1. esp分区

EFI系统分区，即 EFI system partition，[简写](https://baike.baidu.com/item/%E7%AE%80%E5%86%99)为 ESP。ESP虽然 是一个 FAT16 或 FAT32 格式的物理分区，但是其分区标识是 EF (十六进制) 而非常规的 0E 或 0C。因此，该分区在 Windows 操作系统下一般是不可见的。支持 EFI 模式的电脑需要从 ESP 启动系统，EFI 固件可从 ESP 加载 EFI 启动程序和应用程序。

FAT分区与通用性

EFI 规范规定 ESP 使用 FAT 格式。尽管这种老旧的格式有很多局限性，但也不算一无是处。尤其开发者来说，系统中有个 FAT32 分区是很好的，因为经常使用多个系统，包括 Windows、Linux、FreeBSD，而这些系统都能安全地访问 FAT 分区，因此，可以把各个系统的启动相关的文件包括启动管理程序等放到 ESP 里，而且，这也方便在一个系统中去修改另一个系统的启动文件，至少可以把 ESP 当作各个系统的公共交换空间。

独立于操作系统

ESP是一个独立于操作系统之外的分区，操作系统被引导之后，就不再依赖它。这使得 ESP 非常适合用来存储那些系统级的维护性的工具和数据，比如：引导管理程序、驱动程序、系统维护工具、系统备份等，甚至可以在 ESP 里安装一个特殊的操作系统（SlaTaz Linux? PuppyLinux? Win PE?）。

# FAT（文件配置表）

FAT是文件配置表（英语：File Allocation Table，首字母缩略字：FAT），是一种由[微软](https://baike.baidu.com/item/%E5%BE%AE%E8%BD%AF)发明并拥有部分专利的文件系统，供MS-DOS使用，也是所有非NT核心的微软窗口使用的文件系统。

FAT文件系统考虑当时电脑性能有限，所以未被复杂化，因此几乎所有个人电脑的操作系统都支持。这特性使它成为理想的软盘和存储卡文件系统，也适合用作不同操作系统中的数据交流。现在，一般所讲的FAT专指FAT32。

但FAT有一个严重的缺点：当文件删除后写入新数据，FAT不会将文件整理成完整片段再写入，长期使用后会使文件数据变得逐渐分散，而减慢了读写速度。碎片整理是一种解决方法，但必须经常重组来保持FAT文件系统的效率。

# bios

BIOS是英文"Basic Input Output System"的[缩略词](https://baike.baidu.com/item/%E7%BC%A9%E7%95%A5%E8%AF%8D)，直译过来后中文名称就是"基本输入输出系统"。在[IBM](https://baike.baidu.com/item/IBM/9190) PC兼容系统上，是一种业界标准的[固件](https://baike.baidu.com/item/%E5%9B%BA%E4%BB%B6)[接口](https://baike.baidu.com/item/%E6%8E%A5%E5%8F%A3)。[1]  BIOS这个字眼是在1975年第一次由[CP/M](https://baike.baidu.com/item/CP%2FM)操作系统中出现。[2]  BIOS是[个人电脑](https://baike.baidu.com/item/%E4%B8%AA%E4%BA%BA%E7%94%B5%E8%84%91)启动时加载的第一个软件。

**其实，它是一组固化到**[**计算机**](https://baike.baidu.com/item/%E8%AE%A1%E7%AE%97%E6%9C%BA)**内**[**主板**](https://baike.baidu.com/item/%E4%B8%BB%E6%9D%BF)**上一个**[**ROM**](https://baike.baidu.com/item/ROM)[**芯片**](https://baike.baidu.com/item/%E8%8A%AF%E7%89%87)**上的**[**程序**](https://baike.baidu.com/item/%E7%A8%8B%E5%BA%8F)**，它保存着计算机最重要的基本输入输出的程序、开机后自检程序和系统自启动程序，它可从CMOS中读写**[**系统设置**](https://baike.baidu.com/item/%E7%B3%BB%E7%BB%9F%E8%AE%BE%E7%BD%AE)**的具体信息。** 其主要功能是为计算机提供最底层的、最直接的硬件设置和控制。此外，BIOS还向作业系统提供一些系统参数。系统硬件的变化是由BIOS隐藏，程序使用BIOS功能而不是直接控制硬件。现代作业系统会忽略BIOS提供的抽象层并直接控制硬件组件。

