(19)国家知识产权局

(12)发明专利申请

(10)申请公布号 CN 117707628 A
(43)申请公布日 2024.03.15

(21)申请号 202310716499.5

(22)申请日 2023.06.15

(71)申请人 荣耀终端有限公司
地址 518040 广东省深圳市福田区香蜜湖街道东海社区红荔西路8089号深业中城6号楼A单元3401

(72)发明人 罗润发

(74)专利代理机构 上海音科专利商标代理有限公司 31267
专利代理师 贾玉

(51)Int.Cl.
G06F 9/4401(2018.01)

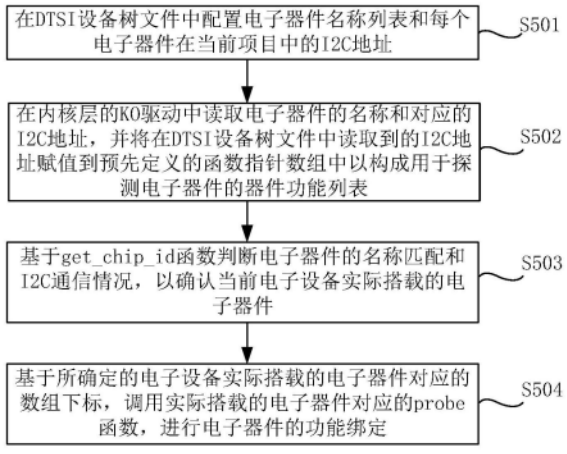
权利要求书2页 说明书19页 附图10页

(54)发明名称

一种器件初始化方法、电子设备和可读存储介质

(57)摘要

本申请涉及智能终端技术领域，具体涉及一种器件初始化方法、电子设备和可读存储介质。本申请提供的器件初始化方法中，在检测到电子设备开机启动时，可以基于电子设备的配置文件例如设备树文件获取第一器件对应的可配置的器件的标识信息和通讯地址，然后可以将预设的可搭载器件中器件的标识信息与设备树文件中读取的可配置的标识信息进行匹配，以将通信地址与对应的可搭载器件进行关联，此外，可以基于对应的可搭载器件的通信情况确定实际搭载的器件，即第一器件，并基于对应的可搭载器件的初始化信息对第一器件进行初始化配置。如此，能够有效简化探测流程，节省资源。



CN 117707628 A

1.一种器件初始化方法,其特征在于,应用于电子设备,所述电子设备包括第一器件;

所述方法包括:

检测到所述电子设备被启动;

基于所述电子设备的设备配置文件获取待初始化的第一器件对应的第一标识信息和第一通讯地址;

确定所述第一标识信息,与存储的第二器件的第二标识信息相匹配;

获取所述第二器件对应的第一初始化信息;

对应于基于第一通讯地址确定所述第二器件的通讯正常时,基于所述第一初始化信息对所述第一器件进行初始化配置。

2.根据权利要求1所述的方法,其特征在于,基于所述电子设备的设备配置文件获取待初始化的第一器件对应的第一标识信息和第一通讯地址;包括:

基于设备树文件获取所述待初始化的第一器件对应的标识信息集合和通讯地址集合;

所述第一标识信息为所述标识信息集合中的标识信息,所述第一通讯地址为所述通讯地址集合中的通讯地址。

3.根据权利要求2所述的方法,其特征在于,所述基于第一通讯地址确定所述第二器件的通讯正常;包括:

基于所述第二器件对应的第一器件标识函数和所述第一通讯地址读取所述第一器件的第三标识信息;

当所述第三标识信息与所述第二器件的第二标识信息匹配,确定所述第二器件的通讯正常。

4.根据权利要求3所述的方法,其特征在于,所述电子设备存储有所述第二器件对应的第一函数指针组,所述第一函数指针组中包括所述第二标识信息、所述第一初始化信息和所述第一器件标识函数;

对应于确定所述第一器件的所述第一标识信息,与存储的所述第二标识信息相匹配,建立所述第一通讯地址与所述第一函数指针组的关联。

5.根据权利要求4所述的方法,所述基于所述第一初始化信息对所述第一器件进行初始化配置,包括:

确定所述第二器件对应的第一函数指针组的标识信息;

基于所述第一函数指针组的标识信息调用所述第一初始化信息对所述第一器件进行初始化配置。

6.根据权利要求1-5任一项所述的方法,其特征在于,所述第一初始化信息包括探测函数、移除函数、电源挂起函数和电源恢复函数。

7.根据权利要求1-5任一项所述的方法,其特征在于,所述电子设备存储有所述第一器件对应的多个可搭载器件的标识信息;

所述确定所述第一标识信息,与存储的第二器件的第二标识信息相匹配;包括:

从所述多个可搭载器件的标识信息中确定出与所述第一标识信息匹配的所述第二标识信息。

8.一种电子设备,其特征在于,包括存储器,用于存储指令;

处理器,用于执行所述指令以使得所述电子设备实现权利要求1-7中任一项所述的器

件初始化方法。

9.一种可读存储介质,其特征在于,所述可读介质上存储有指令,该指令在电子设备上执行时使得所述电子设备执行权利要求1-7中任一项所述的器件初始化方法。

10.一种计算机程序产品,其特征在于,包括指令,所述指令在电子设备上执行时使得所述电子设备执行权利要求1-7中任一项所述的器件初始化方法。

一种器件初始化方法、电子设备和可读存储介质

技术领域

[0001] 本申请涉及智能终端技术领域，特别涉及一种器件初始化方法、电子设备和可读存储介质。

背景技术

[0002] 电子设备可以搭载不同供应商提供的同类型电子器件，通过在电子设备的软件基线(即电子设备中软件文档或源码的稳定版本)上预设多个不同供应商的同类型电子器件的代码信息，使得电子设备可以兼容不同供应商提供的同类型的电子器件。在电子设备开机阶段，例如首次开机时，电子设备需要对其内部的电子器件进行初始化配置，即调用电子器件对应的代码信息对电子器件进行初始化配置使得其在电子设备中电子器件对应的功能可以正常使用。因此，需要通过对电子器件进行探测来确定电子设备真正搭载的是多个同类型电子器件中的具体哪一个电子器件，进而再基于所确定的电子器件的代码信息对电子器件进行初始化配置。

[0003] 例如，通过在手机的软件基线上预设不同供应商提供的色温传感器S1-S5的代码信息，使得手机可以搭载(即兼容)色温传感器S1-S5中任意一个色温传感器。其中，若手机实际搭载的是色温传感器S1。在电子设备开机阶段，需要对手机的色温传感器进行探测，并准确检测出手机实际搭载是的色温传感器S1，进而基于色温传感器S1的代码信息对色温传感器S1进行初始化配置，使其在手机中对应的功能可以正常使用，例如可以包括对手机中相机在拍摄过程中的拍摄参数进行调节，或者对所拍摄的视频、图像的颜色进行调节等功能。

[0004] 如何实现准确检测电子设备中实际搭载的电子器件是目前需要解决的问题。

发明内容

[0005] 为解决上述问题，本申请实施例提供了一种器件初始化方法、电子设备和可读存储介质。

[0006] 第一方面，本申请实施例提供一种器件初始化方法，应用于电子设备，该方法包括：检测到电子设备被启动；基于电子设备的设备配置文件获取待初始化的第一器件对应的第一标识信息和第一通讯地址；确定第一标识信息，与存储的第二器件的第二标识信息相匹配；获取第二器件对应的第一初始化信息；对应于基于第一通讯地址确定第二器件的通讯正常时，基于第一初始化信息对第一器件进行初始化配置。

[0007] 可以理解，电子设备的设备配置文件可以是设备树文件，设备树文件中可以配置有当前项目可以配置的电子器件的各标识信息(例如器件名称)和各通讯地址，例如I2C地址等。其中各标识信息可以组成标识信息列表，各通讯地址可以组成通讯地址列表。

[0008] 可以理解，第一标识信息可以为基于配置文件获取的第一器件的名称列表中的电子器件名称，第一通讯地址可以为基于配置文件获取的第一器件的地址列表中的电子器件地址。

[0009] 可以理解,第一器件可以为电子设备实际搭载的电子器件。

[0010] 可以理解,第二器件的标识信息可以为电子设备预存储的第一器件对应的可搭载电子器件中的一个器件的标识信息。

[0011] 可以理解,设备树文件限定了当前电子设备的配置信息,因此,当设备树文件中不存在的器件名称将不会配置在当前电子设备中,例如,第一器件为色温传感器,可搭载的其中一个色温传感器器件名称为A2,基于设备树文件获取的器件名称为A3和A4,则证明A2对应的电子器件不可能存在于电子设备中,因此,无需对名称为A2的色温传感器进行后续探测,例如无需进行后续通讯是否正常的探测,能够有效简化探测流程,节省资源。

[0012] 在上述第一方面的一种可能实现中,基于电子设备的设备配置文件获取待初始化的第一器件对应的第一标识信息和第一通讯地址;包括:基于设备树文件获取待初始化的第一器件对应的标识信息集合和通讯地址集合;第一标识信息为标识信息集合中的标识信息,第一通讯地址为通讯地址集合中的通讯地址。

[0013] 在上述第一方面的一种可能实现中,基于第一通讯地址确定第二器件的通讯正常;包括:基于第二器件对应的第一器件标识函数和第一通讯地址读取第一器件的第三标识信息;当第三标识信息与第二器件的第二标识信息匹配,确定第二器件的通讯正常。

[0014] 可以理解,可以基于第一器件标识函数,例如get_chip_id函数读取第一器件的第三标识信息,当能够读取到实际搭载的第一器件对应的器件名称且读取到的第一器件的第三标识信息与基于设备树文件获取的对应函数指针组中的器件名称匹配,则可以判断当前探测的第一器件为实际搭载的电子器件且通信正常。

[0015] 可以理解,在一些实施方案中,一些可兼容的电子器件可能接线方式相同,例如器件A和器件B接线方式相同,导致器件A和器件B的I2C通讯情况实质是相同的,因此,可能出现当电子设备实际搭载的电子器件为器件B,在探测器件A时,也探测出通讯情况正常,导致误判器件A是实际搭载的器件,出现误判的问题。而本申请实施例提供的方法,在进行器件探测时,不只是单纯的判断I2C通信是否异常,还判断电子器件的名称是否匹配,基于上述两方面的判断,能够提高探测的精确度,避免一些实施方案中存在的误判情况。

[0016] 在上述第一方面的一种可能实现中,电子设备存储有第二器件对应的第一函数指针组,第一函数指针组中包括第二标识信息、第一初始化信息和第一器件标识函数;对应于确定第一器件的第一标识信息,与存储的第二标识信息相匹配,建立第一通讯地址与第一函数指针组的关联。

[0017] 可以理解,电子设备中存储有第二器件对应的第一函数指针组,当第一器件的第一标识信息与存储的第二器件的第二标识信息相匹配时,将第一通讯地址赋值到第一函数指针组中存储。

[0018] 在上述第一方面的一种可能实现中,基于第一初始化信息对第一器件进行初始化配置,包括:确定第二器件对应的第一函数指针组的标识信息;基于第一函数指针组的标识信息调用第一初始化信息对第一器件进行初始化配置。

[0019] 可以理解,第一函数指针组的标识信息可以是本申请实施例中提及的函数指针组的下标,例如数字标识等。

[0020] 可以理解,基于上述方案,可以根据识别到的实际搭载器件对应的数组,直接匹配函数指针组内对应的功能函数对实际搭载器件进行初始化配置,缩短了在同一基线存在多

个替代电子器件时，功能函数具备多个，需要重新寻找并匹配对应的功能函数导致的加载耗时过长。

[0021] 在上述第一方面的一种可能实现中，第一初始化信息包括探测函数、移除函数、电源挂起函数、电源恢复函数和器件标识函数。

[0022] 在上述第一方面的一种可能实现中，电子设备存储有电子设备中第一器件对应的多个可搭载器件的标识信息；确定第一标识信息，与存储的第二器件的第二标识信息相匹配；包括：从多个可搭载器件的标识信息中确定出与第一标识信息匹配的第二标识信息。

[0023] 第二方面，本申请实施例提供一种电子设备，包括存储器，用于存储指令；处理器，用于执行指令以使得上述第一方面及第一方面的各种可能实现提供的器件初始化方法被实现。

[0024] 第三方面，本申请实施例提供一种可读存储介质，可读介质上存储有指令，该指令在电子设备上执行时使得上述第一方面及第一方面的各种可能实现提供的器件初始化方法被实现。

[0025] 第四方面，本申请实施例提供一种计算机程序产品，包括指令，指令在电子设备上执行时使得电子设备执行上述第一方面及第一方面的各种可能实现提供的器件初始化方法被实现。

附图说明

[0026] 图1根据本申请的一些实施例，示出了一种手机开机配置的场景示意图；

[0027] 图2根据本申请的一些实施例，示出了一种器件初始化方法的流程示意图；

[0028] 图3根据本申请的一些实施例，示出了一种基于mutex锁的器件初始化方法的流程示意图；

[0029] 图4A根据本申请的一些实施例，示出了一种电子设备的系统架构示意图；

[0030] 图4B根据本申请的一些实施例，示出了另一种电子设备的系统架构示意图；

[0031] 图4C根据本申请的一些实施例，示出了又一种电子设备的系统架构示意图；

[0032] 图4D根据本申请的一些实施例，示出了再一种电子设备的系统架构示意图；

[0033] 图5根据本申请的一些实施例，示出了另一种器件初始化方法的流程示意图；

[0034] 图6根据本申请的一些实施例，示出了一种器件初始化方法的具体流程示意图；

[0035] 图7根据本申请的一些实施例，示出了一种电子设备10的示意图。

具体实施方式

[0036] 本申请的说明性实施例包括但不限于一种器件初始化方法、电子设备和可读存储介质。

[0037] 可以理解，本申请实施例可以应用于任意需要进行器件探测的电子设备，包括但不限于手机、平板电脑、可穿戴设备(如智能手表、智能手环等)、车载设备、物联网设备、智能家居设备等。为便于描述，下文以电子设备手机为例介绍本申请实施例的技术方案。

