Tutorial ATmega8 - #2:

Timer0 con interrupt da 1mS

Conoscenze richieste: Programmazione in linguaggio C, Tuorial ATmega8 - #1

Tempo richiesto: 1 - 2 ore circa

Linguaggio tecnico usato: Semplice ed amichevole

Autore: Emanuele Aimone

Contents

INTRODUZIONE

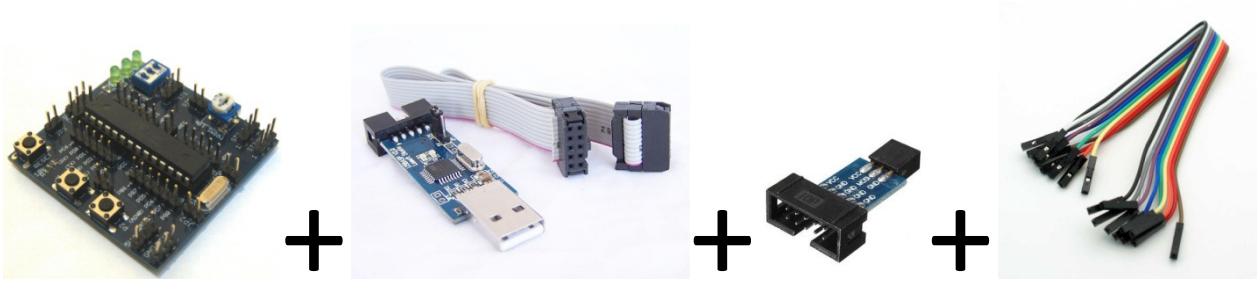
In questo tutorial vedremo come configurare il Timer0 dell'ATmega8 con interrupt con cadenza di 1mS.

GLI ELEMENTI USATI IN QUESTO TUTORIAL

Per il tutorial abbiamo bisogno di una semplice demoboard, il programmatore USBasp, il compilatore WinAVR 20030913, e il caricatore di firmaware (Firmware Uploader) Extreme Burner v1.4.3.

L'hardware

Ddemoboard ATmega8 + USBAsp + Adattatore USBAsp + Jumpers



IL FIRMWARE

Per verificare il corretto funzionamento del Timer0 il firmware farà lampeggiare un Led che deve essere collegato sul pin 23 (PC0).

Lampeggio di un Led con il Timer0

Vediamo subito l'esempio di programma che dobbiamo scrivere:

```
#include <avr/io.h>
#include <avr/interrupt.h>

#define DELAY_LED       500 //mS

//Variabili globali:
unsigned char decontLed;  //usato per cambiare lo stato al Led
                          //dopo il tempo definito con DELAY_LED
//Funzioni:
void timer0_init(void);  //inizializza i registri del Timer0

//Funzione eseguita quando si scatena l'interrupt del Timer0.
//L'interrupt viene generato quando TCNT0 va in overflows.
ISR(TIMER0_OVF_vect){  // timer0 overflow interrupt
        TCNT0 += 6;  //precarica il registro TCNT0 in modo di avere un
                     //interrupt  generato in 1mS

   if(decontLed>0){
                decontLed--;  //fino a quando non è a 0 (passati 500mS)
                              //decrementa la variabile
        }else{
                decontLed = DELAY_LED;          //la variabile viene ricaricata per
                                                //fornire un altra attesa di 500mS
                                                //prima di cambiare di stato il Led
                if((PORTC & 0x01) == 0x00){ //controlla lo stato di PC0
                        PORTC |= 0x01;    // Led ON
                }else{
                        PORTC &= 0xFE;    // Led OFF
                }
        }

}

//il nostro codice principale, da dove il microcontrollore parte:
int main(void){
```

```
   //configura PC0 (PIN23) come Output
   //e tutti gli altri PCx vengono configurati come Input
   DDRC = 0x01;

   cli(); //Disabilita gli interrupts globali

   timer0_init(); //inizializza i registri del Timer0

   sei(); //Abilita gli interrupts globali

   // Ciclo infinito
   while(1){
                    /* qui si può far girare altro codice che quindi
                       di tanto in tanto viene messo in pausa per eseguire
                       la funzione dell'interrupt del Timer0 */
   }
}

//inizializza i registri del Timer0
void timer0_init(void){
        //Vedere pagina 70 del datasheet per il diagramma a blocchi
        //del Timer0

        //Registro TIMSK: Pagina 72 del datasheet
        //Il bit chiamato TOIE0 viene impostato a 1 per abilitare
        //l'interrupt del Timer0
        TIMSK |= (1 << TOIE0);

        //Registro TCCR0: Pagina 71 del datasheet
        //imposta il clock con prescaler a 64
        //Qui per essere sicuri che l'interrupt abbia cadenza 1mS
        //dobbiamo calcolare in base alla frequenza dell'oscillatore
        //la divisione che imponiamo con il prescaler
        TCCR0 |= (1 << CS01) | (1 << CS00);

        //Registro TCNT0: Pagina 72 del datasheet
        //precarica il contatore usato dal Timer0
        TCNT0 = 6;
}
```

Rispetto al tutorial #1 il led viene fatto lampeggiare dentro l'interrupt.