当今，此系统已成为一些病毒[木马](https://baike.baidu.com/item/%E6%9C%A8%E9%A9%AC/530)的目标。一旦此系统被破坏，其后果[不堪设想](https://baike.baidu.com/item/%E4%B8%8D%E5%A0%AA%E8%AE%BE%E6%83%B3)。

# EFI

[EFI](https://baike.baidu.com/item/EFI)，它是一种可扩展[固件](https://baike.baidu.com/item/%E5%9B%BA%E4%BB%B6)接口，英文名Extensible Firmware Interface 的缩写，是[英特尔](https://baike.baidu.com/item/%E8%8B%B1%E7%89%B9%E5%B0%94)一个主导个人电脑技术研发的公司推出的一种在未来的类PC的[电脑系统](https://baike.baidu.com/item/%E7%94%B5%E8%84%91%E7%B3%BB%E7%BB%9F)中替代[BIOS](https://baike.baidu.com/item/BIOS)的升级方案。同时，EFI也是电子燃料注入（系统） 英文名 electronic fuel injection (system) 的缩写，就是利用各种传感器检测发动机的各种状态，经电脑的判断、计算，使发动机在不同工况下，均能获得合适浓度的可燃混合气的一种系统。

### 含义EFI的产生

BIOS技术的兴起源于IBM PC/AT机器的流行以及第一台由[康柏](https://baike.baidu.com/item/%E5%BA%B7%E6%9F%8F)公司研制生产的“克隆”PC。在PC启动的过程中，BIOS担负着[初始化](https://baike.baidu.com/item/%E5%88%9D%E5%A7%8B%E5%8C%96)硬件，检测硬件功能，以及引导操作系统的责任，在早期，BIOS还提供一套运行时的服务程序给操作系统及应用程序使用。BIOS程序存放于一个掉电后内容不会丢失的[只读存储器](https://baike.baidu.com/item/%E5%8F%AA%E8%AF%BB%E5%AD%98%E5%82%A8%E5%99%A8)中，系统加电时处理器的第一条指令的地址会被定位到BIOS的[存储器](https://baike.baidu.com/item/%E5%AD%98%E5%82%A8%E5%99%A8)中，便于使初始化程序得到执行。

众所周知，[英特尔](https://baike.baidu.com/item/%E8%8B%B1%E7%89%B9%E5%B0%94)在近二十年来引领以[x86](https://baike.baidu.com/item/x86)系列处理器为基础的PC技术潮流，它的产品如CPU，芯片组等在PC生产线中占据绝对领导的位置。因此，不少人认为这一举动显示了英特尔公司欲染指[固件](https://baike.baidu.com/item/%E5%9B%BA%E4%BB%B6)产品市场的野心。事实上，EFI技术源于英特尔[安腾处理器](https://baike.baidu.com/item/%E5%AE%89%E8%85%BE%E5%A4%84%E7%90%86%E5%99%A8)(Itanium)平台的推出。安腾处理器是英特尔瞄准服务器高端市场投入近十年研发力量设计产生的与x86系列完全不同的64位新架构。在x86系列处理器进入32位的时代，由于兼容性的原因，新的处理器(i80386)保留了16位的运行方式(实模式)，此后多次处理器的升级换代都保留了这种运行方式。甚至在含[64位扩展技术](https://baike.baidu.com/item/64%E4%BD%8D%E6%89%A9%E5%B1%95%E6%8A%80%E6%9C%AF)的[至强](https://baike.baidu.com/item/%E8%87%B3%E5%BC%BA)系列处理器中，处理器加电启动时仍然会切换到16位的实模式下运行。[英特尔](https://baike.baidu.com/item/%E8%8B%B1%E7%89%B9%E5%B0%94)将这种情况归咎于BIOS技术的发展缓慢。自从PC[兼容机](https://baike.baidu.com/item/%E5%85%BC%E5%AE%B9%E6%9C%BA)厂商通过净室的方式复制出第一套BIOS[源程序](https://baike.baidu.com/item/%E6%BA%90%E7%A8%8B%E5%BA%8F)，BIOS就以16位汇编代码，寄存器参数调用方式，[静态链接](https://baike.baidu.com/item/%E9%9D%99%E6%80%81%E9%93%BE%E6%8E%A5)，以及1MB以下内存固定[编址](https://baike.baidu.com/item/%E7%BC%96%E5%9D%80)的形式存在了十几年。虽然由于各大BIOS厂商的努力，有许多新元素添加到产品中，如PnP BIOS，ACPI，传统USB设备支持等等，但BIOS的根本性质没有得到任何改变。这迫使英特尔在开发更新的处理器时，都必须考虑加进使效能大大降低的兼容模式。有人曾打了一个比喻：这就像保时捷新一代的全自动档跑车被人生套上去一个蹩脚的挂档器。

