[19] 中华人民共和国国家知识产权局

[51] Int. Cl[7]
G06F 9/445
G06F 1/00

[12] 发明专利申请公开说明书

[21] 申请号 01807937.7

[43] 公开日 2003 年 6 月 4 日

[11] 公开号 CN 1422404A

[22] 申请日 2001.2.7 [21] 申请号 01807937.7
[30] 优先权
[32] 2000. 2.14 [33] US [31] 09/503,939
[86] 国际申请 PCT/US01/04079 2001.2.7
[87] 国际公布 WO01/61485 英 2001.8.23
[85] 进入国家阶段日期 2002.10.11
[71] 申请人 英特尔公司
地址 美国加利福尼亚州
[72] 发明人 T·威尔逊 M·克里斯托菲尔森
A·加夫肯 T·多德森
J·H·洛夫莱斯

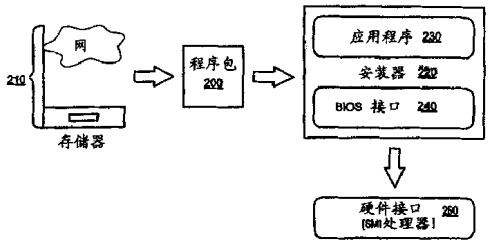
[74] 专利代理机构 中国专利代理(香港)有限公司
代理人 栾本生 王 勇

权利要求书 5 页 说明书 9 页 附图 8 页

[54] 发明名称 模块化的 BIOS 更新机制

[57] 摘要

一种模块化 BIOS 更新机制提供更新选项 ROM 的标准化方法，和提供计算机系统中视频和处理器微代码升级，不需要完全替换系统 BIOS。 MBU 机制提供几个优点。 首先，即使直接的 OEM 支持不可能被识别，新的特性和来自较早释放的 BIOS 故障可被发送到末端用户系统的一个已安装的数据库。 而且，BIOS 部件可被提供为被合法化的修订版集。 采用合法性矩阵，可容易地管理 BIOS 更新。 模块化 BIOS 更新在具有几个存储在单一固件内的独立 BIOS 的系统中特别有用。



ISSN1008-4274

1.一种更新系统BIOS的方法，包括:

接收具有BIOS程序包和有关的签名的数据目标，

利用存储在计算机上的公用密钥和签名认证BIOS程序包，和

如果认证成功，将BIOS程序包写到计算机系统上的固件中。

2.如权利要求1的方法，其中签名是由公用密钥-私用密钥对产生的，该公用密钥被存储在计算机上。

3.如权利要求1的方法，其中该接收包括从存储设备装载数据目标。

4.如权利要求1的方法，其中该接收包括从通信网接收数据目标。

5.如权利要求1的方法，其中该接收包括从计算机网接收数据目标。

6.如权利要求1的方法，在所述写以前还包括:

通过操作系统，将接收到的数据目标分段存储到系统存储器中。

建立识别系统存储器中每个分段的位置表，和

利用此表从各分段组装BIOS程序包。

7.一种对于处理器更新系统BIOS的方法，包括在重新启动处理器时。

确定是否系统存储器包含BIOS程序包，

认证BIOS程序包，和

在成功认证时，将BIOS程序包存入固件中。

8.如权利要求7的方法，还包括:

确定是否BIOS程序包被成功地存储在固件中。

如果是的，报告识别被成功地存储的BIOS程序包的成功标志。

9.一种计算机系统，包括:

一个处理器，

电气上连到处理器的固件，该固件包括:

存储第一系统BIOS的第一存储空间，该第一存储空间是只读存储器，

存储第二系统BIOS的第二存储空间，和将第二系统BIOS的部件与第一系统BIOS的部件关联的检索表。

10.如权利要求9的计算机系统，其中第一存储空间是存储系统

BIOS，和至少一个辅助的BIOS与识别BIOS的检索表。

11.一种发布对于被部署的计算机系统的系统 BIOS 的更新的方法，包括：

根据 BIOS 更新的内容，建立依据公用密钥-私用密钥对中的私用密钥的数字签名，

将BIOS更新和数字签名存储在存储媒体上，

对来自计算机的请求作出响应，将BIOS更新和数字签名传送到计算机。

12.如权利要求 10 的方法，其中传送包括传送来自存储设备的数据目标。

13.如权利要求 10 的方法，其中传送包括传送来自通信网的数据目标。

14.如权利要求 10 的方法，其中传送包括传送来自计算机网的数据目标。

15.如权利要求10的方法，其中公用密钥-私用密钥对被存储在计算机中。

16.一种BIOS处理方法，包括：

执行来自常驻的存储器空间的系统BIOS，

依据以下条件执行辅助的BIOS：

确定是否在可变更的存储器空间中存在辅助的BIOS，

如果在可变更的部分中不存在辅助的BIOS，执行来自常驻的存储器空间的辅助的BIOS。

17.如权利要求16的方法，还包括：

如果在可变更的部分中存在辅助BIOS，执行在该可变更部分中的辅助BIOS。

18.如权利要求 16 的方法，还包括，确定是否预先确定的用户命令已被输入，如果预先确定的用户命令未被输入，执行来自可变更部分的辅助的BIOS。

19.如权利要求16的方法，还包括：

将来自可变更的部分的辅助的BIOS解压，并

执行被解压的辅助的BIOS。

20.一种辅助的BIOS的处理方法，包括：

确定是否在固件的增强空间中存在辅助的BIOS程序包，

如果辅助的BIOS程序包是存在的，确定是否预先确定的用户命令已被输入，

如果预先确定的用户命令未被输入，执行来自增强空间的辅助的BIOS程序包，

否则，执行来自固件的常驻空间的辅助的BIOS。

21.如权利要求20的方法，还包括：

将来自可变更部分的辅助的BIOS解压，并

执行被解压的辅助的BIOS。

22.一种辅助的BIOS的处理方法，包括：

确定是否在固件的增强空间中存在辅助的BIOS程序包，

如果辅助的BIOS程序包是存在的，确定是否预先确定的标志已被设置在固件中，

如果预先确定的用户命令已被设置，执行来自增强空间的辅助的BIOS程序包，

否则，执行来自固件的常驻空间的辅助的BIOS。

23.如权利要求22的方法，还包括：

将来自可变更的部分的辅助的BIOS解压，并

执行被解压的辅助的BIOS。

24.一种辅助BIOS的处理方法，包括：

确定是否在固件的增强空间中存在辅助的BIOS程序包，

如果在增强空间中存在辅助的BIOS程序包，将辅助的BIOS程序包解压，并

执行辅助的BIOS程序包

25.如权利要求24的辅助BIOS处理方法，还包括搜索用于与辅助BIOS程序包有关的解压器的存储器，如果解压器未被找到。执行来自固件的常驻空间的第二辅助BIOS程序包。

