[19] 中华人民共和国国家知识产权局

[51] Int. Cl.
G06F 7/00 (2006.01)
G06F 17/30 (2006.01)
G06F 9/44 (2006.01)

[12] 发明专利申请公开说明书

[21] 申请号 200380103476.8

[43] 公开日 2006 年 5 月 31 日

[11] 公开号 CN 1781075A

[22] 申请日 2003.11.18
[21] 申请号 200380103476.8
[30] 优先权
[32] 2002.11.18 [33] US [31] 10/298,458
[86] 国际申请 PCT/US2003/037001 2003.11.18
[87] 国际公布 WO2004/046910 英 2004.6.3
[85] 进入国家阶段日期 2005.5.17
[71] 申请人 创道软件有限公司
地址 美国加利福尼亚州
[72] 发明人 彭罗生

[74] 专利代理机构 北京集佳知识产权代理有限公司
代理人 徐 谦 杨红梅

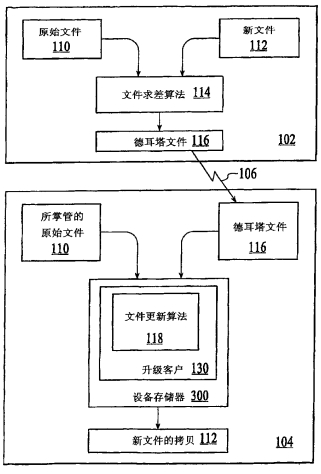
权利要求书 8 页 说明书 34 页 附图 20 页

[54] 发明名称

电子文件更新期间的设备存储器管理

[57] 摘要

在执行存储器管理时，通过执行对第一和第二存储区域的依次搜索，主机设备的升级客户(130)标识并保留足够大以容纳新软件组件的存储块。该新软件组件是原始软件文件(110)的组件的更新版本。当该新组件的大小超过第一和第二存储区域的可用块的大小时，升级客户(130)重写第一存储区域以除去不用的存储器块，重新分配第一和第二存储器区域，将新的组件写入第二存储区域，并更新向量表。为了访问主机设备软件组件，升级客户接收来自主机设备(104)主程序的函数调用，包括对应软件文件的标识信息，从向量表读取对应软件文件的起始地址，并产生对所述对应软件文件的调用。



1．一种用于更新电子文件的系统，其包括：

第一设备，其包括产生德耳塔文件的文件求差和更新系统的第一组件；

第二设备，其从第一设备经由至少一个耦合接收该德耳塔文件，该第二设备包括所述文件求差和更新系统的第二组件，其被配置成

通过以下来更新第二设备的电子文件：

从所述德耳塔文件读取新电子文件的至少一个新组件，该新电子文件是原始电子文件的更新版本；

通过执行对第一和第二存储器区域的依次搜索，标识并保留足够大以容纳该新组件的存储器块，其中第一存储器区域被重写以去除不用的存储器块，并且当新组件的大小超过第一和第二存储器区域的可用存储器块的大小时，第一和第二存储器区域被重新分配；

将新组件写入保留的存储器块；以及

当新组件被写入与包括原始电子文件的对应组件的原始存储器块不同的存储器块时，更新向量表；以及

通过以下来访问该设备的电子文件：

接收来自设备主程序的函数调用，该函数调用包括对应电子文件的标识信息；

从向量表读取对应电子文件的起始地址；以及

使用该起始地址和标识信息来产生用于对应电子文件的调用。

2. 如权利要求 1 所述的系统，其中所述第二设备进一步包括在更新电子文件中使用的第一和第二应用编程界面（API）。

3. 如权利要求 1 所述的系统，其中所述第二设备进一步包括在访问电子文件中使用的第三应用编程界面（API）。

4. 如权利要求 1 所述的系统，其中所述第一设备包括基于处理器的

设备，其可由被掌管于第二设备上的软件的至少一个提供者访问。

5. 如权利要求 1 所述的系统，其中所述第二设备包括至少一个基于处理器的设备，其从个人计算机，便携式计算设备，蜂窝电话，便携式通信设备，和个人数字助理中选择。

6. 如权利要求 1 所述的系统，其中所述至少一个耦合是从以下中选择的：无线耦合，有线耦合，混合无线/有线耦合，和与至少一个网络，包括局域网（LAN），城域网（MAN）和广域网（WAN），专属网，后端网，互联网的耦合，和可移动的固定媒介，包括软盘、硬盘驱动器和光盘只读存储器（CD-ROM），以及电话线，总线和电子邮件消息。

7. 如权利要求 1 所述的系统，其中所述原始和新电子文件包括软件文件，其包括动态链接库文件，共享目标文件，嵌入式软件组件（EBSC），固件文件，可执行文件，包括十六进制数据文件的数据文件，系统配置文件，以及包括个人使用数据的文件。

8. 一种用于主机设备中设备存储器管理的方法，其包括：

接收新电子文件的至少一个被接收组件的标识信息，该新电子文件为原始电子文件的更新版本，其中所述标识信息包括所接收组件的大小；

通过执行对第一和第二存储器区域的依次搜索来标识并保留足够大以容纳所接收的组件的存储器块，其中第一存储器区域被重写以去除不用的存储器块，并且当所述大小超过第一和第二存储器区域的可用存储器块的大小时，第一和第二存储器区域被重新分配；

提供被保留的存储器块的地址；以及

通过将所接收的组件写入所述被保留的存储器块来更新原始电子文件。

9. 如权利要求 8 所述的方法，进一步包括当所接收的组件被写入与包括原始电子文件的对应组件的第一存储器区域中的原始存储器块不同的存储器块时，更新第一表，其中所述第一表包括主机设备电子文件组件的组件信息、该组件信息包括组件标识、只读存储器（ROM）数、起

始地址和大小。

10. 如权利要求 8 所述的方法，其中标识和保留包括当所述大小等于或者小于原始电子文件的对应组件的大小时，保留第一存储器区域的原始存储器块，该原始存储器块包括原始电子文件的对应组件。

11. 如权利要求 10 所述的方法，其中标识和保留包括当所述大小超过原始电子文件的对应组件的大小时，为足够大以存储所接收的组件的存储器块而搜索第二存储器区域和第一存储器区域的不用区段中的至少一个。

12. 如权利要求 8 所述的方法，其中第一存储器区域的重写和第二存储器区域的重新分配进一步包括：

读取第二表，该第二表包括对应于第一区域不用存储器块的至少一个项目；

重写第一存储器区域的组件以依次将第一存储器区域的组件打包并合并不用的存储器块；

依次评估第二存储器区域的每个组件的大小并且在第二存储器区域的组件大小和所合并的不用存储器块的大小允许的情况下将第二存储器区域的组件重写到第一存储器区域；

在依次评估和重写之后，标识所合并的不用存储器块的剩余块；

通过指定所合并的不用存储器块的剩余块作为第二存储器区域的部分来重新分配第二存储器区域；

置位第二存储器区域的指针以维持经重新分配的第二存储器区域的可用存储器的起始地址；以及

更新第二表。

13. 如权利要求 12 所述的方法，其中所述第二表的项目包括不用的存储器块的信息，其包括只读存储器（ROM）数、起始地址和大小。

14. 如权利要求 8 所述的方法，进一步包括通过以下来访问由主机设备的电子文件提供的函数：

接收来自主机设备主程序的函数调用，该函数调用包括对应电子文件的标识信息；

从向量表读取对应电子文件的起始地址；以及

通过使用所述起始地址和标识信息产生用于该对应电子文件的调用。

15．一种用于文件更新期间的设备存储器管理的方法，其包括：

使用所接收组件的标识信息来确定新电子文件的至少一个所接收组件的大小，所述新电子文件是原始电子文件的更新版本；

当该大小等于或者小于原始电子文件的对应组件的大小时，分配第一存储器区域的原始存储器块，以将所接收的组件写入其中，所述原始存储器块包括原始电子文件的对应组件；

当该大小超过原始电子文件对应组件的大小时，为足够大以存储所接收的组件的存储器块而搜索第二存储器区域和第一存储器区域的不用区段中的至少一个；以及

当该大小超过第一和第二存储器区域的可用存储器块的大小时，重写第一存储器区域以去除不用的区段，重新分配第一和第二存储器区域，并分配第二存储器区域的存储器块，以将所接收的组件写入其中。

16．如权利要求15所述的方法，其中第一存储器区域的重新写入进一步包括：

读取表，该表包括对应于第一存储器区域不用区段的至少一个项目；以及

重写第一存储器区域的组件以依次打包第一存储器区域的组件并合并不用的存储器块。

17．如权利要求15所述的方法，其中重新分配第一和第二存储器区域进一步包括：

依次评估第二存储器区域每个组件的大小，并且当第二存储器区域的组件大小和经重写的第一存储器区域的所合并的不用存储器块的大小

允许时，将第二存储器区域的组件写入经重写的第一存储器区域；

在依次评估和写入之后，标识所合并的不用存储器块的剩余块；

通过指定所合并的不用存储器块的剩余块作为第二存储器区域的部分，重新分配第一和第二存储器区域；

置位第二存储器区域的指针以维持经重新分配的第二存储器区域的可用存储器的起始地址；以及

更新该表。

18．一种用于管理电子设备的存储器的方法，其包括：

通过以下来更新该设备的电子文件：

接收新电子文件的至少一个新组件，该新电子文件是原始电子文件的更新版本；

通过执行对第一和第二存储器区域的依次搜索，标识并保留足够大以容纳该新组件的存储器块，其中所述第一存储器区域被重写以除去不用的存储器块，并且当该新组件的大小超过第一和第二存储器区域的可用存储器块的大小时，第一和第二存储器区域被重新分配；

将该新组件写入保留的存储器块；以及

当该新组件被写入与包括原始电子文件的对应组件的原始存储器块不同的存储器块时，更新向量表；以及

通过以下来访问该设备的电子文件：

接收来自设备主程序的函数调用，该函数调用包括对应电子文件的标识信息；

从向量表读取对应电子文件的起始地址；以及

使用该起始地址和标识信息来产生用于对应电子文件的调用。

19．如权利要求18所述的方法，其中所述标识信息包括函数标识和该函数的参量。

20．一种设备，其包括：

装置，其用于接收新电子文件的至少一个新组件，该新电子文件是

原始电子文件的更新版本；

装置，其用于通过执行对第一和第二存储器区域的依次搜索来标识并保留足够大以容纳该新组件的存储器块，其中第一存储器区域被重写以去除不用的存储器块并且当该新组件的大小超过第一和第二存储器区域的可用存储器块的大小时，第一和第二存储器区域被重新分配；

装置，其通过将该新组件写入保留的存储器块来更新原始电子文件；以及

装置，其用于当新组件被写入与包括原始电子文件的对应组件的原始存储器块不同的存储器块时，更新向量表。

21. 如权利要求 20 所述的设备，其中所述设备包括至少一个基于处理器的设备，该设备从个人计算机、便携式计算设备、蜂窝电话、便携式通信设备、和个人数字助理之中选择。

22. 如权利要求 20 所述的设备，其中所述用于标识和保留存储器块的装置是第一应用编程界面（API）。

23. 如权利要求 20 所述的设备，进一步包括：

装置，其用于接收来自设备主程序函数调用，该函数调用包括对应电子文件的标识信息；

装置，其用于从向量表读取对应电子文件的起始地址；以及

装置，其用于使用所述起始地址和标识信息来产生用于对应电子文件的调用。

24. 如权利要求 23 所述的设备，其中所述用于接收函数调用的装置是第二应用编程界面（API）。

25. 一种计算机可读媒介，其包括可执行指令，当在处理系统中被执行时，所述指令通过以下来更新电子文件和文件组件：

接收新电子文件的至少一个被接收的组件的标识信息，该新电子文件是原始电子文件的更新版本，其中所述标识信息包括所接收组件的大小；

通过执行对第一和第二存储器区域的依次搜索来标识并保留足够大以容纳所接收的组件的存储器块，其中所述第一存储器区域被重写以除去不用的存储器块并且当该大小超过第一和第二存储器区域的可用存储器块的大小时，第一和第二存储器区域被重新分配；

提供被保留的存储器块的地址；以及

通过将所接收的组件写入被保留的存储器块来更新原始电子文件。

26. 一种电磁媒介，其包括可执行指令，当在处理系统中被执行时，所述指令通过以下来更新电子文件和文件组件：

接收新电子文件的至少一个被接收的组件的标识信息，该新电子文件是原始电子文件的更新版本，其中所述标识信息包括所接收组件的大小；

通过执行对第一和第二存储器区域的依次搜索来标识并保留足够大以容纳所接收的组件的存储器块，其中所述第一存储器区域被重写以除去不用的存储器块，并且当该大小超过第一和第二存储器区域的可用存储器块的大小时，第一和第二存储器区域被重新分配；

提供被保留的存储器块的地址；以及

通过将所接收的组件写入被保留的存储器块来更新原始电子文件。

27. 一种用于更新电子文件的系统，其包括：

第一设备，其包括产生德耳塔文件的文件求差和更新系统的第一组件；

第二设备，其从第一设备经由至少一个耦合接收该德耳塔文件，所述第二设备包括文件求差和更新系统的第二组件，其被配置成通过以下来更新第二设备的电子文件：

经由该德耳塔文件接收新电子文件的至少一个组件，该新电子文件是原始电子文件的更新版本；

确定所接收组件的大小；

当该大小等于或者小于该原始电子文件的对应组件的大小时，

将所接收的组件写入原始存储器区域，该原始存储器区域包括原始电子文件的对应组件；以及

当该大小超过原始电子文件的对应组件的大小时，将所接收的组件写入与原始存储器区域相关的被保留的存储器区域。

28．一种用于更新电子文件的方法，其包括：

接收新电子文件的至少一个组件，该新电子文件是原始电子文件的更新版本；

确定所接收组件的大小；

当该大小等于或者小于该原始电子文件的对应组件的大小时，将所接收的组件写入原始存储器区域，所述原始存储器区域包括原始电子文件的对应组件；以及

当该大小超过原始电子文件的对应组件的大小时，将所接收的组件写入与原始存储器区域相关的被保留的存储器区域。

29．如权利要求28所述的方法，其中所述至少一个组件包括一组组件。

30．一种设备，其包括：

装置，其用于接收新电子文件的至少一个组件，该新电子文件是原始电子文件的更新版本；

装置，其用于确定所接收组件的大小；

装置，其用于当该大小等于或者小于原始电子文件的对应组件的大小时，将所接收的组件写入原始存储器区域，所述原始存储器区域包括原始电子文件的对应组件；以及

装置，其用于当该大小超过原始电子文件的对应组件的大小时，将所接收的组件写入与原始存储器区域相关的被保留的存储器区域。

电子文件更新期间的设备存储器管理

相关申请

本申请相关于下列美国专利申请：2002 年 5 月 13 日提交的申请号 10/146,545；2002 年 9 月 30 日提交的申请号 10/261,153；2002 年 11 月 12 日提交的申请号 10/292,245；2002 年 11 月 18 日提交的申请号 10/298,393；2002 年 11 月 18 日提交的申请号 10/298,863；2002 年 11 月 18 日提交的申请号 10/298,862；以及 2002 年 11 月 18 日提交的申请号为 10/298,896，所有这些当前都是未决的。

技术领域

所公开的实施例涉及电子文件更新期间的存储器管理。

背景技术

在中央处理单元（CPU）上运行以实现主机设备某种功能性的软件经常随着时间而改变。这种改变可来自于纠正软件缺陷，适于演变的技术，或者添加新特征到主机设备的需要。特别地，在移动无线设备，如蜂窝电话中，嵌入式软件经常包括比其它便携式设备或处理系统多的缺陷并且通常需要较频繁的更新以添加新的特征到该设备。

软件包括一个或多个文件，其可以处于美国信息交换标准码（ASCII）纯文本文件或二进制码的形式。软件文件可被分为较小的单位，其经常被称作组件或模块。在当前技术中，UNIX 平台或者个人计算机（PC）包括多个软件组件，并且每个软件组件通过由对应的操作系统（OS）支持的文件系统来独立地管理和更新。用来更新被掌管于 UNIX 平台或 PC 上的软件文件或软件组件的信息可通过互联网来传递或从第二存储媒介，如软盘，光盘只读存储器（CD-ROM），或小型闪卡（compact

flash card）来装入。

相反，在大多移动无线设备中，典型地使用实时 OS（real-time OS，RTOS），其中所有软件组件被链接为单一的大文件。此外，典型地在这些移动无线设备中不提供有文件系统支持。另外，该单一的大文件需要使用慢的通信链路，如无线电、红外、或串行链路来预装入或嵌入到该设备。

经由慢通信链路来更新大文件的阻碍包括将更新的文件递送到设备的时间。在差别文件被传递到掌管目标用于更新/修订的文件的设备的情况中，此时间包括传递差别文件的通信时间和接收设备处理差别文件并更新/修订作为目标的文件所用的时间，此处称为主机设备处理时间。尽管如所述“相关申请”所描述的，使用德耳塔文件来传递更新的软件文件或者软件组件信息显著减少所述通信时间，但是主机设备处理时间的减少至少部分由主机设备体系结构来规定。

该主机设备可以是众多基于处理器的设备中的任何一种，包括蜂窝电话和其它移动通信设备，个人数字助理（PDA），以及个人计算机。以蜂窝电话为例，典型的蜂窝电话体系结构包括快闪只读存储器（ROM），此处被称为快闪 ROM 或者快闪存储器，以及 RTOS。快闪存储器和 RTOS 两者都潜在地对减少主机设备处理时间的目标产生阻碍。

关于快闪存储器在减少主机设备处理时间中引起的问题涉及对快闪存储器执行写入的方式。快闪存储器以单位被分配，其经常被称作块，扇区，或段，每个都包括大量字节。写入（或者重写）典型的快闪存储器需要擦除并且写入（或者重写）整个单位，意味着单个位或者字节不能被写入（或者重写）。因此存储在快闪存储器中的字节的更新或者修订需要擦除并且重写存储该字节的整个单位。这样，更新或者修订存储在快闪存储器中的文件所需要的时间典型地大于更新或者修订存储在其它类型的可写入（或者重写）单个字节的存储或存储器中的文件所需要的时间。

关于RTOS在减少主机设备处理时间中引起的问题涉及主机设备软件程序被组织和管理的方式。典型蜂窝电话的RTOS不支持用于在运行时间的函数调用的文件管理和动态地址解析。因此，蜂窝电话的所有嵌入软件组件或者程序被链接为单个的，大的寄主程序（host program），并且该寄主程序内的函数直接通过该函数预编译的起始地址从该寄主程序的任何部分被调用。这样，与支持用于函数调用的文件管理和动态地址解析的典型计算机OS相比，典型的蜂窝电话RTOS不具有管理作为分离文件的多个软件组件并支持经由运行时间调用地址解析从一个软件组件到另一个的函数调用的能力。因此，包括RTOS的系统的寄主程序管理单个的大的程序并使用静态寻址来直接访问程序的函数。

当更新和修订软件组件时，该典型设备体系结构可导致问题，这是因为当该修订或者更新导致寄主程序大小改变时，在寄主程序中的经更新/修订的软件组件之后的软件组件的起始地址有作为结果的改变。软件组件的起始地址的这种改变导致需要更新该寄主程序中的对应调用地址以及该软件组件内的指令/数据地址。这在处理时间方面是昂贵的。因此，与嵌入式软件更新/修订相关的主机设备处理时间很大部分上应归于到快闪存储器的数据的低效重写以及嵌入式软件在源代码水平的小的改变可导致该嵌入式软件在二进制代码水平的大的改变这样的事实。

