(12) 按照专利合作条约所公布的国际申请

(19) 世界知识产权组织
国 际 局

(43) 国际公布日
2023 年 2 月 2 日 (02.02.2023)

WIPO | PCT

(10) 国际公布号
WO 2023/005371 A1

(51) 国际专利分类号:
G06F 9/4401 (2018.01) *G06F 8/654* (2018.01)

(21) 国际申请号: PCT/CN2022/094159

(22) 国际申请日: 2022 年 5 月 20 日 (20.05.2022)

(25) 申请语言: 中文

(26) 公布语言: 中文

(30) 优先权:
202110872082.9 2021年7月30日 (30.07.2021) CN

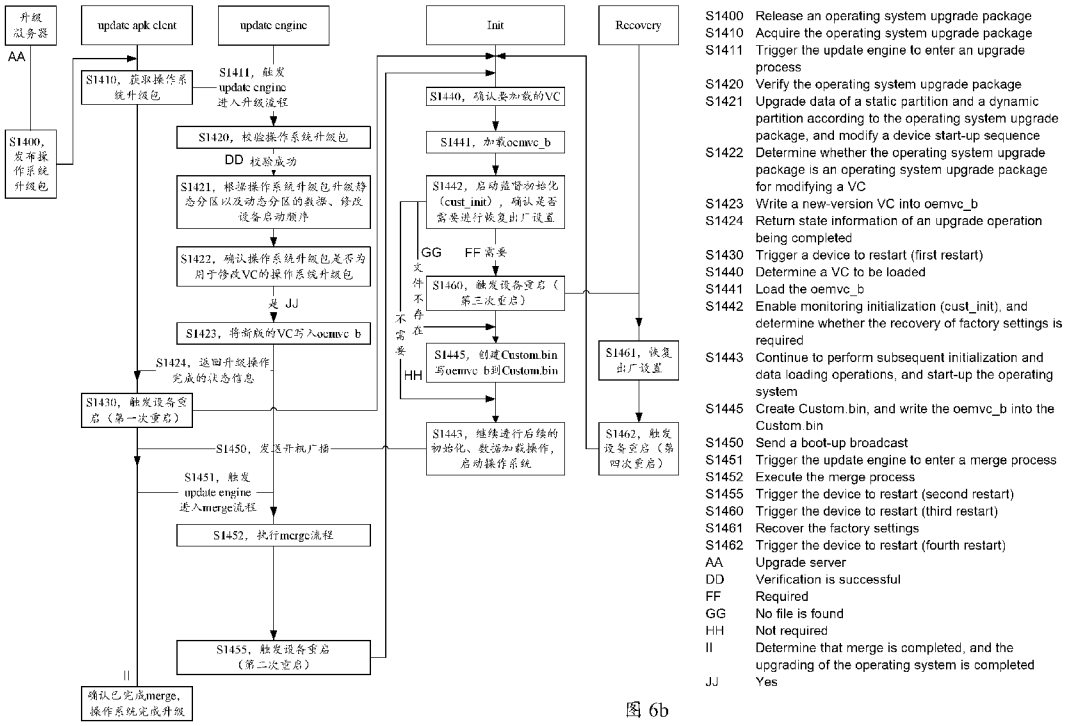
(71) 申请人: 荣耀终端有限公司(**HONOR DEVICE CO., LTD.**) [CN/CN]; 中国广东省深圳市福田区香蜜湖街道红荔西路8089号深业中城6号楼A单元3401, Guangdong 518040 (CN)。

(72) 发明人: **王艳召(WANG, Yanzhao)**; 中国广东省深圳市福田区香蜜湖街道红荔西路8089号深业中城6号楼A单元3401, Guangdong 518040 (CN)。 **陈超(CHEN, Chao)**; 中国广东省深圳市福田区香蜜湖街道红荔西路8089号深业中城6号楼A单元3401, Guangdong 518040 (CN)。 **张赠辉(ZHANG, Zenghui)**; 中国广东省深圳市福田区香蜜湖街道红荔西路8089号深业中城6号楼A单元3401, Guangdong 518040 (CN)。 **黄九林(HUANG, Jiulin)**; 中国广东省深圳市福田区香蜜湖街道红荔西路8089号深业中城6号楼A单元3401, Guangdong 518040 (CN)。

(74) 代理人: **北京汇思诚业知识产权代理有限公司(UNI-INTEL PATENT AND TRADEMARK LAW FIRM)**; 中国北京市朝阳区建国门外

(54) **Title:** METHOD FOR CONFIGURING FORMAT OF OPERATING SYSTEM, AND DEVICE AND STORAGE MEDIUM

(54) **发明名称:** 配置操作系统制式的方法、设备及存储介质



S1400 Release an operating system upgrade package
S1410 Acquire the operating system upgrade package
S1411 Trigger the update engine to enter an upgrade process
S1420 Verify the operating system upgrade package
S1421 Upgrade data of a static partition and a dynamic partition according to the operating system upgrade package, and modify a device start-up sequence
S1422 Determine whether the operating system upgrade package is an operating system upgrade package for modifying a VC
S1423 Write a new-version VC into oemvc_b
S1424 Return state information of an upgrade operation being completed
S1430 Trigger a device to restart (first restart)
S1440 Determine a VC to be loaded
S1441 Load the oemvc_b
S1442 Enable monitoring initialization (cust_init), and determine whether the recovery of factory settings is required
S1443 Continue to perform subsequent initialization and data loading operations, and start-up the operating system
S1445 Create Custom.bin, and write the oemvc_b into the Custom.bin
S1450 Send a boot-up broadcast
S1451 Trigger the update engine to enter a merge process
S1452 Execute the merge process
S1455 Trigger the device to restart (second restart)
S1460 Trigger the device to restart (third restart)
S1461 Recover the factory settings
S1462 Trigger the device to restart (fourth restart)
AA Upgrade server
DD Verification is successful
FF Required
GG No file is found
HH Not required
II Determine that merge is completed, and the upgrading of the operating system is completed
JJ Yes

图 6b

(57) **Abstract:** Provided in the embodiments of the present application are a method for configuring the format of an operating system, and a device, a storage medium and a computer program product. The method is applied to an electronic device. A memory of the electronic device at least comprises a basic partition, a first static partition and a second static partition. A first format file is loaded when start-up is performed from the first static partition; and a second format file is loaded when the start-up is performed from the second static partition. After start-up is performed from the first static partition, the method comprises: acquiring an operating system upgrade

WO 2023/005371 A1

[见续页]

大街永安东里甲3号通用国际中心A座3层, Beijing 100022 (CN)。

(81) 指定国(除另有指明，要求每一种可提供的国家保护)：AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BN, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, 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, 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(除另有指明，要求每一种可提供的地区保护)：ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, RW, SD, SL, ST, SZ, TZ, UG, ZM, ZW), 欧亚 (AM, AZ, BY, KG, KZ, RU, TJ, TM), 欧洲 (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)。

本国际公布：

— 包括国际检索报告(条约第21条(3))。

package, which comprises a third format file; writing format content of the third format file into the second format file; performing a restart to enter a recovery mode; recovering factory settings of the electronic device; and performing a restart, performing start-up from the second static partition, creating a format configuration file in an initialization stage after the second restart, and writing format content of the second format file into the format configuration file. By means of the method in the embodiments of the present application, a format change process can be greatly simplified, and the format change difficulty can be reduced.

(57) 摘要：本申请实施例提供的一种配置操作系统制式的方法、设备、存储介质及计算机程序产品，方法应用于电子设备，电子设备的存储器至少包括基础分区、第一静态分区、第二静态分区，从第一静态分区启动时加载第一制式文件；从第二静态分区启动时加载第二制式文件；从第一静态分区启动之后，方法包括：获取包括第三制式文件的操作系统升级包；将第三制式文件的制式内容写入第二制式文件；重启进入恢复模式；恢复电子设备的出厂设置；重启，从第二静态分区启动，在第二重启之后的初始化环节中，创建制式配置文件，将第二制式文件的制式内容写入制式配置文件。根据本申请实施例的方法，可以大大简化改制流程，降低改制难度。

配置操作系统制式的方法、设备及存储介质

技术领域

本申请涉及计算机技术领域，具体地涉及一种配置操作系统制式的方法、设备、存储介质及计算机程序产品。

背景技术

在现有技术的应用场景中，用户终端需要安装操作系统才可以被用户使用。例如，手机上需要安装手机操作系统（例如：IOS 系统、安卓系统）才可以被用户使用。而在无线通信领域，根据无线通信设备（例如，手机）所处的位置、接入的运营商的不同，无线通信设备的操作系统需要配置对应的制式（vendor_country，VC）；例如，all cn（通用中国制式）、cmcc cn（中国移动中国制式）等。

一般的，在无线通信设备出厂前进行初始操作系统的安装时，会根据其销售区域安装已配置好对应制式的操作系统。无线通信设备出厂后，无需更改操作系统的制式。但是，在实际应用场景中，存在需要更改无线通信设备的操作系统的制式的情况。例如，海外某国家备货原始样机 vendor_country 为 xxxx_ru（运营商 xxxx 的俄罗斯制式），因货物积压或另一个国家缺货需要调货，需要将手机制式调整为匹配的制式。因此，就需要一种更改无线通信设备的操作系统的制式的方法。

发明内容

有鉴于此，本申请提供一种配置操作系统制式的方法、设备、存储介质及计算机程序产品，以利于解决现有技术中如何更改操作系统制式的问题。

第一方面，本申请实施例提供了一种配置操作系统制式的方法，应用于电子设备，电子设备包括处理器以及存储器，存储器包括基础分区、第一静态分区、第二静态分区、动态分区以及用户数据分区；电子设备启动操作系统的过程包括初始化环节，基础分区中保存有第一制式文件以及第二制式文件，电子设备从第一静态分区启动操作系统时，初始化环节中加载第一制式文件；电子设备从第二静态分区启动操作系统时，初始化环节中加载第二制式文件；第一制式文件当前的制式内容为第一制式，电子设备启动后加载基础分区、第一静态分区以及动态分区的数据以从第一静态分区启动操作系统；操作系统启动之后，方法包括：

获取操作系统升级包，操作系统升级包包括第三制式文件，第三制式文件的制式内容为第二制式；

修改电子设备的启动顺序为从第二静态分区启动；

提取第三制式文件；

将第三制式文件的制式内容写入第二制式文件；

触发电子设备的第一重启，

第一重启之后电子设备进入恢复模式；

在恢复模式下，恢复电子设备的出厂设置，包括，删除用户数据分区中保存的制式配置文件；

触发电子设备的第二重启，第二重启之后电子设备从第二静态分区启动操作系统，第二重启之后的初始化环节至少包括：在用户数据分区中创建制式配置文件，将第二制式文件的制式内容写入制式配置文件。

根据第一方面的方法，可以针对采用虚拟 A/B 升级方案的操作系统实现改制；根据第一方面的方法，不需要配置额外的改制工具，设备可以通过下载操作系统升级包自行完成改制操作，从而大大简化改制流程，降低改制难度。

在第一方面的一种实现方式中，获取操作系统升级包之后，方法还包括：

确认操作系统升级包是否用于改写制式，当操作系统升级包用于改写制式时，执行提取第三制式文件的步骤。

在第一方面的一种实现方式中，确认操作系统升级包是否用于改写制式，包括，确认操作系统升级包中是否存在第三制式文件。

在第一方面的一种实现方式中，第二重启之后的初始化环节还包括：

确认用户数据分区中是否存在制式配置文件，当用户数据分区中不存在制式配置文件时，在用户数据分区中创建制式配置文件，将第二制式文件的制式内容写入制式配置文件。

在第一方面的一种实现方式中，第一操作系统以及第二操作系统为对应不同制式的操作系统，第一操作系统对应第一制式，第二操作系统对应第二制式；操作系统升级包还包括动态分区升级数据，动态分区升级数据用于将第一操作系统的动态分区数据更新为第二操作系统的动态分区数据；获取操作系统升级包之前，电子设备启动后加载基础分区、第一静态分区以及动态分区的数据以从第一静态分区启动第一操作系统；

修改电子设备的启动顺序为从第二静态分区启动之前，方法还包括：

在用户数据分区中创建虚拟动态分区，将动态分区升级数据写入到虚拟动态分区；

将第三制式文件的制式内容写入第二制式文件之后，触发电子设备的第一重启之前，方法还包括：

触发电子设备的第三重启，

第三重启之后电子设备从第二静态分区启动操作系统的过程包括：加载基础分区、第二静态分区、动态分区以及虚拟动态分区的数据以启动第二操作系统；

第三重启之后电子设备启动第二操作系统之后，将虚拟动态分区的数据落盘到动态分区。

根据上述第一方面的实现方式的方法，可以在改制的同时实现对动态分区数据的更新，使得在对操作系统改制的同时将操作系统升级到对应的定制操作系统版本，避免进行多次操作系统升级操作，大大简化了操作系统的升级流程，提高了用户体验。

在第一方面的一种实现方式中，操作系统升级包还包括静态分区升级数据，静态分区升级数据用于将第一操作系统的静态分区数据更新为第二操作系统的静态分区数据；

修改电子设备的启动顺序为从第二静态分区启动之前，方法还包括：基于静态分区升级数据更新第二静态分区的数据。

根据上述第一方面的实现方式的方法，可以在改制的同时实现对静态分区数据的更新，使

得在对操作系统改制的同时将操作系统升级到对应的定制操作系统版本，避免进行多次操作系统升级操作，大大简化了操作系统的升级流程，提高了用户体验。

在第一方面的一种实现方式中，方法还包括：

第三重启之后电子设备启动第二操作系统之后，将第二静态分区的数据同步到第一静态分区。

在第一方面的一种实现方式中，将虚拟动态分区的数据落盘到动态分区之后，触发第一重启。

在第一方面的一种实现方式中，在将第三制式文件的制式内容写入第二制式文件的过程中，

第三重启之后电子设备从第二静态分区启动操作系统，其中，在第三重启之后的初始化环节判断是否需要恢复出厂设置，当确认不需要恢复出厂设置时，继续后续的操作系统启动操作。

在第一方面的一种实现方式中，获取操作系统升级包之后，方法还包括，确认操作系统升级包是否用于改写制式，当操作系统升级包用于改写制式时，标记制式改写标志位为第一值；

在第三重启之后的初始化环节判断是否需要恢复出厂设置，包括：当落盘状态为未落盘，并且，制式改写标志位为第一值时，确认不需要恢复电子设备的出厂设置；

将虚拟动态分区的数据落盘到动态分区之后，标记制式改写标志位为第二值。

在第一方面的一种实现方式中，将虚拟动态分区的数据落盘到动态分区之后，触发电子设备的第一重启之前，方法还包括：

触发电子设备的第四重启，第四重启之后电子设备从第二静态分区启动操作系统，其中，在第四重启之后的初始化环节判断是否需要恢复出厂设置，当确认需要恢复出厂设置时，中断操作系统启动操作，触发第一重启。

在第一方面的一种实现方式中，在第四重启之后的初始化环节判断是否需要恢复出厂设置，包括：

确认用户数据分区中保存的制式配置文件与第二制式文件是否匹配；

如果不匹配，确认需要恢复电子设备的出厂设置。

在第一方面的一种实现方式中，获取操作系统升级包之后，方法还包括，确认操作系统升级包是否用于改写制式，当操作系统升级包用于改写制式时，标记制式改写标志位为第一值；

将虚拟动态分区的数据落盘到动态分区之后，标记制式改写标志位为第二值；

在第四重启之后的初始化环节判断是否需要恢复出厂设置，包括：当落盘状态为已落盘，或者，制式改写标志位为第二值时，确认用户数据分区中保存的制式配置文件与第二制式文件是否匹配；

如果匹配，确认不需要恢复电子设备的出厂设置；

如果不匹配，确认需要恢复电子设备的出厂设置。

在第一方面的一种实现方式中，操作系统包括升级包获取工具以及升级引擎，第一操作系统以及第二操作系统为对应不同制式的操作系统，第一操作系统对应第一制式，第二操作系统对应第二制式；获取操作系统升级包之前，电子设备启动后加载基础分区、第一静态分区以及动态分区的数据以从第一静态分区启动第一操作系统；在第一操作系统启动后，方法包括：

升级包获取工具获取操作系统升级包，操作系统升级包包括第三制式文件、静态分区升级数据以及动态分区升级数据；

升级包获取工具触发升级引擎进入升级流程；

升级引擎基于静态分区升级数据更新第二静态分区的数据；

升级引擎在用户数据分区中创建虚拟动态分区，将动态分区升级数据写入到虚拟动态分区，并且，标记落盘状态为未落盘；

升级引擎修改电子设备的启动顺序为从第二静态分区启动；

升级引擎确认操作系统升级包是否用于改写制式；

当操作系统升级包用于改写制式时，升级引擎将第三制式文件的制式内容写入第二制式文件，并且，标记制式改写标志位为第一值；

升级引擎向升级包获取工具返回升级操作完成的状态信息；

升级包获取工具触发电子设备的第三重启，第三重启之后电子设备加载基础分区、第二静态分区、动态分区以及虚拟动态分区的数据以启动第二操作系统；

在第三重启之后启动第二操作系统的初始化环节中：在加载第二制式文件之后，当落盘状态为未落盘，并且，制式改写标志位为第一值时，确认不需要恢复电子设备的出厂设置，继续进行操作系统启动操作；

在第三重启之后启动第二操作系统之后，升级包获取工具触发升级引擎进入落盘流程；

在落盘流程中，升级引擎将虚拟动态分区的数据落盘到动态分区并标记落盘状态为已落盘；将第二静态分区的数据同步到第一静态分区；标记制式改写标志位为第二值；

升级引擎触发电子设备的第四重启，第四重启之后电子设备加载基础分区、第二静态分区、动态分区的数据以启动第二操作系统；

在第四重启之后启动第二操作系统的初始化环节中：在加载第二制式文件后，当落盘状态为已落盘，或者，制式改写标志位为第二值时，确认用户数据分区中保存的制式配置文件与第二制式文件是否匹配；当不匹配时，确认需要恢复电子设备的出厂设置，中断操作系统启动操作，触发第一重启；

第一重启之后电子设备进入恢复模式；在恢复模式下，恢复电子设备的出厂设置；触发第二重启，第二重启之后电子设备从第二静态分区启动操作系统，在第二重启之后的初始化环节中：在用户数据分区中创建制式配置文件，将第二制式文件的制式内容写入制式配置文件。

第二方面，本申请实施例提供了一种电子设备，电子设备包括处理器以及存储器，存储器包括基础分区、第一静态分区、第二静态分区、动态分区以及用户数据分区，电子设备启动操作系统的过程包括初始化环节，基础分区中保存有第一制式文件以及第二制式文件，电子设备从第一静态分区启动操作系统时，初始化环节中加载第一制式文件；电子设备从第二静态分区启动操作系统时，初始化环节中加载第二制式文件；第一制式文件当前的制式内容为第一制式；处理器用于执行存储器上存储的软件代码，以使得电子设备启动后加载基础分区、第一静态分区以及动态分区的数据以从第一静态分区启动操作系统；

并且，在操作系统启动之后，使得电子设备执行如第一方面的方法流程。

第三方面，本申请实施例提供了一种计算机可读存储介质，计算机可读存储介质中存储有计算机程序，当其在计算机上运行时，使得计算机执行如第一方面的方法。

第四方面，本申请实施例提供了一种计算机程序产品，计算机程序产品包括计算机程序，当其在计算机上运行时，使得计算机执行如第一方面的方法。

附图说明

为了更清楚地说明本申请实施例的技术方案，下面将对实施例中所需要使用的附图作简单地介绍，显而易见地，下面描述中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性的前提下，还可以根据这些附图获得其它的附图。

图 1 所示为根据本申请一实施例的操作系统分区架构示意图；

图 2a 所示为根据本申请一实施例的改制系统结构示意图；

图 2b 所示为根据本申请一实施例的改制流程示意图；

图 3 所示为根据本申请一实施例对操作系统进行升级的流程图；

图 4 所示为根据本申请一实施例的数据存储结构示意图；

图 5a 所示为针对图 4 所示实施例的操作系统数据存储结构进行操作系统升级的流程图；

图 5b 所示为一应用场景下设备出厂前进行系统烧录的烧录系统框架结构示意图；

图 6a 所示为根据本申请一实施例的数据存储结构示意图；

图 6b 所示为根据本申请一实施例进行操作系统升级的流程图；

图 7 所示为根据本申请一实施例进行操作系统升级的部分流程图；

图 8 为根据本申请一实施例的手机运行界面示意图；

图 9 为根据本申请一实施例的手机运行界面示意图；

图 10 所示为根据本申请一实施例进行操作系统升级的部分流程图；

图 11a 所示为根据本申请一实施例进行操作系统升级的部分流程图；

图 11b 所示为根据本申请一实施例进行操作系统升级的部分流程图；

图 12 所示为根据本申请一实施例进行操作系统升级的部分流程图；

图 13 所示为根据本申请一实施例进行操作系统升级的流程图；

图 14 所示为根据本申请一实施例进行操作系统升级的流程图。

具体实施方式

为了更好的理解本申请的技术方案，下面结合附图对本申请实施例进行详细描述。

应当明确，所描述的实施例仅仅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作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。

在本申请实施例中使用的术语是仅仅出于描述特定实施例的目的，而非旨在限制本申请。在本申请实施例和所附权利要求书中所使用的单数形式的“一种”、“所述”和“该”也旨在包括多数形式，除非上下文清楚地表示其他含义。

应当理解，本文中使用的术语“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，甲和/或乙，可以表示：单独存在甲，同时存在甲和乙，单独存在乙这三种情况。另外，本文中字符“/”，一般表示前后关联对象是一种“或”的关系。

图 1 所示为根据本申请一实施例的操作系统分区架构示意图。如图 1 所示，Common 分区为基础分区，其保存不参与操作系统升级的系统数据。用户数据分区（Userdata）用于保存用

户的个人数据，例如，用户个人安装的APP、用户个人保存的图片、文档以及视频等个人数据。系统分区用于保存操作系统数据，操作系统升级则是针对系统分区中的数据进行升级操作。系统分区包含多各子分区，例如，bootloader、boot、vendor_boot、dtbo、vbmeta，Super。进一步的，Super 分区包含多个子分区，例如，System、system_ext、vendor、product、Cust、Odm。

一般的，操作系统的制式信息VC（例如，All cn、cmcc cn）存储在Common分区中的一个独立的子分区中，例如，存储在Common分区的oeminfo子分区或者nv子分区。在设备启动操作系统的过程中，设备读取并加载Common分区中的VC信息，并且，VC信息还被写入到保存在用户数据分区（Userdata）的定制信息（Custom.bin 文件）中，以便在操作系统运行过程中被调用。

由于VC被保存在不参与操作系统升级的Common分区中，因此，通常的操作系统升级操作无法改写VC数据。一种可行的改写VC的方式是通过电脑改制工具改写VC。

图2a所示为根据本申请一实施例的改制系统结构示意图。电脑改制工具200接入设备201（例如，手机）并通过网络连接到授权服务器（加密狗）210。在电脑改制工具200接入设备201时，设备201并不运行在正常启动的操作系统状态下，而是处于可以被电脑改制工具200写入数据的模式，例如，手机开机后不启动操作系统，进入调试模式，在调试模式下，手机上的存储器可以被拥有合法授权的电脑改制工具写入数据。

