(19)国家知识产权局

(12)发明专利申请

(10)申请公布号 CN 116414467 A
(43)申请公布日 2023.07.11

(21)申请号 202111650261.4

(22)申请日 2021.12.30

(71)申请人 北京君正集成电路股份有限公司
地址 100193 北京市海淀区西北旺东路10号院东区14号楼

(72)发明人 刘思慧

(74)专利代理机构 北京嘉东律师事务所 11788
专利代理师 田欣欣

(51)Int.Cl.
G06F 9/4401(2018.01)
G06F 1/3296(2019.01)
G06F 1/26(2006.01)

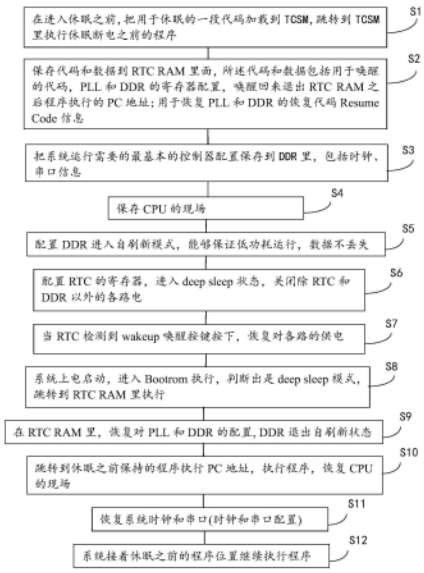
权利要求书2页 说明书5页 附图2页

(54)发明名称

一种低功耗快速唤醒的方法

(57)摘要

本发明提供一种低功耗快速唤醒的方法，对hibernate等级的休眠进行了设计，结合sleep和hibernate的优势，实现了休眠的超低功耗，以及快速唤醒功能。所述方法包括：利用Linux系统提供的休眠流程，使其在休眠时，只有RTC域和DDR域保持供电，并且DDR进入低功耗自刷新状态，其他电源域全部断电；休眠前保存PLL和DDR的配置，系统上电启动时，直接跳到RTC RAM里实现唤醒功能，恢复PLL和DDR的配置，加快唤醒速度；由于DDR进入自刷新模式，里面的数据能够保存不丢失，程序唤醒回来能够继续接着休眠前的位置继续运行，系统不需要重新启动。



CN 116414467 A

1.一种低功耗快速唤醒的方法,其特征在于,所述方法包括:利用应用系统提供的休眠唤醒框架,使其在休眠时,只有RTC域和DDR域保持供电,并且DDR进入低功耗自刷新状态,其他电源域全部断电;休眠前保存PLL和DDR的配置,系统上电启动时,直接跳到RTC RAM里实现唤醒功能,即CPU上电启动的时候,进入Bootrom,在Bootrom里面读取一个用来表示本次复位产生原因的只读寄存器,从而判断上一次断电是不是深度休眠;如果是深度休眠,CPU执行跳转指令,跳到RTC RAM中恢复代码的地址,执行唤醒代码,恢复PLL和DDR的配置,加快唤醒速度;如果不是深度休眠,则走正常启动流程;由于DDR进入自刷新模式,里面的数据能够保存不丢失,程序唤醒回来能够继续接着休眠前的位置继续运行,系统不需要重新启动。

2.根据权利要求1所述的一种低功耗快速唤醒的方法,其特征在于,所述方法的步骤,还包括:

准备开始进入深度休眠;

保存DDR、PLL配置:其中,在RTC RAM中保存包括PLL和DDR配置的DATA、包括恢复PLL和DDR的恢复代码Resume Code;当接收到唤醒源时,启动执行Bootrom,调用恢复代码,回到恢复现场;

保存CPU现场;

进行DDR自刷新;

进入Hibernate状态并断电;

恢复现场;

程序继续执行。

3.根据权利要求2所述的一种低功耗快速唤醒的方法,其特征在于,所述方法进一步包括:

S1.在进入深度休眠deep sleep之前,把用于休眠的一段代码加载到TCSM,跳转到TCSM里执行休眠断电之前的程序;

所述用于休眠的代码、休眠断电之前的程序均是指执行休眠流程,配置系统进入深度休眠模式的代码,包括配置DDR进入自刷新的代码,配置RTC进入deep sleep休眠模式的代码;

