第一章 PS/2 协议

引言

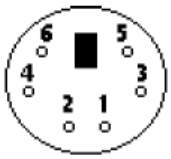
PS/2设备接口用于许多现代的鼠标和键盘，它是由IBM开发并且最初出现在IBM技术参考手册里。但是，当我知道的时候这篇文件就已经很多年没有印刷了，因此关于这个内容现在没有官方的出版物。我无法访问IBM的技术参考手册，所以本文中的所有信息都来自于我自己的经验及一些参考的帮助。

这个文件描述了用于PS/2鼠标、PS/2键盘的接口。我将论及物理和电气接口也包括协议。如果你需要更高级的信息，诸如命令、数据包的格式或者其他关于键盘鼠标的特别细节，详见本文第二章和第三章。

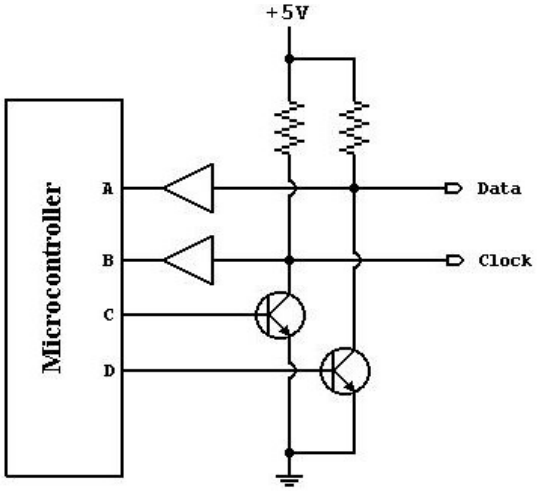
连接器

常用的PS/2端口是6脚的mini-DIN，PC键盘常用的也是6脚的mini-DIN。具有6脚mini-DIN的键盘通常被叫做“PS/2”键盘。现在流行的键盘（和鼠标）大多是PS/2或USB的。这篇文章不适用于USB设备，它们使用了一种完全不同的接口。

PS/2连接器的引脚定义如下所示：

Male 公的	Female 母的	6 脚 Mini-DIN(PS/2)
		1—数据
		2—未实现，保留
		3—电源地
		4—电源+5V
(Plug) 插头	(Socket) 插座	5—时钟
		6—未实现，保留

在PS/2连接器上有四个管脚：电源地、+5V、数据和时钟。主机提供+5V，并且键盘/鼠标的地线连接到主机的电源地上。数据和时钟都是集电极开路(OC)的，因此任何你连接到PS/2鼠标、键盘或主机的设备在时钟和数据线上都要有一个大的上拉电阻（一般取10KΩ）。置“0”就把线拉低，置“1”就让线上浮成高电平。参考右图中数据和时钟线的一般接口。（注意：如果你打算使用象51这样的微控制器，由于它们的I/O管脚是双向的，你可以跳过晶体管和缓冲门，并且通用同一个管脚进行输入和输出。）



一般性描述

PS/2鼠标和键盘履行一种双向同步串行协议。换句话说，每次数据线上发送一位数据并且每在时钟线上发一个脉冲就被读入。键盘/鼠标可以发送数据到主机，而主机也可以发送数据到设备，但主机总是在总线上有优先权，它可以在任何时候抑制来自于键盘/鼠标的通讯，只要把时钟拉低即可。

从键盘/鼠标发送到主机的数据在时钟信号的下降沿（当时钟从高变到低的时候）被读取；从主机发送到键盘/鼠标的数据在上升沿（当时钟从低变到高的时候）被读取。不管通讯的方向怎样，键盘/鼠标总是产生时钟信号。如果主机要发送数据，它必须首先告诉设备开始产生时钟信号（这个过程在后面中详细讲解）。最大的时钟频率是33kHz，而且大多数设备工作在10－20kHz。如果你要制作一个PS/2设备，我推荐你把频率控制在15kHz左右，这就意味着时钟应该是高40us低40us。

所有数据安排在字节中，每个字节为一帧，包含了11－12个位。这些位的含义如下：

- 1个起始位，总是为0
- 8个数据位，低位在前
- 1个校验位，奇校验
- 1个停止位，总是为1
- 1个应答位，仅在主机对设备的通讯中

当主机发送数据给键盘/鼠标时，设备回送一个握手信号来应答数据包已经收到。这个位不会出现在设备发送数据到主机的过程中。

设备到主机的通讯过程

数据和时钟线都是集电极开路结构（正常保持高电平）。当键盘或鼠标等待发送数据时，它首先检查时钟以确认它是否是高电平。如果不是，那么是主机抑制了通讯，设备必须缓冲任何要发送的数据直到重新获得总线的控制权（键盘有16 字节的缓冲区，而鼠标的缓冲区仅存储最后一个要发送的数据包）。如果时钟线是高电平，设备就可以开始传送数据。

如在上一节提及的，键盘和鼠标使用一种每帧包含11位的串行协议。这些位含义是：

- 1个起始位，总是为0
- 8个数据位，低位在前
- 1个校验位，奇校验
- 1个停止位，总是为1

每位在时钟的下降沿被主机读入，如图2和3所示：

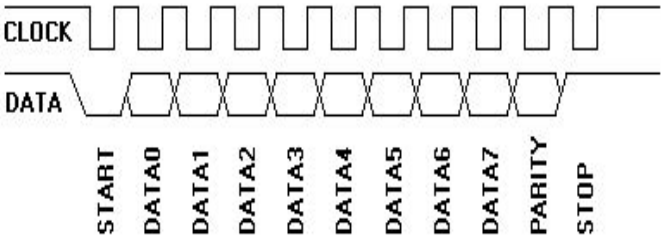


图2 设备到主机的通讯；
当时钟为高，数据线改变状态；
在时钟信号的下降沿数据被锁存。

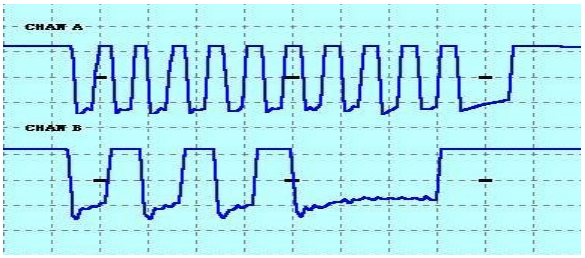


图3 ”Q”键的扫描码从键盘发送到计算机；通道A是时钟信号；
通道B是数据信号。

时钟频率为10－16.7kHz。从时钟脉冲的上升沿到一个数据转变的时间至少要有5us，数据变化到时钟脉冲的下降沿的时间至少要有5us并且不大于25us。这个定时必须严格遵循。主机可以在第11个时钟脉冲（停止位）之前把时钟线拉低，导致设备放弃发送当前字节（这是非常罕见的）。在停止位发送后，设备在发送下个包前至少应该等待50us。这将给主机一定时间处理接收到的字节（主机在收到每个包时，通常自动做这个），在处理字节这段时间内，主机应抑制其发送。在主机释放抑制后，设备至少应该在发送任何数据前等50us。

我推荐仿真键盘/鼠标采用下面的过程发送一字节的数据到主机：

1) 等待Clock线为高电平，即等待主机释放Clock线；
2) 延时50us；
3) 判断Clock线是否为高电平？
   No－－跳到第1步；
4) Data线是否为高电平？
   No－－放弃(跳到从主机读取字节的程序中）。
5) 延迟20us，输出起始位(0)，然后延迟20us，再拉低Clock线保持40us后释放Clock线，形成一个脉冲；
6) 延时20us，测试Clock线是否为高电平？
   No－－跳到第1步；
7) 输出第1个数据位，然后延时20us，再拉低Clock线保持40us后释放Clock线，形成一个脉冲；

8) 重复6-7步发送剩下的7个数据位和校验位；
9) 延时20us，测试Clock线是否为高电平？
No－－跳到第1步；
10) 输出停止位(1)，然后延时30us，再拉低Clock线保持50us后释放Clock线，形成最后一个脉冲。

主机到设备的通讯过程

在这里，被发送的数据包有点不同于设备到主机的通讯过程。

首先，PS/2设备总是产生时钟信号。如果主机要发送数据，它必须首先把时钟和数据线设置为“请求发送”状态，即：

- 通过下拉时钟线至少100us来抑制通讯；
- 通过下拉数据线来应用“请求发送”，然后释放时钟。

PS/2设备应该在不超过10ms的间隔内就要检查这个状态。当设备检测到这个状态，它将开始产生标记下的八个数据位和一个停止位的时钟脉冲。主机仅当时钟线为低的时候改变数据线，而数据在时钟脉冲的上升沿被锁存。这与发生在设备到主机的通讯过程中正好相反。

在停止位发送后，设备要应答接收到的字节，就把数据线拉低并产生最后一个时钟脉冲。如果主机在第11 个时钟脉冲后不释放数据线，设备将继续产生时钟脉冲，直到数据线被释放（然后设备将产生一个错误）。

主机可以在第11个时钟脉冲（应答位）前中止一次传送，只要下拉时钟线至少100us。

要使得这个过程易于理解，主机必须按下面的步骤发送数据到PS/2设备：

1) 把Clock线拉低至少100us；
2) 把Data线拉低；
3) 释放Clock线；
4) 等待PS/2设备把Clock线拉低；
5) 设置/复位Data线发送第一个数据位；
6) 等待PS/2设备把时钟拉高；
7) 等待PS/2设备把时钟拉低；
8) 重复5-7步发送剩下的7个数据位和校验位；
9) 释放Data线，即发送停止位(1)；
10) 等待PS/2设备把Clock线拉高； //此步可省略,因为下一步PS/2设备还是会把Data线拉低的
11) 等待PS/2设备把Data线拉低；
12) 等待PS/2设备把Clock线拉低；
13) 等待PS/2设备释放Clock线和Data线。

图3用图形，图4以单独的时序，表示了由主机产生的信号及由PS/2设备产生的信号。注意应答位时序的改变—数据改变发生在Clock线为高的时候（不同于其它11位是Clock线为低的时候）。

