセス９１に実行時間を与える。このとき、コンテキストスイッチが発生する（図６の符号ｋ）。また、待ち受けプロセス９１は、再び、図６の中央のタスクキュー８５に戻る（図４の矢印ｌ）。

[0071] 次に、本実施形態の動作について述べる。

図７は、本実施形態における〈待ち受け開始時〉、〈待ち受け中〉および〈イベント発生時〉の待ち受けプロセスの挙動を示す図である。図５と同一構成・プロセスには同じ符号を付している。

タスクスケジューラ装置１００は、タスクキュー８５にある実行可能状態(TASK_RUNNING)のプロセスに対してＣＰＵ実行時間を分け与えるものとする。

[0072] 〈待ち受け開始時〉

タスクスケジューラ装置１００は、タスクキュー８５に待ち受けプロセス１０１のみ存在するため、待ち受けプロセス１０１に実行時間を与える（図７の符号ｈｈ）。

[0073] 〈待ち受け中〉

タスクスケジューラ装置１００は、待ち受けプロセス１０１が実行可能状態で残るので、待ち受けプロセス１０１に実行時間を与える（図７の符号ｉｉ）。すなわち、疑似アイドルタスク１０３が、ＣＰＵコア８０（core #n）（図５の〈待ち受け中〉参照）を専有し続けるので、待ち受けプロセス１０１は、タスクキュー８５に残り続けることができ、コンテキストスイッチは発生しない。

[0074] 〈イベント発生時〉

タスクスケジューラ装置１００は、イベント発生時に待ち受けプロセス１０１が呼び出されると、待ち受けプロセス１０１に実行時間を与える（図７の符号ｊｊ）。

[0075] ［実装イメージ］

タスクスケジューラ装置１００の実装イメージについて、比較例（既存技術）と対比しながら説明する。

まず、比較例（既存技術）の実装イメージについて述べる。

図８は、比較例における実装イメージを示す図である。

図８に示すように、driver/hardwareは、cpuidle driver (ACPI/intel_idle)３０６を有する。kernelは、タスクスケジューラ装置１００と、sysfs３０３と、cpuidle governor３０４と、cpuidle framework３０５とを有する。us

er spaceは、epoll_wait３０１と、idle task３０２と、を有する。

タスクスケジューラ装置１００は、待ち受け関数（epoll_wait３０１）が待ち状態（idle task３０２）に入ると、kernelの関数switch_toを使ってプロセスを切替える。epoll_wait３０１からidle task３０２に切り替えるコンテキストスイッチ（図８の符号ｃ）が実行される。

[0076] 図９は、本実施形態における実装イメージを示す図である。

図９に示すように、driver/hardwareは、cpuidle driver (ACPI/intel_idle)３０６を有する。kernelは、タスクスケジューラ装置１００と、sysfs３０３と、cpuidle governor３０４と、cpuidle framework３０５とを有する。user spaceは、epoll_wait３０１と、idle task３０２と、を有する。

タスクスケジューラ装置１００は、待ち受け関数（epoll_wait３０１）が、cpuidle framework３０５に、cpuidle_idle_callで直接発行する。このとき、待ち受け関数（epoll_wait３０１）自体が、実行可能状態のままとなる、すなわち、待ち状態とならない。かつ、ほかにプロセスがいないので、タスクスケジューラ装置１００は、あえて他のプロセスに切り替える先がいない。このことにより、タスクスケジューラ装置１００は、待ち受け関数（epoll_wait３０１）に常に時間を使わせることになる。

[0077] なお、待ち受け関数（epoll_wait３０１）のcpuidle_idle_callでの直接発行では、最大復帰時間を指定してもよい。

また、待ち受け関数（epoll_wait３０１）からkernelのsysfs３０３に対して、sysfs３０３経由で最大復帰時間を指定してもよい（図９の符号ＩＩ）。

[0078] 上記、タスクスケジューラ装置１００が、待ち受け関数（epoll_wait３０１）を、実行可能状態のまま、常に時間を使わせることが、図５の<待ち受け中>で述べた、「疑似アイドルタスク１０３は、epollを発行するが、他プロセス（スレッド）にＣＰＵタイムを譲らずに専有し続ける。」に対応する。

[0079] ［適用範囲］

［適用範囲１］

タスクスケジューラ装置１００の適用範囲について説明する。

待ち受けプロセス１０１は、通知プロセス９０やその他のプロセスと同一CPU core上でも動作することは可能である。一方、効果を最大化するために、待ち受けプロセス１０１は、通知プロセス９０やその他のプロセスと異なるCPU core上にいることが望ましい。

[0080] 図１０は、タスクスケジューラ装置１００の適用範囲１を説明する図である。図１と同じプロセスには、同一符号を付している。

[0081] 〈通知プロセスと待ち受けプロセスが異なるCPU coreで動作〉（図１０左図）

通知プロセス９０と待ち受けプロセス１０１が異なるCPU core（core #mとcore #n）で動作する場合に適用できる。

・sleep状態：core #nはsleepする。

・遅延効果：待ち受けプロセスが高速起床する。

・懸念：特になし。

[0082] 〈通知プロセスと待ち受けプロセスが同一CPU coreに同居〉（図１０中図）

通知プロセス９０と待ち受けプロセス１０１が同一CPU core（core #n）に同居する場合に適用できる。通知プロセス９０と待ち受けプロセス１０１は、ＣＰＵ時間を奪い合う（図１０中図の符号ｍｍ）。

・sleep状態：core #nは通知プロセス９０により完全にはsleepしない。

・遅延効果：通知プロセス９０があるため、遅延改善効果は小さい（通知プロセス９０とのコンテキストスイッチの存在、C-stateがもともと深くない）。

・懸念：待ち受けプロセス１０１が動作可能状態になっているため、通知プロセス９０がイベント検知・通知を送るのが遅くなる。

[0083] 〈待ち受けプロセスとその他プロセスが同一CPU coreに同居〉（図１０右図）

待ち受けプロセス１０１とその他プロセス１０４が同一CPU core（core #n）に同居する場合に適用できる。通知プロセス９０と待ち受けプロセス１０１は、ＣＰＵ時間を奪い合う（図１０右図の符号ｎｎ）。

・sleep状態：core #nはその他プロセス１０４により完全にはsleepしない。

・遅延効果：その他プロセス１０４があるため、遅延改善効果は小さい（そ

の他プロセス１０４とのコンテキストスイッチの存在、C-stateがもともと深くない）。

・懸念：待ち受けプロセス１０１が動作可能状態になっているため、その他プロセス１０４の本来の動作可能時間を奪ってしまう。

[0084] ［適用範囲２］

通知プロセス９０および待ち受けプロセス１０１は、それぞれUser space／Kernel spaceのどこにあっても適用される。

図１１は、タスクスケジューラ装置１００の適用範囲２を説明する図である。図１および図１０と同じプロセスには、同一符号を付している。

図１１左上図は、通知プロセス：User spaceで、待ち受けプロセス：User spaceの場合の例である。

図１１右上図は、通知プロセス：User spaceで、待ち受けプロセス：Kernel spaceの場合の例である。

図１１左下図は、通知プロセス：Kernel spaceで、待ち受けプロセス：User spaceの場合の例である。

図１１右下図は、通知プロセス：Kernel spaceで、待ち受けプロセス：Kernel spaceの場合の例である。

以上、通知プロセス９０および待ち受けプロセス１０１は、User space／Kernel spaceのいずれにも配置可能である。

[0085] ［適用範囲３］

通知に用いるファイルディスクリプタとしては例としてeventfd, singalfd, timerfdなどが挙げられる。

リソースを監視する待ち受け関数としては例としてread(), select(), poll(), epoll()などが挙げられる。いずれの関数もブロッキングあるいはノンブロッキングの動作をするが、ブロッキング時にはTASK_INTERRUPTIBLE/TASK_UNINTERRUPTIBLEのイベント待ちに落ちるので、本発明を適用可能である。

[0086] ［制御シーケンス］

タスクスケジューラ装置１００の制御シーケンスについて説明する。

［制御シーケンス１］

図１２は、イベント発生前の疑似アイドル機能が必要なケースにおけるシーケンスである。

待ち受けプロセス１０１は、疑似アイドル判断部１３０に待ち受け開始を要求する（Ｓ１０１）。

疑似アイドル判断部１３０は、復帰時間決定部１４０に復帰時間決定を要求する（Ｓ１０２）。

復帰時間決定部１４０は、復帰時間を決定する（Ｓ１０３）。ステップＳ１０３における復帰時間決定の詳細フローについては、図１６により後記する。

[0087] 復帰時間決定部１４０は、復帰時間設定部１５０に復帰時間を通知する（Ｓ１０４）。

復帰時間設定部１５０は、C-state制御部６２に対して復帰時間設定を行う（Ｓ１０５１）。

C-state制御部６２は、復帰時間設定部１５０に復帰時間設定の応答を行う（Ｓ１０６）。

[0088] 復帰時間設定部１５０は、C-state制御部６２からの復帰時間設定応答を受けて、復帰時間決定部１４０に復帰時間の応答を行う（Ｓ１０７）。

復帰時間決定部１４０は、疑似アイドル判断部１３０に復帰時間設定部１５０からの復帰時間決定応答を通知する（Ｓ１０８）。

[0089] 疑似アイドル判断部１３０は、疑似アイドル機能要否判断を行う（Ｓ１０９）。ステップＳ１０９における疑似アイドル機能要否判断の詳細フローについては、図１５により後記する。

疑似アイドル判断部１３０は、待ち受けプロセス１０１の待受部１１０に疑似アイドル機能要求を行う（Ｓ１１０）。

[0090] 待受部１１０は、疑似アイドル判断部１３０からの疑似アイドル機能要求を受けて、待ち受け設定および実行可能状態設定を行う（Ｓ１１１）。

待受部１１０は、疑似アイドル判断部１３０に疑似アイドル機能要求に対

する疑似アイドル機能応答を通知する（Ｓ１１２）。

待受部１１０は、疑似アイドル部１２０に疑似アイドル開始を要求する（Ｓ１１３）。

疑似アイドル部１２０は、C-state制御部６２にC-state制御を要求する（Ｓ１１４）。

[0091] ［制御シーケンス２］

図１３は、イベント発生前の疑似アイドル機能が不要なケースにおけるシーケンスである。図１２の制御シーケンスと同じ処理には同一番号を付している。

待ち受けプロセス１０１は、疑似アイドル判断部１３０に待ち受け開始を要求する（Ｓ１０１）。

疑似アイドル判断部１３０は、復帰時間決定部１４０に復帰時間決定を要求する（Ｓ１０２）。

復帰時間決定部１４０は、復帰時間を決定する（Ｓ１０３）。ステップＳ１０３における復帰時間決定の詳細フローについては、図１５により後記する。

[0092] 復帰時間決定部１４０は、復帰時間設定部１５０に復帰時間を通知する（Ｓ１０４）。

復帰時間設定部１５０は、C-state制御部６２に対して復帰時間設定を行う（Ｓ１０５１）。

C-state制御部６２は、復帰時間設定部１５０に復帰時間設定の応答を行う（Ｓ１０６）。

[0093] 復帰時間設定部１５０は、C-state制御部６２からの復帰時間設定応答を受けて、復帰時間決定部１４０に復帰時間の応答を行う（Ｓ１０７）。

復帰時間決定部１４０は、疑似アイドル判断部１３０に復帰時間設定部１５０からの復帰時間決定応答を通知する（Ｓ１０８）。

[0094] 疑似アイドル判断部１３０は、疑似アイドル機能要否判断を行う（Ｓ１０９）。ステップＳ１０９における疑似アイドル機能要否判断の詳細フローに

ついては、図１６により後記する。

疑似アイドル判断部１３０は、待ち受けプロセス１０１の待受部１１０にアイドル機能要求を行う（Ｓ２０１）。

[0095] 待受部１１０は、疑似アイドル判断部１３０からのアイドル機能要求を受けて、待ち受け設定および待ち状態設定を行う（Ｓ２０２）。

待受部１１０は、疑似アイドル判断部１３０にアイドル機能要求に対するアイドル機能応答を通知する（Ｓ２０３）。

待受部１１０は、スケジューラ６１にアイドル開始を要求する（Ｓ２０４）。

[0096] スケジューラ６１は、アイドル部５１にアイドル開始要求を行う（Ｓ２０５）。

アイドル部５１は、C-state制御部６２にC-state制御を要求する（Ｓ１１４）。

[0097] ［制御シーケンス３］

図１４は、イベント発生後の疑似アイドル機能が必要なケースにおけるシーケンスである。

通知プロセス９０は、待受部１１０にファイルディスクリプタ等による通知を行う（Ｓ３０１）。

待受部１１０は、疑似アイドル部１２０に疑似アイドル終了を要求する（Ｓ３０２）。

疑似アイドル部１２０は、C-state制御部６２にC-state制御を要求する（Ｓ３０３）。

[0098] 待受部１１０は、疑似アイドル判断部１３０に疑似アイドル終了を要求する（Ｓ３０４）。

疑似アイドル判断部１３０は、待ち受けプロセス１０１に待ち受け終了を通知する（Ｓ３０５）。

待ち受けプロセス１０１は、処理を再開する（Ｓ３０６）。

[0099] 疑似アイドル判断部１３０は、C-state制御部６２に復帰時間設定を行う（

Ｓ３０７）。

C-state制御部６２は、疑似アイドル判断部１３０に復帰時間設定応答を行う（Ｓ３０８）。

[0100] ［フローチャート］

タスクスケジューラ装置１００のフローチャートについて説明する。

［フローチャート１］

図１５は、疑似アイドル判断部１３０における疑似アイドル機能を実行するフローチャートである。図１２および図１３のステップＳ１０９の「疑似アイドル要否判断」の詳細フローである。

ステップＳ１１で疑似アイドル判断部１３０は、待ち受けプロセス１０１と同一コアに、待ち受けプロセス１０１がsleep中に稼働する他のプロセス１０４は、０か否かを判別する（図１０右図参照）。

[0101] sleep中に稼働する他のプロセス１０４が０の場合（Ｓ１１：Ｙｅｓ）、アイドルタスク発生と判断してステップＳ１２で疑似アイドル判断部１３０は、疑似アイドル機能を実行する。

[0102] ステップＳ１３で疑似アイドル判断部１３０は、ハードウェア設定（ＢＩＯＳ）によりＣＰＵのsleep状態制御が有効になっているか否かを判別する。

[0103] ＣＰＵのsleep状態制御が有効になっている場合（Ｓ１３：Ｙｅｓ）、C-stateありと判断してステップＳ１４で疑似アイドル判断部１３０は、ＯＳのガバナーによる復帰時間が、許容される復帰時間より大きいか（ＯＳのガバナーによる復帰時間＞許容される復帰時間か）否かを判別する。

[0104] ＯＳのガバナーによる復帰時間＞許容される復帰時間の場合（Ｓ１４：Ｙｅｓ）、最大復帰時間制御が必要と判断してステップＳ１５で疑似アイドル判断部１３０は、最大復帰時間に許容される復帰時間を設定（設定最大復帰時間＝許容される復帰時間）して本フローの処理を終了する。

[0105] 上記ステップＳ１１でsleep中に稼働する他のプロセス１０４が０でない場合（Ｓ１１：Ｎｏ）、でアイドルタスクなしと判断してステップＳ１６で疑似アイドル判断部１３０は、疑似アイドル機能を実行せずにステップＳ１７

に進む。

[0106] 上記ステップＳ１３でＣＰＵのsleep状態制御が有効になっていない場合、または、上記ステップＳ１４でＯＳのガバナーによる復帰時間が、許容される復帰時間以下の場合（ＯＳのガバナーによる復帰時間≦許容される復帰時間の場合）、ステップＳ１７で疑似アイドル判断部１３０は、最大復帰時間指定をせずに本フローの処理を終了する。

[0107] ［フローチャート２］

図１６は、復帰時間決定部１４０における復帰時間決定のフローチャートである。図１２および図１３のステップＳ１０３の「復帰時間決定」の詳細フローである。

本フローは、実測したC-state復帰時間を用いて最大復帰時間を定める方法である。

ステップＳ２１で復帰時間決定部１４０は、タイムスロット#nにおいて、実測したC-state＝最大復帰時間としたとき、この実測したC-stateが、最大復帰時間が許容するC-stateと等しいか（C-state＝最大復帰時間が許容するC-stateか）否かを判別する。

[0108] C-state＝最大復帰時間が許容するC-stateの場合（Ｓ２１：Ｙｅｓ）、ステップＳ２２で復帰時間決定部１４０は、（最大復帰時間で許容するC-state復帰時間＋１段深いC-stateにおける復帰時間の増分）が、許容遅延時間より小さいか否かを判別する。

[0109] （最大復帰時間で許容するC-state復帰時間＋１段深いC-stateにおける復帰時間の増分）＜許容遅延時間の場合（Ｓ２２：Ｙｅｓ）、ステップＳ２３で復帰時間決定部１４０は、タイムスロット#n+1における最大復帰時間＋１段深いC-stateにおける復帰時間の増分（許容C-state＋１）を設定して本フローの処理を終了する。

