(19)中华人民共和国国家知识产权局

(12)发明专利申请

(10)申请公布号 CN 106155657 A
(43)申请公布日 2016.11.23

(21)申请号 201510162196.9

(22)申请日 2015.04.08

(30)优先权数据
14/583,229 2014.12.26 US

(71)申请人 美商安迈科技股份有限公司
地址 美国乔治亚州诺克罗斯市奥克布鲁克公园大道5555号200号大楼

(72)发明人 陈信宏 谢东翰 黄冠杰 苏荷穗

(74)专利代理机构 中国商标专利事务所有限公司 11234
代理人 宋义兴

(51)Int.Cl.
G06F 9/44(2006.01)

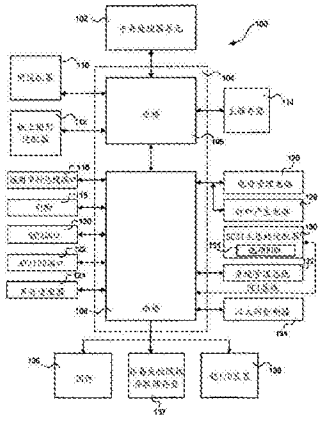
权利要求书3页 说明书12页 附图6页

(54)发明名称

UEFI 固件的方法及其计算机系统

(57)摘要

一种UEFI(Unified Extensible Firmware Interface统一的可延伸固件界面)固件的计算机处理执行方法,其中该UEFI固件支援将多个启动加载器程序其中之一进行初始化设定,该UEFI固件位于一储存器中,该储存器可运作地耦接于该处理器UEFI固件,该方法包含:执行该UEFI固件;检测该处理器中一指令集的类型;根据一模式选择,判断一启动选择模式;从多个组态数据中取得一共同组态数据;从该储存器的该UEFI固件的该多个组态数据中,取得对应于该启动选择模式的一独特组态数据;产生对应于该启动选择模式的一启动加载器地址;以及根据该共同组态数据及该独特组态数据,执行一储存器中位于该启动加载器地址所指位址的启动加载器程序。



CN 106155657 A

1.一种UEFI固件的计算机处理执行方法，其特征在于，该UEFI固件支援将多个启动加载器程序其中之一进行初始化设定，该UEFI固件位于一储存器中，该储存器可运作地耦接于该处理器UEFI固件，该方法包含：

以该处理器执行该UEFI固件；

当执行该UEFI固件时：

检测该处理器中一指令集的类型；

根据一模式选择，判断一启动选择模式，其中该启动选择模式对应于所检测到的该指令集的类型，且该模式选择对应于该UEFI固件执行完后所需执行的一启动加载器程序的类型；

以该处理器，从该储存器中，于该UEFI固件的多个组态数据中取得一共同组态数据；

以该处理器，从该储存器的该UEFI固件的该多个组态数据中，取得对应于该启动选择模式的一独特组态数据，其中该独特组态数据包括处理器指令集及执行时服务；

产生对应于该启动选择模式的一启动加载器地址；以及

根据该共同组态数据及该独特组态数据，执行一储存器中位于该启动加载器地址所指位址的启动加载器程序。

2.如权利要求1所述的UEFI固件的计算机处理执行方法，其特征在于，判断该启动选择模式的步骤进一步包含：

以该处理器从一界面装置接收一选择指令；以及

依据该选择指令判断该启动选择模式。

3.如权利要求2所述的UEFI固件的计算机处理执行方法，其特征在于，该界面装置为一输入/输出装置，该输入/输出装置耦接至该处理器并且可被该UEFI固件侦测到，该选择指令是由该输入/输出装置产生。

4.如权利要求1所述的UEFI固件的计算机处理执行方法，其特征在于，判断该启动选择模式的步骤进一步包含：

以该处理器，在执行该启动加载器程序的步骤前，判断一启动标记是否有在该UEFI固件的组态数据中被设定过，其中该启动标记表示一特定启动加载器程序：

若有设定该启动标记，以该处理器组态该启动选择模式以对应于该启动标记所表示的启动加载器程序；

若没有设定该启动标，以该处理器依据一预设启动加载器程序组态该启动选择模式。

5.如权利要求4所述的UEFI固件的计算机处理执行方法，其特征在于，判断该启动标是否有被设定的步骤进一步包含：

以该处理器，依据该启动选择模式设定该预设启动加载器程序。

6.如权利要求4所述的UEFI固件的计算机处理执行方法，其特征在于，该使用者界面包含多个模式选择项，该方法进一步包含：

以该处理器，接收与该使用者界面相关的一选择指令；以及

从该选择指令判断该启动选择模式。

7.如权利要求1所述的UEFI固件的计算机处理执行方法，其特征在于，在判断该启动选择模式的步骤前，进一步包含：

以该处理器，输出一使用者界面供一显示装置显示，该显示装置耦接至该处理器。

8. 如权利要求 1 所述的 UEFI 固件的计算机处理执行方法，其特征在于，产生该启动加载器地址的步骤进一步包含：

当执行该 UEFI 固件时，检测该处理器的该指令集的类型；

以该处理器，取得相对所侦测到的该指令集类型的该独特组态数据，以致使能对应的执行相对该侦测到处理器指令集类型的该启动加载器程序。

9. 一种 UEFI 固件的计算机系统，其特征在于，包括：

一储存器，储存一 UEFI 固件，该 UEFI 固件支援将多个启动加载器程序其中之一进行初始化设定，其中该 UEFI 固件具有多个指令及多个组态数据对应于不同该启动加载器程序；以及

一处理器，运作地耦接于该储存器，并且执行该 UEFI 固件的指令；

其中该指令是进行以下步骤：

检测该处理器中指令集的类型；

以该处理器，根据一模式选择判断一启动选择模式，其中该启动选择模式对应于所检测到的该指令集的类型，且该模式选择对应于该 UEFI 固件执行完后所需执行的一启动加载器程序的类型；

以该处理器，从该储存器中，于该 UEFI 固件的多个组态数据中取得一共同组态数据；

以该处理器，从该储存器的该 UEFI 固件的该多个组态据中，取得对应于该启动选择模式的一独特组态数据，其中该独特组态数据包括处理器指令集及执行时服务；

产生对应于该启动选择模式的一启动加载器地址；以及

根据该共同组态数据及该独特组态数据，执行一储存器中位于该启动加载器地址所指位址的启动加载器程序。

10. 如权利要求 9 所述的 UEFI 固件的计算机系统，其特征在于，该处理器包含一中央处理单元，以及该储存器包含闪存储存器及只读储存器。

11. 如权利要求 9 所述的 UEFI 固件的计算机系统，其特征在于，该启动加载器程序对应于一操作系统程序。

12. 如权利要求 9 所述的 UEFI 固件的计算机系统，其特征在于，从该指令中判断该启动选择模式的步骤进一步包含：

以该处理器，从一界面装置接收一选择指令；以及

从该选择指令中判断该启动选择模式。

13. 如权利要求 12 所述的 UEFI 固件的计算机系统，其特征在于，该界面装置为一输入 / 输出装置，该输入 / 输出装置耦接于该处理器并且会被该 UEFI 固件侦测到，以及该选择指令由该输入 / 输出装置产生。

14. 如权利要求 9 所述的 UEFI 固件的计算机系统，其特征在于，该界面装置包含一触碰显示器、键盘、滑鼠装置、指示笔、影像感测器及声音感测器。

15. 如权利要求 9 的所述的 UEFI 固件的计算机系统，其特征在于，从该指令中判断该启动选择模式的步骤进一步包含：

以该处理器，在执行该启动加载器程序的步骤前，判断一启动标记是否有在该 UEFI 固件的组态数据中被设定过，其中该启动标记表示一特定启动加载器程序：

若有设定该启动标记，以该处理器组态该启动选择模式以对应于该启动标记所表示的

启动加载器程序；

若没有设定该启动标，以该处理器依据一预设启动加载器程序组态该启动选择模式。

16. 如权利要求 15 所述的 UEFI 固件的计算机系统，其特征在于，在判断该启动标记是否有被设定过的步骤后，进一步包含：

以该处理器，根据该启动选择模式设定该预设启动加载器。

17. 如权利要求 9 所述的 UEFI 固件的计算机系统，其特征在于，进一步包含一显示器装置，该显示器装置用于显示一使用者界面，其中该显示装置耦接于该处理器。

18. 如权利要求 17 所述的 UEFI 固件的计算机系统，其特征在于，该使用者界面包含多个模式选择项，该指令的步骤进一步包含：

以该处理器，接收与该使用者界面相关的一选择指令；以及

从该选择指令判断该启动选择模式。

UEFI 固件的方法及其计算机系统

技术领域

[0001] 本发明关于一种 UEFI 固件的计算机处理执行方法，尤其是关于一种包含多个不同启动加载器之变数 / 变量及指令集，并且是用于辅助不同启动加载器之启动前的初始化设定的 UEFI 固件的计算机处理执行方法及其计算机系统。