然而，[安腾处理器](https://baike.baidu.com/item/%E5%AE%89%E8%85%BE%E5%A4%84%E7%90%86%E5%99%A8)并没有这样的顾虑，它是一个新生的处理器架构，系统[固件](https://baike.baidu.com/item/%E5%9B%BA%E4%BB%B6)和操作系统之间的接口都可以完全重新定义。并且这一次，[英特尔](https://baike.baidu.com/item/%E8%8B%B1%E7%89%B9%E5%B0%94)将其定义为一个可扩展的，标准化的固件接口规范，不同于传统BIOS的固定的，缺乏文档的，完全基于经验和晦涩约定的一个事实标准。基于EFI的第一套系统产品的出现至今已经有五年的时间，如今，英特尔试图将成功运用在高端服务器上的技术推广到市场占有率更有优势的PC产品线中，并承诺在2006年间会投入全力的技术支持。

**比较EFI BIOS和Legacy BIOS**

一个显著的区别就是EFI BIOS是用模块化，C语言风格的参数[堆栈](https://baike.baidu.com/item/%E5%A0%86%E6%A0%88)传递方式，[动态链接](https://baike.baidu.com/item/%E5%8A%A8%E6%80%81%E9%93%BE%E6%8E%A5)的形式构建的系统，较Legacy BIOS而言更易于实现，容错和纠错特性更强，缩短了系统研发的时间。它运行于32位或64位模式，乃至未来增强的处理器模式下，突破传统16位代码的[寻址](https://baike.baidu.com/item/%E5%AF%BB%E5%9D%80)能力，达到处理器的最大寻址。它利用加载EFI驱动的形式，识别及操作硬件，不同于BIOS利用挂载实模式中断的方式增加硬件功能。后者必须将一段类似于驱动的16位代码，放置在固定的0x000C0000至0x000DFFFF之间存储区中，运行这段代码的初始化部分，它将挂载实模式下约定的[中断向量](https://baike.baidu.com/item/%E4%B8%AD%E6%96%AD%E5%90%91%E9%87%8F)向其他程序提供服务。例如，VGA图形及[文本输出](https://baike.baidu.com/item/%E6%96%87%E6%9C%AC%E8%BE%93%E5%87%BA)中断(INT 10h)，磁盘存取中断服务(INT 13h)等等。由于这段存储空间有限(128KB)，Legacy BIOS对于所需放置的驱动代码大小超过空间大小的情况无能为力。另外，Legacy BIOS的硬件服务程序都以16位代码的形式存在，这就给运行于增强模式的操作系统访问其服务造成了困难。因此Legacy BIOS提供的服务在现实中只能提供给操作系统引导程序或[MS-DOS](https://baike.baidu.com/item/MS-DOS)类操作系统使用。而EFI系统下的驱动并不是由可以直接运行在CPU上的代码组成的，而是用EFI Byte Code(EBC)编写而成的。这是一组专用于EFI驱动的虚拟[机器指令](https://baike.baidu.com/item/%E6%9C%BA%E5%99%A8%E6%8C%87%E4%BB%A4)，必须在EFI驱动运行环境(Driver Execution Environment，或DXE)下被解释运行。这就保证了充分的向下兼容性，打个比方说，一个带有EFI驱动的扩展设备，既可以将其安装在[安腾处理器](https://baike.baidu.com/item/%E5%AE%89%E8%85%BE%E5%A4%84%E7%90%86%E5%99%A8)的系统中，也可以安装于支持EFI的新PC系统中，而它的EFI驱动不需要重新编写。这样就无需对系统升级带来的兼容性因素作任何考虑。另外，由于EFI驱动开发简单，所有的PC部件提供商都可以参与，情形非常类似于[现代操作系统](https://baike.baidu.com/item/%E7%8E%B0%E4%BB%A3%E6%93%8D%E4%BD%9C%E7%B3%BB%E7%BB%9F)的开发模式，这个开发模式曾使Windows在短短的两三年时间内成为功能强大，性能优越的操作系统。基于EFI的驱动模型可以使EFI系统接触到所有的硬件功能，在操作操作系统运行以前浏览万维网站不再是天方夜谭，甚至实现起来也非常简单。这对基于传统BIOS的系统来说是件不可能的任务，在BIOS中添加几个简单的USB设备支持都曾使很多BIOS设计师痛苦万分，更何况除了添加对无数网络硬件的支持外，还得凭空构建一个16位模式下的[TCP/IP协议](https://baike.baidu.com/item/TCP%2FIP%E5%8D%8F%E8%AE%AE)栈。