所述跳转到TCSM是指,程序的PC指针跳转到TCSM的地址上,执行TCSM里面的代码;

S2.保存代码和数据到RTC RAM里面,所述代码和数据包括用于唤醒的代码,PLL和DDR的寄存器配置,唤醒回来退出RTC RAM之后程序执行的PC地址;用于恢复PLL和DDR的恢复代码Resume Code信息;

S3.把系统运行需要的最基本的控制器配置保存到DDR里,包括时钟、串口信息,时钟和串口模块的寄存器配置;

S4.保存CPU的现场;所述CPU的现场是指CPU内部的通用寄存器和cp0寄存器当前的值,cp0指协处理器0;

S5.配置DDR进入自刷新模式,能够保证低功耗运行,数据不丢失;

S6.配置RTC的寄存器,关闭除RTC和DDR以外的各路电,进入deepsleep状态;

S7.当RTC检测到唤醒按键WKUP被按下,则恢复对各路的供电;

所述检测是通过RTC内部逻辑实现的,wkup_n引脚的电平从高到低再变高就唤醒;

S8.系统上电启动,进入Bootrom执行,判断出是deep sleep模式,跳转到RTC RAM里执

行唤醒程序;即在Bootrom里面读一个寄存器的值,这个值表示reset的状态,如果是deep sleep,pc指针就跳转到RTCRAM的地址上执行唤醒程序;如果是其他的reset的状态,就走正常启动流程;

S9.在RTC RAM里,恢复对PLL和DDR的配置,DDR退出自刷新状态;

S10.跳转到休眠之前保存的程序执行PC地址,执行程序,恢复CPU的现场:在这个PC地址上执行的程序的功能是恢复CPU的现场;恢复的方法是:读出休眠前保存的CPU通用寄存器和cp0寄存器的值,写回到CPU通用寄存器和cp0寄存器里;

S11.恢复系统时钟和串口,包括时钟和串口配置;

S12.系统接着休眠之前的程序位置继续执行程序。

4.根据权利要求3所述的一种低功耗快速唤醒的方法,其特征在于,所述步骤S2中所述PC地址所对应存放的程序用于恢复CPU的现场。

5.根据权利要求3所述的一种低功耗快速唤醒的方法,其特征在于,所述步骤S8中所述其他的reset的状态,包括上电复位,看门狗复位。

6.根据权利要求1所述的一种低功耗快速唤醒的方法,其特征在于,所述应用系统包括Linux。

7.根据权利要求1所述的一种低功耗快速唤醒的方法,其特征在于,所述休眠唤醒框架的主要流程是:冻结用户态进程和内核态任务,调用注册的设备的suspend的回调函数,休眠核心设备和使CPU进入休眠态,所述唤醒则是休眠的反过程。

8.根据权利要求1所述的一种低功耗快速唤醒的方法,其特征在于,所述保存PLL和DDR配置的方法:读取PLL和DDR的寄存器,把读出来的值写到RTC RAM里的一个地址上。

9.根据权利要求1所述的一种低功耗快速唤醒的方法,其特征在于,所述能够继续接着休眠前的位置继续运行的方法是:在休眠前,保存了CPU的寄存器值,DDR里面的数据也保持不变;唤醒回来,把保存的CPU寄存器的值重新写回,程序继续运行。

一种低功耗快速唤醒的方法

技术领域

[0001] 本发明涉及系统的电源管理技术领域,特别涉及一种低功耗快速唤醒的方法。

背景技术

[0002] 随着科技的发展,涉及系统的休眠与唤醒技术也日益被人们所重视。现有的Linux系统的sleep休眠唤醒:使用Linux内核提供的休眠唤醒,在休眠时,关闭外部时钟,DDR进入自刷新模式,关闭CPU内核core的电,其他电不关。唤醒后程序接着休眠的地方继续执行。Linux系统的hibernate休眠:在hibernate休眠的时候,开发板只保留RTC域的电,其他部分全部断电。唤醒后重新上电程序重新开始执行。其中,所述开发板是用来进行嵌入式系统开发的电路板。