背景技术

[0002] 以传统的计算机系统而言，计算机系统只能启动单一个操作系统。具体而言，操作系统的启动通常是由低阶层指令代码处理，其中低阶层指令码是用来当计算机系统的各个硬件元件与操作系统及执行于该操作系统中其他高阶层软件之间的中介。此低阶层的指令代码常被称为基本输入 / 输出系统 (Basic Input/Output System，亦即 BIOS) 固件，并提供了一组软件程序供高阶层软件可与计算机系统的硬件进行互动。每当计算机系统被通电开启时，该固件会执行一些程序来进行通电自检的测试 (Power-On Self Test，亦即 POST) 以测试并启动计算机系统中的所有硬件元件，并且在其后将控制权转让操作系统。此些硬件元件可包含系统主储存器、硬盘和键盘。

[0003] 然而，随着技术的进步，许多不同的计算机系统和操作系统如雨后春笋般涌现，基于传统 BIOS 标准的启动固件，其原本是专门设计给国际商业机器公司 (International Business Machine、IBM) 的个人电脑，已成为现代操作系统如何控制及操作硬件的限制点。在新的硬件和软件的技术继续被研发出来的情况下，此限制点已成为能硬件及软件间能有更好的互动的主要障碍。故此，有另一组新的 BIOS 固件标准被提出并已被市场上许多业者广泛的采用。这种新的标准被称为统一可延伸固件界面或统一可扩展固定接口 (Unified Extensible Firmware Interface、简称为 UEFI)。

[0004] 随着 UEFI 标准的应用，BIOS 企业可生产 UEFI 固件供各种计算机系统。同时，生产操作系统的公司可通过生产符合 UEFI 标准的操作系统来利用 UEFI 固件所提供的服务优势。然而，UEFI 标准指定固件与操作系统的加载程序 (bootloader) 或核心 (kernel) 必须是大小匹配。例如，一个 64 位的 UEFI 固件只能加载 64 位 UEFI 操作系统启动加载器或核心。因此，对于具有可支援不同处理器模式 (处理器指令集、亦即 processor instruction sets) 的计算机结构 (例如，x86_64) 且符合 UEFI 标准的计算机系统，启动不同操作系统的不同处理器模式分别必须使用到不同的启动加载器映像档案 (boot up image，亦即已被编译的启动加载器的原始码)。例如，为了能在 x86_64 的计算机系统下能够支持启动 32 位或 64 位的操作系统，分别针对该 32 位集 64 位的操作系统的 UEFI 固件必须被编译并安装于该计算机系统中。对于储存量有限或因设计上实体空间不足供摆设足够到的储存器的计算机系统，储存各该 UEFI 固件的原始档案的条件会成为此种空间有限的计算机系统的重大问题。此外，在其他情况下，例如 UEFI 固件已将计算机系统启动为某个处理器模式时，虽然操作系统启动加载器 (或核心 kernel) 可随意更换处理器模式，但执行时期服务 (Runtime Services) 的使用会被禁止 (除非核心又被切换回来)。因此，有必要降低多个 UEFI 启动固件的复杂性以致使能支援多个符合 UEFI 标准的操作系统。

发明内容

[0005] 本发明的一个目的在于提供一种计算机系统及其具有能够支持不同的处理器模式的多个操作系统的UEFI固件的方法。

[0006] 本发明提供一种计算机系统及方法，其具有一UEFI固件支援启动不同操作系统的不同模式，本系统及方法可减少UEFI固件所需要使用的空间。

[0007] 根据本发明的一实施例，计算机系统包括一个储存器及一处理器，其中储存器储存一UEFI固件。该UEFI固件可支援多个不同的启动加载器(bootloader programs)的初始化设定(pre-boot initialization)。其中，该UEFI固件具有多个指令及多个对应于不同启动加载器的组态数据(configuration data)。该处理器是可操作式耦合于该储存器，并且是用于执行该UEFI固件的多个指令；其中，该些指令可包括执行以下步骤：检测该处理器的指令集的类型；以该处理器，依据一模式选择判断一启动选择模式，其中该启动选择模式对应于检测到的指令组类型，且该模式选择是对应于UEFI固件执行后必须执行的启动加载器的类型；以该处理器，从多个该储存器中该UEFI固件的多个组态数据中取得一个共同组态数据；以该处理器从该储存器中，从对应该启动选择模式的该UEFI固件的该多个组态数据中取得一独特组态资料(distinct configuration data)，其中该独特组态数据包括处理器指令集及执行时期服务；产生对应于该启动选择模式的一启动加载器地址(bootloader address)；以及依据该共同组态数据及该独特组态数据，执行位于该储存器中该启动加载器地址对应的指位子的启动加载器程序。

[0008] 于一实施例中，判断该启动选择模式的步骤进一步包含：

[0009] 以该处理器从一界面装置接收一选择指令；以及

[0010] 依据该选择指令判断该启动选择模式。

[0011] 于一实施例中，该界面装置为一输入/输出装置，该输入/输出装置耦接至该处理器并且可被该UEFI固件侦测到，该选择指令是由该输入/输出装置产生。

[0012] 于一实施例中，判断该启动选择模式的步骤进一步包含：

[0013] 以该处理器，在执行该启动加载器程序的步骤前，判断一启动标记是否有在该UEFI固件的组态数据中被设定过，其中该启动标记表示一特定启动加载器程序：

[0014] 若有设定该启动标记，以该处理器组态该启动选择模式以对应于该启动标记所表示的启动加载器程序；

[0015] 若没有设定该启动标，以该处理器依据一预设启动加载器程序组态该启动选择模式。

[0016] 于一实施例中，判断该启动标是否有被设定的步骤进一步包含：

[0017] 以该处理器，依据该启动选择模式设定该预设启动加载器程序。

[0018] 于一实施例中，该使用者界面包含多个模式选择项，该方法进一步包含：

[0019] 以该处理器，接收与该使用者界面相关的一选择指令；以及

[0020] 从该选择指令判断该启动选择模式。

[0021] 于一实施例中，在判断该启动选择模式的步骤前，进一步包含：

[0022] 以该处理器，输出一使用者界面供一显示装置显示，该显示装置耦接至该处理器。

[0023] 于一实施例中，产生该启动加载器地址的步骤进一步包含：

[0024] 当执行该 UEFI 固件时，检测该处理器的该指令集的类型；

[0025] 以该处理器，取得相对所侦测到的该指令集类型的该独特组态数据，以致使能对应的执行相对该侦测到处理器指令集类型的该启动加载器程序。

[0026] 根据本发明的另一个实施例，本发明提供一种用于执行该支援多个不同启动加载器的初始化设定的 UEFI 固件，其中该 UEFI 固件位于一储存器，该储存器是可操作地耦合于该处理器，该方法包括：以该处理器，执行该 UEFI 固件；当执行该 UEFI 固件时：检测该处理器的指令集的类型；依据一模式选择，判断一启动选择模式，其中该启动选择模式对应于所检测到的该指令集的类型，且该模式选择是对应于用于执行该 UEFI 固件后执行的一启动加载器的类型；以该处理器，从该储存器中，从该 UEFI 固件的多个组态数据中取得一个共同组态数据；以该处理器，从该储存器对应于该启动选择模式的 UEFI 固件的该多个组态数据中，取得一独特组态数据，其中该独特组态数据包括处理器指令集及执行时期服务；产生对应该启动选择模式的一启动加载器地址；以及，根据该共同组态数据及该独特组态数据，执行在一储存器中相对该启动加载器地址所指的启动加载器程序。

