(19)国家知识产权局

(12)发明专利申请

(10)申请公布号 CN 114816031 A
(43)申请公布日 2022.07.29

(21)申请号 202210497684.5

(22)申请日 2022.05.09

(71)申请人 海信电子科技(深圳)有限公司
地址 518054 广东省深圳市南山区粤海街道创业路1777号海信南方大厦9楼

(72)发明人 辛将

(74)专利代理机构 北京同达信恒知识产权代理有限公司 11291
专利代理师 刘彩红

(51)Int.Cl.
G06F 1/3234(2019.01)
G06F 1/329(2019.01)

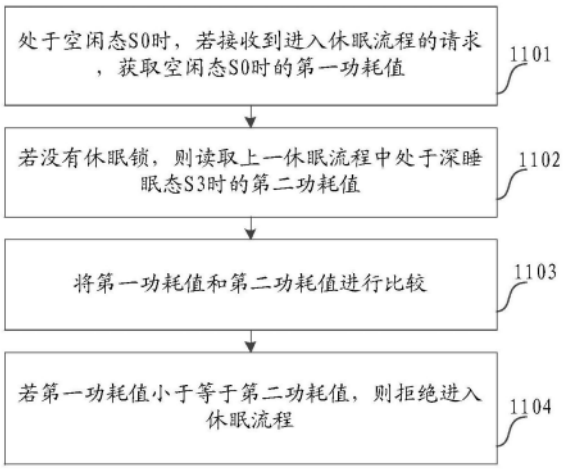
权利要求书1页 说明书13页 附图10页

(54)发明名称

终端设备的省电方法、终端设备及介质

(57)摘要

本申请公开了一种终端设备的省电方法、终端设备及介质，用以解决如何降低终端设备功耗的问题。该方法通过在处于空闲态S0时，若接收到进入休眠流程的请求且内核层没有休眠锁时，通过比较空闲态S0时的第一功耗值和上一休眠流程中处于深睡眠态S3时的第二功耗值；若第一功耗值小于等于第二功耗值，则拒绝进入休眠流程，终端设备一直处于空闲态S0，若第一功耗值大于第二功耗值，则进入休眠流程，终端设备处于深睡眠态S3，由此，可以保证在一些因系统资源使用错误或者其它软件出现错误而导致待机耗电大的情况时，不进深睡眠态S3，可以尽可能的降低终端设备的功耗，达到省电的目的。



CN 114816031 A

1.一种终端设备的省电方法,其特征在于,所述方法包括:

处于空闲态S0时,若接收到进入休眠流程的请求,获取空闲态S0时的第一功耗值;

若没有休眠锁,则读取上一休眠流程中处于深睡眠态S3时的第二功耗值;

将所述第一功耗值和所述第二功耗值进行比较;

若所述第一功耗值小于等于所述第二功耗值,则拒绝进入所述休眠流程。

2.根据权利要求1所述的方法,其特征在于,所述方法还包括:

若所述第一功耗值大于所述第二功耗值,则进入所述休眠流程。

3.根据权利要求1所述的方法,其特征在于,所述拒绝进入所述休眠流程包括:

通过写第一节点的方式触发底层驱动上报第一事件给上层系统;所述第一事件用于指示所述上层系统拒绝进入所述休眠流程。

4.根据权利要求2所述的方法,其特征在于,所述进入所述休眠流程,包括:

通过写第二节点的方式触发底层驱动上报第二事件给上层系统;所述第二事件用于指示所述上层系统进入所述休眠流程。

5.根据权利要求1所述的方法,其特征在于,所述获取空闲态S0时的第一功耗值,包括:

读取电池驱动的内核节点,获得所述空闲态S0的第一功耗值。

6.根据权利要求1所述的方法,其特征在于,所述读取上一休眠流程中深睡眠态S3的第二功耗值,包括:

读取所述深睡眠态S3对应的寄存器中存储的功耗值,获得所述第二功耗值。

7.根据权利要求6所述的方法,其特征在于,所述方法还包括:

当所述上一休眠流程中深睡眠态S3被唤醒时,将所述上一休眠流程中深睡眠态S3的第二功耗值存储在所述寄存器中。

8.一种终端设备,其特征在于,包括:

显示器、处理器和存储器;

所述显示器用于显示屏幕显示区域;

所述存储器,用于存储所述处理器可执行指令;

所述处理器被配置为执行所述指令以实现如权利要求1-7中任一项所述的终端设备的省电方法。

9.一种计算机可读存储介质,其特征在于,当所述计算机可读存储介质中的指令由终端设备执行时,使得所述终端设备能够执行如权利要求1-7中任一项所述的终端设备的省电方法。

10.一种计算机程序产品,其特征在于,包括计算机程序:

所述计算机程序被处理器执行时实现如权利要求1-7中任一项所述的终端设备的省电方法。

终端设备的省电方法、终端设备及介质

技术领域

[0001] 本申请涉及终端技术领域，特别涉及一种终端设备的省电方法、终端设备及介质。

背景技术

[0002] 随着终端设备的日渐普及，终端设备的功耗控制一直是业界难题，终端设备的耗电最主要还是在soc芯片(System on Chip，系统级芯片)本身，soc芯片的状态大致根据耗电程度不同可以分为空闲态S0、睡眠态S1、深睡眠态S3、关机态S4等，一般终端设备在soc loading(SOC承载)较小的时候进到空闲态S0较多，然后如果发起休眠流程则会进入睡眠态S1，最终进入深睡眠态S3，如果终端设备关机则会进入关机态S4。

[0003] 基于以上状态的变更虽然能够节约终端设备的功耗，但是如果进一步节约功耗仍需要优化。

发明内容

[0004] 本申请的目的是提供一种终端设备的省电方法、终端设备及介质，用以解决如何降低终端设备功耗的问题。

[0005] 第一方面，本申请提供一种终端设备的省电方法，所述方法包括：

[0006] 处于空闲态S0时，若接收到进入休眠流程的请求，获取空闲态S0时的第一功耗值；

[0007] 若没有休眠锁，则读取上一休眠流程中处于深睡眠态S3时的第二功耗值；

[0008] 将所述第一功耗值和所述第二功耗值进行比较；

[0009] 若所述第一功耗值小于等于所述第二功耗值，则拒绝进入所述休眠流程。

[0010] 在一种可能的实施方式中，所述方法还包括：

[0011] 若所述第一功耗值大于所述第二功耗值，则进入所述休眠流程。

[0012] 在一种可能的实施方式中，所述拒绝进入所述休眠流程包括：

[0013] 通过写第一节点的方式触发底层驱动上报第一事件给上层系统；所述第一事件用于指示所述上层系统拒绝进入所述休眠流程。

[0014] 在一种可能的实施方式中，所述进入所述休眠流程，包括：

[0015] 通过写第二节点的方式触发底层驱动上报第二事件给上层系统；所述第二事件用于指示所述上层系统进入所述休眠流程。

[0016] 在一种可能的实施方式中，所述获取空闲态S0时的第一功耗值，包括：

[0017] 读取电池驱动的内核节点，获得所述空闲态S0的第一功耗值。

[0018] 在一种可能的实施方式中，所述读取上一休眠流程中深睡眠态S3的第二功耗值，包括：

[0019] 读取所述深睡眠态S3对应的寄存器中存储的功耗值，获得所述第二功耗值。

[0020] 在一种可能的实施方式中，所述方法还包括：

[0021] 当所述上一休眠流程中深睡眠态S3被唤醒时，将所述上一休眠流程中深睡眠态S3的第二功耗值存储在所述寄存器中。

[0022] 在一种可能的实施方式中，所述唤醒所述上一休眠流程中深睡眠态S3，包括：

[0023] 在所述上一休眠流程中深睡眠态S3下，若主动与网络建立连接或因外部中断响应事件触发，则立即唤醒所述上一休眠流程中深睡眠态S3，退出深睡眠态S3，建立链接、发起或响应业务。

[0024] 第二方面，本申请提供一种终端设备，包括：

[0025] 显示器、处理器和存储器；

[0026] 所述显示器用于显示屏幕显示区域；

[0027] 所述存储器，用于存储所述处理器可执行指令；

[0028] 所述处理器被配置为执行所述指令以实现如上述第一方面中任一项所述的终端设备的省电方法。

[0029] 第三方面，本申请提供一种计算机可读存储介质，当所述计算机可读存储介质中的指令由终端设备执行时，使得所述终端设备能够执行如上述第一方面中任一项所述的终端设备的省电方法。

[0030] 第四方面，本申请提供一种计算机程序产品，包括计算机程序：

[0031] 所述计算机程序被处理器执行时实现如上述第一方面中任一项所述的终端设备的省电方法。

[0032] 本申请的实施例提供的技术方案至少带来以下有益效果：

[0033] 本申请实施例通过在处于空闲态S0时，若接收到进入休眠流程的请求且内核层没有休眠锁时，通过比较空闲态S0时的第一功耗值和上一休眠流程中处于深睡眠态S3时的第二功耗值；若第一功耗值小于等于第二功耗值，则拒绝进入休眠流程，终端设备一直处于空闲态S0，若第一功耗值大于第二功耗值，则进入休眠流程，终端设备处于深睡眠态S3，由此，可以保证在一些因系统资源使用错误或者其它软件出现错误而导致待机耗电大的情况时，不进深睡眠态S3，可以尽可能的降低终端设备的功耗，达到省电的目的。

[0034] 本申请的其它特征和优点将在随后的说明书中阐述，并且，部分地从说明书中变得显而易见，或者通过实施本申请而了解。本申请的目的和其他优点可通过在所写的说明书、权利要求书、以及附图中所特别指出的结构来实现和获得。

附图说明

[0035] 为了更清楚地说明本申请实施例的技术方案，下面将对本申请实施例中所需要使用的附图作简单地介绍，显而易见地，下面所介绍的附图仅仅是本申请的一些实施例，对于本领域普通技术人员来讲，在不付出创造性劳动的前提下，还可以根据这些附图获得其他的附图。

