广海社区 WWW.GHOFFICE.COM

11811 阅读 | 8 回复

(分享)模拟键盘,鼠标的一些资料,想做驱动的也可以来看下 [复制链接]

smartqiu

二星会员

发帖 259
金钱 337
威望 333
贡献值 2
社区警告 0
诚信值 0
账号服务 0
热心值 0

加关注 发消息

0楼 发表于: 2007-06-18

只看楼主 倒序阅读

今天用winio做个模拟键盘找了好些资料,贴出来与大家分享

终于弄清了要在写数据前为什么是先读0x64的第0位不为0(这样才能产生中断)
还有看他们的资料都是把释放按键 写成 and 0x80(我不知道是不是我
理解错误,and 在VB中应该是相与吧,我不懂VB),断码应是扫描码的最高位置1,所以应该是 OR 0x80

//--------下面是一些资料----------
PORT 0060-006F - KEYBOARD CONTROLLER 804x (8041, 8042) (or PPI (8255) on PC,XT)
Note: XT uses ports 60h-63h, AT uses ports 60h-64h

0060 RW KB controller data port or keyboard input buffer (ISA, EISA)
should only be read from after status port bit0 = 1
should only be written to if status port bit1 = 0
0060 R- KeyBoard or KB controller data output buffer (via PPI on XT)
PC: input from port A of 8255, if bit7 in 61h set (see #P063)
get scancodes, special codes (in PC: with bit7 in 61h cleared)
(see #P057)

0061 R- KB controller port B control register (ISA, EISA)
system control port for compatibility with 8255 (see #P060)
0061 -W KB controller port B (ISA, EISA) (PS/2 port A is at 0092)
system control port for compatibility with 8255 (see #P059)
0061 -W PPI Programmable Peripheral Interface 8255 (XT only)
system control port (see #P061)
0062 RW PPI (XT only) data port C (see #P062)
0063 RW PPI (XT only) command mode register (see #P064)

0064 R- keyboard controller read status (see #P065,#P066,#P067)
0064 -W keyboard controller input buffer (ISA, EISA) (see #P068)

0064 -W (Amstrad/Schneider PC1512) set 'DIP switch S1' setting
stored in CMOS RAM that PPI should report for compatibility
0065 -W (Amstrad/Schneider PC1512) set 'DIP switch S2' RAM size setting
stored in CMOS RAM, that PPI port C (PORT 0064h) should report for
compatibility

0065 R- communications port (Olivetti M24)

0068 -W (HP-Vectra) control buffer (HP commands) (see #P069)
0069 R- (HP-Vectra) SVC (keyboard request SerViCe port)
006A -W (HP-Vectra) Acknowledge (clear processing, done)

006C-006F HP-HIL (Human Interface Link = async. serial inputs 0-7)

Bitfields for AT keyboard controller input port:
Bit(s) Description (Table P048)
7 keyboard enabled
6 =0 CGA, else MDA
5 =0 manufacturing jumper installed
4 =0 system RAM 512K, else 640K
3-0 reserved
SeeAlso: #P049,#P051

Bitfields for AT keyboard controller input port (Compaq):
Bit(s) Description (Table P049)
7 security lock is unlocked
6 =0 Compaq dual-scan display, 1=non-Compaq display
5 system board dip switch 5 is OFF
4 =0 auto speed selected, 1=high speed selected
3 =0 slow (4MHz), 1 = fast (8MHz)
2 no math coprocessor installed
1-0 reserved
SeeAlso: #P050

Bitfields for AT keyboard controller output port:
Bit(s) Description (Table P050)
7 keyboard data output
6 keyboard clock output
5 input buffer NOT full
4 output buffer NOT empty
3 reserved (see note)
2 reserved (see note)
1 gate A20
0 system reset

Note: bits 2 and 3 are the turbo speed switch or password lock on
  Award/AMI/Phoenix BIOSes. These bits make use of nonstandard
  keyboard controller BIOS functionality to manipulate
    pin 23 (8041 port 22) as turbo switch for AWARD
    pin 35 (8041 port 15) as turbo switch/pw lock for Phoenix
SeeAlso: #P048,#P051

Bitfields for HP Vectra keyboard controller output port:
Bit(s) Description (Table P051)
7-5 reserved
4 output buffer full (OBF) interrupt
3 HP SVC interrupt
2 HP-HIL controller AutoPoll
1 A20 gate
0 system reset
SeeAlso: #P050,#P052

Bitfields for HP Vectra command byte:
Bit(s) Description (Table P052)
7 reserved (0)
6 scancode conversion mode (1 = PC/XT, 0 = PC/AT)
5 unused
4 disable keyboard (unless bit 3 set)
3 override keyboard disable
2 System Flag (may be read from port 0060h)
1 reserved
0 OBF interrupt enable
SeeAlso: #P051

(Table P053)
Values for keyboard commands (data also goes to PORT 0060h):
Value Count Description

EDh double set/reset mode indicators Caps Num Scrl
bit 2 = CapsLk, bit 1 = NumLk, bit 0 = ScrlLk
all other bits must be zero.
EEh sngl diagnostic echo. returns EEh.
EFh sngl NOP (No OPeration). reserved for future use
EF+26h double [Cherry MF2 G80-1501HAD] read 256 bytes of chipcard data
keyboard must be disabled before this and has to
be enabled after finished.
F0h double get/set scan code set
00h get current set
01h scancode set 1 (PCs and PS/2 mod 30, except Type 2 ctrlr)
02h scancode set 2 (ATs, PS/2, default)
03h scancode set 3
F2h sngl read keyboard ID (read two ID bytes)
AT keyboards returns FA (ACK)
MF2 returns AB 41 (translation) or
AB 83 (pass through)
F3h double set typematic rate/delay
format of the second byte:
bit7=0 : reserved
bit6-5 : typemativ delay
00b=250ms 10b= 750ms
01b=500ms 11b=1000ms
bit4-0 : typematic rate (see #P058)
F4h sngl enable keyboard
F5h sngl disable keyboard. set default parameters (no keyboard scanning)
F6h sngl set default parameters
F7h sngl [MCA] set all keys to typematic (scancode set 3)
F8h sngl [MCA] set all keys to make/release
F9h sngl [MCA] set all keys to make only
FAh sngl [MCA] set all keys to typematic/make/release
FBh sngl [MCA] set al keys to typematic

FCh double [MCA] set specific key to make/release
FDh double [MCA] set specific key to make only
FEh sngl resend last scancode
FFh sngl perform internal power-on reset function
Note: each command is acknowledged by FAh (ACK), if not mentioned otherwise.
  See PORT 0060h-R for details.
SeeAlso: #P054

(Table P054)
Values for Mouse functions (for PS/2-like pointing devices):
Value Count Description
E6h sngl set mouse scaling to 1:1
E7h sngl set mouse scaling to 2:1
E8h double set mouse resolution
  (00h=1/mm, 01h=2/mm, 02h=4/mm, 03h=8/mm)
E9h sngl get mouse information
  read two status bytes:
   byte 0: flags (see #P055)
   byte 1: resolution
EAh sngl set mouse to stream mode (mouse sends data on any changes)
EBh sngl get mouse data (from mouse to controller) (see #P056)
  on reading, each data packet consists of 8 bytes:
ECh sngl reset mouse wrap mode (to normal mode)
EEh sngl set wrap mode
F0h sngl set remote mode (instead of stream mode), mouse sends data
   only on issueing command EBh.
F2h sngl read mouse ID (read one, two?? ID bytes)
  00h=mouse
F3h double set mouse sample rate in reports per second
  0Ah=10/s 50h= 80/s
  14h=20/s 64h=100/s
  28h=40/s C8h=200/s

