Proceedings of the 24th IASTED International Multi-Conference
PARALLEL AND DISTRIBUTED COMPUTING AND NETWORKS
February 14-16, 2006, Innsbruck, Austria

DESIGN OF IMPLEMENTATION OF A COMPATIBLE KEYBOARD CONTROLLER FOR KEYBOARDS AND MICE

Ying-Wen Bai and Hsiu-Chen Chen
Department of Electronic Engineering, Fu Jen Catholic University
Taipei, Taiwan, 242, R.O.C.,
bai@ee.fju.edu.tw

ABSTRACT

By integration of the basic functions of both a standard keyboard controller and the key matrix of a scan code, we provide a compatible design for both keyboards and mice. The key matrix of a scan code consists of 16 output pins by 8 input pins, which can support up to 128 keys. Usually, we can build a matrix table based on the mapping relationship between the column and the row of a ROM/RAM. If we store the mapping relationship in the Table of an RAM chip, then we can re-download the table content at run time by means of the commands of either BIOS or the various applications. Hence, we can gain more flexibility and compatibility by means of firmware setting. We can also use the multiple sets of the matrix table pre-stored in a ROM chip. The multiple sets of the matrix table are selected by the command from BIOS. Although this selection method may require more ROM space, 128KB ROM can support a sufficient number of sets of the matrix table to thus provide a feasible solution for a design which is compatible for a keyboard controller.

KEY WORDS
Keyboard Controller (KBC), Firmware, Keyboard (KBD), Mouse

1. Introduction

The keyboard controller (KBC) is one of I/O devices between a keyboard and a PC. The KBC was implemented by 8255 in an early design of a PC/XT. At that time the mouse was connected through a RS232 interface by using a COM Port via IRQ3/IRQ4. In 1987 IBM proposed the Personal System/2 hardware interface, by which a PS/2 keyboard (KBD) and a PS/2 mouse both shared Port "64h" and Port "60h" I/O channel with the 8042 KBC, therefore, increasing the IRQ1 for the KBD scan code input and added IRQ12 for mouse data packet input. Table I shows a KBC comparison between the AT compatible mode and the PS/2 compatible mode [1-4].

TABLE I
KBC COMPARISON BETWEEN THE AT COMPATIBLE MODE AND THE PS/2 COMPATIBLE MODE [1-4].

Port	Name	AT-compatible mode	PS/2-compatible mode
Port	P10	Undefined	Keyboard Data
	P11	Undefined	Mouse Data
	P12	Undefined	Undefined
	P13	Undefined	Undefined
	P14	External RAM (GA20)	External RAM (GA20)
	P15	Manufacturer's Setting	Manufacturer's Setting
	P16	Display Type Switch	Display Type Switch
	P17	Keyboard Inhibit Switch	Keyboard Inhibit Switch
Port 2 (Output Port)	P20	Reset CPU	Reset CPU
	P21	Gate A20	Gate A20
	P22	Undefined	Mouse Data
	P23	Undefined	Mouse Clock
	P24	Input Buffer Full (IBF)	Keyboard IBF interrupt
	P25	Output Buffer Empty	Mouse IBF interrupt
	P26	Keyboard Clock	Keyboard Clock
	P27	Keyboard Data	Keyboard Data
P3	T0	Keyboard Clock (Input)	Keyboard Clock (Input)
	T1	Keyboard Data (Input)	Mouse Clock (Input)

Fig. 1 shows the standard 8042 KBC architecture, in which the I/O address "64h" is the Command Port and "60h" is the Data Port. The PC system bus reads Port "64h" to get the KBC Status Register, and reads Port "60h" to obtain the KBD/mouse data.

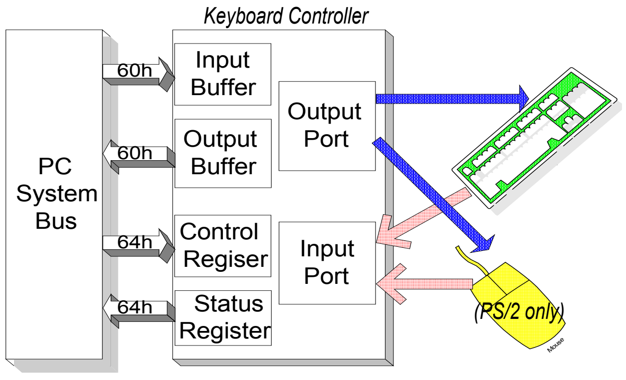


Fig. 1 Standard 8042 KBC architecture

Tables II and III show the type and feature list of the PS/2 KBD type and the PS/2 mouse respectively.