[0036] 图1为本申请实施例提供的终端设备处于正常态的示意图；

[0037] 图2为本申请实施例提供的终端设备处于空闲态的示意图；

[0038] 图3为本申请实施例提供的终端设备处于睡眠态的示意图；

[0039] 图4为本申请实施例提供的终端设备处于深睡眠态的示意图；

[0040] 图5为本申请实施例提供的终端设备处于关机态的示意图；

[0041] 图6为本申请实施例提供的一种终端设备的结构示意图；

[0042] 图7为本申请实施例提供的一种终端设备的软件结构框图；

[0043] 图8为本申请实施例提供的一种终端设备的电源管理流程的示意图；

[0044] 图9为本申请实施例提供的终端设备的内核层的休眠唤醒流程的示意图；

[0045] 图10为本申请实施例提供的一种终端设备的省电方法的流程示意图；

[0046] 图11为本申请实施例提供的另一种终端设备的省电方法的流程示意图；

[0047] 图12为本申请实施例提供的拒绝进入休眠流程的流程示意图；

[0048] 图13为本申请实施例提供的进入休眠流程的流程示意图；

[0049] 图14为本申请实施例提供的另一种终端设备的省电方法的流程示意图；

[0050] 图15为本申请实施例提供的另一种终端设备的示意图；

[0051] 图16为本申请实施例提供的一种终端设备的省电装置的示意图；

[0052] 图17为本申请实施例提供的另一种终端设备的省电装置的示意图；

[0053] 图18为本申请实施例提供的另一种终端设备的省电装置的示意图；

[0054] 图19为本申请实施例提供的另一种终端设备的省电装置的示意图；

[0055] 图20为本申请实施例提供的另一种终端设备的省电装置的示意图。

具体实施方式

[0056] 为使本申请实施例的目的、技术方案和优点更加清楚，下面将结合本申请实施例中的附图，对本申请实施例中的技术方案进行清楚、完整地描述。其中，所描述的实施例是本申请一部分实施例，而不是全部的实施例。基于本申请中的实施例，本领域普通技术人员在没有做出创造性劳动前提下所获得的所有其它实施例，都属于本申请保护的范围。

[0057] 并且，在本申请实施例的描述中，除非另有说明，“/”表示或的意思，例如，A/B可以表示A或B；文本中的“和/或”仅仅是一种描述关联对象的关联关系，表示可以存在三种关系，例如，A和/或B，可以表示：单独存在A，同时存在A和B，单独存在B这三种情况，另外，在本申请实施例的描述中，“多个”是指两个或多于两个。

[0058] 以下，术语“第一”、“第二”仅用于描述目的，而不能理解为暗示或暗示相对重要性或者隐含指明所指示的技术特征的数量。由此，限定有“第一”、“第二”、的特征可以明示或者隐含地包括一个或者更多个该特征，在本申请实施例的描述中，除非另有说明，“多个”的含义是两个或两个以上。

[0059] 随着终端设备的日渐普及，终端设备的功耗控制一直是业界难题，终端设备的耗电最主要还是在soc芯片(System on Chip，系统级芯片)本身，soc芯片的状态大致根据耗电程度不同可以分为空闲态S0、睡眠态S1、深睡眠态S3、关机态S4等，一般终端设备在soc loading(SOC承载)较小的时候进到空闲态S0较多，然后如果发起休眠流程则会进入睡眠态S1，最终进入深睡眠态S3，如果终端设备关机则会进入关机态S4。

[0060] 其中，为了省电，芯片的耗电等级一般分为以下几种，简单来说干活的时候能跑在低频就在低频，不干活的时候能关闭的资源就关闭，有任务需要处理的时候就使能一些资源然后去工作。linux系统中的几种不同的状态(State in Linux)以及对应的标签(Label)、电源状态(State)、ACPI state(Advanced Configuration and Power Interface state，先进设置和电源管理状态)以及对应的终端设备的状态如图1至图5所示。

[0061] 终端设备共有5种状态，分别为正常态、空闲态、睡眠态、深睡眠态、关机态。其中如图1所示，终端设备处于正常态，在State in Linux中的#define PM_SUSPEND_ON((_force

suspend_state_t)0)状态对应的终端设备一切正常。

[0062] 如图2所示,终端设备处于空闲态,在State in Linux中的#define PM_SUSPEND_FREEZE((_force suspend_state_t)1)状态对应的标签为freeze(冻结)、对应的状态为Suspend-to Idle(休眠到空闲状态)、对应的ACPI state为S0,对应终端设备已经frozen processes(冻结进程)+suspended devices(挂起设备)+idle processors(空闲的处理器)。具有轻量化的特点;相对于终端设备一切正常时能节省更多的功耗,因为此时的用户空间被冻结且I/O(Input/Output,输入/输出)设备进入了低功耗状态,且相对于Suspend-To-RAM它具有低延时的优势。Suspend-to Idle即为下文中的Idle状态,也是S0状态。

[0063] 如图3所示,终端设备处于睡眠态,在State in Linux中的#define PM_SUSPEND_STANDBY((_force suspend_state_t)2)状态对应的标签为standby(待机)、对应的状态为Standby/Power-on Suspend(休眠的待机态)、对应的ACPI state为S1,对应终端设备已经frozen processes(冻结进程)+suspended devices(挂起设备)+offline nonboot CPUs(关闭不开机的处理器)+suspend low-level system(暂停低级系统),对CPU的处理更近一步。所以相对于Suspend-To-Idle状态节省了更多的功耗,但是由于需要恢复CPU和一些底层功能,所以唤醒也花费了更多的时间。

[0064] 如图4所示,终端设备处于深睡眠态,在State in Linux中的#define PM_SUSPEND_MEM((_force suspend_state_t)3)状态对应的标签为mem(深睡眠状态)、对应的状态为Suspend-to RAM(休眠到仅保留RAM状态)、对应的ACPI state为S3,对应终端设备仅保留RAM(Random Access Memory,随机存取存储器)自刷新,所有的设备和系统状态都保存在RAM中,所有外设被挂起。此状态使所有的设备进入低功耗状态,相对于Standby/Power-on Suspend状态节省了更多的功耗,唤醒花费的时间也更长。Suspend-to RAM即为下文中的suspend状态,也是S3状态。

[0065] 如图5所示,终端设备处于关机态,在State in Linux中的#define PM_SUSPEND_MAX((_force suspend_state_t)4)状态对应的标签为disk(关机状态)、对应的状态为Suspend-to disk(休眠到关机状态)、对应的ACPI state为S4,对应终端设备关闭所有设备,包括RAM。此状态是最省功耗的模式。相对Suspend-to-RAM能节省更多功耗的原因是数据会被写入磁盘中,RAM也可以被关闭。但是这也导致需要更多的唤醒时间,在resume(恢复)的时候读回到RAM,然后再进行系统和设备状态的恢复工作。但是在一般的嵌入式设备上,并不支持此种状态。

[0066] 虽然图1-图5中展示了终端设备的5种状态,但是在本申请中只针对图2-图5展示的终端设备的四种状态进行讨论。在图2-图5中,从freeze(s0)到standby(s1)到mem(s3)最后到disk(s4),表示四种状态的睡眠程度由浅到深,唤醒花费的时间也由短到长,耗电程度则由大到小。

[0067] 基于以上状态的变更虽然能够节约终端设备的功耗,但是如果进一步节约功耗仍需要优化。

[0068] 现有技术中,终端设备的休眠流程一般是固定的,且不同的阶段耗电程度不同,这样对降低终端设备的功耗有一定的益处,但是相关技术并未提出若因系统资源使用错误或者其它软件错误导致终端设备在深睡眠态时耗电较大时,如何降低终端设备的功耗。

[0069] 有鉴于此,本申请提供了一种终端设备的省电方法、终端设备及介质,用以解决如

何降低终端设备功耗的问题。

[0070] 本申请的发明构思可概括为:本申请实施例中通过在处于空闲态S0时,若接收到进入休眠流程的请求且内核层没有休眠锁时,通过比较空闲态S0时的第一功耗值和上一休眠流程中处于深睡眠态S3时的第二功耗值;若第一功耗值小于等于第二功耗值,则拒绝进入休眠流程,终端设备一直处于空闲态S0,若第一功耗值大于第二功耗值,则进入休眠流程,终端设备处于深睡眠态S3,由此,可以保证在一些因系统资源使用错误或者其它软件出现错误而导致待机耗电大的情况时,不进深睡眠态S3,可以尽可能的降低终端设备的功耗,达到省电的目的。

[0071] 在介绍完本申请的发明构思之后,下面先对本申请提供的终端设备进行说明。

[0072] 图6示出了一种终端设备100的结构示意图。应该理解的是,图6所示终端设备100仅是一个范例,并且终端设备100可以具有比图6中所示的更多的或者更少的部件,可以组合两个或多个的部件,或者可以具有不同的部件配置。图中所示出的各种部件可以在包括一个或多个信号处理和/或专用集成电路在内的硬件、软件、或硬件和软件的组合中实现。

[0073] 图6中示例性示出了根据示例性实施例中终端设备100的硬件配置框图。如图6所示,终端设备100包括:射频(radio frequency,RF)电路110、存储器120、显示单元130、摄像头140、传感器150、音频电路160、无线保真(Wireless Fidelity,Wi-Fi)模块170、处理器180、蓝牙模块181、以及电源190等部件。

[0074] RF电路110可用于在收发信息或通话过程中信号的接收和发送,可以接收基站的下行数据后交给处理器180处理;可以将上行数据发送给基站。通常,RF电路包括但不限于天线、至少一个放大器、收发信机、耦合器、低噪声放大器、双工器等器件。

[0075] 存储器120可用于存储软件程序及数据。处理器180通过运行存储在存储器120的软件程序或数据,从而执行终端设备100的各种功能以及数据处理。存储器120可以包括高速随机存取存储器,还可以包括非易失性存储器,例如至少一个磁盘存储器件、闪存器件、或其他易失性固态存储器件。存储器120存储有使得终端设备100能运行的操作系统。本申请中存储器120可以存储操作系统及各种应用程序,还可以存储执行本申请实施例终端设备的省电方法的程序代码。