3Ch=60/s
F4h sngl enable mouse (in stream mode)
F5h sngl disable mouse (in steam mode), set default parameters
F6h sngl reset to defaults: 100/s, scaling 1:1, stream-mode, 4/mm,
disabled
FEh sngl resend last mouse data (8 bytes, see EBh)
FFh sngl reset mouse
Notes: must issue command D4h to port 64h first to access mouse functions
all commands except ECh and FFh are acknowledged by FAh (ACK) or
FEh (Resend); get mouse ID (F2h) returns mouse ID.
SeeAlso: #P053

Bitfields for mouse status byte 0:
Bit(s) Description (Table P055)
7 unused
6 remote rather than stream mode
5 mouse enabled
4 scaling set to 2:1
3 unused
2 left button pressed
1 unused
0 right button pressed
SeeAlso: #P054,#P056

Format of mouse data packet:
Offset Size Description (Table P056)
00h BYTE status
bit7 : y-data overrun
bit6 : x-data overrun
bit5 : y-data negative
bit4 : x-data negative
bit3-2=0: reserved

bit1 : right button pressed
bit0 : left button pressed
01h BYTE reserved
02h BYTE x-data
03h BYTE reserved
04h BYTE y-data
05h BYTE reserved
06h BYTE z-data (0)
07h BYTE reserved
SeeAlso: #P054,#P055

(Table P057)
Values for keyboard special codes:
00h (MF2 in codeset2&3 or AT keyboards) keydetection/overrun error
00h (mouse) ID
AAh BAT completion code (sent after errorfree Basic Assurance Test)
ABh first byte of general MF2 keyboard ID
EEh Echo command return
FAh Acknowledge (all general commands except Resend and Echo)
FAh (mouse) Acknowledge (all commands except commands ECh,F2h,FFh)
FCh (MF2) BAT Failure Code (error in second half of the power on self test)
FDh (AT-keyboard) BAT Failure Code (error in the second half of the power-on self test)
FEh Resend: CPU to controller should resend last keyboard-command
FEh (mouse) CPU to controller should resend last mouse-command
FFh (MF2 in codeset1) keydetection/overrun error
Note: keyboard stops scanning and waits for next command after returning code FCh or FDh
SeeAlso: PORT 0060h-R

(Table P058)
Values for keyboard typematic rate:

00000b=30.0 10000b=7.5
00001b=26.7 10001b=6.7
00010b=24.0 10010b=6.0
00011b=21.8 10011b=5.5
00100b=20.0 10100b=5.0
00101b=18.5 10101b=4.6
00110b=17.1 10110b=4.3
00111b=16.0 10111b=4.0
01000b=15.0 11000b=3.7
01001b=13.3 11001b=3.3
01010b=12.0 11010b=3.0
01011b=10.9 11011b=2.7
01100b=10.0 11100b=2.5
01101b= 9.2 11101b=2.3
01110b= 8.5 11110b=2.1
01111b= 8.0 11111b=2.0
SeeAlso: #P053

Bitfields for KB controller port B (system control port) [output]:
Bit(s) Description (Table P059)
7 pulse to 1 for IRQ1 reset (PC,XT)
6-4 reserved
3 I/O channel parity check disable
2 RAM parity check disable
1 speaker data enable
0 timer 2 gate to speaker enable
SeeAlso: PORT 0061h-W,#P060

Bitfields for KB controller port B control register (system control port) [input]:
Bit(s) Description (Table P060)
7 RAM parity error occurred
6 I/O channel parity error occurred

5 mirrors timer 2 output condition
4 toggles with each refresh request
3 NMI I/O channel check status
2 NMI parity check status
1 speaker data status
0 timer 2 clock gate to speaker status
SeeAlso: PORT 0061h-R,#P059

Bitfields for Progr. Peripheral Interface (8255) system control port [output]:
Bit(s) Description (Table P061)
7 clear keyboard (only pulse, normally kept at 0)
6 =0 hold keyboard clock low
5 NMI I/O parity check disable
4 NMI RAM parity check disable
3 =0 read low nybble of switches S2
   =1 read high nybble of switches S2
2 reserved, often used as turbo switch
   original PC: cassette motor off
1 speaker data enable
0 timer 2 gate to speaker enable
Note: bits 2 and 3 are sometimes used as turbo switch
SeeAlso: PORT 0061h-W,#P0051,#P062,#P063,#P064

Bitfields for PPI (XT only) data port C:
Bit(s) Description (Table P062)
7 RAM parity error occurred
6 I/O channel parity error occurred
5 timer 2 channel out
4 reserved
   original PC: cassette data input
---
3 system board RAM size type 1

2 system board RAM size type 2
1 coprocessor installed
0 loop in POST

3-0 DIL switch S2 high/low nybble (depending on PORT 0061h bit 3)
SeeAlso: PORT 0062h-RW,#P061,#P063,#P064

Bitfields for PPI (PC,XT only) equipment switches [input]:
Bit(s) Description (Table P063)
7-6 number of disk drives
  00 1 diskette drive
  01 2 diskette drives
  10 3 diskette drives
  11 4 diskette drives
5-4 initial video
  00 reserved (video adapter has on-board BIOS)
  01 40*25 color (mono mode)
  10 80*25 color (mono mode)
  11 MDA 80*25
3-2 memory size (using 256K chips)
  00 256K
  01 512K
  10 576K
  11 640K
3-2 memory size (using 64K chips)
  00 64K
  01 128K
  10 192K
  11 256K
3-2 memory size (original PC)
  00 16K
  01 32K

10 48K
11 64K
1-0 reserved
1 NPU (math coprocessor) present
0 boot from floppy
SeeAlso: #P062,#P064,PORT 0060h-R

Bitfields for PPI (8255) command mode register:
Bit(s) Description (Table P064)
7 activation function (0 = bit set/reset, 1 = mode set function)
6,5 port A mode: 00 = mode0, 01 = mode1, 1x = mode2
4 port A direction: 0 = output, 1 = input
3 port C bits 7-4 direction: 0 = output, 1 = input
2 port B mode: 0 = mode0, 1 = mode1
1 port B direction: 0 = output, 1 = input
0 port C bits 3-0 direction: 0 = output, 1 = input
Note: Attention: Never write anything other than 99h to this port
(better: never write anything to this port, only during BIOS
init), as other values may connect multiple output drivers
and will cause hardware damage in PC/XTs! By setting command
word to 99h, PPI will be set in input/output modes as it is
necessary to support the commonly known IO-ports 60, 61, 62
as desired.
SeeAlso: #P061,#P062,#P063

Bitfields for keyboard controller read status (ISA, EISA):
Bit(s) Description (Table P065)
7 parity error on transmission from keyboard
6 receive timeout
5 transmit timeout
4 keyboard interface inhibited by keyboard lock
3 =1 data written to input register is command (PORT 0064h)

=0 data written to input register is data (PORT 0060h)
2 system flag status: 0=power up or reset 1=selftest OK
1 input buffer full (input 60/64 has data for 8042)
no write access allowed until bit clears
0 output buffer full (output 60 has data for system)
bit is cleared after read access
SeeAlso: PORT 0064h-R,#P066,#P067,#P068

