Programmable Input Timer

May 9, 2008

Contents

0.1 Introduzione

Il PIT (Intel 8253 o 8254) è un componente presente nelle schede madri di tutte le architetture x86 che serve a regolare funzioni di timing. Dispone di tre canali: il canale 0 genera un IRQ0 ogni intervallo di tempo specificato dall'utente; il canale 1 è definito dal sistema; il canale 2 configura i beep. Nel nostro tutorial, parleremo del canale 0, che permette di svolgere molte funzioni utili come il task scheduling.

0.2 Command Register

Per settare le impostazioni del timer e attivarlo, avremo bisogno di scrivere dei dati su delle porte specifiche: il *Command Register* (0x43), e tre *Data register* 0x40, 0x41 e 0x42 collegati ai tre canali. Il Command Register permette di specificare alcune impostazioni, riunite in un singolo byte secondo lo schema qui sotto:

7	6-5	4-3	1	0
Chan.Num.	RW	Mode	BCD	

- *Chan.Num.* è il numero del canale che vogliamo attivare.
- *RW*: indica quale byte vogliamo inviare. 1=LSB (Least significant byte), 2=MSB (Most significant byte), 3=entrambi.

- *Mode*: di queste ce ne sono molte, trattate nel paragrafo seguente.
- *BCD*: è un valore di un solo bit che specifica se il byte che stiamo per mandare è di 16 bit (BCD=0) oppure se si conta secondo BCD (BCD=1).

In DreamOS ho attivato il canale 0, con RW=3, la modalità Square Wave e BCD=0.

0.2.1 Modalità

- Mode 0: Interrupt on terminal count. Il timer inizierà un count-down a partire dal numero passato nel registro dati, secondo la sua frequenza predefinita.
- Mode 1: Hardware triggered one-shot. Simile al precedente, ma il conteggio inizia dopo la ricezione del segnale GATE.
- Mode 2: Rate generator. Il normale timer.
- Mode 3: Square Wave. Simile al precedente, solo la frequenza degli impulsi è calcolata con una formula leggermente differente.
- Mode 4: Software triggered strobe. Genera un basso impulso ogni ciclo di clock. La modalità 5 è una variazione di questa.

0.3 Data register

Come ho precisato prima, i Data Register sono tre per ciascuno dei canali disponibili sul PIT. Dato che noi usiamo il canale 0, l'unico che ci interessa è quello con indirizzo 0x40. In realtà il numero che andremo a passare a questo canale è un "divisore": il PIT dividerà 1193180 Hz per il numero che gli passate per sapere quante volte al secondo generare un IRQ0.

La funzione *configure_PIT*, definita in *include/hardware/8253.h* e il cui codice è in *hardware/8253.c*, mostra bene nella pratica queste operazioni che ho solo descritto. Eccola qui:

```
void configure_PIT ()
{
int divisor = 65535;
asm("cli");
ticks = seconds = 0;
outportb(0x36, PIT_COMREG);
outportb(divisor & 0xFF, PIT_DATAREG0);
outportb (divisor >> 8, PIT_DATAREG0);
asm ("sti");
}
```

0.4 E adesso?

Adesso dobbiamo creare un IRQ handler, ovvero una funzione che venga richiamata tutte le volte che il timer genererà un IRQ0. In DreamOS, la funzione è associata all'IRQ0 tramite le due funzioni *enable_IRQ e add_IRQ_handler,* entrambe chiamate nel file *hardware/pic8259.c*. In questa funzione potete inserire tutte le operazioni che volete. Ecco qui un esempio.

Il nostro handler incrementa una variabile globale chiamata *ticks* per ogni chiamata; la variabile *seconds* è incrementata ogni 100 *ticks* per tenere traccia di quanti secondi sono passati. Ovviamente, se l'OS viene eseguito per molto tempo, le variabili rischiano di assumere valori esageratamente alti: per questo, *ticks* è azzerata ogni secondo mentre *seconds* ogni 86400 secondi, ovvero una giornata.

```
void PIT_handler ()
{
if (++ticks % 100 == 0) {
  ticks = 0;
  if (++seconds > 86400)
    seconds = 0;
  }
}
```

0.5 Implementazione della sleep()

Dopo aver implementato il PIT, abbiamo usato le due variabili *ticks* e *seconds* per creare una funzione simile alla *sleep()* della libreria standard.

```
unsigned int sleep (unsigned int secs)
{
int p = seconds + secs;
while (ticks != 0);
while (seconds < p);
return 0;
}
```

0.6 Conclusione

Spero che questa piccola guida vi sia di aiuto per programmare il vostro gestore personale. Per qualsiasi commento, correzione o consiglio potete scrivere a syn.shainer@gmail.com .

Lisa