[0076] 显示单元130可用于接收输入的数字或字符信息,产生与终端设备100的用户设置以及功能控制有关的信号输入,具体地,显示单元130可以包括设置在终端设备100正面的触摸屏131,可收集用户在其上或附近的触摸操作,例如点击按钮。

[0077] 显示单元130还可用于显示由用户输入的信息或提供给用户的信息以及终端设备100的各种菜单的图形用户界面(graphical user interface,GUI)。具体地,显示单元130可以包括设置在终端设备100正面的显示屏132。其中,显示屏132可以采用液晶显示器、发光二极管等形式来配置。显示单元130可以用于显示本申请中终端设备的屏幕显示区域。

[0078] 其中,触摸屏131可以覆盖在显示屏132之上,也可以将触摸屏131与显示屏132集成而实现终端设备100的输入和输出功能,集成后可以简称触摸显示屏。本申请中显示单元130可以显示应用程序以及对应的操作步骤。

[0079] 摄像头140可用于捕获静态图像或视频。物体通过镜头生成光学图像投射到感光元件。感光元件可以是电荷耦合器件(charge coupled device,CCD)或互补金属氧化物半导体(complementary metal-oxide-semiconductor,CMOS)光电晶体管。感光元件把光信号

转换成电信号,之后将电信号传递给处理器180转换成数字图像信号。

[0080] 终端设备100还可以包括至少一种传感器150,比如加速度传感器151、距离传感器152、指纹传感器153、温度传感器154。终端设备100还可配置有陀螺仪、气压计、湿度计、温度计、红外线传感器、光传感器、运动传感器等其他传感器。

[0081] 音频电路160、扬声器161、麦克风162可提供用户与终端设备100之间的音频接口。音频电路160可将接收到的音频数据转换后的电信号,传输到扬声器161,由扬声器161转换为声音信号输出。终端设备100还可配置音量按钮,用于调节声音信号的音量,还可以用于组合其他按钮,调整封闭区域。另一方面,麦克风162将收集的声音信号转换为电信号,由音频电路160接收后转换为音频数据,再将音频数据输出至RF电路110以发送给比如另一终端设备,或者将音频数据输出至存储器120以便进一步处理。

[0082] Wi-Fi属于短距离无线传输技术,终端设备100可以通过Wi-Fi模块170帮助用户收发电子邮件、浏览网页和访问流媒体等,它为用户提供了无线的宽带互联网访问。

[0083] 处理器180是终端设备100的控制中心,利用各种接口和线路连接整个终端设备的各个部分,通过运行或执行存储在存储器120内的软件程序,以及调用存储在存储器120内的数据,执行终端设备100的各种功能和处理数据。在一些实施例中,处理器180可包括一个或多个处理单元;处理器180还可以集成应用处理器和基带处理器,其中,应用处理器主要处理操作系统、用户界面和应用程序等,基带处理器主要处理无线通信。可以理解的是,上述基带处理器也可以不集成到处理器180中。本申请中处理器180可以运行操作系统、应用程序、用户界面显示及触控响应,以及本申请实施例的终端设备的省电方法。另外,处理器180与显示单元130耦接。

[0084] 蓝牙模块181,用于通过蓝牙协议来与其他具有蓝牙模块的蓝牙设备进行信息交互。例如,终端设备100可以通过蓝牙模块181与同样具备蓝牙模块的可穿戴电子设备(例如智能手表)建立蓝牙连接,从而进行数据交互。

[0085] 终端设备100还包括给各个部件供电的电源190(比如电池)。电源可以通过电源管理系统与处理器180逻辑相连,从而通过电源管理系统实现管理充电、放电以及功耗等功能。终端设备100还可配置有电源按钮,用于终端设备的开机和关机,以及锁屏等功能。

[0086] 图7是本申请实施例的一种终端设备100的软件结构框图。

[0087] 分层架构将软件分成若干个层,每一层都有清晰的角色和分工。层与层之间通过软件接口通信。在一些实施例中,可将Android系统分为四层,从上至下分别为应用程序层,应用程序框架层,安卓运行时(Android runt ime)和系统库,以及内核层。

[0088] 应用程序层可以包括一系列应用程序包。

[0089] 如图7所示,应用程序包可以包括电话、彩信,WiFi,微信,信息,闹钟,图库,日历,WLAN等应用程序。

[0090] 应用程序框架层为应用程序层的应用程序提供应用编程接口(application programming interface,API)和编程框架。应用程序框架层包括一些预先定义的函数。

[0091] 如图7所示,应用程序框架层可以包括窗口管理器,内容提供器,视图系统,电话管理器,资源管理器,通知管理器等。

[0092] 窗口管理器用于管理窗口程序。窗口管理器可以获取显示屏大小,判断是否有状态栏,锁定屏幕,截取屏幕等。

[0093] 内容提供器用来存放和获取数据，并使这些数据可以被应用程序访问。数据可以包括视频，图像，音频，拨打和接听的电话，浏览历史和书签，电话簿、短信息等。

[0094] 视图系统包括可视控件，例如显示文字的控件，显示图片的控件等。视图系统可用于构建应用程序。显示界面可以由一个或多个视图组成的。例如，包括短信息通知图标的显示界面，可以包括显示文字的视图以及显示图片的视图。

[0095] 电话管理器用于提供终端设备100的通信功能。例如通话状态的管理(包括接通，挂断等)。

[0096] 资源管理器为应用程序提供各种资源，比如本地化字符串，图标，图片，布局文件，视频文件等。

[0097] 通知管理器使应用程序可以在状态栏中显示通知信息(例如短信息的消息内容)，可以用于传达告知类型的消息，可以短暂停留后自动消失，无需用户交互。比如通知管理器被用于告知下载完成，消息提醒等。通知管理器还可以是以图表或者滚动条文本形式出现在系统顶部状态栏的通知，例如后台运行的应用程序的通知，还可以是以对话窗口形式出现在屏幕上的通知。例如在状态栏提示文本信息，发出提示音，终端设备振动，指示灯闪烁等。

[0098] Android Runtime包括核心库和虚拟机。Android runtime负责安卓系统的调度和管理。

[0099] 核心库包含两部分：一部分是java语言需要调用的功能函数，另一部分是安卓的核心库。

[0100] 应用程序层和应用程序框架层运行在虚拟机中。虚拟机将应用程序层和应用程序框架层的java文件执行为二进制文件。虚拟机用于执行对象生命周期的管理，堆栈管理，线程管理，安全和异常的管理，以及垃圾回收等功能。

[0101] 系统库可以包括多个功能模块。例如：表面管理器(surface manager)，媒体库(Media Libraries)，三维图形处理库(例如：OpenGL ES)，2D图形引擎(例如：SGL)等。

[0102] 表面管理器用于对显示子系统进行管理，并且为多个应用程序提供了2D和3D图层的融合。

[0103] 媒体库支持多种常用的音频，视频格式回放和录制，以及静态图像文件等。媒体库可以支持多种音视频编码格式，例如：MPEG4，H.264，MP3，AAC，AMR，JPG，PNG等。

[0104] 三维图形处理库用于实现三维图形绘图，图像渲染，合成，和图层处理等。

[0105] 2D(一种动画方式)图形引擎是2D绘图的绘图引擎。

[0106] 内核层是硬件和软件之间的层。内核层至少包含显示驱动，摄像头驱动，音频驱动，传感器驱动。

[0107] 本申请实施例中的终端设备100可以为包括但不限于智能手机、平板电脑、可穿戴电子设备(例如智能手表)、笔记本电脑等电子设备。

[0108] 为了便于理解本申请实施例提供的终端设备的省电方法，下面结合附图对终端设备的电源管理流程进行进一步说明。

[0109] 参照图8，为本申请实施例提供的终端设备的电源管理流程的示意图。如图8所示，目前终端设备的电源管理流程处理部分一般可以分为userspace层(用户空间层)和kernel层(内核层)两部分，其中，userspace层包括Application(应用程序层)、Framework(应用程

序框架层)、JNI(Java Native Interface,Java本机接口)、Hal(Hardware Abstraction Layer,硬件抽象层)。

[0110] 其中Application包括各个应用App(Application,应用程序),例如图8中的App 1、App 2、App3等。Framework包括Powermanager(电源管理)。JNI包括Native接口(本地接口)。Hal包括wakelock(休眠锁)、wakeup_count(唤醒数量)。kernel层包括wake_lock(休眠锁)、wakeup_count(唤醒数量)、suspend(休眠流程)和wakeup source(唤醒源)。

[0111] 其中Application中的应用App(Application,应用程序)或者驱动程序相关模块为了保证自己的程序任务能被cpu(central processing unit,中央处理器)执行完毕,需要通过Framework和JNI向Hal的wakelock申请wake lock(休眠锁)。这样系统若检测到Hal有休眠锁就不会继续向内核层发起休眠流程,不会进行挂起cpu的相关流程,例如如果有任务在执行,那么cpu就处于active(活跃)状态,当任务执行完毕,cpu会进到idle(空闲)进程中,cpu处于空闲态,此时对应上文中的s0状态,不会进入休眠流程。但是如果系统检测到Hal没有休眠锁,就会继续向内核层发起休眠流程,不会阻止cpu跑休眠流程,那么当任务执行完毕,系统会最终跑完所有休眠流程,最终进入到s3状态。

[0112] 同时如图8所示,当系统会进入到s3状态时,内核层会使用wakeup source管理是否进行唤醒机制。其中Hal的wakeup_count的功能是和suspend(休眠)同步。

[0113] 用户空间电源管理进程或者内核线程在发起状态切换前,读取系统的wakeup_count,该值记录了当前的wakeup事件的总数,并将读取的count告知Framework层。Framework层记录该count到一个全局变量中。接着用户空间电源管理进程或者内核线程发起电源状态切换,执行suspend过程。在suspend的过程中,Framework层照常工作,上报wakeup事件,增加wakeup事件的总数。suspend执行的一些时间点会调用Framework层提供的接口,检查是否有wakeup没有处理,即比较当前的wakeup事件的总数和用户空间电源管理进程或者内核线程发起电源状态切换前记录的事件的总数,如果不同,就要进行唤醒机制,终止suspend过程。