Bitfields for keyboard controller read status (MCA):
Bit(s) Description (Table P066)
7 parity error on transmission from keyboard
6 general timeout
5 mouse output buffer full
4 keyboard interface inhibited by keyboard lock
3 =1 data written to input register is command (PORT 0064h)
=0 data written to input register is data (PORT 0060h)
2 system flag status: 0=power up or reset 1=selftest OK
1 input buffer full (60/64 has data for 804x)
no write access allowed until bit clears
0 output buffer full (output 60 has data for system)
bit is cleared after read access
SeeAlso: #P065,#P067,#P068

Bitfields for keyboard controller read status (Compaq):
Bit(s) Description (Table P067)
7 parity error detected (11-bit format only). If an
error is detected, a Resend command is sent to the
keyboard once only, as an attempt to recover.
6 receive timeout. transmission didn't finish in 2mS.
5 transmission timeout error
bit 5,6,7 cause
1 0 0 No clock

1 1 0 Clock OK, no response
1 0 1 Clock OK, parity error
4 =0 security lock engaged
3 =1 data in OUTPUT register is command
=0 data in OUTPUT register is data
2 system flag status: 0=power up or reset 1=soft reset
1 input buffer full (60/64 has data for 804x)
no write access allowed until bit clears
0 output buffer full (port 60 has data for system)
bit is cleared after read access
SeeAlso: #P065,#P066,#P068

(Table P068)
Values for keyboard controller commands (data goes to port 0060)::
Value Description
20h read read byte zero of internal RAM, this is the last KB command
sent to the 8041/8042
Compaq put current command byte on port 0060 (see #P070,#P071)
21-3F read reads the byte specified in the lower 5 bits of the command
in the 804x's internal RAM
60-7F double writes the data byte to the address specified in the 5 lower
bits of the command
60h Compaq Load new command (60 to [64], command to [60]) (see #P071)
(also general AT-class machines)
A0h AMI get ASCIZ copyright message on port 0060
A1h AMI get controller version byte on port 0060
A1h Compaq unknown speedfunction ??
A2h Compaq unknown speedfunction ??
A2h AMI set keyboard controller pins 22 and 23 low
A3h Compaq Enable system speed control
A3h AMI set keyboard controller pins 22 and 23 high
A4h MCA check if password installed

A4h Compaq Toggle speed
A4h AMI set internal system speed flag to low
A5h MCA load password
A5h AMI set internal system speed flag to high
A5h Compaq Special read. the 8042 places the real values of port 2
except for bits 4 and 5 wich are given a new definition in
the output buffer. No output buffer full is generated.
if bit 5 = 0, a 9-bit keyboard is in use
if bit 5 = 1, an 11-bit keyboard is in use
if bit 4 = 0, output-buff-full interrupt disabled
if bit 4 = 1, output-buffer-full interrupt enabled
A6h MCA check password
A6h AMI get internal system speed flag on port 0060
A6h Compaq unknown speedfunction ??
A7h MCA disable mouse port
A7h AMI set internal flag indicating bad write cache
A8h MCA enable mouse port
A8h AMI set internal flag indicating good write cache
A9h MCA test mouse port
A9h AMI get internal flag indicating cache OK to 0060
AAh sngl initiate self-test. will return 55h to data port if self-test
successful, FCh if failed
AAh Compaq initializes ports 1 and 2, disables the keyboard and clears
the buffer pointers. It then places 55h in the output buffer.
ABh sngl initiate interface test. result values:
00h no error
01h keyboard clock line stuck low
02h keyboard clock line stuck high
03h keyboard data line is stuck low
04h keyboard data line stuck high
05h (Compaq only) diagnostic feature
ACh read diagnostic dump. the contents of the 804x RAM, output port,

input port, status word are sent.
ADh sngl disable keyboard (sets bit 4 of commmand byte)
ADh Vectra HP Vectra diagnostic dump
AEh sngl enable keyboard (resets bit 4 of commmand byte)
AFh AWARD Enhanced Command: read keyboard version
B1h AMI set keyboard controller P11 line low
B2h AMI set keyboard controller P12 line low
B3h AMI set keyboard controller P13 line low
B4h AMI set keyboard controller P22 line low
B5h AMI set keyboard controller P23 line low
B8h AMI set keyboard controller P10 line high
B9h AMI set keyboard controller P11 line high
BAh AMI set keyboard controller P12 line high
BBh AMI set keyboard controller P13 line high
BCh AMI set keyboard controller P22 line high
BDh AMI set keyboard controller P23 line high
C0h read read input port and place on PORT 0060h
bit 7 keyboard NOT locked
bit 6 =0 first video is CGA
=1 first video is MDA
bit 5 =0 factory testmode
=1 normal
bit 4 =0 256KB RAM, 1=512KB
bit 5,3-0 are used in Intel chipset 386sx machines with
AMI/Phoenix BIOSes for BIOS specific hardware settings
C0h Compaq places status of input port in output buffer. Use this
command only when the output buffer is empty
C1h MCA Enhanced Command: poll input port Low nibble
C2h MCA Enhanced Command: poll input port High nibble
C8h AMI unblock keyboard controller lines P22 and P23
C9h AMI block keyboard controller lines P22 and P23
CAh AMI read keyboard mode, return in 0060 bit 0

(bit clear if ISA mode, set if PS/2 mode)
CBh AMI set keyboard mode (write back mode byte returned by CAh, modifying only bit 0)
D0h read read output port and place on PORT 0060h (see #P072)
D0h Compaq places byte in output port in output buffer. Use this command only when the output buffer is empty
D1h double write output port. The next byte written to port 0060h will be written to the 804x output port; the original IBM AT and many compatibles use bit 1 of the output port to control the A20 gate.
Important: bit 0 (system reset) should always be set here, as the system may hang constantly, use pulse output port (FEh) instead.
D1h Compaq the system speed bits are not set by this command use commands A1-A6 (!) for speed functions.
D2h MCA Enhanced Command: write keyboard output buffer
D3h MCA Enhanced Command: write pointing device out.buf.
D4h MCA write to mouse/pointing device instead of to keyboard; this controller command must precede every PORT 0060h command directed to the mouse, otherwise it will be sent to the keyboard
D4h AWARD Enhanced Command: write to auxiliary device
DDh sngl disable address line A20 (HP Vectra only???) default in Real Mode
DFh sngl enable address line A20 (HP Vectra only???)
E0h read read test inputs.
bit0 = kbd clock, bit1 = kbd data
Exxx AWARD Enhanced Command: active output port
EDh double this is a two part command to control the state of the NumLock, CpasLock and ScrollLock LEDs
The second byte contains the state to set LEDs.
bit 7-3 reserved. should be set to 0.

bit 2 = 0 Caps Lock LED off
bit 1 = 0 Num Lock LED off
bit 0 = 0 Scroll Lock LED off
F0-FF sngl pulse output port low for 6 microseconds.
bits 0-3 contain the mask for the bits to be pulsed. A bit is
pulsed if its mask bit is zero
bit0=system reset. Don't set to zero. Pulse only!
Note: keyboard controllers are widely different from each other. You
cannot generally exchange them between different machines.
(Award) Derived from Award's Enhanced KB controller advertising sheet.
(Compaq) Derived from the Compaq Deskpro 386 Tech. Ref. Guide.