[0038] 可以理解，本申请实施例可以应用于检测任意需要被电子设备进行检测与配置的电子器件，包括但不限于电子设备中的接近光器件、环境光器件和激光器件(direct time-of-flight，Dtof)等。为便于描述，以下以手机中的电子器件色温传感器为例介绍本申请的

技术方案。

[0039] 为便于理解,首先介绍本申请实施例中涉及的术语。

[0040] (1)软件基线

[0041] 软件基线是电子设备中软件代码文档或源码的一个稳定版本,在软件开发过程中,当对软件产生新的需求,需要进行频繁更新时,可能会产生不可知的错误。此时可以把系统回退到上一个软件基线,以便用一个稳定的版本来满足设备测试等需求,同时修复现有的问题。在本申请实施例中,例如可以在软件基线中预设色温传感器S1-S5的代码信息,以便在检测电子设备实际搭载的色温传感器S1时,可以基于色温传感器S1的代码信息对其进行初始化配置。

[0042] (2)内核目标模块

[0043] 内核目标模块(kernel object module,KO)是内核层中的可加载模块,KO模块中包括驱动代码,可以动态地向内核层添加功能。在电子设备运行时,可以通过加载或卸载KO模块来扩展或减少内核功能。KO模块通常可以包括驱动程序、文件系统、网络协议栈、等内核功能。可以通过编写KO模块来扩展或改进内核层的功能。

[0044] (3)设备树

[0045] 设备树(device tree source,DTSI)是一种描述硬件的数据结构,设备树由一系列被命名的节点和属性组成,硬件设备可以通过采用设备树,使得不用在内核层中进行大量的冗余编码;属性是指成对出现的名称和值。

[0046] 为了方便理解本申请技术方案,下面以图1示出的场景为例,对器件初始化方法的应用场景进行介绍。图1根据本申请的一些实施例,示出了一种手机10的进行开机初始化配置的示意图。

[0047] 参考图1,用户可以通过对手机10的开机键11进行长按,控制手机10进行开机配置。当用户长按手机10的开机键11的时间达到预设时长后,手机10的交互界面显示提示框101,提示用户确定是否进行手机10的激活与配置,用户可以通过点击确认框1011确定开始激活与配置手机10。在手机10开机配置过程中,手机10可以对其内部所使用的电子器件,例如色温传感器进行探测,以确定其搭载的色温传感器的信息,色温传感器的信息可以包括色温传感器的名称信息、型号信息、通信地址信息等。

[0048] 可以理解,色温传感器是一种用于检测光源颜色或色温的传感器。可以用于根据光源的颜色或色温,传递相应的电信号,使得电信号转换后对应的二进制信息可以被机器识别。

[0049] 在一些实施例中,色温传感器可以根据环境的色温来自动调节电子设备的屏幕色温,以及可以在电子设备拍摄时用于探测环境的色温,使得所拍摄视频或图像色彩更准确,或者对图像进行颜色校准等,在此不做限定。

[0050] 可以理解,图1所示的手机10的开机方式只是一种示例,在另一些实施例中,可以通过手机10的其他按键或按键组合控制手机10开机,在此不做限定。

[0051] 如前所述,电子设备可以支持搭载不同供应商提供的同类型电子器件,可以通过在电子设备的软件基线上预设多个不同供应商的同类型电子器件的代码信息,使得电子设备可以兼容不同供应商提供的同类型的电子器件。在电子设备开机阶段,电子设备需要对其内部的电子器件进行初始化配置,即调用电子器件对应的代码信息对电子器件进行初始

化配置使得其对应的功能可以正常使用,因此,需要通过对电子器件进行探测来确定电子设备真正搭载的是多个同类型电子器件中的具体哪一个电子器件,进而再基于所确定的电子器件的代码信息对电子器件进行初始化配置。

[0052] 在本申请一些实施例中,可以依次通过探测函数对电子器件进行探测,并通过判断电子器件的I2C地址通信是否正常来判断其是否是电子设备实际搭载的色温传感器,但是该方案存在需要运行到对可搭载的器件进行驱动,以进行I2C地址通信情况的判断时才能确定是否为实际搭载的器件,且每个器件均需要进行判断,导致流程浪费,资源消耗过大。

[0053] 例如图2示出了一种器件初始化方法的流程示意图。为了便于理解,以电子器件为色温传感器为例进行说明,具体流程包括:

[0054] S201:开始探测模式(Mod_probe)。

[0055] 可以理解,器件探测(Mod_probe)函数是linux的一个驱动命令,可载入指定的单个模块或者载入一组模块至内核层。本申请中,内核层可以调用器件探测(Mod_probe)函数以执行后续流程来实现将色温传感器加载至KO模块中并与上层通信。

[0056] S202:初始化KO模块的公共入口函数(Mod_init)。

[0057] 可以理解,每一个驱动程序都有Mod_init这个语句,表示在系统启动的时自动执行驱动命令,即定义公共人口函数Mod_init。

[0058] 在一些实施例中,各电子器件对应的驱动代码都可以封装在Kernel中的KO模块中,在进行器件探测时,可以初始化KO模块的公共入口函数Mod_init,以实现对驱动的初始化和退出相关的函数进行定义。

[0059] 可以理解,Linux内核编译中需要包含内核层kernel头文件,内核层的KO模块的驱动代码中需要包含下面三个头文件:#include<linux/init.h>、#include<linux/module.h>、#include<linux/kernel.h>。其中,init.h定义了驱动的初始化和退出相关的函数;kernel.h定义了经常用到的函数原型及宏定义,module.h定义了与内核模块相关的函数、变量等。

[0060] S203:基于公共入口函数调用第i个色温传感器的i2c_add_driver函数。

[0061] 在一些实施例中,可以基于公共入口函数调用当前第i个色温传感器的i2c_add_driver函数,使得可以注册当前第i个色温传感器到KO模块的设备链表中。可以理解,设备链表即内核层中可以兼容的电子器件的集合。

[0062] S204:调用第i个色温传感器的real_probe函数,对当前色温传感器进行驱动。

[0063] 在一些实施例中,可以调用当前色温传感器的real_probe函数,也即准备对当前色温传感器进行驱动,使得当前色温传感器可以绑定至内核层的公共节点入口。

[0064] S205:判断当前色温传感器的I2C通信是否正常。

[0065] 在一些实施例中,在对当前色温传感器进行驱动时,判断当前色温传感器的I2C通信是否正常,若通信正常则转至S206,将通信正常的色温传感器绑定至内核层中的公共节点入口。若通信异常,转至步骤S207,将通信异常的色温传感器将作为一个虚拟器件存储在系统内核层中。

[0066] S206:将通信正常的色温传感器绑定至内核层中的公共节点入口。

[0067] 在一些实施例中,如果当前色温传感器的通信正常,则将该通信正常的色温传感

器绑定至内核层中的公共节点入口,使得可以基于内核层的驱动将当前色温传感器与上层通信连接

[0068] S207:将通信异常的色温传感器将作为一个虚拟器件存储在系统中。

[0069] 在一些实施例中,如果当前色温传感器的通信异常,则将该通信异常的色温传感器将作为一个虚拟器件存储在系统中。

[0070] S208:函数返回,i=i+1。

[0071] 可以理解,不论当前色温传感器的通信是否正常,都需要返回步骤S202继续执行下一个,即第i+1个色温传感器的检测流程。

[0072] 如前所述,该方案存在需要运行到对可搭载的器件进行驱动,以进行I2C地址通信情况的判断才能确定是否为实际搭载的器件,且每个器件均需要进行判断,导致流程浪费,资源消耗过大。

[0073] 在另一些实施例中,可以通过增加互斥量(mutual exclusion,Mutex)锁来判断其是否是电子设备实际搭载的色温传感器,在每个器件探测的时候通过增加mutex锁,之后先判断mutex锁的标志位,如果标志位不为空说明已经探测成功,如果标志位为空说明当前还没有器件探测成功,并继续依次调用电子器件的I2C地址来判断I2C地址通信情况,基于I2C地址通信情况实现确定对应的色温传感器是否是电子设备实际搭载的色温传感器。该方案中通过在判断I2C通信之前增加判断Mutex锁的标志位,实现了在一个色温传感器探测完后再探测下一个色温传感器,避免了在当前色温传感未探测完成时继续探测了其他传感器,导致当前色温传感器的探测流程中断,造成的需要再次探测的流程繁琐问题,但仍然需要依次执行上述图2提及的所有色温传感器的I2C通信判断流程,导致流程浪费,资源消耗过大。

[0074] 参考图3,图3示出了一种基于mutex锁的器件初始化方法的流程示意图。为了便于理解,以电子器件为色温传感器为例进行说明,具体流程包括:

[0075] S301:开始探测模式(Mod_probe)。

[0076] 可以理解,步骤S301可以参考上述步骤S201,在此不再赘述。

[0077] S302:初始化KO模块的公共入口函数(Mod_init)。

[0078] 可以理解,步骤S302可以参考上述步骤S202,在此不再赘述。

[0079] S303:基于公共入口函数调用第i个色温传感器的i2c_add_driver函数。

[0080] 可以理解,步骤S303可以参考上述步骤S203,在此不再赘述。

[0081] S304:调用当前色温传感器的real_probe函数。

[0082] 可以理解,步骤S304可以参考上述步骤S204,在此不再赘述。

[0083] S305:通过Mutex锁对当前色温传感器的驱动进程进行锁定(Probe_lock)。

[0084] 在一些实施例中,通过增加互斥量(mutual exclusion,Mutex),即Mutex锁,对当前色温传感器的驱动进程进行Probe-lock,也即是说,将当前色温传感器的驱动进程进行锁定,在当前色温传感器的驱动过程中,无法再对其他色温传感器进行探测并驱动。

[0085] 可以理解,Mutex锁是一种保证串行化的睡眠锁机制,可以挂起当前进程,即锁定当前的检测进程。Mutex锁的使用需要设置一个标志位来确定Mutex锁当前是否处于锁定状态。

[0086] S306:判断Mutex锁的标志位color_name是否为空。

[0087] 在一些实施例中，判断Mutex锁的标志位color_name，当确定标志位color_name为空时，则确定当前没有色温传感器处于驱动进程中，转至步骤S307，调用当前色温传感器的real_probe函数，对当前色温传感器进行驱动；当确定标志位color_name不为空时，则确定当前已经有色温传感器探测成功，转至步骤S310，则不执行当前色温传感器的I2C通信判断，而是解锁当前探测进程以继续执行下一个色温传感器的探测流程。

[0088] S307：调用当前色温传感器的real_probe函数，对当前色温传感器进行驱动。

[0089] 在一些实施例中，当确定标志位color_name为空时，则确定当前没有色温传感器处于驱动进程中，基于I2C调用当前色温传感器的real_probe函数，对当前色温传感器进行驱动，使得当前色温传感器可以绑定至内核层的公共节点入口。

[0090] S308：判断当前色温传感器I2C通信是否正常。

[0091] 在一些实施例中，在对当前色温传感器进行驱动时，判断当前色温传感器的I2C通信是否正常，若通信正常则转至S309，设置标志位color_name表示当前色温传感器为电子设备实际搭载的电子器件；若通信异常，仍转至步骤S310，不执行当前色温传感器的I2C通信判断，而是解锁当前探测进程以继续执行下一个色温传感器的探测流程。

[0092] S309：设置标志位color_name表示当前色温传感器为电子设备实际搭载的电子器件。

[0093] 在一些实施例中，确定当色温传感器的I2C可以正常通信，确定当前色温传感器为电子设备实际搭载的色温传感器，并设置标志位color_name为对应的标识，以表示当前已经有色温传感器探测成功。

[0094] S310：解锁当前色温传感器的调用进程(Probe_unlock)。

[0095] 在一些实施例中，在当前色温传感器的检测进程结束后，通过Probe_unlock函数解锁当前色温传感器的调用进程。

[0096] 在一些实施例中，当确定标志位不为空后，表明当前存在色温传感器正在检测进程中或当前已有色温传感器探测成功，可以通过Probe_unlock函数解锁当前成功探测的色温传感器的调用进程。

[0097] S311：函数返回，i＝i+1。

[0098] 在一些实施例中，当确定电子设备实际搭载的色温传感器后，可以把实际搭载的色温传感器绑定至公共节点入口使得其与上层完成连接。并返回步骤S302，继续探测下一个色温传感器，即不论标志位为空还不为空，是否有色温传感器通信正常，都需要返回步骤S202继续执行下一个，即第i+1个色温传感器的检测流程。

[0099] 在一些实施例中，若当前的色温传感器不是电子设备实际搭载的色温传感器，则将当前色温传感器作为内核层中的悬空虚拟器件，返回步骤S302，继续探测下一个色温传感器。在一些实施例中，当标志位为空时，则直接返回步骤S302，继续探测下一个色温传感器。