[0114] 参照图9,为本申请实施例提供的终端设备的内核层的休眠唤醒流程的示意图。如图9所示,在userspace层的write/sys/power/state中写入电池的状态为进入休眠流程,进入suspend态时,会通知Kenrel层进入休眠流程。如图9所示,内核层的休眠流程主要做的工作是首先处理管理核心(PM core),冻结相关进程,包括prepare console(准备控制台)、freeze process&thread(系统进程冻结)、suepend console(暂停控制台),然后冻结外设(Device),包括suspend device(暂停设备),进低功耗状态,接着管理处理器(CPUS&irqs),包括disable nonboot cpus(使空闲处理器停止运转)、disable irqs(使终端请求无效),不再响应相关中断,最后处理系统核心管理(syscore PM)对系统资源,如bus(总线)等进行低功耗处理,包括suspend syscore(暂停核心处理器),然后check wakeup pending(检查唤醒等待),确认所有资源挂起,没有唤醒机制进行,则到达suspend enter(进到s3状态)。如果此时进行唤醒机制,唤醒流程如图9右边所示的流程,正好与进入休眠流程的流程相反。

[0115] 在了解了终端设备的休眠唤醒流程之后,发现在内核层的休眠流程中会涉及到大量的外设、ap(Access Point,无线接入点)外其它子系统、系统资源等的低功耗操作,涉及到的软件和硬件处理非常复杂,因此比较容易出现一些功耗问题,从而导致待机的实际功

耗反而比终端设备处于idle态时的功耗还要大,因此本申请提供了一种终端设备的省电方法,在系统休眠时的流程上做出优化,增加一个比较功耗的步骤,不进入s3状态而是停留在s0状态,以节省功耗。如图9所示,终端设备在userspace层,即处于s0状态,从userspace层到内核层,即是从s0状态到s3状态的过程。

[0116] Wake lock(休眠锁)是安卓系统的技术,它的主要作用就是阻止系统休眠,让cpu保持在活跃状态,从而保证相关业务程序的正确执行完。因此系统在处于空闲态S0时,若接收到进入休眠流程的请求,在本申请实施例中可以首先执行如图10所示的步骤:

[0117] 在步骤1001中,检测系统中是否存在休眠锁。

[0118] 若检测到休眠锁,则在步骤1002中,系统保留在空闲态S0,并读取空闲态S0的第一功耗值。

[0119] 若没有检测到休眠锁,则在步骤1003中,系统进入休眠流程,直至进入深睡眠态S3。

[0120] 在步骤1004中,唤醒深睡眠态S3,读取深睡眠态S3的第二功耗值。

[0121] 本申请实施例中不涉及休眠锁的操作和功能改变,只是在步骤1003系统进入休眠流程之前增加对功耗值的比较,从而决定系统是否进入休眠流程。如图11所示,该方法包括以下步骤:

[0122] 在步骤1101中,处于空闲态S0时,若接收到进入休眠流程的请求,获取空闲态S0时的第一功耗值。

[0123] 在一种可能的实施方式中,因为终端设备的电池驱动里有相关的内核节点,可以直接读到实时的电流大小,即实时的功耗值,因此本申请中获取空闲态S0时的第一功耗值,可以直接读取电池驱动的内核节点,获得空闲态S0的第一功耗值。

[0124] 由此,可以获取当前处于空闲态S0时的第一功耗值。

[0125] 在步骤1102中,若没有休眠锁,则读取上一休眠流程中处于深睡眠态S3时的第二功耗值。

[0126] 在一种可能的实施方式中,终端设备在suspend态的时候,进程都是冻结的,这个时候的功耗值是直接读取不了的,因此只能等待suspend态被唤醒之后,读取pmic(Power Management Integrated Circuits,电源管理芯片)的相关寄存器,寄存器可以记录suspend态下系统的功耗值。同时因为终端设备处于每次处于深睡眠态S3时,功耗基本是一样的,因此在本申请实施例中读取的是上一休眠流程中深睡眠态S3的第二功耗值,包括读取深睡眠态S3对应的寄存器中存储的功耗值,获得第二功耗值。

[0127] 在一种可能的实施方式中,当上一休眠流程中深睡眠态S3被唤醒时,将上一休眠流程中深睡眠态S3的第二功耗值存储在寄存器中。

[0128] 在一种可能的实施方式中,需要在上一休眠流程中深睡眠态S3被唤醒时存储第二功耗值,因此需要唤醒上一休眠流程中的深睡眠态S3,具体包括:在上一休眠流程中深睡眠态S3下,若主动与网络建立连接或因外部中断响应事件触发,则立即唤醒上一休眠流程中深睡眠态S3,退出深睡眠态S3,建立链接、发起或响应业务。即为唤醒上一休眠流程中深睡眠态S3,此时可以将第二功耗值存储在寄存器中。

[0129] 由此,可以唤醒上一休眠流程中的深睡眠态S3,获取第二功耗值。

[0130] 在步骤1103中,将第一功耗值和第二功耗值进行比较。

[0131] 在步骤1104中，若第一功耗值小于等于第二功耗值，则拒绝进入休眠流程。

[0132] 在一种可能的实施方式中，若第一功耗值大于第二功耗值，则进入休眠流程。

[0133] 其中，图11所示的步骤有终端设备中userspace层的功耗比较进程执行。因此当功耗比较进程获取第一功耗值和第二功耗值的比较结果之后，会通知上层系统中休眠流程管理这一进程，从而休眠流程管理这一进程根据功耗比较进程的通知结果决定是进入休眠流程还是拒绝进入休眠流程。因此在本申请实施例中，拒绝进入休眠流程具体包括：通过写第一节点的方式触发底层驱动上报第一事件给上层系统；第一事件用于指示上层系统拒绝进入休眠流程。进入休眠流程具体包括：通过写第二节点的方式触发底层驱动上报第二事件给上层系统；第二事件用于指示上层系统进入休眠流程。

[0134] 因此，拒绝进入休眠流程可具体实施为如图12所示的步骤：

[0135] 在步骤1201中，功耗比较进程将第一功耗值小于等于第二功耗值的结果写入第一节点，并通知给底层驱动；

[0136] 在步骤1202中，底层驱动收到第一节点的变化，上报第一事件给上层系统的休眠流程管理进程；第一事件用于指示休眠流程管理进程拒绝进入休眠流程。

[0137] 在步骤1203中，休眠流程管理进程接收到第一事件，拒绝进入休眠流程，保留在空闲态S0。

[0138] 由此，可以保证即使在休眠流程发起之后即使没有休眠锁阻止休眠，也能停留在空闲态S0，而不进入功耗较大的休眠流程。

[0139] 同时，进入休眠流程可具体实施为如图13所示的步骤：

[0140] 在步骤1301中，功耗比较进程将第一功耗值大于第二功耗值的结果写入第二节点，并通知给底层驱动；

[0141] 在步骤1302中，底层驱动收到第二节点的变化，上报第二事件给上层系统的休眠流程管理进程；第二事件用于指示休眠流程管理进程进入休眠流程。

[0142] 在步骤1303中，休眠流程管理进程接收到第二事件，进入休眠流程，直至处于深睡眠态S3。

[0143] 为了进一步理解，下面结合图14，对本申请实施体提供的终端设备的省电方法的流程进行总体说明。如图14所示，具体包括以下步骤：

[0144] 在步骤1401中，处于空闲态S0时，若接收到进入休眠流程的请求，获取空闲态S0时的第一功耗值。

[0145] 在步骤1402中，判断有无休眠锁。若有休眠锁，则在步骤1405中，拒绝进入休眠流程，系统保留在空闲态S0。若没有休眠锁，则在步骤1403中，读取上一休眠流程中处于深睡眠态S3时的第二功耗值。

[0146] 在步骤1404中，判断第一功耗值是否大于第二功耗值。若第一功耗值小于等于第二功耗值，则在步骤1405中，拒绝进入休眠流程，系统保留在空闲态S0；若第一功耗值大于第二功耗值，则在步骤1406中，系统进入休眠流程。

[0147] 由此，可以通过比较空闲态S0时的第一功耗值和深睡眠态S3时的第二功耗值，决定终端设备处于空闲态S0还是深睡眠态S3。

[0148] 基于前文的描述，本申请实施例通过在处于空闲态S0时，若接收到进入休眠流程的请求且内核层没有休眠锁时，通过比较空闲态S0时的第一功耗值和上一休眠流程中处于

深睡眠态S3时的第二功耗值;若第一功耗值小于等于第二功耗值,则拒绝进入休眠流程,终端设备一直处于空闲态S0,若第一功耗值大于第二功耗值,则进入休眠流程,终端设备处于深睡眠态S3,由此,可以保证在一些因系统资源使用错误或者其它软件出现错误而导致待机耗电大的情况时,不进深睡眠态S3,可以尽可能的降低终端设备的功耗,达到省电的目的。

[0149] 基于相同的发明构思,本申请实施例还提供一种终端设备,如图15所示,包括:显示器1500、处理器180和存储器120,显示器1500用于显示屏幕显示区域,处理器180负责管理总线架构和通常的处理,存储器120可以存储处理器180在执行操作时所使用的数据。

[0150] 其中,在图15中,总线接口可以包括任意数量的互联的总线和桥,具体由处理器180代表的一个或多个处理器180和存储器120代表的存储器120的各种电路链接在一起。总线架构还可以将诸如外围设备、稳压器和功率管理电路等之类的各种其他电路链接在一起,这些都是本领域所公知的,因此,本文不再对其进行进一步描述。总线接口提供接口。可选的,处理器180可以是CPU(中央处埋器)、ASIC(Application Specific Integrated Circuit,专用集成电路)、FPGA(Field－Programmable Gate Array,现场可编程门阵列)或CPLD(Complex Programmable Logic Device,复杂可编程逻辑器件),处理器也可以采用多核架构。

[0151] 处理器180通过调用存储器120存储的计算机程序,用于按照获得的可执行指令执行本申请实施例提供的任一所述方法。处理器180与存储器120也可以物理上分开布置。

[0152] 在此需要说明的是,本发明实施例提供的上述设备,能够实现上述方法实施例所实现的所有方法步骤,且能够达到相同的技术效果,在此不再对本实施例中与方法实施例相同的部分及有益效果进行具体赘述。

