Computer Labs: The PC Keyboard

2º MIEIC

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

October 9, 2017

Contents

Lab 3 Overview

PC Keyboard Operation: Data Input

Lab 3: The PC's Keyboard - Part 1

- Write functions:

  ```
  int kbd_test_scan(unsigned short assembly)
  int kbd_test_poll()
  ```

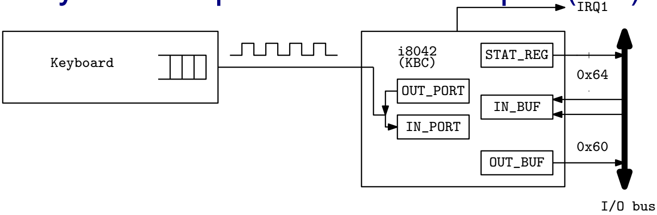
  that require programming the PC's keyboard controller
- These functions are not the kind of functions that you can reuse later in your project
  - The idea is that you design the lower level functions (with the final project in mind).
  - Reusable code should go on a different files from non-reusable code.
- What's new?
  - Program the KBC controller (i8042)
  - In part 2:
    - Mix C with assembly programming
    - Handle interrupts from more than one device

Contents

Lab 3 Overview

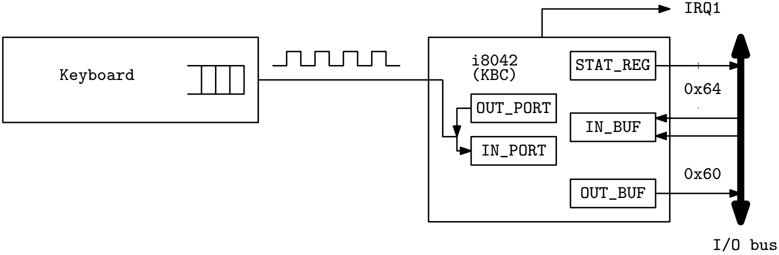
PC Keyboard Operation: Data Input

PC Keyboard Operation: Data Input (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: Data Input (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`

Lab 3: kbd_test_scan (1/2)

What Prints the scancodes, both the **makecode** and the **breakcode**, read from the KBC

- Should terminate when it reads the **breakcode** of the ESC key: 0x81
- The first byte of two byte scancodes is usualy 0xE0
  - This applies to both make and break codes

How Need to subscribe the KBC interrupts

- Upon an interrupt, read the scancode from the OUT_BUF

Note There is no need to configure the KBC

- It is already initialized by Minix

Issue Minix already has an IH installed

- Must be disabled to prevent it from reading the OUT_BUF before your handler does it

Solution Use not only the IRQ_REENABLE but also the IRQ_EXCLUSIVE policy in sys_irqsetpolicy(), i.e. use IRQ_REENABLE|IRQ_EXCLUSIVE

Lab 3: `kbd_test_scan` (2/2)

KBC interrupt subscription in exclusive mode;

`driver_receive()` loop (similar to that of lab 2)

Interrupt handler reads the bytes from the KBC's `OUT_BUF`

- Should read only **one byte per interrupt**
  - The communication between the keyboard and the KBC is too slow
- Later, you may think about including the code that maps the scancodes to a character code
  - IH in Minix are usually out of the critical path
    - They are executed with interrupts enabled and after issuing the EOI command to the PIC
  - In many systems this may not be appropriate. For example, in Linux most DD break interrupt handling in two:

    Top half which is in the critical path, and therefore does minimal processing

    Bottom half which is not in the critical path, and therefore may do additional processing
- Should not print the scancodes (not reusable)

Minix 3 Notes: `driver_receive()` is not Polling

`driver_receive()` is a blocking call. If the process's "IPC queue" is empty:

- The OS will move it to the WAIT state
- The state will be changed to READY, only when a message (or notification) is sent to the process

```
while( 1 ) { /* You may want to use a different condition
  /* Get a request message. */
  if ( driver_receive(ANY, &msg, &ipc_status) != 0 ) {
    printf("driver_receive failed with: %d", r);
    continue;
  }
  if (is_ipc_notify(ipc_status)) { /* received notificat
    switch (_ENDPOINT_P(msg.m_source)) {
    case HARDWARE: /* hardware interrupt notification
      if (msg.NOTIFY_ARG & irq_set) { /* subscribed
        ... /* process it */
      }
      break;
    default:
      break; /* no other notifications expected: do
    }
```

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
- Andries Brouwer's The AT keyboard controller, Ch. 11 of Keyboard scancodes
- Andries Brouwer's Keyboard commands, Ch. 12 of Keyboard scancodes