[0110] 上記ステップＳ２１でC-state＝最大復帰時間が許容するC-stateでない場合（Ｓ２１：Ｎｏ）、復帰時間短縮と判断してステップＳ２４で復帰時間決定部１４０は、タイムスロット#n+1における最大復帰時間－１段浅いC-state

における復帰時間の差分（許容C-state－１）を設定して本フローの処理を終了する。

[0111] 上記ステップＳ２２で（最大復帰時間で許容するC-state復帰時間＋１段深いC-stateにおける復帰時間の増分）≧許容遅延時間の場合（Ｓ２２：Ｎｏ）、ステップＳ２５で復帰時間決定部１４０は、最大復帰時間修正なしとして本フローの処理を終了する。

[0112] 上記フローでは、タイムスロットごとに上位のC-stateに移行しても遅延追加が許容されるか、または下位のC-stateに移行できるかを判断している。

ここで、過去の履歴で周期性がある場合、いきなり目標のC-stateに遷移してもよい。また、実測したC-stateがＣ０の場合busyのため、段階を踏まずいきなりＣ０に戻してもよい。

[0113] 図１７は、図１６の最大復帰時間決定処理を実行した場合の動作例を示す図である。図１７の縦軸は、許容遅延を示し、横軸は時間を示す。また、図１７中の破線は、最大復帰時間で許容するC-state復帰時間を示し、図１７中の実線は、実測したC-state復帰時間を示す。図１７に示すように、許容するC-state復帰時間を設定すると、実測C-state復帰時間がそれに追従する。図１７では、タイムスロットｔ３において目標C-stateのＣ３は、許容遅延を超えているので許容されなかった例を示している（図１７の符号ｏｏ）。

[0114] ［ハードウェア構成］

上記実施形態に係るタスクスケジューラ装置１００（図１）は、例えば図１８に示すような構成のコンピュータ９００によって実現される。

図１８は、タスクスケジューラ装置１００（図１）の機能を実現するコンピュータ９００の一例を示すハードウェア構成図である。

コンピュータ９００は、ＣＰＵ９０１、ＲＯＭ９０２、ＲＡＭ９０３、ＨＤＤ９０４、通信インターフェイス（Ｉ／Ｆ：Interface）９０６、入出力インターフェイス（Ｉ／Ｆ）９０５、およびメディアインターフェイス（Ｉ／Ｆ）９０７を有する。

[0115] ＣＰＵ９０１は、ＲＯＭ９０２またはＨＤＤ９０４に格納されたプログラ

ムに基づいて動作し、タスクスケジューラ装置１００（図１）の各部の制御を行う。ＲＯＭ９０２は、コンピュータ９００の起動時にＣＰＵ９０１によって実行されるブートプログラムや、コンピュータ９００のハードウェアに依存するプログラム等を格納する。

[0116] ＣＰＵ９０１は、入出力Ｉ／Ｆ９０５を介して、マウスやキーボード等の入力装置９１０、および、ディスプレイ等の出力装置９１１を制御する。ＣＰＵ９０１は、入出力Ｉ／Ｆ９０５を介して、入力装置９１０からデータを取得するともに、生成したデータを出力装置９１１へ出力する。なお、プロセッサとしてＣＰＵ９０１とともに、ＧＰＵ（Graphics Processing Unit）等を用いてもよい。

[0117] ＨＤＤ９０４は、ＣＰＵ９０１により実行されるプログラムおよび当該プログラムによって使用されるデータ等を記憶する。通信Ｉ／Ｆ９０６は、通信網（例えば、ＮＷ（Network）９２２）を介して他の装置からデータを受信してＣＰＵ９０１へ出力し、また、ＣＰＵ９０１が生成したデータを、通信網を介して他の装置へ送信する。

[0118] メディアＩ／Ｆ９０７は、記録媒体９１２に格納されたプログラムまたはデータを読み取り、ＲＡＭ９０３を介してＣＰＵ９０１へ出力する。ＣＰＵ９０１は、目的の処理に係るプログラムを、メディアＩ／Ｆ９０７を介して記録媒体９１２からＲＡＭ９０３上にロードし、ロードしたプログラムを実行する。記録媒体９１２は、ＤＶＤ（Digital Versatile Disc）、ＰＤ（Phase change rewritable Disk）等の光学記録媒体、ＭＯ（Magneto Optical disk）等の光磁気記録媒体、磁気記録媒体、導体メモリテープ媒体又は半導体メモリ等である。

[0119] 例えば、コンピュータ９００が本実施形態に係る一装置として構成されるタスクスケジューラ装置１００（図１）として機能する場合、コンピュータ９００のＣＰＵ９０１は、ＲＡＭ９０３上にロードされたプログラムを実行することによりタスクスケジューラ装置１００の機能を実現する。また、ＨＤＤ９０４には、ＲＡＭ９０３内のデータが記憶される。ＣＰＵ９０１は、

目的の処理に係るプログラムを記録媒体９１２から読み取って実行する。この他、ＣＰＵ９０１は、他の装置から通信網（ＮＷ９２２）を介して目的の処理に係るプログラムを読み込んでもよい。

[0120] ［効果］

以上説明したように、複数のコアからなるプロセッサを有し、当該プロセッサ上の時間を用いて所定の命令を実行するプロセスが動作する計算機システム１０００（図１）であって、プロセスは、通知があるまで待ち状態となり、通知がある場合に起床して動作を再開する待ち受けプロセス１０１（図５、図７）と、イベント発生時に待ち受けプロセスを起床させるために通知を送信する通知プロセス９０と、を有し、待ち受けプロセス１０１は、待ち受け状態に入っているときに、プロセッサを専有し続ける疑似アイドルタスク１０３（図５、図７）を備える。

[0121] このようにすることにより、計算機システム１０００は、プロセッサ上にプロセスがないとき発生し何も実行しないアイドルタスク９２（図４、図６）と同じ振る舞いを、待ち受けプロセス１０１にさせることができる。待ち受け起床の際、アイドルタスク９２のスケジュールが行われず、従ってコンテキストスイッチは発生せずに、待ち受けプロセス処理再開となる。すなわち、疑似アイドルタスク１０３は、待ち受けプロセス１０１にアイドルタスク９２と同じ振る舞いをさせ、アイドルタスク９２からのコンテキストスイッチを排除する。このため、イベント発生後に待ち受けプロセスが起床するまでにかかる時間を低減することができる。

[0122] 計算機システム１０００（図１）において、疑似アイドルタスク１０３（図５、図７）は、プロセッサにsleep命令を実行させることを特徴とする。

[0123] このようにすることにより、疑似アイドルタスク１０３は、ＣＰＵにsleep（ＮＯＰ）命令を実行させことができる。すなわち、疑似アイドルタスク１０３は、何も実行しないsleep状態を維持し、アイドル部５１（図１）の代わりにＣＰＵの稼働状態を制御するc-state制御部６２（図１）を呼び出して実行する。これにより、CPU cycleを削減し、該当ＣＰＵコアの消費電力を抑制

することが可能になる。

[0124] 複数のコアからなるプロセッサを有し、当該プロセッサ上の時間を用いて所定の命令を実行するプロセスが動作する計算機システム１０００（図１）において、プロセスのスケジュールを行うタスクスケジューラ装置１００（図１）であって、通知があるまで待ち状態となり、通知がある場合に起床して動作を再開する待ち受けプロセス１０１（図５、図７）と、待ち受け状態に入っているときに、待ち受けプロセス１０１が、プロセッサを専有し続ける疑似アイドルタスク１０３（図５、図７）と、を備える。

[0125] このようにすることにより、タスクスケジューラ装置１００は、コンテキストスイッチを発生せずに、待ち受けプロセス処理再開となるため、イベント発生後に待ち受けプロセスが起床するまでにかかる時間を最小化することができる。

[0126] タスクスケジューラ装置１００（図１）において、疑似アイドルタスク１０３（図５、図７）は、プロセッサにsleep命令を実行させることを特徴とする。

[0127] このようにすることにより、疑似アイドルタスク１０３は、ＣＰＵにsleep（ＮＯＰ）命令を実行させことができる。これにより、CPU cycleを削減し、該当ＣＰＵコアの消費電力を抑制することが可能になる。

[0128] タスクスケジューラ装置１００（図１）において、同一プロセッサ上のプロセス生起状態、および／または、遅延時間の要件をもとに、疑似アイドルタスク１０３を実行するかを判断する疑似アイドル判断部１３０を備えることを特徴とする。

[0129] このようにすることにより、プロセッサ上のプロセス生起状態や遅延時間の要件をもとに、疑似アイドルタスク１０３を適切な条件で実行することができる。疑似アイドルタスク１０３の効果があり、かつ他のプロセスに影響を及ぼさない条件でのみ実行されるので、「プロセッサ占有によるコンテキストスイッチ排除」と「sleep命令を実行させる」効果の実効を図ることができる。

[0130] タスクスケジューラ装置１００（図１）において、疑似アイドルタスク１０３（図５、図７）がタスク発生後に復帰するまでの復帰時間を設定する復帰時間設定部（復帰時間決定部１４０および復帰時間設定部１５０）を備えることを特徴とする。

[0131] このようにすることにより、タスクスケジューラ装置１００は、待ち受けプロセス（待ち受け関数）起床までの遅延時間を短くすることが可能になる。

[0132] タスクスケジューラ装置１００（図１）において、復帰時間制御部（復帰時間決定部１４０および復帰時間設定部１５０）は、決められた復帰時間を最大復帰時間で上書きし、最大復帰時間を短くとることで深いsleep状態へ遷移させないことを特徴とする。

[0133] このようにすることにより、疑似アイドルタスクの最大の復帰時間を指定して、深いsleep状態に落ちないように制御することができる。タスク割り当てから演算開始までの遅延時間を抑制し、リアルタイム性高く演算することを可能とする。

すなわち、タスクスケジューラ装置１００は、待ち受けプロセス（待ち受け関数）起床までの遅延時間を短くすることが可能になる。このため、<要件１：通知時間低減>を満たすことができる。また、タスクスケジューラ装置１００は、最大復帰時間を長くとることで、深いsleep状態へ遷移させ、ＨＷにおいて電力を削減することが可能になり、<要件２：省電力性>を満たすことができる。

[0134] なお、上記実施形態において説明した各処理のうち、自動的に行われるものとして説明した処理の全部又は一部を手動的に行うこともでき、あるいは、手動的に行われるものとして説明した処理の全部又は一部を公知の方法で自動的に行うこともできる。この他、上述文書中や図面中に示した処理手順、制御手順、具体的名称、各種のデータやパラメータを含む情報については、特記する場合を除いて任意に変更することができる。

また、図示した各装置の各構成要素は機能概念的なものであり、必ずしも

物理的に図示の如く構成されていることを要しない。すなわち、各装置の分散・統合の具体的形態は図示のものに限られず、その全部又は一部を、各種の負荷や使用状況などに応じて、任意の単位で機能的又は物理的に分散・統合して構成することができる。

[0135] また、上記の各構成、機能、処理部、処理手段等は、それらの一部又は全部を、例えば集積回路で設計する等によりハードウェアで実現してもよい。また、上記の各構成、機能等は、プロセッサがそれぞれの機能を実現するプログラムを解釈し、実行するためのソフトウェアで実現してもよい。各機能を実現するプログラム、テーブル、ファイル等の情報は、メモリや、ハードディスク、ＳＳＤ（Solid State Drive）等の記録装置、または、ＩＣ（Integrated Circuit）カード、ＳＤ（Secure Digital）カード、光ディスク等の記録媒体に保持することができる。

符号の説明

[0136] ５０　アイドルプロセス（アイドルタスク）

５１　アイドル部

６０　kernel

６１　スケジューラ

６２　C-state制御部

９０　通知プロセス

９２　アイドルタスク

１００　タスクスケジューラ装置

１０１　待ち受けプロセス

１０３　疑似アイドルタスク

１１０　待受部

１２０　疑似アイドル部

１３０　疑似アイドル判断部

１４０　復帰時間決定部（復帰時間制御部）

１５０　復帰時間設定部（復帰時間制御部）

１０００　計算機システム

CPUcore #m, CPUcore #n, …　ＣＰＵコア

請求の範囲

[請求項1] 複数のコアからなるプロセッサを有し、当該プロセッサ上の時間を用いて所定の命令を実行するプロセスが動作する計算機システムであって、

前記プロセスは、通知があるまで待ち状態となり、通知がある場合に起床して動作を再開する待ち受けプロセスと、イベント発生時に待ち受けプロセスを起床させるために通知を送信する通知プロセスと、を有し、

前記待ち受けプロセスは、待ち受け状態に入っているときに、前記プロセッサを専有し続ける疑似アイドルタスクを有する

ことを特徴とする計算機システム。

[請求項2] 前記疑似アイドルタスクは、前記プロセッサにsleep命令を実行させる

ことを特徴とする請求項１記載の計算機システム。

[請求項3] 複数のコアからなるプロセッサを有し、当該プロセッサ上の時間を用いて所定の命令を実行するプロセスが動作する計算機システムにおいて、前記プロセスのスケジュールを行うタスクスケジューラ装置であって、

通知があるまで待ち状態となり、通知がある場合に起床して動作を再開する待ち受けプロセスと、

待ち受け状態に入っているときに、前記待ち受けプロセスが、前記プロセッサを専有し続ける疑似アイドルタスクと、を備える

ことを特徴とするタスクスケジューラ装置。

[請求項4] 前記疑似アイドルタスクは、前記プロセッサにsleep命令を実行させる

ことを特徴とする請求項３記載のタスクスケジューラ装置。

[請求項5] 同一プロセッサ上のプロセス生起状態、および／または、遅延時間の要件をもとに、前記疑似アイドルタスクを実行するかを判断する疑

似アイドル判断部を備える

ことを特徴とする請求項３記載のタスクスケジューラ装置。

[請求項6] 前記疑似アイドルタスクがタスク発生後に復帰するまでの復帰時間を設定する復帰時間制御部を備える

ことを特徴とする請求項３に記載のタスクスケジューラ装置。

[請求項7] 複数のコアからなるプロセッサを有し、当該プロセッサ上の時間を用いて所定の命令を実行するプロセスが動作する計算機システムにおいて、前記プロセスのスケジュールを行うタスクスケジューラ装置の待ちプロセス起床方法であって、

前記タスクスケジューラ装置は、

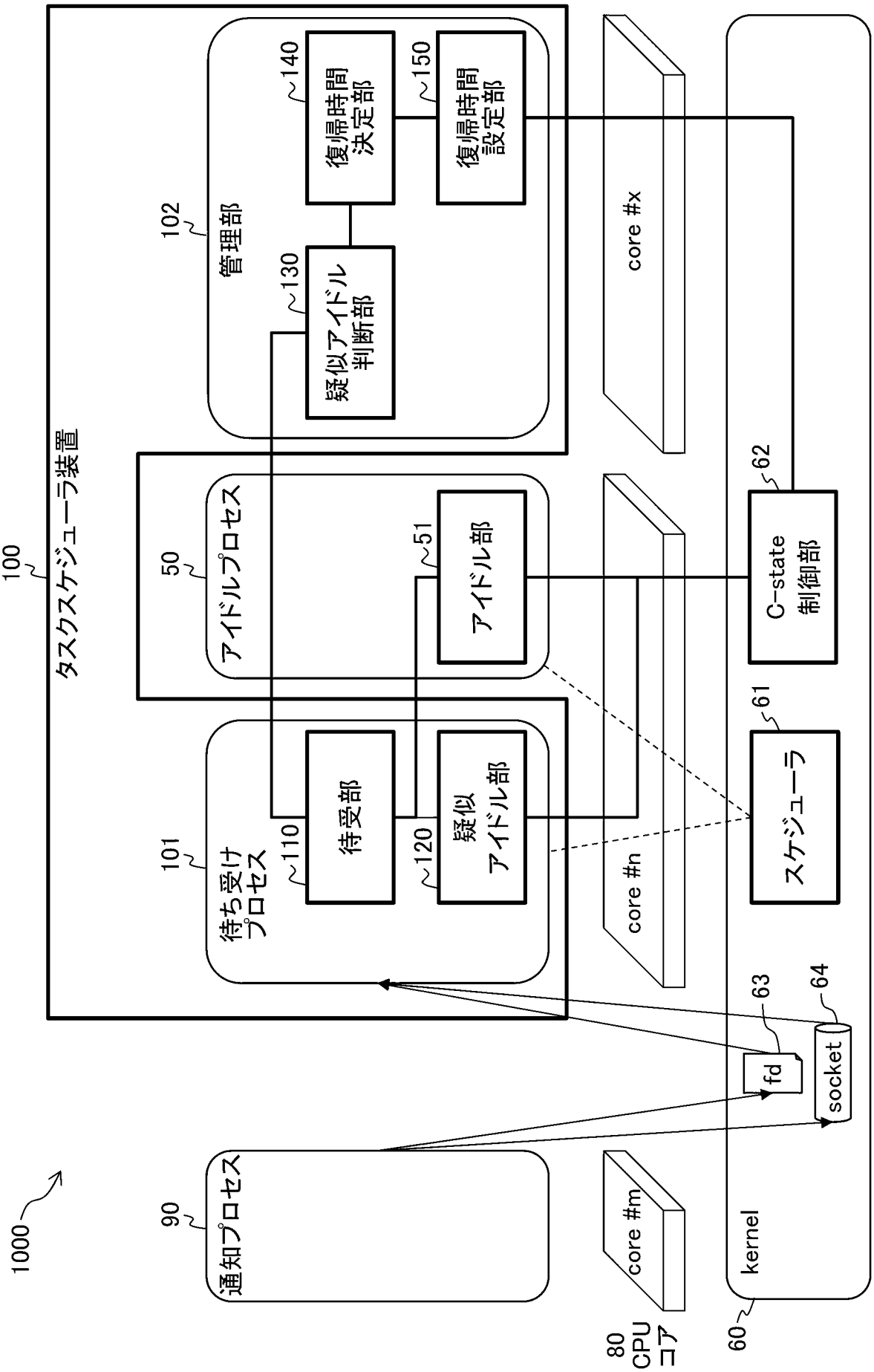
通知があるまで待ち状態となり、通知がある場合に起床して動作を再開する待ち受けプロセスと、