TABLE II
PS/2 KBD TYPE AND FEATURE LIST [1-4].

Keyboard Type	Feature
IBM PC/XT KBD (1981)	• 83 Keys • 5-pin DIN connector • Simple un-bidirectional serial protocol • Uses what we now refer to as code set 1 • No host-to-keyboard commands

IBM AT KBD (1984) – Not backward compatible with XT systems.	• 84~101 Keys • 5-pin DIN connector • Bi-directional serial protocol • Uses what we now refer to as code set 2 • 8 host-to-keyboard commands
IBM PS/2 KBD (1978) – compatible with AT system.	• 84 ~ 101 Keys • 6-pin mini-DIN connector • Bi-directional serial protocol • Offers optional scan code set 3 • 17 host-to-keyboard commands
Modern AT-PS/2 compatible KBD	• Any number of keys (usually 101 or 104 Keys) • 5-pin/6-pin DIN connector (adaptor usually included) • Bi-directional serial protocol • Only scan code set 2 guaranteed • Acknowledges all commands; may not act on all of them

TABLE III
PS/2 MOUSE TYPE AND FEATURE LIST [1-4].

Mouse Type	Features
Standard PS/2 Mouse	• 3-byte data packet • ID = 00h • Doesn't need any command sequence to initial
Microsoft Intelligent Mouse (Scrolling wheel + 3 buttons)	• 4-byte data packet • ID = 03h • Command sequence: Set sample rate 200 Set sample rate 100 Set sample rate 80
Microsoft Intelligent Mouse (Scrolling wheel + 5 buttons)	• 4-byte data packet • ID = 04h • Command sequence: Set sample rate 200 Set sample rate 200 Set sample rate 80

2. KBC, KBD and Mouse Compatible Design

In this paper, we focus on a compatible design for the integration of a KBC, a KBD and a mouse, such as an 8042 KBC compatible design, together with a PS/2 keyboard compatible design as shown in Fig. 4 including power management key support, Windows 2000 key and Internet key support.

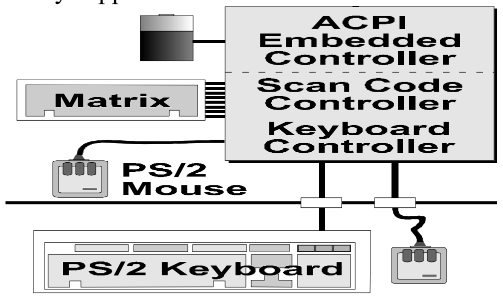


Fig. 4 KBC Architecture for portable computers

2.1 Dual PS/2 Keyboard and dual PS/2 Mouse compatible design

Usually, the KBC chip of the portable computer will support 3 channels of the PS/2 Port. The first one is for the touch pad, the second one is for the PS/2 KBD (external KBD) and the third one is for the PS/2 mouse (external mouse). There is a dual design for the PS/2 KBD (internal and external) and also dual PS/2 mouse (internal and external) supports. When either a dual PS/2 KBD or a dual PS/2 mouse coexists, the external PS/2 devices will have a higher priority.

The compatible design for the standard 8042 KBC includes the following basic functions:

1. Standard KBC command set supported
2. Gate "A20" for high memory supported.
3. Reset CPU supported.
4. System flag for cold boot/warm boot supported.
5. PS/2 keyboard and IRQ1 supported.
6. PS/2 mouse and IRQ12 supported.