[0027] 于一实施例中，该处理器包含一中央处理单元，以及该储存器包含闪存储存器及只读储存器。

[0028] 于一实施例中，该启动加载器程序对应于一操作系统程序。

[0029] 于一实施例中，从该指令中判断该启动选择模式的步骤进一步包含：

[0030] 以该处理器，从一界面装置接收一选择指令；以及

[0031] 从该选择指令中判断该启动选择模式。

[0032] 于一实施例中，该界面装置为一输入 / 输出装置，该输入 / 输出装置耦接于该处理器并且会被该 UEFI 固件侦测到，以及该选择指令由该输入 / 输出装置产生。

[0033] 于一实施例中，该界面装置包含一触碰显示器、键盘、滑鼠装置、指示笔、影像感测器及声音感测器。

[0034] 于一实施例中，从该指令中判断该启动选择模式的步骤进一步包含：

[0035] 以该处理器，在执行该启动加载器程序的步骤前，判断一启动标记是否有在该 UEFI 固件的组态数据中被设定过，其中该启动标记表示一特定启动加载器程序：

[0036] 若有设定该启动标记，以该处理器组态该启动选择模式以对应于该启动标记所表示的启动加载器程序；

[0037] 若没有设定该启动标，以该处理器依据一预设启动加载器程序组态该启动选择模式。

[0038] 于一实施例中，在判断该启动标记是否有被设定过的步骤后，进一步包含：

[0039] 以该处理器，根据该启动选择模式设定该预设启动加载器。

[0040] 于一实施例中，进一步包含一显示器装置，该显示器装置用于显示一使用者界面，其中该显示装置耦接于该处理器。

[0041] 于一实施例中，该使用者界面包含多个模式选择项，该指令的步骤进一步包含：

[0042] 以该处理器，接收与该使用者界面相关的一选择指令；以及

[0043] 从该选择指令判断该启动选择模式。

附图说明

[0044] 图 1 为本发明包含计算装置的一操作环境的一实施例；

[0045] 图 2 为一计算装置的开机 / 启动结构的一实施例；

[0046] 图 3 为从一只读存储器中断执行启动码的一方法；

[0047] 图 4 为用于从一次可编程存储器中执行另一启动码的一方法；

[0048] 图 5 为启动码及一启动修补码的一实施例；

[0049] 图 6 为用于将启动码烧录至一一次可编程存储器的熔丝的方法；

[0050] 图 7 为一系统单晶硅 (SoC) 的实施例。

[0051] 附图符号说明

[0052] 100 计算机系统

[0053] 102 中央处理器单元

[0054] 104 芯片组

[0055] 106 北桥 (northbridge)

[0056] 108 南桥 (southbridge)

[0057] 110 网适配器

[0058] 112 板上图形适配器

[0059] 114 主储存器

[0060] 116 通用串行总线端口

[0061] 118 引脚

[0062] 120 SATA(串行高级技术附件) 端口

[0063] 122 ATA100 端口

[0064] 124 声音适配器

[0065] 126 电源管理电路

[0066] 128 时钟产生电路

[0067] 130 SCSI 主总线适配器

[0068] 132 系统管理总线

[0069] 134 以太网控制器

[0070] 136 固件

[0071] 137 非易失性随机存取储存器

[0072] 138 超 I/O 装置

[0073] 202 操作系统

[0074] 302 OS 启动加载器

[0075] 304 EFI 启动服务

[0076] 306 EFI 执行期服务

[0077] 308 平台性固件

[0078] 312 启动加载器

[0079] 314 其他规格的界面

[0080] 316 平台硬件

[0081] 318 EFI 系统分区

[0082] 322 OS 分区

[0083] 410 处理器

[0084] 420 记忆体

[0085] 422 UEFI 固件

[0086] 424 指令集

[0087] 426 组态数据

[0088] 427 同组态数据

[0089] 428 独特组态数据

[0090] 430 组件

[0091] 440 储存器

[0092] 430A 显示器

[0093] 401、402、403、404、405、406、407、408、409 步骤

[0094] 方法 300

[0095] 310、320、330、340、350、360、370 步骤

具体实施方式

[0096] 本发明的各种实施例是提供用于执行启动加载器固件(bootup firmware)的方法及系统，其启动加载器固件可支援对多个不同启动加载器中其中之一进行初始化设定。在以下的详细描述中，将会搭配附图进行说明，其所示的附图为具体的实施例或示例。然而，这些实施例仅是为了描述本发明及技术范围，而不应被解释为对本发明的一种限制。敬请参照附图，其中类似的标号在该些图示中表示类似的元件，以下将会说明本发明各方面的实施例及其较佳的操作环境。在附图中，标号的最左边的数字标识该附图首次出现的顺序。在说明及附图的不同情况下，使用相同的图标表示为相同的元件。

[0097] 本发明提供一种计算机系统及其方法，用于执行一UEFI固件，该UEFI固件可支援启动加载器程序(bootloader program)的初始化设定。在较佳情况下，该计算机系统包含(但不限于此)笔记本电脑、个人计算机、伺服器、手持式计算设备，如移动式通讯装置及平板电脑，以及可穿戴的计算设备。

[0098] 图1及以下的讨论内容旨在提供可实现本发明的合适的计算环境的简要说明。然而，本领域的技术人员将可认识到，本发明还可以以其他合适的计算环境中实现。此外，本领域的技术人员将理解，本发明也可与其他计算机系统配置，包括多处理器系统(multiprocessor systems)、

[0099] 基于微处理器的或可编程的消费电子产品、小型计算机、大型计算机等实施。本发明也可以在分布式计算环境(distributed computing environment)实施，其中任务是由多个远端处理装置执行，且该些远端处理装置是通过一通信网路彼此连接。

[0100] 请参照图1，图1为一示范式的计算机架构，可用于实现本发明的各种实施例。在此应当理解的是，虽然在此所描述的实施例是以一传统的桌上型计算机装置或伺服器计算装置讨论，然而本发明的实施例亦可与几乎任何类型的计算机装置使用或实现。图1为一计算机系统100的计算机架构的示范，该计算机系统100可被运作地从固件启动一操作系统。图1中的区块用于表示计算机架构中的功能组件，而并非一定是代表单独地实体组件。所描述到的功能组件在保持计算机架构的整体方向及目的的情况下，可以被组合、分离或

被除去。

[0101] 为了提供本文所描述的功能性，该计算机系统100包括一基板或“主板”，其是多个组件或装置可通过一系统总线或其他通信路径的方式连接的一印刷电路板。在一实施例中，中央处理器单元(Central Processing Unit，简称CPU)102与芯片组104相互地运作。中央处理器单元102可为标准的中央处理器，可进行计算机装置运行中所需要的算数及逻辑运算。该中央处理器单元102，在此以及在其他实施例中，可包括一或多个微处理器、微控制器、现场可编程门阵列(Field programmable gate array，简称FPGA)、复杂可编程逻辑器件(Complex programmable logic device，简称CPLD)、专用集成电路(application specific integrated circuit，简称ASIC)及/或任何其它电子计算设备。

[0102] 该芯片组104包括北桥(northbridge)106及南桥(southbridge)108。北桥106提供中央处理器单元102与计算机系统100其余部分之间的一界面。北桥106还提供了一界面至一或多个随机存取储存器(Random Access Memory，简称RAM)，用于当计算机系统100中的主储存器114，以及该北桥106更可能提供一界面至一板上图形适配器(on-board graphics adapter)112。北桥106还可以通过一网适配器110启用网路通信功能。网适配器110能够将该计算机系统100经由一网路连接至一或多个其他的他计算机。网适配器110可进行连接的网路可以包括局域网(Local Area Network，简称LAN)或广域网(Wide Area Network，简称WAN)，例如常见于办公室的LAN及WAN联网环境、企业范围计算机网络(enterprise-wide computer networks)、内联网(intranets)及网路上。该北桥106系连接到该南桥108。