待ち受け状態に入っているときに、前記待ち受けプロセスが、前記プロセッサを専有し続ける疑似アイドルタスクと、を実行する

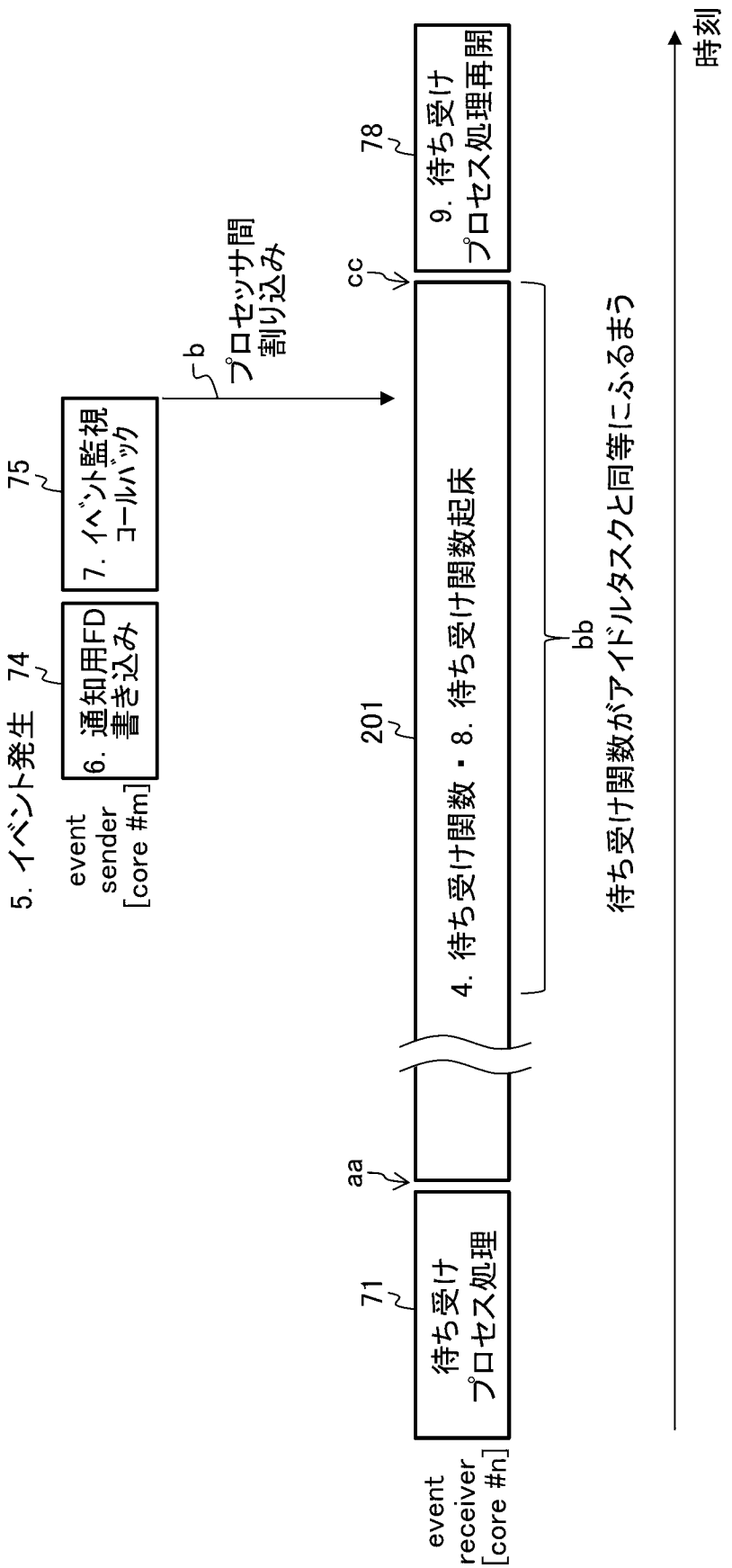
ことを特徴とする待ちプロセス起床方法。

[請求項8] コンピュータを、請求項３ないし６のいずれか一項に記載のタスクスケジューラ装置として機能させるためのプログラム。

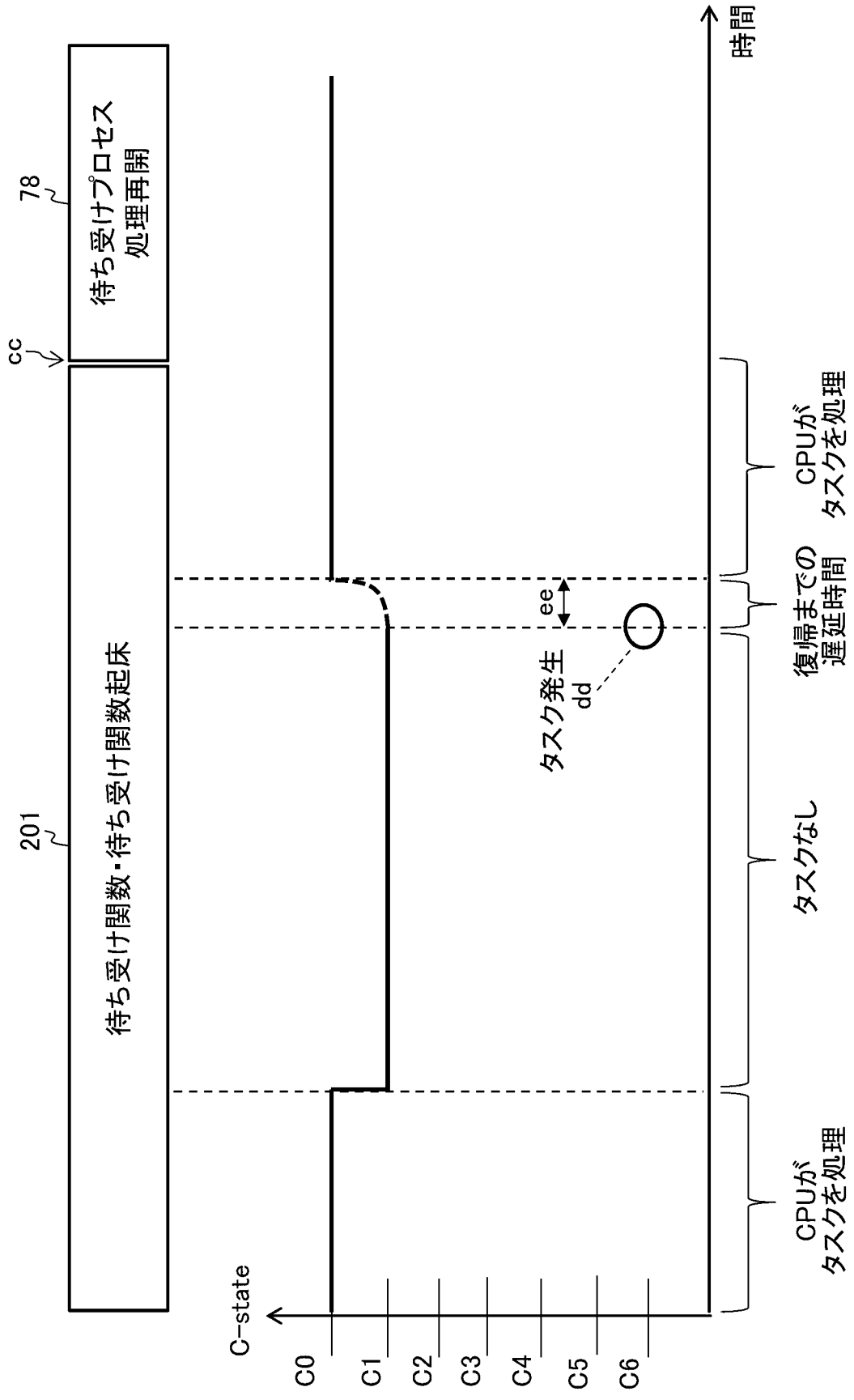
[図1]


1000
100
タスクスケジューラ装置
102
管理部
140
復帰時間決定部
150
復帰時間設定部
130
疑似アイドル判断部
50
アイドルプロセス
51
アイドル部
101
待ち受けプロセス
110
待受部
120
疑似アイドル部
90
通知プロセス
62
C-state制御部
61
スケジューラ
63
fd
64
socket
core #x
core #n
core #m
80
CPUコア
60
kernel

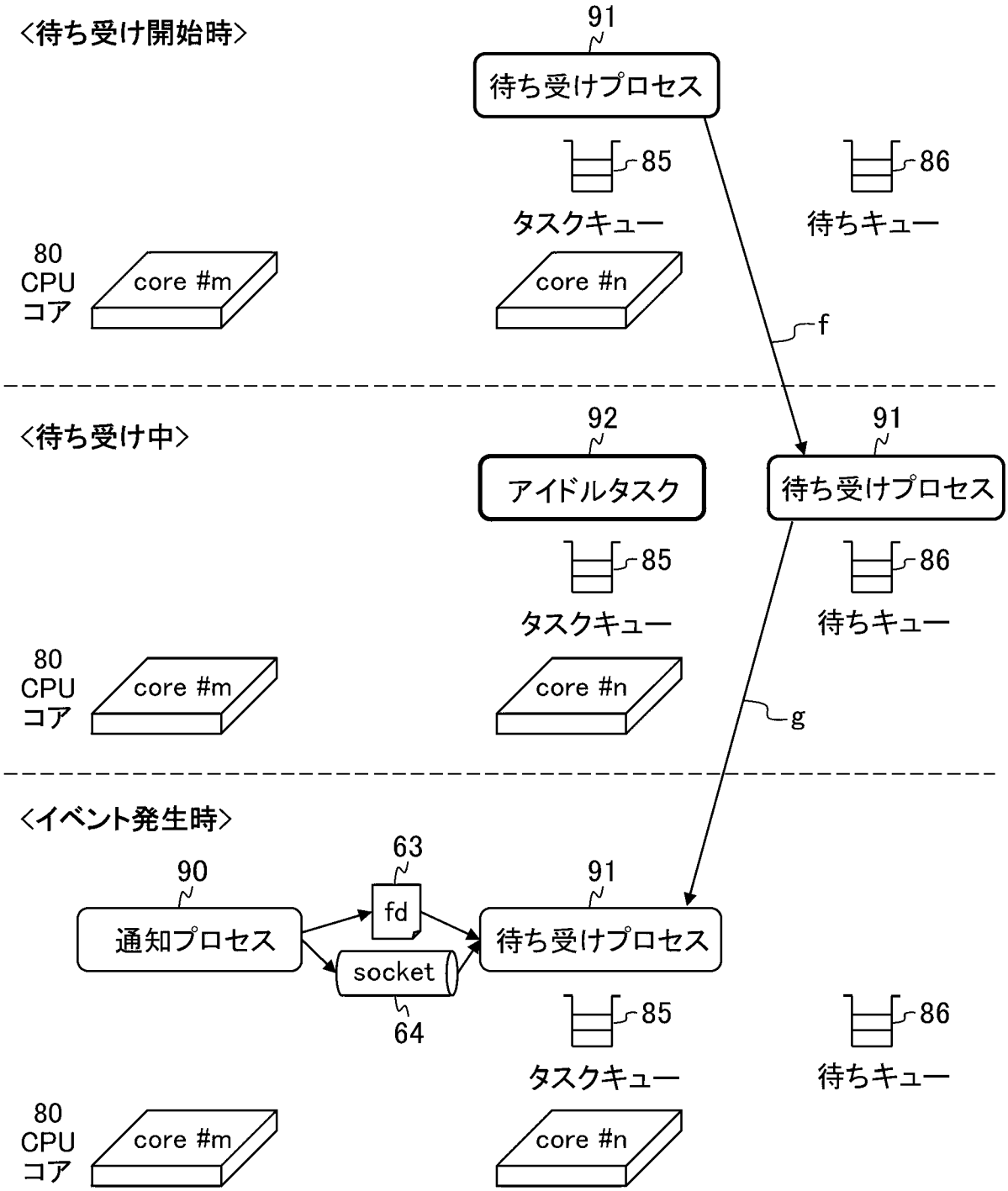
[図2]


5. イベント発生
74
6. 通知用FD
書き込み
event
sender
[core #m]
75
7. イベント監視
コールバック
b
プロセッサ間
割り込み
78
9. 待ち受け
プロセス処理再開
cc
201
4. 待ち受け関数 ・ 8. 待ち受け関数起床
bb
待ち受け関数がアイドルタスクと同等にふるまう
aa
71
待ち受け
プロセス処理
event
receiver
[core #n]
時刻

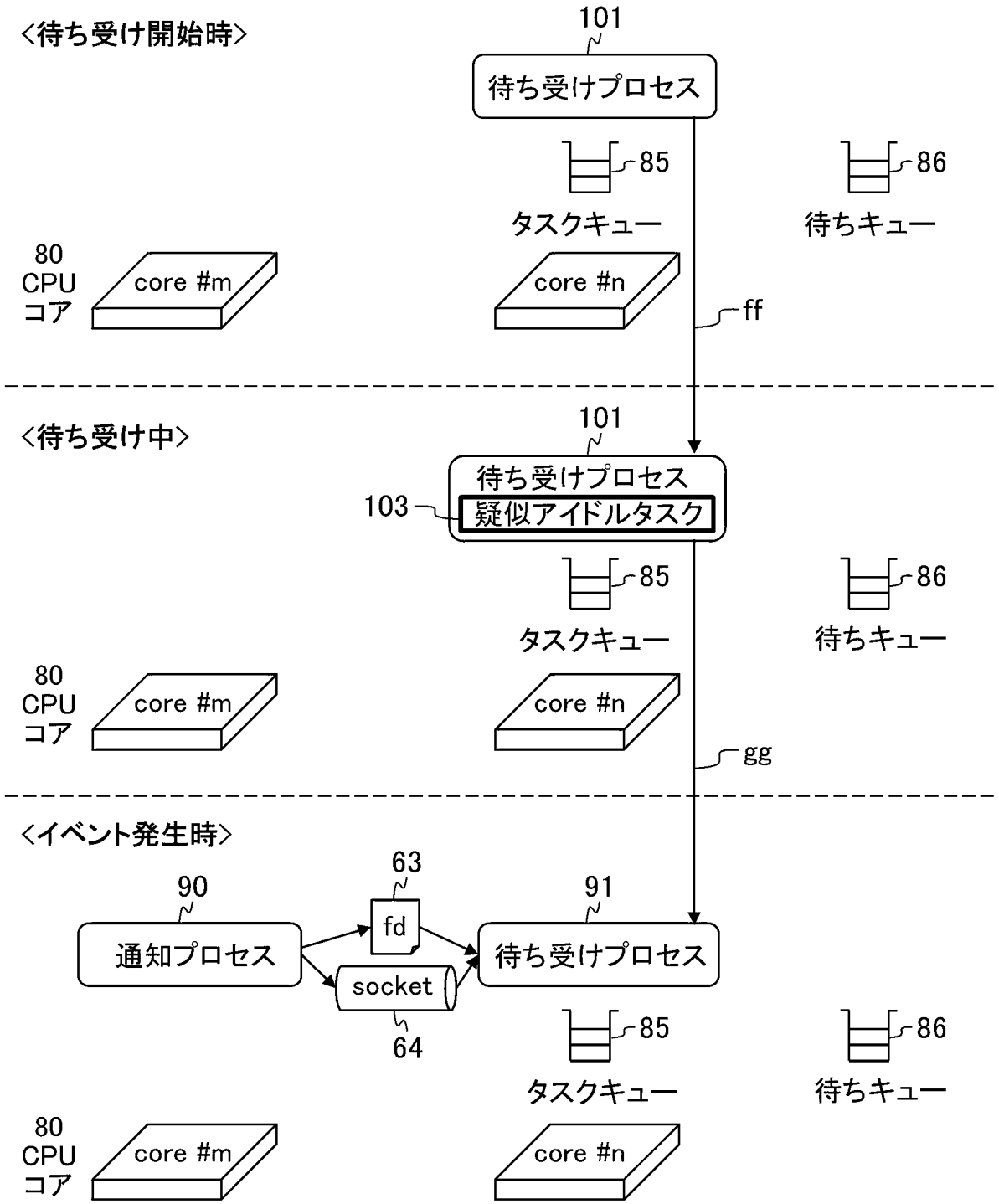
[図3]