In a portable computer, the key matrix emulates a PS/2 KBD, which provides an internal KBD including the following functions:

1. Standard keyboard command set support
2. Based on IBM 101/102 keyboard specification
3. Standard 84-key, 101-key, 104-key keyboard support
4. Both scan code set 1 and set 2 support
5. ACPI power management key support
   - Power key
   - Sleep key
   - Wake key
6. Windows 2000 key and Internet key support
   - Volume up key
   - Volume down key
   - Next track key
   - Previous track key
   - Stop key
   - Play/Pause key
   - Media select key
   - Email reader key
   - Turn on calculator key
   - Turn on my computer key
   - WWW keys
7. Function (Fn) key and function hotkey support

Because the "key matrix" is composed of a 16 (output pin) * 8 (input pin), it can support 128 keys. Usually, the designer will build a "matrix table" in a ROM/RAM chip according to the combination of the column and the row of the key matrix. If the matrix table is stored in a RAM chip, the system will use BIOS or application commands during the run time to re-download the matrix tables to thus accomplish the necessary compatibility for the different matrix tables. If the matrix tables are stored in a ROM chip, then the BIOS will select one of the matrix tables when the system is turned on. This implementation requires more ROM space. Usually, the ROM size is around 64KB/128KB, which includes most of matrix tables from many designs.

Table IV shows an example of a matrix table, by using scan code set 2, in which "BFh" represents none, "00h" represents beep, 01h~7F represents general scan codes, and above "80h" represents special keys such as, W2K keys and Internet keys.

TABLE IV
EXAMPLE OF A MATRIX TABLE, BY USING SCAN CODE SET 2

Offset	0Xh	1Xh	2Xh	3Xh	4Xh	5Xh	6Xh	7Xh
X0h	BF	BF	1C	1B	23	2B	3B	4B
X1h	42	4C	5D	69	72	7A	96	BF
X2h	BF	BF	BF	BF	BF	32	31	BF
X3h	BF	4A	29	C1	93	7B	91	BF
X4h	C7	85	1A	22	21	2A	3A	49
X5h	41	BF	F1	9A	94	7C	89	BF
X6h	BF	BF	16	1E	26	25	3D	44
X7h	43	4D	BF	6C	75	7D	79	BF
X8h	BF	16	1E	26	25	3D	46	BF
X9h	3E	45	09	CB	D1	8F	8E	BF
XAh	C6	BF	0E	D3	D4	2E	36	0A
XBh	55	4E	01	8D	8A	8C	8B	BF
XCh	BF	80	0D	58	04	2C	35	83
XDh	5B	54	66	6B	73	74	BF	BF
XEh	BF	BF	76	BF	0C	34	33	BF
XFh	0B	52	03	BF	70	71	C0	BF

Usually, for a notebook design, we can support the keyboard for multiple countries. When the BIOS executes the "post" procedure, the "COMMAND" will select the corresponding 128-byte for the specific country code and transfer it from ROM into the matrix table in the RAM. When the users press a specific matrix key, the corresponding row and column location will map into the matrix table in the RAM, and then the system will obtain the corresponding scan code and send the scan code to the computer through channel IRQ1.

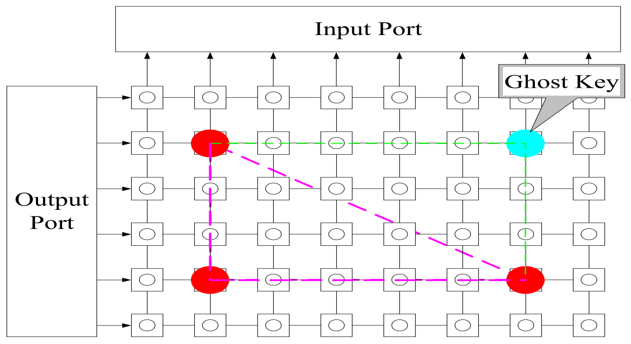


Fig. 5 An example of the generation of the "Ghost Key"

In addition, the compound keys which are pressed by three keys in a keyboard are often used. Due to the circuit characteristics, when the three keys are located at a perpendicular triangle, the fourth key at a corresponding angle can be a valid key due to the voltage drop. The fourth keys are called "ghost keys" which shall be avoided by dislocating the hardware location such as, "Alt", "Ctrl", "Shift" and "Fn". However, for the common characters, the dislocating methods only exchange the different character. For example, if the user presses the "A", "S and "Q" keys, the screen may display "A", "S", "Q", and "W4" characters. To avoid these ghost keys, we provide specific hardware and software solutions as follows.

