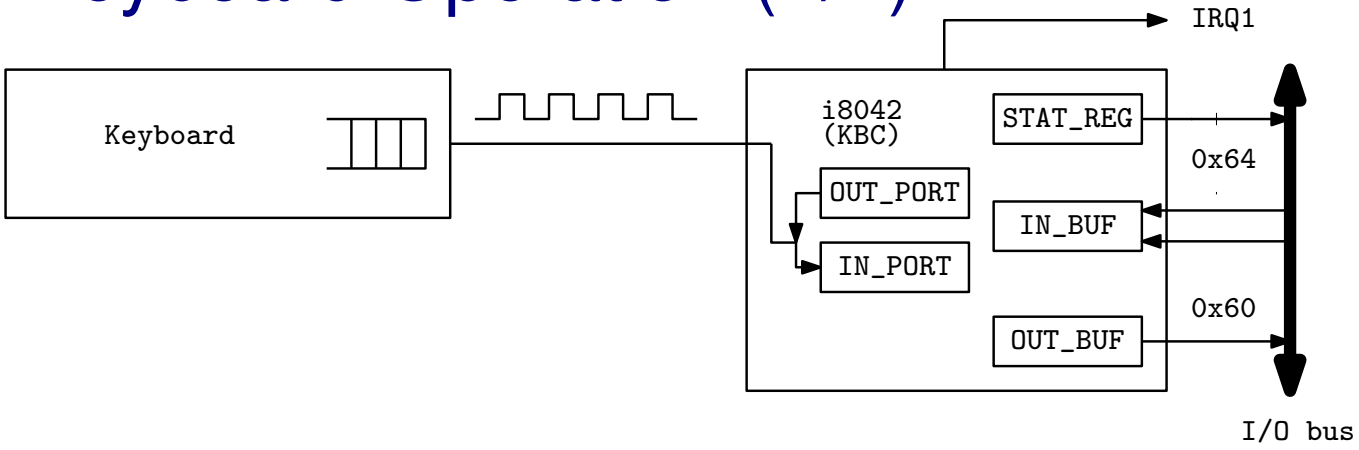
Computer Labs: The PC Keyboard

2º MIEIC

Pedro F. Souto (pfs@fe.up.pt)

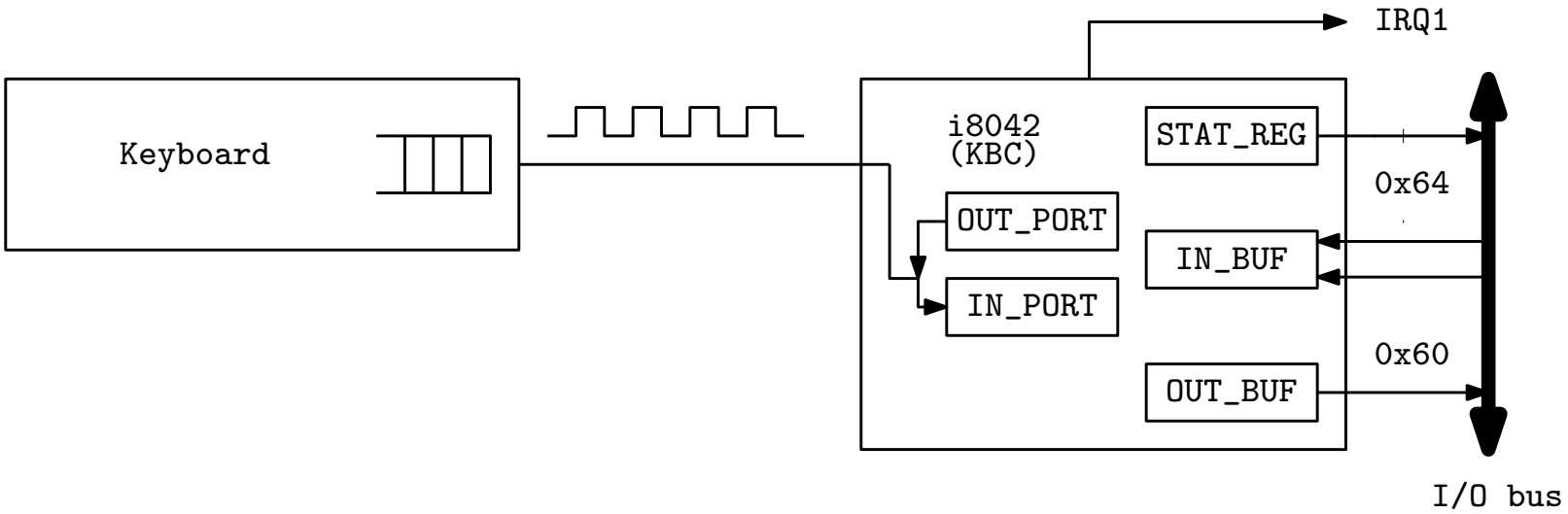
October 7, 2011

PC Keyboard Operation (1/2)