一些人认为BIOS只不过是由于兼容性问题遗留下来的无足轻重的部分，不值得为它花费太大的升级努力。而反对者认为，当BIOS的出现制约了PC技术的发展时，必须有人对它作必要的改变。

### EFI和操作系统

EFI在概念上非常类似于一个低阶的操作系统，并且具有操控所有硬件资源的能力。不少人感觉它的不断发展将有可能代替现代的操作系统。事实上，EFI的缔造者们在第一版规范出台时就将EFI的能力限制于不足以威胁操作系统的统治地位。首先，它只是硬件和预启动软件间的接口规范；其次，EFI环境下不提供中断的访问机制，也就是说每个EFI驱动程序必须用轮询的方式来检查硬件状态，并且需要以解释的方式运行，较操作系统下的驱动效率更低；再则，EFI系统不提供复杂的[存储器](https://baike.baidu.com/item/%E5%AD%98%E5%82%A8%E5%99%A8)保护功能，它只具备简单的[存储器管理](https://baike.baidu.com/item/%E5%AD%98%E5%82%A8%E5%99%A8%E7%AE%A1%E7%90%86)机制，具体来说就是指运行在x86处理器的段保护模式下，以最大[寻址](https://baike.baidu.com/item/%E5%AF%BB%E5%9D%80)能力为限把存储器分为一个平坦的段，所有的程序都有权限存取任何一段位置，并不提供真实的保护服务。当EFI所有组件加载完毕时，系统可以开启一个类似于操作系统Shell的命令解释环境，在这里，用户可以调入执行任何EFI应用程序，这些程序可以是硬件检测及除错软件，引导管理，设置软件，操作系统引导软件等等。理论上来说，对于EFI应用程序的功能并没有任何限制，任何人都可以编写这类软件，并且效果较以前MS-DOS下的软件更华丽，功能更强大。一旦引导软件将控制权交给操作系统，所有用于引导的服务代码将全部停止工作，部分运行时代服务程序还可以继续工作，以便于操作系统一时无法找到特定设备的驱动程序时，该设备还可以继续被使用。