[0003] 唤醒后程序接着休眠的地方继续执行:保存现场-->休-->醒-->恢复现场,像什么都没发生一样,程序接着休之前的位置继续往下执行。

[0004] 唤醒后,重新上电,程序重新开始执行:指从Bootrom开始,走正常启动流程。

[0005] 但是在现有技术中,休眠唤醒的休眠深度和唤醒速度有关系,休眠深度越浅,唤醒越快,功耗越高;休眠深度越深,唤醒速度越慢,功耗越低。对应地,Linux系统的sleep休眠唤醒:只关闭了一部分电,休眠的功耗比较高。Linux系统的hibernate休眠:唤醒相当于系统重新启动,唤醒速度比较慢,系统不能接着休眠之前的位置继续运行。

[0006] 此外,现有技术中的常用术语如下:

[0007] 1.休眠唤醒:Linux系统提供的电源管理方式。

[0008] 2.idle,sleep:Linux电源管理提供的休眠模式。

[0009] 3.hibernate:Linux电源管理提供的休眠模式。

[0010] 4.RTC:real time clock,实时时钟。

[0011] 5.RTC RAM:RTC域中的一段RAM空间,用于在deep sleep的时候保存代码和数据。

[0012] 6.DDR:双倍速率动态随机存储器。

[0013] 7.PLL(Phase Locked Loop):为锁相回路或锁相环,用来统一整合时钟信号,使高频器件正常工作,如内存的存取资料等。

[0014] 8.DDR自刷新:DDR内部每隔64ms发出一次刷新命令,不需要控制器参与,保证数据不丢失。

[0015] 9.TCSM:一段SRAM空间,在本发明中用于保存并运行休眠时的代码。休眠时的代码放到了TCSM里面,DDR进入自刷新之后是不可以访问的。DDR进入自刷新之后,休眠之前,还有代码要执行,所以要把DDR进入自刷新和进入自刷新之后的休眠代码都放到了TCSM里。也可以把全部的休眠代码到放到TCSM里面,现在的做法是全部放到TCSM里面了。

[0016] 10.Bootrom:固化在芯片里面的启动代码。

发明内容

[0017] 为了解决上述问题,本方法的目的在于:本发明提供了一种深度休眠deep sleep,

是利用Linux系统提供的sleep休眠流程,对hibernate等级的休眠进行了软件和硬件的设计,结合sleep和hibernate的优势,实现了休眠的超低功耗,以及快速唤醒功能。

[0018] 具体地,本发明提供一种低功耗快速唤醒的方法,所述方法包括:利用应用系统提供的休眠唤醒框架,使其在休眠时,只有RTC域和DDR域保持供电,并且DDR进入低功耗自刷新状态,其他电源域全部断电;休眠前保存PLL和DDR的配置,系统上电启动时,直接跳到RTC RAM里实现唤醒功能,即CPU上电启动的时候,进入Bootrom,在Bootrom里面读取一个寄存器,所述寄存器是个只读寄存器,用来表示本次复位是由什么原因产生的,包括deep sleep的复位,上电复位,看门狗复位等,从而判断上一次断电是不是深度休眠;如果是深度休眠,CPU执行跳转指令,跳到RTC RAM中恢复代码的地址,执行唤醒代码,恢复PLL和DDR的配置,加快唤醒速度;如果不是深度休眠,则走正常启动流程;由于DDR进入自刷新模式,里面的数据能够保存不丢失,程序唤醒回来能够继续接着休眠前的位置继续运行,系统不需要重新启动。

[0019] 所述方法的步骤,还包括:

[0020] 准备开始进入深度休眠;

[0021] 保存DDR、PLL配置:其中,在RTC RAM中保存包括PLL和DDR配置的DATA、包括恢复PLL和DDR的恢复代码Resume Code;当接收到唤醒源时,启动执行Bootrom,调用恢复代码,回到恢复现场;

[0022] 保存CPU现场;

[0023] 进行DDR自刷新;

[0024] 进入Hibernate状态并断电;

[0025] 恢复现场;

[0026] 程序继续执行。

[0027] 所述方法进一步包括:

[0028] S1.在进入深度休眠deep sleep之前,把用于休眠的一段代码加载到TCSM,跳转到TCSM里执行休眠断电之前的程序;

[0029] 所述用于休眠的代码、休眠断电之前的程序均是指执行休眠流程,配置系统进入深度休眠模式的代码,包括配置DDR进入自刷新的代码,配置RTC进入deep sleep休眠模式的代码;