附图说明

图1是一实施例下的使用字节水平的文件求差和更新（FDU）算法来更新电子文件的系统的块图。

图2是一实施例下的软件升级系统的块图。

图3是一实施例下的设备存储器的块图。

图4是一实施例下的非关键组件更新的流程图。

图5是一实施例下的关键组件更新的流程图。

图6是一实施例下的使用用于每个可升级的EBSC的保留存储器分

配来静态寻址的流程图。

图 7 是图 6 实施例下的被配置成支持静态寻址存储器管理的主机设备存储器的一部分的块图。

图 8 是一实施例下的使用用于可升级的 EBSC 组的保留存储器分配来静态寻址的流程图。

图 9 是图 8 实施例下的被配置成支持静态寻址存储器管理的主机设备存储器的一部分的块图。

图 10 是一实施例下的支持静态寻址的客户设备 ROM 图的块图。

图 11 是一实施例下的使用静态寻址的嵌入式软件发展和安排（depolyment）过程的块图。

图 12 是一实施例下的使用动态寻址来升级 EBSC 或者 EBSC 组的流程图。

图 13 是一实施例下的使用动态寻址的函数调用的流程图。

图 14 是一实施例下的支持动态寻址的客户设备 ROM 图的块图。

图 15 是图 3 和 14 的实施例下的示出包括升级客户、嵌入式软件区域、DMM 库、向量表以及垃圾表（garbage table）的存储器组件中的交互的例子的主机设备存储器块图。

图 16 是一实施例下的使用保留 API 来保留存储器区域以容纳 EBSC 新版本的流程图。

图 17 是一实施例下的使用保留 API 的规则来定位存储器区域以容纳 EBSC 新版本的流程图。

图 18 是一实施例下的嵌入式软件区域的不用区域的收集（垃圾收集）的流程图。

图 19 是一实施例下的在保留 API 的升级操作之前和之后实例主机设备存储器的块图。

图 20 是一实施例下的在升级和垃圾收集操作之后主机设备存储器的块图。

图 21 是一实施例下的使用动态寻址的嵌入式软件发展和安排过程的块图。

在附图中，相同的参考数字用于标识相同或者基本相似的元素或者动作。为了容易地标识对任何特定元素或者动作的讨论，参考数字中的最高有效数位或多个最高有效数位指的是该元素在其中第一次被引入的图号（如，元素 130 关于图 1 被第一次引入和讨论）。

除非下面另有说明，图中所示的各种块和结构的构造和操作具有常规设计。因此，这种块在此不需要进一步详细描述，因为它们将被相关领域的技术人员所理解。为了简短起见并使得不混淆本发明的详细描述，这种进一步的细节被省略。基于此处提供的详细描述，相关领域的技术人员可容易地进行对该图的必要修改。

具体实施方式

在这里详细描述一种存储器管理系统和方法。在执行存储器管理时，通过执行对第一和第二存储器区域的顺序搜索，主机设备的升级客户标识并保留足够大以容纳新软件组件的存储器块。该新软件组件是原始软件文件组件的更新或者升级版本。当该新组件的大小等于或者小于原始电子文件的对应组件的大小时，更新客户分配第一存储器区域的原始存储器块以将所接收的组件写入其中。该原始存储器块包括原始电子文件的对应组件。

当该新组件的大小超过原始电子文件的对应组件的大小时，为了足够大以存储所接收的组件的存储器块，升级客户搜索第一存储器区域的不用区段（portion）。升级客户保留适当大小的存储器块，如果有一个被定位在第一存储器区域中的话，否则升级客户搜索第二存储器区域。升级客户保留适当大小的存储器块，如果有一个被定位在第二存储器区域中的话。

然而，当新组件的大小超过第一和第二存储器区域的可用块的大小

时，升级客户重写第一存储器区域以除去不用的存储器块。在重写第一存储器区域之后，在组件的大小和第一存储器区域的可用块允许的情况下，通过依次将第二存储器区域的软件组件重写到第一存储器区域的不用区段，升级客户重新分配第一和第二存储器区域。随后，升级客户重新指定第一存储器区域的剩余不用区段作为第二存储器区域的部分。在重新分配之后，升级客户将新组件写入第二存储器区域并且更新向量表。

关于访问主机设备软件组件，升级客户接收来自主机设备主程序的函数调用，其中该函数调用包括对应软件文件的标识信息。升级客户使用该标识信息以访问向量表并且从该向量表读取对应软件文件的起始地址。通过使用来自向量表的该起始地址连同标识信息，升级客户产生用于该对应软件文件的调用。

图 1 是一实施例的使用字节水平的文件求差和更新（FDU）算法，在这里被称作 FDU 算法，来更新电子文件的系统的块图。该 FDU 算法包括求差组件和更新组件。求差组件，此处被称作文件求差算法并且在所述“相关申请”中被详细描述，根据电子文件的原始版本和新版本在第一计算机系统中产生差别文件。更新组件，此处被称作文件更新算法并在所述“相关申请”中被详细描述，通过使用该差别文件和所掌管的原始文件的拷贝在第二计算机系统上产生新文件的拷贝。在以下描述中，大量特定细节被引入以提供本发明实施例的全面理解并使得本发明的实施例能够被描述。然而，相关领域的技术人员将理解，本发明可不用一个或者多个所述特定细节或者用其它组件、系统等来实践。另外，熟知的结构或者操作不被示出，或者不被详细描述，以避免混淆本发明的方面。

参阅图 1，第一计算机系统 102 和第二计算机系统 104 经由通信路径 106 通信。这些计算机系统 102 和 104 包括本领域所知的一起操作的计算设备的任何集合。计算机系统 102 和 104 也包括较大计算机系统内的组件。通信路径 106 包括文件在计算机系统 102 和 104 之间被传送和

传递所借助的任何媒介。因此，该路径 106 包括无线连接，有线连接和混合无线/有线连接。通信路径 106 还包括到网络的耦合或者连接，所述网络包括局域网（LAN），城域网（WAN），广域网（WAN），专属网（proprietary networks），局间或者后端网（interoffice or backend network），和互联网。此外，通信路径 106 包括可移动的固定媒介，如软盘，硬盘驱动器，和 CD-ROM 盘，以及快闪 RAM，通用串行总线（USB）连接，RS-232 连接，电话线，总线，和电子邮件消息。

第一通信系统 102 接收电子文件的原始或者旧版本 110 和新版本 112。新文件 112 通常是原始文件 110 的更新的或者修订的版本，但不限于此。电子文件 110 和 112 包括软件文件，其包括动态链接库文件，共享目标文件，嵌入式软件组件（EBSC），固件文件，可执行文件，包括十六进制数据文件的数据文件，系统配置文件，和包括个人使用数据的文件，但不限于此。由于任何类型的文件都可被看作是字节流，以下文件可被描述为字节流。

文件求差算法 114 接收新文件 112，将它与原始文件 110 比较，并计算所比较文件之间的字节水平的差别，如以下所述。文件求差算法 114 也可预处理原始 110 和新 112 文件以在计算文件差别之前减小所述文件的大小。在所述比较期间，文件求差算法 114 产生差别文件 116，此处被称作德耳塔文件。

德耳塔文件 116 的内容提供了新和原始文件之间字节水平差别的有效表示。德耳塔文件 116 包括元数据（meta-data）以及表示相关文件的新的或当前版本和该文件的以前版本之间的差别的替换和/或插入操作的实际数据（actual data），如下所述。文件求差算法 114 使用最小数目的字节和预定的格式或者协议在德耳塔文件 116 中提供原始 110 和新 112 文件之间的任何差别，由此提供在空间上最优化的德耳塔文件。

德耳塔文件 116 经由通信路径 106 被传递或者发送到另一个计算机系统 104。在传递之前，德耳塔文件 116 可使用本领域所知的压缩技术

被压缩，但不限于此。被掌管在接收计算机系统 104 上的文件更新算法 118 使用德耳塔文件 116 连同所掌管的原始文件 110 以产生或者创建新文件 112 的拷贝。然后新文件 112 的拷贝被用于更新被掌管在客户设备 104 上的原始文件 110，其目标用于修订或者更新。一旦完成该更新过程，现在存储在第二计算机系统 104 上的原始文件 110 与在第一计算机系统中所接收的新文件相同。

原始文件和新文件之间的差别典型地小于该新文件，从而使得如果该差别代替整个新的字节流被发送或者存储，则导致显著的存储和传送的节省。这对于掌管经由通常可能很慢并且很贵的连接，例如无线或者蜂窝连接来更新的程序的移动电子设备尤其重要。

图 2 是一实施例的软件升级系统 200 的块图，此处被称作升级系统。在支持客户设备的软件维护和应用管理中，升级系统 200 使用一实施例的德耳塔文件和文件更新算法，其中所述客户设备包括移动电子设备、移动通信设备、蜂窝电话、个人数字助理、计算机，以及其它基于处理器的设备。通过使载体和设备制造者能够经由无线基础设施有效地分配电子文件内容，该支持被提供用于范围从固件到嵌入式应用的所有设备软件。

通过支持提供新的或者修订的软件文件经由服务提供者的无线基础设施的各种机构到移动客户设备，升级系统 200 防止设备再调用（device recall）。这些系统通过接收来自软件分配器的新的或者修订的软件并使用文件求差算法从该新软件产生德耳塔文件而起作用。该德耳塔文件经由服务提供者的基础设施被传递到客户设备。该接收或者客户设备的包括文件更新算法的升级客户使用该德耳塔文件更新被掌管在客户设备上作为目标的软件。

升级系统 200 使能设备软件和硬件之间不同的生命周期。因此该升级系统支持最新的设备功能性，这基于这样的事实：关键软件组件，如 Java™ 连接限制设备配置（CLDC）库、移动画面专家组-1（MPEG-1）

层 III（MP3）驱动、通信软件以及浏览器应用，演变要比主机设备硬件快并且因此以比移动设备制造者交付新一代设备快的频率更新。下面进一步详细描述升级系统 200。

参考图 2，升级系统 200 经由与设备 212 的无线连接来维护客户设备 104 上的嵌入式软件组件，由此使无线载体能够持续提供最新的数据服务给用户。数据系统 200 包括，但不限于，新软件组件分配器或者软件组件分配器 202，服务提供者升级组件 203-205，以及被掌管在客户设备 104 上的升级客户 130。服务提供者升级组件包括耦合在软件组件认证服务器 203 和升级管理器 205 之中的升级服务器 204。

一实施例的软件组件分配器 202 提供基于网的用户界面，通过该界面软件提供者打包并释放新的嵌入式设备软件组件，如改进的 MP3 驱动，升级的 Java™2 平台，Micro 版（J2ME™）移动信息设备简档（MIDP）库，或者添加特征的地址簿应用。软件组件分配器 202 的功能包括登记设备信息和将设备信息提交到软件组件认证服务器。软件组件分配器 202 接收新的和原始的 EBSC 并且通过使用文件求差算法计算字节水平的文件差别，登记并打包嵌入式软件，以及将嵌入式软件包提交给软件组件认证服务器。释放之后，该新软件经由有线、无线或者有线/无线混合网络耦合或者连接 220 被提供给服务提供者升级组件 203-205，但不限于此。

一实施例的软件组件分配器 202 被掌管在客户设备制造者的处理系统上。在可选择的实施例中，软件组件分配器 202 被掌管在软件提供者的处理系统上。在另一个可选择的实施例中，软件组件分配器 202 被掌管在通信服务器提供者的处理系统上，例如升级组件 203-205。

服务提供者升级组件 203-205 被耦合在软件组件分配器 202，客户设备 104 和包括现有网关 210 和通信基础设施 212，计费服务器 214，日志服务器 216 以及鉴权服务器 218 的服务提供者基础设施现有组件 210-218 之中。软件组件认证服务器 203 提供界面给设备制造者并且，

由此接收来自设备制造者的关于嵌入式软件包的新设备信息。软件组件认证服务器 203 还接收来自软件组件分配器的软件组件提交请求，提供新软件包的认可/拒绝通知给提交升级服务器，提供用于所提交和认可的软件包的盘管理，以及将认可的软件包重新打包并分配给升级服务器。此外，软件组件认证服务器 203 提供对于在软件组件提交过程中的潜在侵入和数据窜改的载体-等级安全控制。

在软件组件认证服务器 203 和升级服务器 204 之中充当接口的升级管理器 205 提供基于网的用户界面，无线载体系统管理员通过该界面检验并认可嵌入式设备软件组件升级。升级管理器 205 还为最佳设备管理而配置软件和数据打包，排定远程改变通知的时间，并且控制更新策略监控系统。此外，升级管理器 205 提供与现有基础设施，或者后端系统（计费、用户数据库鉴权、网入口）的整合，由此提供工作流程以确定鉴权、访问控制以及它们到现有计费 214 和日志 216 服务器的整合。

升级服务器 204 提供包括鉴权、连接和与移动设备通信的能力以执行嵌入式软件组件升级。与客户设备 104 通信可通过适于对应服务提供者的与客户设备的连接而进行，所述连接包括无线连接 212、有线连接、混合的有线/无线连接以及其它网络连接。此外，升级服务器 204 支持服务提供者的现有计费，数据收集和日志服务。

作为升级服务器 204 和客户设备 104 之中通信的实例，当德耳塔文件可用于从升级服务器 204 传递到客户设备 104 时，服务器 204 发送用户通知以向客户设备用户通知有可用于更新的软件组件。该用户通知可经由短消息服务（SMS）推送协议（push protocol）、超文本传递协议（HTTP）或者无线应用协议（WAP）采用文本消息的形式，但不限于此。一旦从手机用户接收到确认，升级服务器 204 使用原始的手机数据通信协议将德耳塔文件发送到请求的手机。

响应于收到来自手机的确认，升级服务器 204 鉴权并授权用户和/或请求设备，并检验请求设备预先必备的能力和限制。在鉴权之后，作

为客户设备配置数据管理器的升级服务器 204 标识请求设备 104 的嵌入式软件组件的当前版本，标识适当的德耳塔文件并将其传递到请求设备 104，记录升级事务的状况，并向升级管理器 205 报告结果。升级服务器 204 的实施例包括自动故障恢复机构。此外，升级服务器 204 通过空中传播来激活/去激活该软件升级服务，并将软件变化通知给远程用户。

参考图 1，升级客户 130 被嵌入在客户设备 104 的设备存储器 300 中，但不限于此。升级客户 130 存储并维护主机设备 104 的配置数据，并且使用文件更新算法 118 来提供嵌入式设备软件组件的维护和升级。升级客户 130 支持简单的用户界面并被结合到移动设备软件中。在执行时，升级客户 130 就自动检测任何嵌入式软件组件的远程变化，将嵌入式软件组件升级通知给用户，并基于载体和/或用户控制来升级软件组件，以适于特定的服务提供者。升级客户 130 还包括自动故障恢复机构。

在参与更新进程之前，客户设备确定诸多设备参数的状态。这样做是为了预先使该设备具有用于更新进程的条件，或者检验客户设备的条件是使更新进程一旦开始就可被完成。使客户设备预先具备的条件包括，确定客户设备是否处于基座（cradle）或者充电模式，客户设备是否被连接到串行缆，电池充电状态是否足以进行更新过程，所接收信号强度指示（RSSI）或者信号强度是否足够用于数据传递，以及作为目标的 EBSC 当前是否处于使用中。

一实施例的升级系统 200 支持经由德耳塔文件的诸多类型的软件文件或者组件更新。其更新被支持的文件类型包括可执行文件、字节流文件和数据文件，但是不限于此。可执行文件，或图像文件，包括在客户设备中使用以执行任务的软件文件，例如操作系统（OS），硬件设备驱动，和 K 虚拟机（K Virtual Machine，KVM）文件。字节流文件包括由其它可执行文件使用的文件，例如，图标文件，标志（logo）文件和 MP3 文件。数据文件包括包含个人使用数据和手机参考数据的文件，例如校准配置文件，协议独立组播（PIM）文件，和系统配置文件。

图 3 是一实施例下的设备存储器 300 的块图。一实施例的设备存储器 300 是快闪 ROM，但是许多类型的存储器和/或存储器类型的组合可被用于设备存储器 300 的可选择的实施例中。设备存储器 300 包括嵌入式软件区域 302、用于升级客户设备参数 306 的区域以及设备存储器管理区域 308。升级客户 130 与文件更新算法 118 一起被存储在一实施例的嵌入式软件区域 302 中。设备存储器管理区域 308 掌管设备存储器管理（DMM）库 310、向量表 312 以及垃圾表 314，如下面的详细描述。

一旦收到德耳塔文件，或者可替换地，新的 EBSC，升级客户 130 就控制包括嵌入式软件、嵌入式软件组件（EBSC）以及 EBSC 组的文件的修订和升级。升级客户使用诸多方法依据要被更新的文件类型和由客户设备制造者分配的用来支持这些更新的资源来更新 EBSC。这些更新方法包括如“相关申请”中所描述的如下方法：使用被保留的 ROM 更新操作系统（OS），通信协议和其它关键的软件组件；使用被保留的 RAM 更新 OS，通信协议和其它关键的软件组件；当没有被保留的 ROM 或者 RAM 时更新通信协议；以及单线（single-line）更新非关键 EBSC，但不限于此。

一实施例的更新方法包括非关键组件的更新和关键组件的更新。这些分类是基于目标用于更新的客户设备的软件组件的使用，并且在下面对其进行进一步描述。

非关键组件包括嵌入式软件组件（EBSC），其在更新过程期间故障发生之后容易通过空中被恢复。非关键组件的实例包括浏览器和 KVM 文件，但不限于此。图 4 是一实施例下的非关键组件更新的流程图 400。在块 402，当用德耳塔文件更新时，客户设备经由网络连接从升级服务器接收德耳塔文件。在块 404，一旦在客户设备中收到德耳塔文件，其就被写入设备存储器的指定区域，例如，RAM 和/或 ROM 存储。然后，在块 406，客户设备的升级客户使用德耳塔文件根据原始文件或者 EBSC 产生新文件或者 EBSC 的拷贝。在块 408，新文件的拷贝被适当地写入

保留的 RAM 或 ROM。在块 410，新文件的拷贝随后从保留的存储器写入包含原始文件或者 EBSC 的存储器位置。

关键组件包括在更新进程中使用的软件组件或对于设备操作关键的 EBSC。此外，关键组件还包括在更新过程中发生故障之后不容易通过空中恢复的 EBSC。关键组件的实例包括操作系统文件、协议栈、升级客户文件、通信库和显示器或 LCD 驱动文件，但不限于此。更新进程在这两个分类之间稍有不同。

图 5 是一实施例下的关键组件更新的流程图 500。在块 502，当使用德耳塔文件执行更新时，客户设备经由网络连接从升级服务器接收德耳塔文件。如上所述，德耳塔文件可包括可执行的文件或者 EBSC 升级、字节流和数据文件。一实施例的无线网络是蜂窝服务提供者的无线网络，但不限于此。

在块 504，一旦收到德耳塔文件，目标用于更新的原始文件或者 EBSC 通常就被从客户设备的快闪 ROM 拷贝到指定用于在更新过程中使用的存储器区域，例如，适当的 RAM 和/或 ROM 存储。客户设备分配该指定的存储器区域用于在存储该更新软件组件中使用。在块 506，德耳塔文件还被写入指定用于在更新过程中使用的存储器区域。

然后，在块 508，客户设备的升级客户使用所掌管的原始文件的拷贝与德耳塔文件一起产生新文件的拷贝。在块 510，新文件的拷贝被写入客户设备的预定的存储器区域，例如客户设备的保留的存储器。如必要该过程被重复。

在产生之后，在块 512，新文件的拷贝被从保留的存储器写入包含原始文件的原始存储器位置。当新文件的大小等于或者小于原始文件的大小时，该新文件被写入原始文件的存储器位置，从而替换原始文件。

如上所述，文件的修订涉及新的或者更新的 EBSC 到主机设备存储器位置的写入。通常该新 EBSC 不具有与它所替换的原始 EBSC 完全相同的大小或者完全相同的起始地址。一实施例的升级客户提供包括可升