26.一种视频BIOS处理方法，包括：

确定是否在存储器系统的可变更的固件部分中存在视频BIOS，

如果在可变更的部分中不存在视频BIOS，执行在存储器系统的不可变更固件部分中的视频BIOS。

27.如权利要求26的方法，其中确定和执行步骤在系统BIOS执行

期间完成。

28.如权利要求 26 的方法，还包括，如果在可变更的部分中存在视频 BIOS，执行可变更部分中的视频 BIOS。

29.如权利要求 26 的方法，还包括，确定是否预先确定的用户命令已被输入，如果没有预先确定的用户命令已被输入，执行来自可变更部分的视频 BIOS。

30.如权利要求 26 的方法，还包括:

将来自可变更部分的视频 BIOS 解压和

执行被解压的视频 BIOS。

31.一种视频 BIOS 处理方法，包括:

确定是否视频 BIOS 程序包是存在于固件的增强空间中，

如果视频 BIOS 程序包是存在的，确定预先确定的用户命令是否已被输入，

如果预先确定的用户命令已被输入，执行来自增强空间的视频 BIOS 程序包，

否则，执行来自固件的常驻空间的视频 BIOS。

32.如权利要求 31 的方法，还包括:

将来自可变更的部分的视频 BIOS 解压和

执行被解压的视频 BIOS。

33.一种视频 BIOS 处理方法，包括:

确定是否视频 BIOS 程序包是存在于固件的增强空间中，

如果视频 BIOS 程序包是存在的，确定是否预先确定的标志已被设置在固件中，

如果预先确定的用户命令已被设置，执行来自增强空间的视频 BIOS 程序包。

否则，执行来自固件的常驻空间的视频 BIOS。

34.如权利要求 34 的方法，还包括:

将来自可变更部分的视频 BIOS 解压和

执行被解压的视频 BIOS。

35.一种视频 BIOS 处理方法，包括:

确定是否视频 BIOS 程序包是存在于固件的增强空间中，

如果视频 BIOS 程序包是存在于增强空间中，将视频 BIOS 程序包

解压，和

执行视频 BIOS 程序包。

36. 如权利要求 35 的视频 BIOS 处理方法，还包括搜索用于视频 BIOS 程序包的解压器的存储器，如果解压器未被找到，执行来自固件的常驻空间的第二视频 BIOS 程序包。

37. 一种其上已存储程序指令的计算机可读的媒体，当被处理器执行时，使处理器：

接收具有 BIOS 程序包和有关的签名的数据目标，

利用存储在计算机上的公用密钥和签名来认证 BIOS 程序包，和

如果认证成功，将 BIOS 程序包写到计算机系统上的固件中。

38. 如权利要求 37 的媒体，其中该签名由公用-私用密钥对产生，公用密钥可被计算机从存储器读出。

39. 如权利要求 37 的媒体，其中该接收包括从存储设备装载数据目标。

40. 如权利要求 37 的媒体，其中该接收包括从通信网接收数据目标。

41. 如权利要求 37 的媒体，其中该接收包括从计算机网接收数据目标。

42. 如权利要求 37 的媒体，在所述写以前还包括：

通过操作系统，将接收到的数据目标分段存入系统存储器中，

建立识别在系统存储器中每个分段位置的表，和

利用此表从各分段组装该 BIOS 程序包。

模块化的 BIOS 更新机制

背景

本发明的实施方案在模块化的基础上提供用于更新一种现代计算机系统的系统 BIOS 部件的技术。

因为计算机系统的功能部件被越来越多地集成到单一的集成电路中，以前可能一直存储在被隔离的选件 ROM 中的多个 BIOS 图象也可被集成到一个单一的固件系统中。这种较大规模的集成产生一种能够更新存储在单一固件中的系统 BIOS 或部件而不需要更换整个系统的 BIOS 的需要。

例如，Intel 公司，本发明的受让者正在设计一种将处理器，图形控制器和存储器控制器的功能合并的单个集成电路。这样，该集成电路可以与既包括管理通过系统的输入/输出事务的系统 BIOS，又可包括用于图形控制功能的视频 BIOS 的固件通信。可以与系统 BIOS 的更新无关地公开对视频 BIOS 的升级。因此，在系统 BIOS 的领域中有一种允许对存储媒体中的 BIOS 进行模块更新的需要。例如，更新视频 BIOS 而不干扰存储媒体中的整个系统的 BIOS 可能是有利的。

附图简述

图 1 示出一种依据本发明的实施方案的系统 BIOS。

图 2 是依据本发明的一种实施方案的使用模型。

图 3 是用作说明一种模块化 BIOS 更新（“MBU”）过程的实施方案和在安装 BIOS 程序包期间 BIOS 和驱动器之间交互作用的高等级流程图。

图 4 示出一种依据本发明的实施方案更新固件的方法。

图 5 是一种用作说明各分段的存储器位置的存储器映象。

图 6 是一种用作说明重新组装的 BIOS 程序包的存储器映象。

图 7 示出一种依据本发明的实施方案更新视频 BIOS 的方法。

图 8 示出一种依据本发明的实施方案的样本 MBU 程序包。

详述

本发明的实施方案提供一种 MBU 机制——种更新选项 ROM 并提供视频和处理器微代码而不需要完全替换系统 BIOS 的标准化方法。MBU 机

制提供几个优点。首先，即使直接的 OEM 支持不可能被识别，新的特性和 BIOS 故障修理可被发送到末端用户系统已安装的数据库中。BIOS 部件也可被提供为一套合法的修订版。利用对合法矩阵的再分类，可容易地管理 BIOS 更新。

图 1 示出一个存储器空间 100，存储依据本发明的优选实施方案的系统 BIOS 110。存储器空间 100 可包括第一存储器空间 120 和第二存储器空间 130，第一存储器空间可存储核心 BIOS 140 和一个或多个辅助的 BIOS。例如，图 1 示出存储在第一存储器空间 120 中的核心 BIOS 140 和视频 BIOS 150。第一存储器空间 120 可被提供在固件中。另一种方案是可被提供在通常的 ROM 存储器中。依据一种实施方案，第一存储器空间 120 不需要提供写数据到空间 120。在操作期间，附加的 BIOS 可被写入固件 100。不管什么样的附加 BIOS 被提供在固件 100 中，存储在第一存储器空间 120 中的那些 BIOS 被指望永久地留在其中。因此，第一存储器空间 120 此时可认为是“常驻的空间”。

第二存储器空间 130 可存储附加的 BIOS。因此，它可允许写数据到此。第二存储器空间 130 可存储对常驻的系统 BIOS 的增强内容。因而，第二存储器空间 130 此时可认为是“增强的空间”。