图2b所示为根据本申请一实施例的改制流程示意图。如图2b所示：

S200，电脑改制工具200向授权服务器（加密狗）210请求授权；

S201，请求通过后，电脑改制工具200从授权服务器（加密狗）210处获取授权。

在电脑改制工具200获取授权后，执行S210，电脑改制工具200直接改写设备201存储器的Common分区中保存的VC，例如，将Common分区oeminfo子分区中保存的VC由All cn改写为cmcc cn。

在S210之前，设备201Common分区中保存的VC为All cn；因此，在S210之前设备201存储器的用户数据分区（Userdata）的Custom.bin中，VC为All cn。在S210中，电脑改制工具200仅改写了Common分区中保存的VC，因此，在S210之后，用户数据分区（Userdata）的Custom.bin中，VC仍然为All cn。

S220，设备201重启。具体的，设备201的重启可以由电脑改制工具200触发，例如，电脑改制工具200在改写设备201的VC后向设备201发送重启指令。也可以是用户自行重启设备201。

在220之后，设备201正常启动操作系统。在设备201启动操作系统的过程中，设备执行S230，设备201读取Common分区中保存的VC（cmcc.cn）；检查Common分区中保存的VC与用户数据分区（Userdata）的Custom.bin是否匹配。由于Custom.bin中VC仍然为All cn，因此，Custom.bin中VC与Common分区中保存的VC不匹配；

S240，设备201恢复出厂设置（清除Custom.bin），重启设备。

在实际应用场景中，针对不同的VC，操作系统供应商通常会提供不同的操作系统定制内容。例如，针对中国移动（cmcc），操作系统供应商可以在操作系统中内嵌中国移动专用的数据连接平台。因此，在某些应用场景中，改变VC的同时，为了确保操作系统功能的完整性，需要同步改写操作系统数据（将当前的、对应旧VC的操作系统版本升级为对应新VC的操作

系统版本）。

因此，在 S240 之后，执行 S250，在设备 201 的存储器的系统分区中烧写与 Common 分区中保存的 VC 匹配的操作系统数据，例如，使用与 Common 分区中保存的 VC 匹配的操作系统镜像数据，覆写系统分区中的数据。

进一步的，如果操作系统并未针对不同的 VC 做定制化，则不需要重新烧写操作系统数据。

S260，设备 201 重启，新版本 VC（cmcc cn）生效。

基于图 2 以及图 3 所示的实施例，虽然可以实现 VC 的改写，但是，由于改写操作必须由电脑改制工具进行，因此，改制操作的硬件成本较高；并且，由于在某些网络环境下，无法部署授权服务器（加密狗）节点，因此，改制操作的实现难度较大。

针对 VC 改写，另一种可行的方式是退出操作系统，进入恢复（Recovery）模式，在 Recovery 下对设备的存储器中 common 的 VC 数据进行改写并重写系统分区中的数据。

具体的，图 3 所示为根据本申请一实施例对操作系统进行升级的流程图。假设版本 1.01 的操作系统对应 VC 为第一制式（例如，All cn），版本 1.02 的操作系统对应 VC 为第二制式（例如，cmcc cn）。如图 3 所示，设备执行下述流程以将操作系统由版本 1.01 升级到版本 1.02：

S300，获取操作系统升级包，将操作系统升级包保存到 Userdata 分区，操作系统升级包包括第二制式（cmcc cn）的 VC 数据以及版本 1.02 的操作系统的镜像数据；

S310，重启设备进入恢复（Recovery）模式；

S320，在恢复（Recovery）模式下，读取 Userdata 分区的操作系统升级包；

S321，提取操作系统升级包中版本 1.02 的操作系统的镜像数据，将镜像数据恢复到系统分区；S321 的目的是将设备的操作系统版本升级到匹配第二制式（cmcc cn）的操作系统版本，如操作系统并未针对 All cn、cmcc cn 做定制化，则操作系统升级包不需要包含操作系统的镜像数据；

S322，提取操作系统升级包中的第二制式（cmcc cn）的 VC 数据，使用第二制式（cmcc cn）的 VC 数据改写设备存储器 Common 分区中保存的 VC（例如，oeminfo 子分区中保存的 All cn）；

S323，恢复设备的出厂设置（清除 Custom.bin）；

S330，重启设备，启动操作系统，新版本 VC（cmcc cn）生效。

虽然基于图 3 所示流程可以实现对 VC 的改写，但是，随着数据安全性要求的不断提高，在某些操作系统中，禁止在恢复（Recovery）模式下对用户个人数据进行访问，这就使得在恢复（Recovery）模式设备无法在分区中重新写入操作系统数据。

以采用虚拟 A/B 升级方式的安卓系统为例，图 4 所示为根据本申请一实施例的数据存储结构示意图。如图 4 所示，安卓系统数据存储区包含基础分区（Common）、静态分区（A）、静态分区（B）、动态分区（Super）、用户数据分区（Userdata）。

用户数据分区（Userdata）用于保存用户的个人数据，例如，用户个人安装的 APP、用户个人保存的图片、文档以及视频等个人数据。基础部分中保存的数据为不参与操作系统升级的系统数据。静态分区（A）与静态分区（B）的结构相互对应，子分区命名通过后缀_a 以及_b

相互区分。例如，静态分区（A）包括 bootloader_a、boot_a、vendor_boot_a、dtbo_a、vbmeta_a；静态分区（B）包括 bootloader_b、boot_b、vendor_boot_b、dtbo_b、vbmeta_b。动态分区（Super）包含多个子分区（System、system_ext、vendor、product、Cust、Odm）。

在 Common 分区 oeminfo 子分区中保存制式文件 oem-vc，oem-vc 的内容为设备的制式，例如，all cn。

在设备启动时，从一个静态分区启动。例如，设备从静态分区（A）启动：依次加载基础分区（Common）、静态分区（A）以及动态分区（Super），在加载静态分区（A）的过程中，执行初始化环节，在初始化环节加载设备制式（加载 oem-vc）；设备从静态分区（B）启动：依次加载基础分区（Common）、静态分区（B）以及动态分区（Super），在加载静态分区（B）的过程中，执行初始化环节，在初始化环节加载设备制式（加载 oem-vc）。

以采用主引导记录（Master Boot Record，MBR）格式的通用闪存（Universal Flash Storage，UFS）为例。在 UFS 的 MBR（主引导扇区，UFS 的第一个扇区，即 C/H/S 地址的 0 柱面 0 磁头 1 扇区）中，保存有设备启动顺序描述，例如，从静态分区（A）启动（启动顺序标志为 A）或从静态分区（B）启动（启动顺序标志为 A）。设备启动后首先从 UFS 的 MBR 中读取设备启动顺序。

图 5a 所示为针对图 4 所示实施例的操作系统数据存储结构进行操作系统升级的流程图，当设备当前是从静态分区（A）启动时，设备按照如图 5a 所示的流程实现操作系统的升级。

S500，设备依次加载基础分区（Common）、静态分区（A）以及动态分区（Super），从静态分区（A）启动。

S510，设备获取操作系统升级包。

示例的，在一种可行的实现方案中，设备定期向搜包服务器发起搜包请求，搜包请求包含设备当前运行的操作系统的版本号（例如版本 1.1）；搜包服务器根据搜包请求中的操作系统版本号，检索当前是否存在更新版本号的操作系统安装包（例如版本 1.2）；当存在更新版本的操作系统安装包时，搜包服务器向设备反馈操作系统升级包（例如，由版本 1.1 升级到版本 1.2 的系统增量升级安装包）的下载地址；设备根据操作系统升级包的下载地址下载操作系统升级包。

S520，设备根据操作系统升级包针对静态分区（B）进行数据写入操作以升级静态分区。

在 S520 的执行过程中，存在 S520 执行失败（静态分区升级失败）的情况。针对该情况，设备会中断整个操作系统升级操作，向用户输出升级失败提示（例如，显示升级失败的对话框），自动重新升级或者由用户确定是否重新升级或放弃升级。

为检测 S520 中是否存在静态分区升级失败的情况，在 S520 中，会对数据写入后的静态分区（B）进行数据校验以确认静态分区数据是否成功写入。

例如，在一应用场景中，版本 1.1 升级到版本 1.2 的系统升级安装包包含版本 1.2 的静态分区的全量数据以及版本 1.2 的静态分区的全量数据的哈希值。设备将版本 1.2 的静态分区的全量数据覆写到静态分区（B）中。数据写入之后，设备计算静态分区（B）中数据的哈希值，校验静态分区（B）中数据的哈希值与版本 1.1 升级到版本 1.2 的系统升级安装包中版本 1.2 的静态分区的全量数据的哈希值是否一致。如果一致，则说明数据写入成功，可以进行后续的操作系统升级操作；如果不一致，则说明数据写入失败，升级失败。

又例如，在一应用场景中，版本 1.1 升级到版本 1.2 的系统升级安装包包含版本 1.1 升级到版本 1.2 的静态分区的差分数据、版本 1.1 的静态分区的全量数据的哈希值以及版本 1.2 的静态分区的全量数据的哈希值。

设备在向静态分区（B）写入数据之前，首先计算静态分区（A）中数据的哈希值，校验静态分区（A）中数据的哈希值与版本 1.1 升级到版本 1.2 的系统升级安装包中版本 1.1 的静态分区的全量数据的哈希值是否一致，如果一致，则说明当前静态分区（A）中数据为版本 1.1 的静态分区数据，版本 1.1 升级到版本 1.2 的静态分区的差分数据可用；如果不一致，则版本 1.1 升级到版本 1.2 的静态分区的差分数据不可用，升级失败。

在设备确定版本 1.1 升级到版本 1.2 的静态分区的差分数据可用后，读取静态分区（A）中数据，使用版本 1.1 升级到版本 1.2 的静态分区的差分数据以及静态分区（A）中数据执行还原得到版本 1.2 的静态分区的全量数据，将版本 1.2 的静态分区的全量数据覆写到静态分区（B）中。数据写入之后，设备计算静态分区（B）中数据的哈希值，校验静态分区（B）中数据的哈希值与版本 1.1 升级到版本 1.2 的系统升级安装包中版本 1.2 的静态分区的全量数据的哈希值是否一致。如果一致，则说明数据写入成功，可以进行后续的操作系统升级操作；如果不一致，则说明数据写入失败，升级失败。

以一个静态分区的子分区 boot 为例，在一应用场景中，版本 1.1 升级到版本 1.2 的系统升级安装包包含下述数据：

Name:boot（分区名称，表示当前数据为指向静态分区的子分区 boot 的升级数据）

Start:12222（数据块起始地址，表示静态分区的子分区 boot 的升级数据（差分数据 DELTA1）的起始位置为 12222）

size: 2410（数据大小，表示静态分区的子分区 boot 的升级数据（差分数据 DELTA1）的大小为 2410）

原 hash 值: HASH11（版本 1.1 的静态分区的子分区 boot 的数据的哈希值）

镜像目标 hash 值: HASH12（版本 1.2 的静态分区的子分区 boot 的数据的哈希值）

差分数据 delta: DELTA1（版本 1.1 升级到版本 1.2 的静态分区的差分数据）

在 S520 中，设备通过 common 分区中的 misc 分区读取设备的固定挂载路径，如/dev/block/by-name/misc。从 UFS 器件中读取卡槽位（slot-b），替换得到个子分区路径，如/dev/block/by-name/boot_b。

继续以子分区 boot 为例，设备首先计算/dev/block/by-name/boot_a 下数据的哈希值，校验/dev/block/by-name/boot_a 下数据的哈希值与哈希值 HASH11 是否一致，如果一致，则 DELTA1 可用，如果不一致，则升级操作失败。

当/dev/block/by-name/boot_a 下数据的哈希值与哈希值 HASH11 一致时，设备基于 Start:12222 以及 size: 2410 读取 DELTA1，使用 DELTA1 与/dev/block/by-name/boot_a 下数据执行还原得到版本 1.2 的静态分区的子分区 boot 的全量数据。设备将版本 1.2 的静态分区的子分区 boot 的全量数据写到/dev/block/by-name/boot_b 下。

数据写入后，设备计算/dev/block/by-name/boot_b 下数据的哈希值，校验/dev/block/by-name/boot_b 下数据的哈希值与哈希值 HASH12 是否一致，如果一致，则静态分区的子分区 boot 升级成功，可以针对下一个静态分区子分区进行升级；如果不一致，则升级操作失败。

在一应用场景中，设备根据系统升级安装包所包含的静态分区子分区升级数据对静态分区的子分区进行升级，具体的，如果系统升级安装包包含某个静态分区子分区升级数据，则按照上述 boot 子分区的升级流程对该静态分区（B）中的该子分区进行升级，如果系统升级安装包不包含某个静态分区子分区升级数据，则将静态分区（A）中该子分区的数据直接同步到静态分区（B）中该子分区中。在升级过程中，当一个子分区出现升级错误，哈希校验失败，则中断升级操作，升级失败；当所有子分区升级成功，则静态分区升级成功，可以执行后续步骤。

进一步的，当某个静态分区（静态分区（A）或静态分区（B））升级失败时，静态分区的数据无法用于顺利启动操作系统，为了避免在操作系统启动过程中加载升级失败的静态分区而导致操作系统启动错误，在一应用场景中，静态分区具备对应的状态标记（可启动或不可启动）。设备在加载静态分区数据之前首先读取静态分区状态标记，仅当静态分区的状态标记为可启动时才加载静态分区的数据。在升级静态分区的数据之前，会将静态分区标记为不可启动，在静态分区的数据升级成功后再将静态分区标记为可启动，这样，如果静态分区升级失败，则静态分区的状态会保持在不可启动。设备就不会加载升级失败的静态分区的数据。

例如，在 S520 中，在升级静态分区（B）的数据之前，标记静态分区（B）为不可启动。具体的，静态分区的状态标记保存在 Common 分区。在 S520 中，在升级静态分区（B）的数据之前，将 Common 分区中静态分区的状态标记中 slot-b 标记为不可启动（unbootable）。当 S520 成功执行（所有的哈希校验均成功时）后，标记静态分区（B）为可启动。例如，在 S520 之后，将 Common 分区中静态分区的状态标记中 slot-b 标记为可启动（bootable）。

S530，设备根据操作系统升级包在用户数据分区（Userdata）创建虚拟动态分区，在虚拟动态分区写入动态分区（Super）的升级数据。例如，在操作系统升级包中包含版本 1.2 的动态分区的数据，设备在虚拟动态分区中写入版本 1.2 的动态分区（Super）的数据。

进一步的，在虚拟 A/B 升级方案中，针对动态分区（Super），采用增量升级方式。在升级过程中，用户数据分区（Userdata）的虚拟动态分区中保存的并不是升级后新版本的动态分区（Super）的全部文件，而是旧版本的动态分区（Super）中需要升级的数据在升级后的升级结果。即，用户数据分区（Userdata）的虚拟动态分区中保存的是动态分区的更新数据。

以 system 子分区为例，假设在版本 1.1 中，system 子分区中的数据可以分为 system1、system2 两部分。从版本 1.1 升级到版本 1.2，数据 system2 没有发生变化，数据 syetem1 被升级为 system3。那么，在 S530 中，设备在用户数据分区（Userdata）创建虚拟动态分区，在虚拟动态分区中写入数据 system3。

例如，版本 1.1 升级到版本 1.2 的系统增量升级安装包包含版本 1.1 升级到版本 1.2 的动态分区（Super）更新数据，该动态分区（Super）更新数据包含数据 system3。

进一步的，在虚拟 A/B 升级方案中，基于快照技术（snapshot）实现动态分区（Super）的增量升级。具体的，用户数据分区（Userdata）的虚拟动态分区中，采用写时拷贝（Copy-On-Write，COW）文件保存动态分区（Super）的升级数据。

具体的，用户数据分区（Userdata）中保存的动态分区（Super）的升级数据包含多个 COW 文件，每个 COW 文件对应一个动态分区（Super）的子分区，COW 文件的命名与其所针对的动态分区（Super）子分区相对应。

在 S510 所获取的操作系统升级包中，动态分区（Super）的升级数据的 COW 文件以二进

制代码形式压缩保存。在操作系统升级包中，每个 COW 文件根据其所针对的动态分区（Super）子分区所命名。例如，针对 system 子分区的 COW 文件被命名为 system-cow-img.img.0000。

在 S530 中，设备解包操作系统升级包以获取所有的 COW 文件，为每个 COW 文件附加 A/B 分区标记。具体的，当设备当前从静态分区（A）启动时，可以理解为设备当前运行操作系统所加载的动态分区（Super）为动态分区（A）。在升级操作系统时，用户数据分区（Userdata）中创建的虚拟动态分区是针对动态分区（B）。因此，为 COW 文件附加对应动态分区（B）的名称标记_b。例如，为 system-cow-img.img.0000 附加_b 生成 system_b-cow-img.img.0000。

进一步的，在 S530 中，在用户数据分区（Userdata）中创建 Update 文件夹，将重命名的 COW 文件保存到 Update 文件夹下。例如，在一应用场景中，在向用户数据分区（Userdata）写入 COW 文件后，用户数据分区（Userdata）的 Update 文件夹中包含下述文件：

system_b-cow-img.img.0000；

system_ext_b-cow-img.img.0000；

vendor_b-cow-img.img.0000；

product_b-cow-img.img.0000；

cust_b-cow-img.img.0000；

odm_b-cow-img.img.0000。

具体的，COW 文件中包含 COW 文件自身的 COW 文件地图（快照 map）以及升级数据。COW 文件地图（快照）与 COW 文件所针对的动态分区（Super）的子分区的文件地图相对应。动态分区（Super）的子分区的文件地图用于描述当前版本的操作系统（本次升级之前的版本，例如，版本 1.1）动态分区（Super）的子分区中的所有文件以及各个文件的保存地址。

COW 文件中的升级数据为相较于当前版本的子分区数据，新版本的子分区数据中被更新的文件；COW 文件自身的 COW 文件地图则用于描述被更新的文件与当前版本的子分区中的文件间的对应关系以及被更新的文件的保存地址。

基于动态分区（Super）的子分区的文件地图以及 COW 文件中的 COW 文件地图，就可以使用 COW 文件中的升级数据替换动态分区（Super）的子分区中的对应文件，从而实现动态分区（Super）数据的升级。具体的，在需要获取动态分区（Super）的子分区的文件地图时，可以基于 snapshot 对动态分区（Super）的子分区的数据进行快照操作以生成动态分区（Super）的子分区的文件地图。也可以在制作操作系统升级包时，预先生成动态分区（Super）的子分区的文件地图，将该文件地图加入到 COW 文件中。

以 system 子分区为例，假设 system 子分区中按照以下路径保存数据：

/system/app/A0.XXX；

/system/app/A1.XXX；

/system/app/A2.XXX；

/system/B0.XXX；

/system/B1.XXX；

/system/user/C0.XXX；

/system/user/C1.XXX；

/system/user/C2.XXX；

/system/user/C3.XXX。

system 子分区的文件地图可以是：

/system/app/A0.XXX：024010~024013；

/system/app/A1.XXX：024014~024017；

/system/app/A2.XXX：024018~024020；

/system/B0.XXX：024021~024026；

/system/B1.XXX：024027~024028；

/system/user/C0.XXX：024029~024032；

/system/user/C1.XXX：024033~024035；

/system/user/C2.XXX：024036~024040；

/system/user/C3.XXX：024041~024044。

文件名后的数值（例如，/system/app/A0.XXX：024010~024013 中的 024010~024013）为该文件在动态分区（Super）的 system 子分区的物理保存地址（块地址）。

假设当前操作系统升级需要更新数据/system/app/A2.XXX 以及/system/user/C2.XXX。

可以视为：

/system/app/A2.XXX 以及/system/user/C2.XXX 为 system 子分区数据的 system1 部分；

/system/app/A0.XXX 、 /system/app/A1.XXX 、 /system/B0.XXX 、 /system/B1.XXX 、 /system/user/C0.XXX、/system/user/C1.XXX 以及/system/user/C3.XXX 为 system 子分区数据的 system2 部分。

那么，针对 system 子分区的 COW 文件（system_b-cow-img.img.0000）就包含最新版的 /system/app/A2.XXX 以及/system/user/C2.XXX。

可以视为，最新版的/system/app/A2.XXX 以及/system/user/C2.XXX 为 system3。升级目标是使用 system3 更新掉 system1。

COW 文件（system_b-cow-img.img.0000）自身的 COW 文件地图可以为：

/system/app/A2.XXX：

Map1（原 super 分区中待更新数据的地址）：起始地址 address start：024018（相对于 system 起始地址的偏移量）；偏移量大小 size：2（即 024018~024020 地址段的数据）

Map2（cow 文件中存储的更新数据的地址）：起始地址 address start：045033（相对于 cow 文件存储的起始地址的偏移量）；偏移量大小 size：2（即 045033~045035 地址段的数据）；

/system/user/C2.XXX：

Map1（原 super 分区中待更新数据的地址）：起始地址 address start：024036（相对于 system 起始地址的偏移量）；偏移量大小 size：4（即 024036~024040 地址段的数据）

Map2（cow 文件中存储的更新数据的地址）：起始地址 address start：045036（相对于 cow 文件存储的起始地址的偏移量）；偏移量大小 size：4（即 045036~045040 地址段的数据）；

文件名后的数值（045033~045035 以及 045036~045040）分别为 COW 文件（system_b-cow-img.img.0000）中最新版的/system/app/A2.XXX 以及/system/user/C2.XXX 在用户数据分区（Userdata）的物理保存地址（块地址）。

这样，如果使用地址 045033~045035 上的 A2.XXX 替换掉地址 024018~024020 上的

A2.XXX，并且，使用地址 045036~045040 上的 C2.XXX 替换掉地址 024036~024040 上的 C2.XXX，就可以完成动态分区（Super）的 system 子分区的数据升级。

在 COW 文件成功写入到用户数据分区（Userdata）后，将基础分区（Common）的元数据分区（/metadata）中的落盘状态信息由“已落盘（merged）”改为“未落盘（wait for merge）”。落盘状态信息用于表示当前是否存在需要落盘到动态分区（Super）的 COW 文件。具体的，落盘状态信息包含针对动态分区（Super）的整体标识以及针对每个子分区的子分区标识。当整体标识为“已落盘（merged）”时，代表动态分区（Super）的所有子分区均不需要进行落盘操作；当整体标识为“未落盘（wait for merge）”时，代表动态分区（Super）的一个或多个子分区需要进行落盘操作；当子分区标识为“已落盘（merged）”时，代表该子分区不需要进行落盘操作；当子分区标识为“未落盘（wait for merge）”时，代表该子分区需要进行落盘操作。

进一步的，在某些应用场景中，在 S530 中，设备不仅仅向用户数据分区（Userdata）写入 COW 文件，还会刷新动态分区（Super）的 metadata 中的分区信息。

具体的，图 5b 所示为一应用场景下设备出厂前进行系统烧录的烧录系统框架结构示意图。在采用虚拟 A/B 升级方式的安卓系统中，由于只有静态分区采用 A/B 方案，而动态分区采用升级时构造虚拟动态分区的方案。因此，为了静态分区与动态分区的匹配，如图 5b 所示，在动态分区（Super）的头部的元数据（/supermetadata）中，包含对应静态分区（A）的 Slot0（插槽一数据）以及静态分区（B）的 Slot1（插槽二数据）。Slot0 以及 Slot1 用于保存 Super 分区的分区表。