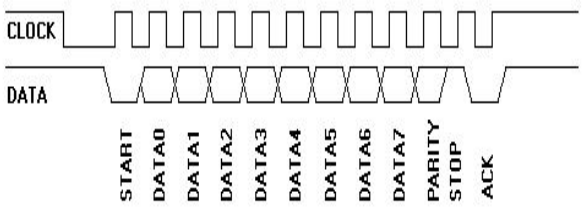


图3 主机到设备的通讯

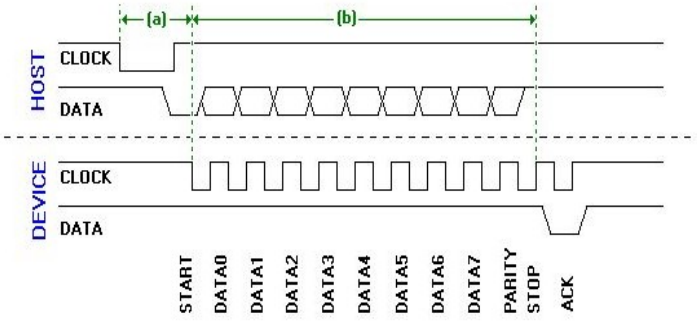


图4 主机到设备通讯的详细过程

图4描述了两个重要的定时条件：(a)和(b)。(a)-从主机最初把Clock线拉低，到PS/2设备开始产生时钟脉冲（即Clock线被PS/2设备拉低），这段时间间隔必须不大于15ms。(b)- 发送数据包（8位数据位和校验位

的总时间必须不大于2ms。如果这两个条件不满足，主机将产生一个错误。在包收到后，主机为了处理数据应立刻把时钟线拉低来抑制通讯。如果主机发送的命令要求有一个应答，这个应答必须在主机释放Clock线后20ms之内被收到；如果没有收到，则主机产生一个错误。在设备到主机通讯的情况中，时钟改变后的5us内不应该发生数据改变的情况。

如果你要仿真一个鼠标或键盘，我推荐你按如下的过程从主机读入数据：

- 在你的主程序中，至少每10ms检测一次Data线是否为低；
- 如果Data线已被主机拉低，则从主机读取一个字节的数据。

即：

1) 等待Clock线为高电平，即等待主机释放Clock线；
2) Data线仍然为低吗？
   No－有错误发生；放弃。
3) 读入8个数据位；　//在读入这些位后，
4) 读入校验位；　　//测试时钟线数否被主机拉低，
5) 读入停止位；　　//这就意味着放弃这次传送。
6) Data线仍旧为低吗？
   Yes－继续产生Clock信号，直到Data线为高电平，然后产生一个错误；
7) 输出应答位；
8) 检查校验位；　　//如果校验位不正确，则产生一个错误；
9) 延迟45us（给主机时间抑制下次的传送）。

按如下次序读取每位（8个数据位、校验位和停止位）：

1) 延迟20us；
2) 把Clock线拉低；
3) 延迟40us；
4) 释放Clock线；
5) 延迟20us；
7) 读Data线。

按如下次序发送应答位：

1) 延迟15us；
2) 把Data线拉低；
3) 延迟5us；
4) 把Clock线拉低；
5) 延迟40us；
6) 释放Clock线；
7) 延迟5us；
8) 释放Data线。

（后注：PS/2设备发送8个数据位时，是按照从最低位到最高位的顺序依次发出的。）

第二章 AT-PS/2 键盘接口

引言

本章试着囊括AT和PS/2键盘各方面的问题，它包含了如低级别信号和协议、扫描码、命令集、初始化、兼容性问题和其他各种信息。我还包含了关于PC键盘控制器的信息，这是由于它们非常相关。

应该说明的是，在这篇文章里提到的信息来自我自己的经验和其他资源，因此可能不正确。我没有参考任何官方的文件，因为没有我能用到的。因而我提出如下的弃权：

ALL INFORMATION WITHIN THIS ARTICLE IS PROVIDED "AS IS" AND WITHOUT ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, WITHOUT LIMITATION, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR A PARTICULAR PURPOSE. I DO NOT GUARANTEE ANY INFORMATION IN THIS ARTICLE IS ACCURATE, AND IT SHOULD BE USED FOR ABSTRACT EDUCATIONAL PURPOSES ONLY.

（译者注：大伙们自己看原文吧，这个申明对要使用该文的人来说很重要！）

相关的历史

现今仍在使用中的绝大多数流行的键盘包括：

USB键盘－－最后出现的键盘，被所有新式的计算机支持（Macintosh 和IBM/及其兼容机）。它们有自己相关的复杂接口，并且不包含这篇文章中。

IBM机器兼容键盘－－也叫做“AT键盘”或“PS/2键盘”，所有现代的PC都支持这个设备。它们是最容易使用的接口，也是本文的主题。

ADB键盘－－连接到老式Macintosh系统的Apple桌面总线，不包含在这篇文章中。

IBM引入了一种新型的键盘作为它每种主要桌面计算机型号的配备。最早的IBM PC和后来的IBM XT使用的我们称之为“XT键盘”。它们很古老并和现代的键盘一点都不相同；关于XT 键盘没有在本文中论及。后来出现了IBM AT系统，再后来出现IBM PS/2。他们引进的键盘我们至今还在使用，也是本文的主题。AT键盘和PS/2键盘是十分相似的设备，但是PS/2使用了更小的连接器并且支持少量附加的特征。虽然如此，它仍保留了与AT系统向后兼容，以及一些曾经流行的附加特征（因为软件总要保持向后兼容）。下面是IBM三种主要键盘的概要：

IBM PC/XT键盘(1981)：	IBM AT键盘(1984)：－(不向后兼容XT系统)
83键	84－101键
5脚DIN连接器	5脚DIN 连接器
简单的单向串行协议	双向串行协议
采用我们现在提及的作为第一套扫描码集	采用我们现在提及的作为第二套扫描码集
没有主机到键盘的命令	八个主机到键盘的命令

IBM PS/2键盘(1987)： －(兼容AT系统，不兼容XT系统)
84－101键
6脚mini-DIN连接器
双向串行协议
提供可选的第三套扫描码集
17个主机到键盘的命令

PS/2键盘最初是AT键盘的扩展，它支持少量附加的主机到键盘的命令并以小型连接器为特征，在这两种设备之间只有这两个区别。但是计算机硬件决不会有象兼容性这样多的标准。由于这种原因，你今天买到的任何键盘都与PS/2和AT系统兼容，但它可能不完全支持原始键盘的所有特征。

今天，“AT键盘”和“PS/2键盘”仅涉及它们的连接器大小。任何给定的键盘支持或不支持哪些设置及命令是每个人的猜测。例如，我现在使用的键盘有一个PS/2风格的连接器，但它仅完全支持七个命令，部分支持两个命令，对其他的命令只是“应答”。作为对照，我做测试的键盘有一个AT风格的连接器但是支持原始PS/2设备的每个特征/命令（还加上少量额外的命令）。这就重要的说明了现代的键盘是兼容性的，而不是标准。如果你的工程依赖于某些不一般的特征，它可能在一些系统上工作，而在另一些上却不能。

现代AT-PS/2兼容键盘

任意数目的按键（通常是101或104）
5脚或6脚连接器，通常包括了适配器
双向串行协议
只有第二套扫描码集是保证的
应答所有的命令；但可能不是所有的都起作用

脚注1)：XT键盘使用了一套与AT和PS/2系统中用的完全不同的协议，因此它不和较新的PC兼容。但是在某些键盘控制器中有一转换过程可以既支持XT又支持AT-PS/2键盘（通过开关、跳线或自适应），同样这些键盘在两类系统都可以工作（再次重申，是通过开关或者自适应）。如果你有这样的PC或键盘，不要被它愚弄了，因为XT键盘并不兼容于现代的计算机。

一般性描述

键盘上包含了一个大型的按键矩阵，它们是由安装在电路板上的处理器（叫做“键盘编码器”）来监视的。具体的处理器在键盘与键盘之间是多样化的，但它们基本上都做着同样的事情：监视哪些按键被按下或释放了，并在适当的时候传送数据到主机。如果有必要，处理器处理所有的去抖动并在它的16字节缓冲区里缓冲数据。你的主板包含了一个“键盘控制器”负责解码所有来自键盘的数据，并告诉你的软件什么事件发生了。在主机和键盘之间的通讯使用IBM的协议。

脚注1)：最初，IBM使用Intel8048微处理器作为它的键盘编码器。如下是现代键盘编码器的简短清单：
Holtek: HT82K28A, HT82K628A, HT82K68A, HT82K68E
EMC: EM83050, EM83050H, EM83052H, EM83053H,
Intel: 8048, 8049
Motorola: 6868, 68HC11, 6805
Zilog: Z8602, Z8614, Z8615, Z86C15, Z86E23
脚注2)：最初，IBM使用Intel的8042微控制器作为它的键盘控制器。现在已经被兼容设备取代，并整合到主板的新品组中。键盘控制器是本文稍后论及的内容。

电气接口/协议

AT和PS/2键盘使用了与PS/2鼠标一样的协议，详见第一章的内容。

扫描码

键盘的处理器花费很多的时间来扫描或监视按键矩阵。如果它发现有键被按下、释放或按住，键盘将发送“扫描码”的信息包到计算机。扫描码有两种不同的类型：“通码”和“断码”。当一个键被按下或按住就发送通码；当一个键被释放就发送断码。每个按键被分配了唯一的通码和断码，这样主机通过查找唯一的扫描码就可以测定是哪个按键。每个键一整套的通断码组成了“扫描码集”。现在有三套标准的扫描码集，分别是第一套、第二套和第三套，所有现代的键盘默认使用第二套扫描码。

那么你能计算出每个按键的扫描码吗？不幸的是，没有一个简单的公式可以计算扫描码。如果你要知道某特定按键的通码和断码，你将不得不查表获得。我已经把所有三套扫描码集中所有的通码和断码做成了表格。

第一套扫描码集－原始的XT扫描码集，某些现代的键盘还支持；
第二套扫描码集－所有现代键盘默认的扫描码集；
第三套扫描码集－可选的PS/2扫描码集（很少使用）。
（译者注：这是我见过的收录最全的扫描码集，详见本文附录。感谢Adam Chapweske所做的工作。）