[0100] 上述方案中虽然通过在判断I2C通信之前增加判断Mutex锁的标志位，实现了在一个色温传感器探测完后再探测下一个色温传感器，避免了在当前色温传感器未探测完成时继续探测了其他传感器，导致当前色温传感器的探测流程中断，造成的需要再次探测的流程繁琐问题，但仍然需要依次执行上述图2提及的所有色温传感器的I2C通信判断流程，导致流程浪费，资源消耗过大。

[0101] 为了解决上述问题,本申请实施例提供一种器件初始化方法,首先,可以预设待初始化器件对应的可兼容电子器件的函数指针组(或称为数组),例如待初始化器件为色温传感器,色温传感器有5个型号的可兼容的色温传感器,则预设5个型号色温传感器分别对应的函数指针组,函数指针组中可以包括各电子器件对应的器件标识信息(例如,器件名称、I2C地址等)、探测(probe)、移除(remove)、电源挂起(suspend)和电源恢复(resume)函数等功能函数、

[0102] 然后可以基于电子设备对应的DTSI设备树文件获取电子设备当前项目可以配置的电子器件的各标识信息(例如器件名称)和各I2C地址等。并将各函数指针组中的器件名称基于DTSI文件获取的器件名称进行匹配,获取对应的待选函数指针组,将基于DTSI设备树文件获取的器件名称对应的I2C地址赋值到对应的待选函数指针组中。将待选函数指针组对应的器件中通信正常的器件作为当前电子设备实际搭载的器件,并基于待选函数指针组中的功能函数进行器件的初始化配置。

[0103] 可以理解,DTSI设备树文件限定了当前电子设备的配置信息,因此,在DTSI设备树文件中不存在的器件名称将不会配置在当前电子设备中,例如,色温传感器对应的其中一个可兼容的色温传感器的器件名称为A2,基于DTSI设备树文件获取的器件名称为A3和A4,则证明A2对应的电子器件不可能存在于电子设备中,因此,无需进行后续探测,例如无需进行后续通讯是否正常的探测,能够有效简化探测流程,节省资源。

[0104] 在一些实施例中,各电子器件对应的函数指针组中还可以包括关键函数(get_chip_id)。

[0105] 在确定出待选函数指针组后,可以分别基于待选函数指针组中的关键函数(get_chip_id)读取当前实际搭载的电子器件的器件名称,当能够读取到实际搭载的电子器件对应的器件名称且读取到的器件名称与对应的待选函数指针组中器件名称匹配,则可以判断当前探测的可搭载电子器件为电子设备实际搭载的电子器件且通信正常,则将当前探测的可搭载电子器件作为电子设备实际搭载的电子器件,进行器件的功能绑定和初始化流程等。若无法读取电子设备实际搭载的电子器件的器件名称或读取到的器件名称与对应待选函数指针组器件名称不匹配,即当前探测的可搭载电子器件不是实际搭载的电子器件,则直接进行下一个电子器件的探测。

[0106] 如此,在探测时,不只是单纯的判断I2C通信是否异常,还判断电子器件的名称是否匹配,能够提高探测的精确度,避免现有技术中存在的误判情况,下边对现有技术中存在的误判情况进行说明:可以理解,一些可兼容的电子器件可能接线方式相同,例如器件A和器件B接线方式相同,导致器件A和器件B的I2C通讯情况实质是相同的,因此,可能出现当电子设备实际搭载的电子器件为器件B,在探测器件A时,也探测出通讯情况正常,导致误判器件A是实际搭载的器件。

[0107] 在一些实施例中,函数指针组包括各电子器件对应的probe、remove、suspend和resume函数等功能函数,在确定出实际搭载电子器件后,可以直接基于函数指针组中的probe函数进行器件的功能绑定和初始化流程,此外,remove函数可以用于在内核层中移除电子器件时清理相应的内存资源;suspend函数可以管理电源管理模块休眠;resume函数可以管理电源恢复。如此,将器件所需使用的函数进行归一化,可以根据识别到的实际搭载器件,直接匹配函数指针组内对应的功能函数,缩短了同基线存在多个替代电子器件时,功能

函数具备多个,需要重新寻找并匹配对应的功能函数导致的加载耗时过长。

[0108] 为了方便理解本申请技术方案,以色温传感器的架构例,对本申请提及的电子设备的系统架构进行说明,

[0109] 在一些实施例中,如图4A所示,电子设备的系统架构可以包括框架层、硬件抽象层、内核层、硬件层。其中:

[0110] 框架层中可以包括相机服务单元,在一些实施例中,当用户打开相机应时,可以通过框架层调用相机服务,例如,获取相机设备属性等。

[0111] 硬件抽象层(hardware abstraction layer,HAL)中可以包括自动曝光、自动白平衡、网络控制系统等。硬件抽象层是对硬件设备的抽象和封装,为系统在不同硬件设备提供统一的访问接口。HAL层处于框架层和内核层之间,HAL层可以屏蔽不同硬件设备的差异,为系统提供统一的访问硬件设备的接口。

[0112] 硬件抽象层可以利用硬件抽象层接口定义语言(interface definition language,HDIL)与框架层进行通信。例如,硬件抽象层可以基于框架层中的获取相机设备属性的通知调用自动曝光、自动白平衡、网络控制等功能。硬件抽象层中还可以包括自动化测试工具通信协议(MMI_apk),可以用于为电子设备的电子器件的测试提供通信测试接口,例如可以为电子设备中色温传感器的颜色测试提供通信测试接口,也可以为电子设备中闪烁传感器的闪烁测试提供通信测试接口。硬件抽象层还可以包括函数服务器,函数服务器中可以包括diag_client函数,该函数可以用于为色温传感器的驱动提供支持,例如可以通过diag_client函数驱动色温传感器进行颜色校准。硬件抽象层中还可以包括传感器硬件抽象层。

[0113] 内核层中可以包括色温传感器驱动。

[0114] 硬件层中可以包括不同型号的色温传感器等。

[0115] 可以理解,上述框架层、硬件抽象层、内核层和硬件层中还可以包括除上述以外的其他单元或模块,在此不做限定。

[0116] 例如,在一些实施例中,如图4B所示,传感器硬件抽象层可以包括色温传感器硬件抽象层和闪烁传感器硬件抽象层。

[0117] 内核层中可以包括公共节点入口和色温传感器驱动文件。内核层的色温传感器的驱动文件中可以包括色温传感器接口以及色温传感器的代码信息等。色温传感器接口可以用于将色温传感器绑定至公共节点入口,从而通过公共节点入口将色温传感器与上层应用做绑定。色温传感器的驱动文件中可以包括驱动代码信息,例如可以包括色温传感器TCS3408的驱动代码信息、色温传感器BU27006的驱动代码信息以及色温传感器A57341的驱动代码信息等。可以理解,上述色温传感器可以是环境光传感器,也可以是频闪传感器,在此不做限定。可以理解,在另一些实施例中,色温传感器的驱动文件中还可以包括除上述色温传感器以外的其他色温传感器的驱动代码信息,在此不做限定。

[0118] 硬件层中可以是不同型号的色温传感器,例如色温传感器的型号可以为TCS3408、BU27006或色温传感器A57341等,在此不做限定。

[0119] 在一些实施例中,基于图4B所示的结构可以实现本申请上述图3所示的方法,例如,可以基于内核层的公共入口函数调用色温传感器TCS3408的i2c_add_driver函数,并调用色温传感器TCS3408的real probe函数对色温传感器TCS3408进行驱动,并判断色温传感

器TCS3408的I2C通信是否正常，如果通信正常，则可以将色温传感器TCS3408通过色温传感器接口绑定至公共节点入口，再通过公共节点入口将色温传感器TCS3408与上层应用进行功能绑定等；如果通信异常则将色温传感器TCS3408作为虚拟器件存储在内核中。进一步，继续探测色温传感器BU27006或A57341等传感器，直至对所有色温传感器完成探测。

[0120] 但是，上述方案中需要依次执行所有色温传感器的I2C通信判断流程，即对所有色温传感器都进行探测，即使当前已经成功探测到有色温传感器为电子设备实际搭载的色温传感器，仍然需要对其他色温传感器进行探测，导致流程浪费，资源消耗过大。

[0121] 此外，上述方案中，每次调用的电子器件都会注册到电源管理模块中，导致不是电子设备实际搭载的其他电子器件会在电源管理模块中变成了虚拟器件，进而导致电源管理模块的每一次挂起(suspend)和恢复(resume)都会调用虚拟器件，造成资源的浪费。

[0122] 在一些实施例中，为解决上述问题，可以如图4C所示，可以在内核层增器件探测和功能绑定模块，如此可以实现通过色温传感器接口中的器件探测和功能绑定模块完成对实际搭载的色温传感器的正确探测后，将实际搭载的色温传感器通过公共节点入口绑定至上层应用。

[0123] 相较于上述图4B中的色温传感器接口，图4C中在色温传感器接口中可以包括器件探测与功能绑定模块，器件探测与功能绑定模块可以实现在正确检测电子设备实际搭载的色温传感器后，再将电子设备实际搭载的色温传感器通过公共节点入口与上层进行通信以及功能绑定等，而对于不是电子设备实际搭载的色温传感器不会通过公共节点入口直接被注册到电源管理模块中，避免出现在电源管理模块中存在悬空的虚拟器件造成电源管理模块对虚拟器件的反复挂起和恢复的操作，即解决了上述方案中提及的资源浪费问题。

[0124] 此外，在基于器件探测和功能绑定模块进行器件探测时，可以把从DTSI设备树文件读取的I2C地址赋值到各可搭载的电子器件对应的函数指针组，其中，函数指针组中包括器件名称(例如器件id)和关键函数(get_chip_id)。然后，通过当前探测的可搭载器件对应的关键函数(get_chip_id)基于对应的I2C地址读取实际搭载的电子器件的id，当能够读取到实际搭载的电子器件对应的id且读取到的id与当前函数指针组中的id匹配，则可以判断当前探测的可搭载电子器件为实际搭载的电子器件且通信正常，则将当前探测的可搭载电子器件作为实际搭载的电子器件，进行器件的功能绑定和初始化流程。当能够读取到实际搭载的电子器件对应的id但读取到的id与当前函数指针组中的id不匹配，则当前探测的可搭载电子器件不是实际搭载的电子器件，并进行下一个电子器件的探测。基于此，在进行器件探测时，不只是单纯的判断I2C通信是否异常，还判断电子器件的名称是否匹配，基于上述两方面的判断，能够提高探测的精确度，避免一些实施方案中存在的误判情况。

[0125] 在一些方案中，相较于本申请实施例中在内核层为传感器提供了抽象化的公共节点入口，一些方案未采用抽象化的公共节点入口，而是每个电子器件都有设置对应的驱动和节点入口，然后在传感器的硬件抽象层中设置采用公共的传感器的接口，以此实现对上的接口统一，而基于上述方法存在明显缺陷，即对多器件的兼容性差，当使用不同供应商提供的替代电子器件时，需要在内核层以及硬件抽象层进行功能扩充，兼容性不高。

[0126] 例如图4D所示，图4D示出了一种色温传感器的接口示意图，可以看出，色温传感器TMF882需要通过TMF882节点入口与色温传感器TMF882的硬件抽象层通信；色温传感器VL5311需要通过VL5311节点入口需要与色温传感器VL5311的硬件抽象层通信。即每个色温

传感器都需要通过其对应的色温传感器接口和节点入口与上层进行绑定,每个接口和节点入口的兼容性不高。

[0127] 下面对本申请实施例提供的器件初始化方法进行详细说明。参考图5,图5示出了一种器件初始化方法的流程示意图。为了便于理解,以电子器件为色温传感器为例进行说明,具体流程包括:

[0128] S501:在DTSI设备树文件中配置电子器件名称列表和每个电子器件在当前项目中的I2C地址。

[0129] 可以理解,DTSI设备树文件中配置了当前项目可以配置的电子器件的各标识信息(例如器件名称)和各通讯地址,例如I2C地址等。其中各标识信息可以组成器件名称列表,各通讯地址可以组成通讯地址列表。

[0130] 在一些实施例中,可以在DTSI设备树文件中配置电子器件的名称,用来匹配在探测电子器件时驱动代码中的电子器件的名称。以及可以在DTSI设备树文件中配置每个电子器件在当前项目中的I2C地址,用来在电子器件名称匹配后,基于其I2C地址判断电子器件的通信情况。可以理解,在DTSI设备树文件中不存在的器件名称将不会配置在当前电子设备中,例如,可搭载的其中一个色温传感器器件名称为A2,基于DTSI设备树文件获取的器件名称为A3和A4,则证明A2对应的电子器件不可能存在于电子设备中,因此,无需进行后续探测,例如无需进行后续通讯是否正常的探测,能够有效简化探测流程,节省资源。

[0131] S502:在内核层的KO驱动中读取电子器件的名称和对应的I2C地址,并将在DTSI设备树文件中读取到的I2C地址赋值到预先定义的函数指针组中以构成用于探测电子器件的器件功能列表。

[0132] 在一些实施例中,函数指针组可以指函数指针组,是在编译阶段预先定义的,函数指针组本质是一个结构体数组,每个函数指针组包括对应可搭载的电子器件的名称、i2c地址和功能函数,功能函数可以包括:get_chip_id、probe、remove、suspend和resume函数等。