- The keyboard has its own controller chip (not shown): the controller@KBD (C@KBD)
- When a key is pressed the C@KBD generates a **scancode (make code)** and puts it in a buffer for sending to the PC
  - Usually, a scancode is one byte long
- The same happens when a key is released
  - Usually, the scancode when a key is released (**break code**) is the make code of that key with the MSB set to 1
- The communication between the C@KBD and the PC is via a serial line
  - I.e. the bits in a byte are sent one after the other over a pair of wires

PC Keyboard Operation (2/2)



- On the PC side this communication is managed by the keyboard controller (KBC)
  - In modern PCs, the KBC is integrated in the motherboard chipset
- When **`OUT_BUF`** is empty:
  1. The KBC signals that via the serial bus
  2. The C@KBD sends the byte at the head of its buffer to the KBC
  3. The KBC puts it in the `OUT_BUF`
  4. The KBC generates an interrupt by raising `IRQ1`

Keyboard Interrupt Handler (IH)

- Needs to read only the byte in the KBC's `OUT_BUF`
  - Communication between the keyboard and the KBC is rather slow
  - IHs should be as fast as possible
- ... and, of course, to output an `EOI` to the PIC
- Conversion from a scancode to a character code (ASCII or some other code) should not be done in the IH
  - IHs should be kept to a minimum
- Note that some scancodes may be more than 1 byte long
  - But again, this can be taken care of outside the IH
- It is possible to operate the KBC in polling mode, but it is not very convenient
  - Why?

Keyboard Commands (1/2)

- In the early PC models, interface with the keyboard used a very simple IC at port `0x60`
- For compatibility, the KBC provides two registers at that port:

  `IN_BUF` i.e. Input Buffer
  `OUT_BUF` i.e. Output Buffer
  and emulates the old interface:

  1. The KBC forwards bytes (commands) written in the `IN_BUF` to the C@KBD
  2. The C@KBD responds with one of 3 values: `0xFA` (ACK), `0xFE` (Resend) or `0xFC` (Error)
  3. The KBC puts the response in the `OUT_BUF` and raises IRQ1

Note The names of the registers `IN_BUF`/`OUT_BUF` are from the point of view of the KBC. The processor:

- Writes to the `IN_BUF`
- Read from the `OUT_BUF`

Keyboard Commands (2/2)

Command	Meaning	Args
`0xFF`	Reset KBD	
`0xF6`	Set default values and enable KBD	
`0xF5`	Disable KBD	
`0xF4`	Clear buffer and enable KBD	
`0xF3`	Change KBD repetition rate/delay	bits 0-4 rate bits 5-6 delay
`0xED`	Switch on/off KBD LEDs	bits 0-2

Note The arguments of commands that require them have to be written to the `IN_BUF` too, and are also acknowledged

- The C@KBD responds with one of 3 values as above.

Thus issuing such a command, requires 4 steps:

1. Write command to the `IN_BUF`
2. Read KBD response from the `OUT_BUF`
3. Write argument to the `IN_BUF`
4. Read KBD response from the `OUT_BUF`

In the case the KBD response is:

Resend (`0xFE`) the last byte should be written again

Error (`0xFC`) the entire sequence should be restarted

Command `0xF3` (Configure Typematic Parameters)

- Is an operating mode in which the keyboard generates a stream of scancodes when the user holds a key down
- The KBD allows to configure this operation via:

  Delay Which specifies the delay for entering typematic mode, counted from the moment the user presses down the key;

  Rate Which specifies the rate at which scancodes are generated, once the keyboard switches to typematic mode.

Command `0xED` (Set KBD LEDs)

Bit 2 Caps Lock indicator

Bit 1 Numeric Lock indicator

Bit 0 Scroll lock indicator

- There is no way to read the value of these LEDs
  - The code that changes them should remember their state

The KBC Commands (of the PC-AT)

- The KBC added a few commands, the **KBC commands**, and two new registers at port `0x64`

  `STAT_REG`: for reading the KBC state
  Not named for writing KBC commands
  - Apparently, this is not different from the `IN_BUF` at port `0x60`
  - The value of input line `A2` is used by the KBC to distinguish KBC commands from KBD commands
  - That is: the KBC has **only one** writable register, the `IN_BUF`

STAT_REG

- Input from/output to KBC requires reading the STAT_REG

Bit	Name	Meaning (if set)
7	Parity	Parity error - invalid data
6	Timeout	Timeout error - invalid data
5	Aux	Mouse data
4	INH	Inhibit flag: 0 if keyboard is inhibited
3	A2	A2 input line: 0 data byte 1 command byte
2	SYS	System flag: 0 if system in power-on reset, 1 if system already initialized
1	IBF	Input buffer full don't write commands or arguments
0	OBF	Output buffer full - data available for reading

- Bits 7 and 6 signal an error in the serial communication line between the keyboard and the KBC
- Do not write to the INPUT_BUF, if bit 1, i.e. the IBF, is set.