[0030] 所述跳转到TCSM是指,程序的PC指针跳转到TCSM的地址上,执行TCSM里面的代码;

[0031] S2.保存代码和数据到RTC RAM里面,所述代码和数据包括用于唤醒的代码,PLL和DDR的寄存器配置,唤醒回来退出RTC RAM之后程序执行的PC地址;用于恢复PLL和DDR的恢复代码Resume Code信息;

[0032] S3.把系统运行需要的最基本的控制器配置保存到DDR里,包括时钟、串口信息,时钟和串口模块的寄存器配置;

[0033] S4.保存CPU的现场;所述CPU的现场是指CPU内部的通用寄存器和cp0寄存器当前的值,cp0指协处理器0;

[0034] S5.配置DDR进入自刷新模式,能够保证低功耗运行,数据不丢失;

[0035] S6.配置RTC的寄存器,关闭除RTC和DDR以外的各路电,进入deep sleep状态;

[0036] S7.当RTC检测到唤醒按键WKUP被按下,则恢复对各路的供电;

[0037] 所述检测是通过RTC内部逻辑实现的,wkup_n引脚的电平从高到低再变高就唤醒;

[0038] S8.系统上电启动,进入Bootrom执行,判断出是deep sleep模式,跳转到RTC RAM里执行唤醒程序;即在Bootrom里面读一个寄存器的值,这个值表示reset的状态,如果是deep sleep,pc指针就跳转到RTC RAM的地址上执行唤醒程序;如果是其他的reset的状态,就走正常启动流程;S9.在RTC RAM里,恢复对PLL和DDR的配置,DDR退出自刷新状态;S10.跳转到休眠之前保存的程序执行PC地址,执行程序,恢复CPU的现场:在这个PC地址上执行的程序的功能是恢复CPU的现场;恢复的方法是:读出休眠前保存的CPU通用寄存器和cp0寄存器的值,写回到CPU通用寄存器和cp0寄存器里;

[0039] S11.恢复系统时钟和串口,包括时钟和串口配置;

[0040] S12.系统接着休眠之前的程序位置继续执行程序。

[0041] 所述步骤S2中所述PC地址所对应存放的程序用于恢复CPU的现场。

[0042] 所述步骤S8中所述其他的reset的状态,包括上电复位,看门狗复位。

[0043] 所述应用系统包括Linux。

[0044] 所述休眠唤醒框架的主要流程是:冻结用户态进程和内核态任务,调用注册的设备的suspend的回调函数,休眠核心设备和使CPU进入休眠态,所述唤醒则是休眠的反过程。

[0045] 所述保存PLL和DDR配置的方法:读取PLL和DDR的寄存器,把读出来的值写到RTC RAM里的一个地址上。

[0046] 所述能够继续接着休眠前的位置继续运行的方法是:在休眠前,保存了CPU的寄存器值,DDR里面的数据也保持不变;唤醒回来,把保存的CPU寄存器的值重新写回,程序继续运行。

[0047] 由此,本申请的优势在于:结合了hibernate休眠的低功耗和sleep的唤醒速度,对于软件和硬件做了设计,方法简单,设计巧妙。

附图说明

[0048] 此处所说明的附图用来提供对本发明的进一步理解,构成本申请的一部分,并不构成对本发明的限定。

[0049] 图1是本申请涉及方法的框图示意图。

[0050] 图2是本发明方法的流程示意图。

具体实施方式

[0051] 为了能够更清楚地理解本发明的技术内容及优点,现结合附图对本发明进行进一步的详细说明。

[0052] 本实施例是一种低功耗快速唤醒的方法,结合了hibernate的低功耗和sleep的唤醒速度,做了软件和硬件的设计。在休眠时,只有RTC域和DDR域保持供电,并且DDR进入低功耗自刷新状态,其他电源域全部断电,降低了功耗。休眠前保存了PLL和DDR的配置,系统上电启动时,直接跳到RTC RAM里实现唤醒功能,恢复PLL和DDR的配置,加快唤醒速度。由于DDR进入自刷新模式,里面的数据能够保存不丢失,程序唤醒回来能够继续接着休眠前的位置继续运行,系统不需要重新启动。所述方法包括:

[0053] 利用linux系统提供的休眠唤醒框架,休眠的主要流程是:冻结用户态进程和内核

态任务，调用注册的设备的suspend的回调函数，休眠核心设备和使CPU进入休眠态；唤醒是休眠的反过程；使其在休眠时，只有RTC域和DDR域保持供电，并且DDR进入低功耗自刷新状态，其他电源域全部断电；休眠前保存PLL和DDR的配置，保存PLL和DDR配置的方法：读取PLL和DDR的寄存器，把读出来的值写到RTC RAM里的某个地址上；

[0054] 系统上电启动时，直接跳到RTC RAM里实现唤醒功能，恢复PLL和DDR的配置，加快唤醒速度；所述跳到RTC RAM是指：CPU上电启动的时候，进入Bootrom，在Bootrom里面读取RTC里面的一个寄存器，判断上一次断电是不是deep sleep：如果是deep sleep，CPU执行跳转指令，跳到RTC RAM的地址上，执行唤醒代码；如果不是deep sleep，走正常启动流程；由于DDR进入自刷新模式，里面的数据能够保存不丢失，程序唤醒回来能够继续接着休眠前的位置继续运行，系统不需要重新启动；所述程序接着休眠前的位置继续运行的方法是：在休眠前，保存了cpu的寄存器值，ddr里面的数据也保持不变。唤醒回来，把保存的cpu寄存器的值重新写回，程序就继续运行了，像什么都没发生一样。

[0055] 如图1所示，所述方法的步骤包括：

[0056] 准备开始进入深度休眠；

[0057] 保存DDR、PLL配置：其中，在RTC RAM中保存包括PLL和DDR配置的DATA、包括恢复PLL和DDR的恢复代码Resume Code；当接收到唤醒源时，启动执行Bootrom，调用恢复代码，回到恢复现场；

[0058] 保存CPU现场；

[0059] 进行DDR自刷新；

[0060] 进入Hibernate状态并断电；

[0061] 恢复现场；

[0062] 程序继续执行。

[0063] 具体的流程如下，所述方法进一步包括，如图2所示：

[0064] S1.在进入休眠之前，把用于休眠的一段代码加载到TCSM，跳转到TCSM里执行休眠断电之前的程序；

[0065] 所述用于休眠的代码、休眠断电之前的程序均是指执行休眠流程，配置系统进入deep sleep模式的代码，包括配置DDR进入自刷新的代码，配置RTC进入deep sleep休眠模式的代码等；

[0066] 所述跳转到TCSM是指程序的PC指针跳转到TCSM的地址上，执行TCSM里面的代码；

[0067] S2.保存代码和数据到RTC RAM里面，所述代码和数据包括用于唤醒的代码，PLL和DDR的寄存器配置，唤醒回来退出RTC RAM之后程序执行的PC地址；用于恢复PLL和DDR的恢复代码Resume Code信息；

[0068] S3.把系统运行需要的最基本的控制器配置保存到DDR里，包括时钟、串口信息；时钟和串口等模块的寄存器配置；

[0069] S4.保存CPU的现场；所述CPU的现场是指CPU内部的通用寄存器和cp0寄存器当前的值；cp0指协处理器0；

[0070] S5.配置DDR进入自刷新模式，能够保证低功耗运行，数据不丢失；

[0071] S6.配置RTC的寄存器，关闭除RTC和DDR以外的各路电，进入deep sleep状态；

[0072] S7.当RTC检测到唤醒按键WKUP按下，恢复对各路的供电；

[0073] 所述RTC检测是RTC内部逻辑实现的,不是软件的操作。wkup_n的电平从高到低再变高就唤醒;

[0074] S8.系统上电启动,进入Bootrom执行,判断出是deep sleep模式,跳转到RTC RAM里执行;具体地判断方法:在Bootrom里面读一个寄存器的值。这个值表示具体是什么reset的状态。如果是deep sleep,pc指针就跳转到RTC RAM的地址上执行唤醒程序。如果是其他的reset,包括上电复位,看门狗复位等,就走正常启动流程。

[0075] S9.在RTC RAM里,恢复对PLL和DDR的配置,DDR退出自刷新状态;S10.跳转到休眠之前保存的程序执行PC地址,执行程序,恢复CPU的现场:在这个PC地址上执行的程序的主要功能是恢复CPU的现场;恢复的方法是:读出休眠前保存的CPU通用寄存器和cp0寄存器的值,写回到CPU通用寄存器和cp0寄存器里;