[0103] 南桥108是负责控制计算机系统100的许多输入/输出功能。特别是，南桥108可提供一或多个通用串行总线(Universal Serial Bus，简称USB)端口116、一声音适配器124、一以太网控制器134及一或多个通用输入/输出(general purpose input/output，简称GPIO)引脚118。南桥108还可以提供一个总线，用于连接周边卡设备，如符合BIOS启动规范(BIOS boot specification，简称BBS)标准的SCSI主总线适配器130。在一实施例中，总线包括周边组件互连(Peripheral component interconnect，简称PCI)总线。南桥108还可以提供一系统管理总线132，用于管理计算机系统100中的各种组件。在南桥108运作时，也可以利用电源管理电路126及时钟产生电路128也可以在南桥108。

[0104] 南桥108还可用于提供一或多个界面来将大容量储存设备连接到计算机系统100。例如，根据一实施例中，南桥108包括串行高级技术附件(Serial Advanced Technology Attachment，简称SATA)适配器，用于提供一或多个串行高级技术附件端口120及一个高级技术附件100适配器，该高级技术附件100适配器是用于提供一或多个ATA100适配器端口122。该串行ATA100适配器端口120及ATA100适配器端口122可被连接至一或多个大容量储存设备，其中大容量储存设备可储存操作系统、应用程序及其他数据。本领域技术人员应已熟知的是，一操作系统包括控制一计算机的运作及分配资源的一组程序。一应用程序是一个在操作系统上运行的软件(或运行于其他时期环境(runtime environment)中)并且会使用计算机资源来执行计算机系统100的使用者所期望被执行的应用程序的特定任务。

[0105] 连接到该南桥108及该SCSI主总线适配器130(以及其相关的计算机可读介质)的大容量储存设备可供计算机系统100非易失性储存(non-volatile storage)的功能。虽

然在本文中计算机可读介质指的是大容量储存装置，如硬碟 / 硬盘或光碟驱动器 (CD-ROM drive)，本领域技术人员应可理解的是，计算机可读介质可为该计算机系统 100 可读取的任何介质。以举例而言（而不受此举例限制），计算机可读介质可以包括计算机储存介质和通信介质。计算机储存介质包括易失性和非易失性 (volatile and non-volatile)、可移动和不可移动介质，以任何方式或资料储存技术（例如，计算机可读指令、资料结构、程序模块或其他资料）实现。计算机储存介质包括（但不限于）RAM、ROM、EPROM、EEPROM、闪存或其他固态储存器技术、CD-ROM、DVD、HD-DVD、BLU-RAY 或其他光学储存、磁带盒、磁带、磁盘储存或其它磁储存设备，或该计算机任何可被利用的介质来储存所需信息 / 资料。

[0106] 南桥 108 可提供一低引脚计数 (Low pin count，简称 LPC) 界面 / 接口，用于连接超 I/O 装置 138。该超 I/O 装置 138 是负责提供多个输入 / 输出端口，包括一键盘端口、一滑鼠端口、一串行界面 / 接口、一并行端口以及其他类型的输入 / 输出端口。该 LPC 界面或另一个接口可被用于连接至一储存介质，例如 ROM 或非易失性随机存取储存器 (NVRAM) 137，例如快闪储存器。该计算机储存介质可被用于储存该固件 136，其中该固件 136 包括用于协助启动该计算机系统 100 以及用于在该计算机系统 100 内元件之间传送数据的指令及数据。

[0107] 固件 136 可包括符合 UEFI 标准的程序代码 (program code)。应当理解的是，除了该固件 136 包括符合 UEFI 标准的固件，其它固件的类型及组合亦可被包括在固件 136 中。举例而言，固件 136 可以包括附加地或替代地一个 BIOS 固件及 / 或其他现技术中以熟知的类型的固件。以下与附图搭配提供 UEFI 固件 136 的运作的相关说明。应当理解的是，计算机系统 100 可以不包括图 1 中所有显示地组件，且亦可能包括在图中未明显示的其他组件，或者可以利用与图 1 完全不同的结构。

[0108] 敬请参照图 2，更多关于符合 UEFI 标准的一系统（可被利用来提供一操作环境给以下各个实施例）将会在下进行说明。如图 2 所示，该系统包括一平台硬件 (platform hardware) 316 及操作系统 (Operating system，简称 OS) 202。一平台性固件 (platform firmware) 308 可通过使用 OS 启动器 (OS loader，或有时被称为启动加载器或 OS 启动加载器) 302 从 EFI 系统分区 (EFI system partition) 318 取得 OS 程序（指令）代码。相同的，该 OS 启动加载器 302 可从其他位子取得 OS 程序代码，例如包括从连接的周边设备或者从固件 136 本身中取得。该 EFI 系统分区 318 亦可为一种结构性可共享系统分区 (architecturally shareable system partition)。因此，该 EFI 系统分区 318 是定义一分区及资料系统，其是设计以允许多个供应商之间能安全地共享大容量的储存。此外，一 OS 分区 322 亦可被使用。

[0109] 一旦被启动，OS 启动加载器 302 会继续启动完整的操作系统 202，可能是以阶段式的方式启动加载，例如通常与 Linux 系统中常会用到的 GRUB 启动加载器。该 OS 启动加载器 302 可以使用 EFI 启动服务 304 并且与其他支援的规格连接 / 当界面，以进行调查、了解、以及初始化各种平台组件和管理该些组件的操作系统 202。因此，其他规格的界面 314 亦可存在于系统中。举例而言，高级配置和电源管理接口 (Advanced Configuration and Power Management Interface，简称 ACPI) 及系统管理 BIOS (System Management BIOS，简称 SMBIOS) 的规格可被支援。

[0110] EFI 启动服务 304 提供界面给各种设备，以及提供可用于启动期间使用的系统功

能。EFI 执行期服务 306 可在启动阶段被提供给该 OS 启动加载器 302，以及在操作系统 202 执行时提供给该操作系统 202 使用。例如，执行时服务可被提供，以确保在操作系统 202 的正常运作中，操作系统 202 可能会需要的硬件资源可适当的被抽象化 (abstraction of base platform hardware resources)。EFI 可扩展平台的固件，通过加载 EFI 驱动程序及 EFI 应用程序镜像 (application images)，可利用到 EFI 定义的执行时服务及启动服务。一旦 EFI 固件被启动，该 EFI 固件将会把控制权转交给该启动加载器 312。

[0111] 敬请参照图 3，图 3 描述 UEFI 固件启动一操作系统的不同阶段。如图 3 所示，当计算机系统 100 通电时，本发明的 UEFI 固件会被计算机系统 100 的处理器执行。该 UEFI 固件将首先进入一个安全阶段 (Security Phase，亦即 SEC)，其中在此阶段中还没有任何储存器 / 记忆体在该计算机系统 100 中被启用或初始化。在本阶段中，由于储存器还尚未被初始化，处理器的高速缓冲储存器 (cache) 以随机储存器 (RAM) 方式使用来初始确认该中央处理单元 (CPU)、该芯片组及该主机板。接着，UEFI 固件会进入预 Pre-EFI 初始化 (Pre-EFI Initialization，简称 PEI) 阶段，其中该计算机系统 100 的中央处理单元、该芯片组、该主机板及该储存器会被初始化。在驱动执行 (Driver Execution，简称 DXE) 阶段，启动服务、执行时服务及驱动程序执行调度服务 (driver execution dispatcher services) 可被执行以初始化计算机系统 100 的任何其他硬件。继 DXE 阶段后，UEFI 固件会进入驱动设备选择 (Boot Device Selection，简称 BDS) 阶段。在 BDS 阶段，会尝试对操作系统对应的启动加载程序的控制台设备以及各种驱动器进行初始化。在暂态系统负载 (Transient System Load，简称 TSL) 阶段中，控制权将会被转给操作系统以继续计算机系统 100 的启动，并且再到达运行时期 (Runtime，简称 RT) 阶段。