[0153] 处理器180用于执行如下步骤:

[0154] 处于空闲态S0时,若接收到进入休眠流程的请求,获取空闲态S0时的第一功耗值;

[0155] 若没有休眠锁,则读取上一休眠流程中处于深睡眠态S3时的第二功耗值;

[0156] 将所述第一功耗值和所述第二功耗值进行比较;

[0157] 若所述第一功耗值小于等于所述第二功耗值,则拒绝进入所述休眠流程。

[0158] 在一种可能的实施方式中,所述处理器180还用于:

[0159] 若所述第一功耗值大于所述第二功耗值,则进入所述休眠流程。

[0160] 在一种可能的实施方式中,所述处理器180还用于:

[0161] 通过写第一节点的方式触发底层驱动上报第一事件给上层系统;所述第一事件用于指示所述上层系统拒绝进入所述休眠流程。

[0162] 在一种可能的实施方式中,所述处理器180还用于:

[0163] 通过写第二节点的方式触发底层驱动上报第二事件给上层系统;所述第二事件用于指示所述上层系统进入所述休眠流程。

[0164] 在一种可能的实施方式中,所述处理器180具体用于:

[0165] 读取电池驱动的内核节点,获得所述空闲态S0的第一功耗值。

[0166] 在一种可能的实施方式中,所述处理器180具体用于:

[0167] 读取所述深睡眠态S3对应的寄存器中存储的功耗值,获得所述第二功耗值。

[0168] 在一种可能的实施方式中,所述处理器180具体用于:

[0169] 当所述上一休眠流程中深睡眠态S3被唤醒时，将所述上一休眠流程中深睡眠态S3的第二功耗值存储在所述寄存器中。

[0170] 在一种可能的实施方式中，所述处理器180还用于：

[0171] 在所述上一休眠流程中深睡眠态S3下，若主动与网络建立连接或因外部中断响应事件触发，则立即唤醒所述上一休眠流程中深睡眠态S3，退出深睡眠态S3，建立链接、发起或响应业务。

[0172] 基于相同的发明构思，本申请实施例还提供一种终端设备的省电装置，如图16所示，包括：

[0173] 第一功耗值获取单元1601，用于处于空闲态S0时，若接收到进入休眠流程的请求，获取空闲态S0时的第一功耗值；

[0174] 第二功耗值获取单元1602，用于若没有休眠锁，则读取上一休眠流程中处于深睡眠态S3时的第二功耗值；

[0175] 功耗值比较单元1603，用于将所述第一功耗值和所述第二功耗值进行比较；

[0176] 休眠流程确定单元1604，用于若所述第一功耗值小于等于所述第二功耗值，则拒绝进入所述休眠流程。

[0177] 在一种可能的实施方式中，所述休眠流程确定单元1604，还用于：

[0178] 若所述第一功耗值大于所述第二功耗值，则进入所述休眠流程。

[0179] 在一种可能的实施方式中，如图17所示，所述装置还包括：

[0180] 第一节点指示单元1701，用于通过写第一节点的方式触发底层驱动上报第一事件给上层系统；所述第一事件用于指示所述上层系统拒绝进入所述休眠流程。

[0181] 在一种可能的实施方式中，如图18所示，所述装置还包括：

[0182] 第二节点指示单元1801，用于通过写第二节点的方式触发底层驱动上报第二事件给上层系统；所述第二事件用于指示所述上层系统进入所述休眠流程。

[0183] 在一种可能的实施方式中，所述第一功耗值获取单元1601，具体用于：

[0184] 读取电池驱动的内核节点，获得所述空闲态S0的第一功耗值。

[0185] 在一种可能的实施方式中，所述第二功耗值获取单元1602，具体用于：

[0186] 读取所述深睡眠态S3对应的寄存器中存储的功耗值，获得所述第二功耗值。

[0187] 在一种可能的实施方式中，如图19所示，所述装置还包括：

[0188] 存储单元1901，用于当所述上一休眠流程中深睡眠态S3被唤醒时，将所述上一休眠流程中深睡眠态S3的第二功耗值存储在所述寄存器中。

[0189] 在一种可能的实施方式中，如图20所示，所述装置还包括：

[0190] 唤醒单元2001，用于在所述上一休眠流程中深睡眠态S3下，若主动与网络建立连接或因外部中断响应事件触发，则立即唤醒所述上一休眠流程中深睡眠态S3，退出深睡眠态S3，建立链接、发起或响应业务。

[0191] 在示例性实施例中，本申请还提供了一种包括指令的计算机可读存储介质，例如包括指令的存储器120，上述指令可由终端设备100的处理器180执行以完成上述终端设备的省电方法。可选地，计算机可读存储介质可以是非临时性计算机可读存储介质，例如，所述非临时性计算机可读存储介质可以是ROM、随机存取存储器(RAM)、CD-ROM、磁带、软盘和光数据存储设备等。

[0192] 在示例性实施例中，还提供一种计算机程序产品，包括计算机程序，所述计算机程序被处理器180执行时实现如本申请提供的终端设备的省电方法。

[0193] 本领域内的技术人员应明白，本申请的实施例可提供为方法、系统、或计算机程序产品。因此，本申请可采用完全硬件实施例、完全软件实施例、或结合软件和硬件方面的实施例的形式。而且，本申请可采用在一个或多个其中包含有计算机可用程序代码的计算机可用存储介质(包括但不限于磁盘存储器、CD-ROM、光学存储器等)上实施的计算机程序产品的形式。

[0194] 本申请是参照根据本申请的方法、设备(系统)、和计算机程序产品的流程图和/或方框图来描述的。应理解可由计算机程序指令实现流程图和/或方框图中的每一流程和/或方框、以及流程图和/或方框图中的流程和/或方框的结合。可提供这些计算机程序指令到通用计算机、专用计算机、嵌入式处理机或其他可编程数据处理设备的处理器以产生一个机器，使得通过计算机或其他可编程数据处理设备的处理器执行的指令产生用于实现在流程图一个流程或多个流程和/或方框图一个方框或多个方框中指定的功能的装置。

[0195] 这些计算机程序指令也可存储在能引导计算机或其他可编程数据处理设备以特定方式工作的计算机可读存储器中，使得存储在该计算机可读存储器中的指令产生包括指令装置的制造品，该指令装置实现在流程图一个流程或多个流程和/或方框图一个方框或多个方框中指定的功能。

[0196] 这些计算机程序指令也可装载到计算机或其他可编程数据处理设备上，使得在计算机或其他可编程设备上执行一系列操作步骤以产生计算机实现的处理，从而在计算机或其他可编程设备上执行的指令提供用于实现在流程图一个流程或多个流程和/或方框图一个方框或多个方框中指定的功能的步骤。

[0197] 显然，本领域的技术人员可以对本申请进行各种改动和变型而不脱离本申请的精神和范围。这样，倘若本申请的这些修改和变型属于本申请权利要求及其等同技术的范围之内，则本申请也意图包含这些改动和变型在内。

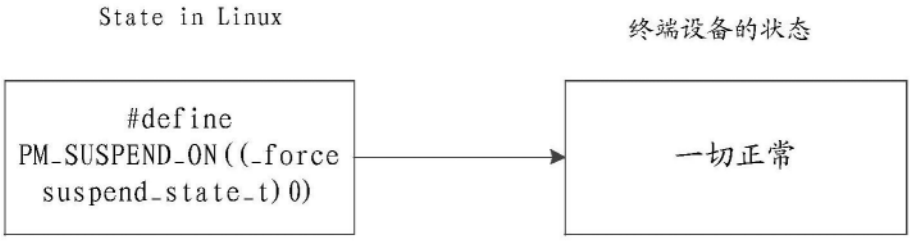

State in Linux
终端设备的状态
#define PM_SUSPEND_ON((_force suspend_state_t)0)
一切正常

图1

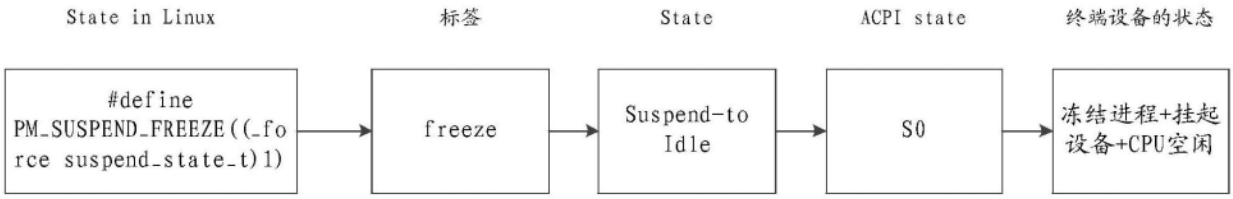

State in Linux
标签
State
ACPI state
终端设备的状态
#define PM_SUSPEND_FREEZE((_force suspend_state_t)1)
freeze
Suspend-to Idle
S0
冻结进程+挂起设备+CPU空闲

图2

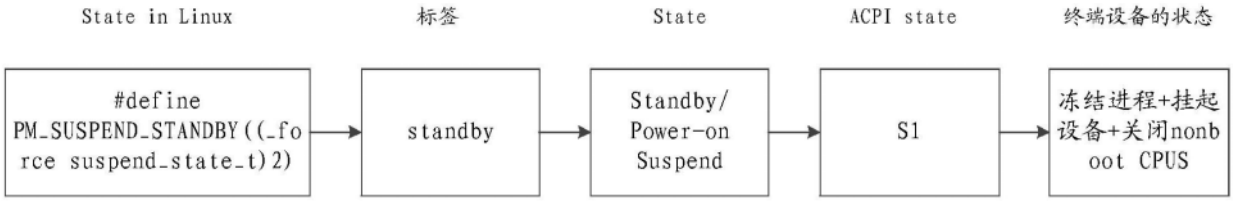

State in Linux
标签
State
ACPI state
终端设备的状态
#define PM_SUSPEND_STANDBY((_force suspend_state_t)2)
standby
Standby/ Power-on Suspend
S1
冻结进程+挂起设备+关闭nonboot CPUS