脚注1)：最初，AT键盘只支持第二套，PS/2键盘默认使用第二套且支持所有这三套。许多现代键盘行为

象PS/2设备，但我遇到少数不支持第一套、第三套或这两套都不支持的。同样，如果你曾经做过低级PC编程，你可能注意到键盘控制器缺省支持第一套扫描码。这是因为键盘控制器转换所有进来的扫描码到第一套（这是为了和XT系统的软件保持兼容）。但是，它仍旧下发第二套扫描码到键盘的串行线。

通码断码和机打重复率

只要一个键被按下，这个键的通码就被发送到计算机。通码只表示键盘上的一个按键，它不表示印刷在按键上的那个字符。这就意味着在通码和ASCII码之间没有已定义好的关联，直到主机把扫描码翻译成一个字符或命令。

虽然多数第二套通码都只有一个字节宽，但也有少数“扩展按键”的通码是两字节或四字节宽。这类的通码第一个字节总是为E0H。

正如键按下通码就被发往计算机一样，只要键一释放，断码就会被发送。每个键都有它自己唯一的通码，它们也都有唯一的断码。幸运的是，你不用总是通过查表来找出按键的断码－－在通码和断码之间存在着必然的联系。多数第二套断码有两字节长，它们的第一个字节是F0H， 第二个字节是这个键的通码。扩展按键的断码通常有三个字节，它们前两个字节是E0H，F0H，最后一个字节是这个按键通码的最后一个字节。作为一个例子，我在下面列出了几个按键的第二套通码和断码：

Key	(Set 2)Make Code	(Set 2) Break Code
"A"	1C	F0,1C
"5"	2E	F0,2E
"F10"	09	F0,09
Right Arrow	E0, 74	E0, F0, 74
Right "Ctrl"	E0, 14	E0, F0, 14

例如：通码和断码是以什么样的序列发送到你的计算机，使得字符“G”出现在你的字处理软件里呢？因为这是一个大写字母，需要发生这样的事件次序：按下“Shift键”，按下“G键”，释放“G键”，释放“Shift键”。与这些时间相关的扫描码如下：“Shift键”的通码（12H），“G键”的通码（34H），“G键”的断码（F0H 34H），“Shift键”的断码（F0H 12H）。因此发送到你的计算机的数据应该是：12H 34H F0H 34H F0H 12H。

如果你按了一个键，这个键的通码被发送到计算机。当你按下并按住这个键，则这个键就变成了机打，这就意味着键盘将一直发送这个键的通码直到它被释放或者其他键被按下。要想证实这点，只要打开一个文本编辑器并按下“A键”。当你首先按下这个键，字符a立刻出现在你的屏幕上。在一个短暂的延迟后，接着出现一整串的“a”，直到你释放“A键”。这里有两个重要的参数：机打延时，是第一个和第二个a之间的延迟机打速率，是在机打延时后每秒有多少字符出现你的屏幕上。机打延时的范围可以从0.25秒到1.00秒；机打速率的范围可以从2.0cps(字符每秒)到30.0cps。你可以用“Set Typematic Rate/Delay”(0xF3)命令来改变机打速率和延时。

机打的数据不被键盘所缓冲。在多个键被按下的情况下，只有最后一个按下的键变成机打。当这个键被释放时，机打重复就停止了，甚至于其他的键依然还按着。

脚注1)：实际上，在第一第二套扫描码里没有“Pause/Break键”的断码。当这个键按下时发送它的通码当它释放时什么都没有被发送。

复位

在上电或软件复位（见“Reset”命令）后，键盘执行诊断自检叫做BAT（基本保证测试）并载入如下的缺省值：

机打延迟为500ms

机打速率为10.9cps

*第二套扫描码集

*置所有按键为机打/通码/断码

注：“*”所指的项在某些键盘上是可变的，而在其他键盘上是硬件编码的（不可变）。

当进入BAT，键盘点亮它的三个LED指示器，并在完成BAT后关闭它们。此时，BAT完成代码发送0xAA（BAT成功）或0xFC（有错误）到主机。

多数键盘忽略它们的时钟和数据线，直到BAT完成代码发送后。所以，“抑制”条件（时钟线拉低）可能不能防止键盘发送它们的BAT完成代码。

命令集

每个发送到键盘的字节都从键盘获得一个0xFA（“应答”）的回应。唯一例外的是键盘对“Resend”和“Echo”命令的回应。在发送下一个字节给键盘之前，主机要等待“应答”。键盘应答任何命令后清除自己的输出缓冲区，下面列出了所有可能被发给键盘的命令。

- 0xFF(Reset)－引起键盘进入“Reset”模式。（见“复位”部分）
- 0xFE(Resend)－只能用在主机接收键盘数据出现了错误后发送。键盘的响应就是重发送最后的扫描码或者命令回应给主机。但是0xFE绝不会作为“Resend”命令的回应而被发送。
- *0xFD(Set Key Type Make)－允许主机指定一个按键只发送通码。这个按键不发送断码或进行机打重复。指定的按键采用它的第三套扫描码。
- *0xFC(Set Key Type Make/Break)－类似于“Set Key Type Make”，只有通码和断码是使能的（机打被禁止了）。
- *0xFB(Set Key Type Typematic)－类似于前两条命令，通码和机打是使能的，而断码被禁止。
- *0xFA(Set All Keys Typematic/Make/Break)－缺省设置。所有键的通码、断码和机打重复都使能（除了“Print Screen”键，它在第一套和第二套中没有断码。）
- *0xF9(Set All Keys Make)－所有键都只发送通码，断码和机打重复被禁止。
- *0xF8(Set All Keys Make/Break)－类似于前两条命令，除了只是机打重复被禁止外。
- *0xF7(Set All Keys Typematic)－类似于前三条命令，仅断码被禁止，通码和机打重复是使能的。
- 0xF6(Set Default)－载入缺省的机打速率/延时（10.9cps/500ms），按键类型(所有按键都使能机打/通码/断码)，以及第二套扫描码集。
- 0xF5(Disable)－键盘停止扫描，载入缺省值（见“Set Default”命令），等待进一步指令。
- 0xF4(Enable)－在用上一条命令禁止键盘后，重新使能键盘。
- 0xF3(Set Typematic Rate/Delay)－主机在这条命令后会发送一个字节的参数来定义机打速率和延时，具体含义如下：

Repeat Rate

Bits 0-4	Rate(cps)	Bits 0-4	Rate(cps)	Bits 0-4	Rate(cps)	Bits 0-4	Rate(cps)
00h	2.0	08h	4.0	10h	8.0	18h	16.0
01h	2.1	09h	4.3	11h	8.6	19h	17.1
02h	2.3	0Ah	4.6	12h	9.2	1Ah	18.5
03h	2.5	0Bh	5.0	13h	10.0	1Bh	20.0
04h	2.7	0Ch	5.5	14h	10.9	1Ch	21.8
05h	3.0	0Dh	6.0	15h	12.0	1Dh	24.0
06h	3.3	0Eh	6.7	16h	13.3	1Eh	26.7
07h	3.7	0Fh	7.5	17h	15.0	1Fh	30.0

Delay

Bits 5-6	Delay (seconds)
00b	0.25
01b	0.50
10b	0.75
11b	1.00

- *0xF2(Read ID)－键盘回应两个字节的设备ID：0xAB 0x83。
- *0xF0(Set Scan Code Set)－主机在这个命令后发送一个字节的参数，指定键盘使用哪套扫描码集。参数字节可以是0x01、0x02或0x03，分别选择扫描码集第一套、第二套或第三套。如果要获得当前正在使用的扫描码集，只要发送带0x00参数的本命令即可。
- 0xEE(Echo)－键盘用“Echo”（0xEE）的回应。
- 0xED(Set/Reset LEDs)－主机在本命令后跟随一个参数字节，用于指示键盘上Num Lock，Caps Lock，和Scroll Lock LED的状态。这个参数字节的定义如下：

MSB LSB

Always 0	Always 0	Always 0	Always 0	Always 0	Caps Lock	Num Lock	Scroll Lock

○ "Scroll Lock" - Scroll Lock LED off(0)/on(1)
○ "Num Lock" - Num Lock LED off(0)/on(1)
○ "Caps Lock" - Caps Lock LED off(0)/on(1)

注：*最初只可用于PS/2键盘。

i8042 键盘控制器

写到文章的这里，从硬件的观点来说所有的信息都已经被提出。但是如果你打算给host PC写一低级的与键盘相关的程序，你却不能直接和键盘通讯。而键盘控制器提供了键盘和周边总线之间的接口。控制器不但封装了所有信号级和协议的细节，又提供了相关转换解释和扫描码及命令的句柄。

Intel 8042或兼容微控制器被用作PC键盘的控制器。在现代的计算机中这个微控制器被隐藏到了主板的芯片组中，主板的芯片组在一个单一封装中整合了很多的控制器。虽然如此，但设备仍旧是存在的，键盘控制器一般仍按“8042”来论及。

依赖于主板的不同，键盘控制器可以工作于两个模式之一：“AT”兼容模式或“PS/2”兼容模式。如果主板支持PS/2鼠标就工作在后一种模式下。在这种情况下，8042的作用是键盘控制器和鼠标控制器。键盘控制器根据键盘端口的连线情况自动检测它应该工作在何种模式下。

8042包含了如下的寄存器：
- 一个字节的输入缓冲区－包含从键盘读入的字节；只读。
- 一个字节的输出缓冲区－包含要写到键盘的字节；只写。
- 一个字节的状态寄存器－8个状态标志；只读。
- 一个字节的控制寄存器－7个控制标志；读写。

前三个寄存器（输入、输出、状态）可以通过0x60和0x64端口直接存取。最后那个寄存器（控制）要使用“Read Command Byte”命令读，使用“Write Command Byte”命令写。下表示出周边端口示如何于8042 接口的：

Port	Read / Write	Function	功能
0x60	Read	Read Input Buffer	读输入缓冲区
0x60	Write	Write Output Buffer	写输出缓冲区
0x64	Read	Read Status Register	读状态寄存器
0x64	Write	Send Command	发送命令

写0x64 端口不会写入到任何特定的寄存器中，但是解释为发送命令给8042。如果命令接收一个参数，则参数被发往0x60端口。同样，命令的任何返回结构可以从0x60端口读出。