201
待ち受け関数・待ち受け関数起床
cc
78
待ち受けプロセス
処理再開
時間
C-state
C0
C1
C2
C3
C4
C5
C6
CPUが
タスクを処理
タスクなし
タスク発生
dd
ee
復帰までの
遅延時間
CPUが
タスクを処理

[図4]

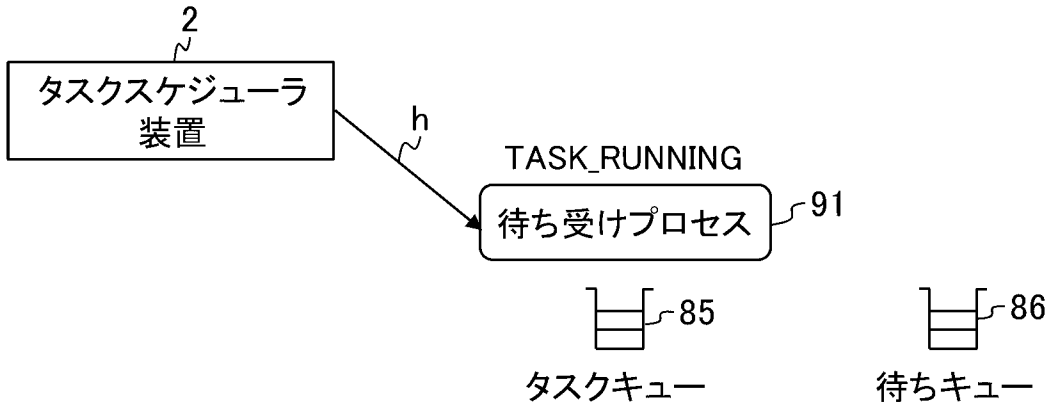

<待ち受け開始時>
91
待ち受けプロセス
85
タスクキュー
86
待ちキュー
80
CPU
コア
core #m
core #n
f
<待ち受け中>
92
アイドルタスク
91
待ち受けプロセス
85
タスクキュー
86
待ちキュー
80
CPU
コア
core #m
core #n
g
<イベント発生時>
63
90
fd
91
通知プロセス
socket
待ち受けプロセス
64
85
タスクキュー
86
待ちキュー
80
CPU
コア
core #m
core #n

[図5]

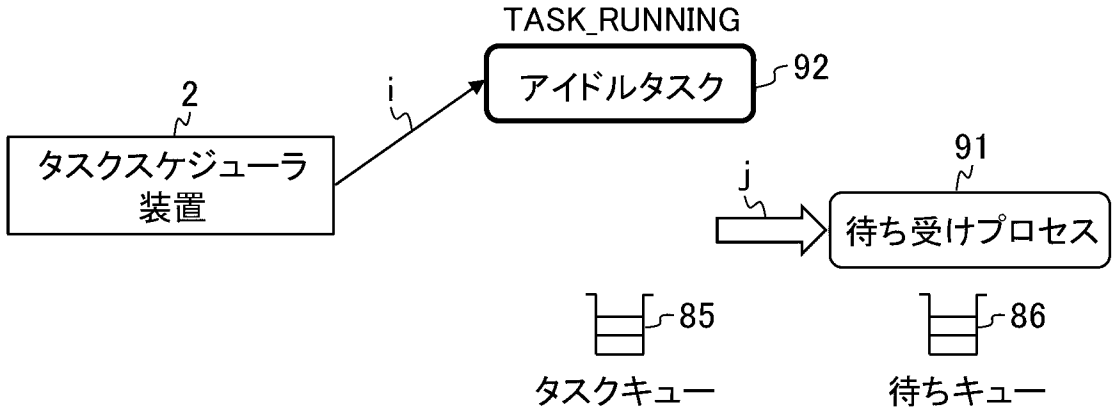

<待ち受け開始時>
101
待ち受けプロセス
85
タスクキュー
86
待ちキュー
80
CPU
コア
core #m
core #n
ff
<待ち受け中>
101
待ち受けプロセス
103
疑似アイドルタスク
85
タスクキュー
86
待ちキュー
80
CPU
コア
core #m
core #n
gg
<イベント発生時>
63
90
91
fd
通知プロセス
待ち受けプロセス
socket
64
85
タスクキュー
86
待ちキュー
80
CPU
コア
core #m
core #n

[図6]

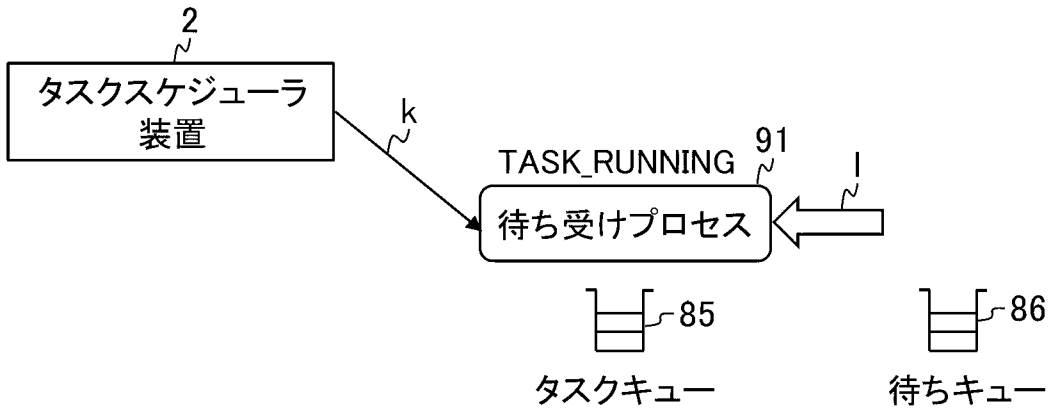
<待ち受け開始時>



<待ち受け中>

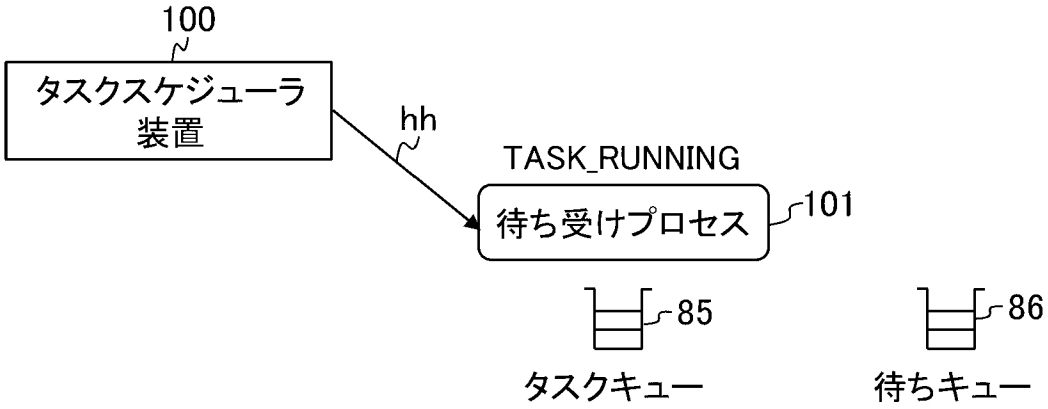


<イベント発生時>

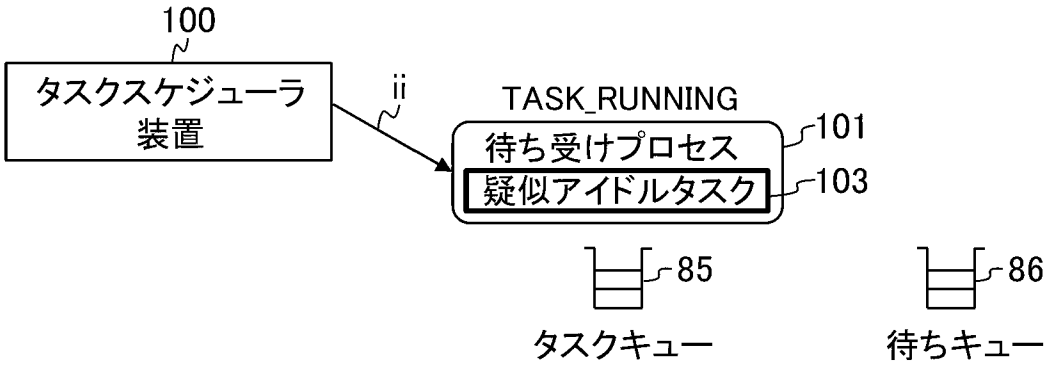


[図7]

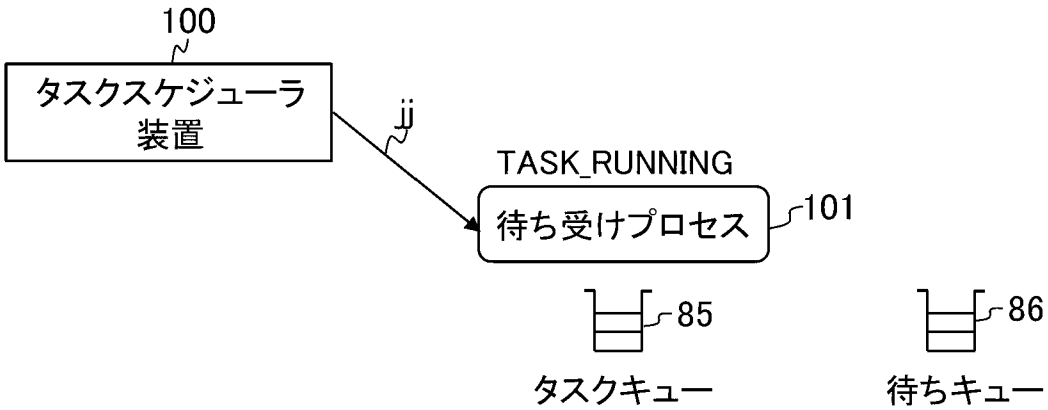
<待ち受け開始時>



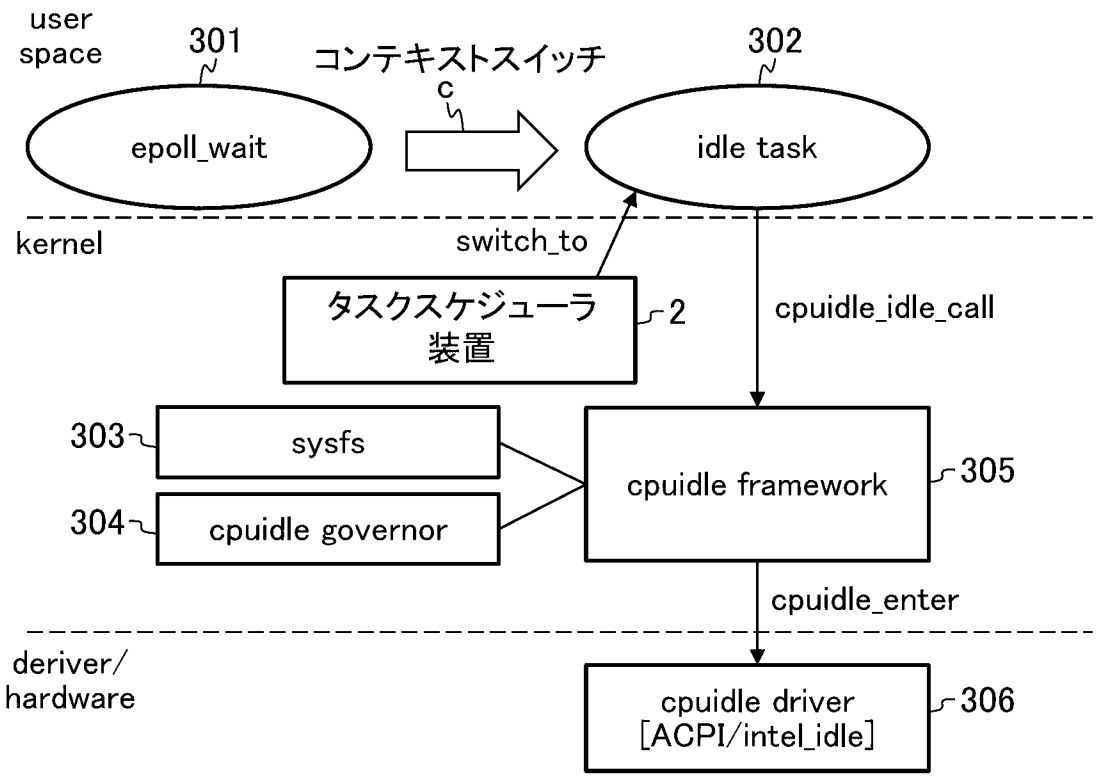
<待ち受け中>



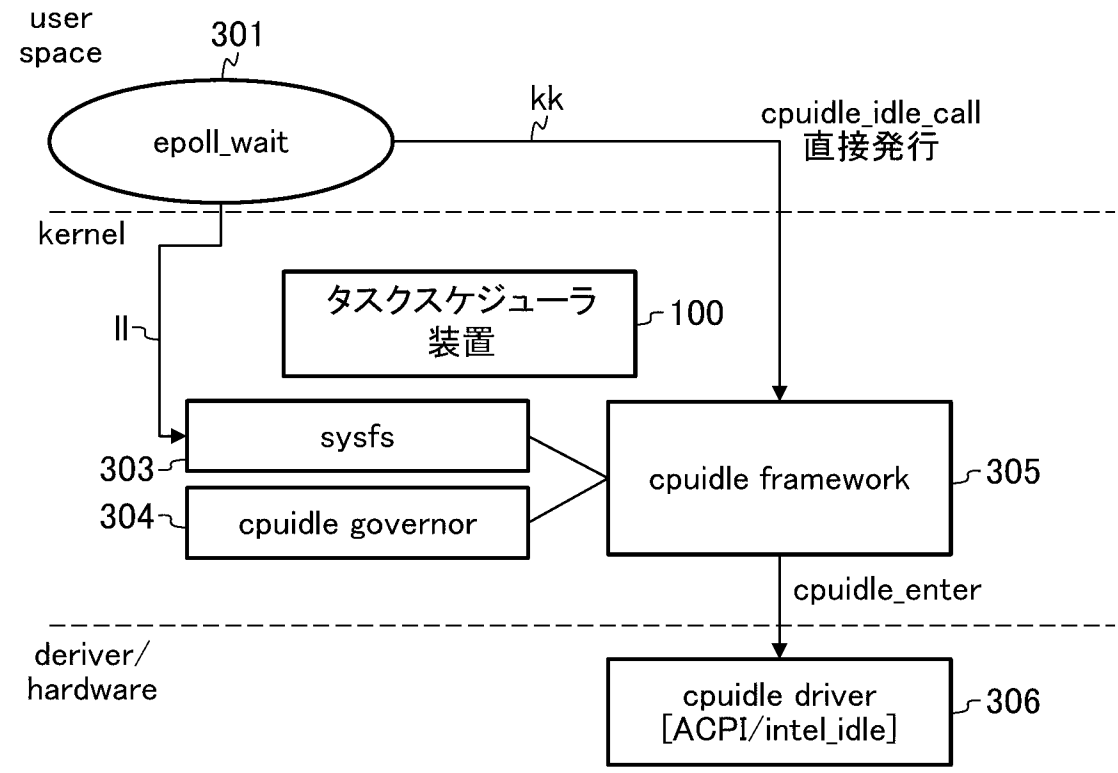
<イベント発生時>



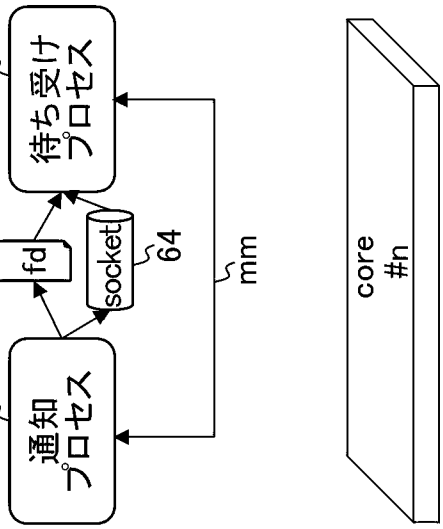
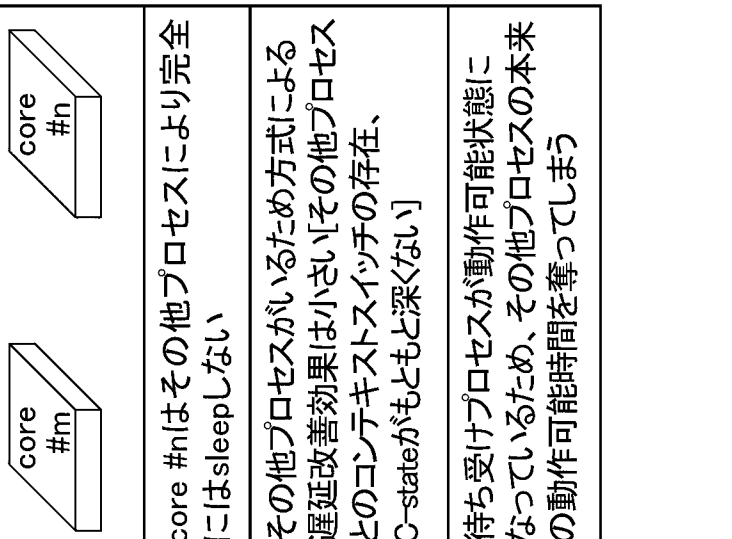
[図8]


user space
301
コンテキストスイッチ
c
302
epoll_wait
idle task
kernel
switch_to
タスクスケジューラ装置
2
cpuidle_idle_call
303
sysfs
304
cpuidle governor
cpuidle framework
305
cpuidle_enter
deriver/ hardware
cpuidle driver [ACPI/intel_idle]
306