图3

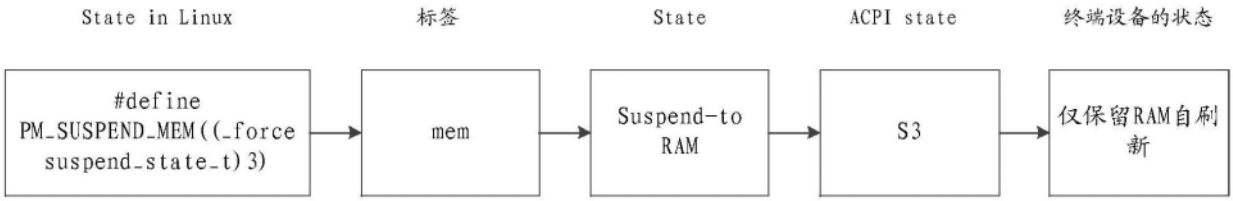

State in Linux
标签
State
ACPI state
终端设备的状态
#define PM_SUSPEND_MEM((_force suspend_state_t)3)
mem
Suspend-to RAM
S3
仅保留RAM自刷新

图4

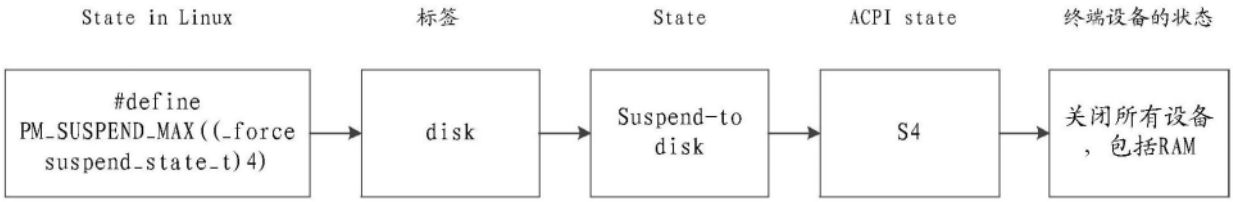

State in Linux
标签
State
ACPI state
终端设备的状态
#define PM_SUSPEND_MAX((_force suspend_state_t)4)
disk
Suspend-to disk
S4
关闭所有设备，包括RAM

图5

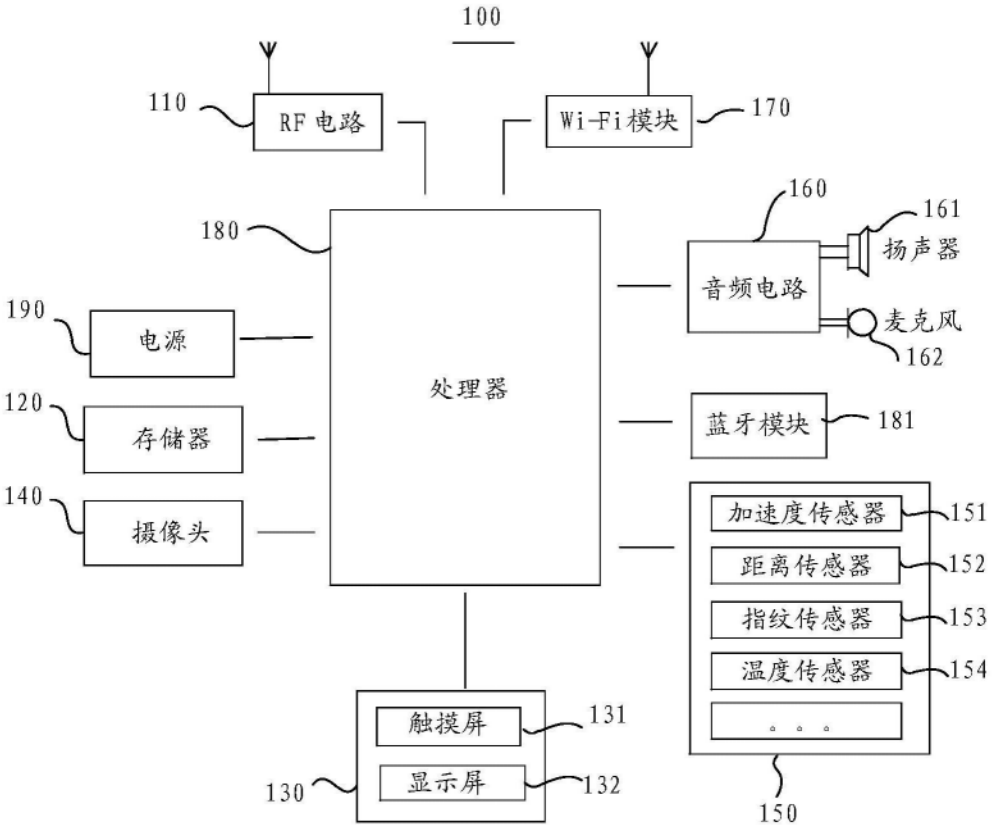

100
110
RF 电路
Wi-Fi模块
170
180
处理器
160
161
扬声器
音频电路
麦克风
162
190
电源
120
存储器
蓝牙模块
181
140
摄像头
加速度传感器
151
距离传感器
152
指纹传感器
153
温度传感器
154
触摸屏
131
显示屏
132
130
150

图6

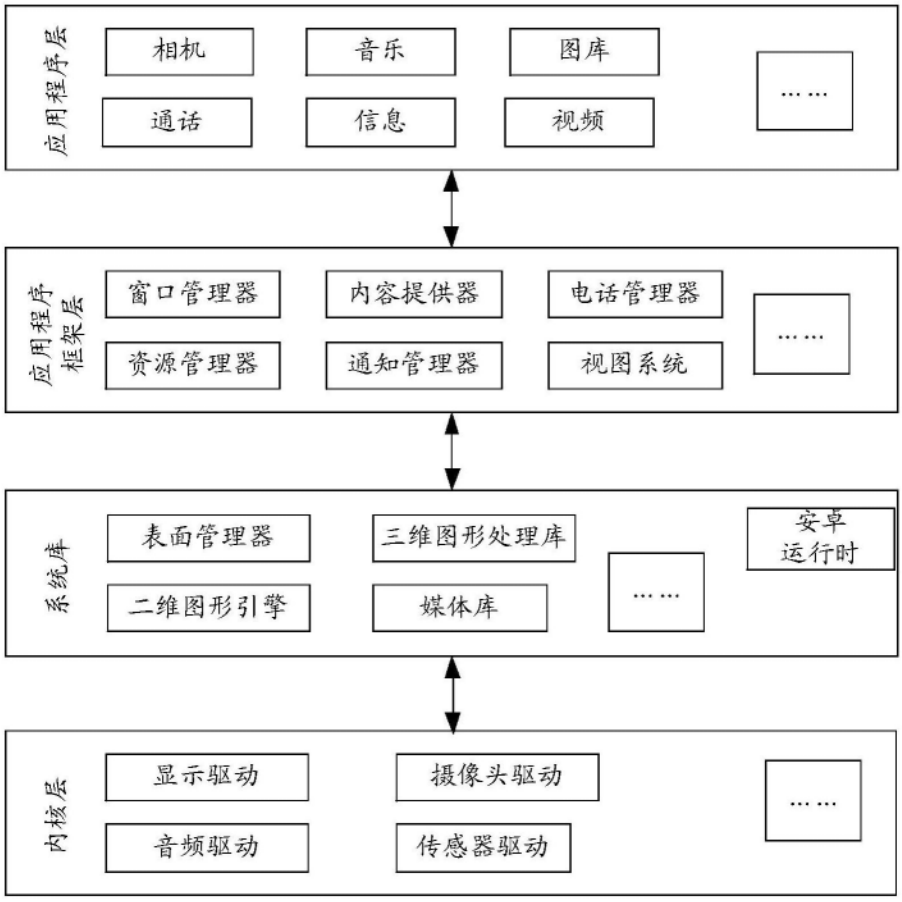

应用程序层
相机
音乐
图库
通话
信息
视频
……
应用程序框架层
窗口管理器
内容提供器
电话管理器
资源管理器
通知管理器
视图系统
……
系统库
表面管理器
三维图形处理库
安卓运行时
二维图形引擎
媒体库
……
内核层
显示驱动
摄像头驱动
音频驱动
传感器驱动
……

图7

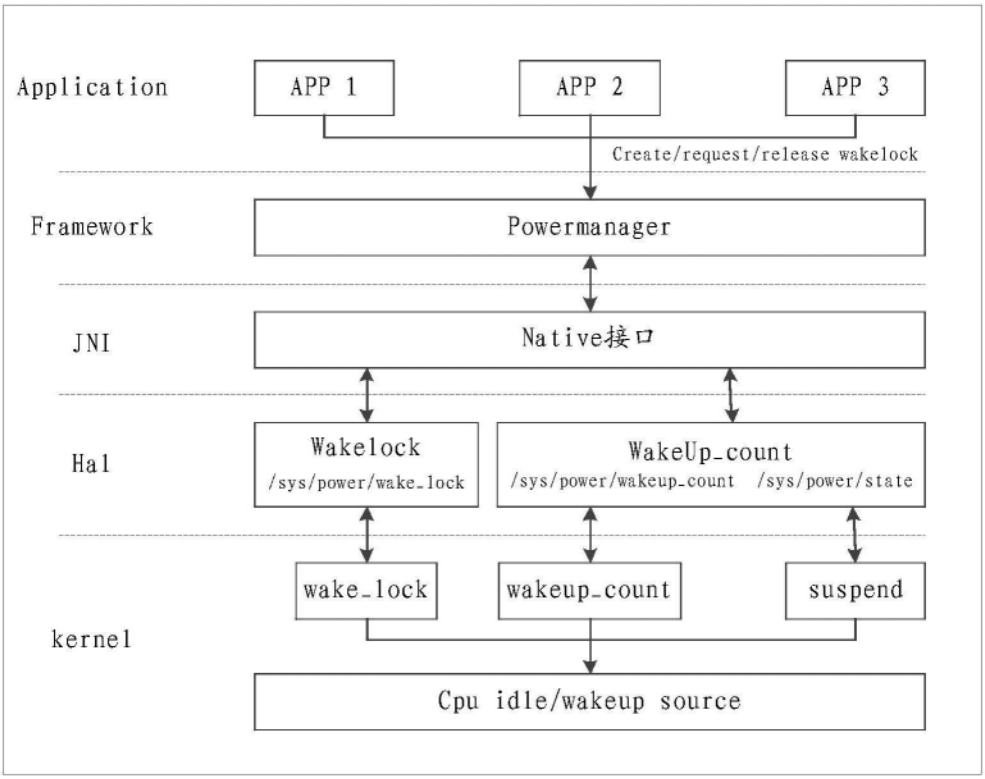

Application
APP 1
APP 2
APP 3
Create/request/release wakelock
Framework
Powermanager
JNI
Native接口
Hal
Wakelock
/sys/power/wake_lock
WakeUp_count
/sys/power/wakeup_count /sys/power/state
kernel
wake_lock
wakeup_count
suspend
Cpu idle/wakeup source