Quando il microcontrollore si avvia inizializza i registri tra cui i registri contenuti nella funzione:

```
void timer0_init(void){
      ..........
}
```

Questa funzione imposta il Timer0 dell'ATmega8. Il Timer0 quando parte incomincia ad incrementare il registro chiamato "TCNT0", essendo TCNT0 un registro a 8bit può arrivare fino a 255, al successivo incremento portato dal Timer0 il nostro TCNT0 si porta a 0 e quindi riparte.

Il Timer0 quindi ogni tanto si sveglia e incrementa il registro TCNT0, ma con quale frequenza incrementa tale registro?

Tutto è determinato dall'impostazione dei registri e dall'oscillatore usato.

In questo caso l'oscillatore usato è esterno, con frequenza 16MHz (il quarzo montato sulla demoboard), questo viene selezionato con i fuses che in fase di caricamento del firmware andremo a vedere come impostare.

Se prendiamo il datasheet dell'ATmega8, vediamo a pagina 71 e 72 un registro chiamato "TCCR0" questo registro dice al Timer0 con quale frequenza incrementare il contatore TCNT0.

Il registro TCCR0 dispone di 3 bit:

Bit	7	6	5	4	3	2	1	0	
	–	–	–	–	–	CS02	CS01	CS00	TCCR0
Read/Write	R	R	R	R	R	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Questi 3 bit fanno da selettore, usando la tabella seguente possiamo quindi determinare la frequenza del Timer0:

Table 34. Clock Select Bit Description

CS02	CS01	CS00	Description
0	0	0	No clock source (Timer/Counter stopped)
0	0	1	$clk_{I/O}$/(No prescaling)
0	1	0	$clk_{I/O}$/8 (From prescaler)
0	1	1	$clk_{I/O}$/64 (From prescaler)
1	0	0	$clk_{I/O}$/256 (From prescaler)
1	0	1	$clk_{I/O}$/1024 (From prescaler)
1	1	0	External clock source on T0 pin. Clock on falling edge
1	1	1	External clock source on T0 pin. Clock on rising edge

Nel nostro caso abbiamo usato :

CS02	CS01	CS00	Description
0	1	1	$clk_{I/O}$/64 (From prescaler)

Quindi essendo:

clk_{io} = 16MHz

Il Timer0 avrà frequenza:

Frequenza_Timer0 = 16MHz / 64 = 0,25MHz = 250KHz

Abbiamo parlato di avere un timer con una cadenza di 1mS. Quindi, il contatore TCNT0 quando passa da 255 a 0 (va in overflow), fa scatenare un interrupt. Quando questo interrupt si scatena, stoppa quello che stava facendo nella funzione main:

```
int main(void){
      ..........
  while(1){
                  /* il codice scritto qui è quello che verrà messo in pausa*/
   }

}
```

E quindi esegue il codice scritto nella funzione ISR(TIMER0_OVF_vect):

```
ISR(TIMER0_OVF_vect){
  ..........
}
```

ATTENZIONE: Il codice scritto nella funzione ISR(TIMER0_OVF_vect) non deve essere molto espanso, altrimenti non finisce di eseguire il codice che già arriva un altro interrupt e riparte dall'inizio di ISR(TIMER0_OVF_vect) e quindi si rischia di non tornare più ad eseguire il codice nella funzione main.