在描述8042时，我偶尔提及它的物理I/O管脚，这些管脚定义如下：

AT-compatible mode

Port 1 (Input Port):			Port 2 (Output Port):			Port 3 (Test Port):		
Pin	Name	Function	Pin	Name	Function	Pin	Name	Function
0	P10	Undefined	0	P20	System Reset 1: Normal 0: Reset computer	0	T0	Keyboard Clock (Input)
1	P11	Undefined	1	P21	Gate A20	1	T1	Keyboard Data (Input)
2	P12	Undefined	2	P22	Undefined	2	--	Undefined
3	P13	Undefined	3	P23	Undefined	3	--	Undefined
4	P14	External RAM 1: Enable external RAM 0: Disable external RAM	4	P24	Input Buffer Full	4	--	Undefined
5	P15	Manufacturing Setting 1: Setting enabled 0: Setting disabled	5	P25	Output Buffer Empty	5	--	Undefined
6	P16	Display Type Switch 1: Color display 0: Monochrome	6	P26	Keyboard Clock 1: Pull Clock low 0: High-Z	6	--	Undefined
7	P17	Keyboard Inhibit Switch 1: Keyboard enabled 0: Keyboard inhibited	7	P27	Keyboard Data: 1: Pull Data low 0: High-Z	7	--	Undefined

PS/2-compatible mode

Port 1 (Input Port):			Port 2 (Output Port):			Port 3 (Test Port):		
Pin	Name	Function	Pin	Name	Function	Pin	Name	Function
0	P10	Keyboard Data (Input)	0	P20	System Reset 1: Normal 0: Reset computer	0	T0	Keyboard Clock (Input)
1	P11	Mouse Data (Input)	1	P21	Gate A20	1	T1	Mouse Clock (Input)
2	P12	Undefined	2	P22	Mouse Data: 1: Pull Data low 0: High-Z	2	--	Undefined
3	P13	Undefined	3	P23	Mouse Clock: 1: Pull Clock low	3	--	Undefined

					0: High-Z			
4	P14	External RAM 1: Enable external RAM 0: Disable external RAM	4	P24	Keyboard IBF interrupt: 1: Assert IRQ 1 0: De-assert IRQ 1	4	--	Undefined
5	P15	Manufacturing Setting 1: Setting enabled 0: Setting disabled	5	P25	Mouse IBF interrupt: 1: Assert IRQ 12 0: De-assert IRQ 12	5	--	Undefined
6	P16	Display Type Switch 1: Color display 0: Monochrome	6	P26	Keyboard Clock 1: Pull Clock low 0: High-Z	6	--	Undefined
7	P17	Keyboard Inhibit Switch 1: Keyboard enabled 0: Keyboard inhibited	7	P27	Keyboard Data: 1: Pull Data low 0: High-Z	7	--	Undefined

（注意：读键盘控制器数据表可能会迷惑你，其中提到的“输入缓冲区”是指“输出缓冲区”，反之亦然。这是取决于你看问题的观点，是从控制器的固件，还是从控制器的接口。在本文中，我所提到的“输入缓冲区”指放置从键盘获得的输入的地方，“输出缓冲器”指放置把输出发送到键盘的数据的地方。）

状态寄存器

8042的状态标志是从0x64端口读出的。它们包含了错误信息、状态信息和输入输出缓冲区里有无数据的指示。这些标志的定义如下：

	MSB							LSB
AT-compatible mode:	PERR	RxTO	TxTO	INH	A2	SYS	IBF	OBF
PS/2-compatible mode:	PERR	TO	MOBF	INH	A2	SYS	IBF	OBF

- OBF（输出缓冲区满）－指示是否成功写入输出缓冲区。
  0：输出缓冲区空－写入到0x60端口成功
  1：输出缓冲区满－不能写到0x60端口
- IBF（输入缓冲区满）－指示是否可以从输入缓冲区中读入。
  0：输入缓冲区空－不能从0x60端口读入数据
  1：输入缓冲区满－有新的数据输入了，可以从0x60端口读入
- SYS（系统标志）－Post读取这个标记测定是否上电复位还是软件复位。
  0：上电－系统处于上电自检中
  1：BAT代码完成－系统已经完成了初始化
- A2（地址线A2）－键盘控制器内部使用。
  0：A2=0 －最后写入的是端口0x60
  1：A2=1 －最后写入的是端口0x64
- INH（禁止标志）指示键盘通讯是否被禁止。
  0：键盘时钟=0 －键盘被禁止
  1：键盘时钟=1 －键盘没有被禁止
- TxTO（发送超时）－指示键盘不可接受输入(比如键盘没有插入)。
  0：无错误－键盘接收了最后一个给它的字节
  1：超时错－键盘在15ms的“请求发送”时间内没有产生时钟信号
- RxTO（接收超时）－指示键盘不能回应命令(键盘可能是坏的)。

0：无错误－键盘回应了最后一个字节
1：超时错－键盘在20ms的命令接收期内没有产生时钟信号

- PERR（校验错误）－指示和键盘的通讯有错误(可能有干扰或者连接松掉了)。
  0：无错误－接收到了奇校验并且收到了适当的命令回应
  1：校验错－接收到偶校验或者0xFE作为命令回应被收到了
- MOBF（鼠标输出缓冲区满）类似于OBF但不包括PS/2鼠标。
  0：输出缓冲区空－写到扩展设备的输出缓冲区成功
  1：输出缓冲区满－不可以写到辅助设备的输出缓冲区端口
- TO（一般性超时）-指示在命令写入或回应中有超时(和TxTO+RxTO一样)。
  0：无错误－键盘收到并回应了最后一条命令
  1：超时错－参考TxTO和RxTO，获得更多信息

『例如：在我的PC上，“8042”状态寄存器的正常值是14h＝00010100b。这个值说明键盘通讯没有被禁止，8042已经完成了自检(“BAT”)。状态寄存器可以从64h端口读取(“IN AL,64h”)』

读键盘的输入

当8042从键盘收到有效的扫描码，就把它转换成第一套相等的扫描码。转换后的扫描码放置在输入缓冲区，IBF（输入缓冲区满）标志被设置，产生IRQ1。而且，此时如果有其他来自键盘的字节，8042将抑制更多的接收（通过“把时钟线拉低”），这样就不会收到更多的扫描码一直到输入缓冲区被清空。

如果中断是使能的，IRQ1将激活键盘驱动程序，这是指向0x09号中断向量的。驱动程序从0x60端口读取扫描码，这个动作会释放IRQ1并复位IBF标志。接着扫描码被驱动程序处理，如回应特殊键的组合更新系统RAM保留给键盘输入的区域里的数据等。

如果你不打算在中断0x09中嵌入个补丁，你可以轮询键盘控制器来输入数据。这是通过禁止8042IBF中断再轮询IBF标志来实现的。数据如果再输入缓冲区是可用的，标志就会被置1，如果数据从输入缓冲区里被读出了，标志就会被清0。读输入缓冲区就是读0x60端口中的数据，IBF标志在0x64端口的第1位。

往键盘写数据

当你写数据到8042的输出缓冲区（通过0x60端口），控制器设置OBF（输出缓冲区满）标志并处理数据。8042将发送这个数据到键盘并等待一个回应。如果键盘没有接收或者在指定时间内没有回应，相应的超时标志就会被设置（见状态寄存器的定义给获得更多信息）。如果键盘继续发送错误的字节，状态寄存器里的“校验错”标志就会被置位。如果没有错误发生，回应字节就会放到输入缓冲区中，IBF（输入缓冲区满）标志被置位，IRQ1被激活，发信号给键盘驱动程序。

键盘控制器命令

发送命令给键盘控制器就是写0x64端口。在命令发送后，命令参数写到0x60端口。结果也返回到0x60端口。在写命令或参数到8042之前，总是要测试OBF（输出缓冲区满）标志的。

- 0x20(读命令字节)－返回命令字节。(参考下面的“写命令字节”)
- 0x60(写命令字节)－存储参数作为命令字节。命令字节定义如下：

	MSB							LSB
AT-compatible mode:	--	XLAT	PC	_EN	OVR	SYS	--	INT
PS/2-compatible mode:	--	XLAT	_EN2	_EN	--	SYS	INT2	INT

- INT(输入缓冲区满中断)－如果设置了，当输入缓冲区有有效数据时产生IRQ1。
  0：IBF中断禁止－你必须轮询状态寄存器的<IBF>来读取输入
  1：IBF中断使能－在中断0x19中的键盘驱动软件俘获输入
- SYS(系统标志)－常用于手动设置/清除状态寄存器中的SYS标志。
  0：上电－告诉POST执行上电自检及初始化

1：BAT代码收到－告诉POST执行热启动测试及初始化

- OVR(忽略禁止)－在某些老主板上忽略键盘的“禁用”开关。
  0：禁用开关使能－如果P17脚为高则禁用键盘
  1：禁用开关禁止－键盘不被禁用甚至于P17为高
- _EN(禁止键盘)－禁止/使能键盘接口。
  0：使能－键盘接口使能
  1：禁止－所有键盘通讯被禁止
- PC(“PC模式”)－???在某种情况下使能键盘接口???
  0：禁止－???
  1：使能－???
  (译者注：这个命令显然作者不知道是如何工作的，所以我也不知道。)
- XLAT(翻译扫描码)－使能/禁止翻译成第一套扫描码。
  0：翻译禁止－从键盘读入的数据直接放入输入缓冲区
  1：翻译使能－扫描码在放入输入缓冲区之前先翻译成第一套的对应值
- INT2(鼠标输入缓冲区满中断)－当设置后。如果鼠标数据有效则产生IRQ12。
  0：辅助IBF中断禁止
  1：辅助IBF中断使能
- _EN2(禁止鼠标)－禁止/使能鼠标接口
  0：使能－辅助的PS/2设备接口被使能
  1：禁止－辅助的PS/2设备接口被禁止