增强的空间 130 可提供存储 BIOS 部件，以便补充或替换存储在常驻的空间 120 中的那些 BIOS 部件。举例来说，增强的空间 130 可存储：

- 微代码更新，校正在代理器中的微代码差错，
- BIOS 更新或修补，修理在常驻的 BIOS 空间中的故障，和
- BIOS 模块，在操作期间替换常驻的空间中的 BIOS，和
- 可更新的视频驱动器参数数据表，提供配置相同驻动器参数表，以便通过多重硅的修订版使用。

规定在常驻的空间 120 中所提供的任何 BIOS 部件在制造以后将不再改变。然而，更新可被存储在增强空间 130 中。增强空间 130 也可存储索引表，映射增强空间 130 和常驻空间 120，中的 BIOS 部件。这样，分配表 130 可包含指针，识别来自常驻空间 120 的哪个系统 BIOS 或系统部件已被增强空间的部件所替代。

图 2 是依据本发明的实施方案的一种使用模型，依据该实施方案，计算机系统可利用 BIOS 的增强功能自主地对自己编程。图 2 示出“MBU 程序包”200，一个可包括一个或多个 BIOS 程序包（未示出）的数据

目标，MBU 程序包 200 可通过输入/输出设备，如电的，磁的或光的存储设备，或通过通信接口，如为内部网，因特网或无线计算机网（合在一起表示为 210）所提供的装置输入到计算机系统。

该计算机系统可包括一个安装器 220。安装器 220 可包括一个应用程序 230 和一个 BIOS 接口驱动器 240，被设计用于依据在此所描述的实施方案处理 MBU 程序包 200。在一种实施方案中，安装器 220 可与计算机网接合，如因特网，以便搜索和下载最近的 MBU 程序包供计算机系统使用。一旦 MBU 程序包 200 被接收到，安装器 220 可将 MBU 程序包拆成离散的 BIOS 程序包并调用可能对更新固件必要的任何硬件接口 250。

依据一种实施方案。BIOS 程序包不仅可包括对 BIOS 自己的更新，也可包括关联的合法性和修订版的信息。安装器 220 可在合法性和修订版的比较方面起作用。对于修订版，在安装以前，安装器 220 可以将 BIOS 更新的修订版信息与可能与以前存储在常驻空间 110 中（图 1）和存储在增强空间 120（图 1）中的系统 BIOS 关联的修订版信息（如果有的话）作比较。在这个实施方案中，如果比较结果确定在程序包中所提供的 BIOS 更新并不比以前所存储的版本更现代时，则安装器 220 可以终止安装过程。

依据实施方案，安装器 220 可将 BIOS 程序包存储在系统存储器中供安装。安装器 220 也设置某种标志在系统存储器中以指明该系统存储器存储 BIOS 程序包。安装器 220 也可使可依附于固件，可防止在正常操作期间将数据写入固件的某种硬件保密锁不能工作。典型情况下，可以利用对系统管理中断（“SMI”）处理器 250 的通常调用来使硬件保密锁不能工作。SMI 处理器 250 可使计算机操作系统不能工作并允许对 BIOS 更新。

一旦安装器 220 已将程序包存入系统存储器，安装过程终止。在安装过程结束时，安装器 220 可按这样一种方式使处理器复位，即处理器重新启动但系统存储器的内容被保持。例如，这可通过对许多 Intel 处理器断言 INIT#来实现。

图 3 是用作说明 MBU 过程 300 的一种实施方案的高等级流程图。图 3 示出在几个执行阶段中的操作：操作系统（“OS”）运行时间 310，OS 重新启动 320，POST330 和 OS 装载 340。在安装器应用程序 230 被

启动后，过程可在 START 点开始。当安装器应用程序 230 启动时，可挂接 MBU 驱动器 240（方框 301）、作为响应，驱动器 240 可以确定被 MBU SMI 处理器支持的接口修订版（方框 302）。如果修订版号码被驱动器辨认出，驱动器可以利用一个建立和返回在 MBU 过程 300 期间对所有调用使用的处理的命令挂接到 SMI 处理器。在初始化完成以后，驱动器 240 可以返回对安装应用程序 230 的控制。如果存在已完成的安装，该安装应用程序 230 可以对程序包的分段建立任何必要的指针（在此已讨论过）并可调用 MBU 驱动器的安装入口点（方框 303）。MBU 驱动器 240 可以保存此信息并返回安装应用程序 230（方框 304）。在安装过程中的这一点上，安装应用程序 230 可以退出，留下挂在 MBU 驱动器更新队列中的安装请求（方框 305，306）。MBU 驱动器 240 可以停留在空间状态中直到在断开期间 OS 发送关闭消息到 MBU 驱动器 240 为止。

当操作系统准备重新启动时，典型情况下，它们发送 MSG_CLOSE 消息到通知目录上的所有驱动器.MBU 驱动器 240 可被包括在这个目录上。在重新启动期间，MBU 驱动器 240 可以接收 MSG_CLOSE 消息并查看它的对于 MBU 更新的未决的更新表（方框 311）。如果存在已完成的更新，MBU 驱动器 240 可在系统存储器中建立一个物理缓存器，可以将更新的程序包从文件装载到主存储器中，并可在存储器中设置标志，通知 BIOS，更新被完成（方框 312）。然后 MBU 驱动器 240 可以释放它与 SMI 处理器 250 的处理并终止（方框 313，314）。然后，OS 可按通常方式关闭。

一旦 OS 已被关闭，它可产生对处理器的 INIT#·INIT#未将存储器控制器复位，保持系统存储器原封不动。INIT#将处理器复位到引导向量，并将存储器的锁定方案复位到允许编程的状态。在图 3 中，作为 POST 阶段 320 示出。在 POST 阶段 320 期间，系统 BIOS 可从系统存储器检索程序包，认证程序包并将程序包写入增强空间（方框 321）。一旦更新完成，系统 BIOS 可以发布系统复位，并开始正常的 POST 过程（方框 322，323）。一旦系统 BIOS 开始装载 OS，可将控制递交到图 3 的 OS 装载阶段 330。

在 OS 装载阶段 330 中，OS 将驱动器装入“驱动器启动”目录（方框 331）。在此目录中的驱动器之一是 MBU 驱动器 240。当 MBU 驱动器

240 装载时，可以挂接到 SMI 处理器 250，并确定被处理器 250 支持的 MBU-SMI 接口的修订版（方框 322）。如果 MBU 驱动器 240 辨认出修订版号码，可以通过建立对驱动器的处理再次尝试挂接到 SMI 处理器 250，供将来的调用使用（方框 323）。然后 MBU 驱动器 240 可以确定最近更新的状态（方框 324）。MBU 驱动器 240 可以保存此状态并留在空间状态直到被安装器应用程序 230 调用为止（方框 335）。更新过程完成。