级 EBSC 的静态寻址和动态寻址的设备存储器管理选项，以容纳新 EBSC 的写入而不考虑 EBSC 的大小或者起始地址。

通过使用保留的存储器区域，可升级 EBSC 的静态寻址一般提供用于在更新期间改变 EBSC 或者 EBSC 组的大小，而起始地址保持不变。因此，当使用静态寻址时，如果 EBSC 或者 EBSC 组的起始地址需要改变，那么整个嵌入式软件（EEBS）文件被重写，但是本实施例不限于此。

在滤及较先进的存储器管理的同时，可升级 EBSC 的动态寻址通常支持每个更新期间的 EBSC 或者 EBSC 组的起始地址和大小两者的修改。一实施例的升级客户通过使用一组应用程序或者编程界面（API）和至少一个数据表之中的交互来支持动态寻址。同样，动态寻址增加了 ROM 使用效率和设备存储器图设计效率，同时以较低的更新故障概率支持较快的更新处理。下面进一步详细描述静态和动态寻址。

一实施例的静态寻址包括两个可选择的办法，其一包括用于每个可升级的 EBSC 的保留存储器分配，其另一个包括用于特定 EBSC 组的保留存储器分配。保留存储器分配修改用于每个 EBSC 或者 EBSC 组的存储器分配以容纳软件更新。因此，将有附加的 ROM 被保留以容纳软件将来的增长，但是附加的存储器不限于 ROM。

图 6 是一实施例下的使用用于每个可升级 EBSC 的保留存储器分配来静态寻址的流程图 600。在操作中，在块 602，升级客户接收新的 EBSC。新 EBSC 可从软件组件分配器接收或者由升级客户从原始 EBSC 和德耳塔文件产生。在块 604，升级客户确定新 EBSC 的大小。在块 606，升级客户确定新 EBSC 的大小是否超过它所替换的对应 EBSC 的大小。如果新 EBSC 的大小不超过原始 EBSC 的大小，那么在块 608，升级客户保留存储原始 EBSC 的存储器块并且随后将新 EBSC 写入其中。如果新 EBSC 的大小超过原始 EBSC 的大小，那么在块 610，升级客户保留存储原始 EBSC 的存储器块以及相关保留存储器区域，并最终将新 EBSC 写入其中。写入新 EBSC 之后，在块 612，操作返回以接收另外的新 EBSC。

图 7 是图 6 实施例下的被配置成支持静态寻址存储器管理的主机设备存储器 702 的一部分的块图。作为比较，未被配置以支持静态寻址的主机设备存储器 704 的一部分被示出。支持静态寻址的存储器 702 包括用于每个可升级 EBSC 的保留存储器区域。在该实例中，可升级 EBSC 在设备制造的时候被标识。然后提供对应于每个可升级 EBSC 的保留存储器区域。尽管保留存储器区域被示为以对应的 EBSC 协同定位，但是它们可被定位或分布在设备存储器的任何地方。为了最小化保留存储器的量，保留的存储器区域不提供给非可升级 EBSC，但是本实施例不限于此。

继续本实例，可升级 EBSC 被标识为 EBSC1，EBSC2，和 EBSC5，并且被保留的存储器区域 1，2，和 5 被分别提供给这些可升级 EBSC 的每一个。每个被保留的存储器区域 1，2 和 5 的大小由设备制造者确定。将来对于可升级 EBSC 的升级和修订使用对应的保留存储器区域，如果该升级/修订引起所升级的 EBSC 的大小超过原始 EBSC 的大小的话，如上所述。

图 8 是一实施例下的使用用于可升级 EBSC 组的保留存储器分配来静态寻址的流程图 800。在操作中，在块 802，升级客户接收新的 EBSC。新 EBSC 又一次可从软件组件分配器接收或者由升级客户从原始 EBSC 和德耳塔文件产生。在块 804，升级客户确定新 EBSC 的大小。在块 806，升级客户确定新 EBSC 的大小是否超过它所替换的对应 EBSC 的大小。如果新 EBSC 的大小不超过原始 EBSC 的大小，那么在块 808，升级客户保留存储包括对应原始 EBSC 的 EBSC 组的存储器块并且随后将新 EBSC 写入其中。如果新 EBSC 的大小超过原始 EBSC 的大小，那么在块 810，升级客户保留存储原始 EBSC 的存储器块以及相关保留存储器区域，并最终将新 EBSC 写入其中。写入新 EBSC 之后，在块 812，操作返回以接收另外的新 EBSC。

图 9 是图 8 实施例下的被配置成支持静态寻址存储器管理的主机设

备存储器 902 的一部分的块图。作为比较，未被配置成支持静态寻址的主机设备存储器 904 的一部分被示出。支持静态寻址的存储器 902 包括为可升级 EBSC 的预定组保留的存储器区域。在该实例中，可升级 EBSC 在设备制造的时候被标识。在该实例中，EBSC1，2，4，5 和 8 是可升级的。

设备制造者使用许多因素中的至少一个来给可升级的 EBSC 分组。例如，具有相似更新频率的 EBSC 可形成一组。具有相似被更新/修订概率的 EBSC 也可形成一组。此外，相关或者互通功能的 EBSC 也可形成一组。为了本实例的目的，进行下列分组：EBSC 1，4，和 5 形成 EBSC 组 1；EBSC 3 和 7 形成组 2；EBSC 6 和 9 形成组 3；以及 EBSC 2 和 8 形成组 4。

分组之后，设备制造者提供对应于每个可升级 EBSC 组的保留存储器区域。尽管保留存储器区域被示为以对应的 EBSC 组协同定位，但是它们可被定位在设备存储器的任何地方。尽管非可升级 EBSC 也被放在一个或者多个组中，但是为了最小化保留存储器的量，没有保留的区域被提供，但是本实施例不限于此。

在此实例中，可升级 EBSC 组被标识为 EBSC 组 1 和 EBSC 组 4，并且保留的存储器区域 G1 和 G4 被分别提供给这些可升级 EBSC 组的每一个。每个被保留的存储器区域 G1 和 G4 的大小由设备制造者确定。将来对于可升级 EBSC 组的升级和修订使用对应的被保留存储器区域，如果该升级/修订引起所升级的 EBSC 组的大小超过原始 EBSC 组的大小的话。

上述对 EBSC 或者 EBSC 组的重写可导致包括被重写的 EBSC 或者 EBSC 组的文件内的子例行程序起始地址的改变。子例行程序起始地址的任何改变导致对应调用地址的改变，主机设备的其它子例行程序通过该调用地址访问被重写的子例行程序。因此，在一实施例中，升级客户支持对应于任何被重写的 EBSC 或者 EBSC 组的调用地址的升级。该调

用地址升级包括重写主机设备存储器中的包括对应于被重写的 EBSC 或者 EBSC 组中的子程序的调用地址的任何块。由于这些调用地址的改变典型地涉及的只是几个字节，因此发现只升级包括该调用地址的块比更新包含该调用地址的 EBSC 有效，但该实施例不限于此。

图 10 是一实施例下的支持静态寻址的客户设备 ROM 图 1000 的块图。该 ROM 包括下列区域，但不限于此：引导代码区域 1002，嵌入式软件区域 1004，EBSC 工作存储器区域 1006，升级客户设备参数区域 1008，以及至少一个为未定目的保留的区域 1010。

引导代码区域 1002 存储设备引导序列代码。嵌入式软件区域 1004 存储客户设备的嵌入式软件。该嵌入式软件包括，例如，浏览器软件，K 虚拟机（KVM），通信库，实时 OS，图形驱动，和升级客户，但本实施例不限于此。如上所述，用于每个可升级 EBSC 或者 EBSC 组的存储器分配被修改以容纳软件更新。因此，有附加的 ROM 被保留以容纳软件将来的增长。附加的保留 ROM 大约为对应 EBSC 或者 EBSC 组初始版本大小的 5%-20%，但不限于此。

在新版本的 EBSC 或者 EBSC 组由升级客户产生之后以及该新版本被写入当前由原始版本的 EBSC 或者 EBSC 组占据的嵌入式软件区域的存储器位置之前，EBSC 工作存储器区域 1006 临时存储该新版本的 EBSC 或者 EBSC 组。工作区域 1006 的估计大小至少为使用该存储器区域的所有 EBSC 或者 EBSC 组中最大的大小，包括关键和非关键组件。

升级客户设备参数区域 1008 存储特定于升级客户的客户设备配置数据和参数。区域 1008 的估计大小对应于与被掌管在客户设备上的设备配置相关的可升级 EBSC 的数目，但不限于此。

图 11 是一实施例下的使用静态寻址的嵌入式软件的发展和安排过程 1100 的块图。该图描述了例如从通过软件组件分配器 1102 的 EBSC 发展到使用具有基于静态寻址的设备存储器管理的一实施例的升级客户的设备 ROM 初始化的过程。

软件组件分配器1102的组件接收新的源代码，并编译1103新EBSC源代码。编译 1103 将所得到的 EBSC 目标代码耦合到链接器 1106 和1116，其作为新 EBSC 的登记的部分，从新 EBSC 目标代码产生新十六进制文件(文本)或者二进制文件 1118。新十六进制文件对应于新 EBSC。

在支持静态寻址中，软件组件分配器或者服务提供者修改原始图文件（map file）以插入用于每个可升级 EBSC 的附加存储器。在一实施例中，链接器 1106 产生修改的图文件 1104。当 EBSC 分组被使用时，原始图文件被修改以分组 EBSC 并插入用于每个可升级 EBSC 组的附加存储器。该图文件 1104 的修改包括使用链接器 1106 编辑链接文件 1106 以插入空文件或者组以及对应的目标文件。

向量产生工具 1105 从修改的图文件 1104 产生向量表文件 1108。向量表文件 1108 被下载到用于升级客户设备 1112 中的客户设备参数的保留的 ROM 区域 1110。参考图 1，升级客户 130 维护向量表 1108，其通常保持不变直到整个嵌入式软件（EEBS）被重写。链接器 1116 还使用修改的图文件 1104 以产生新十六进制或者二进制文件 1118。此外，链接器 1116 指定或者产生一列十六进制文件/图文件路径对关联（path pair association）。十六进制文件 1118 被下载到客户设备 1112 的嵌入式软件区域 1120，但是在可选择的实施例中可下载到客户设备存储器的其它区域。

尽管上述静态寻址减少主机设备处理时间并且使能够更新关键软件组件，但是它不容纳所有的升级和修订。另外，当任何 EBSC 的重写将超过对应保留存储器区域的大小时，静态寻址的使用使整个设备存储器的重写成为必要。因此，实施例使用动态寻址作为静态寻址的替换。

动态寻址滤及新文件版本的大小超过原始文件版本的大小的电子文件更新和修订。在支持动态寻址中，在文件更新过程和主程序调用包括 EBSC 的函数的函数调用过程二者期间，实施例的升级客户有效地管理设备存储器。

在使用动态寻址执行存储器管理中，通过执行对包括至少一个第一和第二存储器区域的主机设备存储器区域的顺序搜索，升级客户标识并保留足够大以容纳新软件组件的存储器块。当新组件的大小超过所搜索的存储器区域的可利用块的大小时，升级客户重写第一存储器区域以除去不用的存储器块，重新分配第一和第二存储器区域，将新组件写入第二存储器区域，并更新向量表，如下所述。

图 12 是一实施例下的使用动态寻址来升级 EBSC 或者 EBSC 组的流程图 1200。在块 1202，升级客户接收新 EBSC 版本。新 EBSC 版本从原始 EBSC 版本和对应的德耳塔文件产生，或者，可选择地，从软件组件分配器被接收。在块 1204，存储器区域被标识并保留于客户设备中以容纳新 EBSC 版本的存储。在块 1206，升级客户的组件将新 EBSC 版本写入保留的存储器区域。在块 1208，进行对向量表的更新以进行由新 EBSC 版本导致的向量表的信息的任何必要的改变。此外，在块 1210，对垃圾表进行更新以反映由新 EBSC 版本的写入导致的不用的存储器区域中的任何改变。

为了使用动态寻址来访问主机设备中的软件组件，升级客户接收来自主机设备的主程序的函数调用，包括对应软件的标识信息，从向量表读取对应软件文件的起始地址，并产生用于对应软件文件的调用。同样，当第一 EBSC 调用第二 EBSC 时，升级客户改变函数调用序列，从而代替使用静态地址来直接调用第二 EBSC，第一 EBSC 使用静态地址调用升级客户的 API。通过使用运行时间向量表的对应静态地址和参量，被调用的 API 将所接收的调用转换为对第二 EBSC 的调用。

图 13 是一实施例下的使用动态寻址的函数调用流程图。主机设备的主程序使用存储在主机设备存储器中的 EBSC 支持用户请求的函数。当一函数被用户请求时，通过使用静态地址来直接调用升级客户的组件，主机设备主程序的 EBSC 调用与特定函数相关的 EBSC。响应于该调用，在块 1302，升级客户接收来自主程序的信息 ，其包括函数或者 EBSC

标识以及该函数的参量。在块 1304，升级客户访问向量表，其包括可用于主程序的每个 EBSC 的项目以及 EBSC 的对应起始地址。在块 1306，升级客户从向量表读取起始地址和相关参量。在块 1308，通过使用起始地址信息以及从主程序接收的被调用的 EBSC 的信息，连同所有被接收的参量，升级客户产生对实际函数（actual function）和对应 EBSC 的调用。

图 14 是一实施例下的支持动态寻址的客户设备 ROM 图的块图。该 ROM 包括下列区域，但不限于此：引导代码区域 1402，嵌入式软件区域 1404，EBSC 更新区域 1406，升级客户设备参数区域 1408，设备存储器管理区域 1410，以及至少一个为未定目的保留的区域 1412。下面描述这些存储器区域的每一个，但是可选择的实施例可使用本领域技术人员所认可的许多不同的区域配置。

引导代码区域 1402 存储设备引导序列代码。嵌入式软件区域 1404 存储客户设备的嵌入式软件。该嵌入式软件包括，例如，浏览器软件，KVM，通信库，实时 OS，图形驱动，和升级客户，但不限于此。

EBSC 更新区域 1406 存储嵌入式软件组件的新版本。该区域 1406 的估计大小将随设备而不同。然而，通常 EBSC 更新区域 1406 大约为嵌入式软件区域 1404 大小的 10%－20%。

升级客户设备参数区域 1408 存储特定于升级客户的客户设备配置数据和参数。升级客户设备参数区域 1408 的估计大小对应于与被掌管在客户设备上的设备配置相关的可升级 EBSC 的数目，但不限于此。

设备存储器管理区域 1410 存储存储器参数，向量表和垃圾项目表，但不限于此。该区域 1410 的估计大小对应于与被掌管在客户设备上的设备管理相关的可升级 EBSC 的数目。

设备存储器管理区域 1410 的存储器参数包括 ROM 数目，起始地址，和为设备存储器管理保留的 ROM 区域的大小，以及为设备存储器管理区域 1410 保留的 ROM 区域中下一个可用存储器的起始地址和大小。存

储器参数还包括向量表和垃圾项目表的大小，以及垃圾项目表中有效项目的数目。向量表包括 EBSC 标识，ROM 数目，起始地址，和可升级 EBSC 的大小。垃圾项目表包括 ROM 数目，起始地址，和所有垃圾项目的大小。

图 15 是图 3 和 14 的实施例下的示出包括升级客户 130、嵌入示软件区域 302、DMM 库 310、向量表 312 以及垃圾表 314 的存储器组件中交互的例子的设备存储器 300 块图。DMM 库 310 包括三个 API 1502-1506。这三个 API 1502-1506 是用于操纵为设备存储器管理保留的 ROM 区域，为 EBSC 保留的区域，以及为 EBSC 更新保留的区域。这些 API 包括用于函数地址转换的 API 1502，用于 EBSC 的新版本保留的 API 1504，以及用于垃圾收集的 API 1506。下面对其每一个进行描述。

函数地址转换 API 1502 从主机设备主程序 1510 产生用于与特定主机设备函数相关的 EBSC 的调用，如上面关于图 13 的描述。主机设备主程序 1510 通过直接调用函数地址转换 API 1502 来调用与特定函数相关的 EBSC。进行 API 1502 的直接调用来代替 EBSC 的直接调用。在该调用时，函数地址转换 API 1502 接收来自主程序 1510 的信息，其包括函数标识信息以及函数的参量。

响应于该调用，函数地址转换 API 1502 访问向量表 312。向量表 312 包括可用于主程序 1510 的每个函数的项目以及设备存储器 302 中函数的对应起始地址。函数地址转换 API 1502 从向量表 312 读取由主程序 1510 请求的函数的起始地址，并使用该起始地址信息以及从主程序 1510 接收的被调用的 EBSC 的信息产生对实际函数连同所有被接收参量的调用。

实施例的升级客户 130 使用用于保留 EBSC 新版本的 API 1504，此处被称作保留 API 1504，来设置并保留存储器中的区域以容纳新版本的 EBSC 或者 EBSC 组。参考图 15，当被升级客户 130 调用时，保留 API 1504 接收新 EBSC 版本的标识信息，包括该新版本的文件大小的信息。该新 EBSC 版本的文件大小是要在设备 ROM 302 中被保留以存储新 EBSC 版

本的区域的大小。保留 API 1504 定位存储器 302 中的适当区域并且一旦成功保留该存储器区域，就返回所请求的保留区域的起始地址。图 16 是一实施例下的使用保留 API 1504 来保留存储器区域以容纳新版本的 EBSC 的流程图。

在分配所请求大小的存储器块时，保留 API 1504 应用如下规则集，但可选择的实施例可应用不同的规则以达到等价的结果。图 17 是一实施例下的使用保留 API 1504 的规则来定位存储器块以容纳新版本的 EBSC 的流程图。在块 1702，一旦接收到关于新 EBSC 大小的信息，在块 1704，保留 API 1504 确定新 EBSC 的大小是否超过原始 EBSC 的大小。当新 EBSC 的大小等于或者小于对应的原始 EBSC 的大小时，在块 1706，保留 API 1504 分配当前由对应原始 EBSC 占据的存储器块以接收新 EBSC。否则，保留 API 1504 试着从保留的存储器区域中的可用存储器定位具有被请求大小的存储器块。

在块 1708，在继续该搜索中，保留 API 1504 为具有被请求大小的存储器块而搜索客户设备的存储器区域。如果在被保留的存储器区域中找到适当大小的块，那么在块 1710，保留 API 1504 分配该存储器块以接收新 EBSC。如果在保留的存储器区域中没有可用的适当大小的存储器块，那么在块 1712，保留 API 1504 访问垃圾表 314。在块 1714，保留 API 1504 使用垃圾表 314 的信息来搜索不使用的主程序存储器区域，试图在对应于垃圾表 314 中的项目的不用区域之中定位被请求大小的存储器块。实施例的不用区域，此处被称作垃圾区域，包括主程序的不用区域，但该实施例不限于此。如果在不用的存储器区域中找到适当大小的存储器块，那么在块 1716，保留 API 1504 分配该块以接收新 EBSC。

如果不能定位适当大小的存储器块，那么在块 1718，实施例的保留 API 1504 起动此处称作垃圾收集的过程。当垃圾表 214 的所有项目都被占据时，保留 API 1504 也起动垃圾收集。在一实施例中，用于垃圾收集的 API 1506，此处被称作垃圾收集 API 1506，被保留 API 1504 调用，

但不限于此。用于垃圾收集的 API 通常不接收参量并清除为 EBSC 和 EBSC 更新保留的区域中的所有垃圾，但不限于此。

如上所述，如果没有为请求的保留大小找到足够的存储器或者垃圾项目表变满，那么 API 1504 就起动垃圾收集。图 18 是一实施例下的垃圾收集的流程图 1800。在起动垃圾收集时，API 1504 调用垃圾收集 API 1506。当被调用时，在块 1802，垃圾收集 API 1506 读取垃圾表项目，并且在块 1804，使设备存储器的嵌入式软件区域被重写，以便将存储器的 EBSC 在存储器中依次打包（pack up）。可选择的实施例可在主存储器中将 EBSC 解包（pack down），但本实施例不限于此。该打包操作去除嵌入式软件区域中不用的区域。

