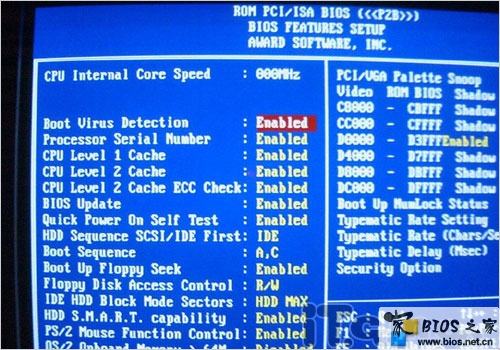
**BIOS最后的璀璨**

BIOS，几乎和PC有着同样的寿命，当年康柏第一台“克隆”PC诞生的时候，它为了简化启动的设置，引入了固化程序的概念，在启动时负责将PC初始化，然后再将控制权交给磁盘上的操作系统。而今天，“康柏”这个品牌已经消失，而BIOS却作为无心插柳柳成荫之作，延续至今。

BIOS伴随了我们十几年，在这么长的日子里，硬件升了一代又一代，电脑换了一台又一台，唯一不变的，就是BIOS。BIOS默默伴随着我们这帮从刚学会打ABCD的毛头孩子长大成人，当我们都变了，它却还是它最初的模样。  
风华已去，佳人已老，BIOS在十几年的守护中，一步步逐渐落后于硬件的发展，趋于落寞，垂垂老暮。BIOS在PC启动时，将PC初始化，然后控制权交给磁盘上的操作系统，在后面的阶段，用户的感觉是在通过操作系统直接和硬件对话，可实际上，操作系统想要与硬件进行沟通，仍然必须通过BIOS。



**我们熟悉的BIOS操作界面**

BIOS的全称是Basic Input/Output System，中文名是基本输入输出系统。BIOS即是操作系统和计算机硬件之间通讯的桥梁，更是充当翻译的角色，从DOS时代起，微软的操作系统一直都是建立在“中断”这个概念上的，程序的切换依靠中断，系统的开关依靠中断，甚至我们按下了机箱上“Reset”键强制重启电脑，也还是中断在后台的作用。为了延续整套的16位中断系统，无论是CPU开发还是软件升级，都得考虑中断模式。

在x86系列处理器进入32位时代后，由于兼容性的原因，新的处理器保留了16位的运行方式，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位扩展技术的至强系列处理器中，处理器加电启动时仍然会切换到16位的实模式下运行。BIOS程序以16位汇编代码、寄存器参数调用方式、静态链接以及1MB以下内存固定编址的形式存在了十几年，虽然各大BIOS厂商近年来努力得对其进行改进，加入了许多新元素到产品中，如ACPI、USB支持等，但BIOS的根本性质没有得到任何改变，16位的运行工作环境是其最为致命的缺点。

现有的BIOS不但在工作方式存在令人不满之处，在工作能力上，也令人颇有微词。BIOS发展到现在，用来存放BIOS程序的芯片最大不过2Mb，换成实际字节就是256KB，面对这个数值，即使你想为BIOS编写一些新的功能，BIOS芯片中也不会有足够的空间让你写入。这也是BIOS这十几年来一直停滞不前的原因之一。

所以BIOS经过了这些年的辉煌期，已经逐渐脱离了时代的发展，成为了PC功能和性能进一步提升的瓶颈，只有寻求BIOS的接任者。而BIOS，必将在璀璨光环的环绕中，落下帷幕，成为历史的记录。

**EFI接过接力棒**

EFI的英文全称是Extensible Firmware Interface，中文名是可扩展固件接口，早在2006年的上半年，Intel曾经在IDF上进行过EFI的演示。要使用EFI系统，必须主板和操作系统都支持EFI功能，目前支持EFI功能的操作系统有Mac OS X、Vista和Server 2003。



EFI在开机时的作用和BIOS一样，就是初始化PC，但在细节上却又不一样。BIOS对PC的初始化，只是按照一定的顺序对硬件通电，简单地检查硬件是否能工作，而EFI不但检查硬件的完好性，还会加载硬件在EFI中的驱动程序，不用操作系统负责驱动的加载工作。 EFI的最革命之处，是颠覆了BIOS的界面概念，让操作界面和Windows一样易于上手。在EFI的操作界面中，鼠标成为了替代键盘的输入工具，各功能调节的模块也做的和Windows程序一样，可以说，EFI就是一个小型化的Windows系统。