- ?0x90-0x9F(写到输出端口)－写命令的低4位到输出端口的低4位(见输出端口的定义)。
- ?0xA1(获得版本号)－返回固件的版本号。
- ?0xA4(获得密码)－如果密码存在则返回0xFA，否则返回0xF1。
- ?0xA5(设置密码)－设置新的密码，发送一个以空字符结束的扫描码字串作为命令的参数。
- ?0xA6(比较密码)－把键盘输入和当前的密码相比较。
- 0xA7(禁止鼠标接口)－仅用于PS/2模式。类似于“禁止键盘接口”(0xAD)命令。
- 0xA8(使能鼠标接口)－仅用于PS/2模式。类似于“使能键盘接口” (0xAE)命令。
- 0xA9(鼠标接口测试)－若通过则返回0x00，若时钟线保持低不变则返回0x01，若时钟线保持高不变则返回0x02，若数据线保持低不变则返回0x03，若数据线保持高不变则返回0x04。
- 0xAA(控制器自检)－若成功则返回0x55。
- 0xAB(键盘接口测试)－若通过则返回0x00，若时钟线保持低不变则为0x01，若时钟线保持高不变则为0x02，若数据线保持低不变则为0x03，若数据线保持高不变则为0x04。
- 0xAD(禁止键盘接口)－设置命令字节的第4位，并禁止所有与键盘间的通讯。
- 0xAE(使能键盘接口)－清除命令字节的第4位，并重新使能与键盘的通讯。
- 0xAF(获得版本)
- 0xC0(读输入端口)－返回输入端口中的值(参考输入端口的定义)。
- 0xC1(拷贝输入端口的LSn)－仅用于PS/2模式。拷贝输入端口的低四位到状态寄存器(参考输入端口定义)。
- 0xC2(拷贝输入端口的MSn)－仅用于PS/2模式。拷贝输入端口的高四位到状态寄存器(参考输入端口定义)。
- 0xD0(读输出端口)－返回输出端口的值(参考输出端口定义)。
- 0xD1(写输出端口)－写参数到输出端口(参考输出端口定义)。
- 0xD2(写键盘缓冲区)－把参数写到输入缓冲区，就像是从键盘接收到的一样。
- 0xD3(写鼠标缓冲区)－把参数写到输入缓冲区，就像是从鼠标接收到的一样。
- 0xD4(写鼠标设备)－发送参数给辅助的PS/2设备。
- 0xE0(读测试端口)－返回测试端口的值(参考测试端口的定义)。
- 0xF0-0xFF(脉冲输出端口)－在脉冲下输出命令的低四位到输出端口的低四位(参考输出端口定义)。

现代键盘控制器

到目前为止，我只讨论了8042键盘控制器。虽然现代的键盘控制器保持了和原始设备的兼容性，但兼容

是他们唯一的需求(和目标)。

我的主板上的键盘控制器是一个重要的例子。我把一个微控制器和LCD并接到我的键盘上来观察键盘控制器发送了哪些数据。上电后键盘控制器发送了“设置LED状态”关闭了所有的LED。然后读取键盘的ID。当我试着写数据到输出缓冲区，我发现键盘控制器只转发了“设置LED状态”命令和“设置机打速率/延时”命令。它不允许任何其他命令发往键盘。但是，在适当时候它通过放置“应答”到输入缓冲区来仿真键盘的应答（和0xEE回应“Echo”命令）。此外，如果键盘发送了一个错误的字节，键盘控制器将拿走错误俘获（发送“Retry”命令“；如果字节仍旧是错的，发送错误代码给键盘并将错误代码放入输入缓冲区）。

再说一次，记住芯片组的设计者对兼容比标准更有兴趣。

初始化

如下的通讯过程发生在我的计算机和键盘之间。当计算机启动后，我相信前三个命令是初始化键盘控制器，后一条命令（使能Numlock LED）是由BIOS发送的，剩下来的命令是由我的OS（Win98SE）发送的。记住，在我计算机上这么结果是明确的，但是它只是给你一个一般性的概念告诉你启动时发生了什么。

```
Keyboard: AA Self-test passed               ;Keyboard controller init
Host:     ED Set/Reset Status Indicators
Keyboard: FA Acknowledge
Host:     00 Turn off all LEDs
Keyboard: FA Acknowledge
Host:     F2 Read ID
Keyboard: FA Acknowledge
Keyboard: AB First byte of ID
Host:     ED Set/Reset Status Indicators    ;BIOS init
Keyboard: FA Acknowledge
Host:     02 Turn on Num Lock LED
Keyboard: FA Acknowledge
Host:     F3 Set Typematic Rate/Delay       ;Windows init
Keyboard: FA Acknowledge
Host:     20 500ms/30.0reports/sec
Keyboard: FA Acknowledge
Host:     F4 Enable
Keyboard: FA Acknowledge
Host:     F3 Set Typematic Rate/delay
Keyboard: FA Acknowledge
Host:     00 250ms/30.0reports/sec
Keyboard: FA Acknowledge
```

第三章 PS/2 鼠标接口

电气接口/协议

PS/2鼠标使用和PS/2键盘一样的协议。这个标准最初出现在IBM技术参考手册里，但我知道现在已经没有关于这个标准的任何官方文件了。不过，你可以查看本文第一章获得我收集的关于这个协议的细节。

输入分辨率和缩放比例

标准的PS/2鼠标支持下面的输入：X(左右)位移，Y(上下)位移，左键，中键和右键。鼠标以一个固定的频率读取这些输入并更新不同的计数器然后标记出反映的移动和按键状态。有很多PS/2指示设备具有额外的输入并可以报告不同于本文描述的数据。一个受欢迎的扩充是我在文章后面介绍的Microsoft的Intellimouse它既支持标准输入也支持滚轮和两个附加的按键。

标准的鼠标有两个计数器保持位移的跟踪：X位移计数器和Y位移计数器。可存放9位的2进制补码，并且每个计数器都有相关的溢出标志。它们的内容连同三个鼠标按钮的状态一起以三字节移动数据包的形式发送给主机（描述见下一部分）。位移计数器表示从最后一次位移数据包被送往主机后有位移量发生。

当鼠标读取它的输入的时候，它记录按键的当前状态，然后检查位移。如果位移发生，它就增加（对正位移）或减少（对负位移）X和/或Y位移计数器的值。如果有一个计数器溢出了，就设置相应的溢出标志。

决定位移计数器增减数量的参数叫分辨率。缺省的分辨率为4计数单位/毫米，主机可以用“设置分辨率”(0xE8)命令改变这个值。

有一个参数不影响位移计数器的值，但是影响这些计数器报告的值（见本章的脚注1）。这个参数就是缩放比例。缺省情况下鼠标使用1：1比例，因此对报告的鼠标位移没有影响。但是主机可以用“设置比例2：1”(0xE7)命令选择2：1比例。如果启用了2：1比例，鼠标在发数据给主机前采用如下的算法运算计数器内容：

Movement Counter	Reported Movement
0	0
1	1
2	1
3	3
4	6
5	9
N>5	2*N

位移数据包

标准的PS/2鼠标发送位移和按键信息给主机采用如下的3字节数据包格式（见本章脚注4）。

	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Byte 1	Y overflow	X overflow	Y sign bit	X sign bit	Always 1	Middle Btn	Right Btn	Left Btn
Byte 2	X Movement							
Byte 3	Y Movement							

位移计数器是一个9 位2 的补码整数。它的最高位作为符号位出现在位移数据包的第一个字节里。这些计数器在鼠标读取输入发现有位移时被更新。这些值是自从最后一次发送位移数据包给主机后位移的累计量（即最后一次包发给主机后，位移计数器被复位）。位移计数器可表示的值的范围是-255到+255。如果超过了范围，相应的溢出位就被设置，并且在复位前，计数器不会增减。

正如我前面提及的，一旦位移数据包成功地发送给主机，位移计数器就会复位。同样鼠标在收到主机不是“Resend”(0xFE)命令外的其他命令，计数器也会复位。

操作模式

根据鼠标工作的模式来处理的数据报告，有四种标准的工作模式：

- Reset－鼠标在上电或收到“Reset”(0xFF)命令后进入Reset模式。
- Stream－这是缺省模式（在Reset执行完成后），也是多数软件使用鼠标的模式。如果主机先前把鼠标设置到了Remote模式，那它可以发送“Set Stream Mode”(0xEA)命令给鼠标，让鼠标重新进入Stream模式。
- Remote－在某些情况下Remote模式很有用，可以通过发送“Set Remote Mode”(0xF0)命令进入。
- Wrap－除了为测试鼠标和它的主机之间的连接外，这个模式不是特别地有用。Wrap模式可以通过发送命令“Set Wrap Mode”(0xEE)命令给鼠标来进入。要退出Wrap模式，主机必须发布“Reset”(0xFF)命令或“Reset Wrap Mode”(0xEC)命令。如果“Reset”(0xFF) 命令收到了，鼠标将进入Reset模式。如果收到的是“Reset Wrap Mode”(0xEC)命令，鼠标将进入Wrap模式前的那个模式。

（注意：鼠标同样可以进入“extended”操作模式，正如本文后面所述。但是，这不是标准PS/2鼠标的特征。）

Reset 模式

鼠标在上电后或应答“Reset”(0xFF)命令就进入reset模式。进入这个模式后，鼠标执行象前面提到的BAT（基本保证测试）一样的自检并设置如下的缺省值：

采样速率－100采样点/秒

分辨率－4个计数值/毫米

缩放比例－1:1

数据报告被禁止

然后发送BAT完成代码，这个代码不是0xAA（BAT成功）就是0xFC（错误）。如果主机收到了不是0xAA的回应，它可能重新给鼠标供电，这样来引起鼠标复位并重新执行BAT。

接着BAT完成代码（0xAA或0xFC）的后面鼠标发送它的设备ID 0x00。这个ID用来区别设备是键盘还是处于扩展模式中的鼠标。我读到的文件中说，主机在没收到设备ID前不会假定发送任何数据。但是我发现有些BIOS在上电复位并收到0xAA后立刻发送“Reset”(0xFF)命令。

鼠标发送自己的设备ID给主机后，它就进入了Stream模式。注意鼠标设置的一个缺省值之一是“数据报告被禁止”。这就意味着鼠标在没收到“使能数据报告”(0xF4)命令之前不会发送任何位移数据包给主机。