(1) Hardware solutions for the avoiding "ghost key" are as shown in Fig. 6, by using an extra resistor or diode, or even by using a voltage comparator in the input terminal of the scan matrix to avoid any "ghost key" confusions. However, the hardware solution can increase the cost of the keyboard design.

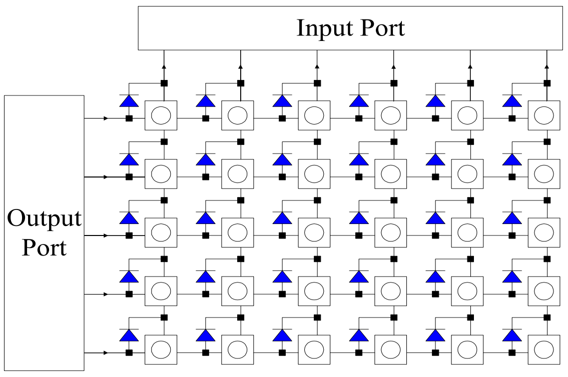
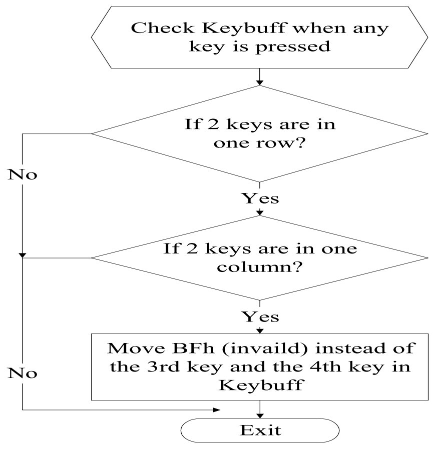


Fig. 6 A hardware solution for avoiding the ghost key

(2) A software solution for the avoiding "ghost key" is to check the combination of the row and column from the matrix scanning. If the three keys are in a perpendicular triangular, then the third key and the fourth key will be ignored.


Check Keybuff when any key is pressed
If 2 keys are in one row?
No
Yes
If 2 keys are in one column?
Yes
No
Move BFh (invaild) instead of the 3rd key and the 4th key in Keybuff
Exit
Fig. 7 A software solution for avoiding the ghost key

Usually, the KBC chip of a portable computer supports 3 channels of the PS/2 port. The first one supports the touch pad, or an internal mouse. The second and third support an external KBD and a second external mouse; however, they can support two KBDs or two mice simultaneously. If the users choose to use two KBDs or two mice simultaneously, then the KBC will be controlled by the

O.S. under a specific driver with a special supporting command set.

In addition, the external PS/2 ports also support to the Hot-Plug and the Hot-Swap function as shown in Figure 8 and Figure 9. With the system power on and the PS/2 device plugged in, the self-test success and the PS/2 device will automatically generate a BAT code to KBC. The KBD's BAT code can be AAh and the mouse's BAT code can be "AAH + 00h" for the mouse ID. When the KBC has received "AAh", the system notices that an external PS/2 device is plugged in and the KBC will send an "ECh" command to the external PS/2 device, and the external PS/2 device will respond "FAh" (ACK) to KBC. We learn the device is a mouse. However, if the response is "FEh", the plugged in device is a KBD. After the plugged device is classified, the KBC will send an "F1h" command once each second to test the validation for both the KBD and the mouse to the external PS/2 device. If the KBC can get a response (FEh) that means the PS/2 device is still connected. If there is no response from the PS/2 device, then the KBC will generate a timeout interrupt.