在该打包操作之后，在块 1806，垃圾收集 API 1506 评估存储在保留的存储器区域中的每个 EBSC 的大小，并且将该大小与在重写之后的主程序的嵌入式软件区域中剩余的任何不用的存储器块进行比较。该评估确定重写嵌入式软件区域之后剩余的不用存储器的量是否能容纳保留存储器区域的任何 EBSC，从而使在可能的情况下，EBSC 可以从保留的存储器区域移出并进入设备存储器的嵌入式软件区域。保留的存储器区域的 EBSC 被依次评估，但本实施例不限于此。在块 1808，在 EBSC 的大小允许的情况下，EBSC 从保留的存储器区域被重写到嵌入式软件区域。在一个实施例中 EBSC 被重写时，它们在打包的 EBSC 之后被依次存储在嵌入式软件区域中。

对存储在保留的存储器区域中的 EBSC 的评估继续进行，直到所有 EBSC 已被移动到嵌入式软件区域，或者直到确定嵌入式软件区域中没有足够大的剩余区域以容纳保留的存储器区域的 EBSC。不能被容纳在嵌入式软件区域中的任何 EBSC 留在保留的存储器区域，但本实施例不限于此。

一旦完成了打包和移动操作，在块 1810，通过重新指定主机设备主存储器的任何不用存储器块作为保留存储器区域，垃圾收集 API 1506 重

新分配主机设备存储器。此外，在块 1812，保留存储器区域指针被复位以维持保留存储器区域中可用存储器的起始地址。另外，在块 1814，垃圾表被重写以反映不用存储器区域的状态。在将新版本的 EBSC 写入与由相同 EBSC 的原始版本占据的存储器区域不同的任何存储器区域之后，升级客户 130 还更新向量表 312。

以下是升级和垃圾收集操作的例子，参考图 15，17，19，和 20。图 19 是一实施例下的在保留 API 的升级操作之前 1902 和之后 1904 实例主机设备存储器的块图。图 20 是一实施例下的在升级和垃圾收集操作之后主机设备存储器 2004 的块图。

该实例从保留 API 接收关于新版本的 EBSC 8 的信息开始。一旦接收到新版本的 EBSC 8 的文件大小，就进行关于该新版本的大小是否等于或者小于原始 EBSC 大小的确定。在该实例中，原始版本 EBSC 8 包含 400 字节，而新版本的 EBSC 8 包含 380 字节。因此，保留 API 分配当前由原始版本 EBSC 占据的区域 1920 以接收新版本的 EBSC 8。

该实例以保留 API 接收关于新版本的 EBSC 5 的信息继续。一旦接收到新版本的 EBSC 5 的文件大小，就进行关于该新版本的大小是否等于或者小于原始 EBSC 大小的确定。在该实例中，原始版本 EBSC 5 包含 300 字节，而新版本的 EBSC 5 包含 360 字节。由于该新版本的大小大于原始版本的大小，因此，保留 API 尝试从保留存储器区域 1910 的可用存储器中定位具有所请求的大小的存储器块。在该实例中，保留存储器区域 1910 中的存储器是可用的，所以保留 API 分配保留存储器区域 1910 中的区域 1922 以接收新版本的 EBSC 5。此外，保留 API 分配当前由原始版本 EBSC 5 占据的区域作为不用区域 1924。

接着，保留 API 接收关于新版本的 EBSC 7 的信息。原始版本 EBSC 7 包含 550 字节，而新版本的 EBSC 7 包含 560 字节。由于该新版本的大小大于原始版的大小，所以保留 API 尝试从保留存储器区域 1910 的可用存储器中定位具有所请求大小的存储器块。在该实例中，保留存储器

区域 1910 中的存储器是可用的，所以保留 API 分配保留存储器区域 1910 中的下一个可用区域 1926 来接收新版本的 EBSC 7。此外，保留 API 分配当前由原始版 EBSC 7 占据的区域作为不用区域 1928。

继续该实例，保留 API 接收关于新版本的 EBSC 2 的信息。原始版本 EBSC 2 包括 330 字节，而新版本的 EBSC 2 包含 360 字节。保留 API 确定保留存储器区域 1910 中没有适当大小的区域可用，并且访问垃圾表 314。垃圾表 314 被使用以试图在使用垃圾表 314 中的项目标识的垃圾区域之中定位请求大小的存储器块。在此实例中，适当大小的不用区域 1930 被标识，并且保留 API 分配不用的区域 1930 以接收新版本的 EBSC 2。

如上所述，如果不能为所请求的大小找到足够的存储器，或者垃圾项目表变满，那么保留 API 启动垃圾收集。对于该实例，假设在保留存储器区域中为新版本的 EBSC 2 分配之后所接收的下一个新 EBSC 版本导致启动垃圾收集。一实施例的垃圾收集在主机设备存储器中将 EBSC 打包，导致新的主机设备存储器配置 1904，其中 EBSC 的顺序是 EBSC 1，EBSC 4，EBSC 2，EBSC 3，EBSC 6，EBSC 8，EBSC 5，和 EBSC 7。一可选择实施例的垃圾收集例行程序在主机设备存储器中将 EBSC 解包。

图 21 一是实施例下的使用动态寻址的嵌入式软件发展和安排过程的块图 2100。该图描述了从通过软件组件分配器 2102 的 EBSC 发展到使用具有基于动态寻址的设备存储器管理的一实施例的升级客户的设备 ROM 初始化的过程。

软件组件分配器 2102 的组件接收新 EBSC 的源代码，并编译 2103 新 EBSC 源代码。编译 2103 将所得到的 EBSC 目标代码耦合到链接器 2106 和 2116，其作为新 EBSC 的登记的部分，从新 EBSC 目标代码产生新十六进制文件（文本）或者二进制文件 2118。该新十六进制文件对应于新 EBSC。

在支持动态寻址中，软件组件分配器2102使用编译器2103和链接器2106来产生图文件2104。该图文件2104被向量产生工具2105使用以产生EBSC的对应初始向量表2108。向量表2108随后被提供给链接器2116，并且链接器2116使用初始向量表2108产生十六进制（文本）或者二进制文件2118。十六进制（文本）或者二进制文件2118被下载到客户设备2112的嵌入式软件区域2120中，但是可选择的实施例可以将十六进制文件2118写入客户设备2112的其它存储器区域。升级客户的DMM库的组件维护向量表2108并且，同样支持EBSC的动态寻址。DMM库与存储在为升级客户设备参数2110保留的区域中的向量表2108重叠（overlap）。

以上所述的用于电子文件更新期间的设备存储器管理的系统和相关方法包括用于更新电子文件的系统，该系统包括：第一设备，其包括产生德耳塔文件的文件求差和更新系统的第一组件；以及第二设备，其从第一设备经由至少一个耦合接收该德耳塔文件。

第二设备包括文件求差和更新系统的第二组件，其被配置以通过以下来更新第二设备的电子文件：从该德耳塔文件读取新电子文件的至少一个新组件，该新电子文件是原始电子文件的更新版本；通过执行对第一和第二存储器区域的依次搜索，标识并保留足够大的存储器块以容纳该新组件，其中第一存储器区域被重写以去除不用的存储器块，并且当新组件的大小超过第一和第二存储器区域的可用的存储器块的大小时，第一和第二存储器区域被重新分配；将新组件写入保留的存储器块；当新组件被写入与包括原始电子文件的对应组件的原始存储器块不同的存储器块时，更新向量表。

第二设备还包括文件求差和更新系统的第二组件，其被配置以通过以下来访问该设备的电子文件：从设备的主程序接收函数调用，该函数调用包括对应电子文件的标识信息；从向量表读取对应电子文件的起始地址；并且使用该起始地址和标识信息来产生用于对应电子文件的调用。

一实施例的该系统的第二设备进一步包括在更新电子文件中使用的第一和第二应用编程界面（API）。

一实施例的该系统的第二设备进一步包括在访问电子文件中使用的第三应用编程界面（API）。

一实施例的该系统的第一设备包括基于处理器的设备，其可由被掌管在第二设备上的软件的至少一个提供者访问。

一实施例的该系统的第二设备包括至少一个基于处理器的设备，其从个人计算机，便携式计算设备，蜂窝电话，便携式通信设备，和个人数字助理中选择。

一实施例的该系统的耦合是以下的至少一个：无线耦合、有线耦合、混合无线/有线耦合，和与至少一个网络的耦合，所述网络包括局域网（LAN），城域网（MAN）和广域网（WAN），专属网，后端网，互联网，和可移动的固定媒介，包括软盘、硬盘驱动器和光盘只读存储器（CD-ROM），以及电话线，总线和电子邮件消息。

一实施例的原始和新电子文件包括软件文件，其包括动态链接库文件、共享目标文件、嵌入式软件组件（EBSC） 、固件文件、可执行文件、包括十六进制数据文件的数据文件、系统配置文件、以及包括个人使用数据的文件。

以上所述的用于电子文件更新期间的设备存储器管理的系统和相关方法包括用于主机设备中设备存储器管理的方法，其包括：接收新电子文件的至少一个被接收组件的标识信息，该新电子文件为原始电子文件的更新版本，其中所述标识信息包括所接收组件的大小；通过执行对第一和第二存储器区域的依次搜索来标识并保留足够大的存储器块以容纳所接收的组件，其中第一存储器区域被重写以去除不用的存储器块，并且当所述大小超过第一和第二存储器区域的可用存储器块的大小时，第一和第二存储器区域被重新分配；提供被保留的存储器块的地址；并且通过将所接收的组件写入被保留的存储器块来更新原始电子文件。

用于主机设备中设备存储器管理的方法进一步包括：当所接收的组件被写入与包括原始电子文件的对应组件的第一存储器区域中的原始存储器块不同的存储器块时更新第一表，其中所述第一表包括主机设备电子文件组件的组件信息，该组件信息包括组件标识，只读存储器（ROM）数，起始地址和大小。

在一实施例的方法中，所述标识和保留包括当所述大小等于或者小于原始电子文件的对应组件的大小时，保留第一存储器区域的原始存储器块，所述原始存储器块包括原始电子文件的对应组件。

在一实施例的方法中，所述标识和保留包括当所述大小超过原始电子文件的对应组件的大小时，为足够大以存储所接收的组件的存储器块而搜索第二存储器区域和第一存储器区域的不用区段中的至少一个。

在一实施例的方法中，第一存储器区域的重写和第二存储器区域的重新分配进一步包括：读取第二表，其包括对应于第一区域不用存储器块的至少一个项目；重写第一存储器区域的组件以依次将第一存储器区域的组件打包并且合并不用的存储器块；依次评估第二存储器区域的每个组件的大小并且在第二存储器区域的组件大小和所合并的不用存储器块的大小允许的情况下将第二存储器区域的组件重写到第一存储器区域；在依次评估和重写之后标识所合并的不用存储器块的剩余块；通过指定所合并的不用存储器块的剩余块作为第二存储器区域的部分来重新分配第二存储器区域；置位第二存储器区域的指针以维持经重新分配的第二存储器区域的可用存储器的起始地址；以及更新第二表。

在一实施例的方法中，所述第二表的项目包括不用的存储器块的信息，其包括只读存储器（ROM）数，起始地址，和大小。

用于主机设备的设备存储器管理的方法进一步包括通过以下来访问由主机设备的电子文件提供的函数：接收来自主机设备的主程序的函数调用，该函数调用包括对应电子文件的标识信息；从向量表读取对应电子文件的起始地址；以及通过使用所述起始地址和标识信息产生用于

该对应电子文件的调用。

以上所述的用于电子文件更新期间的设备存储器管理的系统和相关方法包括用于文件更新期间的设备存储器管理的方法，包括：使用被接收组件的标识信息来确定新电子文件的至少一个所接收组件的大小，所述新电子文件是原始电子文件的更新版本；当该大小等于或者小于原始电子文件的对应组件的大小时，分配第一存储器区域的原始存储器块，以将所接收的组件写入其中，所述原始存储器块包括原始电子文件的对应组件；当该大小超过原始电子文件对应组件的大小时，为足够大以存储所接收的组件的存储器块而搜索第二存储器区域和第一存储器区域的不用区段中的至少一个；以及当该大小超过第一和第二存储器区域的可用存储器块的大小时，重写第一存储器区域以去除不用的区段，重新分配第一和第二存储器区域，并分配第二存储器区域的存储器块，将所接收的组件写入其中。

在一实施例的方法中，第一存储器区域的重写进一步包括：读取表，其包括对应于第一存储器区域不用区段的至少一个项目；重写第一存储器区域的组件以依次打包第一存储器区域的组件并合并不用的存储器块。

在一实施例的方法中，重新分配所述第一和第二存储器区域进一步包括：依次评估第二存储器区域每个组件的大小，并且当第二存储器区域组件的大小和经重写的第一存储器区域的所合并的不用存储器块的大小允许时将第二存储器区域的组件写入经重写的第一存储器区域；在依次评估和写入之后，标识所合并的不用存储器块的剩余块；通过指定所合并的不用存储器块的剩余块作为第二存储器区域的部分，重新分配第一和第二存储器区域；置位第二存储器区域的指针以维持重新分配的第二存储器区域的可用存储器的起始地址；并且更新该表。

以上所述的用于电子文件更新期间的设备存储器管理的系统和相关方法包括用于管理电子设备的存储器的方法，包括通过以下来更新设

备的电子文件：接收新电子文件的至少一个新组件，该新电子文件是原始电子文件的更新版本；通过执行对第一和第二存储器区域的依次搜索，标识并保留足够大的存储器块以容纳新的组件，其中所述第一存储器区域被重写以除去不用的存储器块并且当新组件的大小超过第一和第二存储器区域的可用存储器块的大小时，第一和第二存储器区域被重新分配；将新组件写入保留的存储器块；当新组件被写入与包括原始电子文件的对应组件的原始存储器块不同的存储器块时，更新向量表。

用于管理电子设备的存储器的方法进一步包括通过以下来访问设备的电子文件：接收来自设备主程序的函数调用，该函数调用包括对应电子文件的标识信息；从向量表读取对应电子文件的起始地址；以及使用该起始地址和标识信息来产生用于对应电子文件的调用。

在一实施例的方法中，所述标识信息包括函数标识和该函数的参量。

以上描述的用于电子文件更新期间的设备存储器管理的系统和相关方法包括一设备，其包括：用于接收新电子文件的至少一个新组件的装置，该新电子文件是原始电子文件的更新版本；用于通过执行对第一和第二存储器区域的依次搜索来标识并保留足够大以容纳新组件的存储器块的装置，其中第一存储器区域被重写以去除不用的存储器块并且当新组件的大小超过第一和第二存储器区域的可用存储器块的大小时第一和第二存储器区域被重新分配；通过将新组件写入保留的存储器块以更新原始电子文件的装置；当新组件被写入与包括原始电子文件的对应组件的原始存储器块不同的存储器块时，更新向量表的装置。

一实施例的该设备包括从个人计算机、便携式计算设备、蜂窝电话、便携式通信设备、和个人数字助理中选择的至少一个基于处理器的设备。

一实施例的该设备包括第一应用编程界面（API），其作为用于标识和保留存储器块的装置。

一实施例的该设备进一步包括：从设备的主程序接收函数调用的装

置，该函数调用包括对应电子文件的标识信息；从向量表读取对应电子文件的起始地址的装置；以及使用该起始地址和标识信息来产生用于对应电子文件的调用的装置。

一实施例的该设备包括第二应用编程界面（API），其作为接收函数调用的装置。

以上描述的用于电子文件更新期间的设备存储器管理的系统和相关方法包括计算机可读媒介，其包括可执行指令，当在处理系统中被执行时，所述指令通过以下来更新电子文件和文件组件：接收新电子文件的至少一个所接收的组件的标识信息，该新电子文件是原始电子文件的更新版本，其中所述标识信息包括所接收组件的大小；通过执行对第一和第二存储器区域的依次搜索来标识并保留足够大的存储器块以容纳所接收的组件，其中所述第一存储器区域被重写以除去不用的存储器块并且当该大小超过第一和第二存储器区域的可用存储器块的大小时，第一和第二存储器区域被重新分配；提供保留存储器块的地址；通过将所接收的组件写入保留的存储器块来更新原始电子文件。

以上描述的用于电子文件更新期间的设备存储器管理的系统和相关方法包括电磁媒介，其包括可执行指令，当在处理系统中被执行时，所述指令通过以下来更新电子文件和文件组件：接收新电子文件的至少一个所接收的组件的标识信息，该新电子文件是原始电子文件的更新版本，其中所述标识信息包括所接收组件的大小；通过执行对第一和第二存储器区域的依次搜索来标识并保留足够大的存储器块以容纳所接收的组件，其中所述第一存储器区域被重写以除去不用的存储器块并且当该大小超过第一和第二存储器区域的可用存储器块的大小时，第一和第二存储器区域被重新分配；提供保留存储器块的地址；以及通过将所接收的组件写入保留的存储器块来更新原始电子文件。

以上所述的用于在电子文件更新期间的设备存储器管理的系统和相关方法包括用于更新电子文件的系统，该系统包括：第一设备，其包

括产生德耳塔文件的文件求差和更新系统的第一组件；以及第二设备，其从第一设备经由至少一个耦合接收该德耳塔文件。实施例的第二设备包括文件求差和更新系统的第二组件，其被配置以通过以下来更新第二设备的电子文件：经由该德耳塔文件接收新电子文件的至少一个组件，该新电子文件是原始电子文件的更新版本；确定所接收组件的大小；当该大小等于或者小于该原始电子文件的对应组件的大小时，将所接收的组件写入原始存储器区域，其包括原始电子文件的对应组件；以及当该大小超过原始电子文件的对应组件的大小时，将所接收的组件写入与原始存储器区域相关的保留存储器区域。

以上描述的用于电子文件更新期间的设备存储器管理的系统和相关方法包括用于更新电子文件的方法，包括：接收新电子文件的至少一个组件，该新电子文件是原始电子文件的更新版本；确定所接收组件的大小；当该大小等于或者小于该原始电子文件的对应组件的大小时，将所接收的组件写入原始存储器区域，其包括原始电子文件的对应组件；以及当该大小超过原始电子文件的对应组件的大小时，将所接收的组件写入与原始存储器区域相关的保留存储器区域。在实施例的该方法中，所述至少一个组件包括一组组件。

以上描述的用于电子文件更新期间的设备存储器管理的系统和相关方法包括一设备，其包括：用于接收新电子文件的至少一个组件的装置，该新电子文件是原始电子文件的更新版本；用于确定所接收组件大小的装置；用于当该大小等于或者小于原始电子文件的对应组件的大小时，将所接收的组件写入包括原始电子文件的对应组件的原始存储器区域的装置；以及当该大小超过原始电子文件的对应组件的大小时，将所接收的组件写入与原始存储器区域相关的保留存储器区域的装置。

本发明的方面可作为编程到多种电路中任何一种中的功能性而被实施，该多种电路包括：可编程逻辑设备（PLD），如现场可编程门阵列（FPGA），可编程阵列逻辑（PAL）设备，电可编程逻辑和存储设备以

及标准的基于单元（cell-based）的设备，以及专用集成电路（ASIC）。用来实施本发明方面的一些其它可能性包括：具有存储器的微控制器（如电可擦除可编程只读存储器（EEPROM）），嵌入式微处理器，固件，软件等。此外，本发明的方面可实施在这样的微处理器中，其具有基于软件的电路仿真，离散逻辑（顺序的和组合的），定制设备，模糊（神经）逻辑，量子设备，和上述设备类型的任何混合。当然，基础设备技术可以多种组件类型来提供，如金属氧化物半导体场效应晶体管（MOSFET）技术，像互补金属-氧化物半导体（CMOS），双极技术，像发射极耦合逻辑（ECL），聚合物技术（例如，硅-共轭聚合物和金属-共轭聚合物-金属结构），混合的模拟和数字技术等。