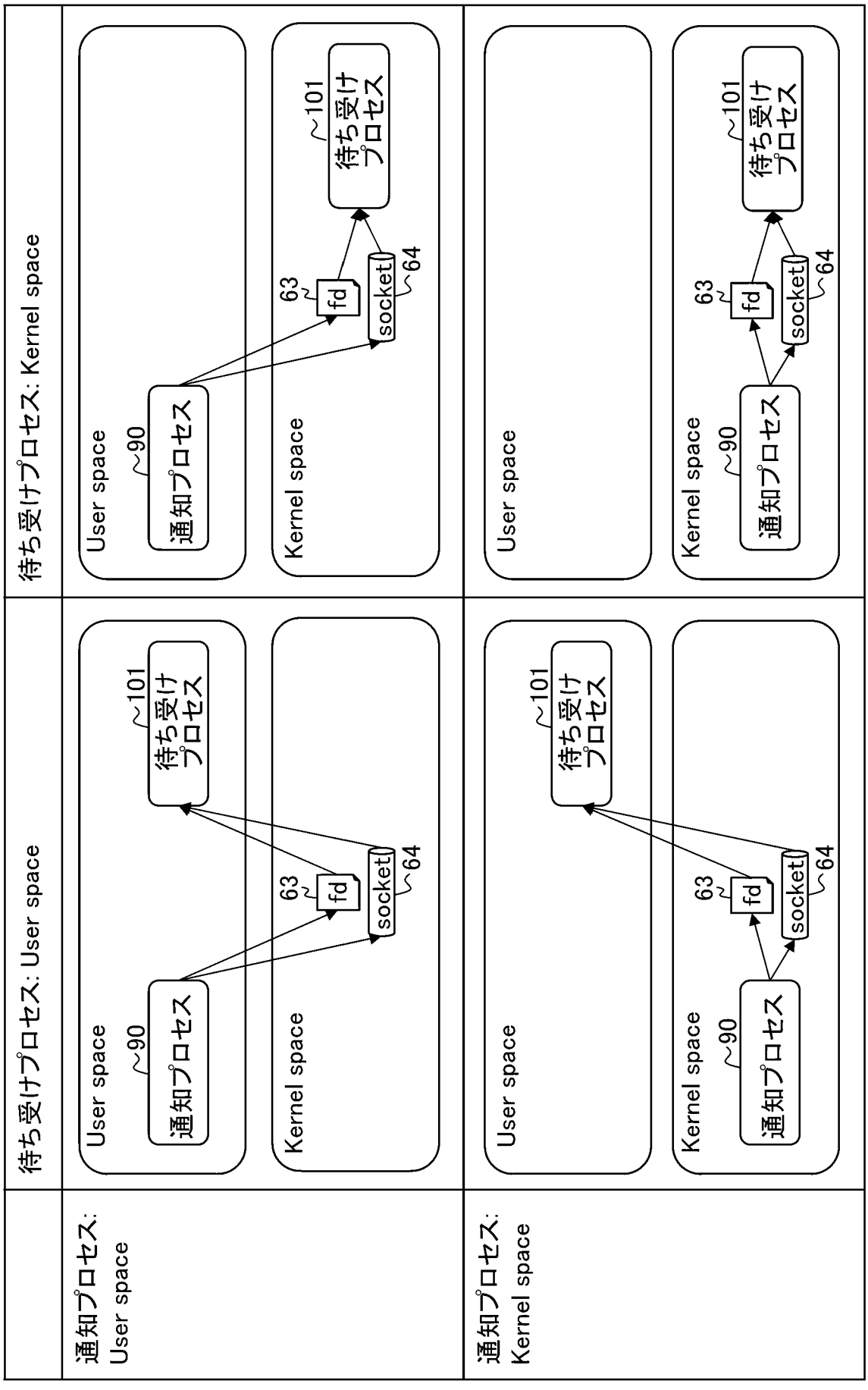
[図9]


user space
301
epoll_wait
kk
cpuidle_idle_call
直接発行
kernel
ll
タスクスケジューラ装置
100
sysfs
303
304
cpuidle governor
cpuidle framework
305
cpuidle_enter
deriver/ hardware
cpuidle driver [ACPI/intel_idle]
306

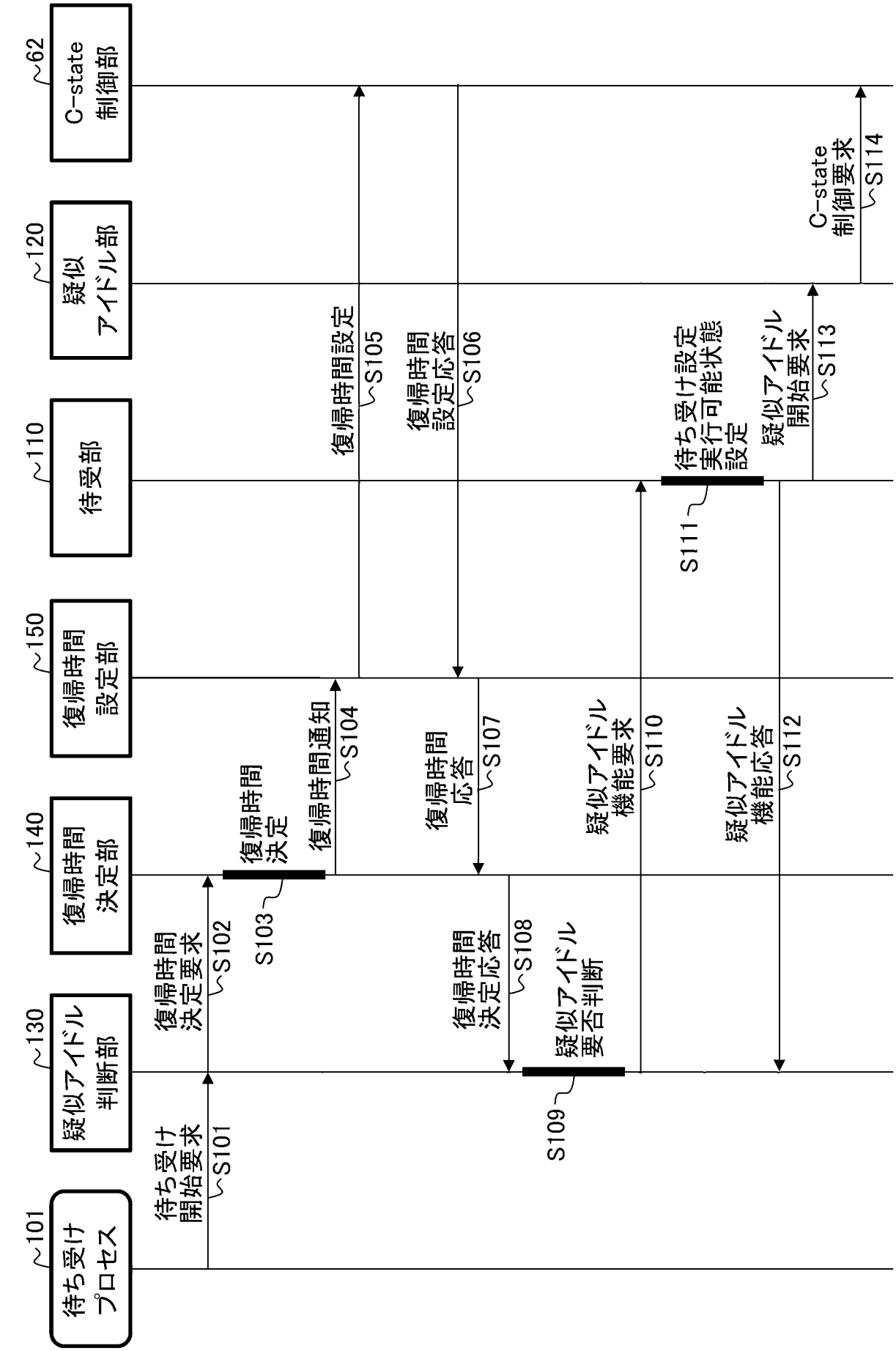
[図10]

	通知プロセスと待ち受けプロセスが異なるCPU coreで動作	通知プロセスと待ち受けプロセスが同一CPU corelに同居	待ち受けプロセスとその他プロセスが同一CPU corelに同居
			
Sleep状態	core #nはsleepする	core #nは通知プロセスにより完全にはsleepしない	core #nはその他プロセスにより完全にはsleepしない
遅延効果	待ち受けプロセスが高速起床する	通知プロセスがいるため方式による遅延改善効果は小さい[通知プロセスとのコンテキストスイッチの存在、C-stateがもとも深くない]	その他プロセスがいるため方式による遅延改善効果は小さい[その他プロセスとのコンテキストスイッチの存在、C-stateがもとも深くない]
懸念	特になし	待ち受けプロセスが動作可能状態になっているため、通知プロセスがイベント検知・通知を送るのが遅くなる	待ち受けプロセスが動作可能状態になっているため、その他プロセスの本来の動作可能時間を奪ってしまう

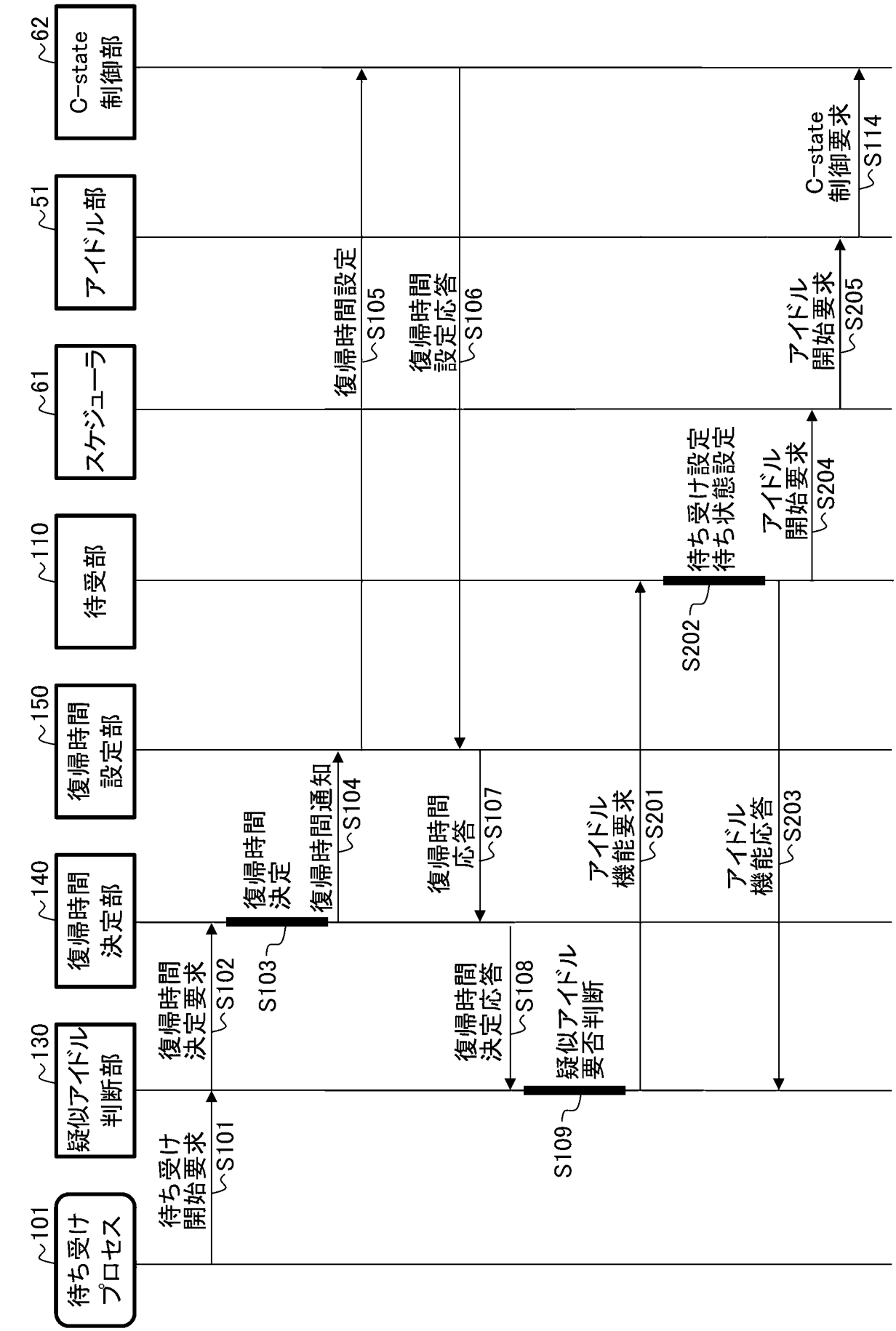
[図11]


通知プロセス: User space
待ち受けプロセス: User space
待ち受けプロセス: Kernel space
通知プロセス: Kernel space
User space
Kernel space
通知プロセス
待ち受けプロセス
fd
socket
90
101
63
64

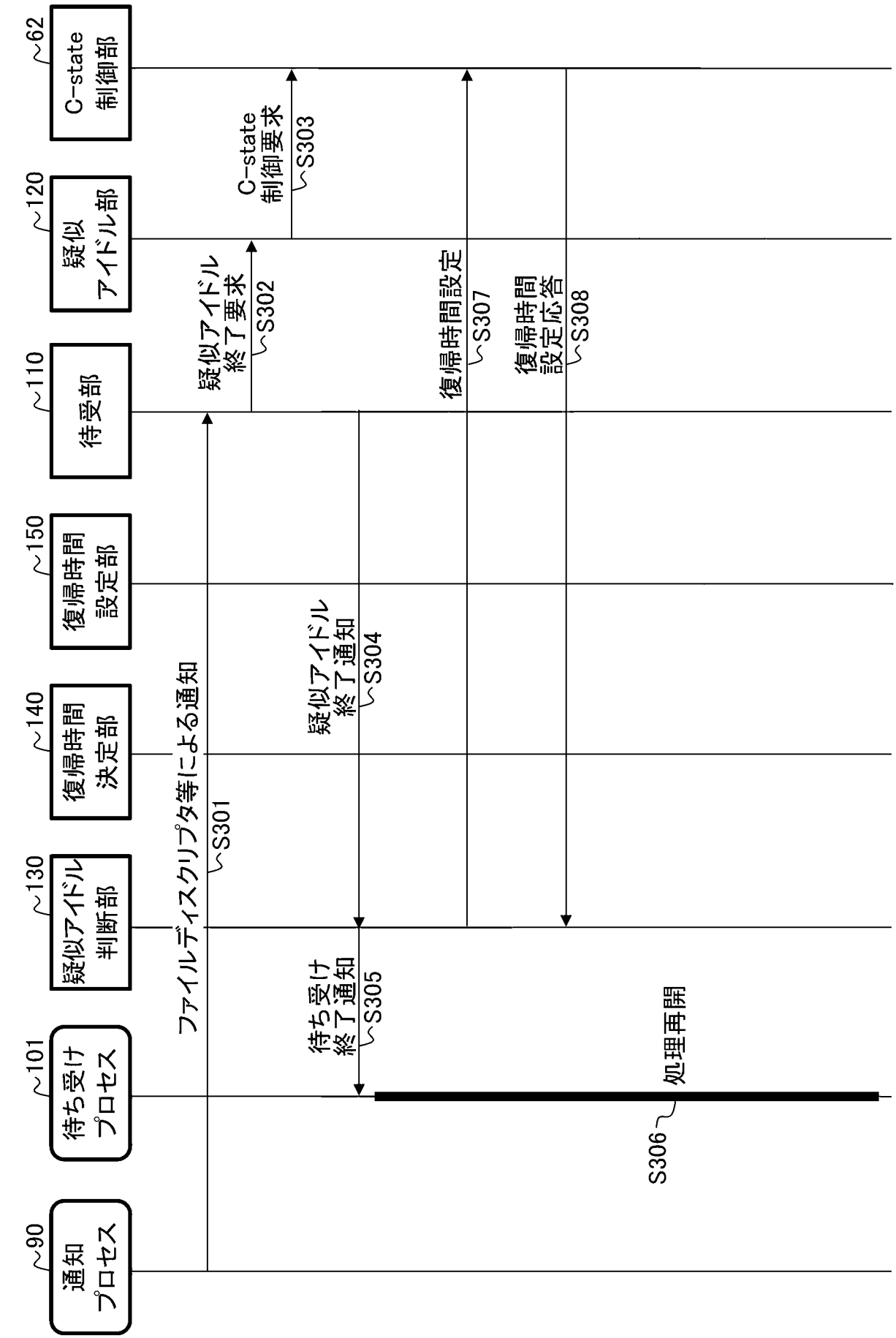
[図12]