[0133] 在一些实施例中,每个函数指针组可以对应有数组标识,例如可以为数字形式的数组下标等。

[0134] 实际探测时,在内核层的KO驱动中读取电子器件的名称和对应的I2C地址,并将各数组中的器件名称基于DTSI设备树文件获取的器件名称进行匹配,获取对应的待选函数指针组,将基于DTSI设备树文件获取的器件名称对应的I2C地址赋值到对应的待选函数指针组的寄存器地址(reg-addr)中。将待选函数指针组对应的电子器件中通信正常的器件作为当前电子设备实际搭载的器件,并基于待选函数指针组中的功能函数进行器件的初始化配置。例如,电子器件可以支持搭载器件A1-A5,其对应有各自的函数指针组,且各自函数指针组下标分为可以为1-5;进一步,基于从DTSI设备数中解析出的器件A3、A4。将器件A1-A5中的每一个器件的名称分别与从设备树解析出的两个器件名称进行匹配,当名称匹配时,将匹配的器件对应的I2C地址赋值到对应函数指针组中存储,进而可以基于所匹配的器件对应的函数指针组中的函数对匹配器件进行初始化配置等。

[0135] 可以理解,器件功能列表中可以包括电子器件的名称、i2c地址和功能函数,例如功能函数可以包括:get_chip_id、probe、remove、suspend和resume等。

[0136] 可以理解,轮询是指依序确定当前轮询的函数指针组中的色温传感器的名称是否与色温传感器名称列表(color_sensor_list)中的色温传感器的名称匹配,若成功匹配则

结束轮询,并将色温传感器对应的I2C地址存储至函数指针组reg-addr中;若不匹配则继续判断下一个色温传感器是否与数组匹配。

[0137] S503:基于get_chip_id函数确定电子器件的名称匹配和I2C通信情况,以确认当前电子设备实际搭载的电子器件。

[0138] 可以理解,get_chip_id函数可以用于判断当前检测的电子器件的名称是否匹配以及I2C通信是否正常。

[0139] 在一些实施例中,可以通过当前探测的可搭载器件对应的关键函数(get_chip_id)基于对应的I2C地址读取实际搭载的电子器件的id(名称),当能够读取到id,且读取到的id与函数指针组中的id匹配,则可以判断正在探测的可搭载电子器件为实际搭载的电子器件且通信正常,并将当前正在探测的可搭载电子器件作为实际搭载的电子器件。

[0140] 在一些实施例中,可以通过监测返回值来确认当前探测的电子器件是否是电子设备实际搭载的电子器件。例如,当返回值为“true”时,确定当前探测的电子器件是电子设备实际搭载的电子器件,并保存当前电子器件的对应的函数指针组的数组下标。

[0141] S504:基于所确定的电子设备实际搭载的电子器件对应的数组下标,调用实际搭载的电子器件对应的probe函数,进行电子器件的功能绑定。

[0142] 在一些实施例中,当get_chip_id函数判断出当前色温传感器的名称匹配以及I2C通信正常时,保存当前色温传感器驱动(color_driver_ops)数组实例,即将数组下标保存。例如,确定电子设备搭载的是当前色温传感器TCS3408,其对应的color_driver_ops数组的下标为3,则将当前色温传感器TCS3408的数组3保存,并基于数组3中的函数对色温传感器TCS3408进行调用和驱动,即将探测成功的色温传感器通过器件探测与功能绑定模块绑定至色温传感器接口,并通过色温传感器接口将探测成功的色温传感器绑定至公共节点入口,通过公共节点入口实现色温传感器TCS3408与上层应用的功能绑定以及初始化等功能。

[0143] 在一些实施例中,当系统加载模块或者为该模块创建实例的时候,例如,在KO模块保存色温传感器TCS3408的数组,系统会自动为KO模块分配色温传感器TCS3408的内存空间,并在分配内存后,自动对环境变量进行初始化,然后利用环境变量初始化硬件、启动操作系统等。

[0144] 在一些实施例中,当色温传感器TCS3408探测失败,则器件探测与功能绑定模块不会色温传感器TCS3408绑定至色温传感器接口,也即不会被绑定至公共节点入口与上层进行功能绑定。

[0145] 基于上述方法,可以根据识别到的实际搭载器件对应的数组,直接匹配函数指针组内对应的功能函数对实际搭载器件进行初始化配置,缩短了在同一基线存在多个替代电子器件时,功能函数具备多个,需要重新寻找并匹配对应的功能函数导致的加载耗时过长。

[0146] 并且在一些实施例中,函数指针组还包括各电子器件对应的probe、remove、suspend和resume函数等功能函数,在确定出实际搭载电子器件后,可以直接基于函数指针组中的probe函数进行器件的功能绑定和初始化流程,此外,remove函数可以用于在内核层中移除电子器件时清理相应的内存资源;suspend函数可以管理电源管理模块休眠;resume函数可以管理电源恢复。如此,将器件所需使用的函数进行归一化,可以根据识别到的实际搭载器件,直接匹配函数指针组内对应的功能函数,缩短了同基线存在多个替代电子器件时,功能函数具备多个,需要重新匹配对应的功能函数导致的加载耗时。

[0147] 此外,在本申请实施例中,通过在DTSI设备树文件中配置部分获取探测结果可能使用的多个电子器件的信息,开创性的采用全局电子器件数组的形式,动态获取电子器件的配置信息,并在对电子器件进行探测时确认实际搭载的电子器件,实现了动态的电子器件识别。当正确检测出电子器件实际搭载的电子器件后,不需要再对其他电子器件再进行探测,大大改善了流程冗余问题,提高了检测效率。

[0148] 另外,对于不是电子设备实际搭载的色温传感器不会通过公共节点入口直接被注册到电源管理模块中,避免出现在电源管理模块中存在悬空的虚拟器件造成电源管理模块对虚拟器件的反复挂起和恢复的操作,即解决了上述方案中提及的资源浪费问题。

[0149] 下面对本申请提供的器件初始化方法的具体流程进行说明。

[0150] 参考图6,图6示出了一种器件初始化方法的具体流程示意图。为了便于理解,以电子器件为色温传感器为例进行说明,具体流程包括:

[0151] S601:初始化公共入口函数(Mod_init)。

[0152] 在一些实施例中,每一个驱动程序都有Mod_init这个语句,表示在系统启动的时自动执行驱动命令,即定义公共人口函数Mod_init。

[0153] S602:初始化color_driver_ops数组。

[0154] 在一些实施例中,对color_driver_ops数组进行初始化,color_driver_ops数组中包括内核层编译的驱动代码中所支持的色温传感器的名称、I2C通信地址、功能函数等。其中,功能函数可以包括probe探测函数、remove移除探测函数、suspend电源挂起函数和resume电源恢复函数等。

[0155] 例如,可搭载的色温传感器有5个,则可以设置5个函数指针组,例如,函数指针组中可以包括色温传感器S1-S5的名称、I2C通信地址、功能函数等。其中,功能函数可以包括probe探测函数、remove移除探测函数、suspend电源挂起函数和resume电源恢复函数等。可以理解,不同色温传感器的函数指针组的数组下标都不相同,例如,色温传感器S1-S5的数组下标可以对应为1-5。可以理解,不同色温传感器的数组下标可以在初始编译时预设为任意数值等,在此不做限定。

[0156] S603:基于公共入口函数调用第i个色温传感器的i2c_add_driver函数。

[0157] 在一些实施例中,可以基于公共入口函数调用第i个色温传感器i2c_add_driver函数,使得可以注册每个色温传感器到KO模块的设备链表中。其中,设备链表为内核层中可以兼容的电子器件的集合。

[0158] S604:Client指针判空。

[0159] 在一些实施例中,可以基于客户端(Client)指针对色温传感器的传输参数进行判空,即对系统内存分配以及驱动代码信息进行检查,以保证例如色温传感器的名称、I2C通信地址、关键函数等传输参数在color_driver_ops数组中不为空。

[0160] S605:Client->dev保存。

[0161] 在一些实施例中,对Client->dev客户端对应的结构体进行保存,即对color_driver_ops数组进行保存。

[0162] 可以理解,color_driver_ops数组中包括内核层编译的驱动代码中支持的每个色温传感器的名称、I2C通信地址、功能函数等,其中功能函数可以包括probe探测函数、remove移除探测函数、suspend电源挂起函数和resume电源恢复函数等。

[0163] S606:从DTSI设备树文件中获取电源管理模块Ido的结构体。

[0164] 在一些实施例中,从内核层的DTSI设备树文件中获取电源管理模块Ido的结构体,可以理解,Ido为电源管理模块,可以控制电源管理模块的挂起和恢复等。

[0165] S607:基于regulator-enable函数给电源管理模块Ido上电。

[0166] 在一些实施例中,通过regulator-enable函数控制电源管理模块Ido上电。

[0167] S608:从DTSI设备树文件中解析色温传感器的地址列表和名称列表。

[0168] 在一些实施例中,基于DTSI设备树文件解析色温传感器的地址列表和名称列表。例如,内核层的驱动代码中包括支持搭载的色温传感器S1-S5的信息,可以通过DTSI设备树文件解析出色温传感器S1-S3的地址列表和名称列表等信息。

[0169] 可以理解,DTSI设备树文件中可以包括电子器件的名称信息和通讯地址信息,可以基于DTSI设备树文件解析出的电子器件。

[0170] S609:轮询数组匹配色温传感器的名称。

[0171] 在一些实施例中,可以基于电子设备对应的DTSI设备树文件获取电子设备当前项目可以配置的色温传感器的各标识信息(例如色温传感器名称)和各I2C地址等。并将各函数指针组中的色温传感器名称基于DTSI文件获取的色温传感器名称进行匹配,获取对应的待选函数指针组,将基于DTSI设备树文件获取的色温传感器名称对应的I2C地址赋值到对应的待选函数指针组中。

[0172] 在一些实施例中,色温传感器的名称可以为:TCS3408、BU27006和A57341等,在此不做限定。基于色温传感器的名称判断当前轮询的色温传感器的名称是否与色温传感器名称列表(color_sensor_list)中的色温传感器的名称匹配,如果匹配转至步骤S610,将当前色温传感器对应的I2C地址存储至数组寄存器地址(reg-addr)中。否则,转至步骤S612,继续轮询判断其是否匹配下一个色温传感器的名称。

[0173] S610:将匹配的色温传感器的I2C地址存储至数组reg-addr中。

[0174] 在一些实施例中,在各函数指针组中的色温传感器名称基于DTSI文件获取的色温传感器名称进行匹配,获取对应的待选函数指针组,并将基于DTSI设备树文件获取的色温传感器名称对应的I2C地址赋值到对应的待选函数指针组的寄存器地址reg-addr中。

[0175] S611:更新数组的地址列表。

[0176] 在一些实施例中,可以理解,函数指针组即是一种数组,初始的数组的寄存器地址reg-addr中的赋值都为0或其他默认数值,当初步确定当前色温传感器为电子设备项目中能够搭载的色温传感器,可以更新数组reg-addr,将通过解析DTSI设备树文件获取的I2C通信地址赋值至reg-addr中。

[0177] S612:更新数组的名称列表。

[0178] 在一些实施例中,可以理解,当初步确定当前轮询的色温传感器的名称与色温传感器名称列表,可以更新当前色温传感器对应的名称。例如,当前对色温传感器S3进行探测,轮询上述步骤S602中进行初始化的数组,判断是否当前色温传感器名称是否与初始化数组中的色温传感器的名称匹配的数组,若有,则将当前数组中色温传感器的名称更新为当前所匹配的色温传感器的名称。否则,不更新数组的名称列表。

[0179] 可以理解,在另一些实施例中,当通过轮询数组判断数组与当前色温传感器的名称不匹配,则可以返回步骤S609,继续轮询下一个色温传感器的名称是否可以与数组中色

温传感器的名称匹配。

[0180] S613:基于get_chip_id函数判断通信是否正常。

[0181] 在一些实施例中,可以通过当前探测的可搭载器件对应的关键函数(get_chip_id)基于对应的I2C地址读取实际搭载的电子器件的id(名称),当能够读取到id,且读取到的id与函数指针组中的id匹配,则可以判断正在探测的可搭载电子器件为实际搭载的电子器件且通信正常,并将当前正在探测的可搭载电子器件作为实际搭载的电子器件。

[0182] 可以理解,chip_id可以为色温传感器类型的一种编码。通过get_chip_id函数可以判断当前色温传感器的名称是否匹配,以及I2C通信是否正常,并且监测返回值,通过返回值来确认当前探测的电子器件是否是电子设备实际搭载的电子器件。例如,当返回值为“true”时,确定当前探测的电子器件是否是电子设备实际搭载的电子器件,并保存当前电子器件的对应的数组下标,即电子器件名称匹配且通信正常时,转至步骤S615,保存当前color_driver_ops实例。否则,转至步骤S614,更新数组下标。

[0183] S614:更新数组下标。

[0184] 在一些实施例中,当get_chip_id函数判断出当前色温传感器通信异常,回步骤S613,继续基于get_chip_id函数去匹配下一个色温传感器的I2C是否通信正常,若下一个色温传感器的名称与数组中对应色温传感器的名称匹配且I2C通信正常,则将数组下标对应更新为对应通信正常的色温传感器的数组下标。

[0185] S615:保存当前color_driver_ops实例和数组下标。

[0186] 在一些实施例中,当get_chip_id函数判断出当前色温传感器的名称匹配且I2C通信正常时,保存当前color_driver_ops数组实例,即将数组下标保存。例如,确定电子设备搭载的是当前色温传感器TCS3408,其对应的color_driver_ops数组的下标为3,则将当前色温传感器TCS3408的数组3保存,并基于数组3中的函数对色温传感器TCS3408进行调用和驱动,使得色温传感器TCS3408可以与上层连接并实现对应功能。