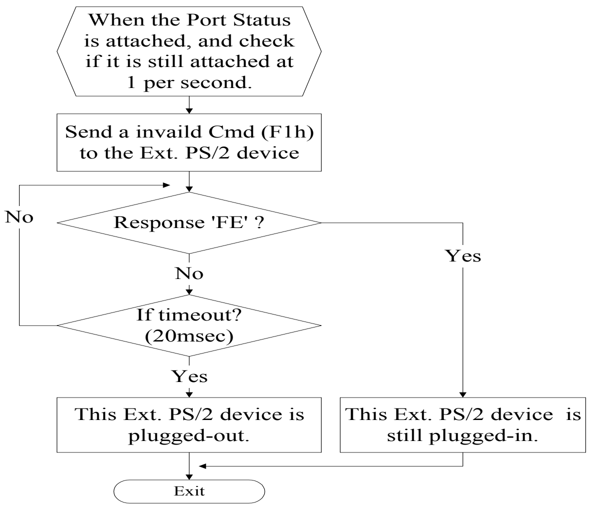

When the Port Status is attached, and check if it is still attached at 1 per second.
Send a invaild Cmd (F1h) to the Ext. PS/2 device
No
Response 'FE' ?
Yes
No
If timeout? (20msec)
Yes
This Ext. PS/2 device is plugged-out.
This Ext. PS/2 device is still plugged-in.
Exit

Fig. 8 Hot-plug detected flowchart

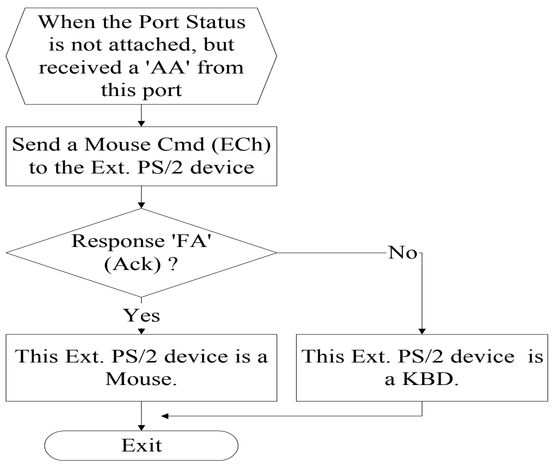

When the Port Status is not attached, but received a 'AA' from this port
Send a Mouse Cmd (ECh) to the Ext. PS/2 device
Response 'FA' (Ack) ?
No
Yes
This Ext. PS/2 device is a Mouse.
This Ext. PS/2 device is a KBD.
Exit

Fig. 9 A Hot-swap detected flowchart

2.2 A PS/2 Mouse compatible design

Currently, there are two major the mouse design types. The first one is the standard PS/2 mouse (3 Byte Data Packet; ID = 00) and the second one is the Intelligent 3D mouse (4 Byte Data Packet; ID = 03h or 04h). For our compatible design, we need to support the 3-byte mouse and the 4-byte mouse simultaneously. Tables V, VI and VII show a PS/2 mouse data packet, a 3D mouse with 3-button data packet respectively.

TABLE V
THE DATA PACKET OF A STANDARD PS/2 MOUSE

	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
Byte1	Y_O	X_O	Y_S	X_S	1	M_B	R_B	L_B
Byte2	X Movement							
Byte3	Y Movement							

Y_O: Y overflow bit
X_O: X overflow bit
Y_S: Y sign bit
X_S: X sign bit
M_B: Middle Button State
R_B: Right Button State
L_B: Left Button State

TABLE VI
THE DATA PACKET OF A 3D MOUSE WITH 3-BUTTON

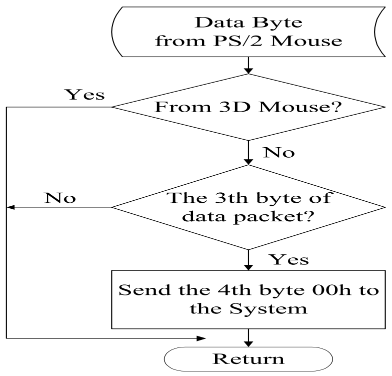
	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
Byte1	Y_O	X_O	Y_S	X_S	1	M_B	R_B	L_B
Byte2	X Movement							
Byte3	Y Movement							
Byte4	Z Movement							

TABLE VII
THE DATA PACKET OF A 3D MOUSE WITH 5-BUTTON

	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
Byte1	Y_O	X_O	Y_S	X_S	1	M_B	R_B	L_B
Byte2	X Movement							
Byte3	Y Movement							
Byte4	0	0	5_B	4_B	Z3	Z2	Z1	Z0

Z0 ~ Z3: Z Movement
4_B: The 4th Button State
5_B: The 5th Button State

To transform a 3-byte data packet into a 4-byte data packet as in Fig. 10, the KBC reads the data packet from the PS/2 mouse and tests to see if the 3D mouse exists. If the response is "yes", the KBC then checks to see if the data is from the 3D mouse. If the received data is not a 3-byte data packet, then the KBC will automatically send out a virtual byte (00h) to the system bus and form a 4-byte data format. Hence, the KBC can support both the PS/2 mouse and the Intelligent 3D mouse.