图 4 示出在依据本发明的实施方案的图 3 的 POST 阶段期间一种更新固件的方法 400。在 BIOS 被重新启动以后（方框 410），BIOS 将经历启动步骤。在启动步骤期间的某个点上，最好在常驻空间中任何辅助 BIOS 的部件被执行以前，BIOS 可以确定是否 BIOS 指明存在 MBU 程序包（主框 420）。如果不是，BIOS 可以退出方法 400 并继续启动步骤，最终使整个系统复位（方框 430）。

然而，如果存储器包含 BIOS 程序包存在的标识符，BIOS 试图认证 MBU 程序包（方框 440）。BIOS 确定是否认证是成功的和 MBU 程序包是有效的（方框 450）。如果不是，如 MBU 程序包是无效的，BIOS 返回 MBU 授权失败（方框 460）并退出方法（方框 430）。

如果认证合法性成功，BIOS 开始将 MBU 程序包写入增强空间（方框 470）。在写结束时，BIOS 确定是否更新成功（方框 480）。如果不是，BIOS 返回 MBU 更新失败（方框 490）并从 BIOS 清除 MBU 更新标志（方框 500）。否则，BIOS 可以设置增强标志（方框 510），并返回指明成功存储的状态标志（方框 520）。一旦增强表已被修正，方法可终止。

在终止方法 400 时，BIOS 可将整个系统复位。系统复位不仅重新启动 BIOS 本身，而且也清除系统存储器。当系统复位时，MBU 程序包可按其内容所确定的方式调用。

程序包认证

正如所指出的那样，MBU 更新过程可包括程序包认证。依据实施方案，MBU 程序包可以包含唯一地识别 MBU 程序包源的认证信息。例如，认证信息可以依据公用-私用密钥对产生。公用密钥可被 BIOS 存储在常驻空间 110 中（图 1）。新接收到的 MBU 程序包可用私用密钥作标志。利用公用密钥，处理器可以查看 MBU 程序包的内容和伴随着 MBU 程序

包的签名以便认证程序包。成功指明MBU程序包被一个真正的源颁发，失败可以指明MBU程序包并不是被一个真正的源颁发的，MBU程序包在颁发以后不知什么原因遭损坏，或发生某种其他的差错。如果认证失败，BIOS可以终止MBU更新过程。

依据实施方案，公用密钥-私用密钥可以基于众所周知的签名算法，如RSA或DSA签名系统。其他的签名算法是已知的。

在实施期间，MBU程序包的颁发者期望保持对私用密钥的控制。对照MBU程序包所包含的签名的成功认证应该表明MBU程序包来自一个授权的源。

缓存器分段表

通常在计算机系统内，存储器空间可被分配成预先确定的单元。例如，许多计算机系统的操作系统通常分配预先确定规模的“页面”，比如说4或8千字节。MBU程序包并不受这些系统页面规模的限制。依据实施方案，MBU程序包可以跨几个页面。这样，当被操作系统240的安装应用程序存储在主存储器中时（图2），MBU程序包的部分可被存储在整个系统存储器的几个不连续的页面中。

在实施方案中，安装应用程序220可与操作系统接合，用以分配页面到MBU程序包的分段上。安装应用程序220也可建立分段缓冲器表，识别可以找到每个分段的存储器位置。举例来说，以下的表1表示可以包含在一个这样的分段缓存器表中的信息：

分段基地址	缓存器号码No.	分段No.	分段规模	下一个分段的地址
0x80_0000	0x0	0x0	8180 (0x1FF4)	0x12_0000
0x12_0000	0x0	0x1	8180 (0x1FF4)	0x09_2000
0x09_2000	0x0	0x2	8180 (0x1FF4)	0x64_6000
0x64_6000	0x0	0x3	1583 (0x62F)	0x4C_A000
0x4C_A000	0x1	0x0	256 (0x100)	0x00_0000

表1

如本例中所示，对于每个BIOS程序包，分段缓存器表可以跟踪多个“缓存器”。第一缓存器包含BIOS代码本身，第二缓存器包含与BIOS代码关联的数字签名。图5是一个存储器映射图600，用作说明表1中所列的分段610-650的存储器位置。

依据实施方案，缓存器分段表可以识别每个缓存器的每个分段，对于分段的缓存器和分段号码，表示在主存储器中分段位置的基地址，表示分段的规模的规模标识符和下一个分段的地址。这种信息可在 INIT 以前提供系统存储器中预先确定的位置。因此，当方法 400 被调用时，处理器可以确定是否缓存器分段表存在，如果是的，从在表（图 3，方框 520）中所识别的缓存器分段组装 MBU 程序包。图 6 是一个存储器映射图，用作说明被重新组装的 BIOS 程序包。然后，处理器可以继续 MBU 更新。

流后处理（Post Flow Processing）

为了支持 MBU 更新，通常的系统 BIOS 可以包括附加的功能，允许安装新处理器微代码更新，从 MBU 块装载已更新的视频 BIOS，检查视频旁路热键，检查指明流程选项的 CMOS 位，和从 MBU 块提出选项 ROM 或 BIOS 程序包。往下描述这些特性。图 7 是依据本发明实施方案的这种过程 800 的流程图。

当准备对视频 BIOS 初始化时，在试图执行常驻的视频 BIOS 图象以前，BIOS 可以为了用于系统的特定的图形设备的现有视频 BIOS 更新扫描 MBU 块。在开始扫描 MBU 块时，系统 BIOS 可以检查，肯定 MBU 块是有效的（方框 810）。这可通过检查 MBU 块的头段 ID 迅速地完成。如果 MBU 块是非法的，常驻的选项 ROM 可被执行（方框 820），另外 BIOS 检查预先确定的旁路热键的按压（方框 830）。

如果热键被按压，它表示常驻视频 BIOS 不应该被执行的一个用户命令（方框 820）。然而，按压旁路热键，并不使系统 BIOS 清除识别 BIOS 程序包存在的 CMOS 位。如果无热键被按压，系统 BIOS 检查 CMOS 位（方框 840）。如果该位被使能，系统 BIOS 开始寻找 MBU 方框中的视频 BIOS 更新（方框 850）。在这点上，视频 BIOS 出现在 MBU 块中（方框 860）。

在 MBU 块中找到视频 BIOS 以后，系统 BIOS 可以确定，是否视频 BIOS 程序包括被压缩（方框 870）。如果是的，系统 BIOS 可以为了与视频 BIOS 程序包关联的解压代码搜索 MBU 块（方框 880，890）。如果无代码被找到，系统 BIOS 可运行常驻的视频 BIOS，因为视频 BIOS 程序包不可能得到（方框 820）。如果代码被找到，BIOS 可将视频 BIOS 解压并执行它（方框 900，910）。