[0187] S616:初始化环境变量。

[0188] 在一些实施例中,可以理解,环境变量是一个字符串指针数组,当系统加载模块或者为该模块创建实例的时候,例如,在KO模块保存色温传感器TCS3408的数组,系统会自动为KO模块分配色温传感器TCS3408的内存空间,并在分配内存后,自动对环境变量进行初始化,然后利用环境变量初始化硬件、启动操作系统等。

[0189] S617:调用当前color_driver_ops数组的probe函数。

[0190] 在一些实施例中,调用上述保存的color_driver_ops数组3的probe函数,以实现对该color_driver_ops数组3对应的色温传感器进行探测,例如,可以调用所保存的color_driver_ops数组3中的probe探测函数、可以基于remove移除函数对色温传感器进行内存资源的清理、可以基于suspend管理电源休眠和基于resume管理电源恢复。

[0191] 基于此,在进行器件探测时,既对I2C通信是否异常进行判断,也判断电子器件的名称是否匹配,可以提高探测的精确度,避免出现存在一种由于一些可兼容的电子器件可能接线方式相同,导致出现当电子设备实际搭载的电子器件为器件B,在探测器件A时,也探测出通讯情况正常,误判器件A是实际搭载的器件的情况。

[0192] 并且在一些实施例中,函数指针组还包括各电子器件对应的probe、remove、suspend和resume函数等功能函数,在确定出实际搭载电子器件后,可以直接基于函数指针

组中的probe函数进行器件的功能绑定和初始化流程，此外，remove函数可以用于在内核层中移除电子器件时清理相应的内存资源；suspend函数可以管理电源管理模块休眠；resume函数可以管理电源恢复。如此，将器件所需使用的函数进行归一化，可以根据识别到的实际搭载器件，直接匹配函数指针组内对应的功能函数，缩短了同基线存在多个替代电子器件时，功能函数具备多个，需要重新匹配对应的功能函数导致的加载耗时。

[0193] 此外，在本申请实施例中，通过在DTSI设备树文件中配置部分获取探测结果可能使用的多个电子器件的信息，开创性的采用全局电子器件数组的形式，动态获取电子器件的配置信息，并在对电子器件进行探测时确认实际搭载的电子器件，实现了动态的电子器件识别。当正确检测出电子器件实际搭载的电子器件后，不需要再对其他电子器件再进行探测，大大改善了流程冗余问题，提高了检测效率。

[0194] 另外，对于不是电子设备实际搭载的色温传感器不会通过公共节点入口直接被注册到电源管理模块中，避免出现在电源管理模块中存在悬空的虚拟器件造成电源管理模块对虚拟器件的反复挂起和恢复的操作，即解决了上述方案中提及的资源浪费问题。

[0195] 图7示出了根据本申请的一些实施例提供的电子设备(以本申请所述的手机10为例)的系统示意图。

[0196] 可以理解，本发明实施例示意的结构并不构成对手机10的具体限定。在本申请另一些实施例中，手机10可以包括比图示更多或更少的部件，或者组合某些部件，或者拆分某些部件，或者不同的部件布置。图示的部件可以以硬件，软件或软件和硬件的组合实现。

[0197] 手机10可以包括处理器110，外部存储器接口120，内部存储器121，通用串行总线(universal serial bus，USB)接头130，充电管理模块140，电源管理模块141，电池142，天线1，天线2，移动通信模块150，无线通信模块160，音频模块170，扬声器170A，受话器170B，麦克风170C，耳机接口170D，传感器模块180，按键190，马达191，指示器192，摄像头193，显示屏194，以及用户标识模块(subscriber identification module，SIM)卡接口195等。其中传感器模块180可以包括压力传感器180A，陀螺仪传感器180B，气压传感器180C，磁传感器180D，加速度传感器180E，距离传感器180F，接近光传感器180G，指纹传感器180H，温度传感器180J，触摸传感器180K，环境光传感器180L，骨传导传感器180M，色温传感器180N等。

[0198] 处理器110可以包括一个或多个处理单元，例如：处理器110可以包括应用处理器(AP)，调制解调处理器，图形处理器(graphics processing unit，GPU)，图像信号处理器(image signal processor，ISP)，控制器，视频编解码器，数字信号处理器(digital signal processor，DSP)，基带处理器(BP)，和/或神经网络处理器(neural-network processing unit，NPU)等。其中，不同的处理单元可以是独立的器件，也可以集成在一个或多个处理器中。

[0199] 处理器110中还可以设置存储器，用于存储指令和数据。

[0200] 充电管理模块140用于从充电器检测到充电输入。其中，充电器可以是无线充电器，也可以是有线充电器。在一些实施例中，充电管理模块140可以通过USB接口130检测到有线充电器的充电输入。在另一些实施例中，充电管理模块140可以通过手机10的无线充电线圈检测到无线充电输入。充电管理模块140为电池142充电的同时，还可以通过电源管理模块141为电子设备供电。

[0201] 电源管理模块141用于连接电池142，充电管理模块140与处理器110。电源管理模

块141检测到电池142和/或充电管理模块140的输入，为处理器110，内部存储器121，外部存储器120，显示屏194，摄像头193，和无线通信模块160等供电。电源管理模块141还可以用于监测电池容量，电池循环次数，电池健康状态(漏电，阻抗)等参数。在一些实施例中，电源管理模块141也可以设置于处理器110中。在一些实施例中，电源管理模块141和充电管理模块140也可以设置于同一个器件中。在一些实施例中，电源管理模块141可以为本申请实施例中电子器件的配置进行供电配置。

[0202] 手机10的无线通信功能可以通过天线1，天线2，移动通信模块150，无线通信模块160，调制解调处理器以及基带处理器等实现。

[0203] 天线1和天线2用于发射和检测到电磁波信号。手机10中的每个天线可用于覆盖单个或多个通信频带。不同的天线还可以复用，以提高天线的利用率。例如：可以将天线1复用为无线局域网的分集天线。在另外一些实施例中，天线可以和调谐开关结合使用。

[0204] 移动通信模块150可以提供应用在手机10上的包括2G/3G/4G/5G等无线通信的解决方案。

[0205] 调制解调处理器可以包括调制器和解调器。其中，调制器用于将待发送的低频基带信号调制成中高频信号。解调器用于将检测到的电磁波信号解调为低频基带信号。随后解调器将解调得到的低频基带信号传送至基带处理器处理。低频基带信号经基带处理器处理后，被传递给应用处理器。

[0206] 无线通信模块160可以提供应用在手机10上的包括无线局域网(wireless local area networks,WLAN)(如无线保真(wireless fidelity,Wi-Fi)网络)，蓝牙(bluetooth，BT)，全球导航卫星系统(global navigation satellite system,GNSS)，调频(frequency modulation,FM)，近距离无线通信技术(near field communication,NFC)，红外技术(infrared,IR)等无线通信的解决方案。

[0207] 手机10通过GPU，显示屏194，以及应用处理器等实现显示功能。GPU为图像处理的微处理器，连接显示屏194和应用处理器。GPU用于执行数学和几何计算，用于图形渲染。处理器110可包括一个或多个GPU，其执行程序指令以生成或改变显示信息。

[0208] 显示屏194用于显示图像，视频等。显示屏194包括显示面板。

[0209] 手机10可以通过ISP，摄像头193，视频编解码器，GPU，显示屏194以及应用处理器等实现拍摄功能。

[0210] ISP用于处理摄像头193反馈的数据。

[0211] 摄像头193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。

[0212] 数字信号处理器用于处理数字信号，除了可以处理数字图像信号，还可以处理其他数字信号。例如，当手机10在频点选择时，数字信号处理器用于对频点能量进行傅里叶变换等。

[0213] 外部存储器120一般指外存储器，常见的外部存储器有硬盘、软盘、光盘、U盘、Micro SD卡等，用于实现扩展手机10的存储能力。外部存储器可以通过外部存储器接口或者总线与处理器110通信，实现数据存储功能。

[0214] 内部存储器121，也可以称为“内存”，可以用于存储计算机可执行程序代码，可执行程序代码包括指令。

[0215] 手机10可以通过音频模块170，扬声器170A，受话器170B，麦克风170C，耳机接口170D，以及应用处理器等实现音频功能。例如音乐播放，录音等。

[0216] 压力传感器180A用于感受压力信号，可以将压力信号转换成电信号。

[0217] 陀螺仪传感器180B可以用于确定手机10的运动姿态。在一些实施例中，可以通过陀螺仪传感器180B确定手机10围绕三个轴(即，x，y和z轴)的角速度。

[0218] 气压传感器180C用于测量气压。

[0219] 磁传感器180D包括霍尔传感器。

[0220] 加速度传感器180E可检测手机10在各个方向上(一般为三轴)加速度的大小。

[0221] 距离传感器180F，用于测量距离。

[0222] 接近光传感器180G可以包括例如发光二极管(LED)和光检测器，例如光电二极管。发光二极管可以是红外发光二极管。

[0223] 环境光传感器180L用于感知环境光亮度。手机10可以根据感知的环境光亮度自适应调节显示屏194亮度。

[0224] 指纹传感器180H用于采集指纹。手机10可以利用采集的指纹特性实现指纹解锁，访问应用锁，指纹拍照，指纹接听来电等。

[0225] 温度传感器180J用于检测温度。

[0226] 触摸传感器180K，也称“触控面板”。触摸传感器180K可以设置于显示屏194，由触摸传感器180K与显示屏194组成触摸屏，也称“触控屏”。

[0227] 骨传导传感器180M可以获取振动信号。

[0228] 色温传感器180N可以用于检测光源颜色或色温。也可以用于根据光源的颜色或色温，传递相应的电信号，使得电信号转换后对应的二进制信息可以被机器识别。在一些实施例中，色温传感器可以根据环境的色温来自动调节电子设备的屏幕色温，以及可以在电子设备拍摄时用于探测环境的色温，使得所拍摄视频或图像色彩更准确，或者对图像进行颜色校准等。

[0229] 按键190包括开机键，音量键等。按键190可以是机械按键。也可以是触摸式按键。手机10可以检测到按键输入，产生与手机10的用户设置以及功能控制有关的键信号输入。

[0230] 马达191可以产生振动提示。马达191可以用于来电振动提示，也可以用于触摸振动反馈。

[0231] 指示器192可以是指示灯，可以用于指示充电状态，电量变化，也可以用于指示消息，未接来电，通知等。

[0232] SIM卡接口195用于连接SIM卡。

[0233] 本申请的各方法实施方式均可以以软件、磁件、固件或这些实现方法的组合中。

[0234] 可将程序代码应用于输入指令，以执行本文描述的各功能并生成输出数据。可以按已知方式将输出数据应用于一个或多个输出设备。

[0235] 程序代码可以用高级程序化语言或面向对象的编程语言来实现，以便与处理系统通信。在需要时，也可用汇编语言或机器语言来实现程序代码。事实上，本文中描述的机制不限于任何特定编程语言的范围。在任一情形下，该语言可以是编译语言或解释语言。

[0236] 至少一个实施例的一个或多个方面可以由存储在计算机可读存储介质上的表示性指令来实现，指令表示处理器中的各种逻辑，指令在被机器读取时使得该机器制作用于

执行本文所述的技术的逻辑。被称为“IP核”的这些表示可以被存储在有形的计算机可读存储介质上，并被提供给多个客户或生产设施以加载到实际制造该逻辑或处理器的制造机器中。

[0237] 虽然本申请的描述将结合较佳实施例一起介绍，但这并不代表此申请的特征仅限于该实施方式。恰恰相反，结合实施方式作发明介绍的目的是为了覆盖基于本申请的权利要求而有可能延伸出的其它选择或改造。为了提供对本申请的深度了解，以下描述中将包含许多具体的细节。本申请也可以不使用这些细节实施。此外，为了避免混乱或模糊本申请的重点，有些具体细节将在描述中被省略。需要说明的是，在不冲突的情况下，本申请中的实施例及实施例中的特征可以相互组合。

[0238] 此外，各种操作将以最有助于理解说明性实施例的方式被描述为多个离散操作；然而，描述的顺序不应被解释为暗示这些操作必须依赖于顺序。特别是，这些操作不需要按呈现顺序执行。

[0239] 如这里所使用的，术语“模块”或“单元”可以指代、是或者包括：专用集成电路(ASIC)、电子电路、执行一个或多个软件或固件程序的(共享、专用或组)处理器和/或存储器、组合逻辑电路和/或提供所描述的功能的其他合适的组件。

[0240] 在附图中，以特定布置和/或顺序示出一些结构或方法特征。然而，应该理解，可以不需要这样的特定布置和/或排序。在一些实施例中，这些特征可以以不同于说明性附图中所示的方式和/或顺序来布置。另外，在特定图中包含结构或方法特征并不意味着暗示在所有实施例中都需要这样的特征，并且在一些实施例中，可以不包括这些特征或者可以与其他特征组合。

[0241] 本申请的实施例可实现为在可编程系统上执行的计算机程序或程序代码，该可编程系统包括多个处理器、存储系统(包括易失性和非易失性存储器和/或存储元件)、多个输入设备以及多个输出设备。

[0242] 在一些情况下，所公开的实施例可以以硬件、固件、软件或其任何组合来实现。在一些情况下，至少一些实施例的一个或多个方面可以由存储在计算机可读存储介质上的表示性指令来实现，指令表示处理器中的各种逻辑，指令在被机器读取时使得该机器制作用于执行本申请所述的技术的逻辑。被称为“IP核”的这些表示可以被存储在有形的计算机可读存储介质上，并被提供给多个客户或生产设施以加载到实际制造该逻辑或处理器的制造机器中。