101 待ち受けプロセス
130 疑似アイドル判断部
140 復帰時間決定部
150 復帰時間設定部
110 待受部
120 疑似アイドル部
62 C-state制御部
待ち受け開始要求 S101
復帰時間決定要求 S102
復帰時間決定 S103
復帰時間通知 S104
復帰時間設定 S105
復帰時間設定応答 S106
復帰時間応答 S107
復帰時間決定応答 S108
疑似アイドル要否判断 S109
疑似アイドル機能要求 S110
疑似アイドル機能応答 S112
待ち受け設定実行可能状態設定 S111
疑似アイドル開始要求 S113
C-state制御要求 S114

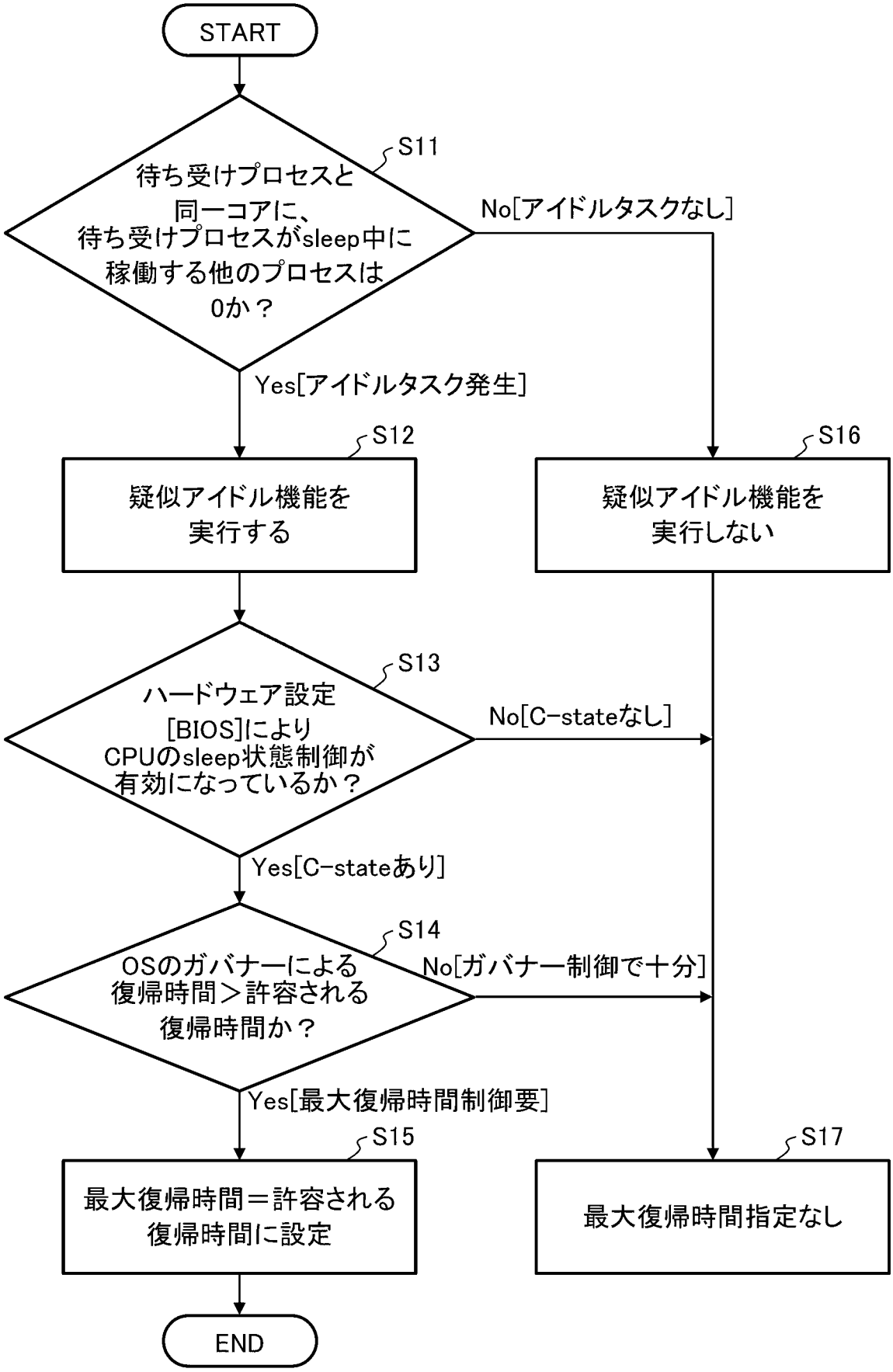
[図13]


~101
待ち受け
プロセス
~130
疑似アイドル
判断部
~140
復帰時間
決定部
~150
復帰時間
設定部
~110
待受部
~61
スケジューラ
~51
アイドル部
~62
C-state
制御部
待ち受け
開始要求
S101
復帰時間
決定要求
S102
復帰時間
決定
S103
復帰時間通知
S104
復帰時間設定
S105
復帰時間
設定応答
S106
復帰時間
応答
S107
復帰時間
決定応答
S108
疑似アイドル
要否判断
S109
アイドル
機能要求
S201
アイドル
機能応答
S203
待ち受け設定
待ち状態設定
S202
アイドル
開始要求
S204
アイドル
開始要求
S205
C-state
制御要求
S114

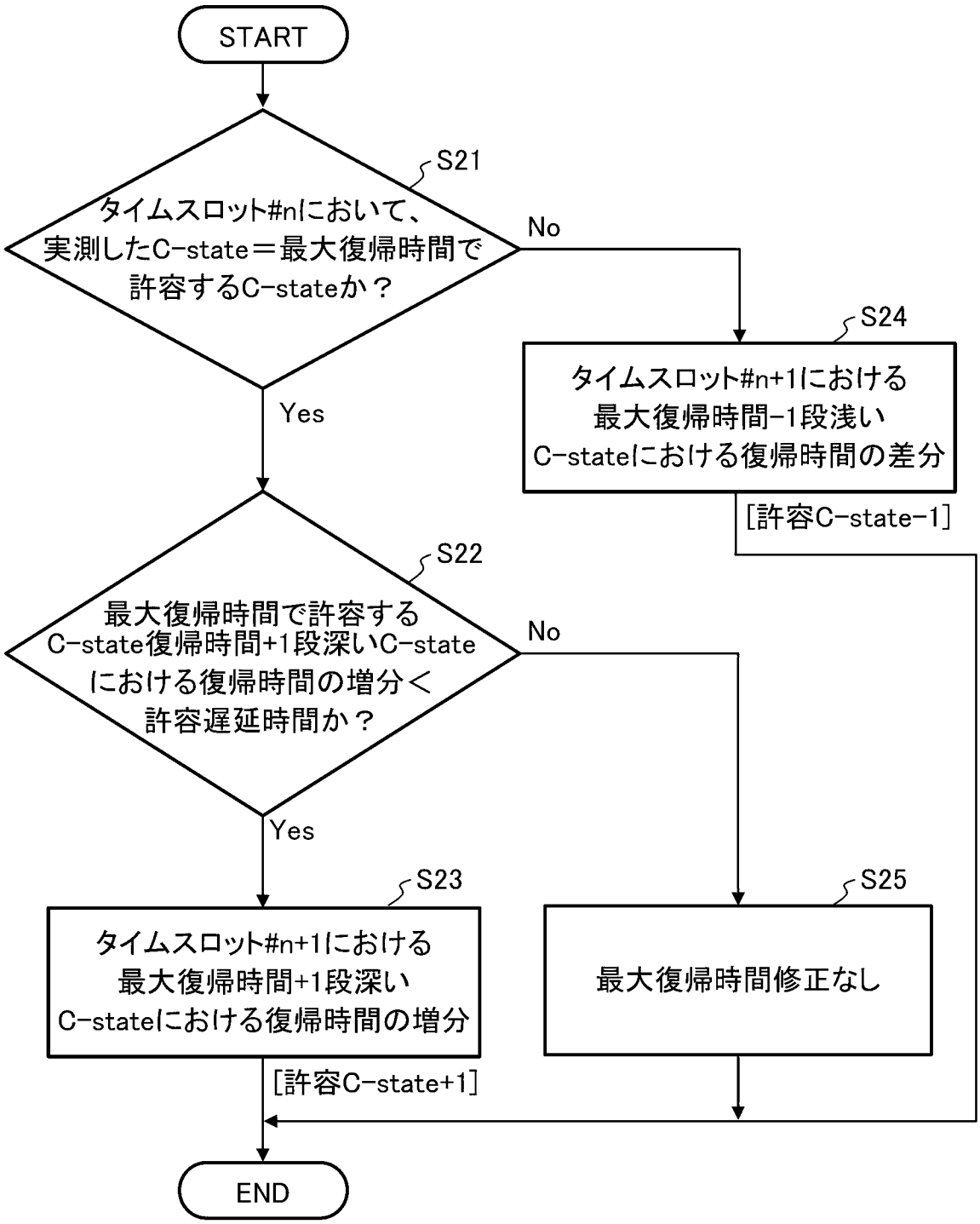
[図14]


~90
通知プロセス
~101
待ち受けプロセス
~130
疑似アイドル判断部
~140
復帰時間決定部
~150
復帰時間設定部
~110
待受部
~120
疑似アイドル部
~62
C-state制御部
ファイルディスクリプタ等による通知
S301
疑似アイドル終了要求
S302
C-state制御要求
S303
疑似アイドル終了通知
S304
待ち受け終了通知
S305
処理再開
S306
復帰時間設定
S307
復帰時間設定応答
S308

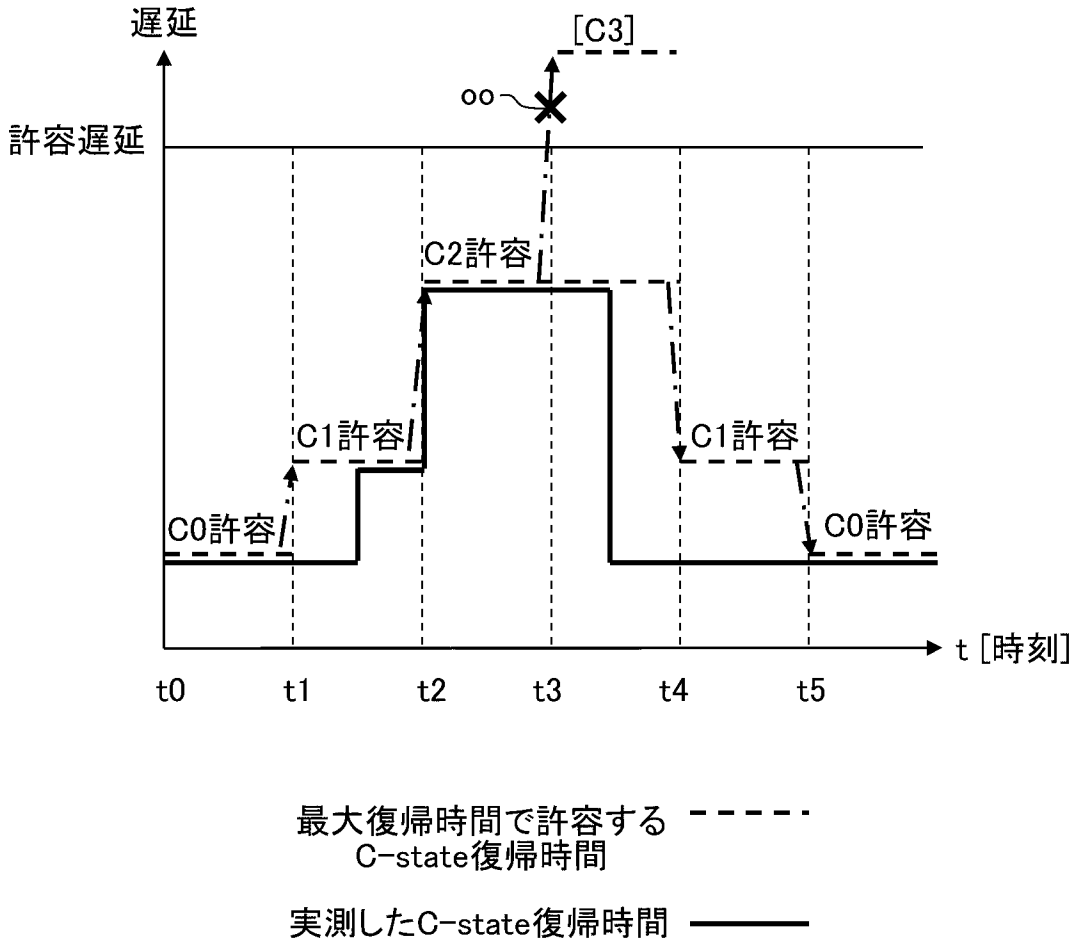
[図15]


START
S11
待ち受けプロセスと
同一コアに、
待ち受けプロセスがsleep中に
稼働する他のプロセスは
0か？
No[アイドルタスクなし]
Yes[アイドルタスク発生]
S12
疑似アイドル機能を
実行する
S16
疑似アイドル機能を
実行しない
S13
ハードウェア設定
[BIOS]により
CPUのsleep状態制御が
有効になっているか？
No[C-stateなし]
Yes[C-stateあり]
S14
OSのガバナーによる
復帰時間＞許容される
復帰時間か？
No[ガバナー制御で十分]
Yes[最大復帰時間制御要]
S15
最大復帰時間＝許容される
復帰時間に設定
S17
最大復帰時間指定なし
END

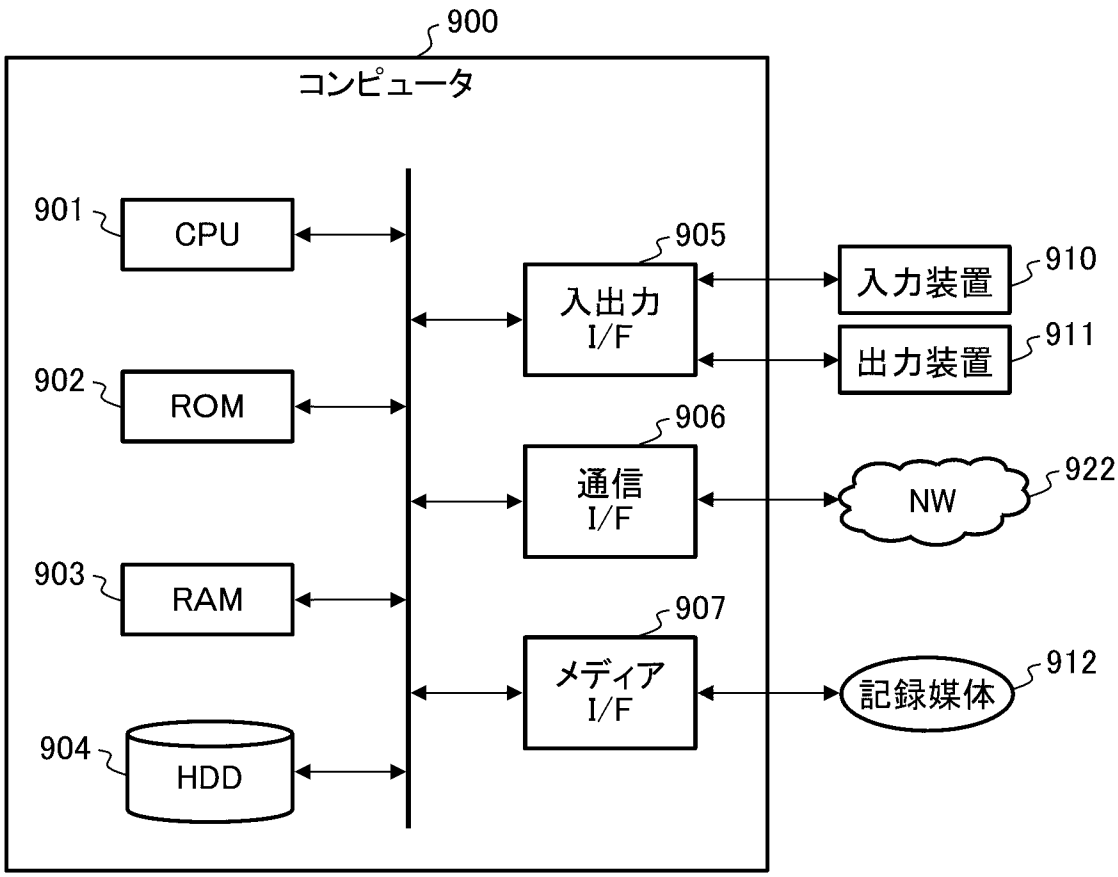
[図16]