对于操作系统来说，如果主板使用的是BIOS，那么操作系统就必须面对所有的硬件，大到主板显卡，小到鼠标键盘，每次重装系统或者系统升级，都必须手动安装新的驱动，否则硬件很可能无法正常工作。而基于EFI的主板则方便很多，因为EFI架构使用的驱动基于EFI Byte Code。EFI Byte Code有些类似于Java的中间代码，并不由CPU直接执行操作，而是需要EFI层进行翻译。对于不同的操作系统来说，EFI将硬件层很好地保护了起来，所有操作系统看到的，都只是EFI留给EFI Byte Code的程序接口，而EFI Byte Code又直接和Windows的API联系，这就意味着无论操作系统是Windows还是Linux，只要有EFI Byte Code支持，只需要一份驱动程序就能吃遍所有操作系统平台。

      更为神奇的是，EFI Byte Code驱动还能绕过操作系统，直接安装在EFI环境中，这样对硬件的控制就由EFI层负责，EFI向操作系统直接提供硬件操作的接口，不需要操作系统再调用驱动。这种方式的优点是不需要进入操作系统，只需要进入EFI界面，更新驱动程序就可以完成，而且不需要对每一个操作系统进行驱动升级，只要EFI界面中升级一次，所有上层的操作系统都可以直接调用新的EFI接口。

EFI在开机之始就能够驱动所有的硬件，网络当然也不会例外，所以在EFI的操作界面中，程序可以直接连接上互联网，向外界求助操作系统的维修信息或者在线升级驱动程序。

**更方便的编程方式**

有人会问：既然EFI功能那么强大，那它存放在什么地方？是存放在原来的BIOS芯片中吗？答案当然是No。BIOS芯片只有256KB，远远不够EFI使用。EFI是以小型磁盘分区的形式存放在硬盘上的。EFI的安装，必须在支持EFI功能的主板上，使用光驱引导系统，然后对磁盘进行EFI化的处理，这个处理的过程，主要就是划分EFI独用的磁盘空间。

EFI的存储空间大约为50MB到100MB，具体视驱动文件多少而定。在这部分空间中，包含以下几个部分：

1. Pre-EFI初始化模块

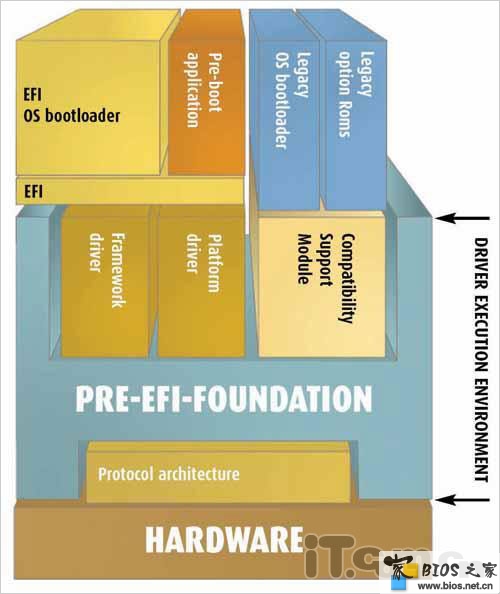
2. EFI驱动执行环境

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



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、北桥、南桥、内存和硬盘的初始化工作，紧接着载入EFI驱动。当EFI驱动程序被载入运行后，系统便具有控制所有硬件的能力。在EFI规范中，一种突破传统MBR磁盘分区结构限制的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），并且分区类型将由GUID来表示。在众多的分区类型中，EFI系统分区可以被EFI系统存取，用于存放部分驱动和应用程序。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。

由于EFI驱动开发简单，所有的硬件厂商都可以参与，为自家的硬件定制最为合适的驱动。基于EFI的驱动模型可以使EFI系统接触到所有的硬件功能，不进入操作操作系统就浏览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。

**EFI是否固若金汤？**

很多人担心EFI这种开放的模式将会导致新的安全隐患，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，因为只读芯片中的EFI代码足够用来引导计算机从光驱启动，此时插入EFI的安装盘，对EFI的系统存储区域进行修复或者覆盖安装，就能将PC恢复到正常。而且这个修复过程对操作系统来说，等于是从两台配置一模一样配置机器中的一台转移到另一台，并不会出现需要重新识别硬件的情况。 EFI在概念上非常类似于一个低等级的操作系统，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，它只是硬件和操作系统间的接口规范；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低得多；第三，EFI系统不提供复杂的存储器保护功能，它只具备简单的存储器管理机制，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。

EFI的命令行控制模式

EFI的设计架构中，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的程序只限于类似Java伪执行文件的能力，并没有直接访问磁盘所有资源的能力，而且在进入操作系统后的大多数情况下，EFI部分的代码都进入沉睡模式，即使有针对EFI的病毒，也无法造成进一步的影响。