[0243] 因此，本申请的各实施例还包括非瞬态的计算机可读存储介质，该介质包含指令或包含设计数据，诸如硬件描述语言(HDL)，它定义本申请中描述的结构、电路、装置、处理器和/或系统特征。

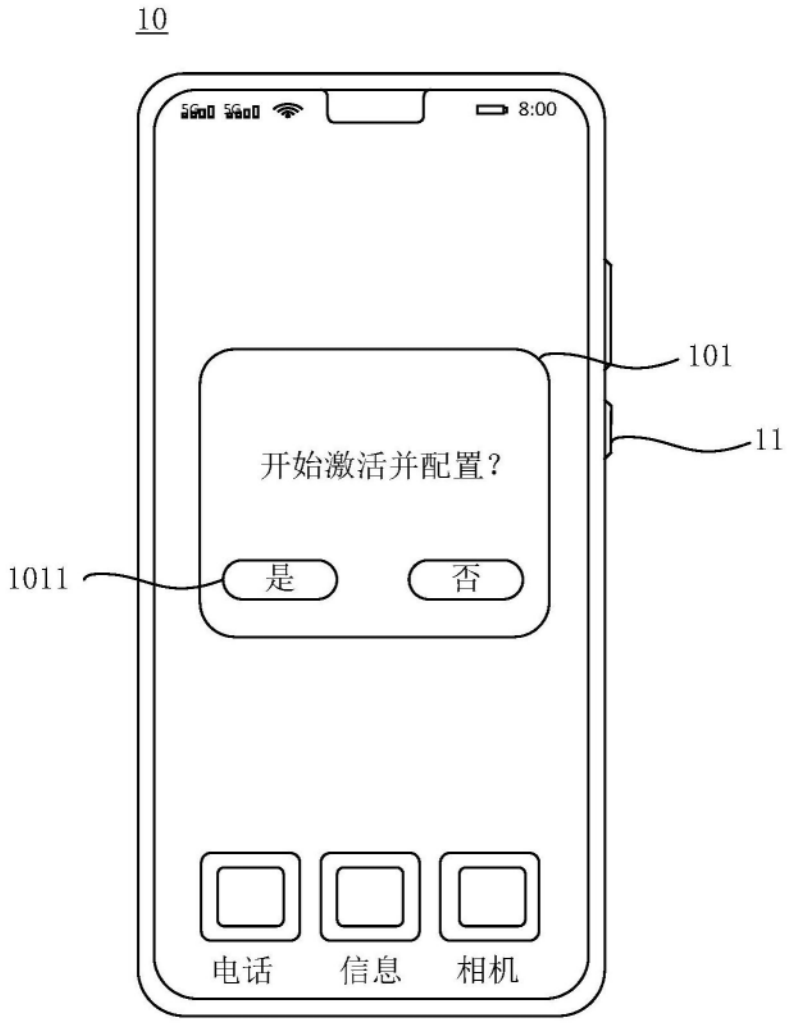


图1

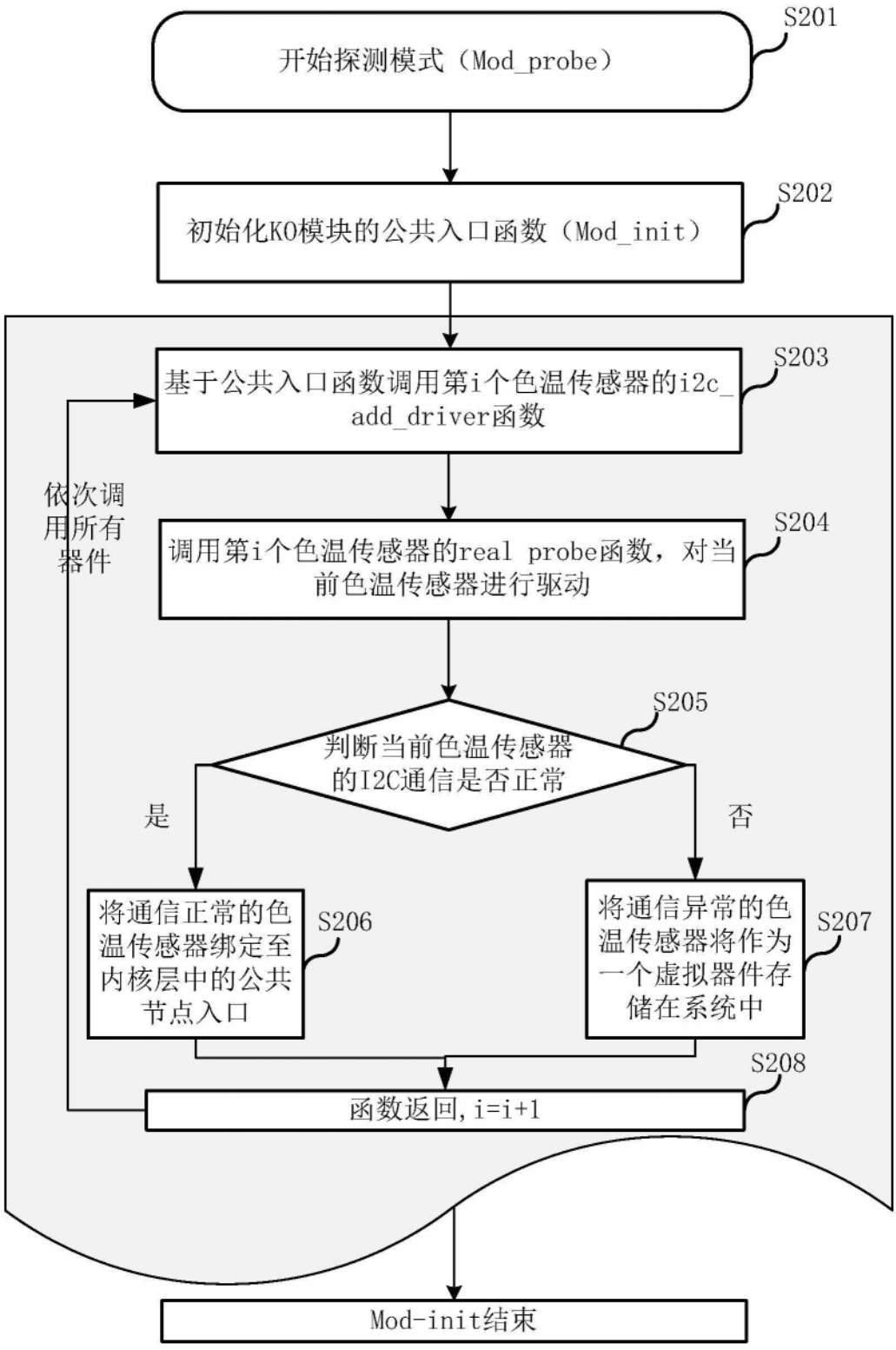

S201
开始探测模式（Mod_probe）
S202
初始化KO模块的公共入口函数（Mod_init）
S203
基于公共入口函数调用第i个色温传感器的i2c_add_driver函数
依次调用所有器件
S204
调用第i个色温传感器的real probe函数，对当前色温传感器进行驱动
S205
判断当前色温传感器的I2C通信是否正常
是
否
将通信正常的色温传感器绑定至内核层中的公共节点入口
S206
将通信异常的色温传感器将作为一个虚拟器件存储在系统中
S207
S208
函数返回,i=i+1
Mod-init结束

图2

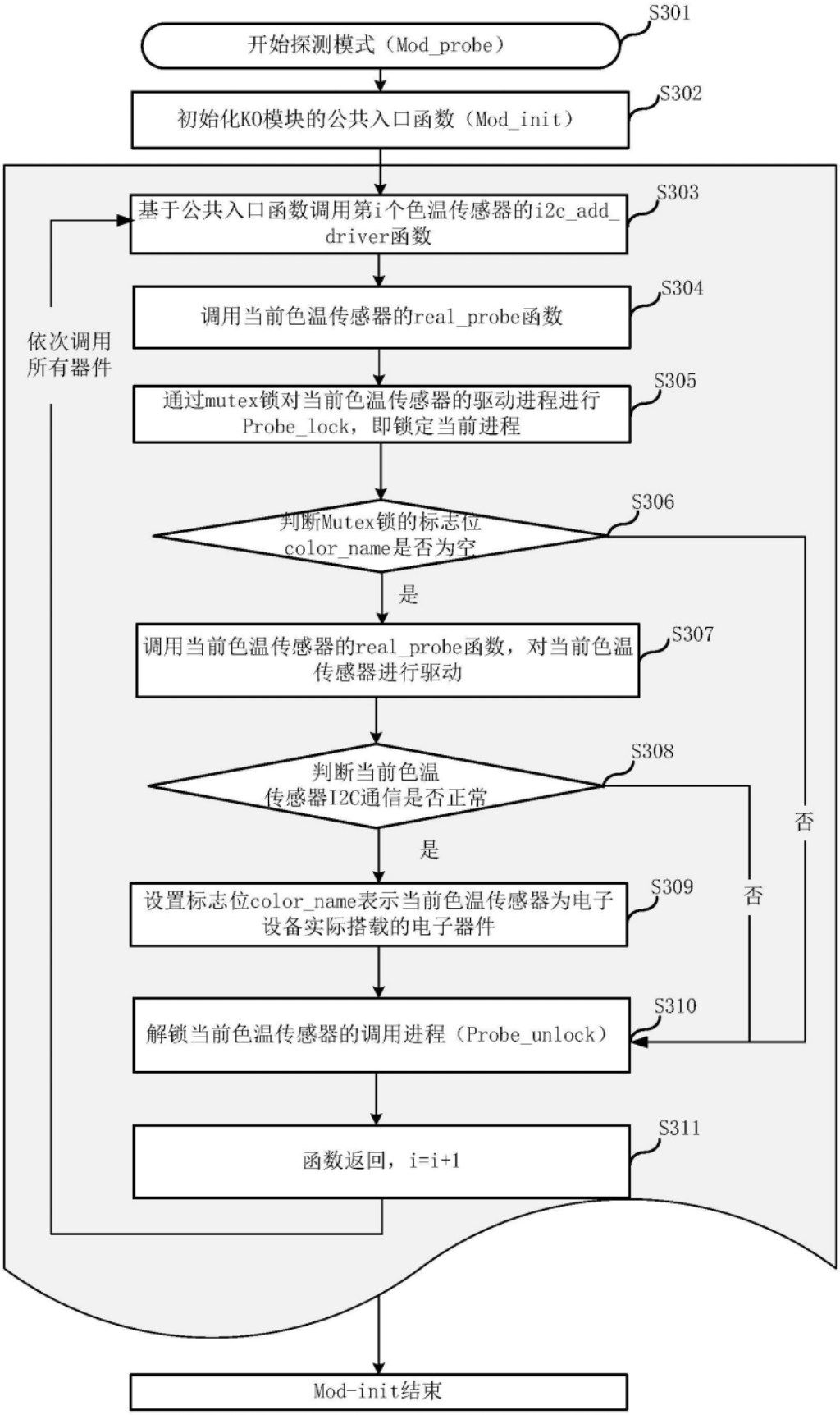

S301
开始探测模式（Mod_probe）
S302
初始化KO模块的公共入口函数（Mod_init）
S303
基于公共入口函数调用第i个色温传感器的i2c_add_driver函数
S304
调用当前色温传感器的real_probe函数
S305
通过mutex锁对当前色温传感器的驱动进程进行Probe_lock，即锁定当前进程
S306
判断Mutex锁的标志位color_name是否为空
是
否
S307
调用当前色温传感器的real_probe函数，对当前色温传感器进行驱动
S308
判断当前色温传感器I2C通信是否正常
是
否
S309
设置标志位color_name表示当前色温传感器为电子设备实际搭载的电子器件
S310
解锁当前色温传感器的调用进程（Probe_unlock）
S311
函数返回，i=i+1
依次调用所有器件
Mod-init结束