Tornando alla cadenza del Timer di 1mS, se la funzione ISR(TIMER0_OVF_vect) viene esguita quando TCNT0 va in overflow allora se parte da 0 e arriva a 255 la nostra funzione ISR(TIMER0_OVF_vect) verrà eseguita ogni:

$$\frac{1}{\text{Frequenza\_Timer0}\ /256} = \frac{1}{250\text{KHz}\ /256} = 1{,}024\ mS$$

Non è proprio 1mS, quindi se noi invece di lasciare il registro TCNT0 essere incrementato di 256 volte lo aiutiamo incrementandolo di 6 con l'istruzione:

```
TCNT0 += 6;
```

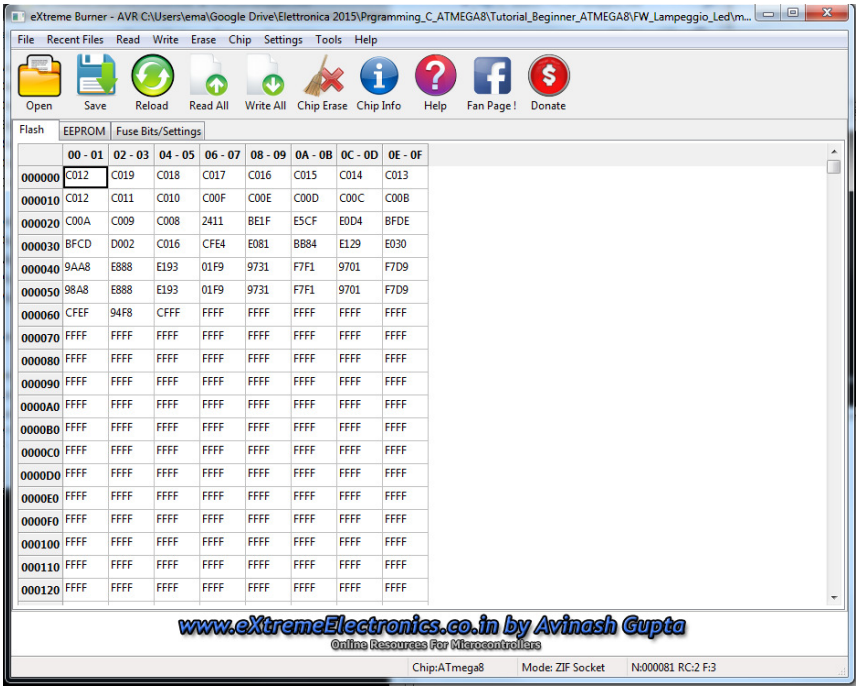
$$\frac{1}{250\text{KHz}\ /(256-6)} = 1\ mS$$

Avremo quindi 1mS esatto!

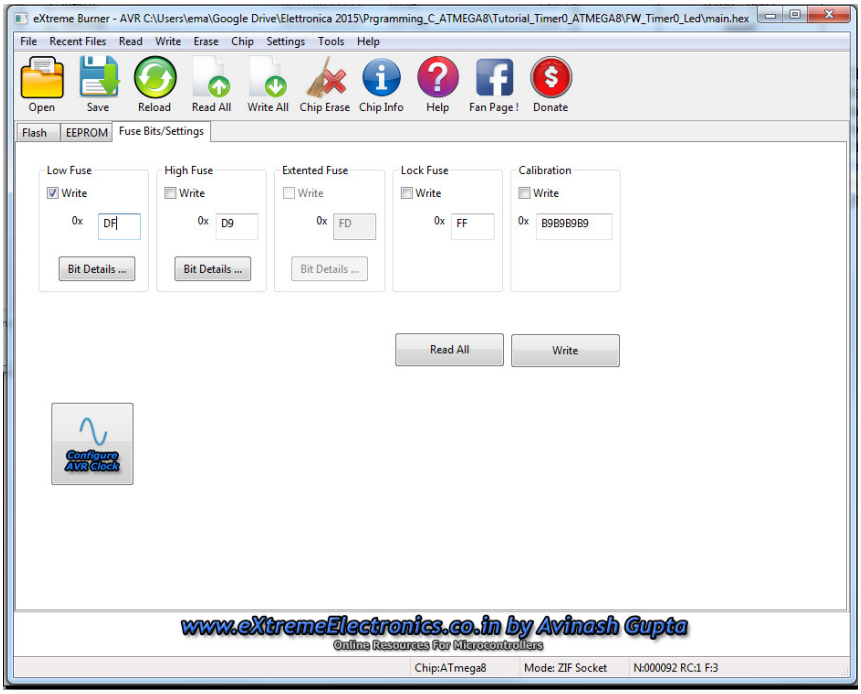
Ora al solito compiliamo con la procedura vista nel Tutorial #1.