[0076] S11.恢复系统时钟和串口,包括时钟和串口配置;

[0077] S12.系统接着休眠之前的程序位置继续执行程序。

[0078] 所述步骤S2中的PC地址对应存放的程序用于恢复CPU的现场,即resume pc的值是一个地址,这个地址上放着程序,程序的功能是恢复CPU现场。

[0079] 所述应用系统不一定是Linux系统,其他系统也可以,只是本实施例只在Linux上进行操作。

[0080] 以上所述仅为本发明的优选实施例而已,并不用于限制本发明,对于本领域的技术人员来说,本发明实施例可以有各种更改和变化。凡在本发明的精神和原则之内,所作的任何修改、等同替换、改进等,均应包含在本发明的保护范围之内。

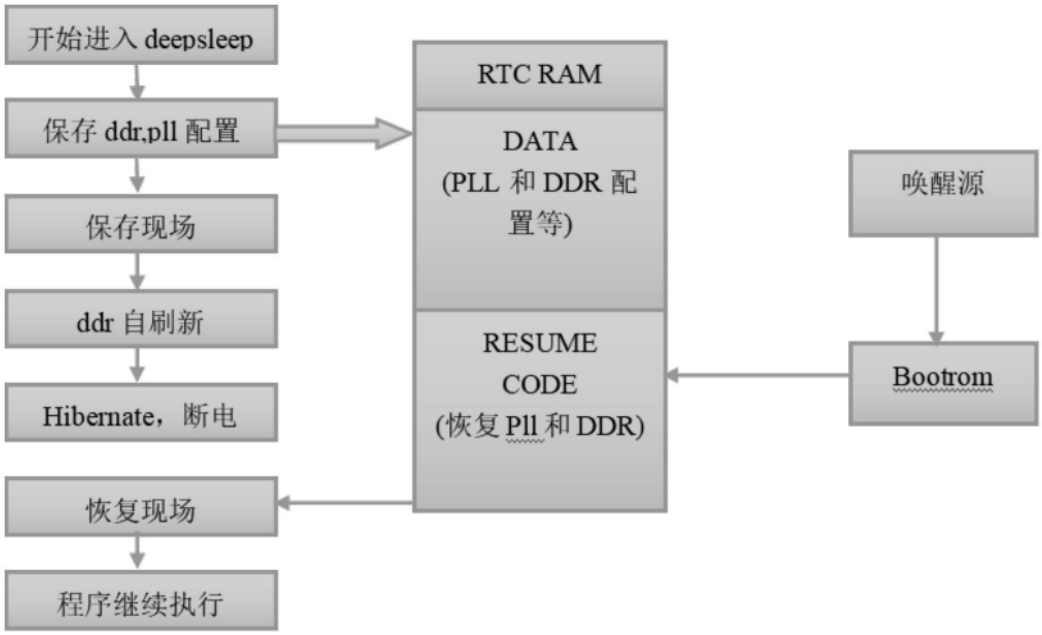

开始进入 deepsleep
保存 ddr,pll 配置
保存现场
ddr 自刷新
Hibernate，断电
恢复现场
程序继续执行
RTC RAM
DATA
(PLL 和 DDR 配置等)
RESUME
CODE
(恢复 Pll 和 DDR)
唤醒源
Bootrom