(Table P069)
Values for HP Vectra control buffer command code:
00h-54h insert standard key make code into 8041 scancode buf
55h-77h insert HP key make code into 8041 scancode buffer
7Ah pass through next data byte
7Bh set RAM Switch to 0
7Ch set RAM Switch to 1 (default)
7Dh set CRT Switch to 0
7Eh set CRT Switch to 1 (default)
7Fh reserved
80h-D4h insert standard key break code into scancode buffer
D5h-F7h insert HP key break code into scancode buffer
F8h enable AutoPoll
F9h disable AutoPoll
FAh-FEh reserved
FFh keyboard overrun
SeeAlso: PORT 0068h-W

Bitfields for Compaq keyboard command byte:
Bit(s) Description (Table P070)

7 reserved
6 =1 convert KB codes to 8086 scan codes
5 =0 use 11-bit codes, 1=use 8086 codes
4 =0 enable keyboard, 1=disable keyboard
3 ignore security lock state
2 this bit goes into bit2 status reg.
1 reserved (0)
0 generate interrupt when output buffer full
SeeAlso: #P071

Bitfields for keyboard command byte (alternate description):
Bit(s) Description (Table P071)
7 reserved (0)
6 IBM PC compatibility mode
5 IBM PC mode
no parity, no stop bits, no translation
(PS/2) force mouse clock low
4 disable keyboard (clock)
3 inhibit override
(PS/2) reserved
2 system flag
1 reserved (0)
(PS/2) enable mouse output buffer full interrupt
0 enable output buffer full interrupt
SeeAlso: #P070,#P072

Bitfields for keyboard controller output port:
Bit(s) Description (Table P072)
7 keyboard data (output)
6 keyboard clock (output)
5 input buffer empty
4 output buffer empty

3 undefined
2 undefined
used by Intel 386sx Chipset with AMI/Phoenix BIOSes for BIOS-specific configuration of turbo switch
1 gate address A20
0 system reset
Note: bit 0 (system reset) should always be set when writing the output port, as the system may hang constantly; use pulse output port (command FEh) instead.

(QQ:834449529)

回复 举报

分享到

smartqiu

二星会员

发帖 259
金钱 337
威望 333

1楼 发表于: 2007-06-18 只看该作者

温馨提示：忘记账号、密码、安全问题及其他常见站务问题，请点此查看站点帮助

通过对i8042 键盘控制器编程控制鼠标

现在我们用的鼠标，大多是支持PS/2协议的鼠标，这样的鼠标也被称为PS/2 鼠标，PS/2协议其实支持两种设备，一种是键盘，一种是鼠标，它是由IBM公司制定的，协议的本身定义了键盘与鼠标同主机进行通迅的规则，其中包括了大量的物理及电器方面的信息，比如鼠标连接线的插头的管脚（针）数，每个管脚（针）的用途，电平是多少等，不过幸运的是，我们并不需要对这样的硬件细节有详细的了解，就可以完成我们的操作系统，我们需要了解的就是怎样初始化鼠标，以及怎样从鼠标中获得信息。

这里，我们首先来看看怎样初始化鼠标。根据PS/2协议，鼠标是由键盘的控制器(i8042)进行控制的，键盘控制器（i8042）总共有两个通道，一个通道由键盘使用，另一个通道由鼠标使用，我们对鼠标进行操作也是通过i8042芯片来完成

贡献值 2
社区警告 0
诚信值 0
账号服务 0
热心值 0
加关注 发消息

的，因此，现在的重点就是了解并熟悉怎样对i8042进行编程，来完成对鼠标的控制。

i8042支持两种工作模式--AT模式及PS/2模式，这都是由IBM所定义的一些规范，i8042在计算机启动时会自动检测用户是否使用的支持PS/2协议的键盘及鼠标，以决定是否工作在PS/2模式下，现在我们假设我们使用的都是PS/2键盘及鼠标，因此，现在i8042工作在PS/2模式下（请记住这一点，即i8042可以工作在AT模式或者PS/2模式下，并且现在假设其工作在PS/2模式下，这在后面将会用到）。

与i8042有关的I/O端口共有两个，一个是0x60端口，一个是0x64端口，如果我们想获得i8042的状态，我们需要读0x64端口，这将返回i8042中状态寄存器的内容。如果我们想向i8042发送命令，我们需要先把命令发送到0x64端口，然后紧接着把这个命令的参数发送到0x60端口，然后我们可以通过读0x60端口，获得i8042返回给我们的数据。

下面我们就来看看，应当发送什么样的命令去控制鼠标，这涉及到下面几个需要发送给i8042的命令：

0xA8命令：许可i8042的鼠标通道，即允许鼠标操作。

0xD4命令：把发往0x60端口的参数数据发给鼠标。

0x60命令：把发往0x60端口的参数数据写入i8042的控制寄存器。

从上面的分析我们可以基本窥见怎样操作鼠标。首先，我们应向i8042的0x64端口发送0xA8命令，以许可i8042的鼠标通道，以便完成对鼠标的操作。其次我们应向i8042的0x64端口发送0xD4命令，以通知i8042我们需要控制鼠标，并把控制鼠标的命令发到i8042的0x60端口，再从i8042的0x60端口取回鼠标发给我们的数据，这一过程无疑是比较简单的，我们先来看看，我们应向鼠标发送什么样的控制命令，然后再看看实际的代码。

控制鼠标的命令非常之多，比如0xFF命令可以让鼠标复位；0xFE命令可以让鼠标重新发送上次的数据包；0xF3命令可以设置鼠标的采样率，也即鼠标滑动时的灵敏度；0xF4命令可以允许鼠标向主机发送数据包等。这里最重要的就是0xF4命令，而其它的设置鼠标的命令我们暂时可以不用理会，因为使用默认值已经能很好的完成本实验了。要理解0xF4命令有什么作用，我们需要先了解一下鼠标的四种工作模式：Reset模式，Stream模式，Remote模式及Wrap模式。

首先是Reset模式，当鼠标初次加电或收到0xFF命令之后，鼠标就处于此模式，然后鼠标将进行一系列的初始化及检测工作，包括设定默认的采样率等，完成初始化极检测之后，鼠标将进入Stream模式。

在Stream模式下，如果鼠标被移动，或者有键被按下，那么鼠标将向主机发送数据包，并提请一个中断请求，主机需要响应此中断，并在中断处理程序中取得鼠标发送的数据包。如果在Stream模式下，我们向鼠标发送0xF0命令，将使鼠标进入Remote模式。

Remote模式同Stream模式差不多，主要工作就是检测鼠标是否被移动及是否有键被按下，不过它与Stream模式的区别在于，它并不会主动的向主机提请中断请求，也即它不会主动的向主机发送数据包，而是被动的等待主机使用0xEB(读数据命令)后，再向主机提请中断请求，发送数据包。换句话说，如果在Remote模式下，你每次欲读数据时，均需要向鼠标发送0xEB命令，而如果是在Stream模式下，鼠标会自动向你发送数据。

Wrap模式主要用于检测鼠标与主机之间的连线是否正常，主机向它发送命令，它一般不会执行此命令，而只是简单的将此命令原封不同的回送给主机，主机可比较它发出的命令及接收到的命令是否一致，并以此来认定鼠标与主机之间的连线是否正常。