例如，在 UFS 的 MBR 中，设备启动顺序描述中，配置 Slot0 对应从静态分区（A）启动，配置 Slot1 对应从静态分区（B）启动。在设备启动时，根据启动的静态分区的不同，选择从 Slot0 或 Slot1 中的一个中获取 Super 分区的分区信息。例如，在设备由静态分区 A 启动时，在加载 Super 分区时，设备首先读取 Slot0，以获取 Super 分区的子分区地址；在设备由静态分区 B 启动时，在加载 Super 分区时，设备首先读取 Slot1，以获取 Super 分区的子分区地址。

具体的，Slot0 以及 Slot1 中包含多个子分区描述组，每个子分区描述组对应 Super 分区的一个子分区。每个子分区描述组包含：

名称（Name）项，其值为子分区的名称；

组（Group）项，其值为子分区类型；

属性（Attributes）项，其值为分区读写属性，例如，只读属性（readonly）；

地址（Extents）项，其值为子分区的地址（例如，分区大小、偏移量）。

在 Name 项以及 Group 项中，值的后缀为_a ，则对应静态分区（A）；值的后缀为_b ，则对应静态分区（B）。

在由静态分区 A 启动，加载 Super 分区时，首先读取 Slot0。在读取 Slot0 时，由于后缀为_a 对应静态分区（A），设备读取 Slot0 中 Name 项和/或 Group 项后缀为_a 的分区描述组中 Extents 项的值，以获取 Super 分区的子分区地址。

在由静态分区 B 启动，加载 Super 分区时，首先读取 Slot1。在读取 Slot1 时，由于后缀为_b 对应静态分区（B），设备读取 Slot0 中 Name 项和/或 Group 项后缀为_b 的分区描述组中 Extents 项的值，以获取 Super 分区的子分区地址。

在S510所获取的操作系统升级包中，包含1.2版本的动态分区（Super）的分区信息，在S530中，设备从操作系统升级包中提取1.2版本的动态分区（Super）的分区信息，使用1.2版本的动态分区（Super）的分区信息刷新静态分区（B）所对应的Slot1中的分区信息。

以System子分区为例，假设在S530之前，动态分区（Super）的/supermetadata 的Slot 1中包含如下内容：

Metadata version: 10.2
Metadata size: 1300 bytes
Metadata max size: 65536 bytes
Metadata slot count: 3
Header flags: virtual_ab_device
Partition table: ----------
Name: system_b
Group: ry_dynamic_partitions_b
Attributes: readonly,updated
Extents: 0 .. 6995967 linear super 2048

在S510所获取的操作系统升级包中，1.2版本的动态分区（Super）的分区信息包含如下内容：

Name: system
Group: ry_dynamic_partitions
Extents: 0 .. 699XXXX linear super 2048

在S530中，设备通过当前需要升级的静态分区为静态分区（B）定位到对应静态分区（B）的动态分区（Super）的/supermetadata 的Slot 1，使用1.2版本的动态分区（Super）的分区信息刷新Slot 1中的内容。在S530之后，动态分区（Super）的/supermetadata 的Slot 1中包含如下内容：

Metadata version: 10.2
Metadata size: 1300 bytes
Metadata max size: 65536 bytes
Metadata slot count: 3
Header flags: virtual_ab_device
Partition table: ----------
Name: system_b
Group: ry_dynamic_partitions_b
Attributes: readonly,updated
Extents: 0 .. 699XXXX linear super 2048

进一步的，在S530的执行过程中，存在S530执行失败的情况。针对该情况，设备会中断整个操作系统升级操作，向用户输出升级失败提示（例如，显示升级失败的对话框），自动重新升级或者由用户确定是否重新升级或放弃升级。（参照S520中静态分区数据写入失败）

具体的，当用户数据分区（Userdata）的存储空间不足时会导致 S530 执行失败。在 S530 中，在设备根据操作系统升级包在用户数据分区（Userdata）创建虚拟动态分区的过程中，虚拟动态分区的大小是由操作系统升级包中版本 1.2 的动态分区的数据的大小决定的。当用户数据分区（Userdata）上的空余空间不足以创建虚拟动态分区时，S530 执行失败。

例如，在一应用场景中，设备从操作系统升级包中提取 COW 文件并将 COW 文件写入用户数据分区（Userdata）的 Update 文件夹中。操作系统升级包中包含 COW 文件内容以及 COW 文件大小。在 S530 中，设备首先根据操作系统升级包中的 cow 文件名称以及 COW 文件大小在用户数据分区（Userdata）的 Update 文件夹下创建空 COW 文件，然后从操作系统升级包中提取 COW 文件数据写入空 COW 文件。

以 system 子分区为例，在操作系统升级包中，针对 system 子分区的 COW 文件被命名为 system-cow-img.img.0000，system-cow-img.img.0000 的大小为 XXXX。设备在用户数据分区（Userdata）的 Update 文件夹下创建 system_b-cow 文件，system_b-cow 文件的大小为 XXXX，内容为空。在 system_b-cow 文件创建完成后，设备就可以从系统升级安装包中提取 system-cow-img.img.0000，写入 system_b-cow 并改名为 system_b-cow-img.img.0000。

设备在用户数据分区（Userdata）的 Update 文件夹下创建空 COW 文件 system_b-cow、system_ext_b-cow、vendor_b-cow、product_b-cow、cust_b-cow、odm_b-cow。当所有的空 COW 文件创建完成后，设备就可以从系统升级安装包中提取 COW 文件数据，写入空 COW 文件并改名。

最终用户数据分区（Userdata）的 Update 文件夹中包含下述文件：

system_b-cow-img.img.0000；

system_ext_b-cow-img.img.0000；

vendor_b-cow-img.img.0000；

product_b-cow-img.img.0000；

cust_b-cow-img.img.0000；

odm_b-cow-img.img.0000。

在创建空 COW 文件的过程中，设备每次创建一个空 COW 文件，在一个空 COW 文件创建成功后创建下一个。在此过程中，当一个空 COW 文件创建失败，则说明用户数据分区（Userdata）的存储空间不足，S530 执行失败，操作系统升级失败。

进一步的，在 S530 中，COW 文件的提取失败也会导致 S530 执行失败。具体的，在操作系统升级包中，以二进制代码形式保存 COW 文件，在将 COW 文件写入用户数据分区（Userdata）时，首先需要从操作系统升级包中提取 COW 文件，将 COW 文件打开，将 COW 文件数据写入到用户数据分区（Userdata）。在上述过程中，如果操作系统升级包存在数据错误，COW 文件无法提取或打开，则 S530 执行失败，操作系统升级失败。

进一步的，在 S530 中，COW 文件的写入失败也会导致 S530 执行失败。为检测 COW 文件写入是否成功，在 S530 中，在将 COW 文件写入用户数据分区（Userdata）后，还需要对动

态分区（Super）+COW 文件进行整体校验，校验动态分区（Super）+COW 文件的有效性，验证当前版本的动态分区(Super)数据+COW 文件的合成结果是否为新版本的动态分区(Super)数据。

具体的，以从 1.1 版本升级到 1.3 版本为例，计算动态分区（Super）中不需要升级的数据（从版本 1.1 到版本 1.2 未发生变化的数据）与 COW 文件中升级数据（从版本 1.1 到版本 1.2 需要升级的数据）的合成结果的哈希值，判断该哈希值与 1.3 版本中动态分区（Super）的完整数据的哈希值是否一致，如果一致，则说明 COW 文件有效；如果不一致，则说明 COW 文件无效，升级失败，中断升级进程并报错；其中，1.3 版本中动态分区（Super）的完整数据的哈希值被保存在操作系统升级包中。

具体的，在校验过程中，基于 snapshot 合并动态分区（Super）+COW 文件。在 snapshot 的实现过程中，动态分区（Super）与 COW 文件的合并并不是物理意义上的合并，而是将 system 子分区的整体文件地图与 COW 文件自身的 COW 文件地图进行合并，生成新版本的子分区数据的文件地图。

例如，将 system 子分区的文件地图：

/system/app/A0.XXX：024010~024013；

/system/app/A1.XXX：024014~024017；

/system/app/A2.XXX：024018~024020；

/system/B0.XXX：024021~024026；

/system/B1.XXX：024027~024028；

/system/user/C0.XXX：024029~024032；

/system/user/C1.XXX：024033~024035；

/system/user/C2.XXX：024036~024040；

/system/user/C3.XXX：024041~024044。

与 COW 文件地图：

/system/app/A2.XXX：045033~045035；

/system/user/C2.XXX：045036~045040。

合并。则得到 system 子分区的新版本的文件地图：

/system/app/A0.XXX：024010~024013；

（指向动态分区（Super）中/system/app 下的 A0.XXX）

/system/app/A1.XXX：024014~024017；

（指向动态分区（Super）中/system/app 下的 A1.XXX）

/system/app/A2.XXX：045033~045035；

（指向用户数据分区（Userdata）中/Update/system_b-cow-img.img.0000 中的 A2.XXX）

/system/B0.XXX：024021~024026；

（指向动态分区（Super）中/system 下的 B0.XXX）

/system/B1.XXX：024027~024028；

（指向动态分区（Super）中/system 下的 B1.XXX）

/system/user/C0.XXX：024029~024032；

（指向动态分区（Super）中/system/user 下的 C0.XXX）

/system/user/C1.XXX：024033~024035；

（指向动态分区（Super）中/system/user 下的 C1.XXX）

/system/user/C2.XXX：045036~045040；

（指向用户数据分区（Userdata）中/Update/system_b-cow-img.img.0000 中的 C2.XXX）

/system/user/C3.XXX：024041~024044。

（指向动态分区（Super）中/system/user 下的 C3.XXX）

在新版本的 system 子分区的文件地图中，/system/app/A2.XXX 的保存地址并不是指向存储器上动态分区(Super)上的/system/app/A2.XXX,而是指向存储器上用户数据分区(Userdata)中 system_b-cow-img.img.0000 中的 A2.XXX；/system/user/C2.XXX 的保存地址并不是指向存储器上动态分区(Super)上的/system/user/C2.XXX,而是指向存储器上用户数据分区(Userdata)中 system_b-cow-img.img.0000 中的 C2.XXX。

在校验过程中，按照上述合成方式，获取动态分区（Super）的所有子分区的新版本的文件地图（如果用户数据分区（Userdata）中并未写入某个子分区的对应 COW 文件，则直接以该子分区的文件地图为新版本的文件地图）。将所有子分区的新版本的文件地图组合生成动态分区（Super）的新版本的文件系统。

基于动态分区（Super）的新版本的文件系统读取数据，读取动态分区（Super）的新版本的文件系统所包含的所有文件并计算哈希值。

S531，将设备的启动顺序由可从静态分区（A）启动变更为可从静态分区（B）启动。

例如，改写主引导记录（Master Boot Record，MBR）的启动顺序标识，将启动顺序标识由 A 改写为 B。在设备上电后，当设备读取到启动顺序标识为 A，设备从静态分区（A）启动，启动过程中加载静态分区（A）；当设备读取到启动顺序标识为 B，设备从静态分区（B）启动，启动过程中加载静态分区（B）。

进一步的,在 S531 中,还将 Common 分区中静态分区的状态标记中 slot-b 标记为 bootable。

S532，设备重启。退出当前的操作系统，切断设备电源，再次开启设备电源。

S540，设备依次加载基础分区（Common）、静态分区（B）。具体的，在加载静态分区（B）之前，设备首先根据静态分区（B）的状态标记来确认静态分区（B）是否可以用于启动操作系统。例如，当静态分区（B）的状态标记为 bootable，则可以用于启动操作系统；当静态分区（B）的状态标记为 unbootable，则不可以用于启动操作系统；

S541，设备加载动态分区（Super）以及用户数据分区（Userdata）的虚拟动态分区。

具体的，设备读取元数据（/metadata）中的落盘状态信息，基于落盘状态信息确定是否需要从用户数据分区（Userdata）的指定路径中检索 COW 文件，并采用 snapshot 合并加载动态分区（Super）以及 COW 文件。

进一步的，在 S541 中，设备并不加载动态分区（Super）以及用户数据分区（Userdata）中的全部 COW 文件，而是根据操作系统启动需求加载对应的文件。具体的，在 S541 中，设备根据操作系统启动需求确定需要加载的文件，基于 snapshot 从动态分区（Super）或虚拟动态分区中的 COW 文件中提取对应的文件进行加载。

具体的，在S541中，当动态分区（Super）的子分区首存在对应的COW文件时，先基于snapshot生成动态分区（Super）各个子分区的新版本的文件地图。生成新版本的文件地图的过程可以参照S530。设备根据操作系统启动需求确定需要加载的文件，基于动态分区（Super）子分区的新版本的文件地图进行文件加载。

例如，操作系统启动需求加载system子分区下目录user（/system/user）中的所有数据。设备读取元数据（/metadata）中的落盘状态信息，落盘状态信息中system子分区的子分区标识为“未落盘（wait for merge）”，因此，设备在用户数据分区（Userdata）中/Update下搜索COW文件，在Update下搜索到COW文件system_b-cow-img.img.0000后，基于snapshot，根据system_b-cow-img.img.0000中的COW文件的文件地图生成system子分区的新版本的文件地图。按照system子分区的新版本的文件地图中/system/user下所有文件的保存地址进行数据加载，例如，根据system子分区的新版本的文件地图中：

/system/user/C0.XXX：024029~024032；

/system/user/C1.XXX：024033~024035；

/system/user/C2.XXX：045036~045040；

/system/user/C3.XXX：024041~024044。

加载地址024029~024032处的C0.XXX、地址024033~024035处的C1.XXX、地址045036~045040处的C2.XXX以及地址024041~024044处的C3.XXX。

进一步的，在加载system子分区下目录user（/system/user）中的所有数据时，当落盘状态信息中system子分区的子分区标识为“已落盘（merged）”时，设备就不会在用户数据分区（Userdata）中/Update下搜索COW文件，而是直接加载system子分区下目录user（/system/user）中的所有数据。

进一步的，在加载system子分区下目录user（/system/user）中的所有数据时，当落盘状态信息中system子分区的子分区标识为“未落盘（wait for merge）”时，如果设备在用户数据分区（Userdata）中/Update下未搜索到对应system子分区的COW文件时，则说明升级过程中数据写入错误（COW文件写入错误或者落盘状态信息写入错误），此时设备回滚系统并报错。

进一步的，在S541中，在加载文件之前，设备还需要对加载文件进行校验。不同于S530，在S541中，不对动态分区（Super）+COW文件进行整体验证，而是仅对需要加载的文件进行验证。例如，基于dmverity进行校验（dm-verity是dm（device mapper）的一个目标（target），是一个虚拟块设备，专门用于文件系统的校验）。校验成功则加载文件，校验失败则重启设备，回滚系统或者尝试再次加载文件。

S550，设备成功启动，进入用户交互界面。

S551，设备将虚拟动态分区的数据落盘到动态分区（Super）。

在本申请说明书的描述中，落盘操作指的是，在操作系统升级过程中，将用户数据分区（Userdata）上虚拟动态分区中保存的动态分区（Super）升级文件（COW文件）写入到动态分区（Super）中，使得动态分区（Super）的文件完成数据升级，以便设备在下次启动时不需要加载动态分区（Super）和虚拟动态分区，只需加载动态分区（Super）就可以完成设备启动。

具体的，设备在启动成功后进行开机广播，开机广播后开启升级进程。升级进程读取基础分区（Common）的元数据（/metadata）中的落盘状态信息，如果落盘状态信息为“已落盘

（merged）”，则设备进入正常启动模式。

如果落盘状态信息为“未落盘（wait for merge）”，升级进程将用户数据分区（Userdata）中的 COW 文件落盘到动态分区（Super）中。

具体的，升级进程将用户数据分区（Userdata）中的 COW 文件中的升级数据写入到动态分区（Super）中的对应地址上，使得动态分区（Super）中的全部数据均为升级后的新版本的数据。

例如，基于 system 子分区的文件地图中的/system/app/A2.XXX：024018~024020 以及 COW 文件地图中的/system/app/A2.XXX：045033~045035，将地址 045033~045035 上的数据写入到地址 024014~024017 上；基于 system 子分区的文件地图中的/system/user/C2.XXX：024036~024040 以及 COW 文件地图中的/system/user/C2.XXX：045036~045040，将地址 045036~045040 上的数据写入到地址 024036~024040 上。

在此之后升级进程删除用户数据分区（Userdata）中的 COW 文件，将存储空间归还给用户数据分区（Userdata）；并且，将基础分区（Common）的元数据（/metadata）中的落盘状态信息由“未落盘（wait for merge）”改为“已落盘（merged）”。

在 S520 中，静态分区升级的数据操作是针对静态分区（B）中的操作系统数据的，其并不会影响到当前启动的静态分区（A）的操作系统数据；并且，在 S530 中，动态分区升级的数据操作是在用户数据分区（Userdata）中所创建的虚拟动态分区上完成的，其并不会影响到当前挂载的动态分区（Super）。因此，在整个操作系统升级的过程中，用户可以正常使用设备；并且，在 S531 完成后，设备并不需要立即重启，可以由用户自行选择重启时机；这样，操作系统的升级过程并不会对用户的正常手机操作产生影响，从而大大提高了用户体验。进一步的，针对动态分区（Super），仅在需要进行升级时才会在用户数据分区（Userdata）上创建虚拟动态分区，因此有效提高了数据存储空间利用率。

针对图 4 所示分区结构的操作系统改写 VC，一种逻辑上可行的应用方案是基于图 3 所示流程，在用户数据分区（Userdata）中保存操作系统升级包，该操作系统升级包包含新的 VC 以及图 4 所示的数据存储结构中各个分区的镜像数据。设备重启后进入恢复（Recovery）模式，在恢复（Recovery）模式下读取用户数据分区（Userdata）中保存的操作系统升级包，将操作系统升级包中的镜像数据恢复到各个分区，并且，使用操作系统升级包中的 VC 改写 Common 分区中保存的 VC。

然而，在采用虚拟 A/B 升级方式的安卓系统中，为了确保用户数据安全，用户数据分区（Userdata）中保存的数据是被加密的，只有在安卓系统正常启动下才可以对用户数据分区（Userdata）中保存的数据进行访问。也就是说，当设备重启进入恢复（Recovery）模式后，其无法访问用户数据分区（Userdata）中保存的数据。这样，设备就无法在恢复（Recovery）模式中读取操作系统升级包。

针对上述问题，本申请提出了一种基于虚拟 A/B 升级的操作系统升级方法。在 Common 分区中保存两套 VC，例如，如图 6a 所示，在 oeminfo 子分区中，保存 oemvc_a（第一制式文件）以及 oemvc_b（第二制式文件）。在操作系统启动的过程中，如果从静态分区（A）（第一

静态分区）启动，则在需要加载 VC 时读取并加载 oemvc_a；如果从静态分区（B）（第二静态分区）启动，则在需要加载 VC 时读取并加载 oemvc_b。

具体的，操作升级操作是由设备上安装的操作系统更新程序所完成的。具体的，升级包获取工具（update apk clent）以及升级引擎（update engine）是操作系统中的两个模块。升级包获取工具（update apk clent）用于获取操作系统的升级安装包，将操作系统升级包下载并保存到用户数据分区（Userdata）。参照 S210。

例如，升级包获取工具（update apk clent）定期向搜包服务器发起搜包请求，搜包请求包含设备当前操作系统版本号（例如版本 1.1）、设备 SN 号、产品型号、制式等信息（例如版本 1.1）；搜包服务器根据搜包请求中的操作系统版本号，检索安装包服务器上是否存在更新版本号的操作系统安装包（例如版本 1.2）；当存在更新版本的操作系统安装包时，搜包服务器向升级包获取工具（update apk clent）反馈操作系统升级包（例如，由版本 1.1 升级到版本 1.2 的操作系统升级包）的下载地址（例如，URL 地址）以及系统升级安装包对应的 filelist 文件；升级包获取工具（update apk clent）根据操作系统升级包的下载地址从安装包服务器下载操作系统升级包。

当升级包获取工具（update apk clent）获取到操作系统升级包后，其启动升级引擎（update engine），由升级引擎（update engine）根据操作系统升级包进行操作系统的升级。

具体的，当升级包获取工具（update apk clent）获取到操作系统升级包后，升级包获取工具（update apk clent）设置升级引擎（update engine）的启动属性，将启动属性设置为 true。常驻操作系统后台的服务 servicemanger 会监控升级引擎（update engine）302 的启动属性，当 servicemanger 检测到升级引擎（update engine）的启动属性为 true 时，servicemanger 启动升级引擎（update engine）。

升级包获取工具（update apk clent）通过 binder 通信获取升级引擎（update engine）的状态，当升级包获取工具（update apk clent）确认升级引擎（update engine）成功启动时，升级包获取工具（update apk clent）向升级引擎（update engine）传递升级参数（例如，当前的升级操作是文件更新操作还是文件落盘操作），触发升级引擎（update engine）进入升级流程。具体的升级流程可以参照 S520~S551。

图 6b 所示为根据本申请一实施例进行操作系统升级的流程图。设备当前的 VC 为第一制式，例如，all cn。具体的，如图 6a 所示，在设备 Common 分区的子分区中（例如 oeminfo 子分区）的 oemvc_a 为第一制式（all cn）。在设备正常启动操作系统的过程中，设备加载 Common 分区、静态分区（A）以及动态分区，从静态分区（A）启动操作系统。在加载静态分区（A）的过程中执行初始化（init）环节，在初始化环节中读取 oemvc_a 中的 VC 内容并加载；例如，从 oemvc_a 中读取到 all cn，将 all cn 写入全局变量 cmdline）。

当需要将设备的 VC 改写为第二制式时，例如，cmcc cn；设备需要将 oemvc_b 中的 VC 内容写为 cmcc cn，并且，设备启动时从静态分区（B）启动。设备执行如图 6b 所示的流程以实现改写 VC。

S1400，升级服务器推送发布用于修改 VC 的操作系统升级包。

具体的，操作系统升级包中的 VC 数据以 targe_vendor_country.mbn 文件的形式保存在操作系统升级包的根目录中，targe_vendor_country.mbn 文件的内容为 VC 数据，例如，“all/cn”、“cmcc/cn”。

具体的，操作系统升级包包含第二制式的 VC 数据（cmcc cn）。即，在操作系统升级包的根目录下存在 targe_vendor_country.mbn 文件（第三制式文件），targe_vendor_country.mbn 文件的内容为“cmcc/cn”。

进一步的，在某些应用场景中，操作系统会针对不同的 VC 做定制化处理时，操作系统升级包中还包含操作系统升级数据（包含静态分区升级数据以及动态分区升级数据）。操作系统升级数据所对应的操作系统版本与操作系统升级包中的 VC 数据匹配。

例如，针对 All cn（第一制式），其对应的操作系统为通用操作系统：操作系统版本 1.0（其并未内嵌任何网络服务商的定制应用）（第一操作系统）；针对 cmcc cn（第二制式），其对应的操作系统为中国移动定制操作系统（第二操作系统）：操作系统版本 1.01（其内嵌针对中国移动的定制应用）。那么，在操作系统升级包中就包含对应从操作系统版本 1.0 升级到操作系统版本 1.01 的操作系统升级数据，操作系统升级数据的具体内容可以参照图 5a 所示升级流程中的操作系统升级包的内容。

设备依次加载基础分区（Common）、静态分区（A）以及动态分区（Super），从静态分区（A）启动。S1400 可以在设备启动后执行，也可以在设备启动前执行。设备在从静态分区（A）启动的状态下，update apk clent 执行 S1410，获取操作系统升级包。操作系统升级包的获取过程可以参照 S510。

update apk clent 执行 S1411，触发 update engine 进入升级流程。

在升级流程中，update engine 执行 S1420，校验操作系统升级包。具体的，校验操作系统升级包的数字签名是否合法，确认操作系统升级包是否为合法的升级包。