START
S21
タイムスロット#nにおいて、
実測したC-state＝最大復帰時間で
許容するC-stateか？
No
Yes
S24
タイムスロット#n+1における
最大復帰時間-1段浅い
C-stateにおける復帰時間の差分
[許容C-state-1]
S22
最大復帰時間で許容する
C-state復帰時間+1段深いC-state
における復帰時間の増分＜
許容遅延時間か？
No
Yes
S23
タイムスロット#n+1における
最大復帰時間+1段深い
C-stateにおける復帰時間の増分
S25
最大復帰時間修正なし
[許容C-state+1]
END

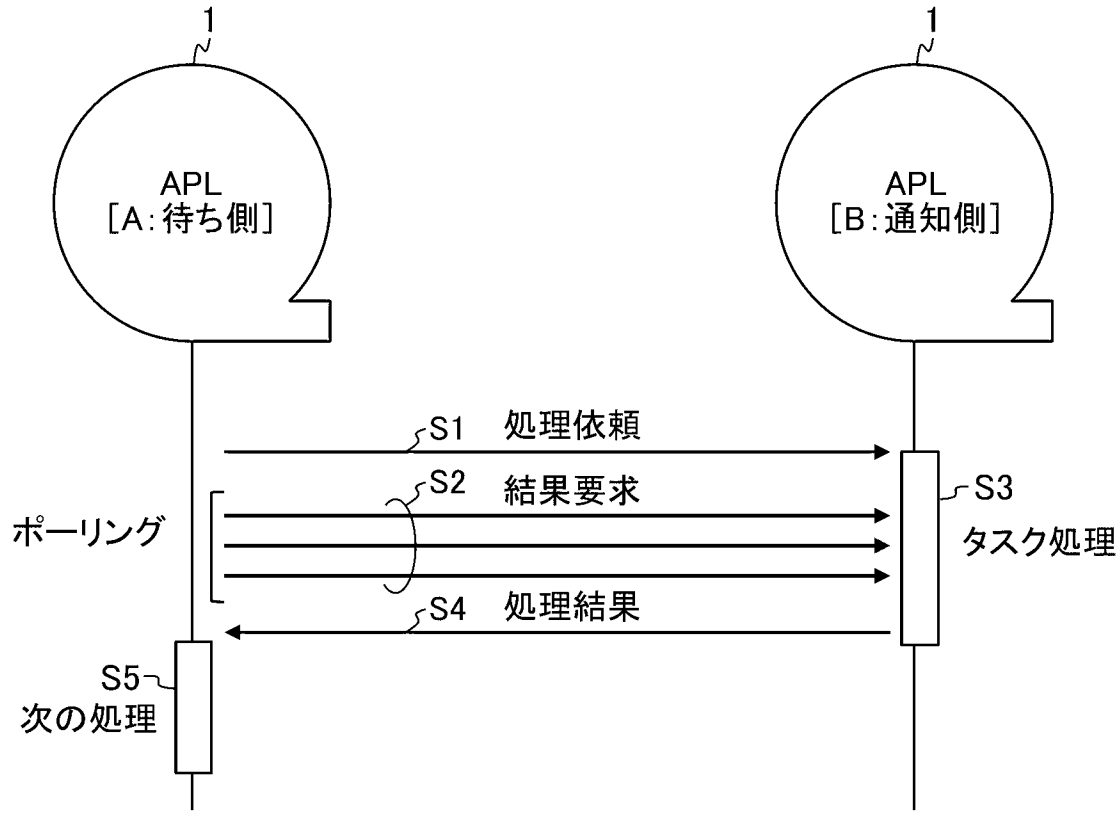
[図17]


遅延
[C3]
oo
許容遅延
C2許容
C1許容
C1許容
C0許容
C0許容
t [時刻]
t0
t1
t2
t3
t4
t5
最大復帰時間で許容する
C-state復帰時間
実測したC-state復帰時間

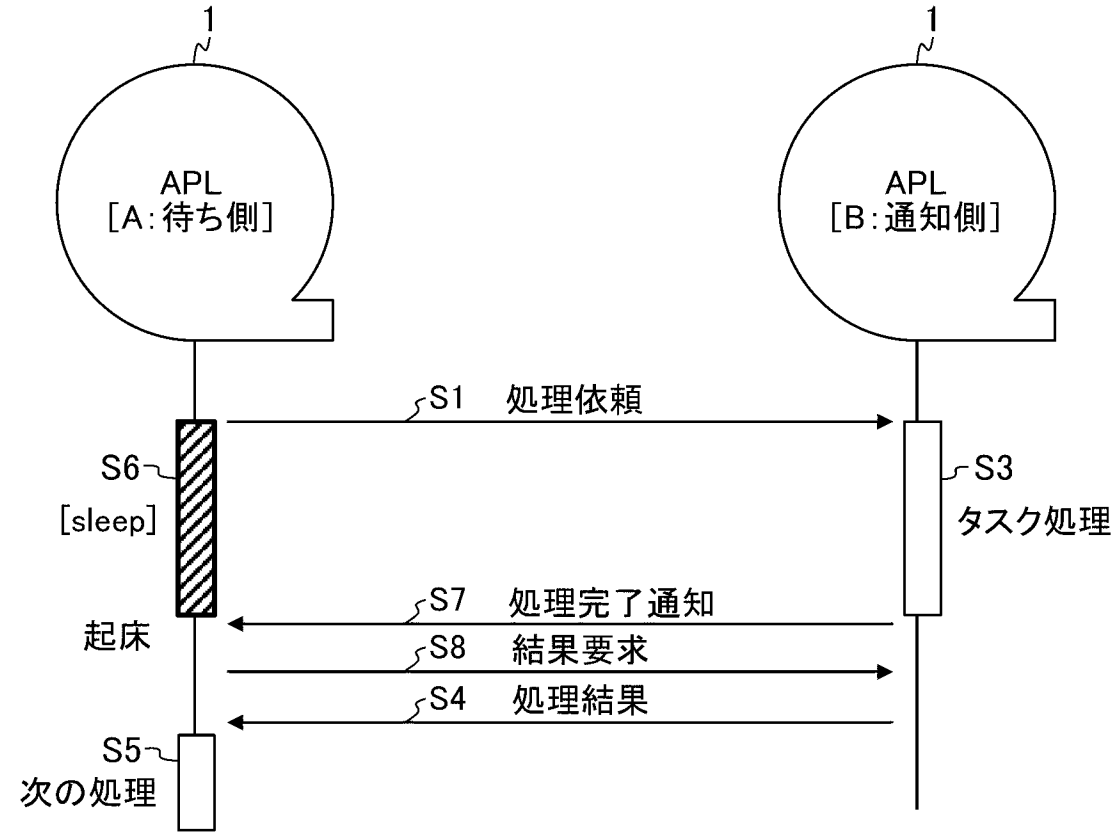
[図18]


900
コンピュータ
901
CPU
902
ROM
903
RAM
904
HDD
905
入出力
I/F
906
通信
I/F
907
メディア
I/F
910
入力装置
911
出力装置
922
NW
912
記録媒体

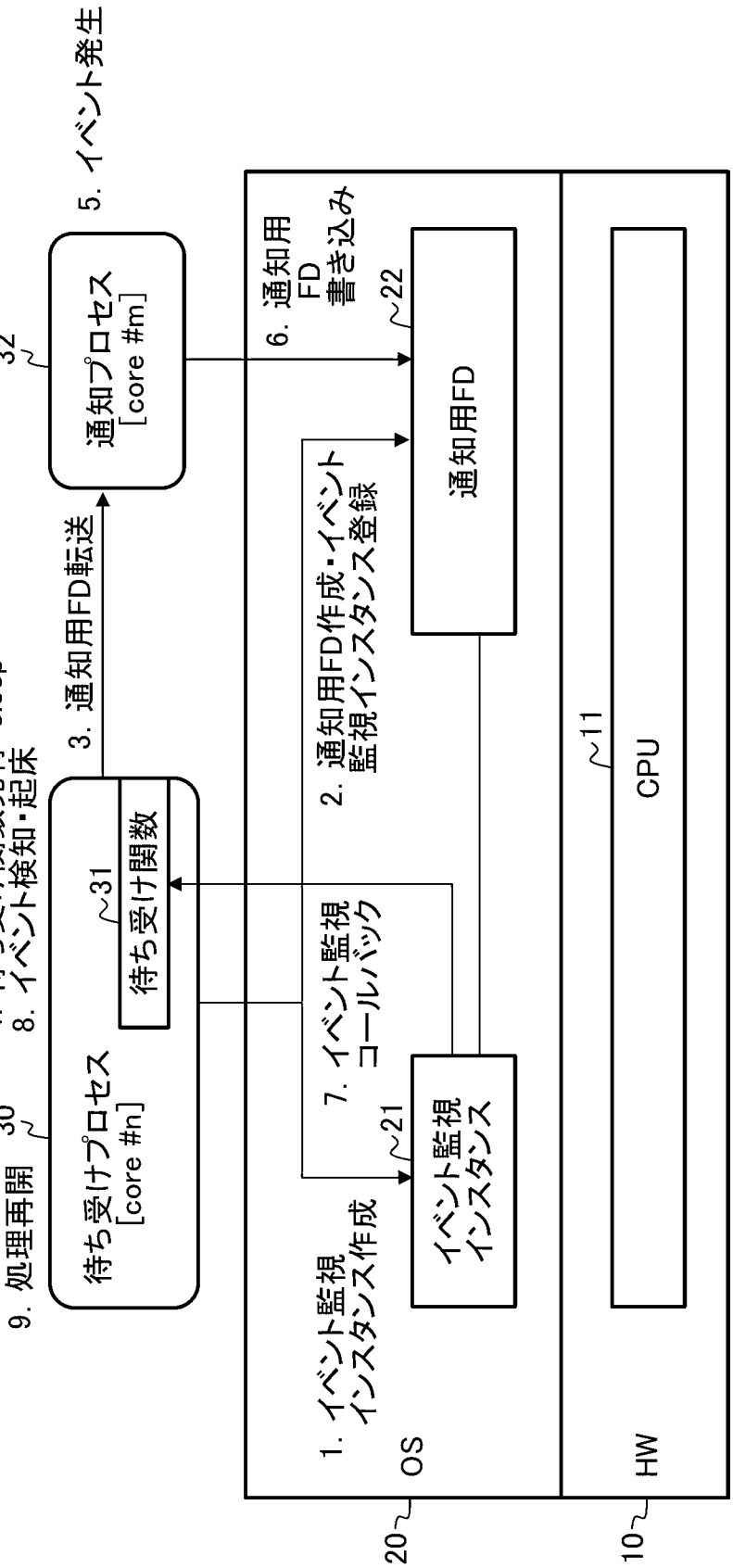
[図19]


1
1
APL
[A:待ち側]
APL
[B:通知側]
S1 処理依頼
S2 結果要求
ポーリング
S3
タスク処理
S4 処理結果
S5
次の処理

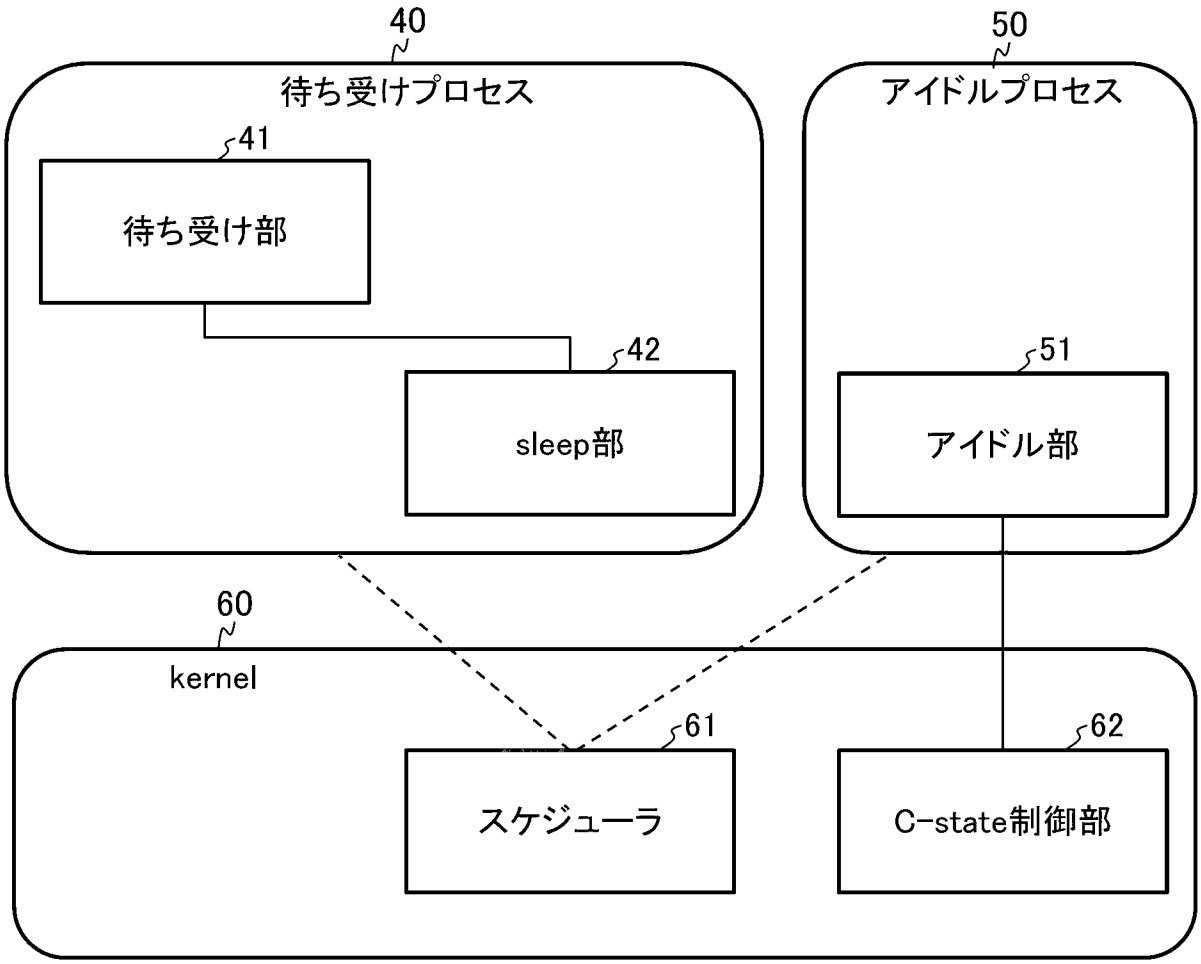
[図20]


1
1
APL
[A:待ち側]
APL
[B:通知側]
S1 処理依頼
S6
[sleep]
S3
タスク処理
S7 処理完了通知
起床
S8 結果要求
S4 処理結果
S5
次の処理