从上面的描述中我们可以看出，我们需要关心的只有Reset模式及Stream模式，但对于操作系统编写人员而非BIOS编写人员来说，真正需要关心的只有Stream模式，这是因为当计算机启动的时候，BIOS会自动检测鼠标，与鼠标进行通信，这个时候它会向鼠标发送0xFF（复位）命令，然后鼠标会自检，并通知主机自检是否正常，然后鼠标就将处于Stream模式，此时，鼠标已经开始检测鼠标是否移动及是否有键按下了，但是它不会立即就向主机发送数据，因为有可能主机还没有进入真正的操作系统，主机还正在启动中，因此，鼠标会等待主机的通知，直到主机给它发送0xF4命令后，它才开始向主机发送数据。

所以，在操作系统中初始化鼠标的动作就很简单了，请看下面pyos中初始化鼠标的相关代码：

```
// 许可 鼠标

void mouse_enable_mouse()

{

  // 对 8042 键盘控制芯片进行编程
```

```
 // 允许 鼠标 接口

 io_write_to_io_port( 0x64 , 0xa8 ) ;

 // 通知 8042 下个字节的发向 0x60 的数据将发给 鼠标

 io_write_to_io_port( 0x64 , 0xd4 ) ;

 // 允许 鼠标 发数据

 io_write_to_io_port( 0x60 , 0xf4 ) ;

 // 通知 8042,下个字节的发向 0x60 的数据应放向 8042 的命令寄存器

 io_write_to_io_port( 0x64 , 0x60 ) ;

 // 许可键盘及 鼠标 接口及中断

 io_write_to_io_port( 0x60 , 0x47 ) ;

}
```

有了上面的描述，这段代码就相当简单了，首先它向i8042的0x64端口发送了一个0xA8命令，通知i8042，允许鼠标通道。然后，它向i8042的0x64端口发送了一个0xD4命令，这个命令表示，下面发给0x60的命令需要发给鼠标，所以，紧接着，它向i8042的0x60端口，也即向鼠标，发送了0xF4命令，这个命令将允许经过BIOS初始化后，现在已处于Stream模式下的鼠标给主机发送数据包。随后，它向i8042的0x64端口发送了0x60命令，表示，下面发向0x60端口的数据需要写入i8042的控制寄存器，最后它向i8042的0x60端口发送了值为0x47的数据，这个数据被写入了i8042的控制寄存器。下面，我们就来看看这个控制寄存器，以明白，这里为什么需要向它发送这样一个值为0x47的数据。

下面就是i8042的控制寄存器的格式，这个控制寄存器总共有8位，即1个字节。

位0：键盘中断标志位，如果置1，那么如果有键盘动作，则i8042将提请IRQ1中断。

位1：鼠标中断标志位，如果置1，那么如果有鼠标动作，则i8042将提请IRQ12中断（在AT模式下这位不使用，只在PS/2模式下有效。这里可以回忆一下前面我们提到的i8042可以工作在AT或者PS/2两种模式下的描述）。

位2：系统标志位，上电的时候是0，自检成功后是1。

位3：键盘锁标志位，如果是1，将忽略键盘锁，这主要是为了兼容一些老式的带锁的键盘，而且这位只在AT模式下使用，PS/2模式下将不使用此位。

位4：键盘接口标志位，如果置1，将禁止使用键盘接口。

位5：在AT模式下，这是AT键盘协议位。置0的时候，i8042将使用AT协议，如果置1，将使用XT协议。在PS/2模式下，这是鼠标接口（通道）标志位，如果置1将禁止鼠标接口（通道）。

位6：键盘扫描码转换标志位。如果置1将把真实的键盘扫描码转换为第一套键盘扫描码（有关此内容详见《编写操作系统之键盘交互的实现》一文）。

位7：保留，应置为0。

二、鼠标数据包简析

在上节中，我们知道了怎样对i8042进行编程，以完成对鼠标的初始化操作，那么当鼠标的初始化完成之后，如果有鼠标动作发生，比如用户移动了一下鼠标，或者按下了鼠标键，那么鼠标将把数据包发回给主机，现在我们就来看看鼠标发回给主机的数据包的结构，在了解这个结构之前，我们先要知道现存的总共有两类鼠标，一类就是所谓的2D（二维）鼠标，它就是我们平常用的那种没有滚轮的鼠标，由于这种鼠标在位移上只有X与Y两个方向，所以称之为2D（二维）鼠标；还有一类就是现在比较常见的3D（三维）鼠标，它们中间存在有一个滚轮，而这个滚轮会产生一个额外的Z位移量，因此，它在位移上有X、Y、Z三个方向，所以又称之为3D（三维）鼠标。下面，我们就来看看这两类鼠标发给主机的数据包有什么不同。下面，我们先来看看二维鼠标。

位0：左键按下标志位，为1表示左键被按下。

位1：右键按下标志位，为1表示右键被按下。

位2：中键按下标志位，为1表示中键被按下。

位3：保留位，总是为1。

位4：X符号标志位，为1表示X位移量为负。

位5：Y符号标志位，为1表示Y位移量为负。

位6：X溢出标志位，为1表示X位移量溢出了。

位7：Y溢出标下位，为1表示Y位移量溢出了。

下面，我们再来看看三维鼠标的数据包结构。

三维鼠标数据包中第一个数据包每位的含义与二维鼠标数据包中第一个数据包中每位含义完全相同，唯一不同的就在于它每次会多发送一个数据包，即第4个数据包，这个数据包包含了Z的位移量，同X、Y位移量相同的是，它们都是以补码表示的。不过与X及Y位移量不同的是，Z位移量是4位的，其中最高位（第四位）是符号位，因此，Z位移量的有效的范围为：-8~7。而X与Y的位移量是9位的，最高一位（第9位）是符号位，这个符号位在第一个数据包中表示，故，X与Y的位移量的有效范围为：-256~255。

看到这里，你或许有疑问了，系统是怎么来知道到我到底应当接收3个数据包还是接收4个数据包的呢？三维鼠标的标准是由微软制定的，最初，这种三维的鼠标只工作在标准的PS/2模式下，如果你想让它工作在三维模式下，你需要用0xF3这个设置鼠标采样率的命令，按如下的顺序进行操作：

1. 设置鼠标采样率为200

2. 设置鼠标采样率为100

3. 设置鼠标采样率为80

这之后，如果你的鼠标是个三维鼠标,那么，它将转到三维模式下进行工作，这个时候，主机向它发送0xF2（获得鼠标类型ID）命令，你的工作在三维模式下的鼠标将向主机返回它的类型ID，但如果你的鼠标不支持三维模式，即如果你的鼠标只是一个二维鼠标，它返回给主机的类型ID将是0，这样，主机就能够知道现在你用的鼠标是什么类型的鼠标，并由此知道应当接受3个还是4个数据包了。本实验将只操作标准的二维鼠标，如果你有兴趣，你可以对程序进行改动，以让它支持三维鼠标。