当操作系统升级包通过校验后，update engine 执行 S1421，根据操作系统升级包升级静态分区（B）以及动态分区的数据、修改设备启动顺序，具体参照 S520、S530 以及 S531。

update engine 执行 S1422，确认当前的操作系统升级包是否为用于修改 VC 的操作系统升级包。

如果当前的操作系统升级包为用于修改 VC 的操作系统升级包，update engine 执行 S1423，将操作系统升级包中的最新版的 VC 写入 oemvc_b。

在 update engine 完成 S1423 的升级操作后，update engine 执行 S1424，向 update apk clent 返回升级操作完成的状态信息。

S1430，update apk clent 触发设备重启（第一次重启）（第三重启），例如，向用户弹框提示，用户用选择立即重启或稍后重启。

设备第一次重启后从静态分区（B）启动操作系统（第二操作系统）。参照 S540 以及 S541。

在设备加载静态分区（B）的过程中，设备会执行初始化（init）环节。在初始化环节中，设备执行 S1440，确认当前需要加载的 VC。具体的，确认加载 oemvc_a 或加载 oemvc_b。

在 S1440 中，设备根据当前加载的静态分区确定加载的 VC。由于在第一次重启之后，设备从静态分区（B）启动。因此，在 S1440 中，确定当前需要加载对应静态分区（B）的 oemvc_b。设备执行 S1441，加载 oemvc_b。

之后设备执行 S1442，启动监督初始化（cust_init），确认是否需要进行恢复出厂设置。

由于操作系统升级过程中，落盘操作是将用户数据分区（userdata）中的动态分区升级数据落盘到动态分区，而在恢复出厂设置后，用户数据分区（userdata）中的数据可能会被清空。这就使得如果在未落盘的时候进行恢复出厂设置，就会导致后续无法完成落盘操作。因此，在 S1442 中，设备首先根据当前的操作系统启动状态来确认是否需要进行恢复出厂设置。由于当前尚未进行落盘操作，操作系统升级尚未完成，因此，在 S1442 中，判定设备不需要进行恢复出厂设置。因此设备可以执行 S1443，继续进行后续的初始化、数据加载操作，完成静态分区（B）以及动态分区（Super）的加载，从静态分区（B）启动操作系统。

设备从静态分区（B）启动后，设备执行 S1450，发送开机广播。

由于此时落盘状态信息为“未落盘（wait for merge）”，update apk clent 执行 S1451，update apk clent 触发 update engine 进入 merge 流程。

S1452，update engine 执行 merge 流程，参照 S551。

merge 完成后，update engine 执行 S1455，update engine 触发设备重启（第二次重启）（第四重启）。

设备第二次重启后依次加载基础分区（Common）、静态分区（B）以及动态分区（Super），从静态分区（B）启动。

在设备第二次重启后，在设备加载静态分区(B)的初始化(init)环节。设备再次执行 S1440，确认当前需要加载的 VC。参照首次执行 S1440 的描述，在第二次重启之后，设备从静态分区（B）启动。因此，当前需要加载 oemvc_b。设备执行 S1441，加载 oemvc_b。

之后设备再次执行 S1442，启动监督初始化(cust_init)，确认是否需要进行恢复出厂设置。

由于当前已进行落盘操作，因此，在 S1442 中，进一步根据其他条件确认是否需要进行恢复出厂设置。

具体的，在设备启动操作系统的过程中，当设备操作系统自身的 VC（oem vc）与设备保存在 Userdata 的 Custom.bin 不匹配时，操作系统的运行就可能出现错误。因此，在监督初始化（cust_init）环节，在当前操作系统的状态为已落盘时，需要验证操作系统自身的 VC 与设备保存在 Userdata 的 Custom.bin 是否匹配，如果不匹配，则需要进行恢复出厂设置，重新配置操作系统参数。

具体的，在 S1442 中，在当前的落盘状态为已落盘时，确认当前加载的 oemvc_b 与设备当前的配置（保存在 Userdata 的 Custom.bin）是否匹配。

由于当前 oemvc_b 已被更新，为新版的 VC（第二制式）。而设备当前的配置（保存在 Userdata 的 Custom.bin）是根据操作系统原始的 oemvc_a（第一制式）所生成的，因此，oemvc_b 与 Custom.bin 不匹配，Custom.bin 设备需要进行恢复出厂设置。

因此，设备执行 S1460，触发设备重启（第三次重启）（第一重启）。

设备第三次重启后进入恢复（Recovery）模式，在 Recovery 模式下，设备执行 S1461，恢复出厂设置。

设备完成恢复出厂设置后，执行 S1462，触发设备重启（第四次重启）（第二重启）。

设备第四次重启后依次加载基础分区（Common）、静态分区（B）以及动态分区（Super），从静态分区（B）启动。

在设备第四次重启后，在设备加载静态分区（B）的初始化（init）环节。设备再次执行 S1440，确认当前需要加载的 VC。

在第四次重启之后，设备从静态分区（B）启动。因此，当前需要加载 oemvc_b。设备执行 S1441，加载 oemvc_b。

之后设备再次执行 S1442，启动监督初始化（cust_init），确认是否需要进行恢复出厂设置。

由于当前已进行落盘操作，因此，在 S1442 中，进一步确认当前加载的 oemvc_b 与设备当前的配置（保存在 Userdata 的 Custom.bin）是否匹配。在这个环节中，由于在 S1461 中，设备恢复出厂设置，在恢复出厂设置的过程中，保存在 Userdata 的 Custom.bin 被删除，因此设备无法读取到 Custom.bin，当前 Userdata 中不存在 Custom.bin。在设备无法读取到 Custom.bin 时，设备不需要进行恢复出厂设置。

在设备无法读取到 Custom.bin 时，设备执行 S1445，创建 Custom.bin，写 oemvc_b 到 Custom.bin。

之后设备执行 S1443，继续进行后续的初始化、数据加载操作，完成静态分区（B）以及动态分区（Super）的加载，从静态分区（B）启动。

设备从静态分区（B）启动后，设备执行 S1450，发送开机广播。update apk clent 确认已完成 merge，操作系统完成升级。

根据图 6b 所示实施例的方法，可以针对采用虚拟 A/B 升级方案的操作系统实现改制；根据图 6b 所示实施例的方法，不需要配置额外的改制工具，设备可以通过下载操作系统升级包自行完成改制操作，从而大大简化改制流程，降低改制难度。

进一步的，根据图 6b 所示实施例的方法，可以在改制的同时实现对静态分区以及动态分区数据的更新，使得在对操作系统改制的同时将操作系统升级到对应的定制操作系统版本，避免进行多次操作系统升级操作，大大简化了操作系统的升级流程，提高了用户体验。

具体的，图 7 所示为根据本申请一实施例进行操作系统升级的部分流程图。设备执行图 7 所示的流程以实现图 6b 所示的流程中第一次重启之前的操作。

S1500，设备依次加载基础分区（Common）、静态分区（A）以及动态分区（Super），从静态分区（A）启动。

S1510，升级包获取工具（update apk clent）获取操作系统升级包，操作系统升级包的获取过程可以参照 S510。操作系统升级包的具体文件结构可以参照 S600 中的描述。

S1511，升级包获取工具（update apk clent）启动升级引擎（update engine）进入升级流程。

S1512，升级引擎（update engine）校验操作系统升级包的签名是否合法。

在操作系统升级包通过签名校验后，升级引擎（update engine）执行 S1520，根据操作系统升级包中的操作系统升级数据升级静态分区（B）以及动态分区（Super），具体执行参照 S520、S530 以及 S531。

在 S1520 之后（在升级引擎（update engine）执行 S520、S530 以及 S531 之后），升级引擎（update engine）还执行下述步骤：

S1521，确认当前的操作系统升级包是否为用于修改 VC 的操作系统升级包。具体的，检测 S1510 所获取的操作系统升级包的根目录下是否存在 targe_vendor_country.mbn 文件（第三

制式文件）。

如果操作系统升级包的根目录下不存在 targe_vendor_country.mbn 文件，则说明本次操作系统升级不需要改写 VC，执行 S1522，重启设备，从静态分区（B）启动并执行落盘操作，参照 S532~SS551。

如果操作系统升级包的根目录下存在 targe_vendor_country.mbn 文件，则说明本次操作系统升级需要改写 VC，执行 S1523，将 targe_vendor_country.mbn 文件中的 VC（cmcc cn）写入 oemvc_b。

S1524，将制式改写标志位（oem）置为 1（第一值）。

制式改写标志位（oem）用于标识是否执行了新制式写入。制式改写标志位（oem）置为 1（第一值）标识执行了新制式写入，制式改写标志位（oem）置为 0（第二值）标识未执行新制式写入。

制式改写标志位（oem）可以保存在 Common 分区的元数据（/metadata）中，也可以保存在 Common 分区中用于保存 oem-vc 的子分区中（例如，oeminfo 子分区）。

进一步的，在实际应用场景中，S1523 以及 S1524 之间的执行先后顺序并不需要强制性限定。在另一实施例中，也可以先执行 S1524，再执行 S1523。

在 S1524 之后，升级引擎（update engine）执行 S1525，向升级包获取工具（update apk clent）上报当前状态（已完成 oemvc_b 的写入）。

升级包获取工具（update apk clent）接收到升级引擎（update engine）的状态上报后，执行 S1530，弹出对话框，提示用户当前的操作系统升级需要重启设备。

具体的，S1511~ S1524 的执行均可以在用户正常使用手机的过程中后台执行。

具体的，图 8 为根据本申请一实施例的手机运行界面示意图。当手机顺利启动操作系统后，进入如图 8 中 801 所示的系统界面。手机进行操作系统升级时，用户可以调出图 8 中 802 所示的界面，从而向用户展示操作系统升级进度。用户也可以关闭图 8 的 802 所示的操作界面，正常使用手机，在用户关闭 802 所示的界面后，S711~ S724 的执行转为后台执行。例如手机显示 801 或 803 所示的界面。

在 S730 中，当手机需要重启时，手机向用户输出重启提示，由用户确认是否立即重启。

例如，图 9 为根据本申请一实施例的手机运行界面示意图。手机当前展示如图 8 中 802 所示的界面，向用户展示操作系统升级进度。在 S730 中，手机展示如图 9 中 903 所示的界面，由用户确认立即重启或是稍后重启。

又例如，手机当前展示如图 8 中 803 所示的界面，用户使用聊天应用。在 S730 中，手机展示如图 9 中 901 所示的界面，弹出提示通知。用户点击提示通知，进入如图 9 中 903 所示的界面，由用户确认立即重启或是稍后重启。或者，用户打开下拉通知栏，手机展示如图 9 中 902 所示的界面。在下拉通知栏中，用户点击提示通知，进入如图 9 中 903 所示的界面，由用户确认立即重启或是稍后重启。

进一步的，在 S730 中，如果用户点击重启按钮，执行 S731，设备立即重启。

在 S730 中，如果用户点击稍后按钮，执行 S732，设备暂停升级流程，并在预设的定时升级节点重新启动升级流程，升级流程启动后设备重启。

图 10 所示为根据本申请一实施例进行操作系统升级的部分流程图。设备执行图 10 所示的步骤以实现图 6b 所示流程中的初始化环节。

具体的，由于在 S1520 中，升级引擎（update engine）根据操作系统升级包中的操作系统升级数据升级静态分区（B）以及动态分区（Super）（升级引擎（update engine）执行了 S531），因此在 S1531 或 S1132 之后，设备重启后从静态分区（B）启动操作系统。操作系统的启动流程可以参照 S540 以及 S541。在设备加载静态分区（B）的过程中，设备执行初始化（init）环节，在初始化（init）环节，设备执行图 10 所示的下述步骤：

S1600，确认当前加载的静态分区。例如，当前加载的静态分区为静态分区（B）。

S1610，读取并加载静态分区对应的 oem-vc。具体的，当前加载的静态分区为静态分区（B），将 oemvc_b（cmcc cn）写入到全局变量 cmdline 中。

S1620，启动监督初始化（Cust_init）。S1620 的执行参照 S1442。

进一步的，为了避免判断错误导致设备意外跳过恢复出厂设置步骤，在 S1620 中，除了基于落盘状态确认是否需要进行进一步的判断是否进行恢复出厂设置（根据操作系统 VC 与 Custom.bin 是否一致判断是否进行恢复出厂设置）之外，还根据本次启动前是否写入新制式来确认是否需要进行进一步的判断是否进行恢复出厂设置。具体的，在 S1620 中，执行下述步骤。

S1630，判断当前的落盘状态，例如，读取基础分区（Common）的元数据分区（/metadata）中的落盘状态信息。由于在 S1520 中，升级引擎（update engine）根据操作系统升级包中的操作系统升级数据升级静态分区（B）以及动态分区（Super）（升级引擎（update engine）执行了 S530），因此在 S1630 中，落盘状态信息为“未落盘（wait for merge）”。

当落盘状态信息为“未落盘（wait for merge）”时，执行 S1631，读取制式改写标志位（oem）；由于在 S1524 中，将制式改写标志位（oem）置为 1，因此，在 S1631 中，制式改写标志位 oem=1。

当 oem=1 时，设备执行 S1640，执行后续的初始化操作以及静态分区（B）加载操作（不再进一步的判断是否进行恢复出厂设置）。

设备完成加载静态分区（B）之后，设备继续加载动态分区，启动操作系统。参照 S541 以及 S550。

图 11a 所示为根据本申请一实施例进行操作系统升级的部分流程图。设备执行图 11a 所示流程以实现图 6b 所示流程中第二次重启之前的操作。在操作系统启动运行后（在 S550 或 S643 之后），系统发送开机广播，启动升级包获取工具（update apk clent）。由于落盘状态信息为“未落盘（wait for merge）”，升级包获取工具（update apk clent）触发执行落盘操作。升级包获取工具（update apk clent）启动升级引擎（update engine），升级引擎（update engine）执行如图 11a 所示的下述操作：

S1700，执行落盘操作，参照 S551（在 S551 之后，落盘状态信息为“已落盘（merged）”）。S1730，将制式改写标志位（oem）置为 0（第二值）。

在 S1730 之后，升级引擎（update engine）触发设备重启（S1740）。

进一步的，为了在操作系统升级后，从静态分区（A）以及静态分区（B）均能顺利启动设备。在 S1740 之前，升级引擎（update engine）还执行下述步骤。

S1710，将静态分区（B）的数据同步到静态分区（A）。参照 1110。

S1720，将 oemvc_b（cmcc cn）同步到 oemvc_a。具体的，将 oemvc_a 由 All cn 改写为 cmcc cn。

具体的，本申请对 S1710 的具体实现方式不做具体限制，本领域的技术人员可以采用多种可行的实现方式实现 S1710。

例如，图 11b 所示为根据本申请一实施例进行操作系统升级的部分流程图。设备执行如图 11b 所示的下述流程以实现 S1710。

S1800，读取设备存储器上与分区表相关的描述数据（该参数在设备出厂时预存在设备中），合成存储器的总分区表。

例如，以采用主引导记录（Master Boot Record，MBR）格式的通用闪存（Universal Flash Storage，UFS）。从 UFS 的 MBR（主引导扇区，UFS 的第一个扇区，即 C/H/S 地址的 0 柱面 0 磁头 1 扇区）中读取 UFS 上各个分区的大小及位置信息，获取分区表（Dpt）。

S1810，从总分区表中读取后缀名为_b 的所有静态子分区，生成用于描述静态分区（B）各个子分区的列表 1，列表 1 包括静态分区（B）中各个子分区的名称以及地址。例如：

编号	子分区名称	子分区地址（文件路径）	被选定状态
1	bootloader_b	/dev/block/by-name/bootloader_b	0
2	boot_b	/dev/block/by-name/boot_b	0
3	vendor_boot_b	/dev/block/by-name/vendor_boot_b	0
4	dtbo_b	/dev/block/by-name/dtbo_b	0
5	vbmeta_b	/dev/block/by-name/vbmeta_b	0

表 1

S1820，从总分区表中读取后缀名为_a 的所有静态子分区，生成用于描述静态分区（A）各个子分区的列表 2，列表 2 包括静态分区（A）中各个子分区的名称以及地址。例如：

编号	子分区名称	子分区地址（文件路径）
1	bootloader_a	/dev/block/by-name/bootloader_a
2	boot_a	/dev/block/by-name/boot_a
3	vendor_boot_a	/dev/block/by-name/ vendor_boot_a
4	dtbo_a	/dev/block/by-name/dtbo_a
5	vbmeta_a	/dev/block/by-name/vbmeta_a

表 2

这里需要说明的是，在表 1 以及表 2 中，以文件路径的方式指代该子分区的地址，在实际应用场景中，本领域的技术人员可以使用多种不同的方式描述子分区的地址。例如，采用线性地址描述。

S1830，在列表 1 中选定一个未被选定过的子分区（第一子分区），获取该子分区的名称（第一子分区名称）以及地址（第一文件路径）。

具体的，在 S1830 之前，列表 1 中的子分区均未被选定。在 S1830 中，可以按照列表 1 中

子分区的排列顺序（编号顺序）依次选定子分区，也可以从所有未被选定过的子分区中随机选定。

进一步的，在选定一个子分区后，标记该子分区以便在后续确认该子分区是否被选定过。例如，如表 1 所示，在表 1 中增加被选定状态列，被选定状态的初始值为 0，如子分区被选定，则被选定状态修改为 1。

S1840，将 S1830 中选定的子分区与列表 2 中的各个子分区做去后缀匹配；确定列表 2 中，去掉后缀后，与 S1830 中选定的子分区名称一致的子分区（第二子分区名称）以及在列表 2 中，该第二子分区名称对应的子分区地址（第二文件路径）；

S1841，读取第一文件路径下的数据；

S1842，将读取到的数据覆写到第二文件路径下。

S1850，判断列表 1 中是否还存在未被选定过的子分区；

如果存在，返回步骤 S1830，重新选定第一子分区；

如果不存在，静态分区同步结束。

以表 1 以及表 2 为例，在一应用场景中，设备执行下述流程：

选定表 1 中被选定状态为 0 的第一个子分区（编号 1 的 bootloader_b 子分区），将编号 1 的被选定状态修改为 1；

使用 bootloader_b 在表 2 中的所有子分区名称中做去后缀匹配，bootloader_a 与 bootloader_b 在分别去掉_a 以及_b 后一致，因此，根据 bootloader_b 匹配到 bootloader_a（第二子分区）；

从表 1 中读取到 bootloader_b 对应的文件路径/dev/block/by-name/bootloader_b（第一文件路径）；

从表 2 中读取到 bootloader_a 对应的文件路径/dev/block/by-name/bootloader_a（第二文件路径）；

读取/dev/block/by-name/bootloader_b 下的数据，将读取到的数据覆写到/dev/block/by-name/bootloader_a；

表 1 中仍存在被选定状态为 0 的子分区，选定表 1 中被选定状态为 0 的第一个子分区（编号 2 的 boot_b 子分区），将编号 2 的被选定状态修改为 1；

使用 boot_b 在表 2 中的所有子分区名称中做去后缀匹配，boot_a 与 boot_b 在分别去掉_a 以及_b 后一致，因此，根据 boot_b 匹配到 boot_a；

从表 1 中读取到 boot_b 对应的文件路径/dev/block/by-name/boot_b；

从表 2 中读取到 boot_a 对应的文件路径/dev/block/by-name/boot_a；

读取/dev/block/by-name/boot_b 下的数据，将读取到的数据覆写到/dev/block/by-name/boot_a；

表 1 中仍存在被选定状态为 0 的子分区，选定表 1 中被选定状态为 0 的第一个子分区（编号 3 的 vendor_boot_b 子分区），将编号 3 的被选定状态修改为 1；

使用 vendor_boot_b 在表 2 中的所有子分区名称中做去后缀匹配，vendor_boot_a 与 vendor_boot_b 在分别去掉_a 以及_b 后一致，因此，根据 vendor_boot_b 匹配到 vendor_boot_a；

从表 1 中读取到 vendor_boot_b 对应的文件路径/dev/block/by-name/vendor_boot_b；

从表 2 中读取到 vendor_boot_a 对应的文件路径/dev/block/by-name/vendor_boot_a；

读取/dev/block/by-name/vendor_boot_b 下的数据，将读取到的数据覆写到/dev/block/by-name/vendor_boot_a；

表 1 中仍存在被选定状态为 0 的子分区，选定表 1 中被选定状态为 0 的第一个子分区（编号 4 的 dtbo_b 子分区），将编号 4 的被选定状态修改为 1；

使用 dtbo_b 在表 2 中的所有子分区名称中做去后缀匹配，dtbo_a 与 dtbo_b 在分别去掉_a 以及_b 后一致，因此，根据 dtbo_b 匹配到 dtbo_a；

从表 1 中读取到 dtbo_b 对应的文件路径/dev/block/by-name/dtbo_b；

从表 2 中读取到 vendor_boot_a 对应的文件路径/dev/block/by-name/dtbo_a；

读取/dev/block/by-name/dtbo_b 下的数据，将读取到的数据覆写到/dev/block/by-name/dtbo_a；

表 1 中仍存在被选定状态为 0 的子分区，选定表 1 中被选定状态为 0 的第一个子分区（编号 5 的 vbmeta_b 子分区），将编号 5 的被选定状态修改为 1；

使用 vbmeta_b 在表 2 中的所有子分区名称中做去后缀匹配，vbmeta_a 与 vbmeta_b 在分别去掉_a 以及_b 后一致，因此，根据 vbmeta_b 匹配到 vbmeta_a；

从表 1 中读取到 vbmeta_b 对应的文件路径/dev/block/by-name/vbmeta_b；

从表 2 中读取到 vendor_boot_a 对应的文件路径/dev/block/by-name/vbmeta_a；

读取/dev/block/by-name/vbmeta_b 下的数据，将读取到的数据覆写到/dev/block/by-name/vbmeta_a；

表 1 中不存在被选定状态为 0 的子分区，静态分区同步完成。

在 S1740 之后，由于在 S1520 中，升级引擎（update engine）根据操作系统升级包中的操作系统升级数据升级静态分区（B）以及动态分区（Super）（升级引擎（update engine）执行了 S531），并且，在图 10 以及图 11a 所示流程中，并未修改设备的启动顺序，因此在 S1740 之后，设备重启后从静态分区（B）启动操作系统。操作系统的启动流程可以参照 S540 以及 S541。在设备加载静态分区（B）的过程中，设备执行初始化（init）环节，在初始化（init）环节，设备再次执行图 10 所示步骤。

具体的，由于在 S1700 中执行落盘操作，落盘状态信息为“已落盘（merged）”。因此，在 S1650 之后，在再次执行的 S1630 中，落盘状态信息为“已落盘（merged）”。

当落盘状态信息为“已落盘（merged）”时，执行 S1632，判断当前加载的静态分区对应的 VC 与用户数据分区（Userdata）中的定制信息（Data/Custom.bin）是否一致。具体的，当前加载的静态分区为静态分区（B），对应的 VC 为 oemvc_b（cmcc cn）。由于 Data/Custom.bin 在之前的流程中并未被改写，因此在 S1632 中，Custom.bin 对应 S1720 之前的 oemvc_a（All cn），oemvc_b（cmcc cn）与 Custom.bin 不一致。