[0112] 图 4 显示本发明的一实施例。如图 4 所示，计算机系统 100 可包括处理器 410、记忆体 420、组件 430、储存器 440 以及显示器 430A，其中该处理器 410 是分别耦合至该记忆体 420、该组件 430、该储存器 440 及该显示器 430A。本领域的技术人员应该能够理解的是，图 4 中所描述的计算机系统 100 可与图 1 的计算机系统结合或分开的看待。在本实施例中，记忆体 420 可为快闪储存器或 CMOS，用以储存 UEFI 固件 422，其中该 UEFI 固件 422 具有指令集 424 和组态数据 426。储存器 440 可为硬碟 (hard disk drive)、外接式储存器、快闪储存器、网络硬盘或任何其他的储存器。其中，包括计算机系统 100 的操作系统用的启动加载器的该 EFI 分区 (EFI partition) 是储存于该储存器 440 中。该组件 430 及该显示器 430A 可被视为是在启动期间内被执行的 UEFI 固件初始化的硬件设备。

[0113] 图 5 为执行于计算机系统 100 中的 UEFI 固件的流程示意图。参照图 4 及 5，当计算机系统 100 于步骤 401 中通电时，该计算机系统 100 将会开始执行该 UEFI 固件 422。在一实施例中，UEFI 固件 422 可侦测及确认该处理器 410 的指令集 (instruction set) 来判断该计算机系统 100 的处理器架构的种类。举例而言，在图 5 的示范所示，若该处理器被确定为一个非 x86 处理器架构，例如 ARM 处理器架构，该 UEFI 固件 422 将会执行步骤 403 以继续进行启动的程序。出于解释的目的，本发明以下内容将会以 x86 的芯片组架构进行说明。然而，本领域技术人员应能轻易了解到本发明可应用于其他不同芯片组架构中，以下的解释说明并不意味着本发明应限制于 x86 架构范围下。如图 4 及 5 所示，若处理器 410 被确认是为 x86 架构，该 UEFI 固件 422 将会前进到步骤 404 并进入 PEI 阶段。当处理器的预验证 (pre-verification) 及初始化进行完成后，通过执行该 UEFI 固件 422，该处理器将会

在步骤405中判断处理器的处理器模式或指令集的类型。在本实施例中，若该处理器模式被确定是32位元，该UEFI固件422将会进行DXE阶段的步骤406。然而，若该处理器模式确定为64位，该UEFI固件422将会进行到DXE阶段的步骤408。

[0114] 如上述所提过，在UEFI标准下于启动期间(boot period)中，该UEFI固件422只能引用相同大小的指令集的启动加载器程序。以图5中的实施例为举例，若计算机系统100是以64位模式初始化预操作系统环境，该UEFI固件只能引用/启动相对于一64位操作系统的启动加载器程序。同样的，若该UEFI固件422以32位模式将预操作环境(pre-OS environment)进行初始化，该UEFI固件只能引用或呼叫相对于一32位操作系统的启动加载器程序。由于这一事实，即32位和64位指令集是完全不同的，UEFI固件的制造商通常会需要编译对应于32位模式及64位模式的两个单独的固件映像，以确保计算机系统100可以分别被启动至32位操作系统或64位操作系统。然而，在此情况下会增加储存固件映像所占取的总储存空间。此外，在提供固件更新或针对UEFI固件中各别映像进行修补程序或维护时会变得十分复杂及困难，因为制造商将会需要为每个不同的映像进行重安装程序(reflash)，或者为了能同步所有不同固件映像的更新而去跟踪每个映像的组态表(configuration table)的每个变量数，此现象会增加系统的复杂度。

[0115] 请参见图6A，图6A为UEFI固件的一新的数据结构的实施例。如图6A所示，本文提议将不同的固件映像包装在单一个映像包装中。为了实现这一点，在本实施例中，32位固件映像和64位固件映像各自的组态表中的变量数据将会进行比较，并归类为共同数据或独特数据。任何变量数据同时共同存在于32位及64位的固件映像中将会被归类为共同数据；相对的，任何变量数据不共同存在于32位及64位的固件中（亦即，只存在于32位的或64位的固件映像）将会被归类为独特数据。当编译成单一的固件映像时，所有共同数据将会被分组至一共同组态数据427的列表中，而所有的独特数据将会被分组至一独特组态数据428的列表中。举例而言，如图6所示的组态数据126的一实施例，共同组态数据127可包括各种变量数据，其可表示预设启动的操作系统或表示所有不同启动加载器共同可用的启动图像/标志(boot up logo)。通过将共同数据分组再一起，可从该些不同固件映像中删除重复的变量数据。以此方式，固件映像包装的总尺寸可以被减少。此数据结构的另一个优点在于，制造商可以更好地跟踪不同固件映像的变量数据，同时又可更简单的进行UEFI固件的更新。此外，在操作系统环境中的期间，变量数据的任何变化将会在计算机系统100被重启动时即时的被反映出来。举例而言，若在操作系统的64位模式下更改预设启动模式为32位模式，当计算机系统100下次被重启动时，计算机系统100会先侦测到该预设启动模式为32位模式，并且会对应的初始化各个硬件组件以及以32位处理器模式进行启动程序。通过此方式，相对的32位操作系统将会被启动起来。

[0116] 在其他不同实施例中，当计算机系统100通电并且在PEI阶段后但在DXE阶段前，该UEFI固件可指示该处理器来先初始化一硬件控制台，例如键盘、触碰显示器或任何其它能够允许使用者输入简单的指令的控制台。在此情况下，使用者可键入或输入代表一模式选择的命令/指令；其中，该模式选择是指一特定的处理模式及/或使用者所欲计算机系统100启动的操作系统。举例而言，按压功能键“1”可代表指示给UEFI固件以下资信：使用者想在计算机系统100启动至32位模式的操作系统。或按压功能键“2”可代表使用者希望该UEFI固件启动至64位模式的操作系统。

[0117] 如图5及图6A所示，由于不同的处理器模式需要相应不同的指令集以使得处理器能够理解和正确执行指令，UEFI固件422亦可以有指令424的例表，其中该指令424的列表具有指令I1和I2。再回到双种操作系统（32位及64位）的例子中，指令I1可以对应于32位指令集而指令I2可以对应于64位的指令集。当执行UEFI固件422到达步骤405时，在本实施例中，组态数据126中的变量数据可先被检查以确定哪个处理器模式应当被执行。例如，若一GUID预设启动模式是设定为32位模式时，UEFI固件422的执行将会进入步骤406的DXE阶段。如前述提起过，在DXE阶段下，对应于处理器模式的指令集及执行时服务可被执行。在此举例中，32位的指令及I2将会被执行。处理器将会取得对应于启动选择模式（从共同组态数据427的检查预设启动模式的变量数据来判断）的共同组态数据427及独特组态数据428。处理器接着会对应于该启动模式选择产生一启动加载器地址，以致使在储存器440(EFI分区）中于相对该启动加载器地址所指的位址的启动加载器程序可根据取得的共同组态数据及独特组态数据被执行。换句话说，只有对应于该启动选择模式的指令424及组态数据126会被取得来进行EFI分区内该对应的启动加载器程序的启动。通过此方式，在进入OS环境前（pre-OS environment)，硬件组件的初始化可相对于该特定的操作系统及特定的启动加载器程序（以及对应于该启动选择模式）被定制化。

[0118] 在本实施例中，在步骤406或步骤408的DXE阶段中，执行时服务可被启动。本领域技术人员应当理解的是，在比较32位处理器模式及64位处理器模式时，由于启动执行时服务需要呼叫具有不同参数（parameters）的方法（function)，亦即会使用不同的function call，步骤405至406以及步骤405至408分别会产生不同执行时服务，其分别是被各别的操作系统使用。举例而言，与启动该启动加载器程序的执行类似，执行时服务的执行可依据对应于该启动选择模式的一或多个该指令集424、该共同组态数据427及该独特组态数据428的组合。

[0119] 请参考图7，图7示出UEFI固件的计算机处理器执行方法的流程图，其中该UEFI固件支援多个启动加载器程序的初始化程序。如图7所示，该方法300包括步骤310至370。此些步骤将在下面详细描述：

[0120] 步骤310包括以该处理器执行该UEFI固件。如图1及4所示，该UEFI固件(136,422)可被储存于一储存器420中，其中该储存器可包括非易失性随机存取储存器(Non-Volatile Random Access Memory，简称NVRAM)或快闪储存器(flash memory)。如图4所示，处理器410是耦接于该储存器420，并且会执行该UEFI固件422。