图8

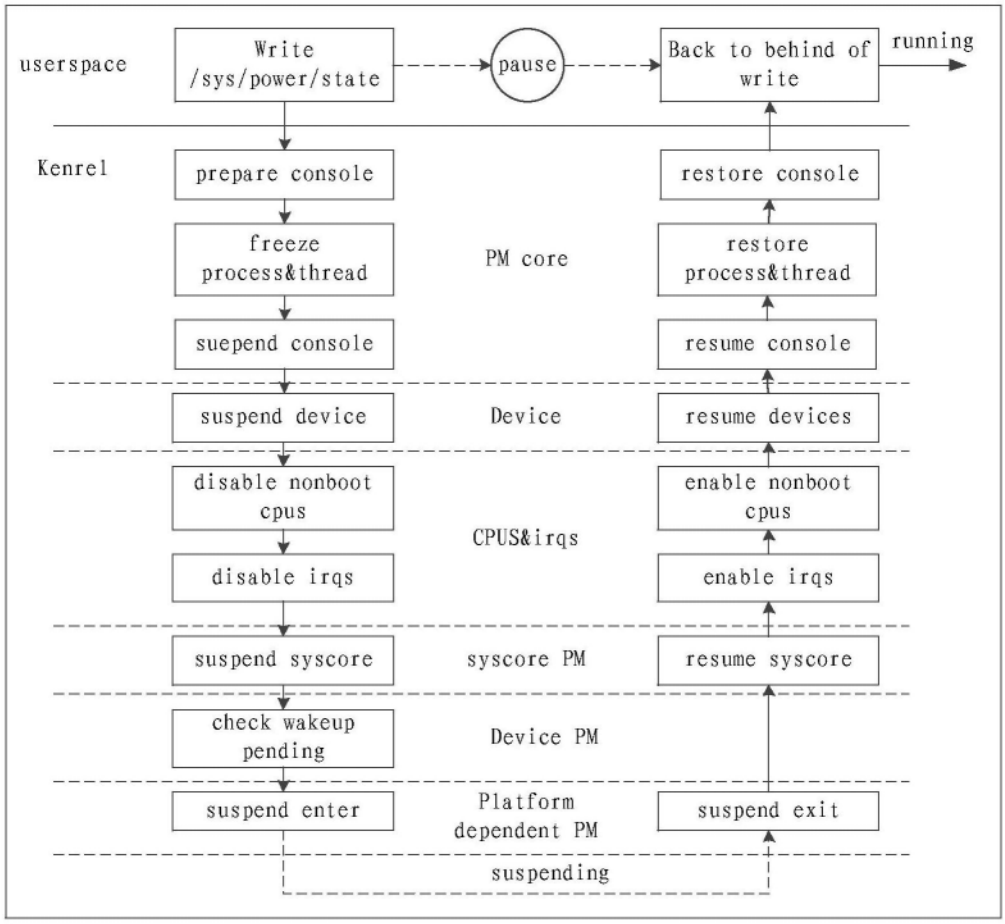

userspace
Write /sys/power/state
pause
Back to behind of write
running
Kenrel
prepare console
freeze process&thread
suepend console
suspend device
disable nonboot cpus
disable irqs
suspend syscore
check wakeup pending
suspend enter
PM core
Device
CPUS&irqs
syscore PM
Device PM
Platform dependent PM
suspending
restore console
restore process&thread
resume console
resume devices
enable nonboot cpus
enable irqs
resume syscore
suspend exit

图9

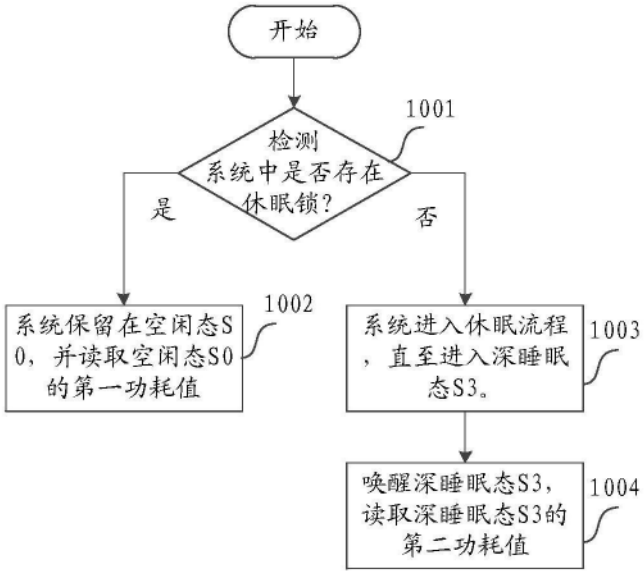

开始
检测系统中是否存在休眠锁?
1001
是
否
系统保留在空闲态S0，并读取空闲态S0的第一功耗值
1002
系统进入休眠流程，直至进入深睡眠态S3。
1003
唤醒深睡眠态S3，读取深睡眠态S3的第二功耗值
1004

图10

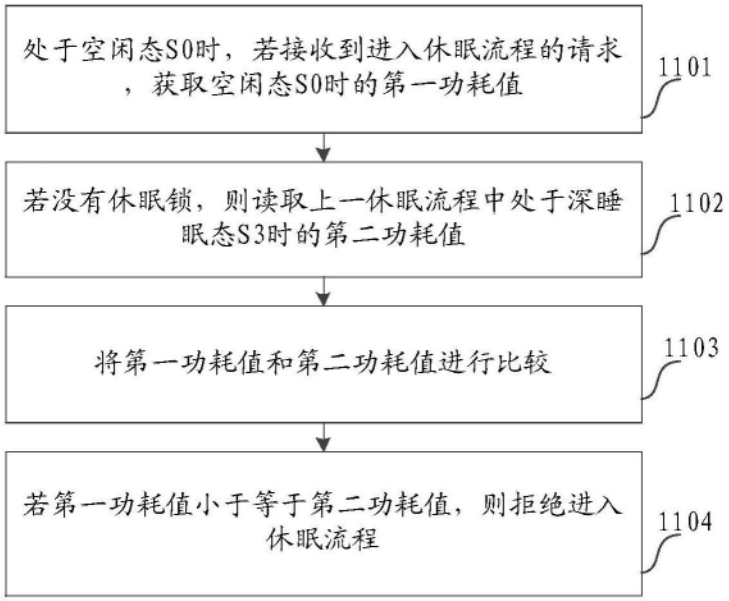

处于空闲态S0时，若接收到进入休眠流程的请求，获取空闲态S0时的第一功耗值
1101
若没有休眠锁，则读取上一休眠流程中处于深睡眠态S3时的第二功耗值
1102
将第一功耗值和第二功耗值进行比较
1103
若第一功耗值小于等于第二功耗值，则拒绝进入休眠流程
1104

图11

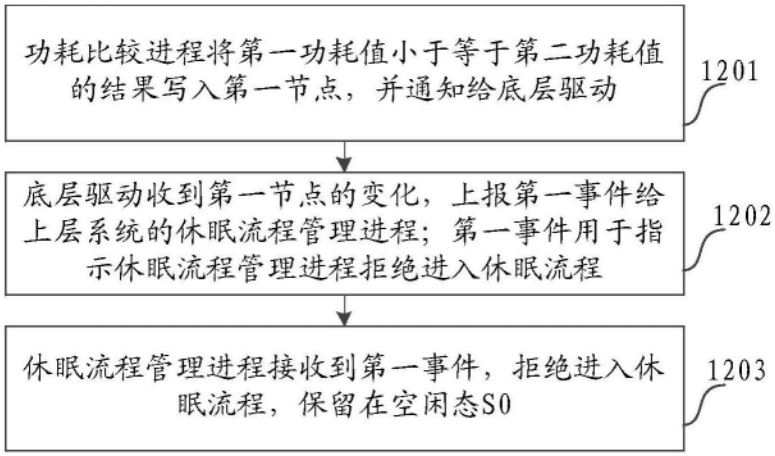

功耗比较进程将第一功耗值小于等于第二功耗值的结果写入第一节点，并通知给底层驱动
1201
底层驱动收到第一节点的变化，上报第一事件给上层系统的休眠流程管理进程；第一事件用于指示休眠流程管理进程拒绝进入休眠流程
1202
休眠流程管理进程接收到第一事件，拒绝进入休眠流程，保留在空闲态S0
1203