进一步的，当落盘状态信息为“未落盘（wait for merge）”，执行 S1631，读取制式改写标志位（oem）。当 oem=0 时，也执行 S1632。

当前加载的静态分区（静态分区（B））对应的 VC（oemvc_b）与用户数据分区（Userdata）中的定制信息（Data/Custom.bin）不一致时，设备执行 S1650，恢复出厂设置。

图 12 所示为根据本申请一实施例进行操作系统升级的部分流程图。在 S1650 中，设备执行如图 12 所示的下述步骤：

S1200，设备重启；

S1210，执行恢复出厂设置操作，包括：

格式化 data/metadata/cache；

删除 data/Custom.bin；

S1220，恢复出厂设置后，设备重启。

在 S1220 之后，设备重启后从静态分区（B）启动操作系统。操作系统的启动流程可以参照 S540 以及 S541。在设备加载静态分区（B）的过程中，设备执行初始化（init）环节，在初始化（init）环节，设备再次执行图 10 所示步骤。

具体的，由于在 S1650 中，Data/Custom.bin 已被删除，因此在再次执行的 S1632 中，Data/Custom.bin 不存在。

当 Data/Custom.bin 不存在时，设备执行 S1651，创建 Data/Custom.bin，并将当前加载的静态分区对应的 oem-vc（oemvc_b）写入 Custom.bin。

在 S1651 之后，设备执行 S1640，执行后续的初始化操作以及静态分区（B）加载操作。由于在 S1700 中，设备已完成落盘操作，因此，在设备完成加载静态分区（B）之后，设备正常加载动态分区（Super），操作系统正常启动。

进一步的，在图 6 所示实施例中，在第二次重启之后的初始化环节判断是否需要恢复出厂设置，并确认需要恢复出厂设置。在另一种可行的方案中，可以跳过第二次重启之后的恢复出厂设置判断，在第二次重启之后直接进入恢复（Recovery）模式，在 Recovery 模式下，设备恢复出厂设置。

图 13 所示为根据本申请一实施例进行操作系统升级的流程图。设备当前的 VC 为第一制式，例如，all cn。具体的，如图 6a 所示，在设备 Common 分区的子分区中（例如 oeminfo 子分区）的 oemvc_a 为第一制式（all cn）。在设备正常启动操作系统的过程中，设备加载 Common 分区、静态分区（A）以及动态分区，从静态分区（A）启动操作系统。在加载静态分区（A）的过程中执行初始化（init）环节，在初始化环节中读取 oemvc_a 中的 VC 内容并加载；例如，从 oemvc_a 中读取到 all cn，将 all cn 写入全局变量 cmdline）。

当需要将设备的 VC 改写为第二制式时，例如，cmcc cn；设备需要将 oemvc_b 中的 VC 内容写为 cmcc cn，并且，设备启动时从静态分区（B）启动。设备执行如图 13 所示的流程以实现改写 VC。

S1300、S1310、S1321、S1320、S1321、S1322、S1323、S1324 以及 S1330，参照 S1400、S1410、S1421、S1420、S1421、S1422、S1423、S1424 以及 S1430。

在 S1324 之后，update apk clent 执行 S1330，触发设备重启（第一次重启）（第三重启）。

S1330 后从静态分区（B）启动操作系统（第二操作系统）。参照 S540 以及 S541。

在设备加载静态分区（B）的过程中，设备会执行初始化（init）环节。在初始化环节中，设备执行 S1340，确认当前需要加载的 VC。具体的，确认加载 oemvc_a 或加载 oemvc_b。参

照 S1440。

设备执行 S1341，加载 oemvc_b。

之后设备执行 S1342，启动监督初始化（cust_init），确认是否需要进行恢复出厂设置。

由于当前尚未进行落盘操作，操作系统升级尚未完成，因此，设备不需要进行恢复出厂设置。因此设备可以执行 S1343，继续进行后续的初始化、数据加载操作，完成静态分区（B）以及动态分区（Super）的加载，从静态分区（B）启动操作系统。

设备从静态分区（B）启动后，设备执行 S1350，发送开机广播。

由于此时落盘状态信息为“未落盘（wait for merge）”，update apk clent 执行 S1351，update apk clent 触发 update engine 进入 merge 流程。

S1352，update engine 执行 merge 流程，参照 S551。

merge 完成后，update engine 执行 S1355，update engine 触发设备重启（第二次重启）（第一重启）。

设备第二次重启后进入恢复（Recovery）模式，在 Recovery 模式下，设备执行 S1361，恢复出厂设置。

设备完成恢复出厂设置后，执行 S1362，触发设备重启（第三次重启）（第二重启）。

设备第三次重启后依次加载基础分区（Common）、静态分区（B）以及动态分区（Super），从静态分区（B）启动。

在设备第三次重启后，在设备加载静态分区(B)的初始化(init)环节。设备再次执行 S1340，确认当前需要加载的 VC。设备执行 S1441，加载 oemvc_b。

之后设备再次执行 S1342，启动监督初始化(cust_init)，确认是否需要进行恢复出厂设置。参照 S1442。

当前 Userdata 中不存在 Custom.bin。因此，设备不需要进行恢复出厂设置。

设备执行 S1345，创建 Custom.bin，写 oemvc_b 到 Custom.bin。

之后设备执行 S1343，继续进行后续的初始化、数据加载操作，完成静态分区（B）以及动态分区（Super）的加载，从静态分区（B）启动。

设备从静态分区（B）启动后，设备执行 S1350，发送开机广播。update apk clent 确认已完成 merge，操作系统完成升级。

进一步的，在某些应用场景中，操作系统并未针对不同的 VC 做定制化处理时，操作系统升级包中不包含操作系统升级数据(不包含静态分区升级数据以及动态分区升级数据)。因此，设备在改写 VC 过程中也就不需要执行落盘操作（S1452）。基于此，在另一种可行的方案中，可以跳过第一次重启之后的落盘操作，在第一次重启之后直接进入恢复（Recovery）模式，在 Recovery 模式下，设备恢复出厂设置。

图 14 所示为根据本申请一实施例进行操作系统升级的流程图。设备当前的 VC 为第一制式，例如，all cn。具体的，如图 6a 所示，在设备 Common 分区的子分区中（例如 oeminfo 子分区）的 oemvc_a 为第一制式(all cn)。在设备正常启动操作系统的过程中，设备加载 Common 分区、静态分区（A）以及动态分区，从静态分区（A）启动操作系统。在加载静态分区（A）

的过程中执行初始化（init）环节，在初始化环节中读取 oemvc_a 中的 VC 内容并加载；例如，从 oemvc_a 中读取到 all cn，将 all cn 写入全局变量 cmdline）。

当需要将设备的 VC 改写为第二制式时，例如，cmcc cn；设备需要将 oemvc_b 中的 VC 内容写为 cmcc cn，并且，设备启动时从静态分区（B）启动。设备执行如图 14 所示的流程以实现改写 VC。

S1900，升级服务器推送发布用于修改 VC 的操作系统升级包。具体的，操作系统升级包不包含操作系统升级数据。操作系统升级包中的 VC 数据的格式可以参照 S1900。

设备依次加载基础分区（Common）、静态分区（A）以及动态分区（Super），从静态分区（A）启动。update apk clent 执行 S1910，获取操作系统升级包。操作系统升级包的获取过程可以参照 S510。

update apk clent 执行 S1911，触发 update engine 进入升级流程。

在升级流程中，update engine 执行 S1920，校验操作系统升级包。具体的，校验操作系统升级包的数字签名是否合法，确认操作系统升级包是否为合法的升级包。

当操作系统升级包通过校验后，update engine 执行 S1921，确认操作系统升级包中是否包含操作系统升级数据。

当操作系统升级包中不包含操作系统升级数据时，update engine 执行 S1922，确认当前的操作系统升级包是否为用于修改 VC 的操作系统升级包。

如果当前的操作系统升级包为用于修改 VC 的操作系统升级包，update engine 执行 S1923，将操作系统升级包中的最新版的 VC 写入 oemvc_b，并且，将操作系统的启动顺序修改为从静态分区（B）启动。

在 update engine 完成 S1923 的升级操作后，update engine 执行 S1924，向 update apk clent 返回升级操作完成的状态信息。

S1930，update apk clent 触发设备重启（第一次重启）（第一重启），例如，向用户弹框提示，用户用选择立即重启或稍后重启。

设备第一次重启后进入恢复（Recovery）模式，在 Recovery 模式下，设备执行 S1961，恢复出厂设置。

设备完成恢复出厂设置后，执行 S1962，触发设备重启（第二次重启）（第二重启）。

设备第二次重启后依次加载基础分区（Common）、静态分区（B）以及动态分区（Super），从静态分区（B）启动。

在设备第二次重启后，在设备加载静态分区（B）的初始化（init）环节。设备执行 S1940，确认当前需要加载的 VC。在第二次重启之后，设备从静态分区（B）启动。因此，当前需要加载 oemvc_b。设备执行 S1941，加载 oemvc_b。

之后设备执行 S1942，启动监督初始化（cust_init），确认是否需要进行恢复出厂设置。

设备确认当前加载的 oemvc_b 与设备当前的配置（保存在 Userdata 的 Custom.bin）是否匹配。由于在 S1961 中，设备恢复出厂设置，在恢复出厂设置的过程中，保存在 Userdata 的 Custom.bin 被删除。当前 Userdata 中不存在 Custom.bin。因此，设备不需要进行恢复出厂设置。设备执行 S1945，创建 Custom.bin，写 oemvc_b 到 Custom.bin。

之后设备执行 S1943，继续进行后续的初始化、数据加载操作，完成静态分区（B）以及

动态分区（Super）的加载，从静态分区（B）启动。

设备从静态分区（B）启动后，设备执行 S1950，发送开机广播。update apk client 确认已完成 merge，操作系统完成升级。

可以理解的是，上述实施例中的部分或全部步骤或操作仅是示例，本申请实施例还可以执行其它操作或者各种操作的变形。此外，各个步骤可以按照上述实施例呈现的不同的顺序来执行，并且有可能并非要执行上述实施例中的全部操作。

进一步的，一般的，对于一个技术的改进可以很明显地区分是硬件上的改进（例如，对二极管、晶体管、开关等电路结构的改进）还是软件上的改进（对于方法流程的改进）。然而，随着技术的发展，当今的很多方法流程的改进已经可以视为硬件电路结构的直接改进。设计人员几乎都通过将改进的方法流程编程到硬件电路中来得到相应的硬件电路结构。因此，不能说一个方法流程的改进就不能用硬件实体模块来实现。例如，可编程逻辑器件（Programmable Logic Device, PLD）（例如现场可编程门阵列（Field Programmable Gate Array，FPGA））就是这样一种集成电路，其逻辑功能由访问方对器件编程来确定。由设计人员自行编程来把一个数字装置“集成”在一片 PLD 上，而不需要请芯片制造厂商来设计和制作专用的集成电路芯片。而且，如今，取代手工地制作集成电路芯片，这种编程也多半改用“逻辑编译器（logic compiler）”软件来实现，它与程序开发撰写时所用的软件编译器相类似，而要编译之前的原始代码也得用特定的编程语言来撰写，此称之为硬件描述语言（Hardware Description Language，HDL），而 HDL 也并非仅有一种，而是有许多种，如 ABEL（Advanced Boolean Expression Language）、AHDL（Altera Hardware Description Language）、Confluence、CUPL（Cornell University Programming Language）、HDCal、JHDL（Java Hardware Description Language）、Lava、Lola、MyHDL、PALASM、RHDL（Ruby Hardware Description Language）等，目前最普遍使用的是 VHDL（Very-High-Speed Integrated Circuit Hardware Description Language）与 Verilog。本领域技术人员也应该清楚，只需要将方法流程用上述几种硬件描述语言稍作逻辑编程并编程到集成电路中，就可以很容易得到实现该逻辑方法流程的硬件电路。

因此，本申请实施例所提出的方法流程可以以硬件方式实现，例如，使用控制器，控制器控制触摸屏以实现本申请实施例所提出的方法流程。

控制器可以按任何适当的方式实现，例如，控制器可以采取例如微处理器或处理器以及存储可由该（微）处理器执行的计算机可读程序代码（例如软件或固件）的计算机可读介质、逻辑门、开关、专用集成电路（Application Specific Integrated Circuit，ASIC）、可编程逻辑控制器和嵌入微控制器的形式，控制器的例子包括但不限于以下微控制器：ARC 625D、Atmel AT91SAM、Microchip PIC18F26K20 以及 Silicone Labs C8051F320，存储器控制器还可以被实现为存储器的控制逻辑的一部分。本领域技术人员也知道，除了以纯计算机可读程序代码方式实现控制器以外，完全可以通过将方法步骤进行逻辑编程来使得控制器以逻辑门、开关、专用集成电路、可编程逻辑控制器和嵌入微控制器等的形式来实现相同功能。因此这种控制器可以被认为是一种硬件部件，而对其内包括的用于实现各种功能的装置也可以视为硬件部件内的结构。或者甚至，可以将用于实现各种功能的装置视为既可以是实现方法的软件模块又可以是硬件部件内的结构。

与上述实施例对应，本申请还提供了一种电子设备。电子设备包括用于存储计算机程序指令的存储器和用于执行程序指令的处理器，其中，当该计算机程序指令被该处理器执行时，触发电子设备执行如本申请实施例所述的方法步骤。

本申请还提供一种计算机程序产品，计算机程序产品包括计算机程序，当其在计算机上运行时，使得计算机执行本申请实施例提供的部分或全部步骤。

本领域的技术人员可以清楚地了解到本发明实施例中的技术可借助软件加必需的通用硬件平台的方式来实现。基于这样的理解，本发明实施例中的技术方案本质上或者说对现有技术做出贡献的部分可以以软件产品的形式体现出来，该计算机软件产品可以存储在存储介质中，如 ROM/RAM、磁碟、光盘等，包括若干指令用以使得一台计算机设备（可以是个人计算机，服务器，或者网络设备等）执行本发明各个实施例或者实施例的某些部分所述的方法。

本说明书中各个实施例之间相同相似的部分互相参见即可。尤其，对于装置实施例和终端实施例而言，由于其基本相似于方法实施例，所以描述的比较简单，相关之处参见方法实施例中的说明即可。

权 利 要 求 书

1.一种配置操作系统制式的方法，其特征在于，应用于电子设备，所述电子设备包括处理器以及存储器，所述存储器包括基础分区、第一静态分区、第二静态分区、动态分区以及用户数据分区；所述电子设备启动操作系统的过程包括初始化环节，所述基础分区中保存有第一制式文件以及第二制式文件，所述电子设备从所述第一静态分区启动操作系统时，所述初始化环节中加载所述第一制式文件；所述电子设备从所述第二静态分区启动操作系统时，所述初始化环节中加载所述第二制式文件；所述第一制式文件当前的制式内容为第一制式，所述电子设备启动后加载所述基础分区、所述第一静态分区以及动态分区的数据以从所述第一静态分区启动所述操作系统；所述操作系统启动之后，所述方法包括：

获取操作系统升级包，所述操作系统升级包包括第三制式文件，所述第三制式文件的制式内容为第二制式；

修改所述电子设备的启动顺序为从所述第二静态分区启动；

提取所述第三制式文件；

将所述第三制式文件的制式内容写入所述第二制式文件；

触发所述电子设备的第一重启，

所述第一重启之后所述电子设备进入恢复模式；

在所述恢复模式下，恢复所述电子设备的出厂设置，包括，删除所述用户数据分区中保存的制式配置文件；

触发所述电子设备的第二重启，所述第二重启之后所述电子设备从所述第二静态分区启动操作系统，所述第二重启之后的初始化环节至少包括：在所述用户数据分区中创建所述制式配置文件，将所述第二制式文件的制式内容写入所述制式配置文件。

2.根据权利要求 1 所述的方法，其特征在于，所述获取操作系统升级包之后，所述方法还包括：

确认所述操作系统升级包是否用于改写制式，当所述操作系统升级包用于改写制式时，执行所述提取所述第三制式文件的步骤。

3.根据权利要求 2 所述的方法，其特征在于，确认所述操作系统升级包是否用于改写制式，包括，确认所述操作系统升级包中是否存在所述第三制式文件。

4.根据权利要求 1 所述的方法，其特征在于，所述第二重启之后的初始化环节还包括：

确认所述用户数据分区中是否存在所述制式配置文件，当所述用户数据分区中不存在所述制式配置文件时，在所述用户数据分区中创建所述制式配置文件，将所述第二制式文件的制式内容写入所述制式配置文件。

5.根据权利要求 1 所述的方法，其特征在于，第一操作系统以及第二操作系统为对应不同制式的操作系统，所述第一操作系统对应所述第一制式，所述第二操作系统对应所述第二制式；所述操作系统升级包还包括动态分区升级数据，所述动态分区升级数据用于将所述第一操作系统的动态分区数据更新为所述第二操作系统的动态分区数据；所述获取操作系统升级包之前，所述电子设备启动后加载所述基础分区、所述第一静态分区以及动态分区的数据以从所述第一静态分区启动所述第一操作系统；

所述修改所述电子设备的启动顺序为从所述第二静态分区启动之前，方法还包括：

在所述用户数据分区中创建虚拟动态分区，将所述动态分区升级数据写入到所述虚拟动态分区；

所述将所述第三制式文件的制式内容写入所述第二制式文件之后，所述触发所述电子设备的第一重启之前，所述方法还包括：

触发所述电子设备的第三重启，

所述第三重启之后所述电子设备从所述第二静态分区启动操作系统的过程包括：加载所述基础分区、所述第二静态分区、所述动态分区以及所述虚拟动态分区的数据以启动所述第二操作系统；

所述第三重启之后所述电子设备启动所述第二操作系统之后，将所述虚拟动态分区的数据落盘到所述动态分区。

6 根据权利要求 5 所述的方法，其特征在于，所述操作系统升级包还包括静态分区升级数据，所述静态分区升级数据用于将所述第一操作系统的静态分区数据更新为所述第二操作系统的静态分区数据；

所述修改所述电子设备的启动顺序为从所述第二静态分区启动之前，方法还包括：基于所述静态分区升级数据更新所述第二静态分区的数据。

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：

所述第三重启之后所述电子设备启动所述第二操作系统之后，将所述第二静态分区的数据同步到所述第一静态分区。

8.根据权利要求 5 所述的方法，其特征在于，所述将所述虚拟动态分区的数据落盘到所述动态分区之后，触发所述第一重启。

9.根据权利要求 5 所述的方法，其特征在于，在所述将所述第三制式文件的制式内容写入所述第二制式文件的过程中，

所述第三重启之后所述电子设备从所述第二静态分区启动操作系统，其中，在所述第三重启之后的初始化环节判断是否需要恢复出厂设置，当确认不需要恢复出厂设置时，继续后续的操作系统启动操作。

10.根据权利要求 9 所述的方法，其特征在于，所述获取操作系统升级包之后，所述方法还包括，确认所述操作系统升级包是否用于改写制式，当所述操作系统升级包用于改写制式时，标记制式改写标志位为第一值；

所述在所述第三重启之后的初始化环节判断是否需要恢复出厂设置，包括：当落盘状态为未落盘，并且，所述制式改写标志位为所述第一值时，确认不需要恢复所述电子设备的出厂设置；

所述将所述虚拟动态分区的数据落盘到所述动态分区之后，标记所述制式改写标志位为第二值。

11.根据权利要求 5 所述的方法，其特征在于，所述将所述虚拟动态分区的数据落盘到所述动态分区之后，所述触发所述电子设备的第一重启之前，所述方法还包括：

触发所述电子设备的第四重启，所述第四重启之后所述电子设备从所述第二静态分区启动操作系统，其中，在所述第四重启之后的初始化环节判断是否需要恢复出厂设置，当确认需要恢复出厂设置时，中断操作系统启动操作，触发所述第一重启。

12.根据权利要求 11 所述的方法，其特征在于，所述在所述第四重启之后的初始化环节判断是否需要恢复出厂设置，包括：

确认所述用户数据分区中保存的制式配置文件与所述第二制式文件是否匹配；

如果不匹配，确认需要恢复所述电子设备的出厂设置。

13.根据权利要求 11 所述的方法，其特征在于，所述获取操作系统升级包之后，所述方法还包括，确认所述操作系统升级包是否用于改写制式，当所述操作系统升级包用于改写制式时，标记制式改写标志位为第一值；

所述将所述虚拟动态分区的数据落盘到所述动态分区之后，标记所述制式改写标志位为第二值；

所述在所述第四重启之后的初始化环节判断是否需要恢复出厂设置，包括：当落盘状态为已落盘，或者，所述制式改写标志位为所述第二值时，确认所述用户数据分区中保存的制式配置文件与所述第二制式文件是否匹配；

如果匹配，确认不需要恢复所述电子设备的出厂设置；

如果不匹配，确认需要恢复所述电子设备的出厂设置。

14.根据权利要求 1 所述的方法，其特征在于，所述操作系统包括升级包获取工具以及升级引擎，第一操作系统以及第二操作系统为对应不同制式的操作系统，所述第一操作系统对应所述第一制式，所述第二操作系统对应所述第二制式；所述获取操作系统升级包之前，所述电子设备启动后加载所述基础分区、所述第一静态分区以及动态分区的数据以从所述第一静态分区启动所述第一操作系统；在所述第一操作系统启动后，所述方法包括：

所述升级包获取工具获取所述操作系统升级包，所述操作系统升级包包括所述第三制式文件、静态分区升级数据以及动态分区升级数据；

所述升级包获取工具触发所述升级引擎进入升级流程；

所述升级引擎基于所述静态分区升级数据更新所述第二静态分区的数据；

所述升级引擎在所述用户数据分区中创建虚拟动态分区，将所述动态分区升级数据写入到所述虚拟动态分区，并且，标记落盘状态为未落盘；

所述升级引擎修改所述电子设备的启动顺序为从所述第二静态分区启动；

所述升级引擎确认所述操作系统升级包是否用于改写制式；

当所述操作系统升级包用于改写制式时，所述升级引擎将所述第三制式文件的制式内容写入所述第二制式文件，并且，标记制式改写标志位为第一值；

所述升级引擎向所述升级包获取工具返回升级操作完成的状态信息；

所述升级包获取工具触发所述电子设备的第三重启，所述第三重启之后所述电子设备加载所述基础分区、所述第二静态分区、所述动态分区以及所述虚拟动态分区的数据以启动所述第二操作系统；

在所述第三重启之后启动所述第二操作系统的初始化环节中：在加载所述第二制式文件之后，当所述落盘状态为未落盘，并且，所述制式改写标志位为所述第一值时，确认不需要恢复所述电子设备的出厂设置，继续进行操作系统启动操作；

在所述第三重启之后启动所述第二操作系统之后，所述升级包获取工具触发所述升级引擎进入落盘流程；

在所述落盘流程中，所述升级引擎将所述虚拟动态分区的数据落盘到所述动态分区并标记所述落盘状态为已落盘；将所述第二静态分区的数据同步到所述第一静态分区；标记所述制式改写标志位为第二值；

所述升级引擎触发所述电子设备的第四重启，所述第四重启之后所述电子设备加载所述基础分区、所述第二静态分区、所述动态分区的数据以启动所述第二操作系统；

在所述第四重启之后启动所述第二操作系统的初始化环节中：在加载所述第二制式文件后，当所述落盘状态为已落盘，或者，所述制式改写标志位为所述第二值时，确认所述用户数据分区中保存的制式配置文件与所述第二制式文件是否匹配；当不匹配时，确认需要恢复所述电子设备的出厂设置，中断操作系统启动操作，触发所述第一重启；