[0121] 方法300的步骤320包括：当执行该UEFI固件422时，检测该处理器的指令集的类型。如图4及5所示，当该计算机系统100通电时，执行的UEFI固件422会确认该处理器的指令集的类型。在本实施例中，为简单起见，图5已被简化以描绘该指令集是x86处理器/芯片组架构或非x86处理器/芯片组架构的指令集。本领域技术人员应当理解，任何其他类型的芯片组架构亦可以被使用。例如ARM芯片组架构也可以被使用。通过UEFI固件422确认该处理器的指令集的类型，本发明所揭露的单一封装/包装的该UEFI固件422可供UEFI固件制造商及供应商一种简单的单一编译启动加载器源代码映像(singular compiled boot up source code)，其可自动检测芯片组架构的类型并相应地初始化硬件组件。通过此方式，UEFI固件制造商只需要提供一编译映像给多个具有不同芯片组架构的不同的计算机系统。

[0122] 步骤 330 包括根据模式选择判断一启动选择模式，其中该启动选择模式是对应该侦测到的指令集的类型，以及该模式选择是对应于该 UEFI 固件执行完后需要执行的启动加载器程序的类型。在本实施例中，指令集的类型是指计算机系统 100 中该处理器的芯片组架构。举例而言，若检测到的指令集类型属于 x86_64 的芯片组架构，该启动选择模式可以是一个处理器模式在 16 位、32 位或 64 位，其对应于 16 位、32 位或 64 位指令集的向后支援 (backwards support) 的 x86_64 芯片组。该模式选择较佳地是根据从组态数据 426 列表中于共同组态数据 426 里侦测到的一预设启动变数 (default bootup variable) 产生。在一实施例中，组态数据 426 的共同组态数据 427 中的该预设启动变数可被视为一启动标记，其是代表计算机系统 100 应该要启动加载的启动加载器程序。在本实施例中，此启动标记可被设为 GUID 的变量 / 变数以使得任何在运行中的操作系统可改变预设的启动模式。通过此方式，在下次计算机系统 100 重新开机或通电时，UEFI 固件 422 的执行将会启动加载对应于该启动标记被设定的操作系统。举例而言，若使用者是使用 64 位的操作系统，并且希望能在计算机系统 100 重新启动时使用 32 位的操作系统，使用者可通过 64 位操作系统更改预设启动变量或启动标记来使得在下一次计算机系统 100 通电启动时 32 位的操作系统会被自动启动了。然而，在其它不同的实施例中，模式选择可根据使用者的输入来确定。例如，在 PEI 阶段后一控制台（例如，键盘或触碰显示器）可被初始化，以使得使用者可输入或键入一命令以指示 UEFI 固件以某个处理器模式或某个操作系统启动。在另一个实施例中，此使用者的指令输入可由处理器侦测并处理，并且处理器可相对的设定该启动标记。通过此方式，由于 UEFI 固件尚未在 pre-OS 环境下以任何特定的处理器模式进行初始化，该 UEFI 固件有时间及机会先检查该启动标记并相应的启动欲启动的操作系统。

[0123] 方法 300 的步骤 340 包括以该处理器从储存器终于该 UEFI 固件的多个组态数据其中之一取得该共同组态数据。在本实施例中，组态数据 426 的该共同组态数据 427 包括为所有操作系统进行 pre-OS 环境的初始化的共同变量。例如，若有一特定的标志图像需要在 POST 期间显示（无论是哪个操作系统将被启动），该标志图像的组态变量会被认为是共同于所有启动加载器程序。因此，此共同组态变量将会被储存于该共同组态数据 427 中，而通过此方式可将此组态变量的复本移除掉，以致使 UEFI 固件的总大小可被降低（例如，客户可有更多储存空间在原始的 UEFI 固件上增加新功能等用途）。换句话说，通过将多个 UEFI 固件映像包装为单一个映像包装并且删除掉副本的共同变量，可减少储存空间的使用空间以利于其他用途。

[0124] 步骤 350 包括以该处理器从该储存器中于对应该启动选择模式的 UEFI 固件的该多个组态数据里取得一独特组态数据，其中该独特组态数据包括处理器指令集 (processor instruction sets) 及执行时服务 (runtime services)。在本实施例中，对应于启动选择模式的独特组态数据 427 是从组态数据 426 取得。换言之，不属共同于所有操作系统且为独特于某个启动选择模式的组态变量才会被取得。在另一个实施例中，独特组态数据 427 可包括处理器的指令集，并且亦可具有指示供执行时期服务给操作系统。

[0125] 步骤 360 包括根据启动选择模式产生一启动加载器地址。在本实施例中，UEFI 固件 422 的处理器执行将会产生对应于该启动选择模式的启动加载器地址。启动加载器地址较佳为位于储存器中（例如，储存器 440）具有一启动加载器程序可启动加载欲执行的操作系统的一地址。举例而言，若启动选择指的是 32 位的处理器模式，该处理器较佳将产生指

向可以启动相对的 32 位操作系统的该启动加载器程序的地址。

[0126] 步骤 370 包括依据该共同组态数据及该独特组态数据，执行位于对应该启动加载器地址的储存器中的该启动加载器程序。在本实施例中，为了能定制硬件设备的初始化于指定的处理器模式（指令集）以致使能符合相对地操作系统，取得的该共同组态数据 427 将与对应于该特定的启动加载器程序的独特组态数据 428 结合在一起。通过此方式，可使得必要的 pre-OS 环境被完全初始化。换句话说，通过将共同组态数据 427 与对应于目标的操作系统的独特组态数据 428 结合，UEFI 固件中对应该目标操作系统的处理模式的指令集可被执行以初始化任何必要的硬件组件。此外，随着独特组态数据 428 及对应于目标操作系统的指令集 I1 或 I2，UEFI 固件 422 亦可以执行执行时服务，以使得在目标操作系统已启动运行后可经由该执行时服务使用一些硬件的服务。

[0127] 本发明已由上述相关实施例加以描述，然而上述实施例仅为实施本发明的范围。必须指出的是，已公开的实施例并未限制本发明的范围。相反地，包含于权利要求范围的精神及范围的修改及均等设置包含于本发明的范围内。