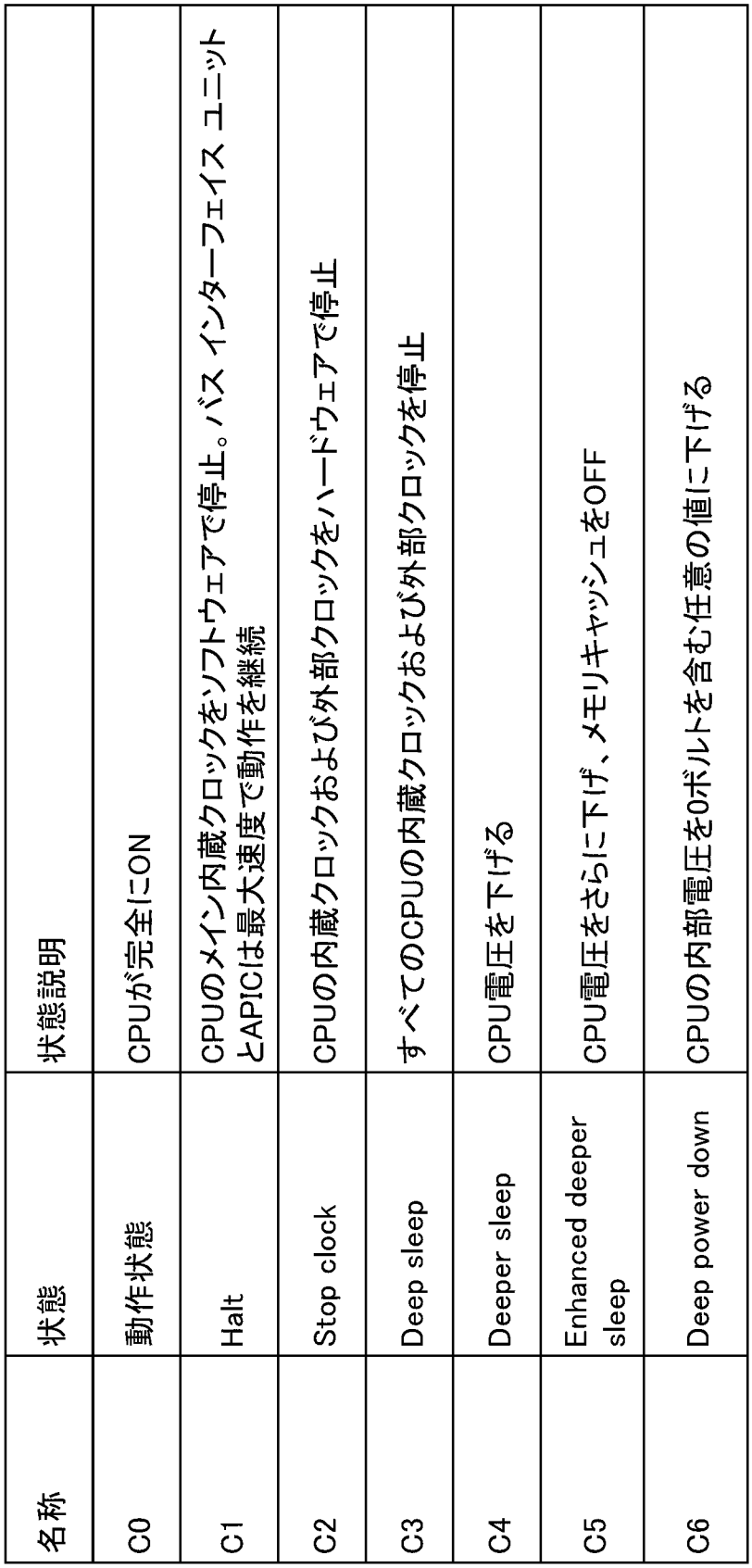
[図21]


1. イベント監視インスタンス作成
2. 通知用FD作成・イベント監視インスタンス登録
3. 通知用FD転送
4. 待ち受け関数発行・sleep
5. イベント発生
6. 通知用FD書き込み
7. イベント監視コールバック
8. イベント検知・起床
9. 処理再開
OS
HW
20
10
11
21
22
30
31
32
待ち受けプロセス [core #n]
待ち受け関数
通知プロセス [core #m]
イベント監視インスタンス
通知用FD
CPU

[図22]

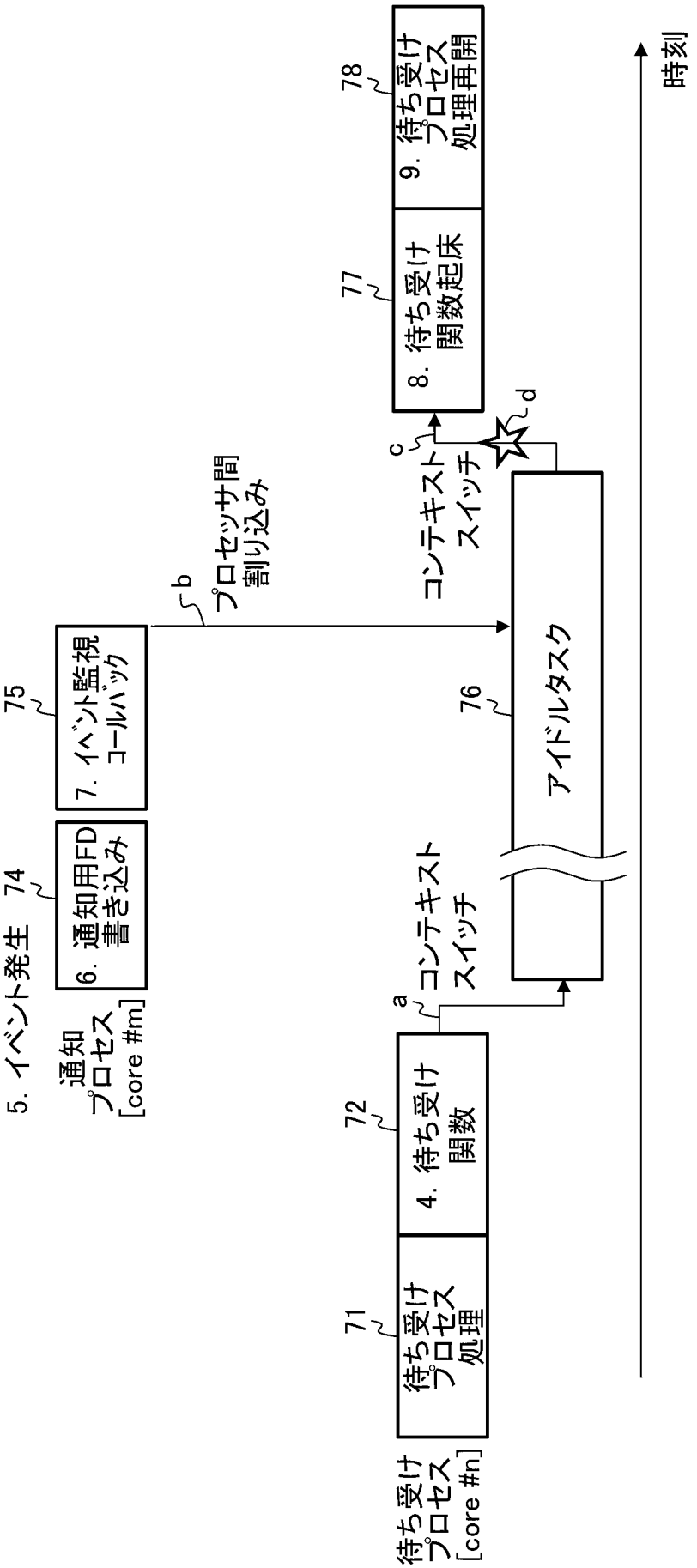

40
待ち受けプロセス
41
待ち受け部
42
sleep部
50
アイドルプロセス
51
アイドル部
60
kernel
61
スケジューラ
62
C-state制御部

[図23]

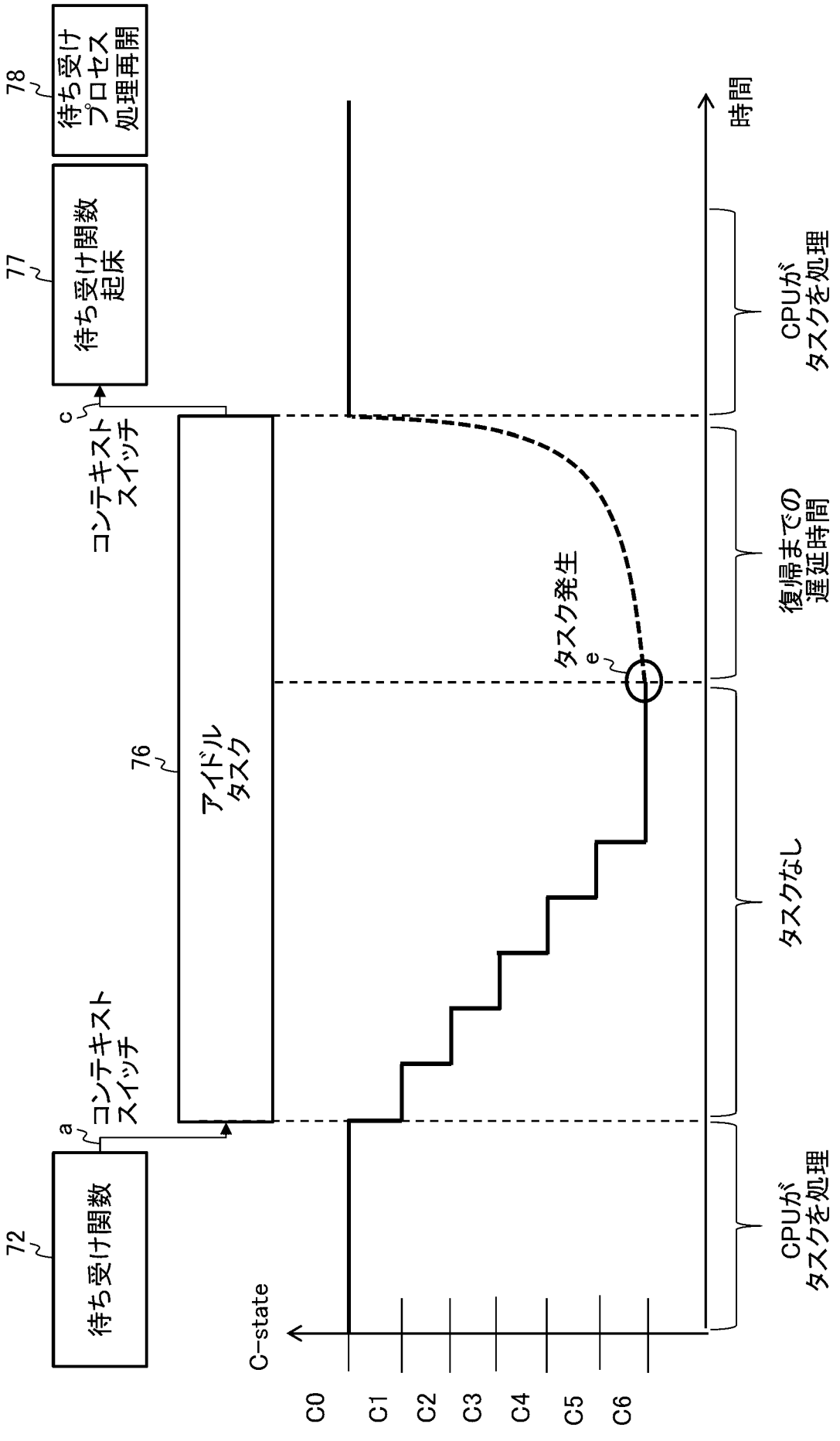


名称	状態	状態説明
C0	動作状態	CPUが完全にON
C1	Halt	CPUのメイン内蔵クロックをソフトウェアで停止。バス インターフェイス ユニットとAPICは最大速度で動作を継続
C2	Stop clock	CPUの内蔵クロックおよび外部クロックをハードウェアで停止
C3	Deep sleep	すべてのCPUの内蔵クロックおよび外部クロックを停止
C4	Deeper sleep	CPU電圧を下げる
C5	Enhanced deeper sleep	CPU電圧をさらに下げ、メモリキャッシュをOFF
C6	Deep power down	CPUの内部電圧を0ボルトを含む任意の値に下げる

[図24]


時刻
待ち受けプロセス [core #n]
通知プロセス [core #m]
5. イベント発生
71
待ち受けプロセス処理
72
4. 待ち受け関数
a
コンテキストスイッチ
アイドルタスク
76
7. イベント監視コールバック
75
6. 通知用FD書き込み
74
b
プロセッサ間割り込み
c
コンテキストスイッチ
d
77
8. 待ち受け関数起床
78
9. 待ち受けプロセス処理再開

[図25]


72
待ち受け関数
a
コンテキスト
スイッチ
76
アイドル
タスク
c
コンテキスト
スイッチ
77
待ち受け関数
起床
78
待ち受け
プロセス
処理再開
タスク発生
e
C-state
C0
C1
C2
C3
C4
C5
C6
時間
CPUが
タスクを処理
復帰までの
遅延時間
タスクなし
CPUが
タスクを処理

INTERNATIONAL SEARCH REPORT

International application No.
PCT/JP2022/029347

A. CLASSIFICATION OF SUBJECT MATTER

G06F 9/54(2006.01)i
FI: G06F9/54 C

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

G06F9/54

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Published examined utility model applications of Japan 1922-1996
Published unexamined utility model applications of Japan 1971-2022
Registered utility model specifications of Japan 1996-2022
Published registered utility model applications of Japan 1994-2022

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	JP 59-55563 A (FUJITSU LTD) 30 March 1984 (1984-03-30) p. 3, lower left column, line 1 to p. 4, upper right column, line 6, fig. 1-3	1-4, 7-8
Y		5-6
Y	塩田紳二, Windows 10の電力管理を支えるACPIを見る, [online]. KADOKAWA ASCII RESEARCH LABIRATRIES, INC. 15 September 2019, pp. 1-4, [retrieved on 12 October 20022], Internet <URL: https://ascii.jp/elem/000/001/937/1937460/2/> in particular, see chapter entitled "Processor Power States", (SHIODA, Shinji. See ACPI that supports power management for Windows 10.)	5-6
A	JP 2009-153192 A (CANON INC) 09 July 2009 (2009-07-09) paragraph [0157]	1-8

☐ Further documents are listed in the continuation of Box C. ☑ See patent family annex.

* Special categories of cited documents:
"A" document defining the general state of the art which is not considered to be of particular relevance
"E" earlier application or patent but published on or after the international filing date
"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)
"O" document referring to an oral disclosure, use, exhibition or other means
"P" document published prior to the international filing date but later than the priority date claimed
"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
12 October 2022	**25 October 2022**
Name and mailing address of the ISA/JP **Japan Patent Office (ISA/JP)** **3-4-3 Kasumigaseki, Chiyoda-ku, Tokyo 100-8915** **Japan**	Authorized officer Telephone No.

INTERNATIONAL SEARCH REPORT
Information on patent family members

International application No.
PCT/JP2022/029347

Patent document cited in search report	Publication date (day/month/year)	Patent family member(s)	Publication date (day/month/year)
JP 59-55563 A	30 March 1984	(Family: none)	
JP 2009-153192 A	09 July 2009	(Family: none)	

国際調査報告

国際出願番号
PCT/JP2022/029347

A. 発明の属する分野の分類（国際特許分類（ＩＰＣ））

G06F 9/54(2006.01)i
FI: G06F9/54 C

B. 調査を行った分野

調査を行った最小限資料（国際特許分類（ＩＰＣ））

G06F9/54

最小限資料以外の資料で調査を行った分野に含まれるもの

日本国実用新案公報	１９２２－１９９６年
日本国公開実用新案公報	１９７１－２０２２年
日本国実用新案登録公報	１９９６－２０２２年
日本国登録実用新案公報	１９９４－２０２２年

国際調査で使用した電子データベース（データベースの名称、調査に使用した用語）

C. 関連すると認められる文献

引用文献のカテゴリー*	引用文献名　及び一部の箇所が関連するときは、その関連する箇所の表示	関連する請求項の番号
X	JP 59-55563 A (富士通株式会社) 30.03.1984 (1984-03-30) 第3頁左下欄第1行目－第4頁右上欄第6行目,第1図－第3図	1-4,7-8
Y		5-6
Y	塩田紳二, Windows 10の電力管理を支えるACPIを見る, [online], KADOKAWA ASCII Research Labiratries, Inc., 2019.09.15, 第1頁－第4頁, [2022年10月12日検索],インターネット<ＵＲＬ：https://ascii.jp/elem/000/001/937/1937460/2/> 特に「プロセッサーパワーステート」と題した章を参照	5-6
A	JP 2009-153192 A (キヤノン株式会社) 09.07.2009 (2009-07-09) 段落0157	1-8

☐ Ｃ欄の続きにも文献が列挙されている。

☑ パテントファミリーに関する別紙を参照。

* 引用文献のカテゴリー
“A” 特に関連のある文献ではなく、一般的技術水準を示すもの
“E” 国際出願日前の出願または特許であるが、国際出願日以後に公表されたもの
“L” 優先権主張に疑義を提起する文献又は他の文献の発行日若しくは他の特別な理由を確立するために引用する文献（理由を付す）
“O” 口頭による開示、使用、展示等に言及する文献
“P” 国際出願日前で、かつ優先権の主張の基礎となる出願の日の後に公表された文献
“T” 国際出願日又は優先日後に公表された文献であって出願と抵触するものではなく、発明の原理又は理論の理解のために引用するもの
“X” 特に関連のある文献であって、当該文献のみで発明の新規性又は進歩性がないと考えられるもの
“Y” 特に関連のある文献であって、当該文献と他の１以上の文献との、当業者にとって自明である組合せによって進歩性がないと考えられるもの
“&” 同一パテントファミリー文献

国際調査を完了した日	国際調査報告の発送日
12.10.2022	25.10.2022

名称及びあて先
日本国特許庁(ISA/JP)
〒100-8915
日本国
東京都千代田区霞が関三丁目４番３号

権限のある職員（特許庁審査官）
三坂　敏夫 5B 4178

電話番号 03-3581-1101 内線 3545

国際調査報告
パテントファミリーに関する情報

国際出願番号
PCT/JP2022/029347

引用文献	公表日	パテントファミリー文献	公表日
JP 59-55563 A	30.03.1984	（ファミリーなし）	
JP 2009-153192 A	09.07.2009	（ファミリーなし）	