Data Byte from PS/2 Mouse
From 3D Mouse?
Yes
No
The 3th byte of data packet?
No
Yes
Send the 4th byte 00h to the System
Return
Fig. 10 The flowchart for the 3D mouse emulation

3. The Architecture of the KBC ROM Design

Usually, the flash ROM design of the KBC can be classified into three categories:
(1) On-chip flash ROM architecture as shown in Fig. 11
(2) Share BIOS flash ROM architecture as shown in Fig. 12
(3) Build-in BIOS ROM architecture as shown in Fig. 13

By using the design of the "Share BIOS flash ROM", the KBC image is combined with the BIOS image. Both the KBC image and the BIOS image share the same flash ROM, which, currently, uses 4MB/512KB. If the code size of the KBC occupies more space, then the code size of BIOS can share less space. Therefore, the assembly version (56 Kbytes) of KBC code can be lower in the cost in comparison with C language version of KBC code (96 Kbytes).

To reduce the development time of the KBC code, the vendor would normally like to use C programming language. However, the code size of the C version can cause the chip size to increase from 512 Kbytes to 1024 Kbytes, which increases the cost. To balance the cost and the development time, we design a compromise architecture that provides the standard code which includes the device polling program for the KBC/KBD/Mouse compatible programs which is occupy the code size 30KB by using Assembly programming language. This design provides a hook interface to the C programming language. This hook interface can provide the OEM vendors with a way to design their custom features by using C programming language.

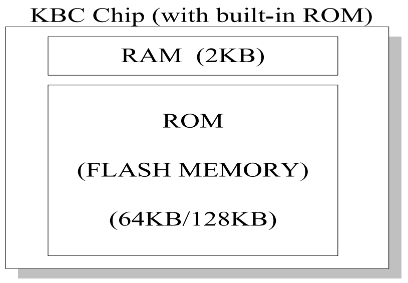


Fig. 11 On-chip flash ROM (single-chip) architecture

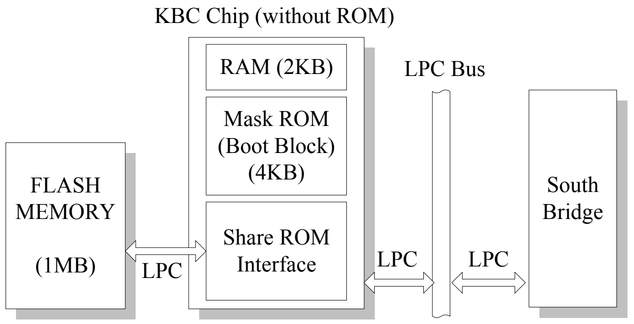


Fig. 12 Share-BIOS ROM architecture

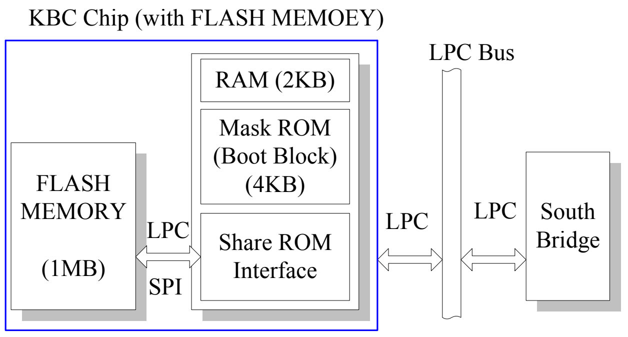


Fig. 13 Built-in BIOS ROM architecture

3.1 Support different flash ROM types

Usually, the flash ROM has two types: "boot-mode" and "on-board flash". When the flash ROM is empty or the flash fails, we must use the "boot-mode" and the external serial port to connect to another computer with can execute a flash utility, download the flash control program into RAM and execute it for erasing/programming flash ROM and to calculate the checksum if the downloading is correct or not. The other type is "on-board flash" by which the program in the flash ROM is complete and can execute. During the development or firmware update phase, we boot the system to a DOS mode; we use flash utility (or application) with a specific flash command to complete the flash ROM programming.