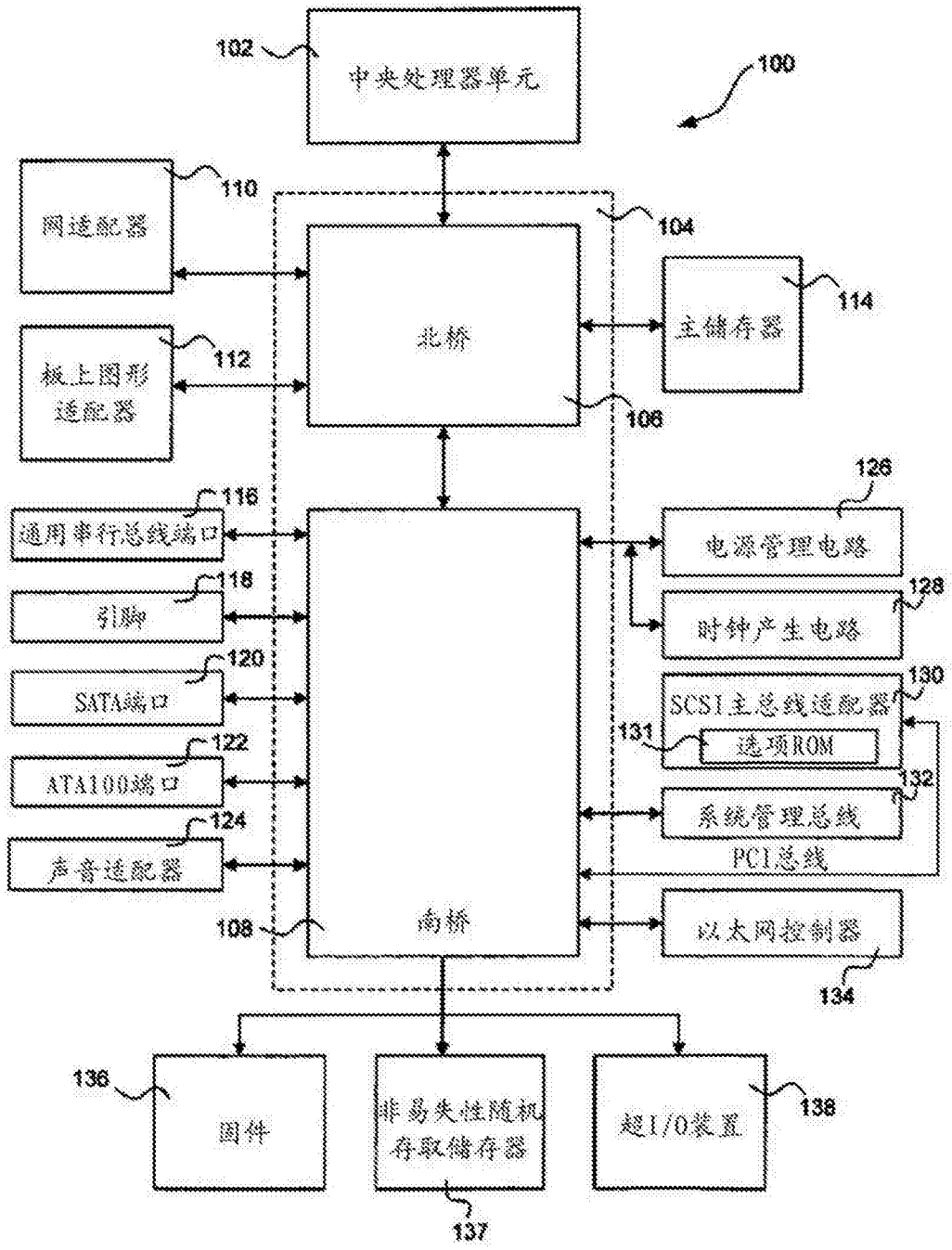

102
中央处理器单元
100
110
网适配器
104
北桥
114
主储存器
112
板上图形适配器
106
126
116
通用串行总线端口
电源管理电路
118
引脚
128
时钟产生电路
120
SATA端口
130
SCSI主总线适配器
131
选项ROM
122
ATA100端口
132
系统管理总线
124
声音适配器
PCI总线
108
南桥
以太网控制器
134
136
固件
非易失性随机存取储存器
超I/O装置
138
137

图 1

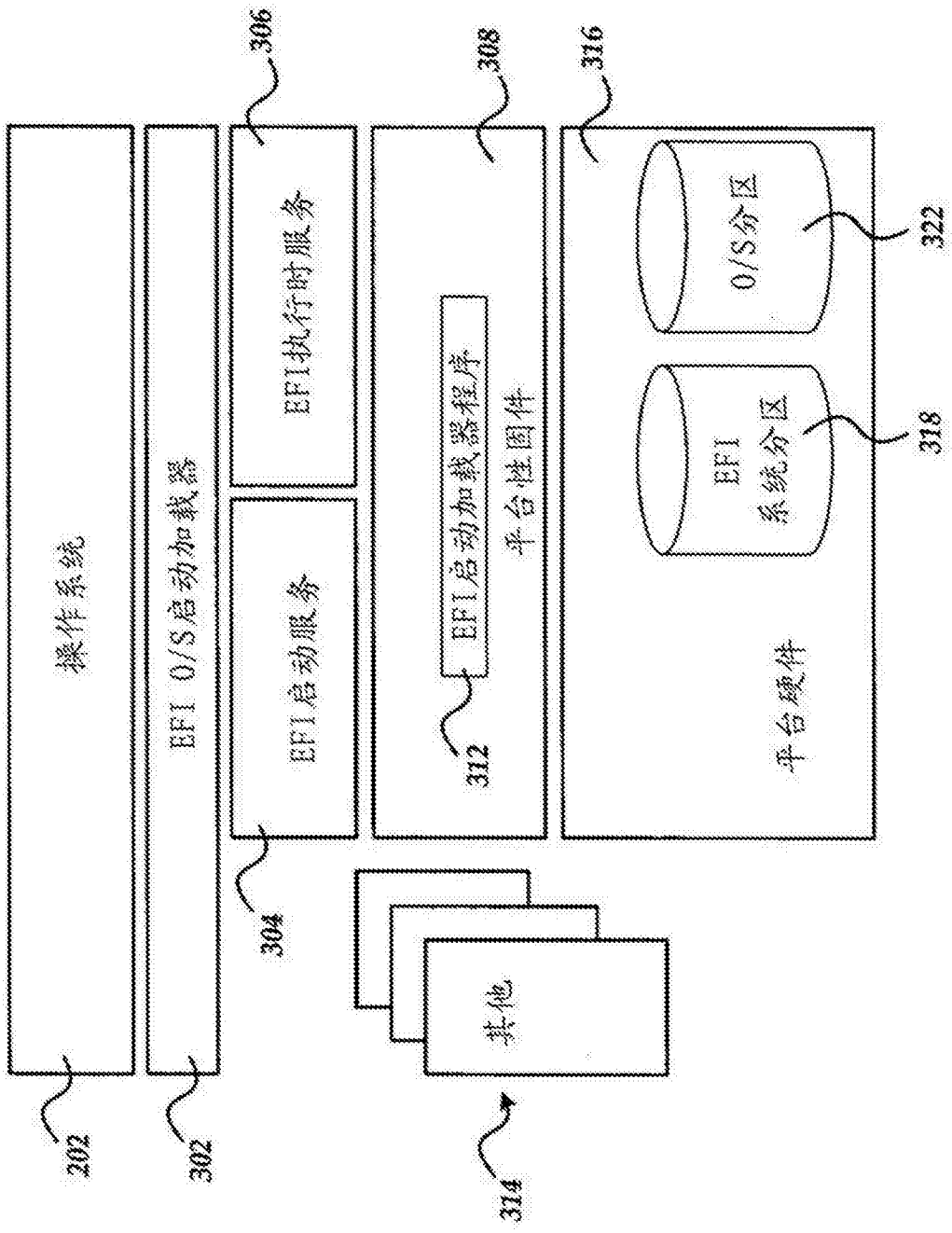

操作系统
EFI O/S启动加载器
EFI启动服务
EFI执行时服务
EFI启动加载器程序
平台性固件
平台硬件
O/S分区
EFI
系统分区
其他
202
302
304
306
308
312
314
316
318
322