虽然未必可能，固件的 MBU 区域可能受到损伤或成为无功能。在这样一种情况下，POST 应该使系统回复到存储在系统 BIOS 中的视频 BIOS 选项 ROM 的常驻版本。可通过用户输入强制执行常驻视频 BIOS。如果系统确定在视频适配器已被初始化以前初期的引导期间用户已经按压预先确定的旁路热键，系统 BIOS 可以省略 MBU 区域的扫描并利用常驻的集成化的系统 BIOS 引导。

依据实施方案，为了单一的重新启动，旁路热键使增强空间的系统 BIOS 不能工作。它不需要使来自试图从非挥发性存储器装载选项 ROM 更新的未来的引导不能工作。为了完全防止来自 POST 期间执行的受损的视频 BIOS 修补程序，在 CMOS 建立的例行程序中可以清除 CMOS 使能位。

当“使用 MBU 视频 BIOS” CMOS 选项被设置时，无旁路键被按压，BIOS 可以扫描 MBU 块，寻找视频 BIOS 更新模块。如果未找到视频 BIOS 更新模块。BIOS 可以执行作为系统 BIOS 的部件存储的常驻 BIOS。否则，BIOS 抽出视频 BIOS 更新模块，执行它并运行其余的系统 BIOS（方框 920）。

装载选项 ROM

增强空间也可以存储选项 ROM。依据实施方案，可以通过单独的 CMOS 选项位使选项 ROM 不能工作。

在执行任何的选项 ROM 以前，系统 BIOS 可以检查 MBU 块中被更新的选项 ROM，并在系统 BIOS 图象中所存储的任何选项 ROM 以前执行那些选项 ROM。

CMOS 选项

图 7 示出依据本发明实施方案的一种更新视频 BIOS 的方法。系统 BIOS 可以提供一种 CMOS 选项，允许用户有能力使 MBU 块中被更新的视频 BIOS 不能使用。在其中，这一位被称为“使用 MBU 视频 BIOS” CMOS 位，当 MBU 块的成功更新被完成时，设置“使用 MBU 视频 BIOS” CMOS 位。举例来说，对于 Intel 处理器，系统 BIOS 可以从系统管理模式（“SMM”）环境设置这一位。当这个 CMOS 位被设置时，后继的引导可以使系统 BIOS 在执行常驻的视频 BIOS 以前搜索用于被更新的视频 BIOS 的 MBU 块。如果 MBU 块是空的，系统 BIOS 可以清除并使“使用 MBU 视频 BIOS” CMOS 位非使能。

正如以上所指出的那样，MBU 程序包可以包括用于装入增强空间的几个 BIOS 程序包。MBU 可以依据预先确定的结构构成，以便使安装器能够识别和处理 MBU 程序包的组成部件。

图 8 示出依据本发明的实施方案的一个样本 MBU 程序包。在 MBU 程序包 1000 内，可以存在多个 BIOS 程序包 1010，1020，1030，其中的一些可被压缩并可改变规模。BIOS 程序包 1010，1020，1030 可被相互邻近地放置。存储在 MBU 程序包 1000 中的每个 BIOS 程序包 1010，1020，1030 可以包括识别目标并规定它的规模的目标头段 1011，1021，1031。目标数据可跟随在目标头段项目之后。

在此详细地展示和描述了本发明的几种实施方案。然而，将认识到本发明的修改和变化都被以上的讲授内容所包含并在所附权利要求的范围内，而并不偏离本发明的精神和所指定的范围。