除非上下文中明确地另有需要，在所有说明和权利要求中，单词“包括”、“包含”等应被理解为包括的意思，与排他的或详尽的意思相对；也就是说，是“包括，但不限于”的意思。使用单数或复数的词也分别包括复数或单数。此外，单词“在此”、“在以下”以及相似意思的词，当在本申请中被使用时，指的是作为一个整体的本申请而不是指本申请的任何特定的部分。

本发明的所说明的实施例的以上描述不是旨在穷尽或将本发明限制到所公开的确切形式。尽管出于说明的目的，本发明的特定实施例和实例在此被描述，在本发明的范围内各种等效的修改是可能的，如相关领域的技术人员将认识到的。此处所提供的本发明的教导可被用于其它处理系统和通信系统，不仅用于上述的蜂窝电话系统。

以上描述的各种实施例的元素和动作可被组合以提供进一步的实施例。根据上面详细的描述，对于本发明的这些和其它变化可被进行。

以上所有参考以及美国专利和专利申请在此被引入作为参考。如果必要的话，本发明的方面可被修改以使用上面描述的各种专利和申请的系统、功能和概念来提供本发明更进一步的实施例。

一般来说，在下面权利要求中所用的术语将不被解释为将本发明限

制在说明书和权利要求中所公开的特定实施例，但是应被理解为包括在权利要求下操作以提供用于文件求差的方法的所有处理系统。因此，本发明不被此公开内容所限制，而是本发明的范围完全由权利要求确定。

尽管本发明的某些方面在下面以某些权利要求的形式来呈现，但是发明人设想了处于任何数目的权利要求形式的本发明的各个方面。例如，尽管本发明的只一个方面被陈述为在计算机可读媒介中被实施，但是其它方面可同样在计算机可读媒介中被实施。因此，发明者保留在提交本申请之后添加附加权利要求以将这样的附加权利要求形式用于本发明的其它方面的权利。

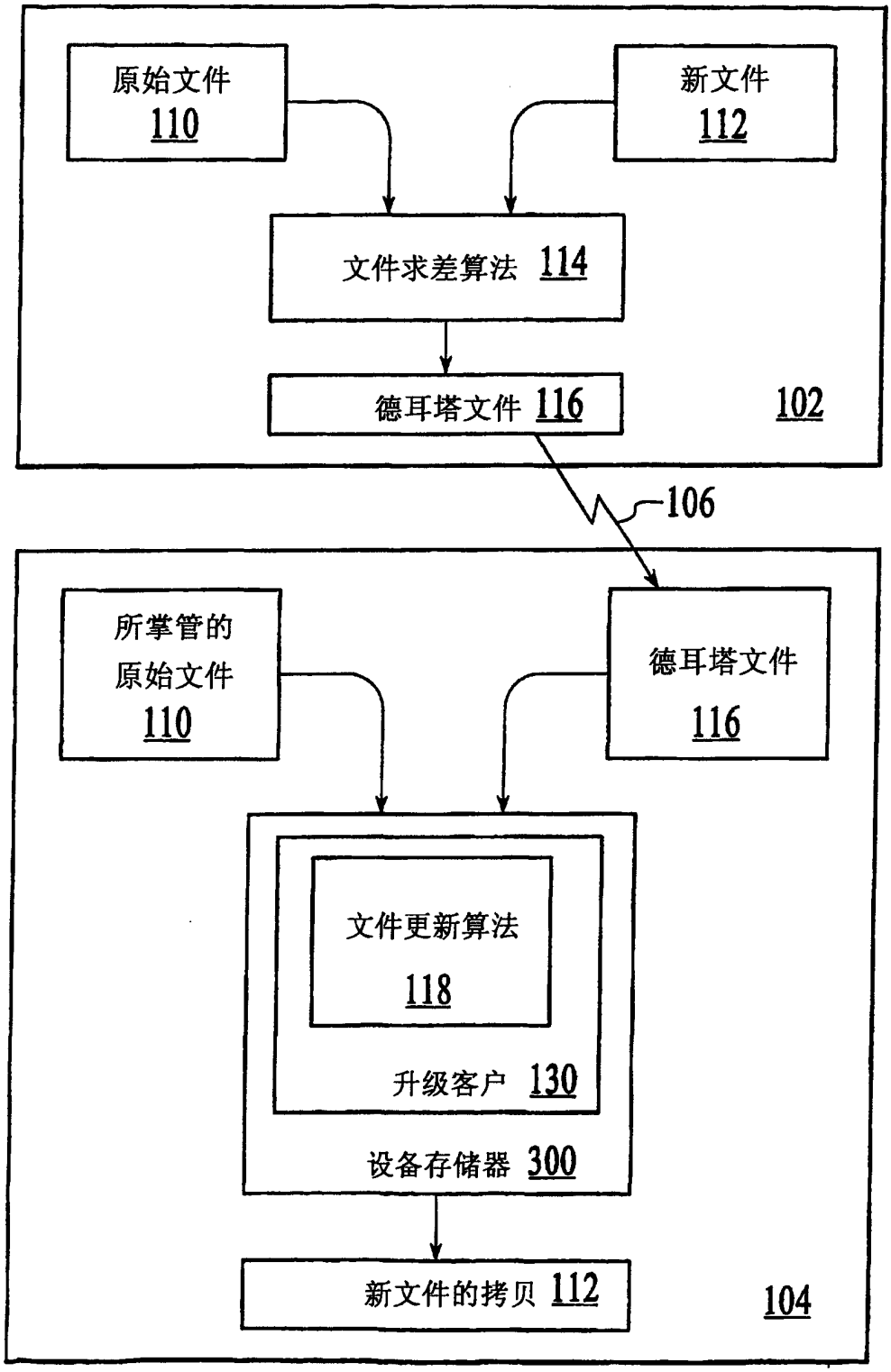

原始文件
110
新文件
112
文件求差算法 114
德耳塔文件 116
102
106
所掌管的
原始文件
110
德耳塔文件
116
文件更新算法
118
升级客户 130
设备存储器 300
新文件的拷贝 112
104

图1

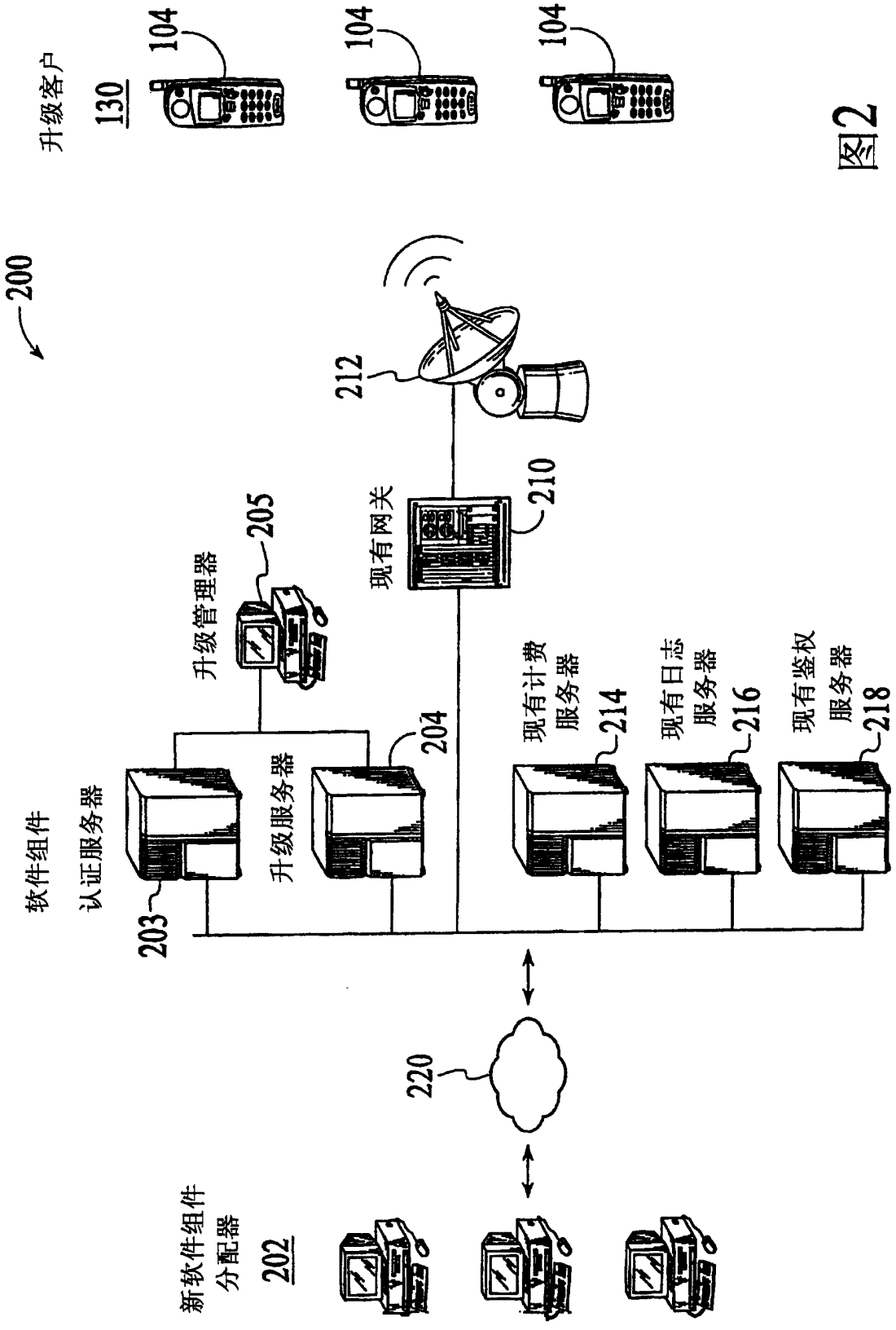

200
升级客户
130
104
104
104
212
现有网关
210
升级管理器
205
软件组件
认证服务器
203
升级服务器
204
现有计费
服务器
214
现有日志
服务器
216
现有鉴权
服务器
218
220
新软件组件
分配器
202

图2

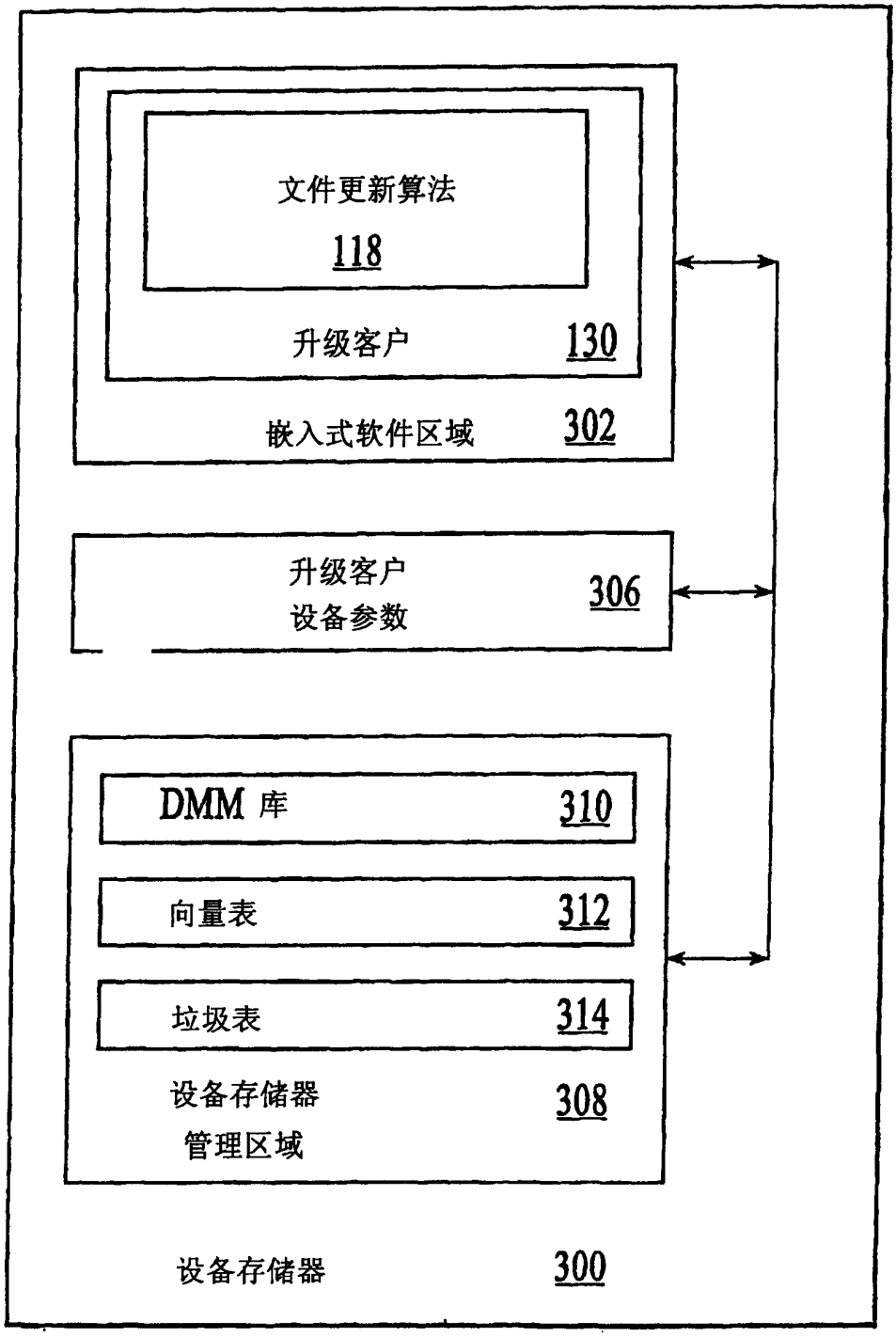

文件更新算法
118
升级客户
130
嵌入式软件区域
302
升级客户
设备参数
306
DMM 库
310
向量表
312
垃圾表
314
设备存储器
管理区域
308
设备存储器
300

图3

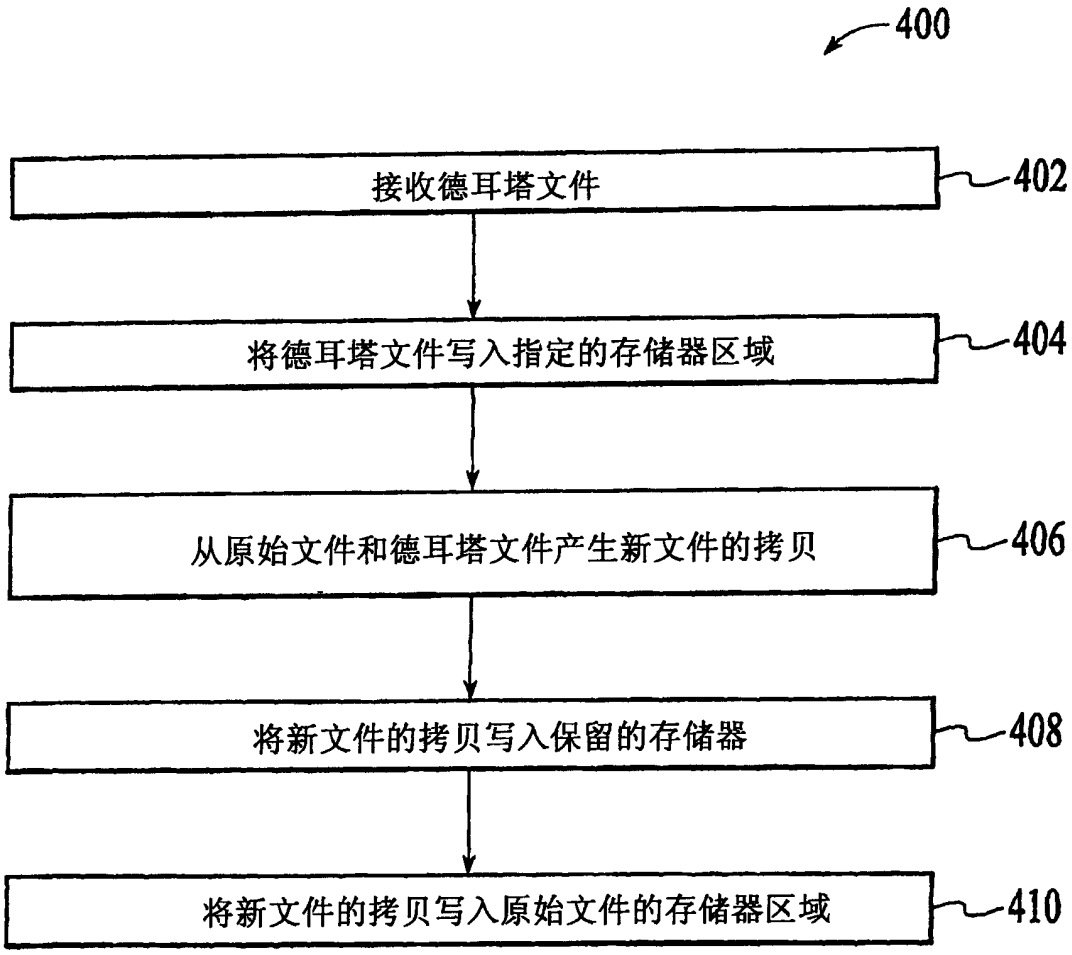

400
接收德耳塔文件
402
将德耳塔文件写入指定的存储器区域
404
从原始文件和德耳塔文件产生新文件的拷贝
406
将新文件的拷贝写入保留的存储器
408
将新文件的拷贝写入原始文件的存储器区域
410

图4

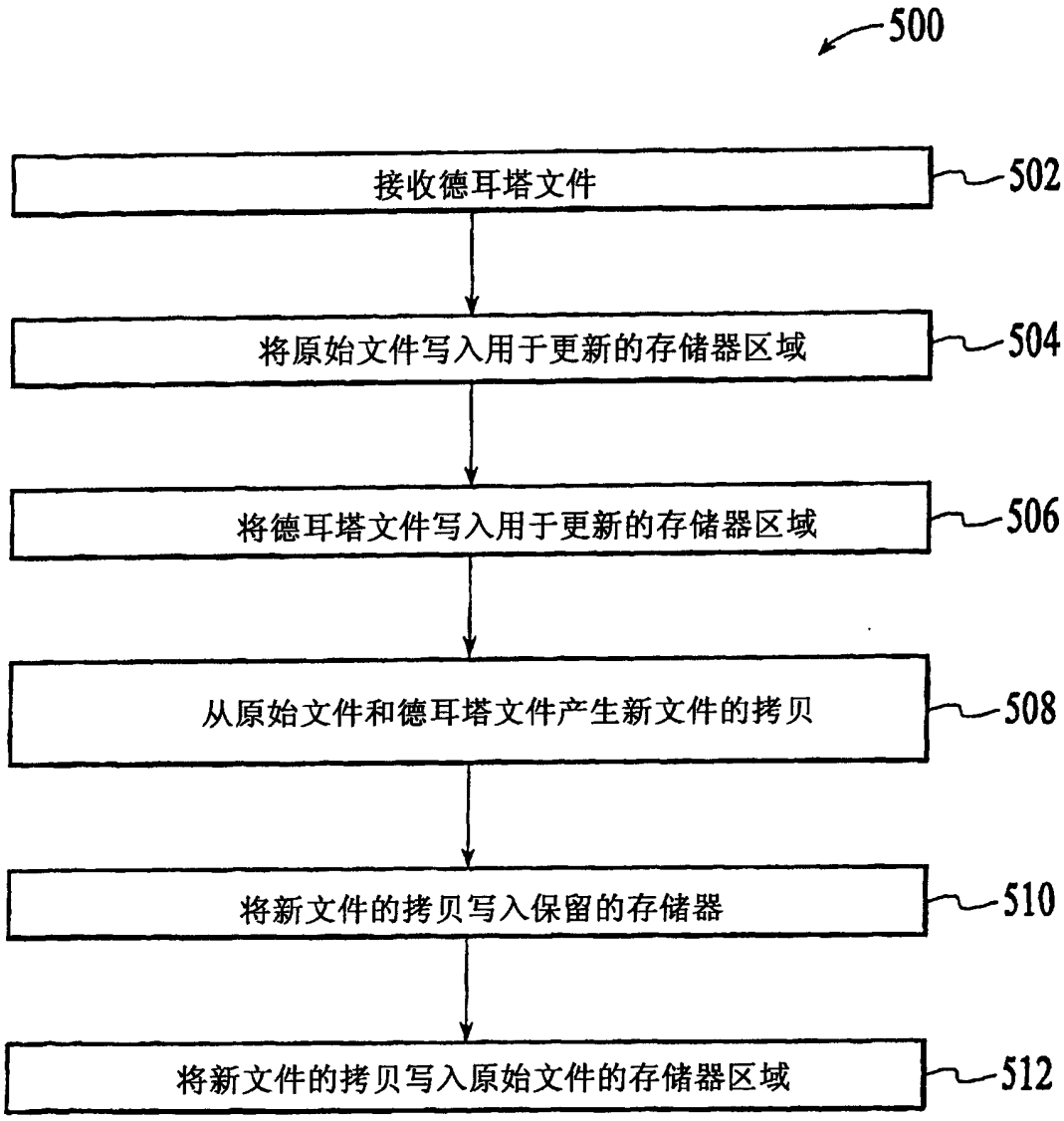

500
接收德耳塔文件
502
将原始文件写入用于更新的存储器区域
504
将德耳塔文件写入用于更新的存储器区域
506
从原始文件和德耳塔文件产生新文件的拷贝
508
将新文件的拷贝写入保留的存储器
510
将新文件的拷贝写入原始文件的存储器区域
512