Stream 模式

在Stream模式中，一旦鼠标检测到位移或发现一个或多个鼠标键的状态改变了，就发送位移数据包。数据报告的最大速率被认为是采样速率。参数的范围从10采样点/秒到200采样点/秒。这个参数的缺省值是100采样点/秒，主机可以用“设置采样速率”(0xF3)命令来改变它。Stream模式是操作的缺省模式。

Remote 模式

在这个模式下，鼠标以当前的采样速率读取输入并更新它的计数器和标志，但是它只在主机请求数据的时候才报告给主机位移（和按键状态）。主机通过“读数据”(0xEB)命令来获得数据。在收到命令后，鼠标发送位移数据包并复位它的位移计数器。

Wrap 模式

这是一个“回声”模式，鼠标收到的每个字节都会被发回主机。甚至收到的是一个有效的命令，鼠标都不会应答这条命令－它只把这个字节回送给主机。但是有两个例外：“Reset”(0xff)命令和“Reset Wrap Mode”(0xEC)命令。鼠标认为这两条命令是一有效的命令并且不会回送它们到主机。

Intellimouse 的扩展

对标准的PS/2 鼠标的一个流行的扩展是微软的Intellimouse。它包括支持五个鼠标按键和三个位移轴

（左右、上下和滚轮）。这些附加特征要求使用4字节的位移数据包而不是标准3字节包。由于标准PS/2鼠标的驱动程序不能识别该数据包格式，因此在没有能识别改数据包的驱动程序被安装的情况下，Intellimouse操作起来跟标准的PS/2鼠标一样。所以如果Intellimouse在一台仅支持标准PS/2鼠标的电脑上使用时，除滚轮和增加的第四、第五个按键外，它仍能正常工作。

微软的Intellimouse工作起来象标准的PS/2鼠标（也就是，使用3字节位移数据包，和标准PS/2鼠标一样回应所有命令，报告设备ID 0x00）。要进入滚轮模式，主机应该发送如下的命令序列：

Set sample rate 200
Set sample rate 100
Set sample rate 80

主机然后应该发布“获得设备ID”(0xF2)命令并等待回应。如果安装的是标准PS/2鼠标(非Intellimouse)，它回应设备ID 0x00。在这种情况下，主机会认为这个鼠标没有滚轮并继续把它当作是标准PS/2鼠标。但是，如果安装的是微软的Intellimouse，它返回的ID是0x03。这就告诉主机挂接的定点设备有滚轮，并且主机认为鼠标使用4字节的位移数据包：

	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Byte 1	Y overflow	X overflow	Y sign bit	X sign bit	Always 1	Middle Btn	Right Btn	Left Btn
Byte 2	X Movement							
Byte 3	Y Movement							
Byte 4	Z Movement							

Z位移是2的补码表示滚轮的自上次数据报告以来的位移。有效值的范围在-8到+7。这意味着数值实际只有低四位高四位仅用作符号扩展位。

要进入滚轮+5键模式，主机要发送如下命令序列：

Set sample rate 200
Set sample rate 200
Set sample rate 80

主机接着发布“获得设备ID”(0xF2)命令并等待回应。微软的Intellimouse用0x04这样设备ID应答，并且使用如下的4字节位移数据包：

	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Byte 1	Y overflow	X overflow	Y sign bit	X sign bit	Always 1	Middle Btn	Right Btn	Left Btn
Byte 2	X Movement							
Byte 3	Y Movement							
Byte 4	Always 0	Always 0	5th Btn	4th Btn	Z3	Z2	Z1	Z0

Z0-Z3是2的补码用于表示从上次数据报告以来滚轮的位移量，有效范围从-8到+7。

第4键：1＝第4键按下了；0＝第4键没有按下。

第5键：1＝第5键按下了；0＝第5键没有按下。

你也许见过有两个滚轮的鼠标：一个是垂直的一个是水平的。这种鼠标使用上面介绍的微软Intellimouse数据包格式。如果垂直的滚轮向上滚动，Z计数器加1；如果这个滚轮向下滚动，Z计数器减1；这是滚轮的正常操作。但是如果水平的滚轮向右滚动，Z计数器增加2；向左滚动，Z计数器减少2。看上去象用一种临时的途径实现了第二个滚轮，但是由于放置了两个滚轮，所以不可能同时使用这两个滚轮。（如果你试着欺骗软件同时使用它们，你会发现软件忽略了水平的那个滚轮。）

命令集

下面列出的是仅可发送给鼠标的命令。如果鼠标工作在Stream模式，主机在发送任何其他命令之前要先禁止数据报告（命令0xF5）。

● 0xFF(Reset)－鼠标用“应答”(0xFA)回应这条命令，并进入Reset模式。
● 0xFE(Resend)－只要从鼠标收到无效数据，主机就发送这条命令。鼠标的回应是重新发送它最后发给主机的数据包（见本章脚注2、3）。如果鼠标用了另外一个非法的包来回应，主机要么发布另一条“Resend”命令，要么发布“Error”命令，要么让鼠标重建上电来复位它，或者禁止通讯（把时钟线拉低）。采取什么样的动作取决于主机。
● 0xF6(Set Defaults)－鼠标用“应答”(0xFA)来回应，然后载入如下的值：采样率＝100，分辨率＝4个值/毫米，比例＝1：1，禁止数据报告。接着鼠标清空它所有的位移计数器，并进入stream模式。
● 0xF5(Disable Data Reporting)－鼠标用“应答”(0xFA)回应命令，然后禁止数据报告，并复位它的位移计数器。这仅对Stream模式下的数据报告有效，并且它不能禁止采样。禁止的stream模式功能与remote模式相同。
● 0xF4(Enable Data Reporting)－鼠标用“应答”(0xFA)回应命令，然后使能数据报告并复位它的位移计数器。这条命令可以对在Remote模式（或Stream模式）下的鼠标发布，但只对Stream模式下的数据报告有效。
● 0xF3(Set Sample Rate)－鼠标用“应答”(0xFA)回应命令，然后从主机读入一个或更多字节。鼠标保留这个字节作为新的采样速率。在收到采样速率后，鼠标再次用“应答”(0xFA)回应并复位它的位移计数器。有效的采样速率是10、20、40、60、80、100和200采样点/秒。
● 0xF2(Get Device ID)－鼠标用“应答”(0xFA)回应，命令后面跟着它的设备ID（对标准PS/2鼠标来说是0x00）。鼠标同样会复位它的位移计数器。
● 0xF0(Set Remote Mode)－鼠标用“应答”(0xFA)回应，然后复位它的位移计数器，并进入Remote模式。
● 0xEE(Set Wrap Mode)－鼠标用“应答”(0xFA)回应，然后复位它的位移计数器，并进入wrap模式。
● 0xEC(Reset Wrap Mode)－鼠标用“应答”(0xFA)回应，然后复位它的位移计数器，并进入wrap模式之前的那个模式（stream模式或remote 模式）。
● 0xEB(Read Data)－鼠标用“应答”(0xFA)回应，然后发送位移数据包。这是在remote模式中读数据的位移方法。在数据包成功地被发送后，鼠标将复位它的位移计数器。
● 0xEA(Set Stream Mode)鼠标用“应答”(0xFA)回应，然后复位它的位移计数器，并进入stream模式。
● 0xE9(Status Request)－鼠标用“应答”(0xFA)回应，然后发送如下3个字节的状态包（然后复位它的位移计数器）：

	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Byte 1	Always 0	Mode	Enable	Scaling	Always 0	Left Btn	Middle Btn	Right Btn
Byte 2	Resolution							
Byte 3	Sample Rate							

右键、中键、左键 ＝1，表示键被按下；＝0，表示键没有按下。
Scaling = 1，缩放比例位2：1；＝0，比例为1：1。（见命令0xE7和0xE6）
Enable = 1，表示数据报告被使能；＝0，表示数据报告被禁止。（见命令0xF5和0xF4）
Mode = 1，表示remote模式被使能；＝0，表示stream模式被使能。（见命令0xF0和0xEA）
● 0xE8(Set Resolution)－鼠标用“应答”(0xFA)回应，然后从主机读取一个字节，并再次用“应答”(0xFA)回应，然后复位它的位移计数器。从主机读入的字节决定了分辨率，如下示：

Byte Read from Host	Resolution
0x00	1 count/mm
0x01	2 count/mm
0x02	4 count/mm
0x03	8 count/mm

● 0xE7(Set Scaling 2:1)－鼠标用“应答”(0xFA)回应，然后使能2：1比例。（在本文前面讨论过）
● 0xE6(Set Scaling 1:1)－鼠标用“应答”(0xFA)回应，然后使能1：1比例。（在本文前面讨论过）

对于标准鼠标而言只有“Resend”(0xFE)和“Error”(0xFC)命令会发送给主机。这两条命令的工作情况和主机到设备间的命令一样。

初始化

正常情况下，PS/2鼠标仅在计算机启动的守候被检测和初始化。因此，鼠标不能热插拔，只要你增加或移走PS/2鼠标，你就要重新启动你的机器。此外，如果你在计算机开着的时候增加或移走PS/2鼠标，可能会损坏某些主板。

PS/2鼠标的初始检测发生在POST期间。如果鼠标检测到了，BIOS将允许操作系统配置/使能鼠标。否则OS将禁止在鼠标总线上的通讯。如果你启动一台插上鼠标的计算机，然后在windows里拔掉又重新插上鼠标，OS也许能检测到鼠标又重新插上了。Microsoft显然想试着支持这样的操作，但是只可能大约50%的情况下，它是工作的。

下面是在我的计算机（运行Win98SE）和鼠标之间的通讯，启动的时候插有一个PS/2鼠标。PS/2鼠标的初始化过程相当的典型。如果你要仿真一个PS/2鼠标，必须（至少）能支持如下的命令序列：