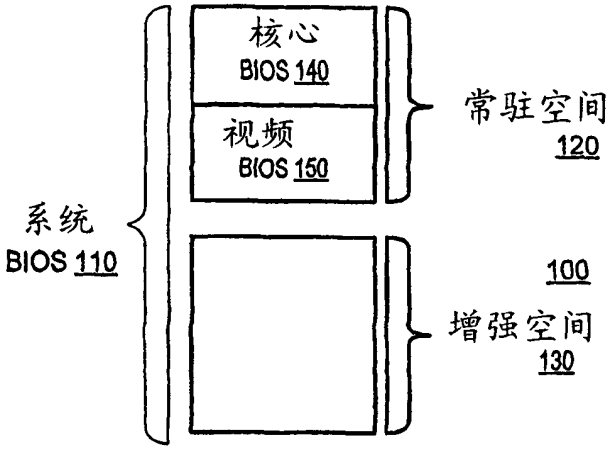


图 1

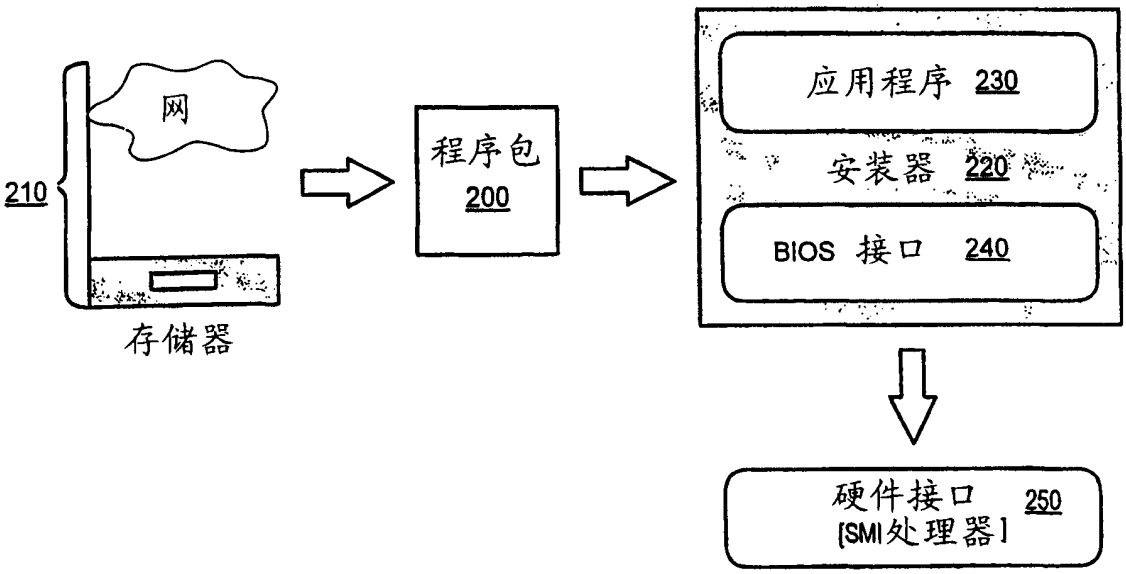


图 2

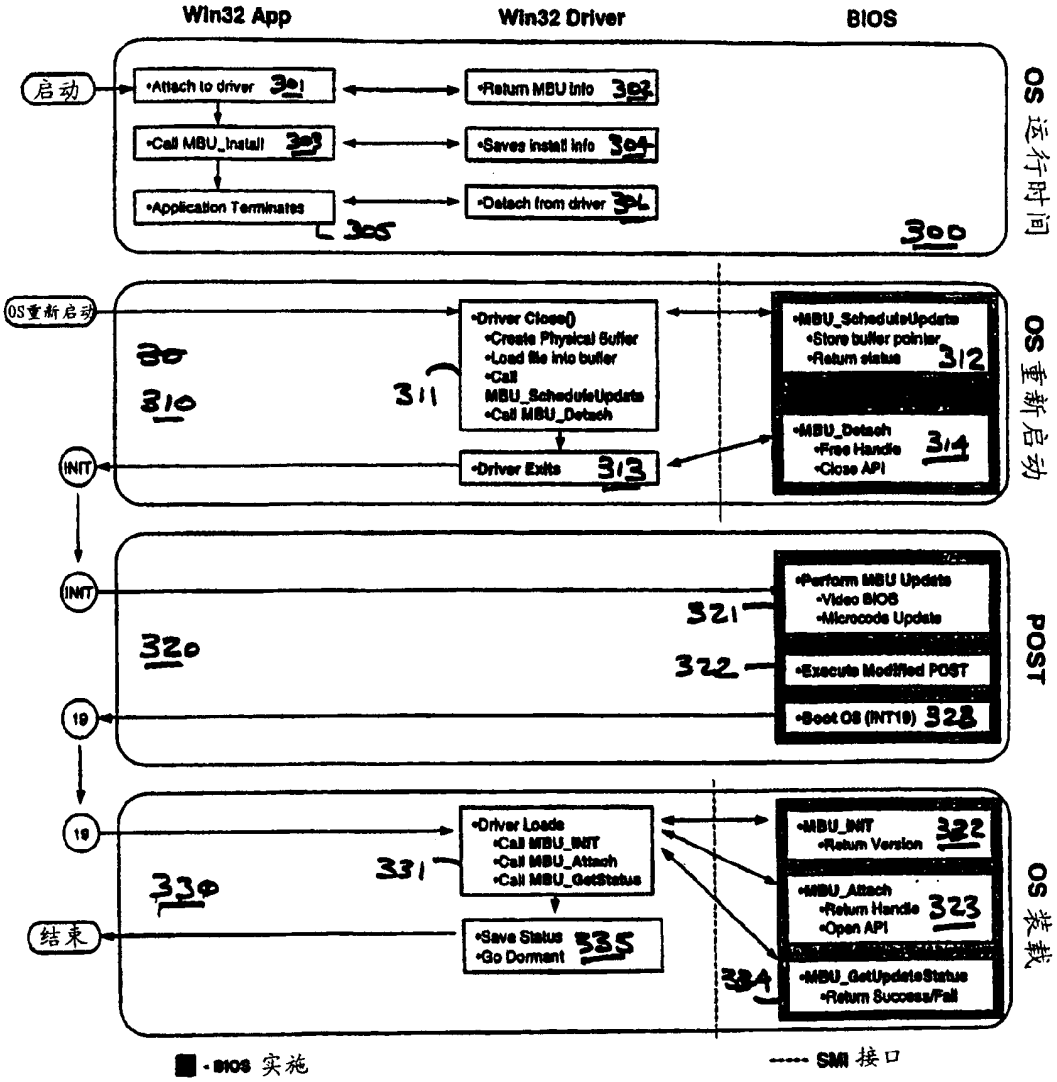

Win32 App
Win32 Driver
BIOS
启动
•Attach to driver 301
•Call MBU_Install 303
•Application Terminates 305
•Return MBU Info 302
•Saves Install Info 304
•Detach from driver 306
300
OS 运行时间
OS重新启动
•Driver Close()
•Create Physical Buffer
•Load file into buffer
•Call MBU_ScheduleUpdate
•Call MBU_Detach
311
•Driver Exits 313
•MBU_ScheduleUpdate
•Store buffer pointer
•Return status 312
•MBU_Detach
•Free Handle
•Close API 314
310
INIT
INT
320
•Perform MBU Update
•Video BIOS
•Microcode Update 321
•Execute Modified POST 322
•Boot OS (INT19) 328
19
POST
330
•Driver Loads
•Call MBU_INIT
•Call MBU_Attach
•Call MBU_GetStatus
331
•Save Status
•Go Dormant 335
结束
•MBU_INIT
•Return Version 322
•MBU_Attach
•Return Handle
•Open API 323
•MBU_GetUpdateStatus
•Return Success/Fail 324
OS 装载
BIOS 实施
SMI 接口