图5

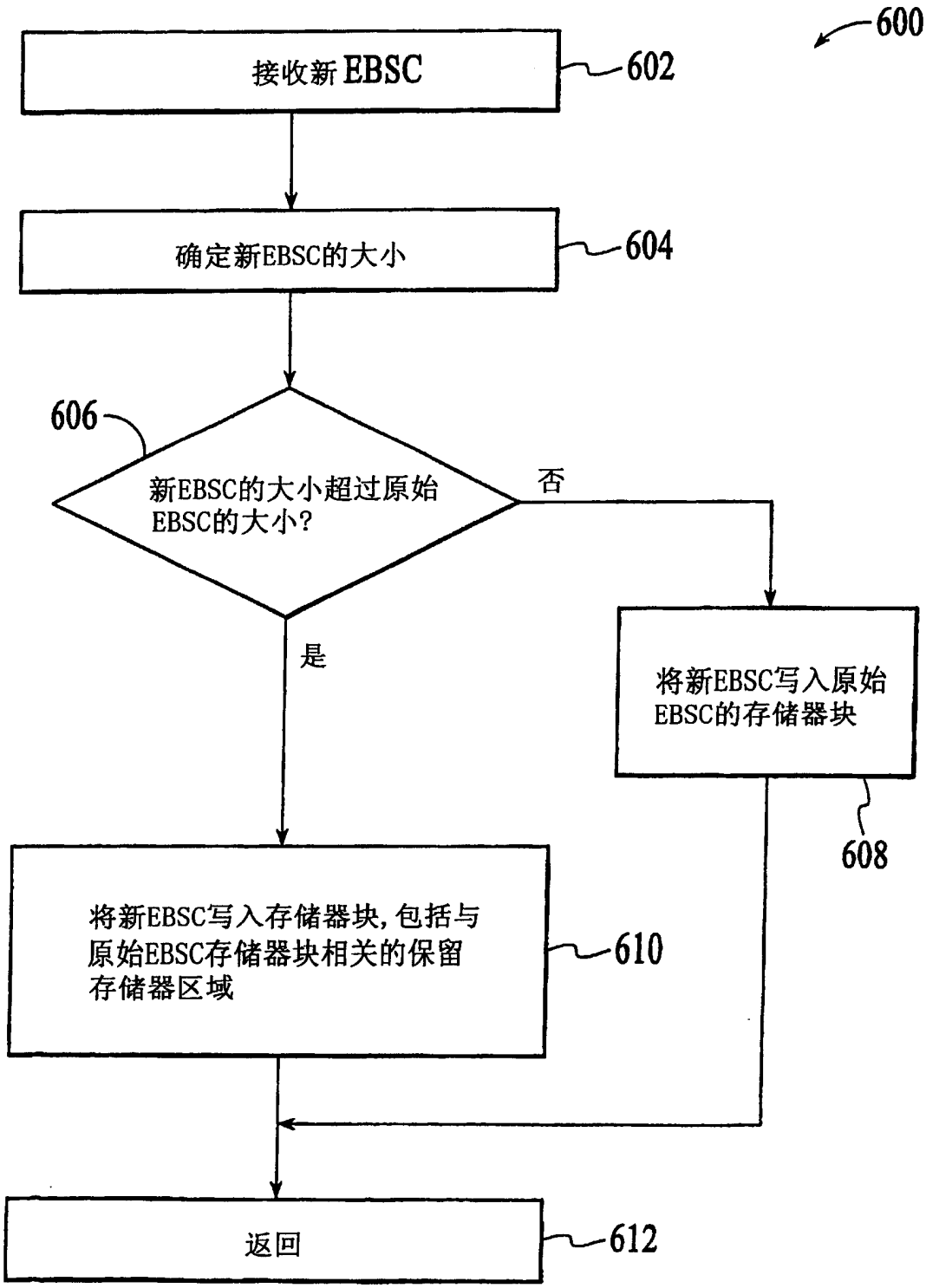

600
接收新EBSC
602
确定新EBSC的大小
604
606
新EBSC的大小超过原始EBSC的大小?
否
是
将新EBSC写入原始EBSC的存储器块
608
将新EBSC写入存储器块,包括与原始EBSC存储器块相关的保留存储器区域
610
返回
612

图6

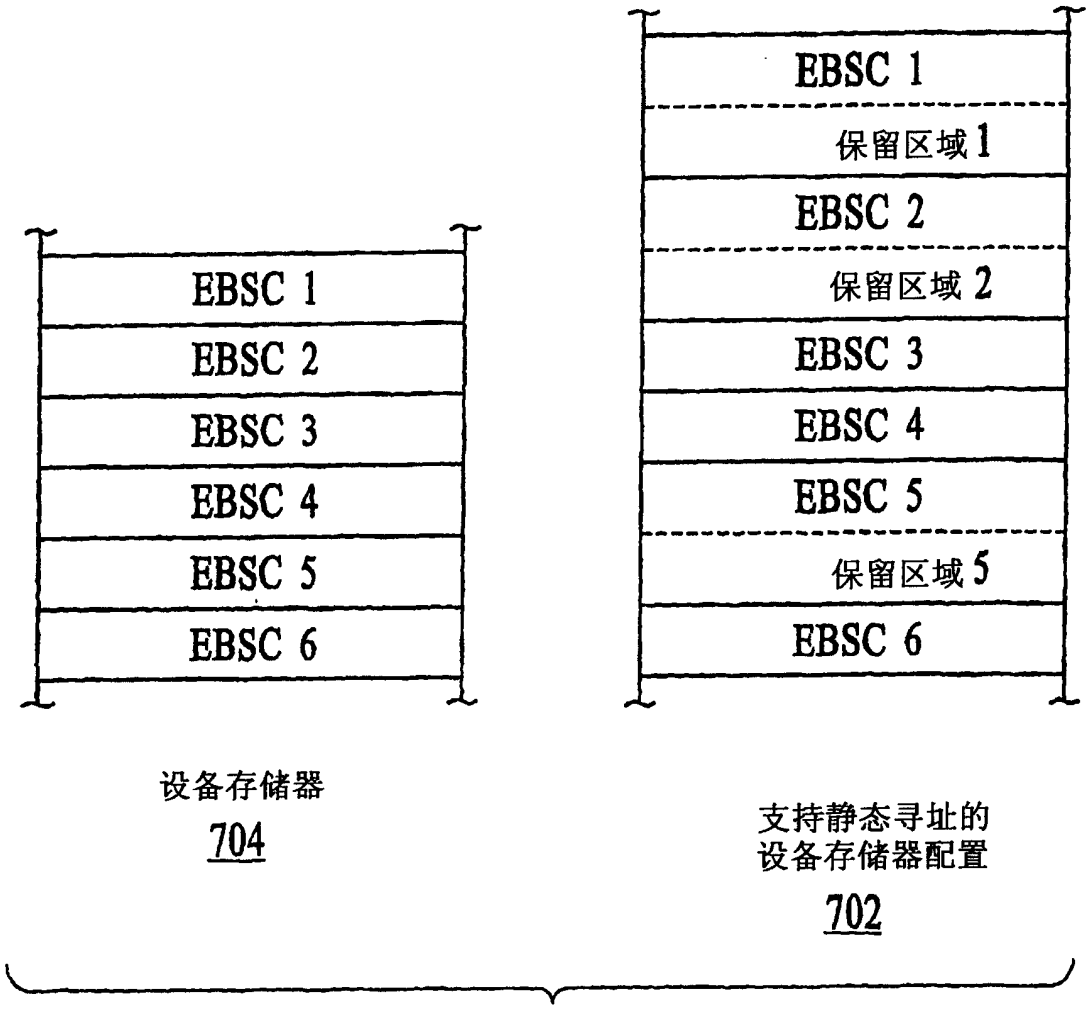

EBSC 1
EBSC 2
EBSC 3
EBSC 4
EBSC 5
EBSC 6
设备存储器
704
EBSC 1
保留区域1
EBSC 2
保留区域2
EBSC 3
EBSC 4
EBSC 5
保留区域5
EBSC 6
支持静态寻址的
设备存储器配置
702

图7

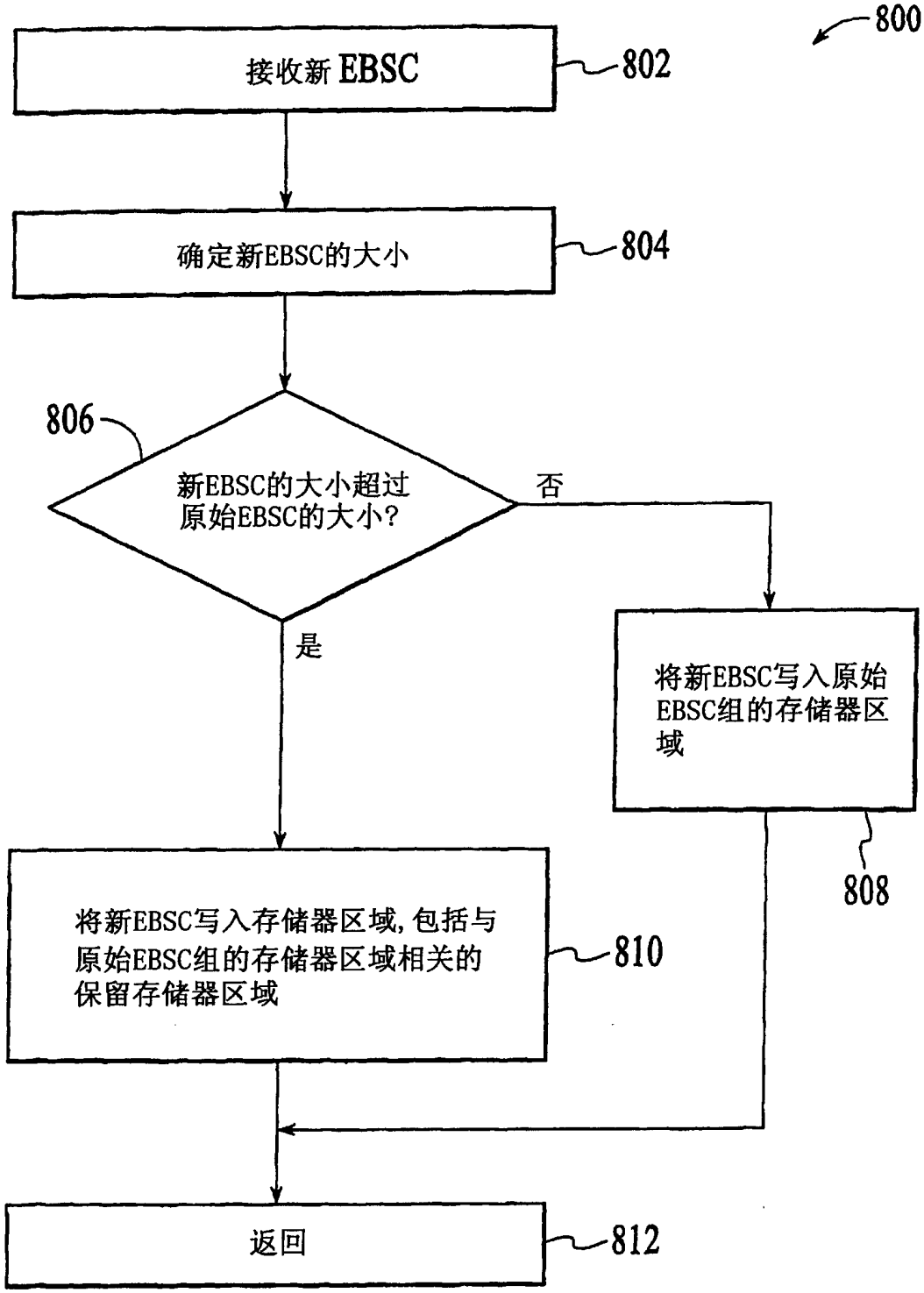

800
接收新 EBSC
802
确定新EBSC的大小
804
806
新EBSC的大小超过原始EBSC的大小?
否
是
将新EBSC写入原始EBSC组的存储器区域
808
将新EBSC写入存储器区域,包括与原始EBSC组的存储器区域相关的保留存储器区域
810
返回
812

图8

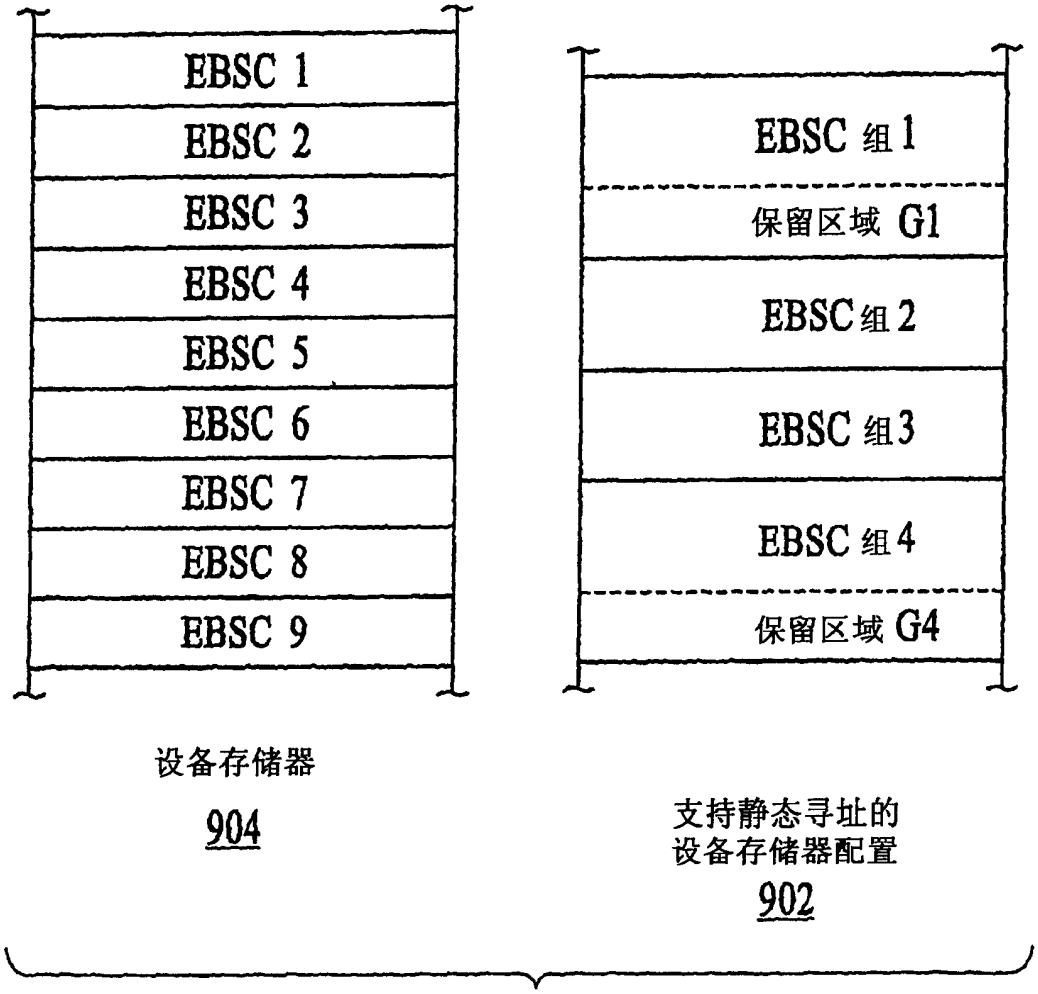

EBSC 1
EBSC 2
EBSC 3
EBSC 4
EBSC 5
EBSC 6
EBSC 7
EBSC 8
EBSC 9
EBSC 组1
保留区域 G1
EBSC组2
EBSC 组3
EBSC 组4
保留区域 G4
设备存储器
904
支持静态寻址的
设备存储器配置
902