现在，我们来看看，这个数据包是怎么被主机获得的。在《保护模式下的8259A芯片编程及中断处理探究(上下)》这篇实验报告中我们知道了，鼠标中断是通过IRQ12提出的，因此，在操作系统中，我们需要为IRQ12这号中断编写一个中断处理程序，然后在这个中断处理程序中读取鼠标发来的数据包，下面，就让我们来看一看这段存在于pyos中的真实的代码：

```
// 鼠标 中断处理函数

void mouse_handle_for_mouse_interrupt()

{

  static int x_position = 444 ;                // 定义 鼠标 的初始 x 坐标

  static int y_position = 300 ;                // 定义 鼠标 的初始 y 坐标

  static int count = 0 ;                       // 此变量用来记录这是第几个数据包了

                                               // 因为 鼠标 发来的每个数据包

                                               // 都会引起一个中断
```

```
static int x_sign = 0 ;                    // 用来表示 x 位移量的符号

static int y_sign = 0 ;                    // 用来表示 y 位移量的符号

static struct message_message_struct message ;  // 定义一个消息

                                           // 这个消息用来通知操作系统内核有鼠标动作

char ch = io_read_from_io_port( 0x60 ) ;       // 通过 0x60 端口，获得 鼠标 发来的数据包

switch( ++count ){                         // 检测是第几个数据包

  case 1 :

    // 收到的是第一字节

    // 检测按键信息

    message.dose_left_button_down = ch & 0x1 ;

    message.dose_right_button_down = ch & 0x2 ;

    // 获得 x 及 y 位移量的符号

    x_sign = ch & 0x10 ? 0xffffff00 : 0 ;

    y_sign = ch & 0x20 ? 0xffffff00 : 0 ;

    break ;

  case 2 :
```

```
        // 收到的是第二字节，即 x 的位移量
        x_position += ( x_sign | ch ) ;              // 计算新的 x 坐标
        break ;
    case 3 :
        // 收到的是第三字节，即 y 的位移量
        y_position += -( y_sign | ch ) ;             // 计算新的 y 坐标
        // 把消息发送给内核
        message.message_type = MESSAGE_MOUSE_MESSAGE ;  // 设定消息的类型
        message.x_position = x_position ;            // 设定消息的参数（x 的坐标）
        message.y_position = y_position ;            // 设定消息的参数（y 的坐标）
        message_put_message( &kernel_message_queue , message ) ; // 把消息放进内核的消息队列中
        count = 0 ;                                  // 三个数据包已接收完毕，故清零计数器
        break ;
    }
}
```

(QQ:834449529)

回复 举报

smartqiu

二星会员

发帖	259
金钱	337
威望	333
贡献值	2
社区警告	0
诚信值	0
账号服务	0
热心值	0

加关注 发消息

2楼 发表于: 2007-06-18 只看该作者

i8042 :键盘控制器

以上的讨论都是针对硬件,其实,如果写一个底层的键盘相关的软件for PC,是不该直接和键盘通信的. 主板上一般都有键盘控制器,它在键盘和外设总线间是一个接口. 这个控制器处理signal-level的东东和协议的细节 ,同时提供转换,解释,处理扫描码,执行命令的功能.
PC 的键盘一般使用Intel 8042/兼容 的微控制器.现代计算机上,这个功能一般集成到南桥 . 然而,这个设备逻辑上仍然叫做 "the 8042".基于主板的不同,键盘控制器可以工作于:"AT-兼容" 模式, 或者 "PS/2-兼容" 模式. 如果主板支持 PS/2 鼠标就会使用后者. 这时, 8042 既是键盘控制器又是鼠标控制器. 键盘控制器根据硬连线的方式自动决定工作于哪种模式.

8042 包含如下寄存器:

1字节的输入缓存 - 包含从键盘来的数据,只读
1字节的输出缓存 - 包含,要被写到键盘的数据;只写
1字节的状态寄存器 - 8 个状态位; 只读
1字节控制寄存器 - 7 个控制位; 读写?

前三个寄存器可以被cpu通过io端口0x60 and 0x64直接访问.最后一个必须使用"Read Command Byte" 命令读, 使用"Write Command Byte" 命令写.(见关于键盘的其他帖)

8042的端口在cpu的io空间地址如下:

port----Read/Write-----Function
0x60----Read----------Read Input Buffer
0x60----Write---------Write Output Buffer
0x64----Read----------Read Status Register
0x64----Write---------Send Command

写端口 0x64 不会写任何指定的寄存器, 只是给8042一个命令. 如果命令有参数,参数就送到端口0x60. 命令的返回值也从端口 0x60去读.
("input buffer" : input from the keyboard, "output buffer": output to be sent to the keyboard.)

键盘复位
加电时,(或者"Reset" 命令) , 键盘进行 BAT (Basic Assurance Test)并装载下面的缺省值:

Typematic delay 500 ms.
Typematic rate 10.9 cps.
*选择扫描码集 set 2.
*把所有键设置为 typematic/make/break 统统使能.

* 一些键盘可以改变,一些不能.

进入 BAT后, 三个 LED 点亮,BAT 完成之后熄灭.同时,BAT 代码返回给host: 0xAA (BAT successful) ,

0xFC (Error). 许多键盘在BAT时忽略CLOCK 和 DATA 线, 直到BAT完成.因此, "禁止条件" (CLOCK line low) 不能键盘阻止向host发送BAT code.

扫描码集

到目前为止一共有三套扫描码集(Scan Code Set)，ps/2 键盘默认使用第二套。不过可以设置 i8042，让 i8042 把得到的 Scan Code 翻译成 Scan Code Set 1 中的 Scan Code ，这样键盘驱动从 i8042 得到的所有 Scan Code 都是第一套中的 S

can Code（实际中驱动也是这么做的）。所以我们只讨论 Scan Code Set 1 。需要说明的是 Scan Code 和 ASCII码完全不相同。

在 Scan Code Set 1 中，大多数键的 Make Code，Break Code 都是一个字节。他们的 Make Code 的最高位都为0，也就是他们的 Make Code 都小于 0x7F。而他们的 Break Code 为其 Make Code 或运算 80h ，也就是把 Make Code 的低7位不变，最高位设置为1。

还有一些扩展按键，他们的 Scan Code 是双字节的。他们的第一个字节都是E0h，表明这是一个扩展键。第2个字节，和单字节 Scan Code 的情况相同。

还有一个特殊的键，Pause/Break 键，它的 Make Code 为 E1,1D,45 E1,9D,C5，注意是 E1h 开头。而且它没有 Break Code 。

KEY	MAKE	BREAK	KEY	MAKE	BREAK	KEY	MAKE	BREAK
ESC	01	81	A	1E	9E	KP EN	E0,1C	E0,9C
1	02	82	S	1F	9F	R_CTRL	E0,1D	E0,9D
2	03	83	D	20	A0			
3	04	84	F	21	A1	KP /	E0,35	E0,B5
4	05	85	G	22	A2	R_ALT	E0,38	E0,B8
5	06	86	H	23	A3			
6	07	87	J	24	A4	HOME	E0,47	E0,C7
7	08	88	K	25	A5	UP ARROW	E0,48	E0,C8
8	09	89	L	26	A6	PG UP	E0,49	E0,C9
9	0A	8A	;	27	A7			
0	0B	8B	'	28	A8	L ARROW	E0,4B	E0,CB
-	0C	8C	`	29	89	R ARROW	E0,4D	E0,CD
=	0D	8D	\	2B	AB			
BKSP	0E	8E	Z	2C	AC	END	E0,4F	E0,CF
L_CTRL	1D	9D	X	2D	AD	D ARROW	E0,50	E0,D0