图 2

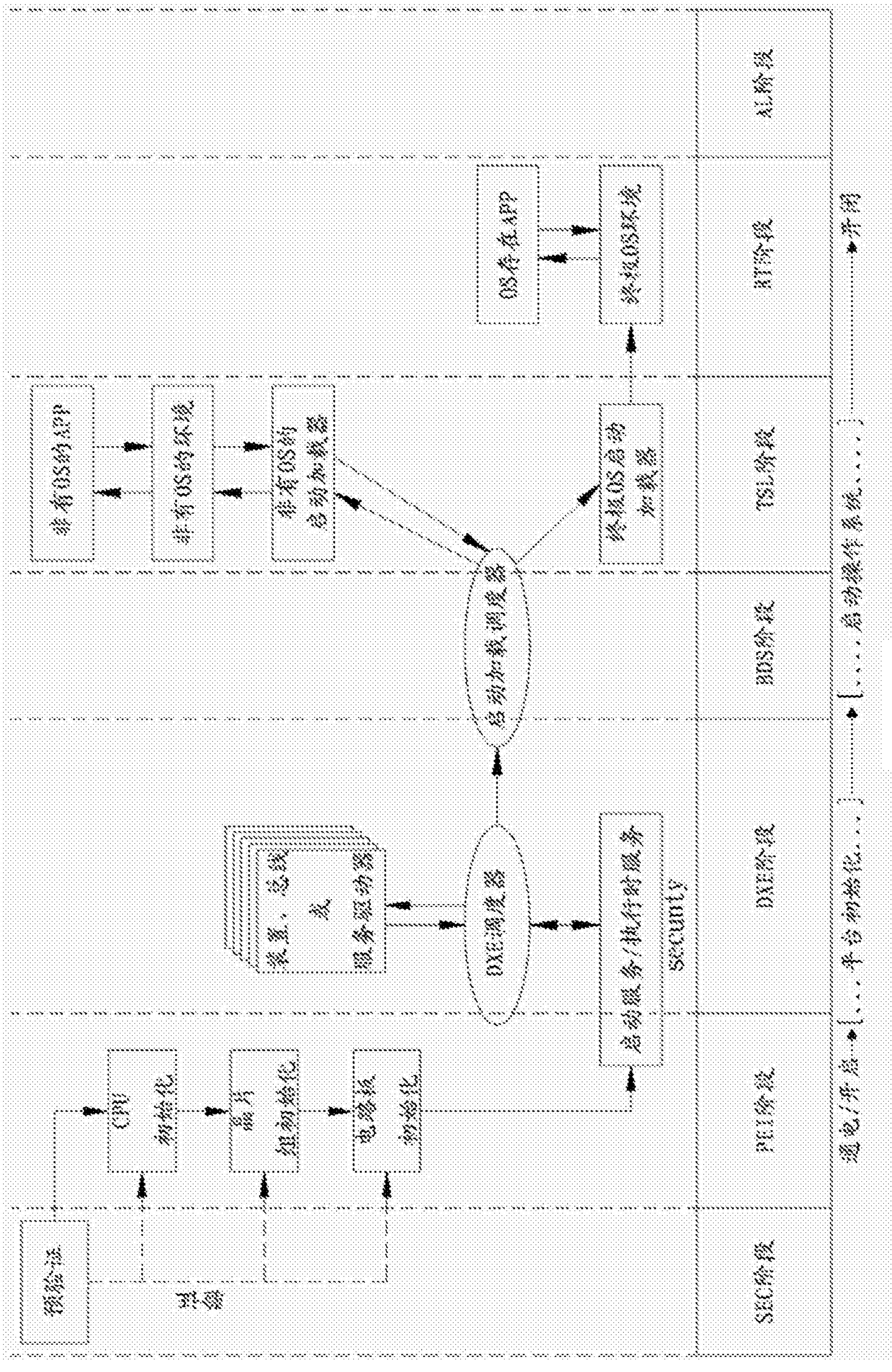

SEC阶段
PEI阶段
DXE阶段
BDS阶段
TSL阶段
RT阶段
AL阶段
预验证
CPU 初始化
DXE调度器
security
OS存在APP

图 3

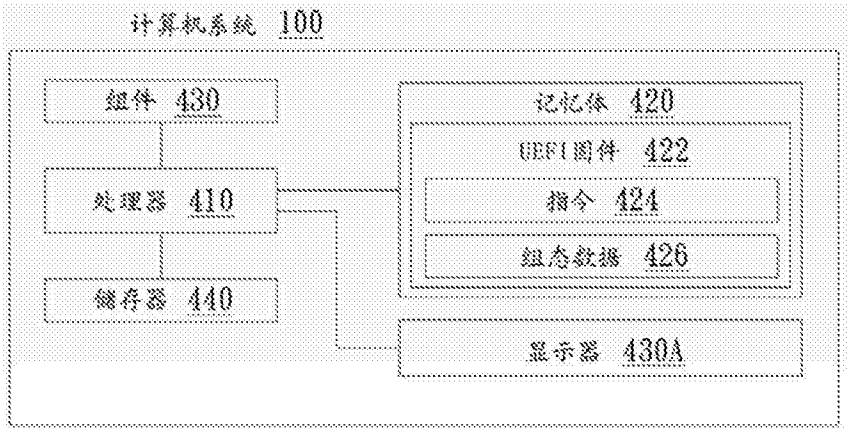

计算机系统 100
组件 430
处理器 410
储存器 440
记忆体 420
UEFI固件 422
指令 424
组态数据 426
显示器 430A

图 4

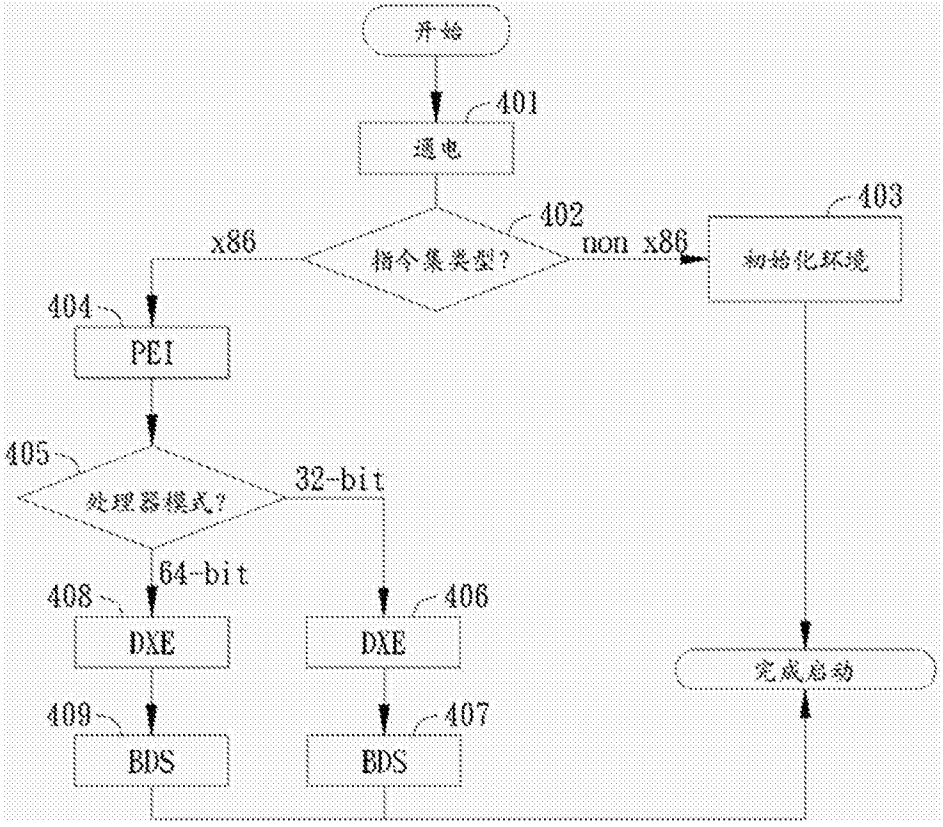

开始
401
通电
402
指令集类型?
x86
non x86
403
初始化环境
404
PEI
405
处理器模式?
32-bit
64-bit
408
DXE
406
DXE
409
BDS
407
BDS
完成启动

图 5

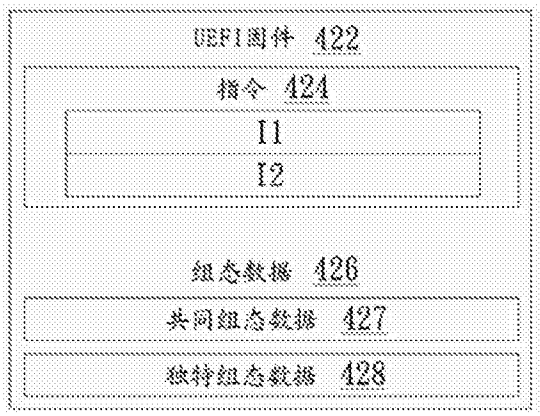

UEFI固件 422
指令 424
I1
I2
组态数据 426
共同组态数据 427
独特组态数据 428

图 6A


组态数据 426
共同组态数据 427
预设启动的OS
启动图形
独特组态数据 428
FV_MAIN1
FV_MAIN2

图 6B


300
开始
310
执行该UEFI固件
320
侦测处理器的指令的类型
330
根据一模式选择判断一启动选择模式
340
从储存器里于UEFI固件的多个组态数据中取得一共同组态数据
350
从储存器里于相对该启动选择模式的该UEFI固件的多个组态数据中取得一独特组态数据
360
相对于该启动选择模式产生一启动加载器地址
370
根据该共同组态数据及该独特组态数据，执行位于储存器中相对于该启动加载器地址表示的位于的启动加载器程序
结束

图 7