上电复位：

```
Mouse: AA Self-test passed
Mouse: 00 Mouse ID
Host:  FF Reset command
Mouse: FA Acknowledge
Mouse: AA Self-test passed
Mouse: 00 Mouse ID
Host:  FF Reset command
Mouse: FA Acknowledge
Mouse: AA Self-test passed
Mouse: 00 Mouse ID
Host:  FF Reset command
Mouse: FA Acknowledge
Mouse: AA Self-test passed
Mouse: 00 Mouse ID
Host:  F3 Set Sample Rate     : Attempt to Enter Microsoft
Mouse: FA Acknowledge         : Scrolling Mouse mode
Host:  C8 decimal 200         :
Mouse: FA Acknowledge         :
Host:  F3 Set Sample Rate     :
Mouse: FA Acknowledge         :
Host:  64 decimal 100         :
Mouse: FA Acknowledge         :
Host:  F3 Set Sample Rate     :
Mouse: FA Acknowledge         :
Host:  50 decimal 80          :
Mouse: FA Acknowledge         :
Host:  F2 Read Device Type    :
Mouse: FA Acknowledge         :
Mouse: 00 Mouse ID            : Response 03 if microsoft scrolling mouse
Host:  F3 Set Sample Rate
Mouse: FA Acknowledge
Host:  0A decimal 10
Mouse: FA Acknowledge
Host:  F2 Read Device Type
Mouse: FA Acknowledge
Mouse: 00 Mouse ID
```

```
Host:  E8 Set resolution
Mouse: FA Acknowledge
Host:  03 8 Counts/mm
Mouse: FA Acknowledge
Host:  E6 Set Scaling 1:1
Mouse: FA Acknowledge
Host:  F3 Set Sample Rate
Mouse: FA Acknowledge
Host:  28 decimal 40
Mouse: FA Acknowledge
Host:  F4 Enable
Mouse: FA Acknowledge
```
初始化完成；

如果我按下左键：
```
Mouse: 09 1 1 00001001 bit0 = Left button state; bit3 = always 1
Mouse: 00 1 1 No X-movement
Mouse: 00 1 1 No Y-movement
```

然后释放左键：
```
Mouse: 08 0 1 00001000 bit0 = Left button state; bit3 = always 1
Mouse: 00 1 1 No X-movement
Mouse: 00 1 1 No Y-movement
```

下面还是我的计算机和鼠标间的通讯，但这次启动时插有一个仿真的Intellimouse：

上电复位：
```
Mouse: AA Self-test passed
Mouse: 00 Mouse ID
Host:  FF Reset command
Mouse: FA Acknowledge
Mouse: AA Self-test passed
Mouse: 00 Mouse ID
Host:  FF Reset command
Mouse: FA Acknowledge
Mouse: AA Self-test passed
Mouse: 00 Mouse ID
Host:  FF Reset command
Mouse: FA Acknowledge
Mouse: AA Self-test passed
Mouse: 00 Mouse ID
Host:  F3 Set Sample Rate         : Attempt to Enter Microsoft
Mouse: FA Acknowledge             : Scrolling Mouse mode
Host:  C8 decimal 200             :
Mouse: FA Acknowledge             :
Host:  F3 Set Sample Rate         :
Mouse: FA Acknowledge             :
Host:  64 decimal 100             :
Mouse: FA Acknowledge             :
Host:  F3 Set Sample Rate         :
```

```
Mouse: FA Acknowledge            :
Host:  50 decimal 80             :
Mouse: FA Acknowledge            :
Host:  F2 Read Device Type       :
Mouse: FA Acknowledge            :
Mouse: 03 Mouse ID               : Response 03 if microsoft scrolling mouse
Host:  E8 Set Resolution
Mouse: FA Acknowledge
Host:  03 8 counts/mm
Mouse: FA Acknowledge
Host:  E6 Set scaling 1:1
Dev:   FA Acknowledge
Host:  F3 Set Sample Rate
Mouse: FA Acknowledge
Host:  28 decimal 40
Mouse: FA Acknowledge
Host:  F4 Enable device
Mouse: FA Acknowledge
```
初始化完成；

如果我按下鼠标左键：
```
Mouse: 09 00001001 bit0 = Left button state; bit3 = always 1
Mouse: 00 No X-movement
Mouse: 00 No Y-movement
Mouse: 00 No Z-movement
```

然后释放鼠标左键：
```
Mouse: 08 00001000 bit0 = Left button state; bit3 = always 1
Mouse: 00 No X-movement
Mouse: 00 No Y-movement
Mouse: 00 No Z-movement
```

在我下载并安装了微软支持第4和第5键的Intellimouse的驱动后，发现了如下序列：

```
...开始的部分和前面相同...
Host:  F3 Set Sample Rate      : Attempt to Enter Microsoft
Mouse: FA Acknowledge          : Scrolling Mouse mode.
Host:  C8 decimal 200          :
Mouse: FA Acknowledge          :
Host:  F3 Set Sample Rate      :
Mouse: FA Acknowledge          :
Host:  64 decimal 100          :
Mouse: FA Acknowledge          :
Host:  F3 Set Sample Rate      :
Mouse: FA Acknowledge          :
Host:  50 decimal 80           :
Mouse: FA Acknowledge          :
Host:  F2 Read Device Type     :
Mouse: FA Acknowledge          :
```

```
Mouse: 03 Mouse ID             : Response 03 if microsoft scrolling mouse.
Host:  F3 Set Sample Rate      : Attempt to Enter Microsoft 5-button
Mouse: FA Acknowledge          : Scrolling Mouse mode.
Host:  C8 decimal 200          :
Mouse: FA Acknowledge          :
Host:  F3 Set Sample Rate      :
Mouse: FA Acknowledge          :
Host:  C8 decimal 200          :
Mouse: FA Acknowledge          :
Host:  F3 Set Sample Rate      :
Mouse: FA Acknowledge          :
Host:  50 decimal 80           :
Mouse: FA Acknowledge          :
Host:  F2 Read Device Type     :
Mouse: FA Acknowledge          :
Mouse: 04 Mouse ID             : Response 04 if 5-button scrolling mouse.
```
...初始化的剩余部分和前面相同...

仿真/接口

如果你要建立一个真正的完全实现的鼠标或主机，你应该实现本文描述的所有特征（当然除了微软的Intellimouse的扩展，因为它是可选的）。但最起码你的设备要可以象下面讲的那样操作：

要想仿真一个鼠标：

- 绝对不要在时钟线为低的时候发送数据。如果主机拉低时钟线，就准备从主机读取数据。
- 上电后500ms左右发送0xAA 0x00。
- 在发送任何位移/按键数据前，等待主机发送使能(0xF4)命令。
- 要仿真的各种鼠标功能如下：

Emulated Action	Data sent to host
Move up one	0x08,0x00,0x01
Move down one	0x28,0x00,0xFF
Move right one	0x08,0x01,0x00
Move left one	0x18,0xFF,0x00
Press left button	0x09,0x00,0x00
Release left button	0x08,0x00,0x00
Press middle button	0x0C,0x00,0x00
Release middle button	0x08,0x00,0x00
Press right button	0x0A,0x00,0x00
Release right button	0x08,0x00,0x00

- 用“0xFA”回应“Reset”(0xFF)命令，然后跳转到你程序的开始处。（即，发送0xAA 0x00，在发送位移数据前等待使能命令。）
- 用“0xFA 0x00”回应“获得设备ID”(0xF2)命令。
- 用“0xFA 0x00 0x02 0x64”回应“状态请求”(0xE9)命令。
- 用“应答”(0xFA)回应所有其他的命令。

要和鼠标接口：

- 等待鼠标发送“0xAA”后，然后发送“使能”(0xF4)命令。

● 然后鼠标将发送本文前面描述的那种3字节的位移数据包。

脚注：

1) 2:1比例仅适用于Stream模式自动数据报告中，对于回应“Read Data”(0xEB)命令的报告数据是无效的。

2) 鼠标和主机不缓冲“Resend”(0xFF)命令。这意味着“0xFE”绝不会作为“Resend”命令的回应来发送。

3) 一个数据包可以是3字节的位移数据包，或4字节的位移数据包(Intellimouse的)，或3字节的状态包(见“Status Request”(0xE9)命令)，或2字节的完成代码ID包(0xAA 0x00或0xFC 0x00)，或1字节的命令回应。

4) 我自己经验的一点小建议：即使位移数据包中首字节的第3位被设置了，某些驱动（诸如标准PS/2鼠标驱动，包括windows98SE）的并不关心且指示忽略掉这一位。但是其他驱动可能会检查这一位是否设置了，如果没设置则认为是一个错误。我提及这点的意思是，如果你设计一个鼠标，你要检查由你的鼠标发送出来的每个位移数据包中这一位是否被设置了。如果没有，那么你的鼠标在你计算机上测试时可能工作得很好，而到别的使用不同鼠标驱动的计算机上就不工作了。例如，如果使用MS Intellimouse驱动，位移数据包中首字节的第3位没有设置，驱动程序将丢弃这个包，然后发送“Disable Data Reporting”(0xF5)命令，跟着是“Set Defaults”(0xF6)命令。然后使用windows启动时相同的命令序列重新初始化鼠标(见上面的初始化部分)。

附录一 第一套键盘扫描码

*所有的值都是十六进制的。

101、102 和 104 键的键盘：

KEY	MAKE	BREAK	KEY	MAKE	BREAK	KEY	MAKE	BREAK
A	1E	9E	9	0A	8A	[	1A	9A
B	30	B0	`	29	89	INSERT	E0,52	E0,D2
C	2E	AE	-	0C	8C	HOME	E0,47	E0,97
D	20	A0	=	0D	8D	PG UP	E0,49	E0,C9
E	12	92	\	2B	AB	DELETE	E0,53	E0,D3
F	21	A1	BKSP	0E	8E	END	E0,4F	E0,CF
G	22	A2	SPACE	39	B9	PG DN	E0,51	E0,D1
H	23	A3	TAB	0F	8F	U ARROW	E0,48	E0,C8
I	17	97	CAPS	3A	BA	L ARROW	E0,4B	E0,CB
J	24	A4	L SHFT	2A	AA	D ARROW	E0,50	E0,D0
K	25	A5	L CTRL	1D	9D	R ARROW	E0,4D	E0,CD
L	26	A6	L GUI	E0,5B	E0,DB	NUM	45	C5
M	32	B2	L ALT	38	B8	KP /	E0,35	E0,B5
N	31	B1	R SHFT	36	B6	KP *	37	B7
O	18	98	R CTRL	E0,1D	E0,9D	KP -	4A	CA
P	19	99	R GUI	E0,5C	E0,DC	KP +	4E	CE
Q	10	19	R ALT	E0,38	E0,B8	KP EN	E0,1C	E0,9C
R	13	93	APPS	E0,5D	E0,DD	KP .	53	D3
S	1F	9F	ENTER	1C	9C	KP 0	52	D2
T	14	94	ESC	01	81	KP 1	4F	CF
U	16	96	F1	3B	BB	KP 2	50	D0
V	2F	AF	F2	3C	BC	KP 3	51	D1
W	11	91	F3	3D	BD	KP 4	4B	CB
X	2D	AD	F4	3E	BE	KP 5	4C	CC
Y	15	95	F5	3F	BF	KP 6	4D	CD
Z	2C	AC	F6	40	C0	KP 7	47	C7
0	0B	8B	F7	41	C1	KP 8	48	C8
1	02	82	F8	42	C2	KP 9	49	C9
2	03	83	F9	43	C3	]	1B	9B
3	04	84	F10	44	C4	;	27	A7
4	05	85	F11	57	D7	'	28	A8
5	06	86	F12	58	D8	,	33	B3
6	07	87	PRNT SCRN	E0,2A, E0,37	E0,B7, E0,AA	.	34	B4
7	08	88	SCROLL	46	C6	/	35	B5
8	09	89	PAUSE	E1,1D,45 E1,9D,C5	-NONE-			