图 3

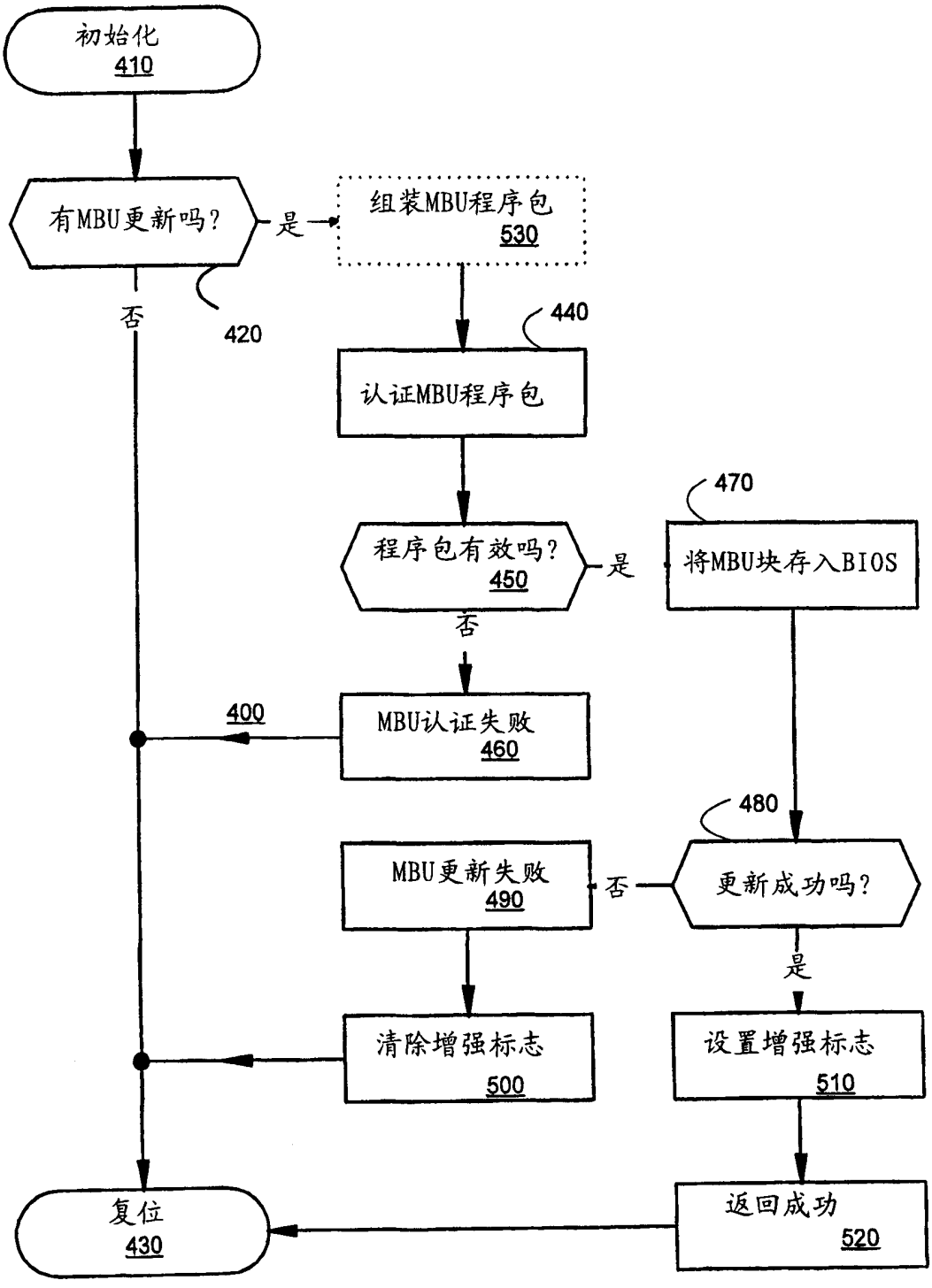

初始化
410
有MBU更新吗?
是
组装MBU程序包
530
否
420
440
认证MBU程序包
470
程序包有效吗?
450
是
将MBU块存入BIOS
否
400
MBU认证失败
460
480
MBU更新失败
490
否
更新成功吗?
是
清除增强标志
500
设置增强标志
510
复位
430
返回成功
520