图9

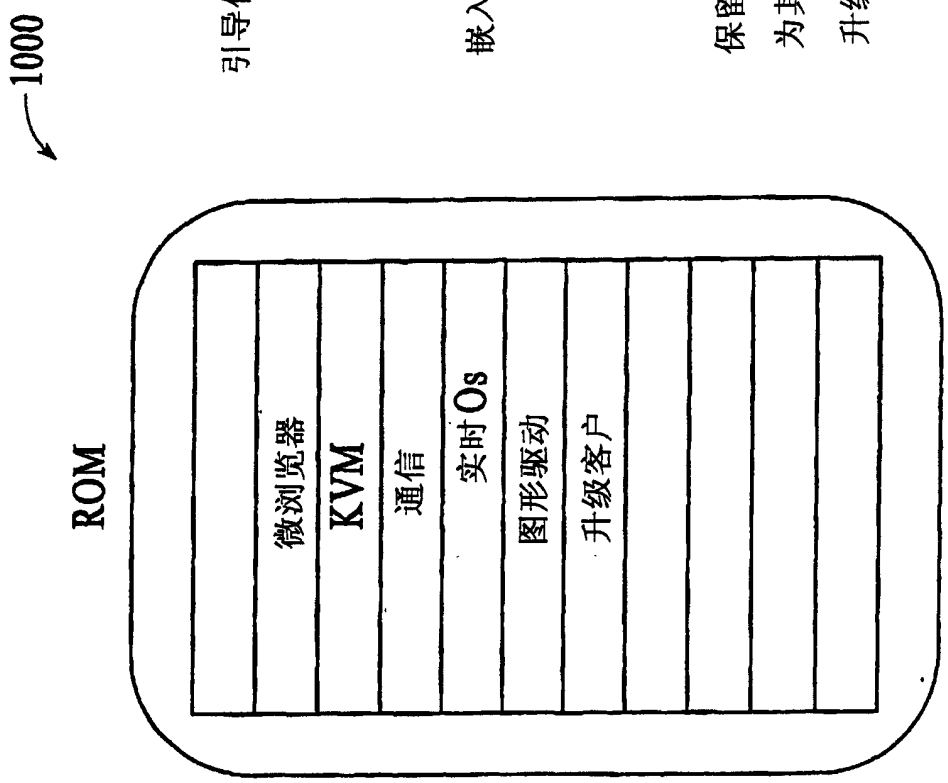

1000
ROM
微浏览器
KVM
通信
实时Os
图形驱动
升级客户
引导代码区域1002
嵌入式S/W 区域 1004
保留的工作区域1006
为其它目的保留的区域1010
升级客户设备参数区域1008

图10

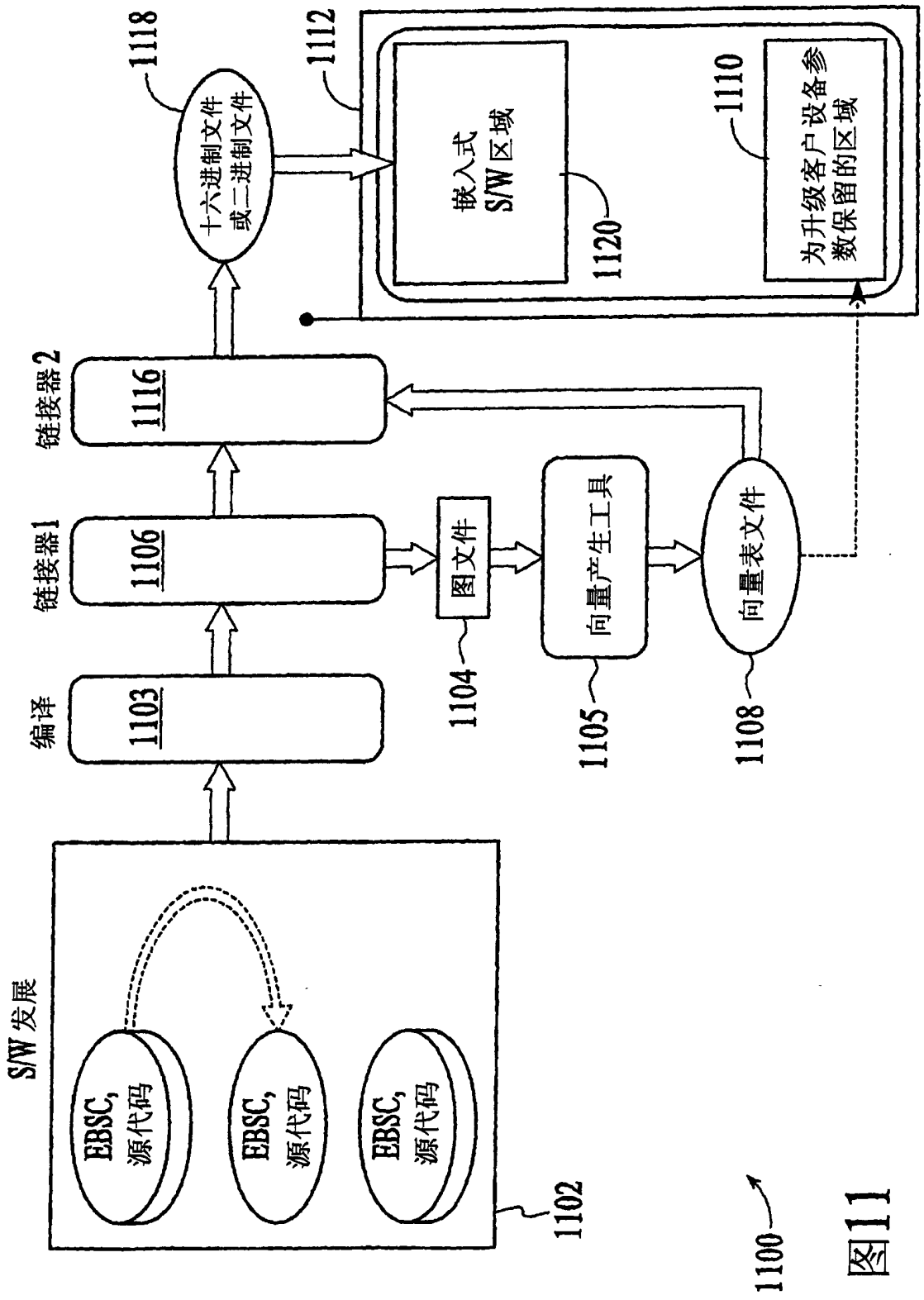

S/W 发展
EBSC, 源代码
EBSC, 源代码
EBSC, 源代码
1102
编译
1103
链接器1
1106
图文件
1104
向量产生工具
1105
向量表文件
1108
链接器2
1116
十六进制文件或二进制文件
1118
1112
嵌入式 S/W 区域
1120
为升级客户设备参数保留的区域
1110
1100

图11

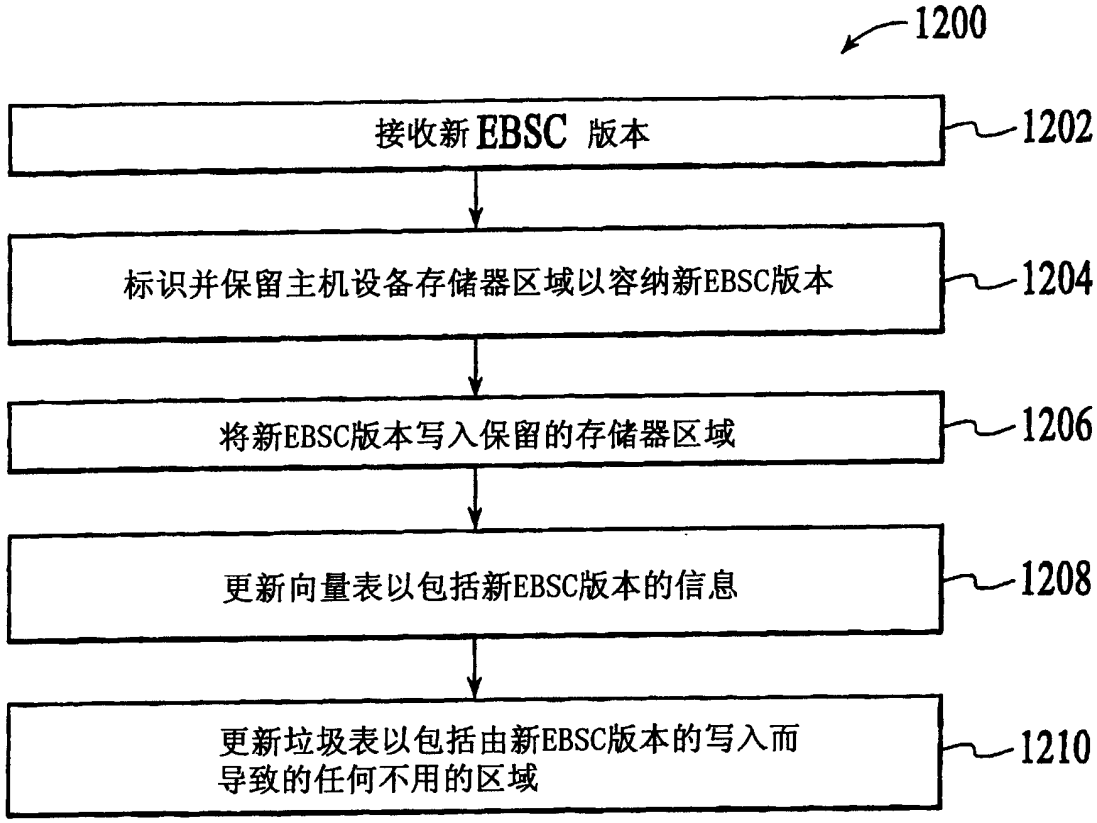

1200
接收新EBSC 版本
1202
标识并保留主机设备存储器区域以容纳新EBSC版本
1204
将新EBSC版本写入保留的存储器区域
1206
更新向量表以包括新EBSC版本的信息
1208
更新垃圾表以包括由新EBSC版本的写入而导致的任何不用的区域
1210

图12

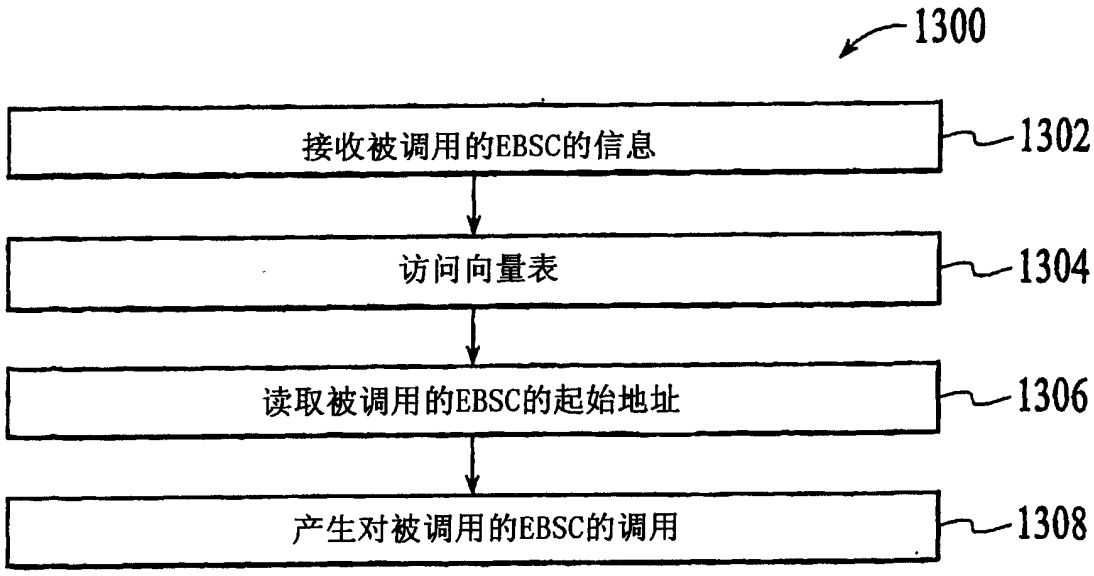

1300
接收被调用的EBSC的信息
1302
访问向量表
1304
读取被调用的EBSC的起始地址
1306
产生对被调用的EBSC的调用
1308

图13

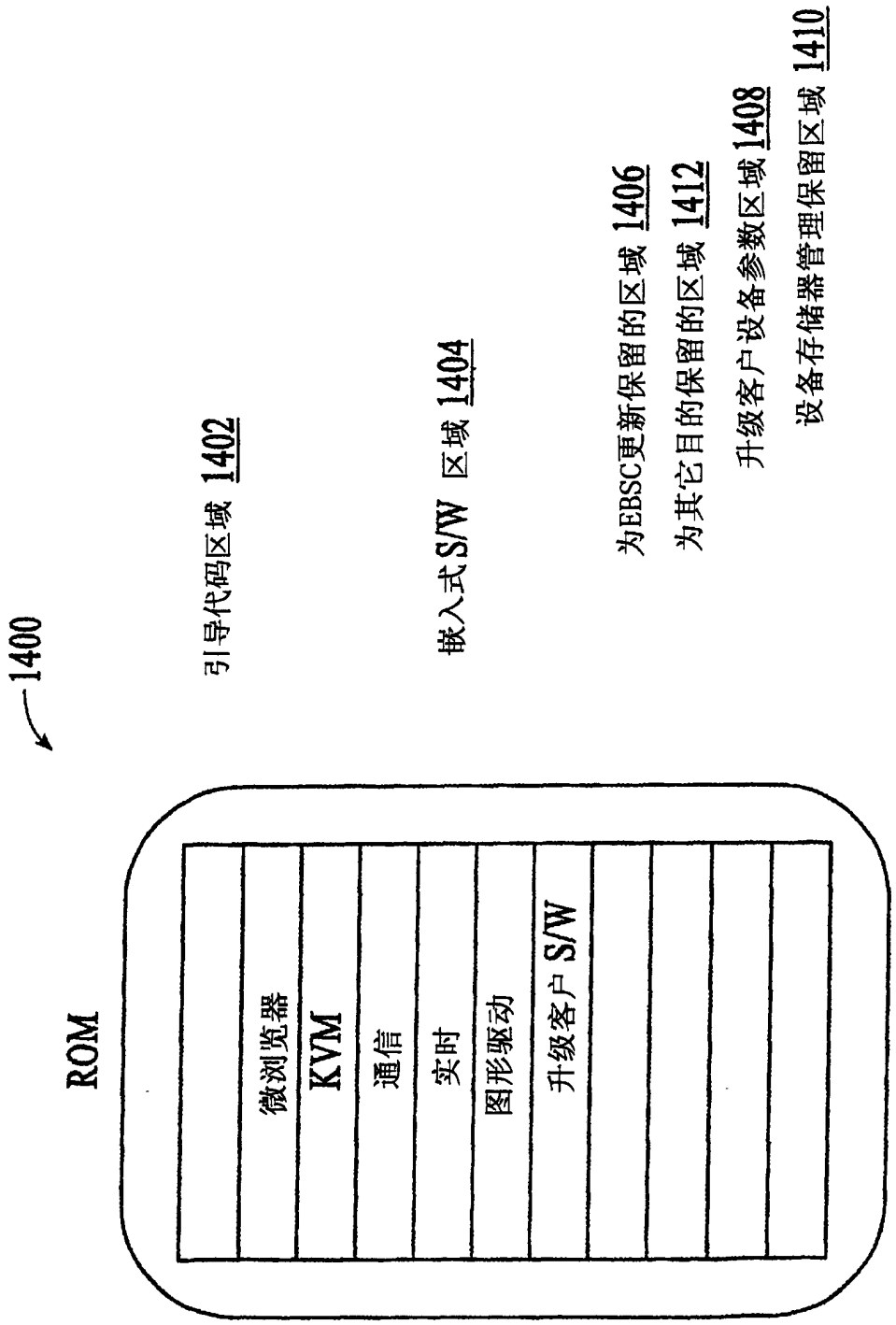

1400
ROM
引导代码区域 1402
嵌入式S/W 区域 1404
为EBSC更新保留的区域 1406
为其它目的保留的区域 1412
升级客户设备参数区域 1408
设备存储器管理保留区域 1410
微浏览器
KVM
通信
实时
图形驱动
升级客户 S/W

图14

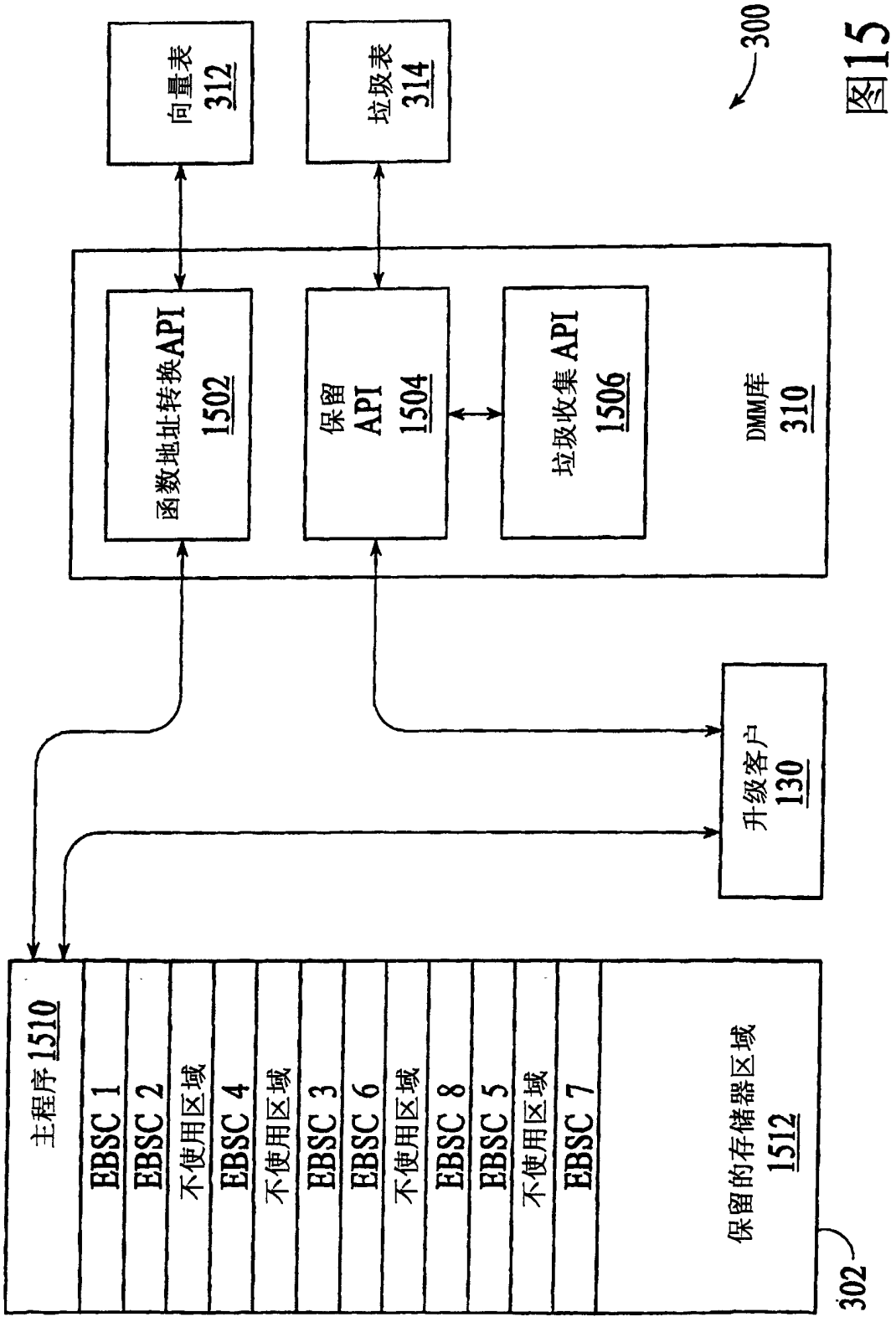

300
向量表 312
垃圾表 314
函数地址转换API 1502
保留 API 1504
垃圾收集 API 1506
DMM库 310
升级客户 130
主程序 1510
EBSC 1
EBSC 2
不使用区域
EBSC 4
不使用区域
EBSC 3
EBSC 6
不使用区域
EBSC 8
EBSC 5
不使用区域
EBSC 7
保留的存储器区域 1512
302