### EFI的组成

一般认为，EFI由以下几个部分组成：

1. Pre-EFI初始化模块
2. EFI驱动执行环境
3. EFI驱动程序
4. 兼容性支持模块(CSM)
5. EFI高层应用
6. GUID 磁盘分区

### 各部分功能

在实现中，EFI初始化模块和驱动执行环境通常被集成在一个[只读存储器](https://baike.baidu.com/item/%E5%8F%AA%E8%AF%BB%E5%AD%98%E5%82%A8%E5%99%A8)中。Pre-EFI初始化程序在系统开机的时候最先得到执行，它负责最初的CPU，主桥及[存储器](https://baike.baidu.com/item/%E5%AD%98%E5%82%A8%E5%99%A8)的初始化工作，紧接着载入EFI驱动执行环境(DXE)。当DXE被载入运行时，系统便具有了枚举并加载其他EFI驱动的能力。在基于PCI架构的系统中，各PCI桥及PCI适配器的EFI驱动会被相继加载及初始化；这时，系统进而枚举并加载各桥接器及适配器后面的各种总线及[设备驱动程序](https://baike.baidu.com/item/%E8%AE%BE%E5%A4%87%E9%A9%B1%E5%8A%A8%E7%A8%8B%E5%BA%8F)，周而复始，直到最后一个设备的驱动程序被成功加载。正因如此，EFI驱动程序可以放置于系统的任何位置，只要能保证它可以按顺序被正确枚举。例如一个具PCI[总线接口](https://baike.baidu.com/item/%E6%80%BB%E7%BA%BF%E6%8E%A5%E5%8F%A3)的ATAPI大容量存储适配器，其EFI驱动程序一般会放置在这个设备的符合PCI规范的扩展[只读存储器](https://baike.baidu.com/item/%E5%8F%AA%E8%AF%BB%E5%AD%98%E5%82%A8%E5%99%A8)(PCI Expansion ROM)中，当PCI总线驱动被加载完毕，并开始枚举其子设备时，这个存储适配器旋即被正确识别并加载它的驱动程序。部分EFI驱动程序还可以放置在某个磁盘的EFI专用分区中，只要这些驱动不是用于加载这个磁盘的驱动的必要部件。在EFI规范中，一种突破传统MBR[磁盘分区](https://baike.baidu.com/item/%E7%A3%81%E7%9B%98%E5%88%86%E5%8C%BA)结构限制的GUID磁盘分区系统(GPT)被引入，新结构中，磁盘的分区数不再受限制(在MBR结构下，只能存在4个[主分区](https://baike.baidu.com/item/%E4%B8%BB%E5%88%86%E5%8C%BA))，并且分区类型将由GUID来表示。在众多的分区类型中，EFI[系统分区](https://baike.baidu.com/item/%E7%B3%BB%E7%BB%9F%E5%88%86%E5%8C%BA)可以被EFI系统存取，用于存放部分驱动和应用程序。很多人担心这将会导致新的安全性因素，因为EFI系统比传统的BIOS更易于受到计算机病毒的攻击，当一部分EFI驱动程序被破坏时，系统有可能面临无法引导的情况。实际上，系统引导所依赖的EFI驱动部分通常都不会存放在EFI的GUID分区中，即使分区中的驱动程序遭到破坏，也可以用简单的方法得到恢复，这与操作系统下的驱动程序的存储习惯是一致的。CSM是在x86平台EFI系统中的一个特殊的模块，它将为不具备EFI引导能力的操作系统提供类似于传统BIOS的系统服务。

### EFI的发展

[英特尔](https://baike.baidu.com/item/%E8%8B%B1%E7%89%B9%E5%B0%94)无疑是推广EFI的积极因素，由于业界对其认识的不断深入，更多的厂商正投入这方面的研究。包括英特尔，AMD在内的一些PC生产厂家联合成立了联合可扩展[固件](https://baike.baidu.com/item/%E5%9B%BA%E4%BB%B6)接口论坛，它将在近期推出第一版规范。这个组织将接手规划EFI发展的重任，并将英特尔的EFI框架解释为这个规范的一个具体实现。另外，各大BIOS提供商如Phoenix, AMI等，原先被认为是EFI发展的阻碍力量，不断的推出各自的解决方案。分析人士指出，这是由于BIOS厂商在EFI架构中重新找到了诸如Pre-EFI启动环境之类的市场位置，然而，随着EFI在PC系统上的成功运用，以及英特尔新一代[芯片组](https://baike.baidu.com/item/%E8%8A%AF%E7%89%87%E7%BB%84)的推出，这一部分市场份额将会不出意料的在英特尔的掌控之中。