**和BIOS说再见**

EFI的出现，可以说是充分弥补了BIOS原有的不足。因为BIOS过于自信芯片的安全，所以当遇上CIH病毒，启动机制也被完全破坏。而EFI将主要程序文件放在了硬盘上，被破坏了还可以使用光盘进行维修，对操作系统而言，这种“破坏-维修”的方式是完全透明的，不会影响操作系统的使用。虽然看起来EFI更容易受到损坏，但也更为易于修复。



BIOS在经历了十几年发展之后，也终于走到了尽头，外观上的落后、功能上的羸弱、安全上的薄弱、性能上的不足，都严重制约着它的进一步发展。虽然在这些日子里，BIOS能够带给我们基本的功能，但PC要进步，就必须寻求更高更好的技术。

EFI作为BIOS的替代者，无论是界面、功能还是安全性，都要远远高于后者，而且作为未来主板的趋势所向，EFI上能执行的程序会越来越多，EFI能够提供的基本功能也就越来越强。今天，微星在CES展会上展示了EFI主板的强大，因为和普通BIOS主板在设计难度以及生产兼容性上并不冲突，所以可以相信，拥有诸多优点的EFI会取代BIOS，让PC越来越易于使用。



微星支持EFI技术的P35 Neo3主板



EFI BIOS界面



**编辑总结**：Intel作为EFI大力的推广者和制定者，能看到EFI逐渐从服务器平台走向桌面级市场，其中辛酸甘苦只有自己才知道。从初期厂商对EFI的概念毫无兴趣，到今天各大BIOS提供商如Phoenix, AMI等，原先被认为是EFI发展的阻碍力量，现在也不断的推出各自的解决方案。支持EFI功能的主板也逐渐退出。一切的一切，都似乎预示着我们可以和BIOS说声再见，让技术的进步来记录历史。

 一些常见的关于BIOS/EFI的问题以及我的简短回答:  
1) BIOS一般有多大?  
传统bios(以后说legacy bios)一般都是512KB,而早期的EFI bios也是512KB.现在EFI基本上是1MB了.  
  
2) BIOS用什么工具开发?  
legacy bios一般用MASM 6.11开发,同时还会配上一些厂商自己写的buid tools. EFI则使用Viusal Studio.NET 2003以及MASM 6.11开发(没想到吧~)  
  
3) EFI boot是怎么一回事?  
EFI有自己独特的boot方式,完全抛弃掉了传统的0磁道0扇区的MBR概念.EFI的boot方式与文件系统息息相关.过去的legacy bios由于不带文件系统,不得已选择从硬盘上特定空间装载程序的办法,而EFI则附带了完整的文件系统支持,所以不再对硬盘有特定的要求,EFI下的操作系统加载程序事实上存储在boot\ia32\bootia32.efi文件里.(假定是IA32架构).这是一个EFI应用程序.  
  
  
下面是一些深入学习bios的资源汇总:  
1. BIOS Boot Specification  
业内一般叫BBS,详细描述bios启动时必须要做的所有事情,如何区分启动设备,如何选择启动设备等等.  
http://www.phoenix.com/NR/rdonlyres/56E38DE2-3E6F-4743-835F-B4A53726ABED/0/specsbbs101.pdf  
  
 2. UEFI Specification  
 UEFI规范,详细描述了UEFI bios必须支持的接口.以及UEFI bios的模型,提供的服务等等. 开发UEFI必备的.  
http://www.uefi.org  
  
 3. Ralf Brown's Interrupt List  
这个人似乎就一辈子都都在收集中断的东西,对legacy bios学习很有用.  
http://www.ctyme.com/rbrown.htm  
  
 4. El Torito CD-ROM Boot  
描述了bios如何从光驱上boot的细节.  
http://www.phoenix.com/NR/rdonlyres/98D3219C-9CC9-4DF5-B496-A286D893E36A/0/specscdrom.pdf  
  
 5. USB Specification  
 USB设备规范  
http://www.usb.org  
  
 6. Plug-and-Play Specifications  
 MS的PnP规范  
http://www.microsoft.com/hwdev/tech/pnp/default.asp  
  
 7. BIOS Writer's Guide  
 bios开发的圣经,由cpu厂商给出.Intel的绝对看不到,Intel的是绝密级的文档.AMD的倒是可以看到,不同的cpu有不同的BWG.这里给出一个amd比较新的cpu的BWG:  
 http://www.amd.com/us-en/assets/content\_type/white\_papers\_and\_tech\_docs/31116.pdf