图15

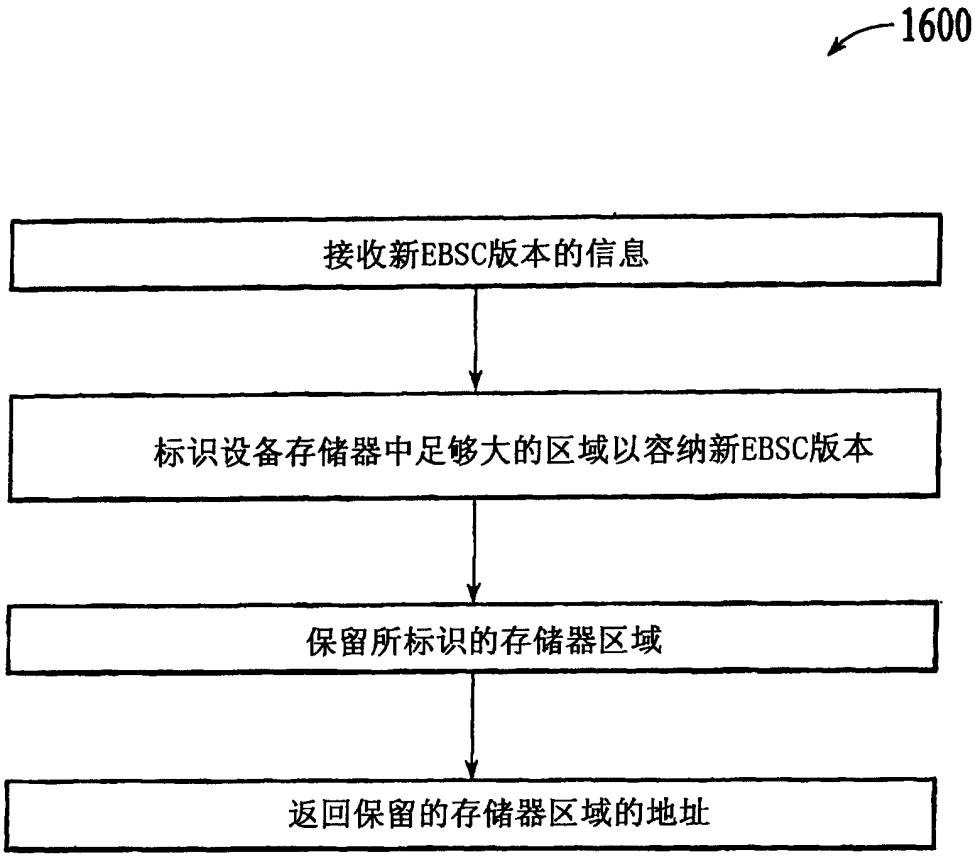

1600
接收新EBSC版本的信息
标识设备存储器中足够大的区域以容纳新EBSC版本
保留所标识的存储器区域
返回保留的存储器区域的地址

图16

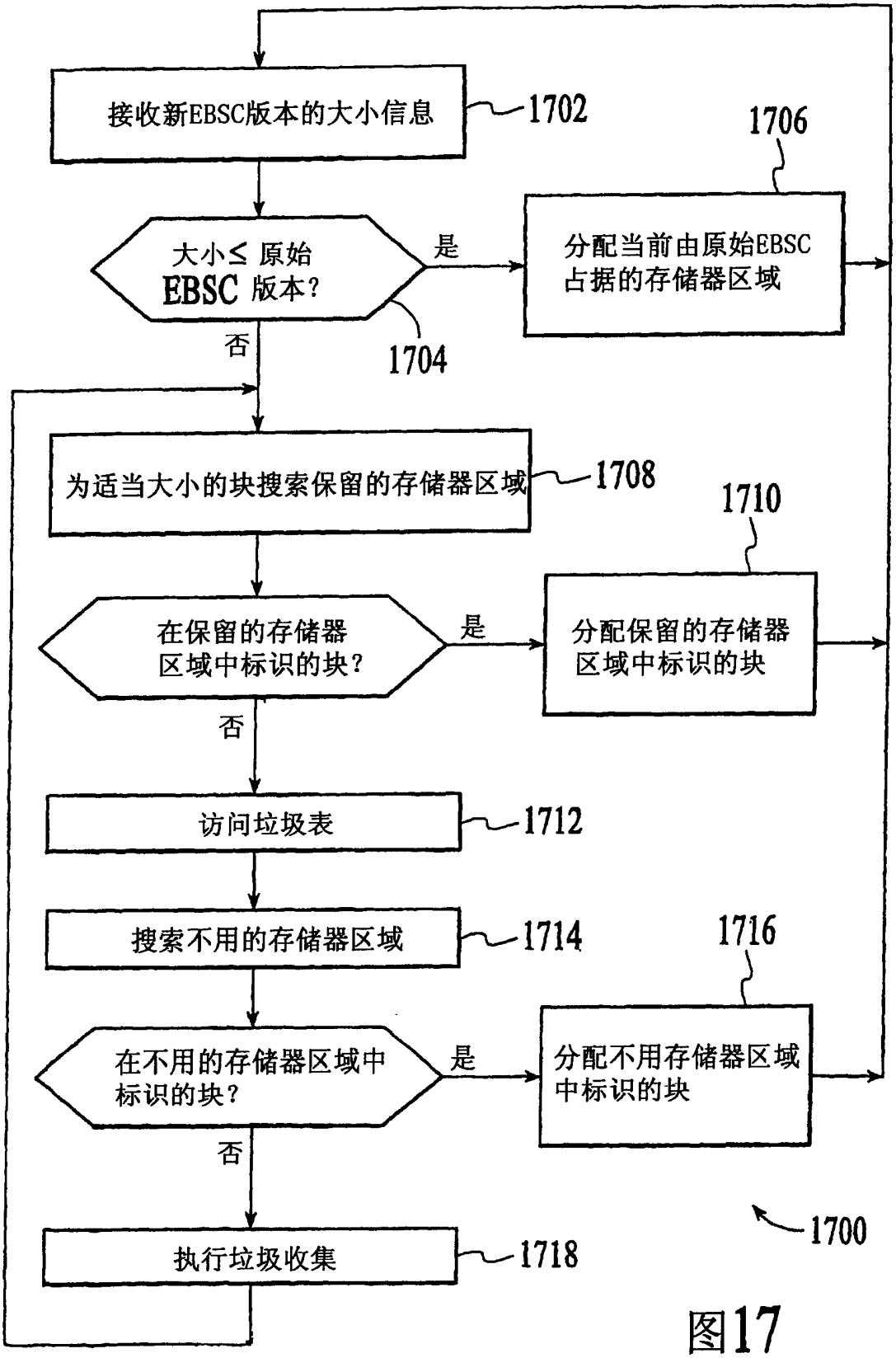

接收新EBSC版本的大小信息
1702
1706
大小≤ 原始 EBSC 版本?
是
分配当前由原始EBSC占据的存储器区域
否
1704
为适当大小的块搜索保留的存储器区域
1708
1710
在保留的存储器区域中标识的块?
是
分配保留的存储器区域中标识的块
否
访问垃圾表
1712
搜索不用的存储器区域
1714
1716
在不用的存储器区域中标识的块?
是
分配不用存储器区域中标识的块
否
1700
执行垃圾收集
1718

图17

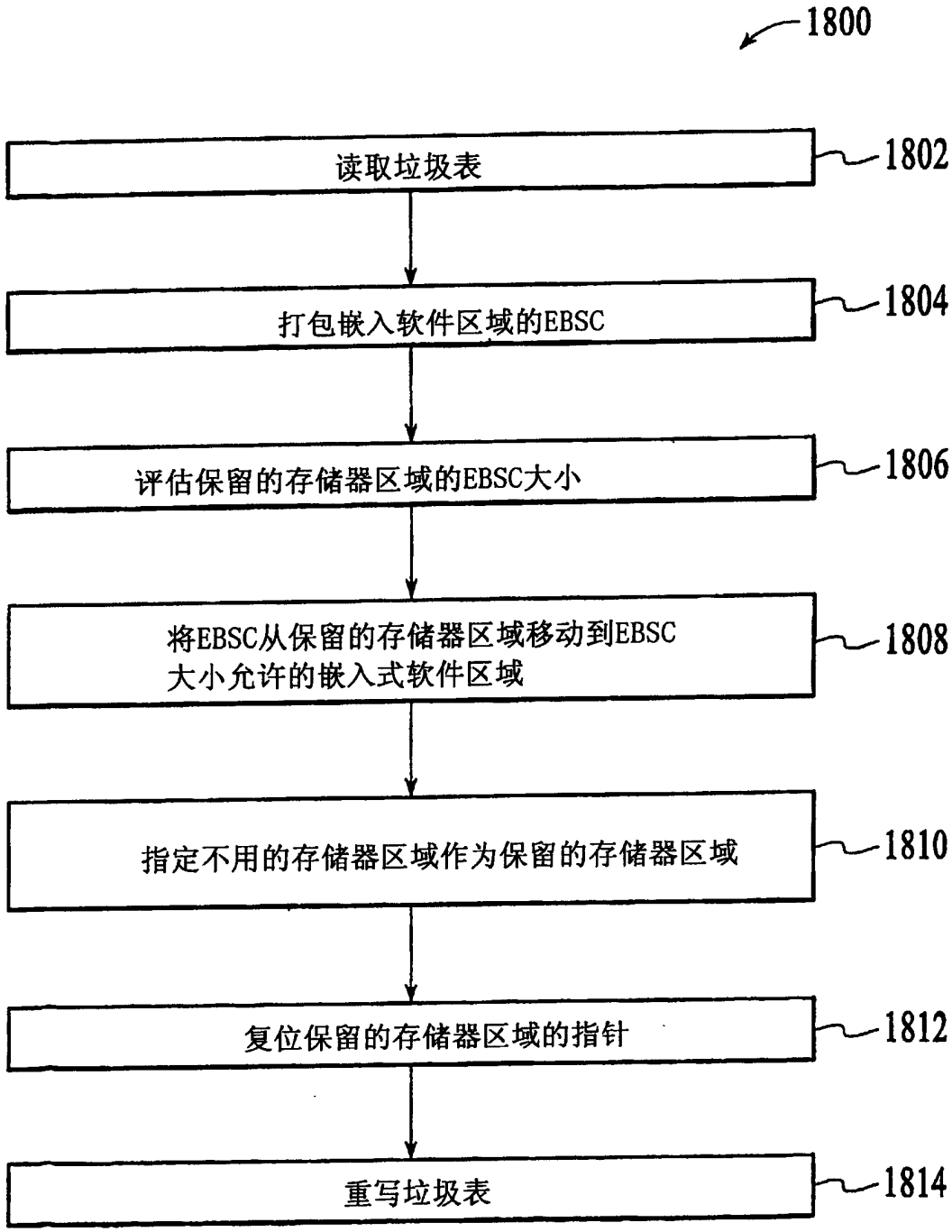

1800
读取垃圾表
1802
打包嵌入软件区域的EBSC
1804
评估保留的存储器区域的EBSC大小
1806
将EBSC从保留的存储器区域移动到EBSC
大小允许的嵌入式软件区域
1808
指定不用的存储器区域作为保留的存储器区域
1810
复位保留的存储器区域的指针
1812
重写垃圾表
1814

图18

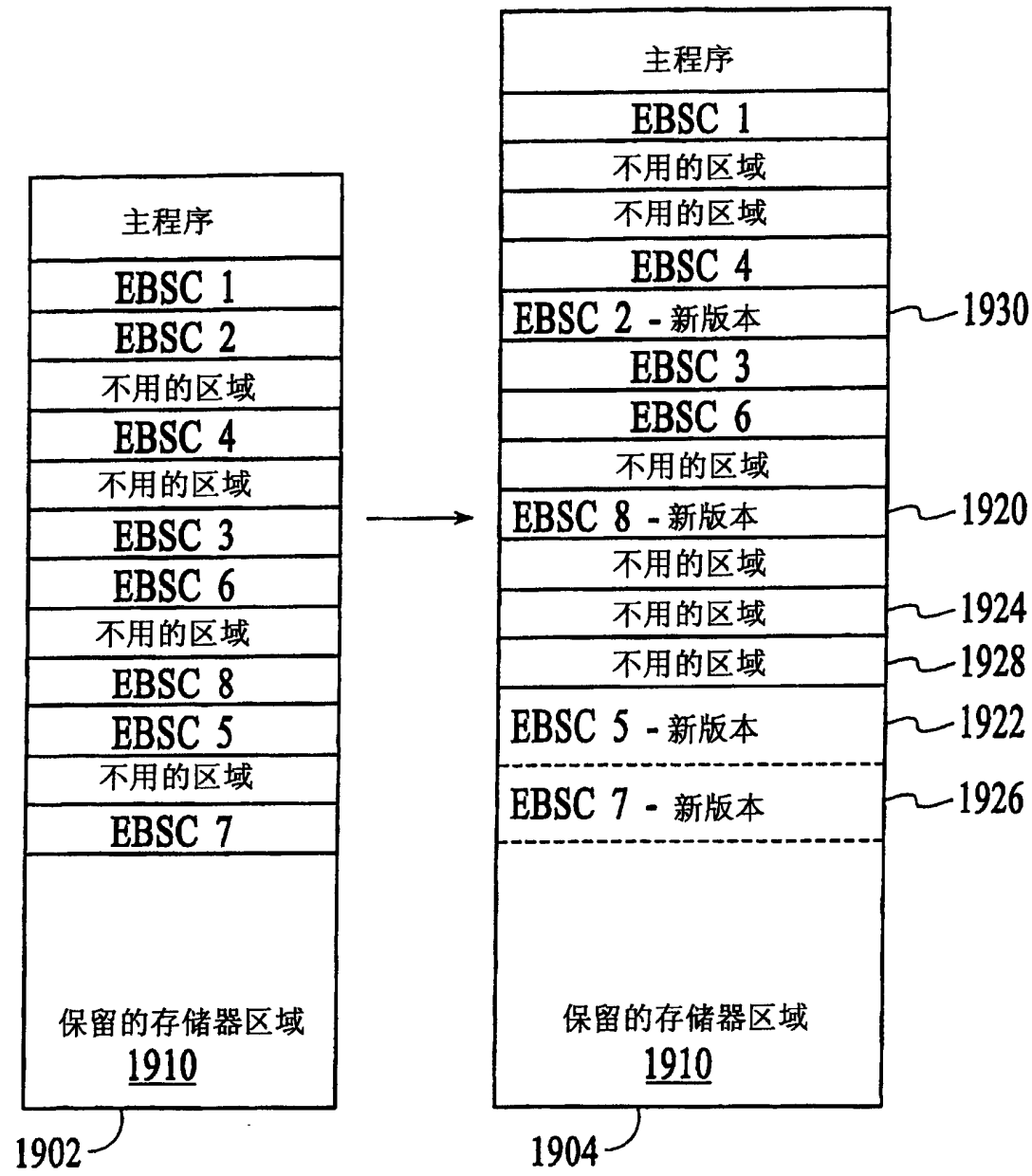

主程序
EBSC 1
EBSC 2
不用的区域
EBSC 4
不用的区域
EBSC 3
EBSC 6
不用的区域
EBSC 8
EBSC 5
不用的区域
EBSC 7
保留的存储器区域
1910
1902
主程序
EBSC 1
不用的区域
不用的区域
EBSC 4
EBSC 2 - 新版本
1930
EBSC 3
EBSC 6
不用的区域
EBSC 8 - 新版本
1920
不用的区域
不用的区域
1924
不用的区域
1928
EBSC 5 - 新版本
1922
EBSC 7 - 新版本
1926
保留的存储器区域
1910
1904

图19

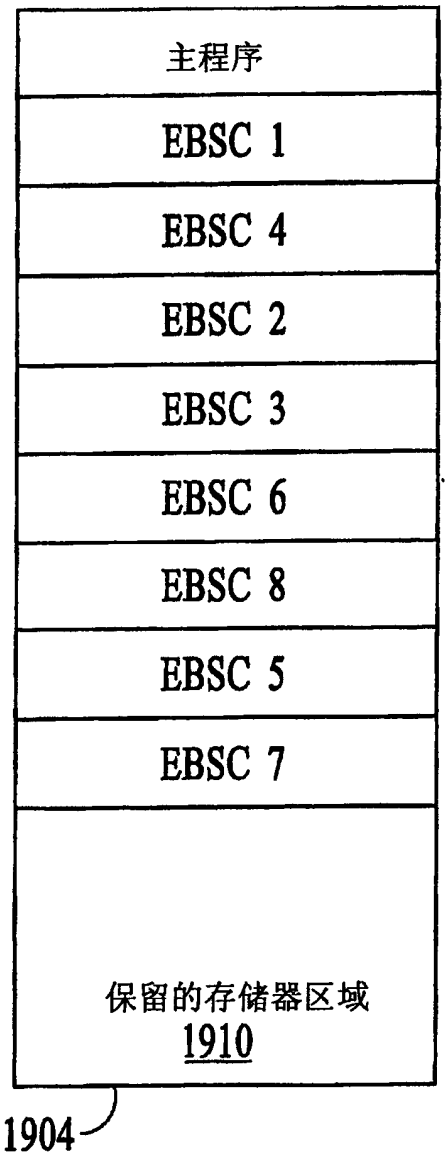

主程序
EBSC 1
EBSC 4
EBSC 2
EBSC 3
EBSC 6
EBSC 8
EBSC 5
EBSC 7
保留的存储器区域
1910
1904

图20

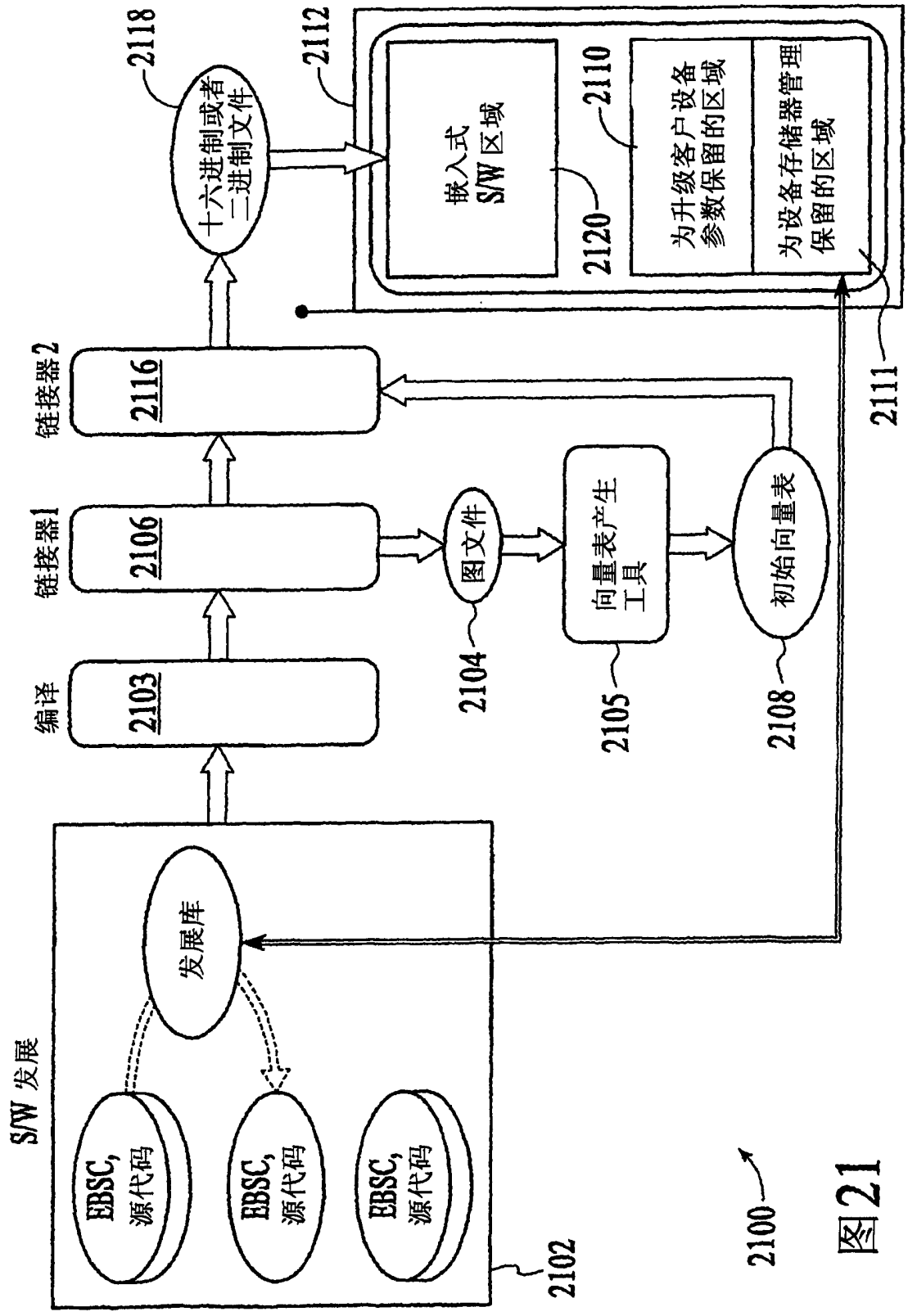

S/W 发展
发展库
EBSC,
源代码
EBSC,
源代码
EBSC,
源代码
2102
编译
2103
链接器1
2106
链接器2
2116
2118
十六进制或者
二进制文件
2112
嵌入式
S/W 区域
2110
2120
为升级客户设备
参数保留的区域
为设备存储器管理
保留的区域
2111
图文件
2104
向量表产生
工具
2105
初始向量表
2108
2100

图21