图3

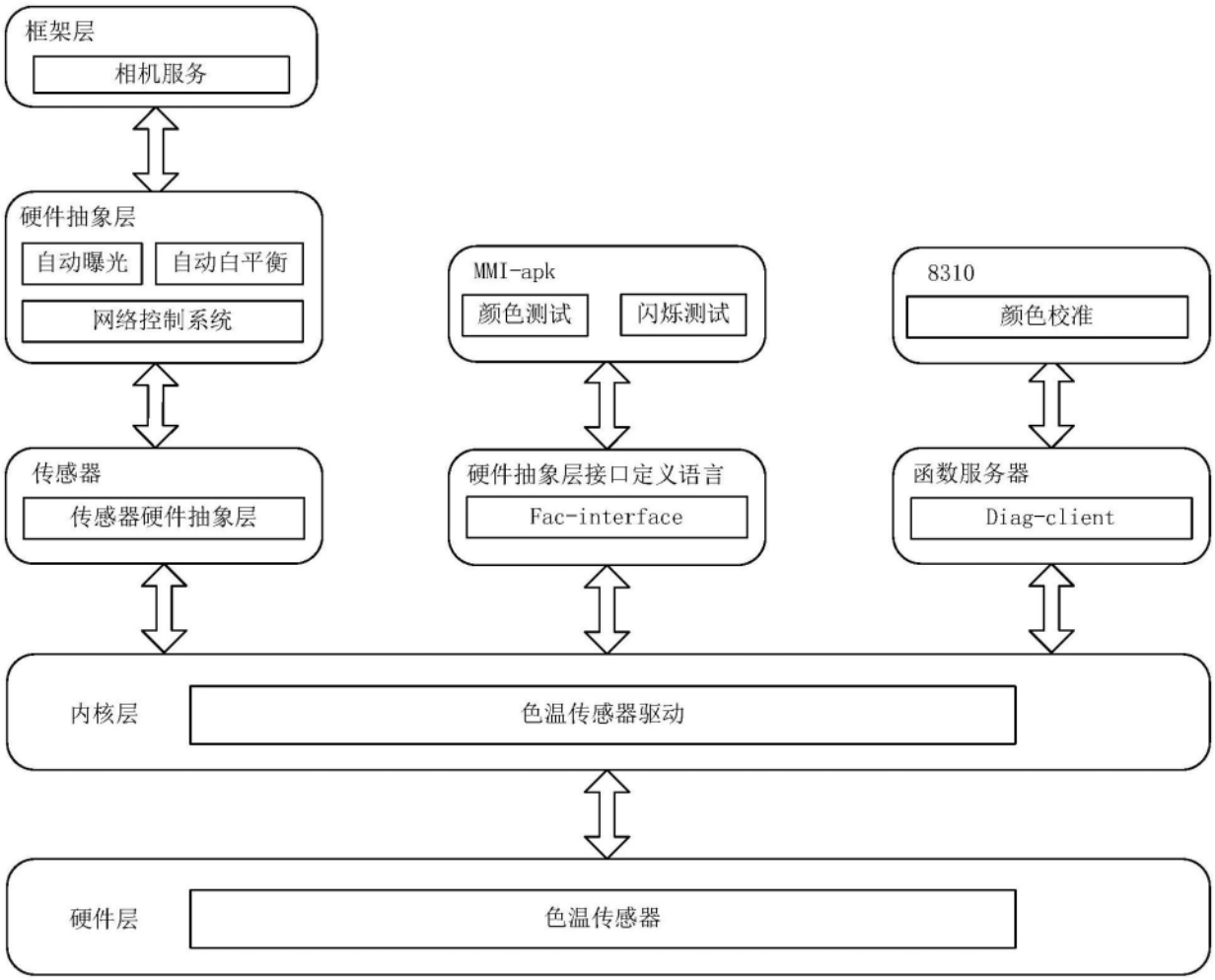

框架层
相机服务
硬件抽象层
自动曝光
自动白平衡
网络控制系统
MMI-apk
颜色测试
闪烁测试
8310
颜色校准
传感器
传感器硬件抽象层
硬件抽象层接口定义语言
Fac-interface
函数服务器
Diag-client
内核层
色温传感器驱动
硬件层
色温传感器

图4A

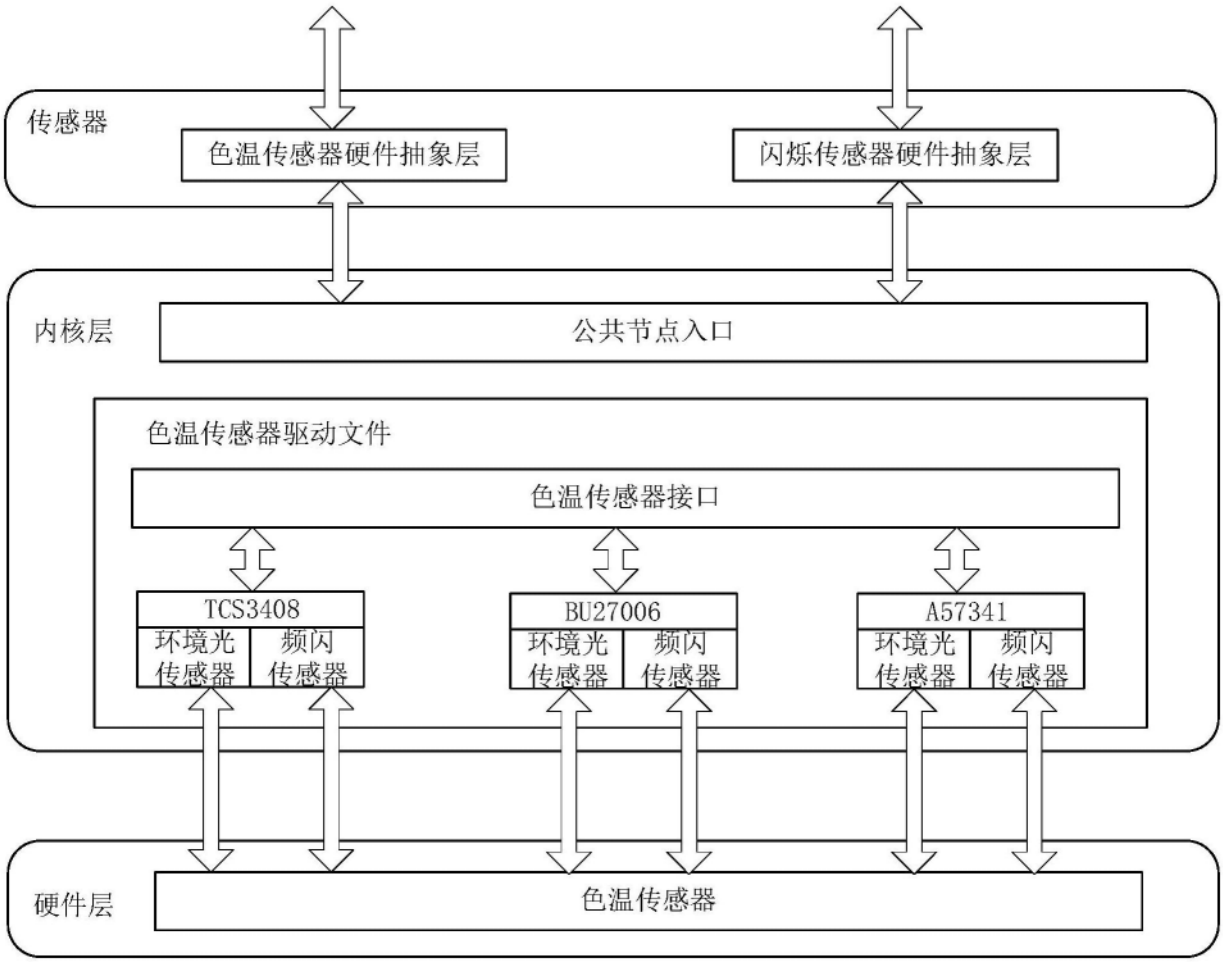

传感器
色温传感器硬件抽象层
闪烁传感器硬件抽象层
内核层
公共节点入口
色温传感器驱动文件
色温传感器接口
TCS3408
环境光传感器
频闪传感器
BU27006
环境光传感器
频闪传感器
A57341
环境光传感器
频闪传感器
硬件层
色温传感器

图4B

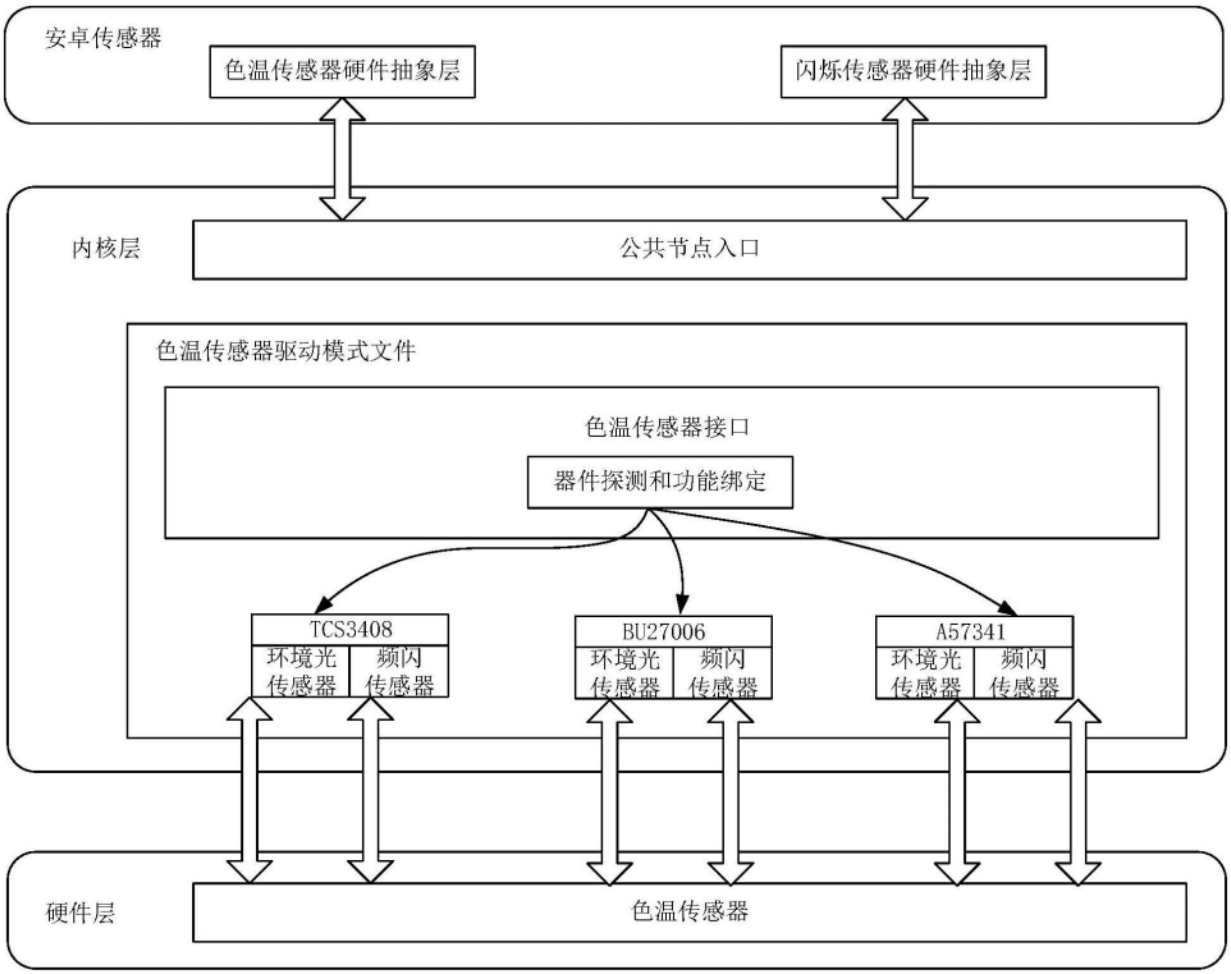

安卓传感器
色温传感器硬件抽象层
闪烁传感器硬件抽象层
内核层
公共节点入口
色温传感器驱动模式文件
色温传感器接口
器件探测和功能绑定
TCS3408
环境光传感器
频闪传感器
BU27006
环境光传感器
频闪传感器
A57341
环境光传感器
频闪传感器
硬件层
色温传感器

图4C

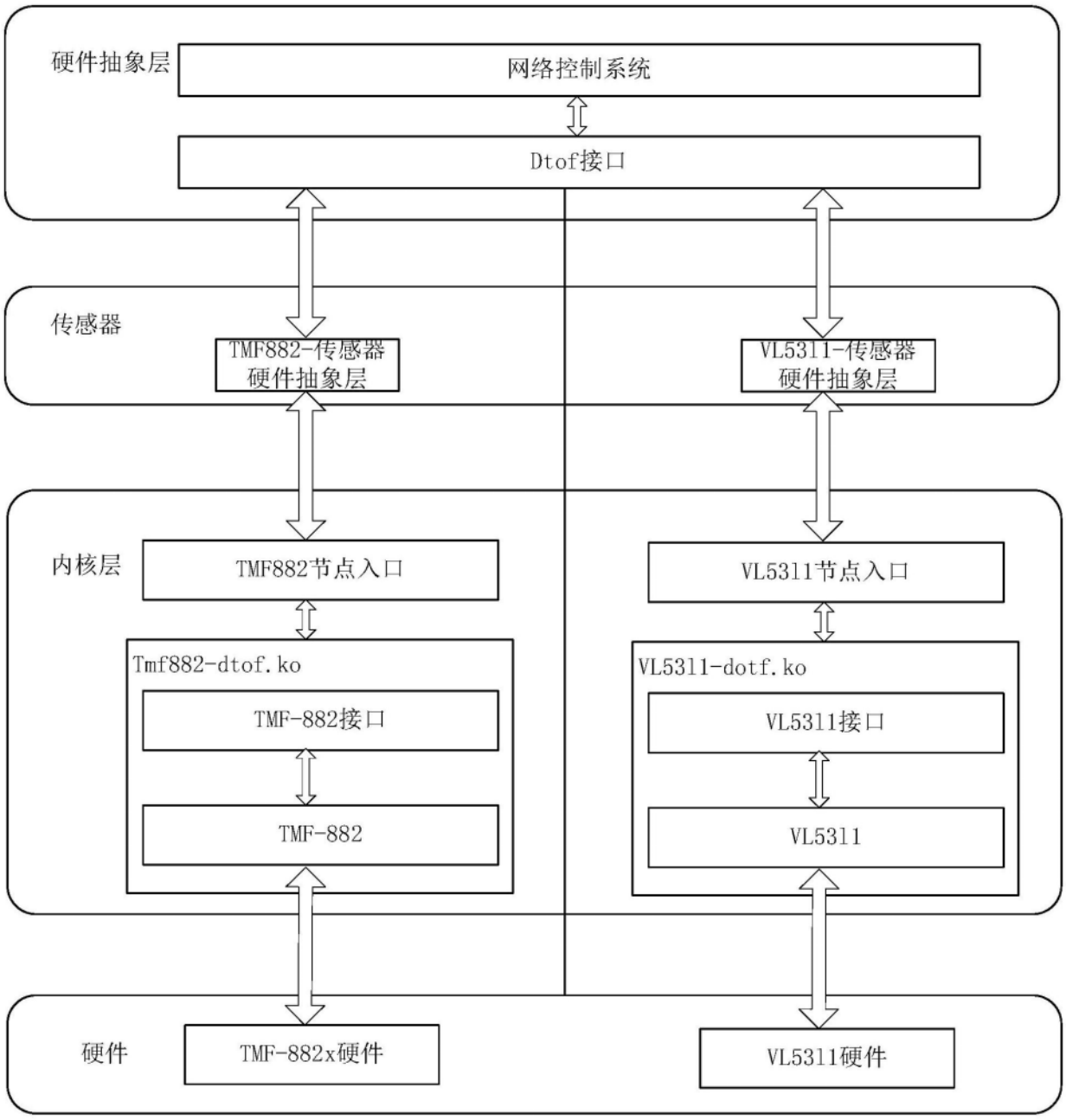

硬件抽象层
网络控制系统
Dtof接口
传感器
TMF882-传感器
硬件抽象层
VL5311-传感器
硬件抽象层
内核层
TMF882节点入口
VL5311节点入口
Tmf882-dtof.ko
TMF-882接口
TMF-882
VL5311-dotf.ko
VL5311接口
VL5311
硬件
TMF-882x硬件
VL5311硬件