图12

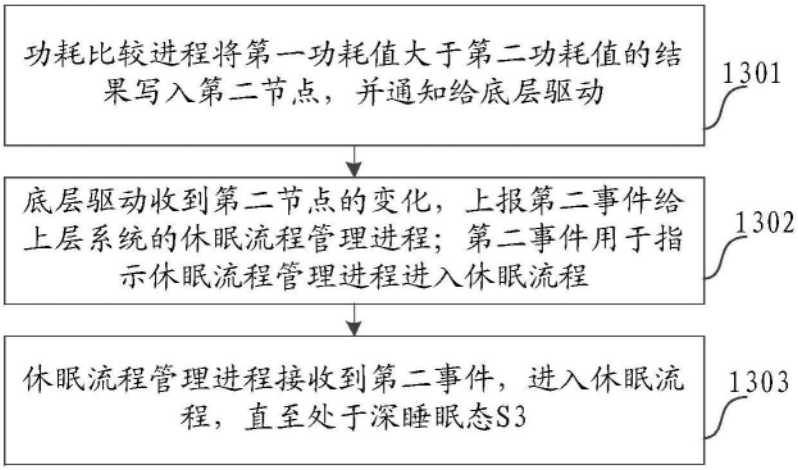

功耗比较进程将第一功耗值大于第二功耗值的结果写入第二节点，并通知给底层驱动
1301
底层驱动收到第二节点的变化，上报第二事件给上层系统的休眠流程管理进程；第二事件用于指示休眠流程管理进程进入休眠流程
1302
休眠流程管理进程接收到第二事件，进入休眠流程，直至处于深睡眠态S3
1303

图13

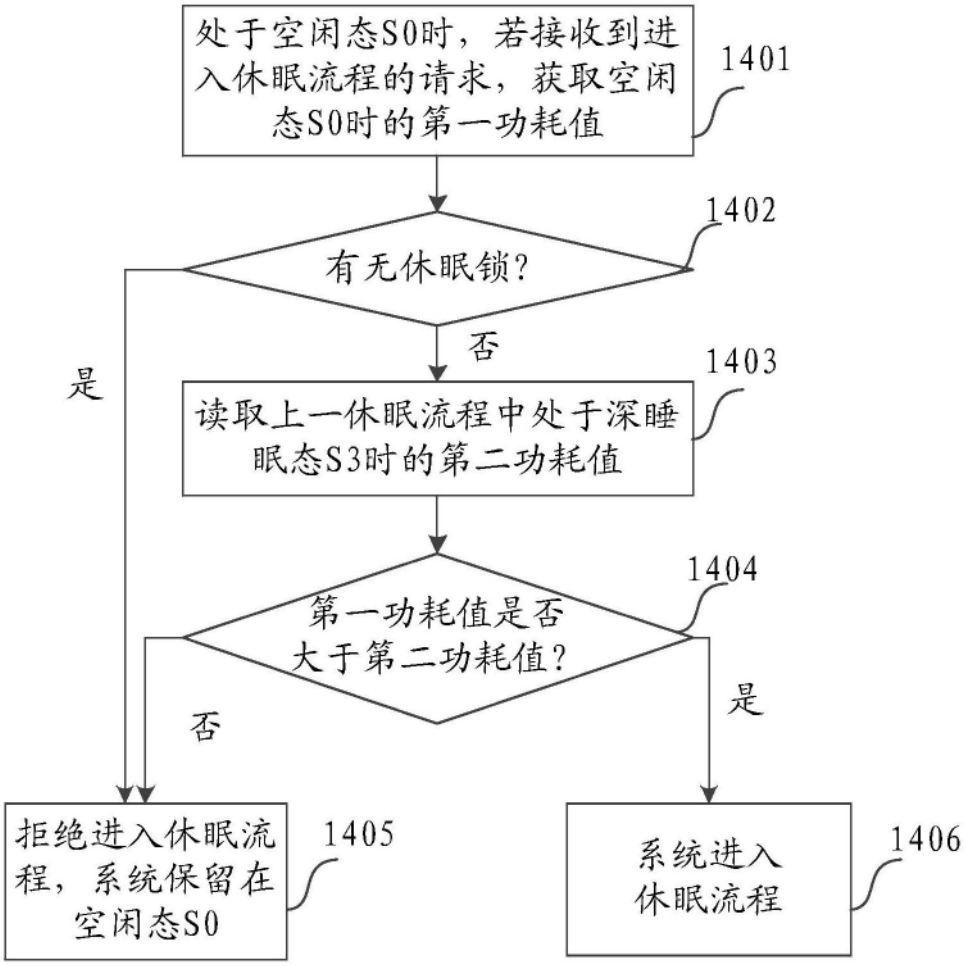

处于空闲态S0时，若接收到进入休眠流程的请求，获取空闲态S0时的第一功耗值
1401
有无休眠锁？
1402
是
否
读取上一休眠流程中处于深睡眠态S3时的第二功耗值
1403
第一功耗值是否大于第二功耗值？
1404
否
是
拒绝进入休眠流程，系统保留在空闲态S0
1405
系统进入休眠流程
1406

图14

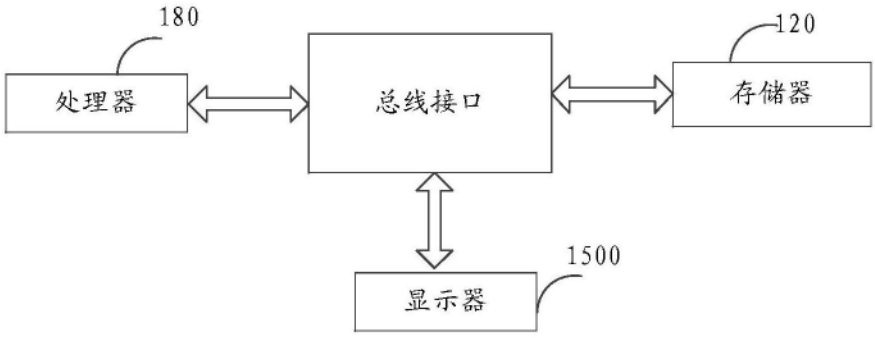

180
处理器
总线接口
120
存储器
1500
显示器

图15

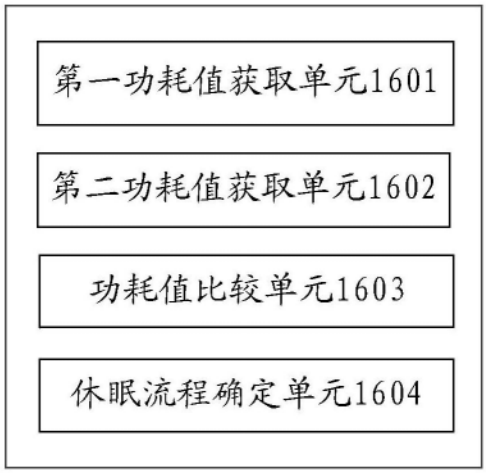

第一功耗值获取单元1601
第二功耗值获取单元1602
功耗值比较单元1603
休眠流程确定单元1604

图16

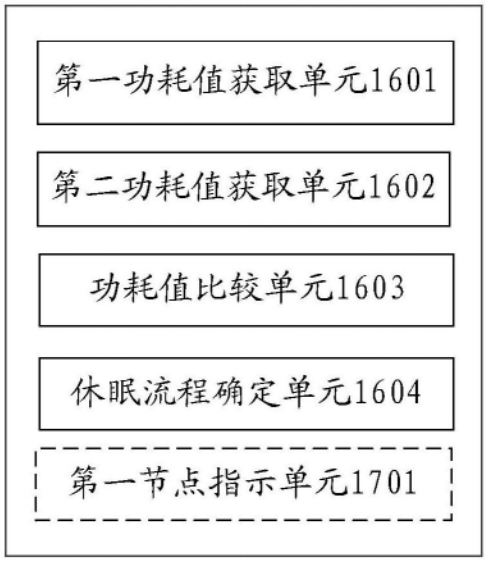

第一功耗值获取单元1601
第二功耗值获取单元1602
功耗值比较单元1603
休眠流程确定单元1604
第一节点指示单元1701

图17

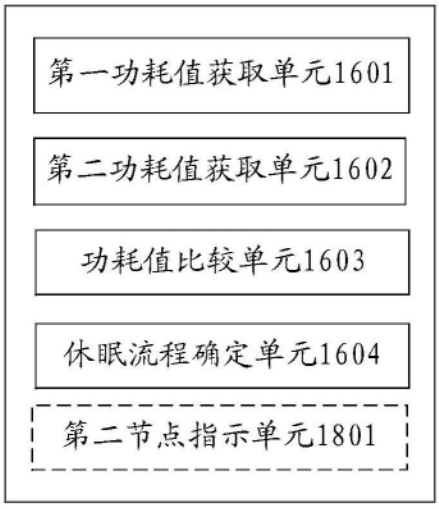

第一功耗值获取单元1601
第二功耗值获取单元1602
功耗值比较单元1603
休眠流程确定单元1604
第二节点指示单元1801

图18

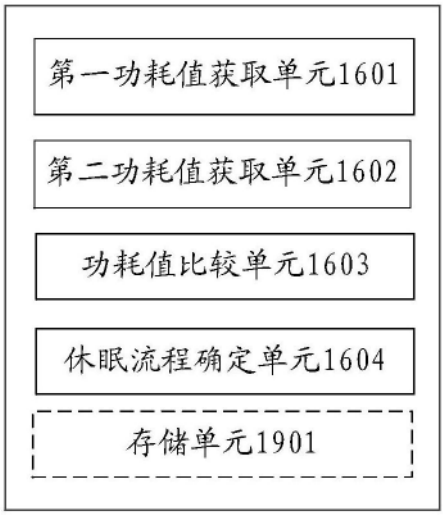

第一功耗值获取单元1601
第二功耗值获取单元1602
功耗值比较单元1603
休眠流程确定单元1604
存储单元1901

图19

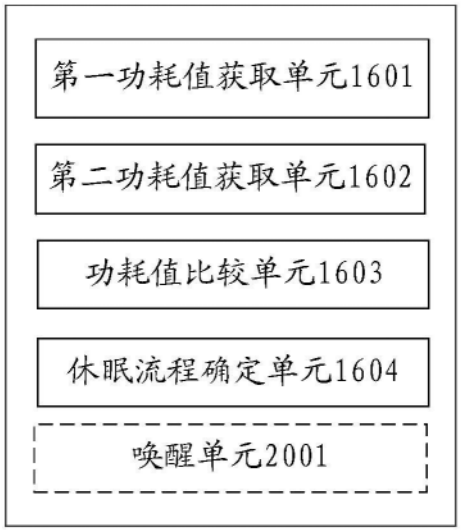

第一功耗值获取单元1601
第二功耗值获取单元1602
功耗值比较单元1603
休眠流程确定单元1604
唤醒单元2001

图20