# UEFI

新型[UEFI](https://baike.baidu.com/item/UEFI)，全称“统一的可扩展固件接口”(Unified Extensible Firmware Interface)， 是一种详细描述类型接口的标准。这种接口用于操作系统自动从预启动的操作环境，加载到一种操作系统上。

可扩展固件接口（Extensible Firmware Interface，EFI）是 Intel 为 PC 固件的[体系结构](https://baike.baidu.com/item/%E4%BD%93%E7%B3%BB%E7%BB%93%E6%9E%84)、接口和服务提出的建议标准。其主要目的是为了提供一组在 OS 加载之前（启动前）在所有平台上一致的、正确指定的启动服务，被看做是有近20多年历史的 BIOS 的继任者。

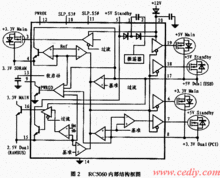
UEFI是由EFI1.10为基础发展起来的，它的所有者已不再是Intel，而是一个称作Unified EFI Form的国际组织。

ACPI

ACPI表示高级配置和电源管理接口（Advanced Configuration and Power Management Interface）。对于Windows2000，ACPI定义了Windows 2000、BIOS和系统硬件之间的新型工作接口。这些新接口包括允许Windows 2000控制电源管理和设备配置的机制。

**高级配置与电源接口**（**Advanced Configuration and Power Interface**），简称**ACPI**，1997年由Intel、Microsoft、Toshiba 所共同制定提供操作系统应用程序管理所有电源管理接口。2000年8月推出 ACPI 2.0规格。2004年9月推出 ACPI 3.0规格。2009年6月16日则推出 ACPI 4.0规格。2011年12月推出ACPI5.0规格。

## 六种状态

[](https://baike.baidu.com/pic/ACPI/299421/0/027a45b58edbeeab36d3cabc?fr=lemma&ct=single)ACPI开关控制器

分别是S0到S5，它们代表的含义分别是：

S0--实际上这就是我们平常的工作状态，所有设备全开，功耗一般会超过80W；

S1--也称为POS（Power on Suspend），这时除了通过CPU时钟控制器将CPU关闭之外，其他的部件仍然正常工作，这时的功耗一般在30W以下；（其实有些CPU[降温软件](https://baike.baidu.com/item/%E9%99%8D%E6%B8%A9%E8%BD%AF%E4%BB%B6)就是利用这种工作原理）

S2--这时CPU处于停止运作状态，总线时钟也被关闭，但其余的设备仍然运转；

S3--这就是我们熟悉的STR（Suspend to RAM），这时的功耗不超过10W；

S4--也称为STD（Suspend to Disk），这时系统主电源关闭，[硬盘存储](https://baike.baidu.com/item/%E7%A1%AC%E7%9B%98%E5%AD%98%E5%82%A8)S4前数据信息，所以S4是比S3更省电状态.

S5--这种状态是最干脆的，就是连电源在内的所有设备全部关闭，即关机（shutdown），功耗为0。

我们最常用到的是S3状态，即Suspend to RAM（挂起到内存）状态，简称STR。顾名思义，STR就是把系统进入STR前的工作状态数据都存放到内存中去。在STR状态下，电源仍然继续为内存等最必要的设备供电，以确保数据不丢失，而其他设备均处于关闭状态，系统的耗电量极低。一旦我们按下Power按钮（主机电源开关），系统就被唤醒，马上从内存中读取数据并恢复到STR之前的工作状态。内存的读写速度极快，因此我们感到进入和离开STR状态所花费的时间不过是几秒钟而已；而S4状态，即STD（挂起到硬盘）与STR的原理是完全一样的，只不过数据是保存在硬盘中。由于硬盘的读写速度比内存要慢得多，因此用起来也就没有STR那么快了。STD的优点是只通过软件就能实现，比如Windows 2000就能在不支持STR的硬件上实现STD。