所述第一重启之后所述电子设备进入恢复模式；在所述恢复模式下，恢复所述电子设备的出厂设置；触发所述第二重启，所述第二重启之后所述电子设备从所述第二静态分区启动操作系统，在所述第二重启之后的初始化环节中：在所述用户数据分区中创建所述制式配置文件，将所述第二制式文件的制式内容写入所述制式配置文件。

15.一种电子设备，其特征在于，所述电子设备包括处理器以及存储器，所述存储器包括基础分区、第一静态分区、第二静态分区、动态分区以及用户数据分区，所述电子设备启动操作系统的过程包括初始化环节，所述基础分区中保存有第一制式文件以及第二制式文件，所述电子设备从所述第一静态分区启动操作系统时，所述初始化环节中加载所述第一制式文件；所述电子设备从所述第二静态分区启动操作系统时，所述初始化环节中加载所述第二制式文件；所述第一制式文件当前的制式内容为第一制式；所述处理器用于执行所述存储器上存储的软件代码，以使得所述电子设备启动后加载所述基础分区、所述第一静态分区以及动态分区的数据以从所述第一静态分区启动操作系统；

并且，在所述操作系统启动之后，使得所述电子设备执行如权利要求 1~19 中任一项所述的方法流程。

16、一种计算机可读存储介质，其特征在于，所述计算机可读存储介质中存储有计算机程序，当其在计算机上运行时，使得计算机执行如权利要求 1~14 中任一项所述的方法。

17、一种计算机程序产品，其特征在于，所述计算机程序产品包括计算机程序，当其在计算机上运行时，使得计算机执行如权利要求 1~14 中任一项所述的方法。

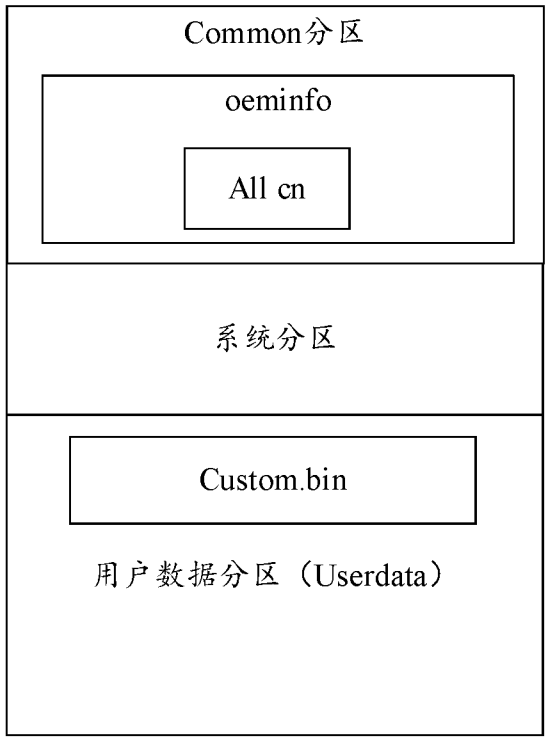

Common分区
oeminfo
All cn
系统分区
Custom.bin
用户数据分区（Userdata）

图 1

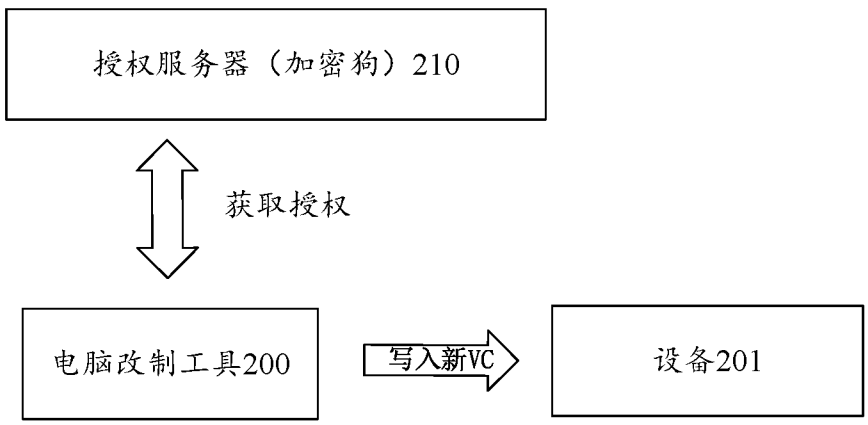

授权服务器（加密狗）210
获取授权
电脑改制工具200
写入新VC
设备201

图 2a

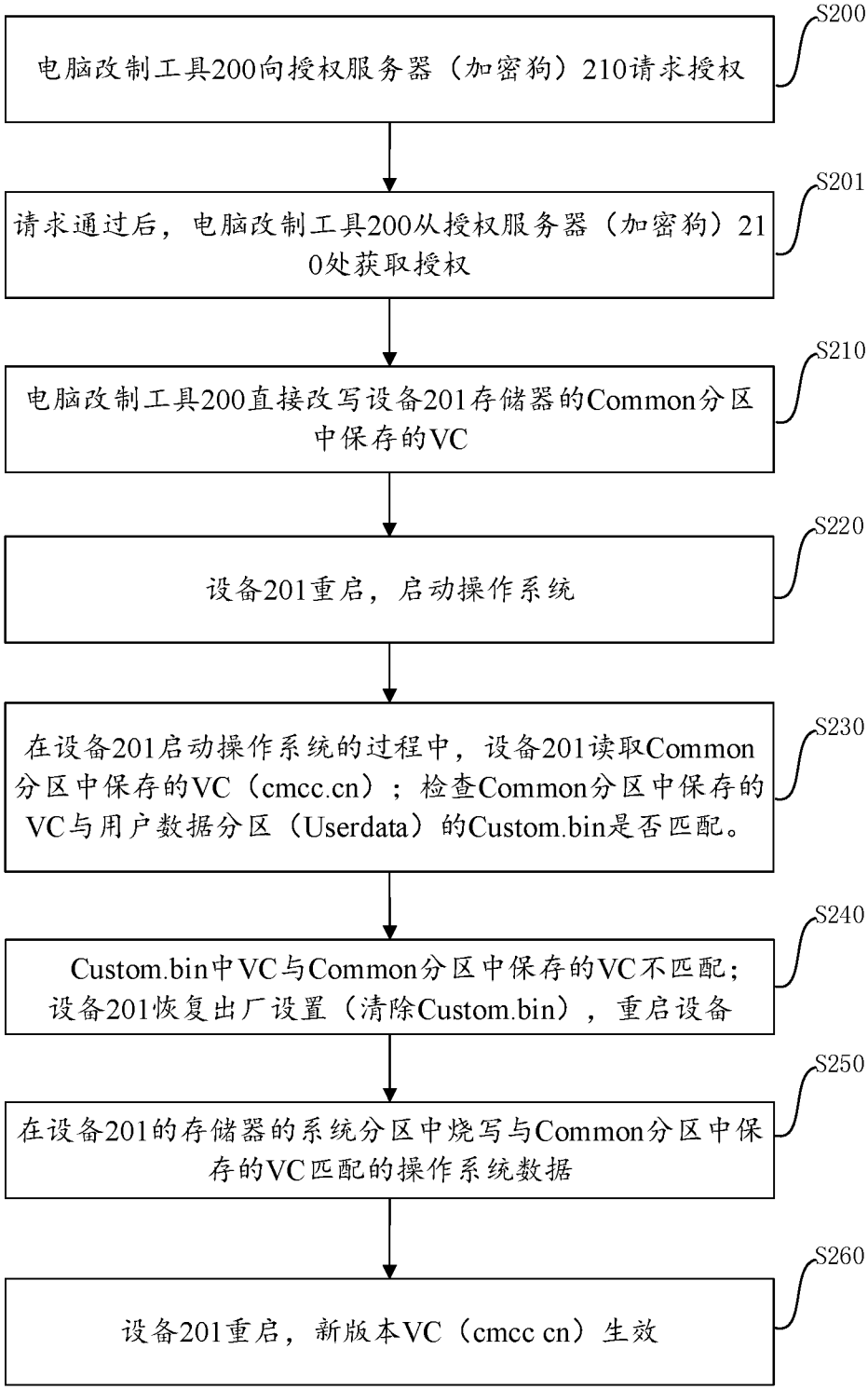

S200
电脑改制工具200向授权服务器（加密狗）210请求授权
S201
请求通过后，电脑改制工具200从授权服务器（加密狗）210处获取授权
S210
电脑改制工具200直接改写设备201存储器的Common分区中保存的VC
S220
设备201重启，启动操作系统
S230
在设备201启动操作系统的过程中，设备201读取Common分区中保存的VC（cmcc.cn）；检查Common分区中保存的VC与用户数据分区（Userdata）的Custom.bin是否匹配。
S240
Custom.bin中VC与Common分区中保存的VC不匹配；设备201恢复出厂设置（清除Custom.bin），重启设备
S250
在设备201的存储器的系统分区中烧写与Common分区中保存的VC匹配的操作系统数据
S260
设备201重启，新版本VC（cmcc cn）生效

图 2b

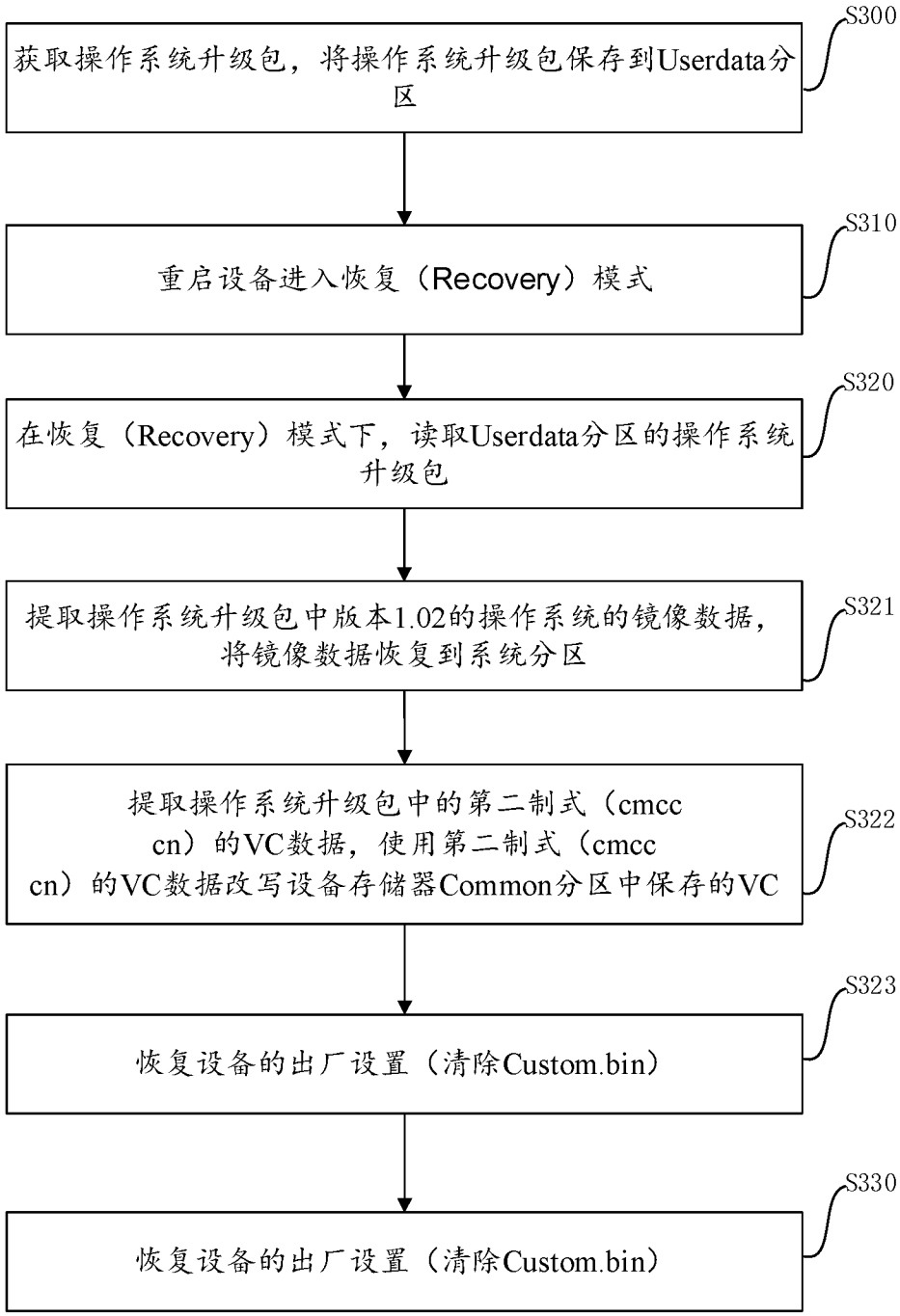

S300
获取操作系统升级包，将操作系统升级包保存到Userdata分区
S310
重启设备进入恢复（Recovery）模式
S320
在恢复（Recovery）模式下，读取Userdata分区的操作系统升级包
S321
提取操作系统升级包中版本1.02的操作系统的镜像数据，将镜像数据恢复到系统分区
S322
提取操作系统升级包中的第二制式（cmcc cn）的VC数据，使用第二制式（cmcc cn）的VC数据改写设备存储器Common分区中保存的VC
S323
恢复设备的出厂设置（清除Custom.bin）
S330
恢复设备的出厂设置（清除Custom.bin）

图 3

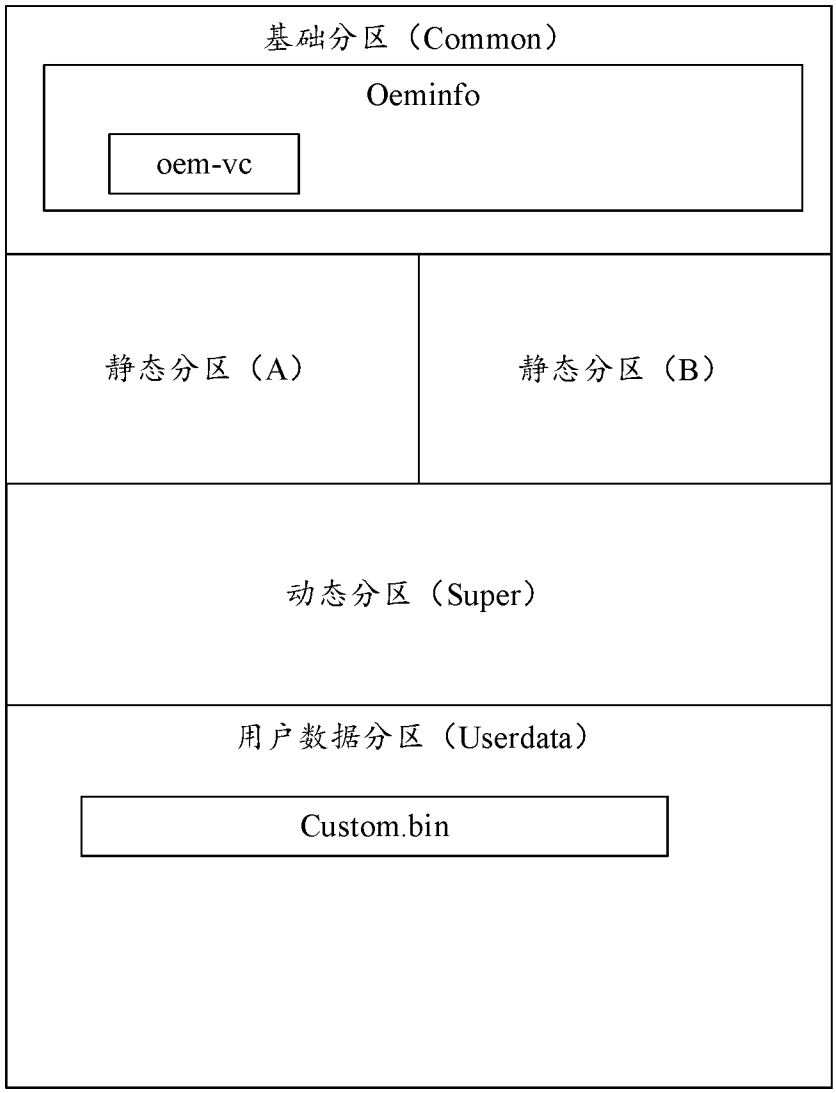

基础分区（Common）
Oeminfo
oem-vc
静态分区（A）
静态分区（B）
动态分区（Super）
用户数据分区（Userdata）
Custom.bin

图 4

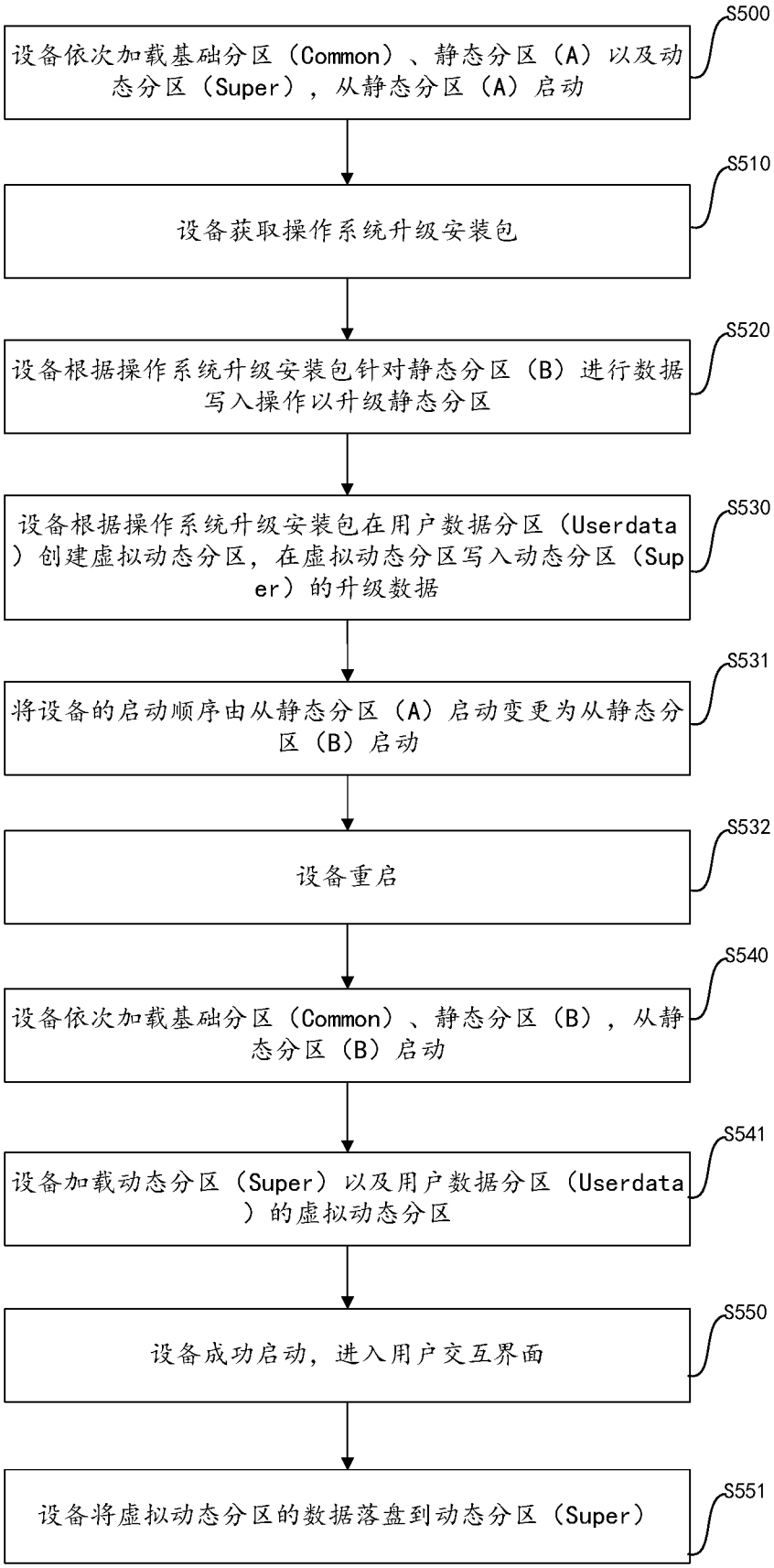

S500
设备依次加载基础分区（Common）、静态分区（A）以及动态分区（Super），从静态分区（A）启动
S510
设备获取操作系统升级安装包
S520
设备根据操作系统升级安装包针对静态分区（B）进行数据写入操作以升级静态分区
S530
设备根据操作系统升级安装包在用户数据分区（Userdata）创建虚拟动态分区，在虚拟动态分区写入动态分区（Super）的升级数据
S531
将设备的启动顺序由从静态分区（A）启动变更为从静态分区（B）启动
S532
设备重启
S540
设备依次加载基础分区（Common）、静态分区（B），从静态分区（B）启动
S541
设备加载动态分区（Super）以及用户数据分区（Userdata）的虚拟动态分区
S550
设备成功启动，进入用户交互界面
S551
设备将虚拟动态分区的数据落盘到动态分区（Super）

图 5a

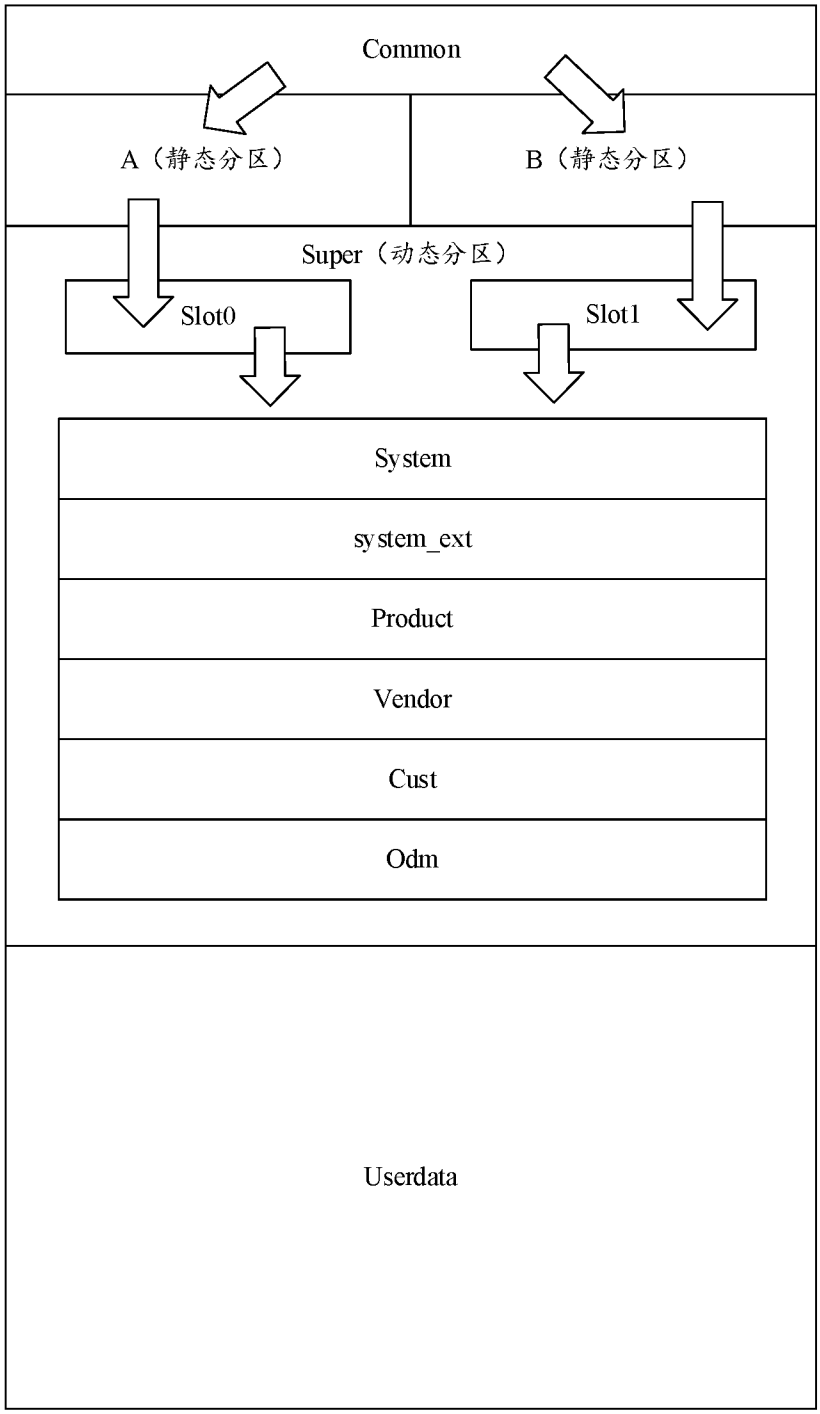

Common
A（静态分区）
B（静态分区）
Super（动态分区）
Slot0
Slot1
System
system_ext
Product
Vendor
Cust
Odm
Userdata