ACPI 扫描码

Key	**Make Code**	**Break Code**
Power	E0, 5E	E0, DE
Sleep	E0, 5F	E0, DF
Wake	E0, 63	E0, E3

Windows 多媒体扫描码：

Key	Make Code	Break Code
Next Track	E0, 19	E0, 99
Previous Track	E0, 10	E0, 90
Stop	E0, 24	E0, A4
Play/Pause	E0, 22	E0, A2
Mute	E0, 20	E0, A0
Volume Up	E0, 30	E0, B0
Volume Down	E0, 2E	E0, AE
Media Select	E0, 6D	E0, ED
E-Mail	E0, 6C	E0, EC
Calculator	E0, 21	E0, A1
My Computer	E0, 6B	E0, EB
WWW Search	E0, 65	E0, E5
WWW Home	E0, 32	E0, B2
WWW Back	E0, 6A	E0, EA
WWW Forward	E0, 69	E0, E9
WWW Stop	E0, 68	E0, E8
WWW Refresh	E0, 67	E0, E7
WWW Favorites	E0, 66	E0, E6

附录二 第二套键盘扫描码

*所有的值都是十六进制的。

101、102 和 104 键的键盘：

KEY	MAKE	BREAK	KEY	MAKE	BREAK	KEY	MAKE	BREAK
A	1C	F0,1C	9	46	F0,46	[	54	FO,54
B	32	F0,32	`	0E	F0,0E	INSERT	E0,70	E0,F0,70
C	21	F0,21	-	4E	F0,4E	HOME	E0,6C	E0,F0,6C
D	23	F0,23	=	55	FO,55	PG UP	E0,7D	E0,F0,7D
E	24	F0,24	\	5D	F0,5D	DELETE	E0,71	E0,F0,71
F	2B	F0,2B	BKSP	66	F0,66	END	E0,69	E0,F0,69
G	34	F0,34	SPACE	29	F0,29	PG DN	E0,7A	E0,F0,7A
H	33	F0,33	TAB	0D	F0,0D	U ARROW	E0,75	E0,F0,75
I	43	F0,43	CAPS	58	F0,58	L ARROW	E0,6B	E0,F0,6B
J	3B	F0,3B	L SHFT	12	FO,12	D ARROW	E0,72	E0,F0,72
K	42	F0,42	L CTRL	14	FO,14	R ARROW	E0,74	E0,F0,74
L	4B	F0,4B	L GUI	E0,1F	E0,F0,1F	NUM	77	F0,77
M	3A	F0,3A	L ALT	11	F0,11	KP /	E0,4A	E0,F0,4A
N	31	F0,31	R SHFT	59	F0,59	KP *	7C	F0,7C
O	44	F0,44	R CTRL	E0,14	E0,F0,14	KP -	7B	F0,7B
P	4D	F0,4D	R GUI	E0,27	E0,F0,27	KP +	79	F0,79
Q	15	F0,15	R ALT	E0,11	E0,F0,11	KP EN	E0,5A	E0,F0,5A
R	2D	F0,2D	APPS	E0,2F	E0,F0,2F	KP .	71	F0,71
S	1B	F0,1B	ENTER	5A	F0,5A	KP 0	70	F0,70
T	2C	F0,2C	ESC	76	F0,76	KP 1	69	F0,69
U	3C	F0,3C	F1	05	F0,05	KP 2	72	F0,72
V	2A	F0,2A	F2	06	F0,06	KP 3	7A	F0,7A
W	1D	F0,1D	F3	04	F0,04	KP 4	6B	F0,6B
X	22	F0,22	F4	0C	F0,0C	KP 5	73	F0,73
Y	35	F0,35	F5	03	F0,03	KP 6	74	F0,74
Z	1A	F0,1A	F6	0B	F0,0B	KP 7	6C	F0,6C
0	45	F0,45	F7	83	F0,83	KP 8	75	F0,75
1	16	F0,16	F8	0A	F0,0A	KP 9	7D	F0,7D
2	1E	F0,1E	F9	01	F0,01	]	5B	F0,5B
3	26	F0,26	F10	09	F0,09	;	4C	F0,4C
4	25	F0,25	F11	78	F0,78	'	52	F0,52
5	2E	F0,2E	F12	07	F0,07	,	41	F0,41
6	36	F0,36	PRNT SCRN	E0,12, E0,7C	E0,F0, 7C,E0, F0,12	.	49	F0,49
7	3D	F0,3D	SCROLL	7E	F0,7E	/	4A	F0,4A
8	3E	F0,3E	PAUSE	E1,14,77, E1,F0,14, F0,77	-NONE-			

ACPI 扫描码：

Key	Make Code	Break Code
Power	E0, 37	E0, F0, 37
Sleep	E0, 3F	E0, F0, 3F
Wake	E0, 5E	E0, F0, 5E

Windows 多媒体扫描码：

Key	Make Code	Break Code
Next Track	E0, 4D	E0, F0, 4D
Previous Track	E0, 15	E0, F0, 15
Stop	E0, 3B	E0, F0, 3B
Play/Pause	E0, 34	E0, F0, 34
Mute	E0, 23	E0, F0, 23
Volume Up	E0, 32	E0, F0, 32
Volume Down	E0, 21	E0, F0, 21
Media Select	E0, 50	E0, F0, 50
E-Mail	E0, 48	E0, F0, 48
Calculator	E0, 2B	E0, F0, 2B
My Computer	E0, 40	E0, F0, 40
WWW Search	E0, 10	E0, F0, 10
WWW Home	E0, 3A	E0, F0, 3A
WWW Back	E0, 38	E0, F0, 38
WWW Forward	E0, 30	E0, F0, 30
WWW Stop	E0, 28	E0, F0, 28
WWW Refresh	E0, 20	E0, F0, 20
WWW Favorites	E0, 18	E0, F0, 18

附录三 第三套键盘扫描码

KEY	MAKE	BREAK	KEY	MAKE	BREAK	KEY	MAKE	BREAK
A	1C	F0,1C	9	46	F0,46	[	54	F0,54
B	32	F0,32	`	0E	F0,0E	INSERT	67	F0,67
C	21	F0,21	-	4E	F0,4E	HOME	6E	F0,6E
D	23	F0,23	=	55	F0,55	PG UP	6F	F0,6F
E	24	F0,24	\	5C	F0,5C	DELETE	64	F0,64
F	2B	F0,2B	BKSP	66	F0,66	END	65	F0,65
G	34	F0,34	SPACE	29	F0,29	PG DN	6D	F0,6D
H	33	F0,33	TAB	0D	F0,0D	U ARROW	63	F0,63
I	43	F0,48	CAPS	14	F0,14	L ARROW	61	F0,61
J	3B	F0,3B	L SHFT	12	F0,12	D ARROW	60	F0,60
K	42	F0,42	L CTRL	11	F0,11	R ARROW	6A	F0,6A
L	4B	F0,4B	L WIN	8B	F0,8B	NUM	76	F0,76
M	3A	F0,3A	L ALT	19	F0,19	KP /	4A	F0,4A
N	31	F0,31	R SHFT	59	F0,59	KP *	7E	F0,7E
O	44	F0,44	R CTRL	58	F0,58	KP -	4E	F0,4E
P	4D	F0,4D	R WIN	8C	F0,8C	KP +	7C	F0,7C
Q	15	F0,15	R ALT	39	F0,39	KP EN	79	F0,79
R	2D	F0,2D	APPS	8D	F0,8D	KP .	71	F0,71
S	1B	F0,1B	ENTER	5A	F0,5A	KP 0	70	F0,70
T	2C	F0,2C	ESC	08	F0,08	KP 1	69	F0,69
U	3C	F0,3C	F1	07	F0,07	KP 2	72	F0,72
V	2A	F0,2A	F2	0F	F0,0F	KP 3	7A	F0,7A
W	1D	F0,1D	F3	17	F0,17	KP 4	6B	F0,6B
X	22	F0,22	F4	1F	F0,1F	KP 5	73	F0,73
Y	35	F0,35	F5	27	F0,27	KP 6	74	F0,74
Z	1A	F0,1A	F6	2F	F0,2F	KP 7	6C	F0,6C
0	45	F0,45	F7	37	F0,37	KP 8	75	F0,75
1	16	F0,16	F8	3F	F0,3F	KP 9	7D	F0,7D
2	1E	F0,1E	F9	47	F0,47	]	5B	F0,5B
3	26	F0,26	F10	4F	F0,4F	;	4C	F0,4C
4	25	F0,25	F11	56	F0,56	'	52	F0,52
5	2E	F0,2E	F12	5E	F0,5E	,	41	F0,41
6	36	F0,36	PRNT SCRN	57	F0,57	.	49	F0,49
7	3D	F0,3D	SCROLL	5F	F0,5F	/	4A	F0,4A
8	3E	F0,3E	PAUSE	62	F0,62			