图4D

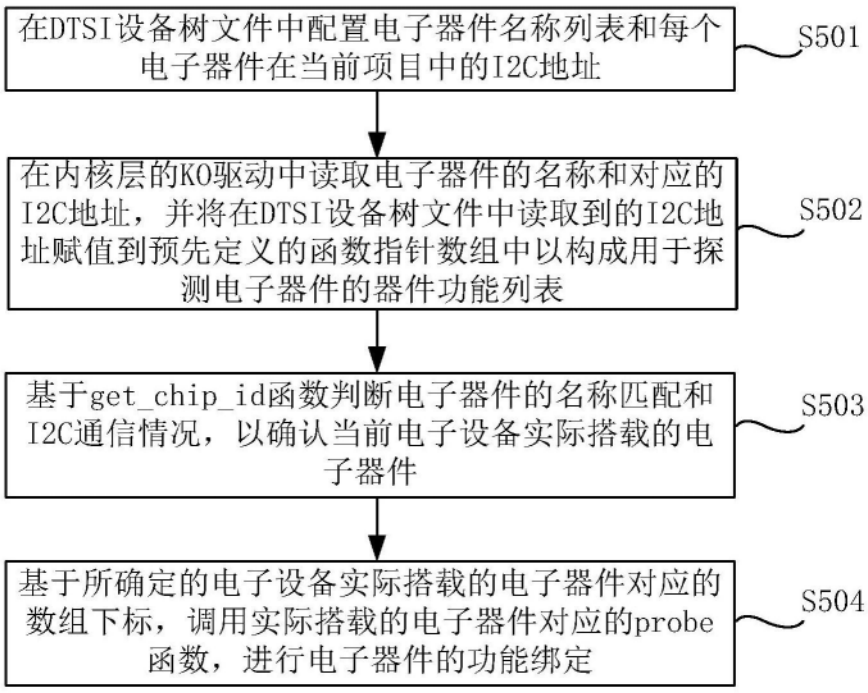

在DTSI设备树文件中配置电子器件名称列表和每个电子器件在当前项目中的I2C地址
S501
在内核层的KO驱动中读取电子器件的名称和对应的I2C地址，并将在DTSI设备树文件中读取到的I2C地址赋值到预先定义的函数指针数组中以构成用于探测电子器件的器件功能列表
S502
基于get_chip_id函数判断电子器件的名称匹配和I2C通信情况，以确认当前电子设备实际搭载的电子器件
S503
基于所确定的电子设备实际搭载的电子器件对应的数组下标，调用实际搭载的电子器件对应的probe函数，进行电子器件的功能绑定
S504

图5

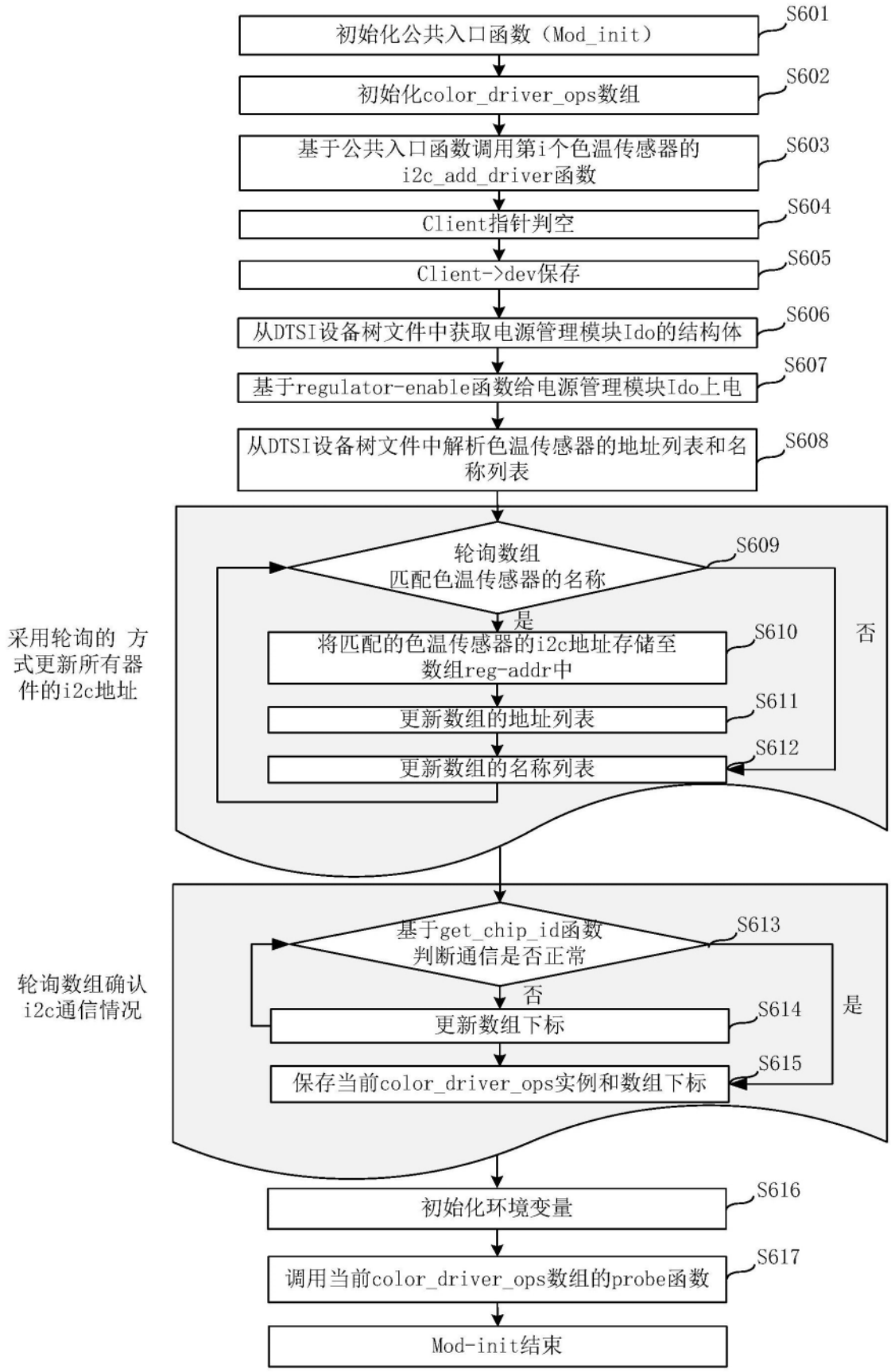

初始化公共入口函数（Mod_init）
S601
初始化color_driver_ops数组
S602
基于公共入口函数调用第i个色温传感器的i2c_add_driver函数
S603
Client指针判空
S604
Client->dev保存
S605
从DTSI设备树文件中获取电源管理模块Ido的结构体
S606
基于regulator-enable函数给电源管理模块Ido上电
S607
从DTSI设备树文件中解析色温传感器的地址列表和名称列表
S608
采用轮询的 方式更新所有器件的i2c地址
轮询数组匹配色温传感器的名称
S609
是
否
将匹配的色温传感器的i2c地址存储至数组reg-addr中
S610
更新数组的地址列表
S611
更新数组的名称列表
S612
轮询数组确认i2c通信情况
基于get_chip_id函数判断通信是否正常
S613
否
是
更新数组下标
S614
保存当前color_driver_ops实例和数组下标
S615
初始化环境变量
S616
调用当前color_driver_ops数组的probe函数
S617
Mod-init结束

图6

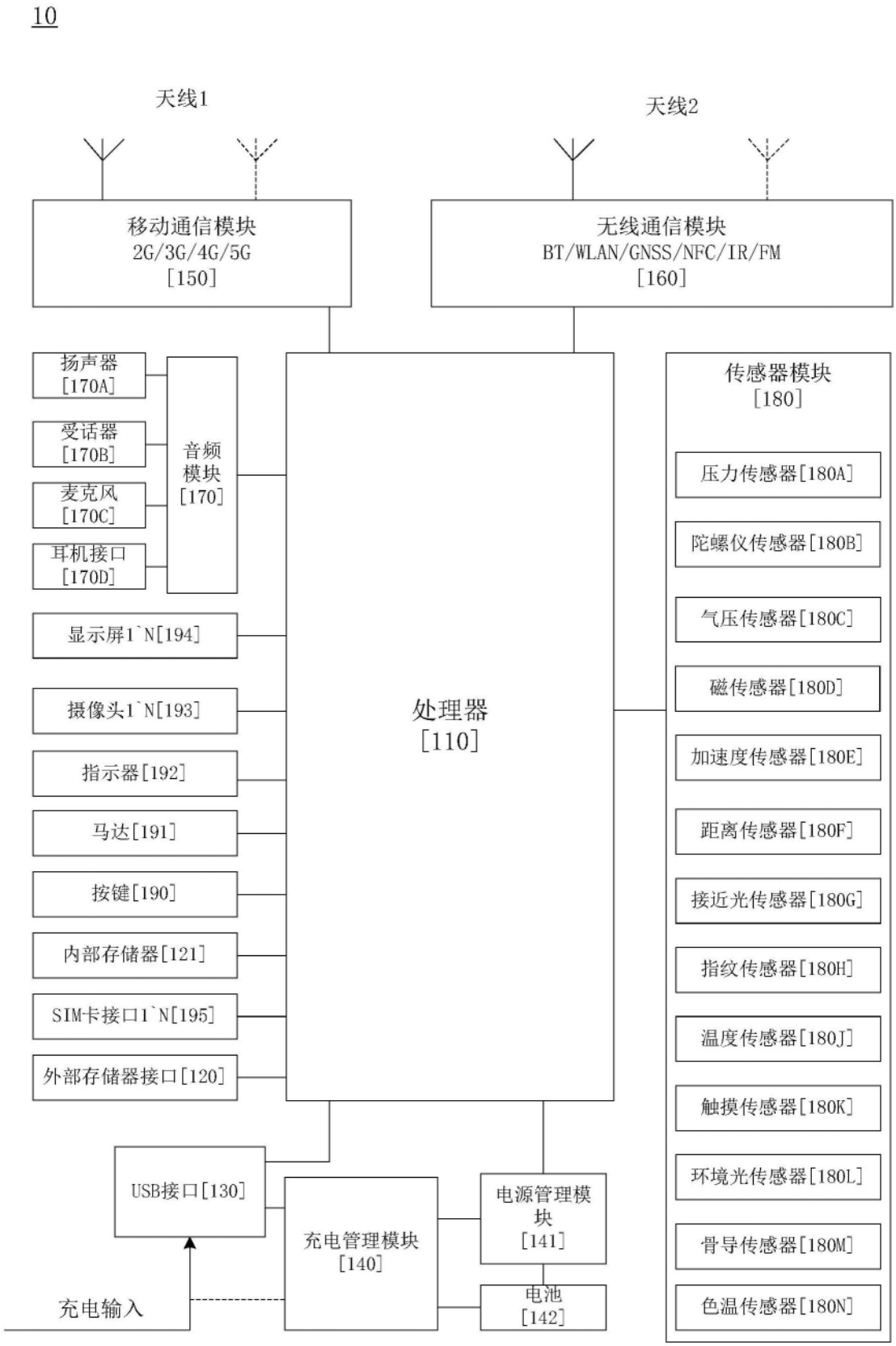

10
天线1
天线2
移动通信模块
2G/3G/4G/5G
[150]
无线通信模块
BT/WLAN/GNSS/NFC/IR/FM
[160]
扬声器
[170A]
受话器
[170B]
麦克风
[170C]
耳机接口
[170D]
音频
模块
[170]
显示屏1`N[194]
摄像头1`N[193]
指示器[192]
马达[191]
按键[190]
内部存储器[121]
SIM卡接口1`N[195]
外部存储器接口[120]
处理器
[110]
传感器模块
[180]
压力传感器[180A]
陀螺仪传感器[180B]
气压传感器[180C]
磁传感器[180D]
加速度传感器[180E]
距离传感器[180F]
接近光传感器[180G]
指纹传感器[180H]
温度传感器[180J]
触摸传感器[180K]
环境光传感器[180L]
骨导传感器[180M]
色温传感器[180N]
USB接口[130]
充电管理模块
[140]
电源管理模
块
[141]
电池
[142]
充电输入

图7