图 5b


基础分区（Common）
Oeminfo
oemvc_a
oemvc_b
静态分区（A）
静态分区（B）
动态分区（Super）
用户数据分区（Userdata）
Custom.bin

图 6a

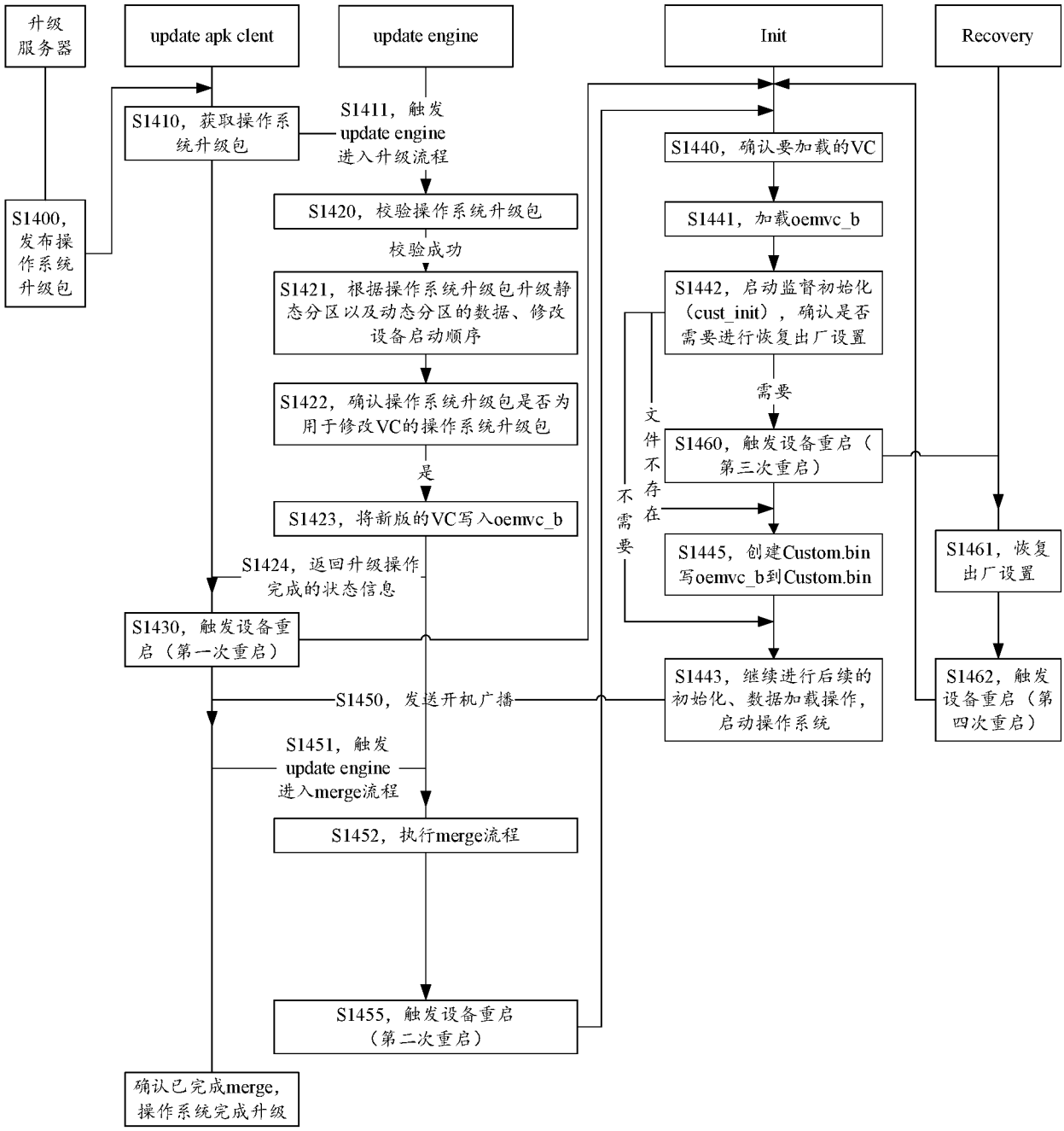

升级服务器
update apk clent
update engine
Init
Recovery
S1400，发布操作系统升级包
S1410，获取操作系统升级包
S1411，触发update engine进入升级流程
S1420，校验操作系统升级包
校验成功
S1421，根据操作系统升级包升级静态分区以及动态分区的数据、修改设备启动顺序
S1422，确认操作系统升级包是否为用于修改VC的操作系统升级包
是
S1423，将新版的VC写入oemvc_b
S1424，返回升级操作完成的状态信息
S1430，触发设备重启（第一次重启）
S1440，确认要加载的VC
S1441，加载oemvc_b
S1442，启动监督初始化（cust_init），确认是否需要进行恢复出厂设置
需要
文件不存在
不需要
S1460，触发设备重启（第三次重启）
S1445，创建Custom.bin写oemvc_b到Custom.bin
S1461，恢复出厂设置
S1450，发送开机广播
S1443，继续进行后续的初始化、数据加载操作，启动操作系统
S1462，触发设备重启（第四次重启）
S1451，触发update engine进入merge流程
S1452，执行merge流程
S1455，触发设备重启（第二次重启）
确认已完成merge，操作系统完成升级

图 6b

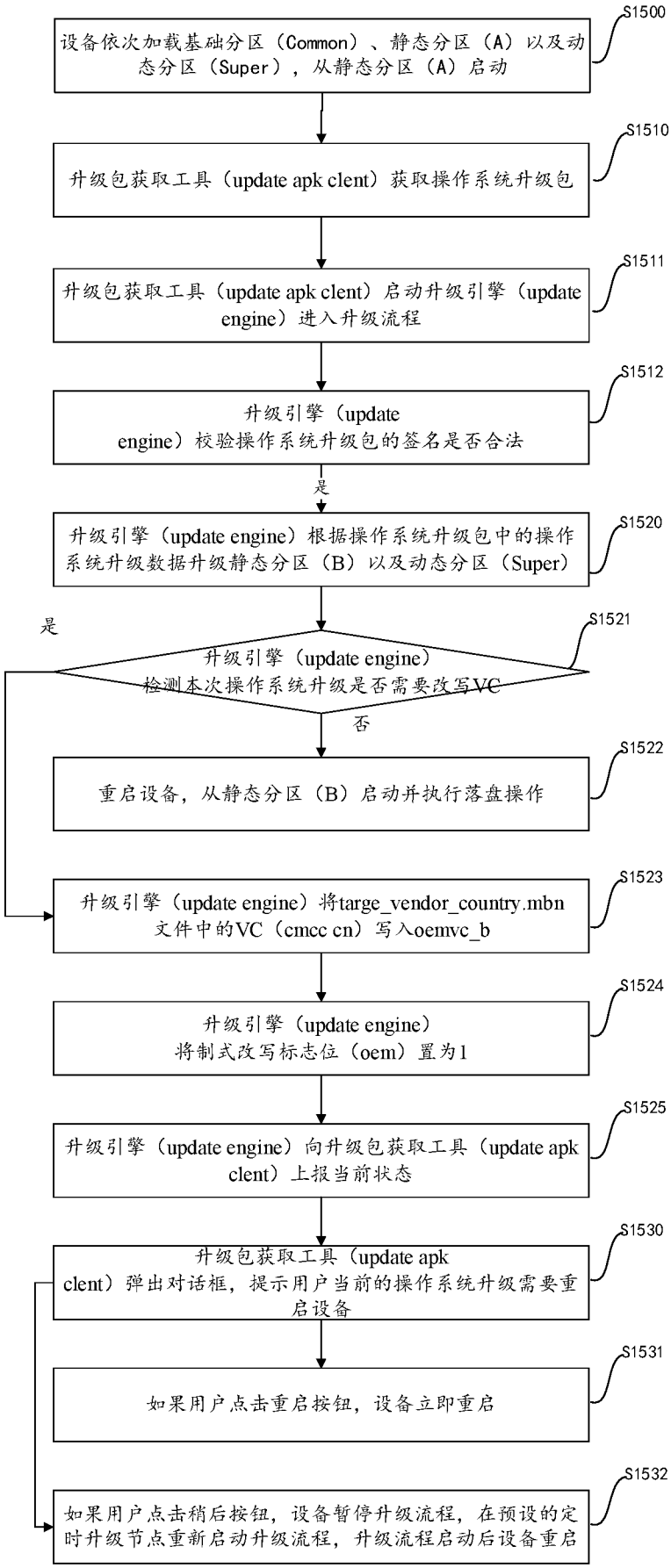

S1500
设备依次加载基础分区（Common）、静态分区（A）以及动态分区（Super），从静态分区（A）启动
S1510
升级包获取工具（update apk clent）获取操作系统升级包
S1511
升级包获取工具（update apk clent）启动升级引擎（update engine）进入升级流程
S1512
升级引擎（update engine）校验操作系统升级包的签名是否合法
是
S1520
升级引擎（update engine）根据操作系统升级包中的操作系统升级数据升级静态分区（B）以及动态分区（Super）
是
S1521
升级引擎（update engine）检测本次操作系统升级是否需要改写VC
否
S1522
重启设备，从静态分区（B）启动并执行落盘操作
S1523
升级引擎（update engine）将targe_vendor_country.mbn文件中的VC（cmcc cn）写入oemvc_b
S1524
升级引擎（update engine）将制式改写标志位（oem）置为1
S1525
升级引擎（update engine）向升级包获取工具（update apk clent）上报当前状态
S1530
升级包获取工具（update apk clent）弹出对话框，提示用户当前的操作系统升级需要重启设备
S1531
如果用户点击重启按钮，设备立即重启
S1532
如果用户点击稍后按钮，设备暂停升级流程，在预设的定时升级节点重新启动升级流程，升级流程启动后设备重启

图 7

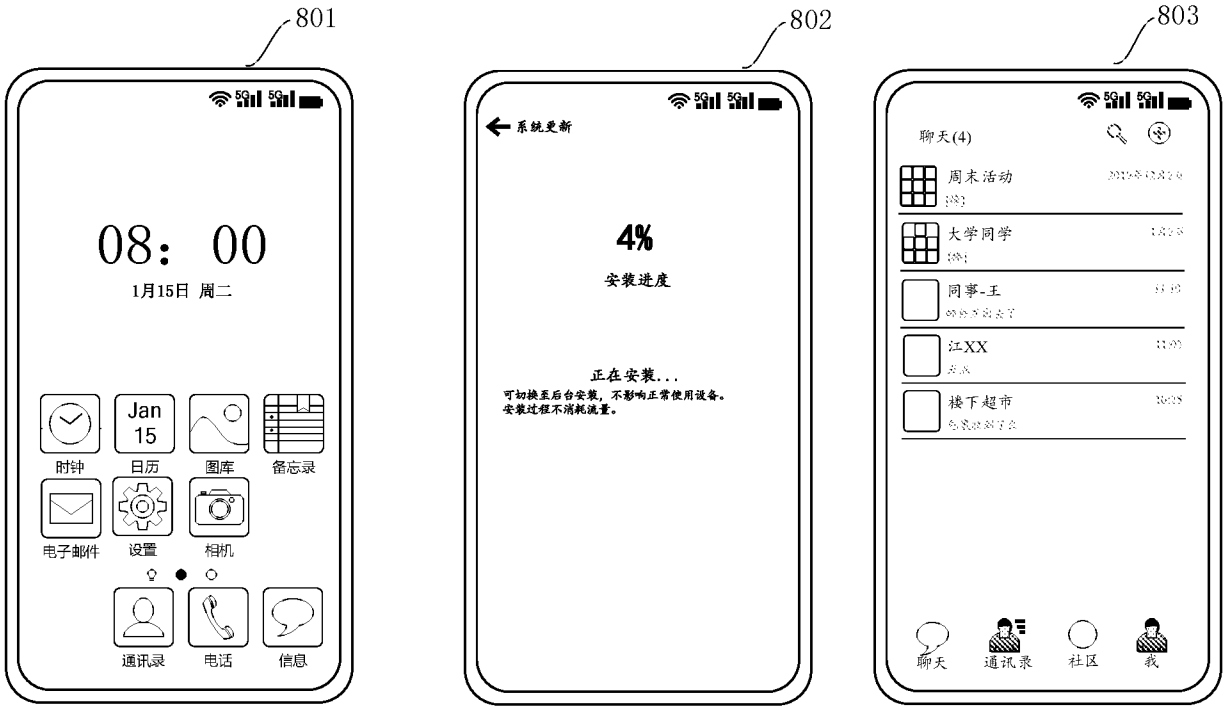

801
08：00
1月15日 周二
时钟
日历
图库
备忘录
电子邮件
设置
相机
通讯录
电话
信息
802
系统更新
4%
安装进度
正在安装...
可切换至后台安装，不影响正常使用设备。
安装过程不消耗流量。
803
聊天(4)
周末活动
大学同学
同事-王
江XX
楼下超市
聊天
通讯录
社区
我

图 8

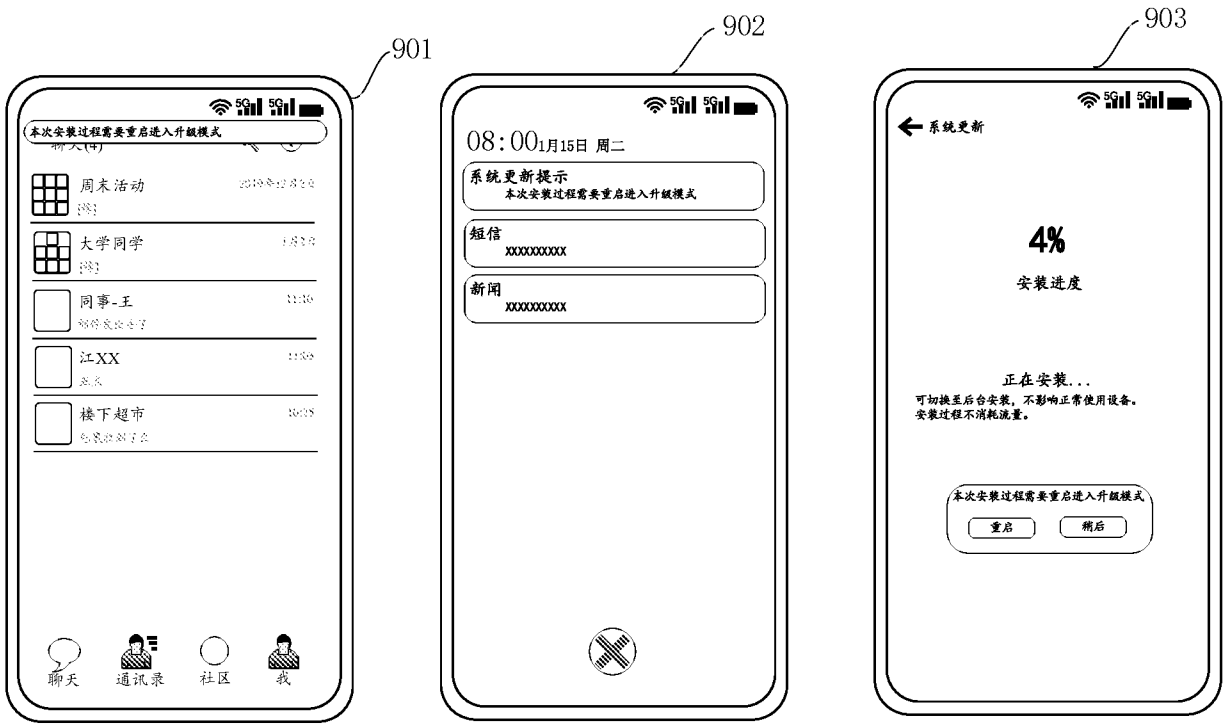

901
本次安装过程需要重启进入升级模式
周末活动
大学同学
同事-王
江XX
楼下超市
聊天
通讯录
社区
我
902
08:00 1月15日 周二
系统更新提示
本次安装过程需要重启进入升级模式
短信
XXXXXXXXX
新闻
XXXXXXXXX
903
系统更新
4%
安装进度
正在安装...
可切换至后台安装，不影响正常使用设备。
安装过程不消耗流量。
本次安装过程需要重启进入升级模式
重启
稍后

图 9

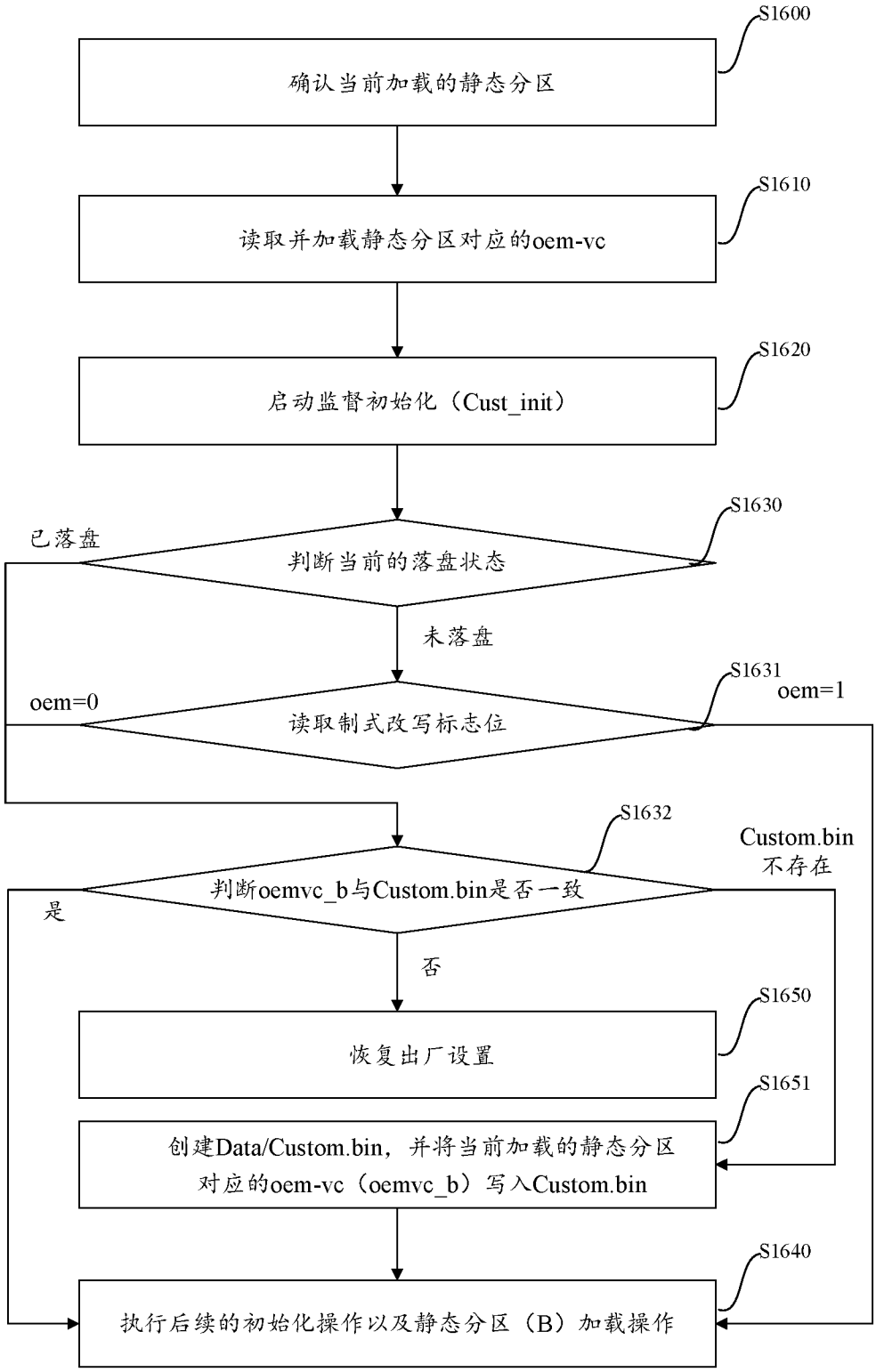

S1600
确认当前加载的静态分区
S1610
读取并加载静态分区对应的oem-vc
S1620
启动监督初始化（Cust_init）
S1630
已落盘
判断当前的落盘状态
未落盘
S1631
oem=0
oem=1
读取制式改写标志位
S1632
Custom.bin
不存在
判断oemvc_b与Custom.bin是否一致
是
否
S1650
恢复出厂设置
S1651
创建Data/Custom.bin，并将当前加载的静态分区
对应的oem-vc（oemvc_b）写入Custom.bin
S1640
执行后续的初始化操作以及静态分区（B）加载操作