For the KBC "On-chip ROM architecture", the flash utility execute the command, by means of "on board flash" or through the serial port based on the boot mode flash. We download the flash control program to the specific RAM block; then the "program counter" will be set to a specific address and start the execution of the KBC programming.

For the KBC "Share-ROM architecture", when the system want to update flash ROM, the KBC must switch into "idle mode" until the completion of the updating or the KBC jumps to the loop in the RAM while updating and finishing the flash ROM, and use the specific command to wake up the KBC or exit the loop program.

3.2 The compatible design of the boot block

During the procedure of the flash ROM, the flash failure can stop the notebook booting procedure. Hence, we provide the boot block mechanism, which separates the KBC ROM into two parts and preserves 4Kbytes for the use of the boot block. When the KBC is turned on, the boot block will be executed the check checksum and the signature in main-program, if the KBC operation is correct, then either the system will jump to the main program, or else the system will stay in the looping of the boot block.

When the system is turned on, it will send the self-test command (AAh) to the KBC. If the KBC response is "55h", then the KBC is normal. If the system receives an "F1h" response from the KBC, then the main program of the KBC need to be re-flashing and the BIOS will execute the specific program to re-flash the KBC ROM. To provide the re-flash function, the BIOS ROM needs to pre-store the image of the KBC into the BIOS ROM. This image only include some of the functions of the boot block such as, power management, the checking functions and the response of the system commands of the main program which does not need too much space of the KBC ROM.

To avoid including the KBC image in the BIOS, we also propose another design, which expands the space of the boot block from 8Kbytes to 10 Kbytes in order to boot the DOS mode. Then, the system loads the KBC image from a disk or portable memory by using the flash utility in order to finish the re-flash function without increasing the BIOS load. This design may not support any PS/2 devices initially. However, the KBD need to response to the command from the system, otherwise, the system may recognize the KBD initial failure during the post procedure. If the system recognizes the KBD initial failure, it will disable IRQ1. When the system boots to the DOS mode and IRQ1is disabled, the USB KBD won't work.

In addition, the boot block prevents any flash failure during the design and development procedure and then saves one much time to maintain the system. Moreover, the end-users can re-download a new KBC image and flash utility through the Internet from the notebook vendors. Even the system has a flash failure; the new KBC image can be downloaded used to turn on to the system by the firmware updating without sending the notebook back to the vendor for this service.

4. Conclusion

To obtain our compatible design, we need to provide both the basic functions of the KBC and a RAM/ROM built in table for the keyboard type transformation from the input of the key matrix. If we implement the transformation relation in a RAM chip, we need to design a command from either BIOS or the applications to re-download the mapping table. If we implement the transformation relation in a ROM chip, the system requires a larger amount of ROM for the multiple selections for the different keyboard types. In addition, to increase the compatibility of the different mouse data standards, we also design the transformation from a 3-byte data packet to a 4-byte data packet, so that the KBC can support both the PS/2 mouse and the Intelligent 3D mouse.

Although the USB keyboard and mouse have been designed by polling USB controllers, before the operating system has booted or while the operating system is under DOS mode, as the USB driver hasn't started, the USB keyboard and mouse, therefore, can't work. Usually, this problem can be solved by using either the "64h" or "60h" I/O Port emulation.

References

[1] Adam Chapweske, The PS/2 Mouse/Keyboard Protocol, http://www.Computer-Engineering.org, May. 2003.

[2] Adam Chapweske, The AT-PS/2 Keyboard Interface, http://www.Computer-Engineering.org, 2001.

[3] Adam Chapweske, The PS/2 Mouse Interface, http://www.Computer-Engineering.org, 2001.

[4] R. A. Dayan, J. D. Rutledge, PS/2 mouse architecture type 1 and 2 (IBM Corporation, Jun. 1996).

[5] Hans-Peter Messmer, *The indispensable PC hardware book: your hardware questions answered* (Addison-Wesley, second edition, 1995).

[6] Intel Corporation, *LaGrande Technology Trusted Mobile Keyboard Controller* (Rev. 0.95b, Nov. 2004).