L_SHFT 2A AA C 2E AE PG DN E0,51 E0,D1
R_SHFT 36 B6 V 2F AF INSERT E0,52 E0,D2
KP * 37 B7 B 30 B0 DELETE E0,53 E0,D3
L_ALT 38 B8 N 31 B1
SPACE 39 B9 M 32 B2 L GUI E0,5B E0,DB
CAPS 3A BA , 33 B3 R GUI E0,5C E0,DC
F1 3B BB . 34 B4 APPS E0,5D E0,DD
F2 3C BC / 35 B5
F3 3D BD KP 7 47 C7
F4 3E BE KP 8 48 C8
F5 3F BF KP 9 49 C9
F6 40 C0 KP - 4A CA
F7 41 C1
F8 42 C2 KP 4 4B CB
F9 43 C3 KP 5 4C CC
F10 44 C4 KP 6 4D CD
KP + 4E CE
NUM 45 C5
SCROLL 46 C6 KP 1 4F CF
F11 57 D7 KP 2 50 D0
F12 58 D8 KP 3 51 D1
KP 0 52 D2
KP . 53 D3

PRNT SCRN E0,2A, E0,37 E0,B7, E0,AA

PAUSE E1,1D,45 E1,9D,C5 -NONE

i8042 有 4 个 8 bits 的寄存器，他们是 Status Register（状态寄存器），Output Buffer（输出缓冲器），Input Buffer（输入缓冲器），Control Register（控制寄存器）。使用两个 IO 端口，60h 和 64h。

Status Register（状态寄存器）

状态寄存器是一个8位只读寄存器，任何时刻均可被cpu读取。其各位定义如下

Bit7: PARITY-EVEN(P_E): 从键盘获得的数据奇偶校验错误
Bit6: RCV-TMOUT(R_T): 接收超时，置1
Bit5: TRANS_TMOUT(T_T): 发送超时，置1
Bit4: KYBD_INH(K_I): 为1，键盘没有被禁止。为0，键盘被禁止。
Bit3: CMD_DATA(C_D): 为1，输入缓冲器中的内容为命令，为0，输入缓冲器中的内容为数据。
Bit2: SYS_FLAG(S_F): 系统标志，加电启动置0，自检通过后置1
Bit1: INPUT_BUF_FULL(I_B_F): 输入缓冲器满置1，i8042 取走后置0
BitO: OUT_BUF_FULL(O_B_F): 输出缓冲器满置1，CPU读取后置0

Output Buffer（输出缓冲器）

输出缓冲器是一个8位只读寄存器。驱动从这个寄存器中读取数据。这些数据包括，扫描码，发往 i8042 命令的响应，间接的发往 i8048 命令的响应。

Input Buffer（输入缓冲器）

输入缓冲器是一个8位只写寄存器。缓冲驱动发来的内容。这些内容包括，发往 i8042 的命令，通过 i8042 间接发往 i8048 的命令，以及作为命令参数的数据。

Control Register（控制寄存器）

也被称作 Controller Command Byte （控制器命令字节）。其各位定义如下

Bit7: 保留，应该为0
Bit6: 将第二套扫描码翻译为第一套

Bit5: 置1，禁止鼠标
Bit4: 置1，禁止键盘
Bit3: 置1，忽略状态寄存器中的 Bit4
Bit2: 设置状态寄存器中的 Bit2
Bit1: 置1，enable 鼠标中断
BitO: 置1，enable 键盘中断

2个端口 0x60,0x64

驱动中把 0x60 叫数据端口
驱动中把 0x64 叫命令端口

1.5 命令

驱动可以直接给 i8042 发命令，可以通过 i8042 间接给 i8048 发命令。命令这部分内容直接来自 %26lt; 参考资料 [1] %26gt;。

1.5.1 发给i8042的命令

驱动对键盘控制器发送命令是通过写端口64h实现的，共有12条命令，分别为

20h
准备读取8042芯片的Command Byte；其行为是将当前8042 Command Byte的内容放置于Output Register中，下一个从60H端口的读操作将会将其读取出来。

60h
准备写入8042芯片的Command Byte；下一个通过60h写入的字节将会被放入Command Byte。

A4h
测试一下键盘密码是否被设置；测试结果放置在Output Register，然后可以通过60h读取出来。测试结果可以有两种值：FAh=密码被设置；F1h=没有密码。

A5h
设置键盘密码。其结果被按照顺序通过60h端口一个一个被放置在Input Register中。密码的最后是一个空字节（内容为0）。

A6h
让密码生效。在发布这个命令之前，必须首先使用A5h命令设置密码。

AAh
自检。诊断结果放置在Output Register中，可以通过60h读取。55h=OK。

ADh
禁止键盘接口。Command Byte的bit-4被设置。当此命令被发布后，Keyboard将被禁止发送数据到Output Register。

AEh
打开键盘接口。Command Byte的bit-4被清除。当此命令被发布后，Keyboard将被允许发送数据到Output Register。

C0h
准备读取Input Port。Input Port的内容被放置于Output Register中，随后可以通过60h端口读取。

D0h
准备读取Outport端口。结果被放在Output Register中，随后通过60h端口读取出来。

D1h
准备写Output端口。随后通过60h端口写入的字节，会被放置在Output Port中。

D2h
准备写数据到Output Register中。随后通过60h写入到Input Register的字节会被放入到Output Register中，此功能被用来模拟来自于Keyboard发送的数据。如果中断被允许，则会触发一个中断。

1.5.2 发给8048的命令

共有10条命令，分别为

EDh
设置LED。Keyboard收到此命令后，一个LED设置会话开始。Keyboard首先回复一个ACK（FAh），然后等待从60h端口写入的LED设置字节，如果等到一个，则再次回复一个ACK，然后根据此字节设置LED。然后接着等待。。。直到等到一个非LED设置字节(高位被设置)，此时LED设置会话结束。

EEh
诊断Echo。此命令纯粹为了检测Keyboard是否正常，如果正常，当Keyboard收到此命令后，将会回复一个EEh字节。

F0h
选择Scan code set。Keyboard系统共可能有3个Scan code set。当Keyboard收到此命令后，将回复一个ACK，然后等待一个来自于60h端口的Scan code set代码。系统必须在此命令之后发送给Keyboard一个Scan code set代码。当Keyboard收到此代码后，将再次回复一个ACK，然后将Scan code set设置为收到的Scan code set代码所要求的。

F2h
读取Keyboard ID。由于8042芯片后不仅仅能够接Keyboard。此命令是为了读取8042后所接的设备ID。设备ID为2个字节，Keyboard ID为83ABh。当键盘收到此命令后，会首先回复一个ACK，然后，将2字节的Keyboard ID一个一个回复回去。

F3h
设置Typematic Rate/Delay。当Keyboard收到此命令后，将回复一个ACK。然后等待来自于60h的设置字节。一旦收到，将回复一个ACK，然后将Keyboard Rate/Delay设置为相应的值。

F4h
清理键盘的Output Buffer。一旦Keyboard收到此命令，将会将Output buffer清空，然后回复一个ACK。然后继续接受Keyboard的击键。

F5h
设置默认状态(w/Disable)。一旦Keyboard收到此命令，将会将Keyboard完全初始化成默认状态。之前所有对它的设置都将

失效——Output buffer被清空，Typematic Rate/Delay被设置成默认值。然后回复一个ACK，接着等待下一个命令。需要注意的是，这个命令被执行后，键盘的击键接受是禁止的。如果想让键盘接受击键输入，必须Enable Keyboard。