图 10

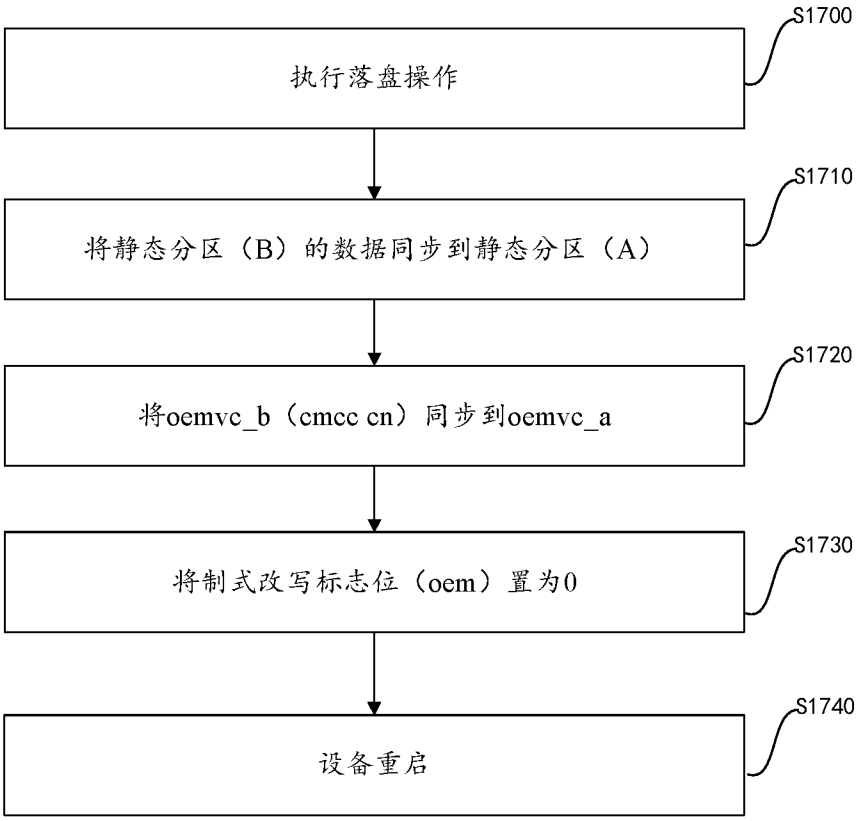

S1700
执行落盘操作
S1710
将静态分区（B）的数据同步到静态分区（A）
S1720
将oemvc_b（cmcc cn）同步到oemvc_a
S1730
将制式改写标志位（oem）置为0
S1740
设备重启

图 11a

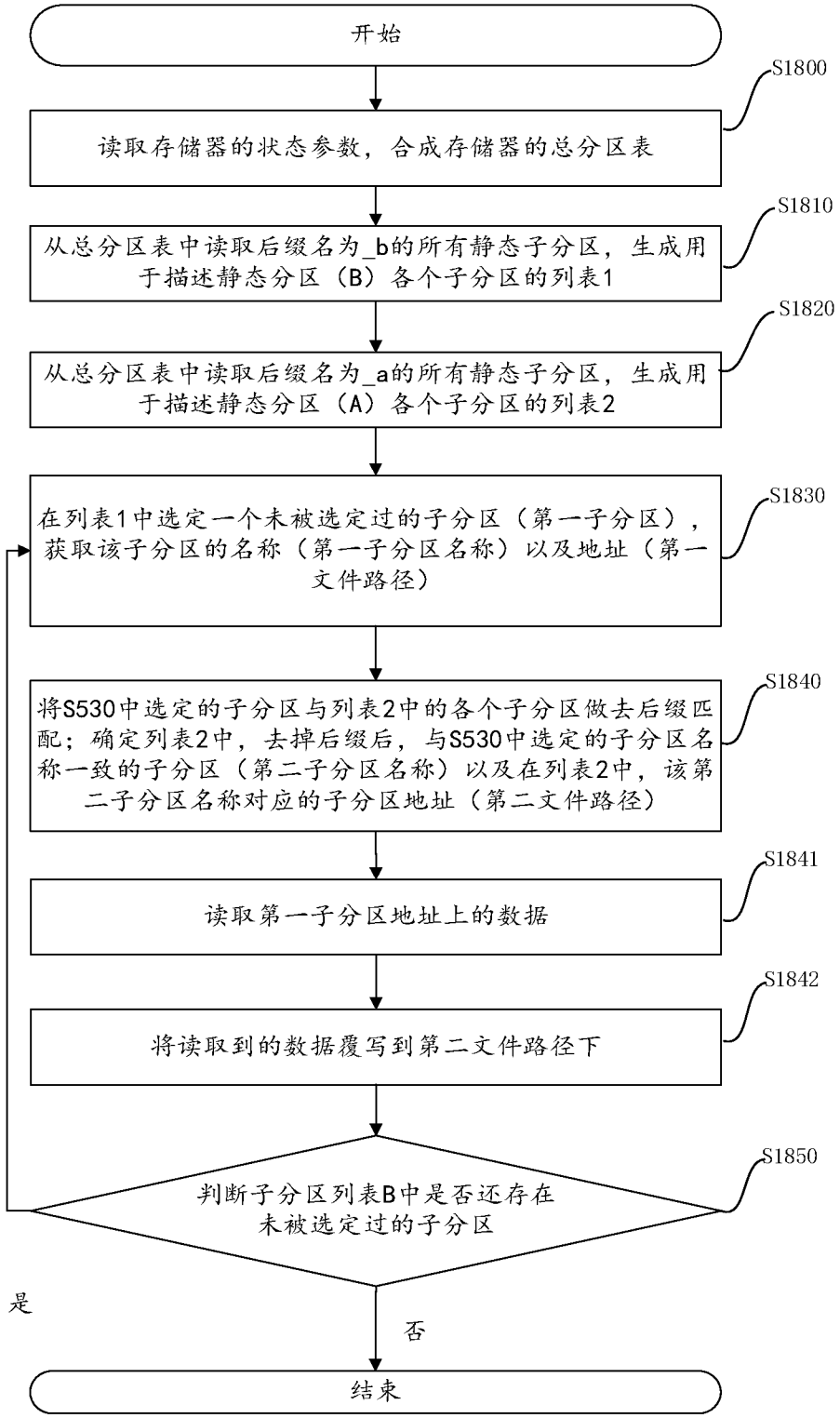

开始
S1800
读取存储器的状态参数，合成存储器的总分区表
S1810
从总分区表中读取后缀名为_b的所有静态子分区，生成用于描述静态分区（B）各个子分区的列表1
S1820
从总分区表中读取后缀名为_a的所有静态子分区，生成用于描述静态分区（A）各个子分区的列表2
S1830
在列表1中选定一个未被选定过的子分区（第一子分区），获取该子分区的名称（第一子分区名称）以及地址（第一文件路径）
S1840
将S530中选定的子分区与列表2中的各个子分区做去后缀匹配；确定列表2中，去掉后缀后，与S530中选定的子分区名称一致的子分区（第二子分区名称）以及在列表2中，该第二子分区名称对应的子分区地址（第二文件路径）
S1841
读取第一子分区地址上的数据
S1842
将读取到的数据覆写到第二文件路径下
S1850
判断子分区列表B中是否还存在未被选定过的子分区
是
否
结束

图 11b

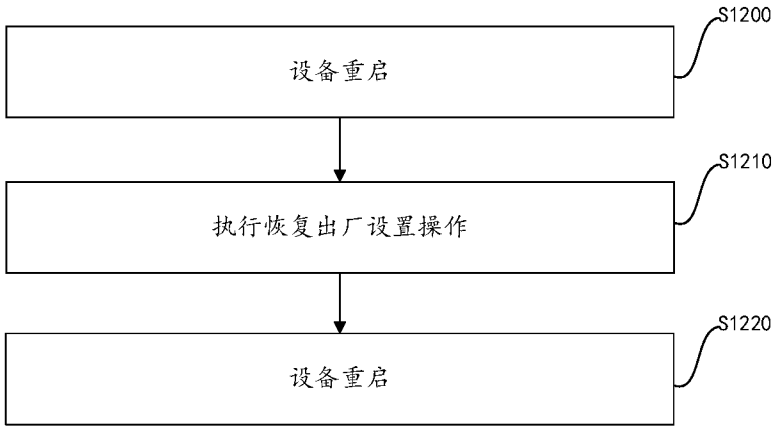


图 12

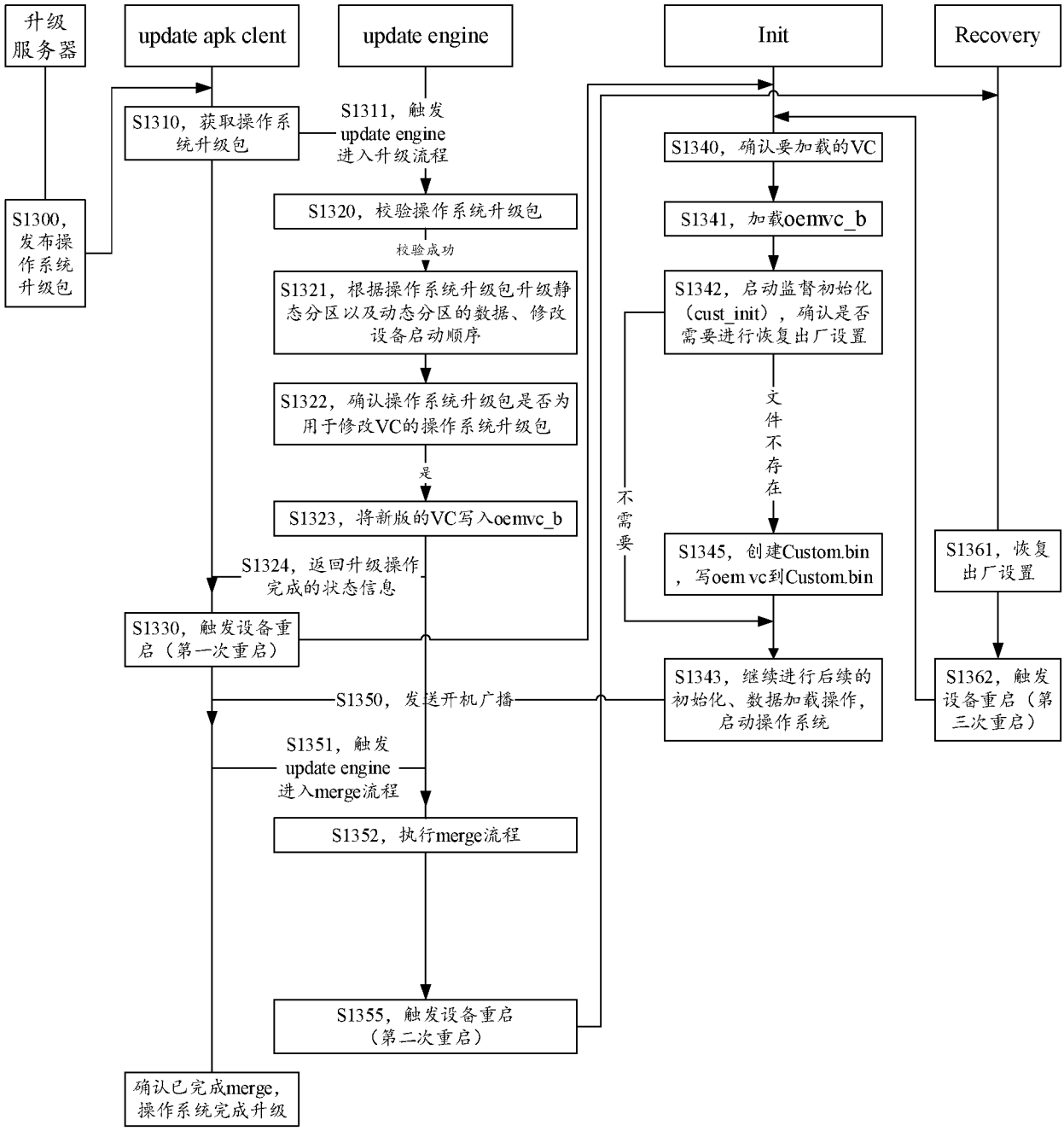

升级服务器
update apk clent
update engine
Init
Recovery
S1310，获取操作系统升级包
S1311，触发update engine进入升级流程
S1340，确认要加载的VC
S1300，发布操作系统升级包
S1320，校验操作系统升级包
S1341，加载oemvc_b
校验成功
S1321，根据操作系统升级包升级静态分区以及动态分区的数据、修改设备启动顺序
S1342，启动监督初始化（cust_init），确认是否需要进行恢复出厂设置
S1322，确认操作系统升级包是否为用于修改VC的操作系统升级包
文件不存在
是
S1323，将新版的VC写入oemvc_b
不需要
S1345，创建Custom.bin，写oem vc到Custom.bin
S1361，恢复出厂设置
S1324，返回升级操作完成的状态信息
S1330，触发设备重启（第一次重启）
S1343，继续进行后续的初始化、数据加载操作，启动操作系统
S1362，触发设备重启（第三次重启）
S1350，发送开机广播
S1351，触发update engine进入merge流程
S1352，执行merge流程
S1355，触发设备重启（第二次重启）
确认已完成merge，操作系统完成升级

图 13

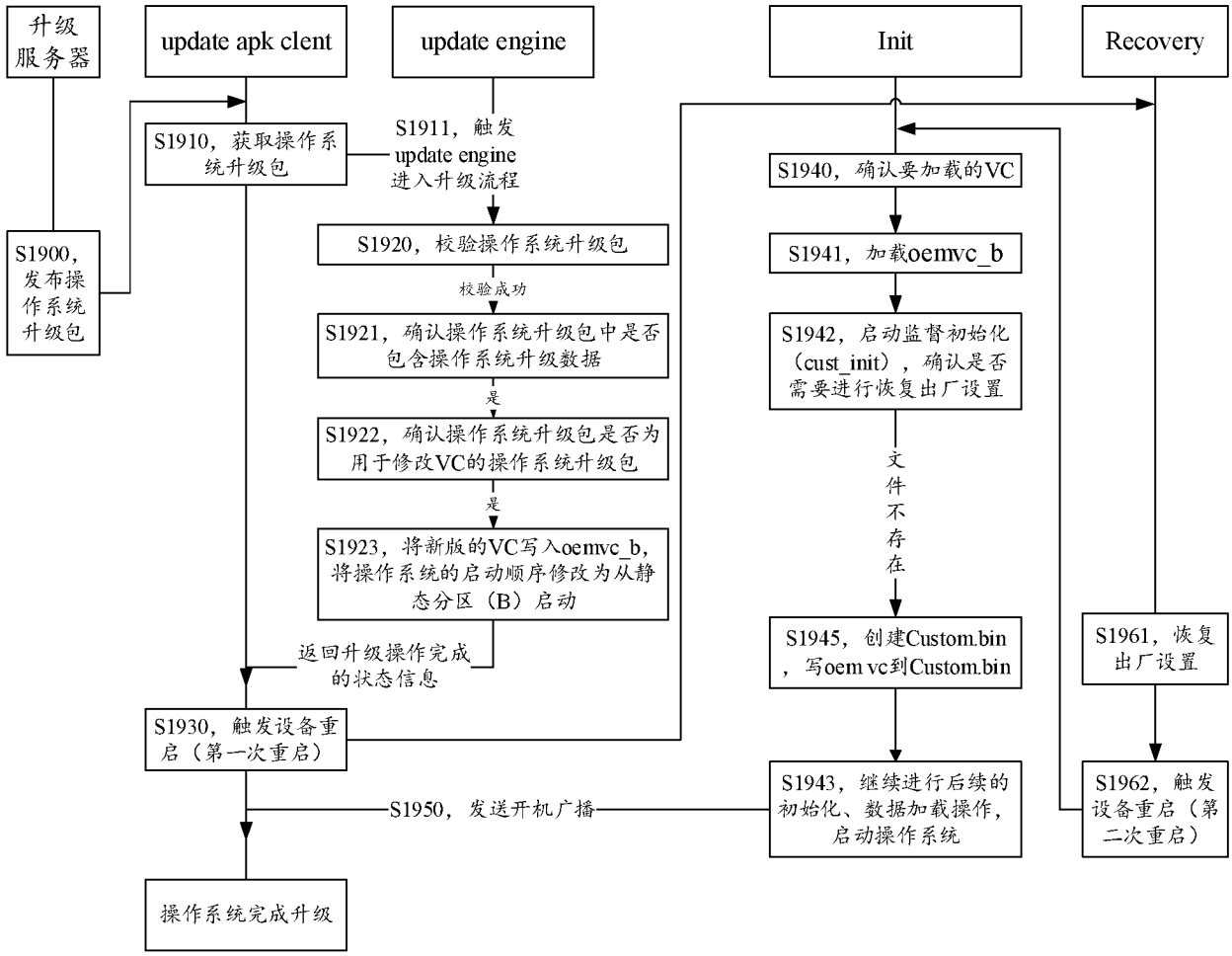

升级服务器
update apk clent
update engine
Init
Recovery
S1900，发布操作系统升级包
S1910，获取操作系统升级包
S1911，触发update engine进入升级流程
S1920，校验操作系统升级包
校验成功
S1921，确认操作系统升级包中是否包含操作系统升级数据
是
S1922，确认操作系统升级包是否为用于修改VC的操作系统升级包
是
S1923，将新版的VC写入oemvc_b，将操作系统的启动顺序修改为从静态分区（B）启动
返回升级操作完成的状态信息
S1930，触发设备重启（第一次重启）
S1940，确认要加载的VC
S1941，加载oemvc_b
S1942，启动监督初始化（cust_init），确认是否需要进行恢复出厂设置
文件不存在
S1945，创建Custom.bin，写oem vc到Custom.bin
S1943，继续进行后续的初始化、数据加载操作，启动操作系统
S1950，发送开机广播
S1961，恢复出厂设置
S1962，触发设备重启（第二次重启）
操作系统完成升级

图 14

INTERNATIONAL SEARCH REPORT

International application No.
PCT/CN2022/094159

A. CLASSIFICATION OF SUBJECT MATTER

G06F 9/4401(2018.01)i; G06F 8/654(2018.01)n

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)

G06F

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

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

CNABS; CNTXT; VEN; USTXT; EPTXT; WOTXT; CNKI: 操作系统, 分区, 静态, 动态, 制式, 格式, 启动, 重启, 加载, 顺序, 次序, 恢复模式, 配置, 初始化, OS, subarea, static, dynamic, format, startup, restart, load, order, sequence, recovery, configuration, initialization

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	CN 110543321 A (SHENZHEN YINGBO SUPERCOMPUTING TECHNOLOGY CO., LTD.) 06 December 2019 (2019-12-06) description, paragraphs [0036]-[0084]	1-17
A	CN 107786729 A (VIVO COMMUNICATION TECHNOLOGY CO., LTD.) 09 March 2018 (2018-03-09) entire document	1-17
A	CN 103729210 A (GUANGDONG OPPO MOBILE TELECOMMUNICATIONS CO., LTD.) 16 April 2014 (2014-04-16) entire document	1-17
A	US 2018314832 A1 (TOSHIBA KK et al.) 01 November 2018 (2018-11-01) entire document	1-17

☐ 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
21 July 2022	**29 July 2022**
Name and mailing address of the ISA/CN **China National Intellectual Property Administration (ISA/CN)** **No. 6, Xitucheng Road, Jimenqiao, Haidian District, Beijing 100088, China** Facsimile No. **(86-10)62019451**	Authorized officer Telephone No.

INTERNATIONAL SEARCH REPORT
Information on patent family members

International application No.
PCT/CN2022/094159

Patent document cited in search report	Publication date (day/month/year)	Patent family member(s)	Publication date (day/month/year)
CN 110543321 A	06 December 2019	None	
CN 107786729 A	09 March 2018	None	
CN 103729210 A	16 April 2014	CN 103729210 B	01 December 2017
US 2018314832 A1	01 November 2018	EP 3399408 A1	07 November 2018
		JP 2018190143 A	29 November 2018
		US 10628588 B2	21 April 2020
		JP 6873811 B2	19 May 2021

国际检索报告

国际申请号
PCT/CN2022/094159

A. 主题的分类

G06F 9/4401(2018.01)i; G06F 8/654(2018.01)n

按照国际专利分类(IPC)或者同时按照国家分类和IPC两种分类

B. 检索领域

检索的最低限度文献(标明分类系统和分类号)

G06F

包含在检索领域中的除最低限度文献以外的检索文献

在国际检索时查阅的电子数据库(数据库的名称，和使用的检索词（如使用））

CNABS;CNTXT;VEN;USTXT;EPTXT;WOTXT;CNKI：操作系统，分区，静态，动态，制式，格式，启动，重启，加载，顺序，次序，恢复模式，配置，初始化，OS，subarea，static，dynamic，format，startup，restart，load，order，sequence，recovery，configuration，initialization

C. 相关文件

类 型*	引用文件，必要时，指明相关段落	相关的权利要求
A	CN 110543321 A (深圳市英博超算科技有限公司) 2019年12月6日 (2019-12-06) 说明书第[0036]-[0084]段	1-17
A	CN 107786729 A (维沃移动通信有限公司) 2018年3月9日 (2018-03-09) 全文	1-17
A	CN 103729210 A (广东欧珀移动通信有限公司) 2014年4月16日 (2014-04-16) 全文	1-17
A	US 2018314832 A1 (TOSHIBA KK 等) 2018年11月1日 (2018-11-01) 全文	1-17

☐ 其余文件在C栏的续页中列出。

☑ 见同族专利附件。

* 引用文件的具体类型：
“A” 认为不特别相关的表示了现有技术一般状态的文件
“E” 在国际申请日的当天或之后公布的在先申请或专利
“L” 可能对优先权要求构成怀疑的文件，或为确定另一篇引用文件的公布日而引用的或者因其他特殊理由而引用的文件（如具体说明的）
“O” 涉及口头公开、使用、展览或其他方式公开的文件
“P” 公布日先于国际申请日但迟于所要求的优先权日的文件
“T” 在申请日或优先权日之后公布，与申请不相抵触，但为了理解发明之理论或原理的在后文件
“X” 特别相关的文件，单独考虑该文件，认定要求保护的发明不是新颖的或不具有创造性
“Y” 特别相关的文件，当该文件与另一篇或者多篇该类文件结合并且这种结合对于本领域技术人员为显而易见时，要求保护的发明不具有创造性
“&” 同族专利的文件

国际检索实际完成的日期	国际检索报告邮寄日期
2022年7月21日	2022年7月29日
ISA/CN的名称和邮寄地址 中国国家知识产权局(ISA/CN) 中国北京市海淀区蓟门桥西土城路6号 100088 传真号 (86-10)62019451	受权官员 张永辉 电话号码 （86-512）88995934

国际检索报告
关于同族专利的信息

国际申请号
PCT/CN2022/094159

检索报告引用的专利文件			公布日（年/月/日）	同族专利			公布日（年/月/日）
CN	110543321	A	2019年12月6日	无			
CN	107786729	A	2018年3月9日	无			
CN	103729210	A	2014年4月16日	CN	103729210	B	2017年12月1日
US	2018314832	A1	2018年11月1日	EP	3399408	A1	2018年11月7日
				JP	2018190143	A	2018年11月29日
				US	10628588	B2	2020年4月21日
				JP	6873811	B2	2021年5月19日

S1400 Release an operating system upgrade package
S1410 Acquire the operating system upgrade package
S1411 Trigger the update engine to enter an upgrade process
S1420 Verify the operating system upgrade package
S1421 Upgrade data of a static partition and a dynamic partition according to the operating system upgrade package, and modify a device start-up sequence
S1422 Determine whether the operating system upgrade package is an operating system upgrade package for modifying a VC
S1423 Write a new-version VC into oemvc_b
S1424 Return state information of an upgrade operation being completed
S1430 Trigger a device to restart (first restart)
S1440 Determine a VC to be loaded
S1441 Load the oemvc_b
S1442 Enable monitoring initialization (cust_init), and determine whether the recovery of factory settings is required
S1443 Continue to perform subsequent initialization and data loading operations, and start-up the operating system
S1445 Create Custom.bin, and write the oemvc_b into the Custom.bin
S1450 Send a boot-up broadcast
S1451 Trigger the update engine to enter a merge process
S1452 Execute the merge process
S1455 Trigger the device to restart (second restart)
S1460 Trigger the device to restart (third restart)
S1461 Recover the factory settings
S1462 Trigger the device to restart (fourth restart)
AA Upgrade server
DD Verification is successful
FF Required
GG No file is found
HH Not required
II Determine that merge is completed, and the upgrading of the operating system is completed
JJ Yes