

Computer Labs: The i8254 Timer/Counter

2º MIEIC

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

October 7, 2014

Lab 2: The PC's Timer/Counter - Part I

- ▶ Write a set of functions:

```
int timer_test_config(unsigned long timer)
int timer_test_square(unsigned long rate)
```

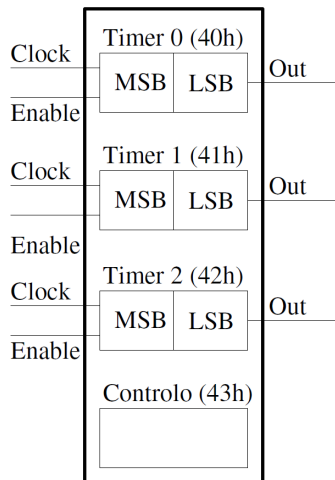
that require programming the PC's Timer/Counter

- ▶ These functions are at a higher level than those of the previous lab
 - ▶ The idea is that you design the lower level functions (with the final project in mind)
 - ▶ In this lab we have also defined the lower level functions
- ▶ What's new?
 - ▶ Program an I/O controller: the PC's timer counter (i8254)
 - ▶ Use interrupts (Part II)

The i8254

- ▶ It is a programmable timer/counter
 - ▶ Each PC has a functionally equivalent circuit, nowadays it is integrated in the so-called south-bridge
 - ▶ Allows to measure time in a precise way, independently of the processor speed
- ▶ It has 3 16-bit counters, each of which
 - ▶ May count either in binary or BCD
 - ▶ Has 6 counting modes

i8254 Block Diagram



- ▶ Three independent 16-bit counters
 - ▶ Ports 40h, 41h and 42h
 - ▶ MSB and LSB addressable separately
 - ▶ 6 counting modes
- ▶ An 8 bit-control register
 - ▶ Port 43h
 - ▶ Programming of each counter independently

i8254 Control Word

- ▶ Written to the Control Register (0x43)

Bit	Value	Function
7,6		Counter selection
	00	0
	01	1
	10	2
5,4		Counter Initialization
	01	LSB
	10	MSB
	11	LSB followed by MSB
3,2,1		Counting Mode
	000	0
	001	1
	x10	2
	x11	3
	100	4
	101	5
0		BCD
	0	Binary (16 bits)
	1	BCD (4 digits)

Example

- ▶ Timer 2 in mode 3
- ▶ Counting value: 1234 = 0x04D2

Control Register: 10111110

Timer2 LSB 0xD2

Timer2 MSB 0x04

i8254 Counting Modes

Mode 0 Interrupt on terminal count – for counting events

- ▶ `OUT` goes high and remains high when count reaches 0

Mode 1 Hardware retriggerable one-shot

- ▶ `OUT` goes low and remains low until count reaches 0, the counter is reloaded on a rising edge of the `ENABLE` input

Mode 2 Rate Generator (divide-by-N counter)

- ▶ `OUT` goes low for one clock cycle when count reaches 0, the counter is reloaded with its initial count afterwards, and ...

Mode 3 Square Wave Generator – for Lab 2

- ▶ Similar to mode 2, except for the duty-cycle: `OUT` will be high for half of the cycle and low for the remaining half of the cycle

i8254: Read-Back Command

- ▶ Allows to retrieve
 - ▶ the programmed configuration
 - ▶ and/or the current counting valueof one or more timers
- ▶ Written to the Control Register ($0x43$)
 - ▶ The configuration is read from the timer's data register
 - ▶ The 6 LSBs match that of the Control Word

Read-Back Command Format

Bit	Value	Function
7,6		Read-Back Command
	11	
5		$\overline{\text{COUNT}}$
	0	Read counter value
4		$\overline{\text{STATUS}}$
	0	Read programmed mode
3		Select Timer 2
	1	Yes
2		Select Timer 1
	1	Yes
0		Select Timer 0
	1	Yes
0		Reserved

Read-Back Status Format

Bit	Value	Function
7		Output
6		Null Count
5,4		Counter Initialization
3,2,1		Programmed Mode
0		BCD

i8254: Use in the PC (1/2)

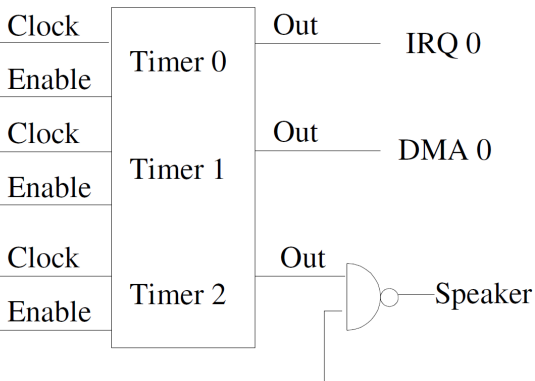
Frequência de entrada:

1 193 181 Hz

5 Volt

Bit 0 porto 61h

Bit 1 porto 61h



- ▶ Timer 0 is used to provide a time base.
- ▶ Timer 1 is used for DRAM refresh
 - ▶ Via DMA channel 0(Not sure this is still true.)
- ▶ Timer 2 is used for tone generation

i8254: Use in the PC (2/2)

- ▶ The i8254 is mapped in the I/O address space:

```
Timer 0:          0x40
Timer 1:          0x41
Timer 2:          0x42
Control Register: 0x43
```

- ▶ Need to use IN/OUT assembly instructions

- ▶ Minix 3 provides the `SYS_DEVIO` kernel call for doing I/O

```
#include <minix/syslib.h>
```

```
int sys_inb(port_t port, unsigned long *byte);
int sys_outb(port_t port, unsigned long byte);
```

- ▶ Need to write to the control register before accessing any of the timers

Minix 3 and Timer 0

- ▶ At start up, Minix 3 programs Timer 0 to generate a square wave with a fixed frequency
 - ▶ Timer 0 will generate an interrupt at a fixed rate:
 - ▶ Its output is connected to `IRQ0`
- ▶ Minix 3 uses these interrupts to measure time
 - ▶ The interrupt handler increments a global variable on every interrupt
 - ▶ The value of this variable increments at a fixed, known, rate
- ▶ Minix 3 uses this variable mainly for:
 - ▶ Keeping track of the date/time
 - ▶ Implementing SW timers

Lab 2: Part 1 - Reading Timer Configuration

What to do? Read Timer 0 configuration in Minix

```
int timer_test_config(unsigned long timer)
```

1. Write read-back command to read input timer configuration:
 - ▶ Make sure 2 MSBs are both 1
 - ▶ Read only the status
2. Read the timer port
3. Display the configuration in a user-friendly way

How to design it? Try to develop an API that can be used in the project.

```
int timer_get_config(unsigned long timer, unsigned char *st)  
int timer_show_config(unsigned long timer);
```

Lab 2: Part 1 - Generating a Square Wave

What to do? Change the rate at which Timer 0 generates interrupts.

```
int timer_test_square(unsigned long freq)
```

1. Write control word to configure Timer 0:
 - ▶ Select square wave mode (mode 3)
 - ▶ Preferably, LSB followed by MSB
2. Load Timer 0 with the value of the divisor to generate the frequency corresponding to the desired rate
 - ▶ Depends on the previous step

How to design it? Try to develop an API that can be used in the project.

```
int timer_set_square(unsigned char timer,  
                    unsigned long freq)
```

How do we know it works? Use the `date` command.

Further Reading

- ▶ [Lab 2 Handout](#)
- ▶ [i8254 Data-sheet](#)