F6h
设置默认状态。和F5命令唯一不同的是，当此命令被执行之后，键盘的击键接收是允许的。

FEh
Resend。如果Keyboard收到此命令，则必须将刚才发送到8042 Output Register中的数据重新发送一遍。当系统检测到一个来自于Keyboard的错误之后，可以使用自命令让Keyboard重新发送刚才发送的字节。

FFh
Reset Keyboard。如果Keyboard收到此命令，则首先回复一个ACK，然后启动自身的Reset程序，并进行自身基本正确性检测（BAT-Basic Assurance Test）。等这一切结束之后，将返回给系统一个单字节的结束码（AAh=Success, FCh=Failed），并将键盘的Scan code set设置为2。

1.5.3 读到的数据

00h/FFh
当击键或释放键时检测到错误时，则在Output Bufer后放入此字节，如果Output Buffer已满，则会将Output Buffer的最后一个字节替代为此字节。使用Scan code set 1时使用00h，Scan code 2和Scan Code 3使用FFh。

AAh
BAT完成代码。如果键盘检测成功，则会将此字节发送到8042 Output Register中。

EEh
Echo响应。Keyboard使用EEh响应从60h发来的Echo请求。

F0h
在Scan code set 2和Scan code set 3中，被用作Break Code的前缀。

FAh
ACK。当Keyboard任何时候收到一个来自于60h端口的合法命令或合法数据之后，都回复一个FAh。

FCh
BAT失败代码。如果键盘检测失败，则会将此字节发送到8042 Output Register中。

FEh
Resend。当Keyboard任何时候收到一个来自于60h端口的非法命令或非法数据之后，或者数据的奇偶交验错误，都回复一个FEh，要求系统重新发送相关命令或数据。

83ABh
当键盘收到一个来自于60h的F2h命令之后，会依次回复83h，ABh。83AB是键盘的ID。

1.6.1 读取状态寄存器

读取状态寄存器的方法，对64h端口进行读操作。

1.6.2 读数据

需要读取的数据有，i8042从i8048得到的按键的扫描码，i8042命令的ACK，i8042从i8048得到的i8048命令的ACK，需要命令重发的RESEND，一些需要返回结果的命令得到的结果。

当有数据需要被驱动读走的时候，数据被放入输出缓冲器，同时将状态寄存器的bit0（OUTPUT_BUFFER_FULL）置1，引发键盘中断（键盘中断的IRQ为1）。由于键盘中断，引起由键盘驱动提供的键盘中断服务例程被执行。在键盘中断服务例程中，驱动会从i8042读走数据。一旦数据读取完成，状态寄存器的bit0会被清0。

读数据的方法，首先，读取状态寄存器，判断bit0，状态寄存器bit0为1，说明输出缓冲器中有数据。保证状态寄存器bit0为1，然后对60h端口进行读操作，读取数据。

这里我们要谈一点很有用的题外话，前面提到的 IRQ，是 Interrupt Request line，中断请求线，是一个硬件线，它和中断向量是不同的。中断向量是用来在中断描述符表(IDT)中查找中断服务例程的那个序号。键盘的 IRQ 是1，键盘中断服务例

程对应的中断向量可不是1。这点要弄清楚。

1.6.3 向i8042发命令

当命令被发往i8042的时候，命令被放入输入缓冲器，同时引起状态寄存器的 Bit1 置1，表示输入缓冲器满，同时引起状态寄存器的 Bit2 置1，表示写入输入缓冲器的是一个命令。

向i8042发命令的方法，首先，读取状态寄存器，判断bit1，状态寄存器bit1为0，说明输入缓冲器为空，可以写入。保证状态寄存器bit1为0，然后对64h端口进行写操作，写入命令。

1.6.4 间接向i8048发命令

向i8042发这些命令，i8042会转发i8048，命令被放入输入缓冲器，同时引起状态寄存器的Bit1 置1，表示输入缓冲器满，同时引起状态寄存器的 Bit2 置1，表示写入输入缓冲器的是一个命令。这里我们要注意，向i8048发命令，是通过写60h端口，而后面发命令的参数，也是写60h端口。i8042如何判断输入缓冲器中的内容是命令还是参数呢，我们在后面发命令的参数中一起讨论。

向i8048发命令的方法，首先，读取状态寄存器，判断bit1,状态寄存器bit1为0，说明输入缓冲器为空，可以写入。保证状态寄存器bit1为0，然后对60h端口进行写操作，写入命令。

1.6.5 发命令的参数

某些命令需要参数，我们在发送命令之后，发送它的参数，参数被放入输入缓冲器，同时引起状态寄存器的Bit1 置1，表示输入缓冲器满。这里我们要注意，向i8048发命令，是通过写60h端口，发命令的参数，也是写60h端口。i8042如何判断输入缓冲器中的内容是命令还是参数呢。i8042是这样判断的，如果当前状态寄存器的Bit3 为1，表示之前已经写入了一个命令，那么现在通过写60h端口放入输入缓冲器中的内容，就被当做之前命令的参数，并且引起状态寄存器的 Bit3 置0。如果当前状态寄存器的 Bit3 为0，表示之前没有写入命令，那么现在通过写60h端口放入输入缓冲器中的内容，就被当做一个间接发往i8048的命令，并且引起状态寄存器的 Bit3 置1。

向i8048发参数的方法，首先，读取状态寄存器，判断bit1,状态寄存器bit1为0，说明输入缓冲器为空，可以写入。保证状态寄存器bit1为0，然后对60h端口进行写操作，写入参数。

站点帮助：验证问题答案举例，如验证问题：3+5=？，请输入中文答案：八

(QQ:834449529)

回复 举报

小元

二星会员

发帖	102
金钱	1165
威望	127
贡献值	0
社区警告	0
诚信值	0
账号服务	0
热心值	0

加关注 发消息

3楼 发表于: 2007-06-18 只看该作者

有点晕了.....

回复 举报

ujff77

4楼 发表于: 2007-06-19 只看该作者

一星会员

发帖	89
金钱	970
威望	99
贡献值	0
社区警告	0
诚信值	0
账号服务	0
热心值	0

加关注 发消息

的确不错，我也正在考虑用汇编写按键算了，还是这个管用，反正挂机时我都不上电脑的

回复 举报

小元

二星会员

发帖	102
金钱	1165

5楼 发表于: 2007-06-26 只看该作者

威望 127
贡献值 0
社区警告 0
诚信值 0
账号服务 0
热心值 0

加关注 发消息

回复 举报

caridle

二星会员

发帖 246
金钱 1423
威望 486
贡献值 0
社区警告 0
诚信值 0
账号服务 0
热心值 0

加关注 发消息

6楼 发表于: 2007-06-27 只看该作者

这个，还用操作硬件的模式太原始了哦，也麻烦

回复 举报

天涯乱子

7楼 发表于: 2007-06-27 只看该作者

二星会员

发帖 234
金钱 162
威望 307
贡献值 1
社区警告 0
诚信值 0
账号服务 0
热心值 0

加关注 发消息

太深奥了看不懂

回复 举报

jokersky

8楼 发表于: 2010-10-23 只看该作者

三星会员

好像不太王道

发帖 355
金钱 1008
威望 464
贡献值 20
社区警告 0
诚信值 0
账号服务 0
热心值 0

加关注 发消息

回复 举报

返回列表

回复 发帖