图 4

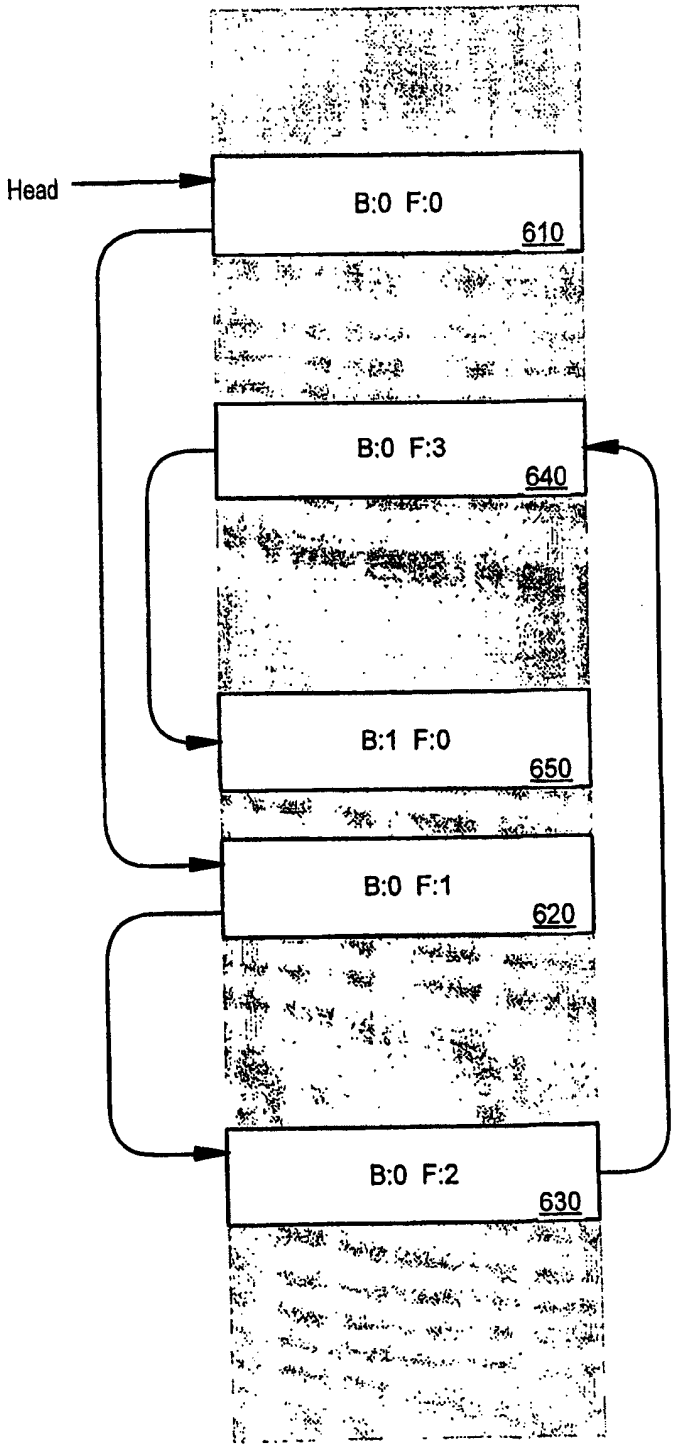


图 5

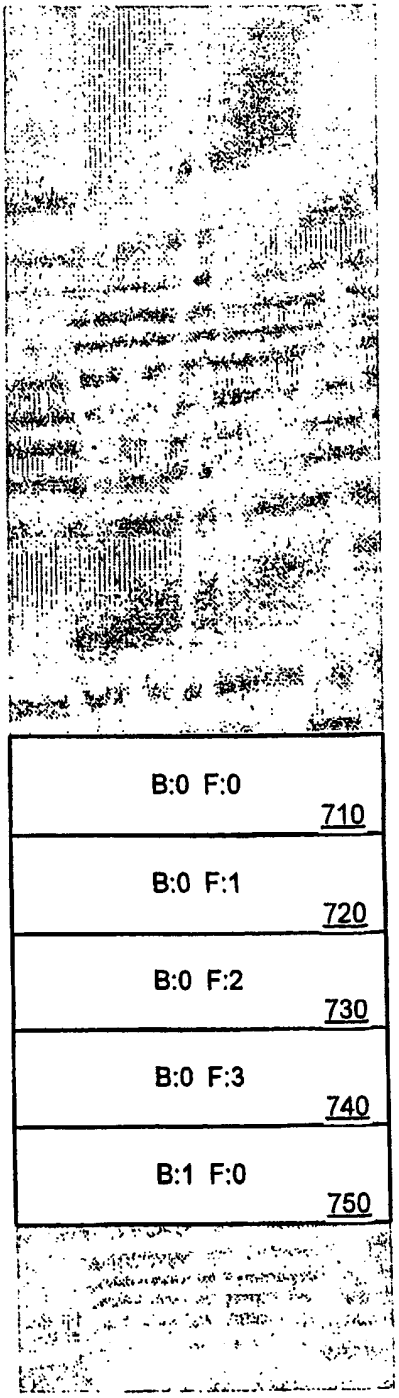


图 6

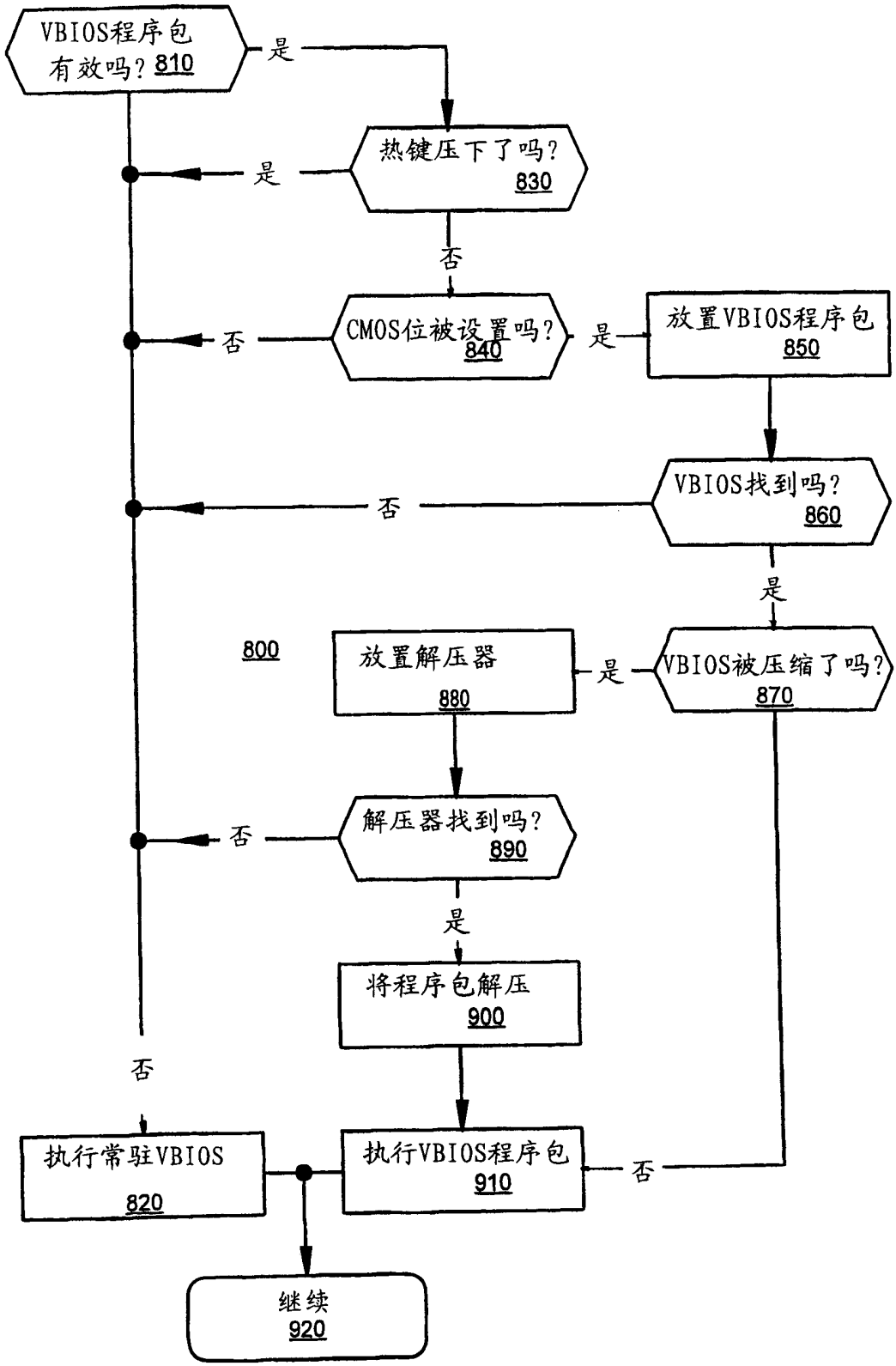

VBIOS程序包
有效吗? 810
是
热键压下了吗?
830
是
否
CMOS位被设置吗?
840
否
是
放置VBIOS程序包
850
VBIOS找到吗?
860
否
是
800
放置解压器
880
是
VBIOS被压缩了吗?
870
解压器找到吗?
890
否
是
将程序包解压
900
否
执行常驻VBIOS
820
执行VBIOS程序包
910
否
继续
920

图 7

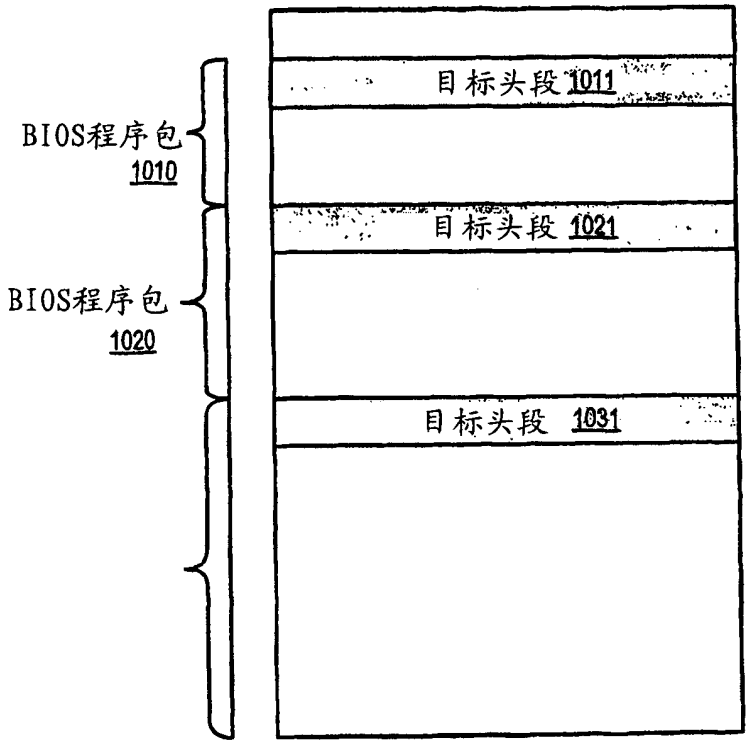

目标头段 1011
BIOS程序包
1010
目标头段 1021
BIOS程序包
1020
目标头段 1031
1000

图 8