图1

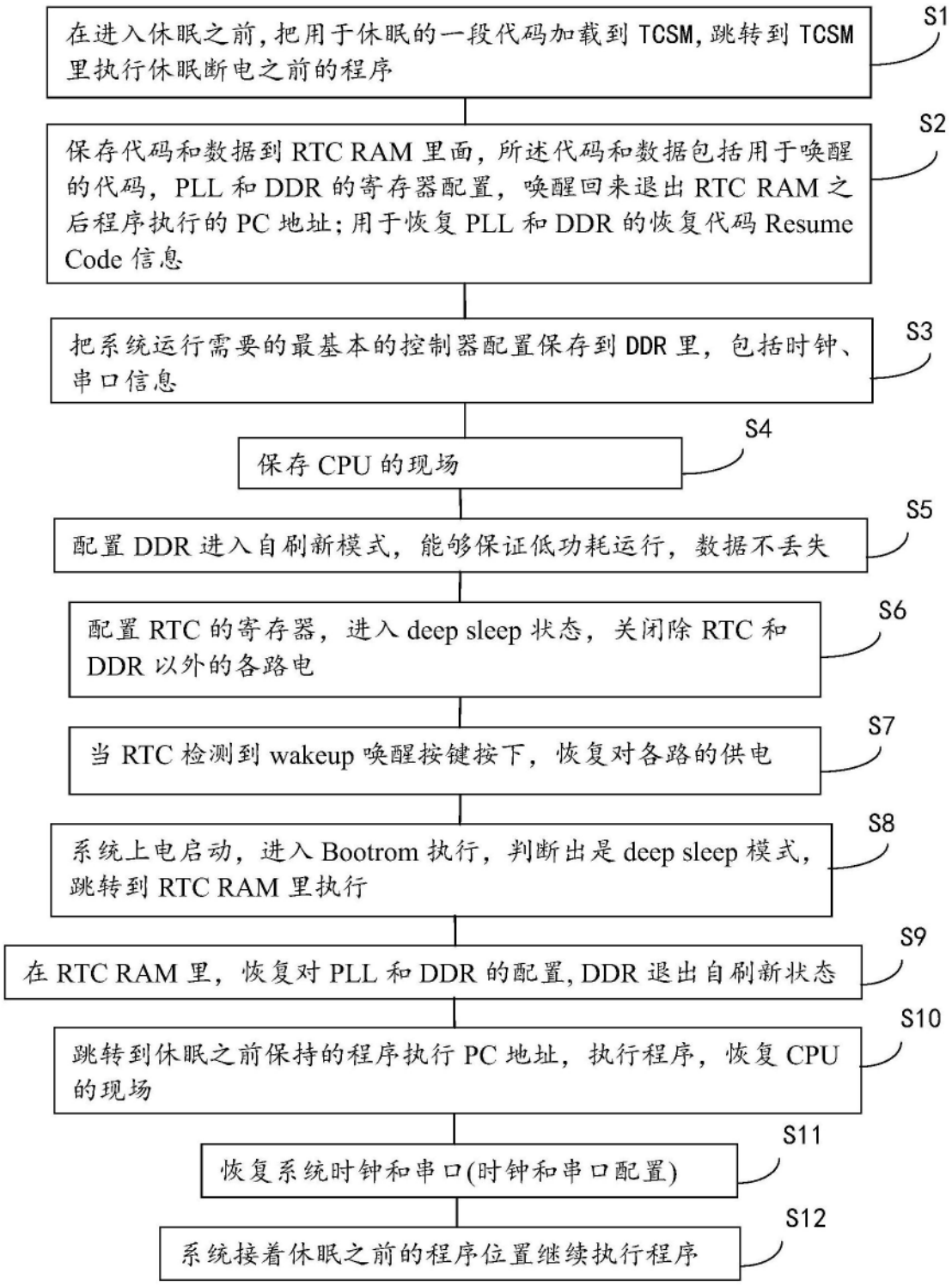

在进入休眠之前，把用于休眠的一段代码加载到 TCSM，跳转到 TCSM 里执行休眠断电之前的程序
S1
保存代码和数据到 RTC RAM 里面，所述代码和数据包括用于唤醒的代码，PLL 和 DDR 的寄存器配置，唤醒回来退出 RTC RAM 之后程序执行的 PC 地址；用于恢复 PLL 和 DDR 的恢复代码 Resume Code 信息
S2
把系统运行需要的最基本的控制器配置保存到 DDR 里，包括时钟、串口信息
S3
保存 CPU 的现场
S4
配置 DDR 进入自刷新模式，能够保证低功耗运行，数据不丢失
S5
配置 RTC 的寄存器，进入 deep sleep 状态，关闭除 RTC 和 DDR 以外的各路电
S6
当 RTC 检测到 wakeup 唤醒按键按下，恢复对各路的供电
S7
系统上电启动，进入 Bootrom 执行，判断出是 deep sleep 模式，跳转到 RTC RAM 里执行
S8
在 RTC RAM 里，恢复对 PLL 和 DDR 的配置，DDR 退出自刷新状态
S9
跳转到休眠之前保持的程序执行 PC 地址，执行程序，恢复 CPU 的现场
S10
恢复系统时钟和串口(时钟和串口配置)
S11
系统接着休眠之前的程序位置继续执行程序
S12

图2