Keyboard-Related KBC Commands for PC-AT/PS2

- These commands must be written using address `0x64`
  - Arguments, if any, must be passed using address `0x60`
  - Return values, if any, are passed in the `OUT_BUF`

Command	Meaning	Args (A)/ Return (R)
`0x20`	Read Command Byte	Returns Command Byte
`0x60`	Write Command Byte	
`0xAA`	Check KBC (Self-test)	Returns 0x55, if OK Returns 0xFC, if error
`0xAB`	Check Keyboard Interface	Returns 0, if OK
`0xAD`	Disable KBD Interface	Inhibits KBD from sending data
`0xAE`	Enable KBD Interface	

- There are several others related to the mouse

(KBC “Command Byte”)

7	6	5	4	3	2	1	0
–	–	DIS2	DIS	–	–	INT2	INT

`DIS2` 1: disable mouse

`DIS` 1: disable keyboard

`INT2` 1: enable interrupt on OBF, from mouse;

`INT` 1: enable interrupt on OBF, from keyboard

– : Either not used or not relevant

Read Use KBC command 0x20, which must be written to `0x64`

Write Use KBC command 0x60, which must be written to `0x64`

Keyboard Programming/Configuration

`STAT_REG`: @ address `0x64`

- Read the KBC state

`IN_BUF`: Can be used to **write**:

Commands to the KBC access via address `0x64`;
Commands to the keyboard access via address `0x60`
Arguments of either commands access via address `0x60`

`OUT_BUF`: Can be used to **read**:

Scandcodes both make and break, received from the keyboard;
Return values from KBC commands;
Return values from keyboard commands;
Confirmation protocol messages `ACK`, `Resend` `Error`

Note These addresses belong to the I/O address space

- Need to use `IN/OUT` assembly instructions or the library functions `sys_inb()/sys_outb()` of the kernel API

Issuing a Command to the KBC

```
#define STAT_REG    0x64
#define KBC_CMD_REG 0x64

    while( 1 ) {
        sys_inb(STAT_REG, &stat); /* assuming it returns OK */
        /* loop while 8042 input buffer is not empty */
        if( (stat & IBF) == 0 ) {
            sys_outb(KBC_CMD_REG, cmd); /* no args command */
            return 0;
        }
        delay(WAIT_KBC);
    }
```

Note 1 Cannot output to the `KBC_CMD_REG` while the input buffer is full

Note 2 Code leaves the loop only when it succeeds to output the data to the `KBC_CMD_REG`

- To make your code resilient to failures in the KBC/keyboard, it should give up after “enough time” for the KBC to send a previous command/data to the KBD.

Reading Return Value/Data from the KBC

```
while( 1 ) {
    sys_inb(STAT_REG, &stat); /* assuming it returns OK */
    /* loop while 8042 output buffer is empty */
    if( stat & OBF ) {
        sys_inb(OUT_BUF, &data); /* assuming it returns OK

        if ( (stat &(PAR_ERR | TO_ERR)) == 0 )
            return data;
        else
            return -1;
    }
    delay(WAIT_KBC);
}
```

Note 1 Code leaves the loop only upon some input from the `OUT_BUF`.

- It is not robust against failures in the KBC/keyboard

Note 2 Must mask IRQ1, otherwise the keyboard IH may run before we are able to read the `OUT_BUF`

KBC Programming Issues

Interrupts If the command have responses, and interrupts are enabled, the IH will "steal" them away from other code

- The simplest approach is just to disable interrupts.

Timing KBD/KBC responses are not immediate.

- Code needs to wait for long enough, but not indefinitely

Concurrent Execution The C@KBD continuously scans the KBD and may send scancodes, while your code is writing commands to the KBC:

- How can you prevent accepting a scancode as a response to a command?
  - It is easier to solve this for KBC commands than for KBD commands.

Further Reading

- IBM's Functional Specification of the 8042 Keyboard Controller (IBM PC Technical Reference Manual)
- W83C42 Data Sheet, Data sheet of an 8042-compatible KBC
- Adam Chapweske's The AT-PS/2 Keyboard Interface
- Andries Brouwer's The AT keyboard controller, Ch. 11 of Keyboard scancodes
- Andries Brouwer's Keyboard commands, Ch. 12 of Keyboard scancodes
- Randal Hyde's The PC Keyboard, Ch. 20 of the Art of Assembly Language