Caricare il firmware

Apriamo Extreme Burner v1.4.3 e clicchiamo sul tasto Open per andare a caricare il file main.hex:



Ora è molto importante impostare i Fuse, andiamo sul tab chiamato "Fuse Bits/Settings" e impostiamo i Fuse esattamente come l'immagine che segue:



Quindi:

Low Fuse = 0xDF

High Fuse = 0xD9

Lock Fuse = 0xFF

Calibration = 0xB9B9B9B9

Mettete anche la spunta su write su di almeno il "Low Fuse"!

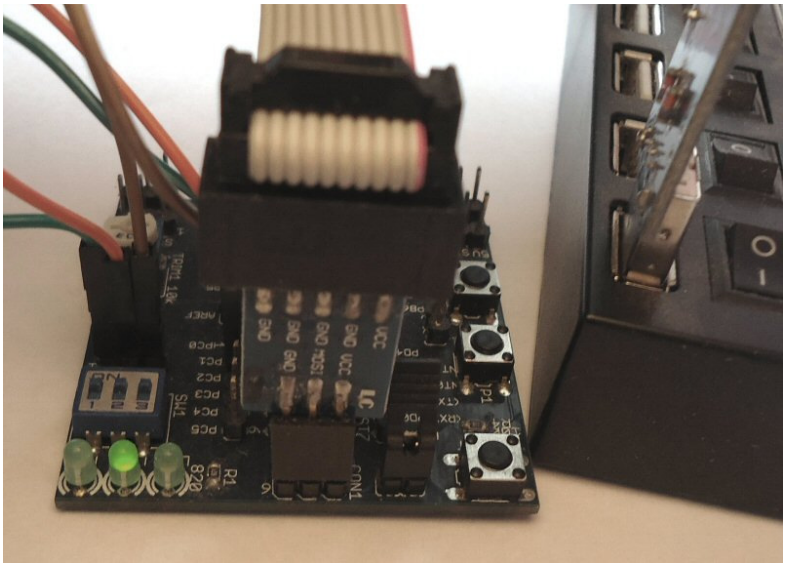
Rispetto al tutorial #1 il "Low Fuse" è cambiato, questa volta abbiamo impostato il clock esterno con quarzo a 16MHz.

Per capire come impostare il clock possiamo vedere il datasheet dell'ATmega8 a pagina 26 oppure ci affidiamo al fuse calculator:

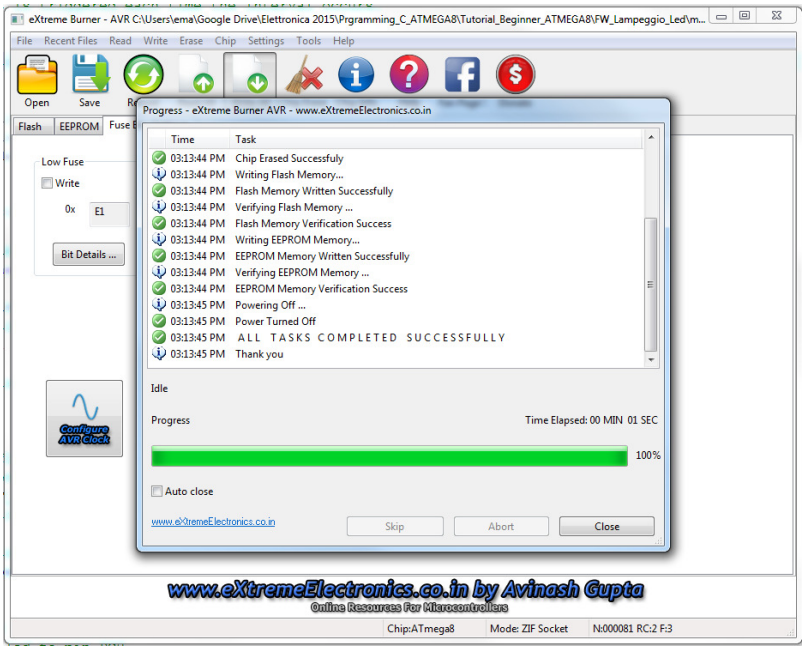
http://www.engbedded.com/fusecalc/

Ricordate di non cambiare i fuse se non capite cosa state facendo, potreste non poter più riprogrammare il vostro ATmega8.

Ora possiamo finalmente caricare il programma, assicuriamoci dei collegamenti come da foto:



Clicchiamo su Write All, se il caricamento è andato a buon fine allora avremo un schermata tipo questa:



Se avete collegato con un jumper il PIN23 al led centrale lo vedremo immediatamente lampeggiare!

SE NON LAMPEGGIA, POSSIBILI VERIFICHE:

1) Sulla demoboard usata il quarzo da 16MHz deve essere collegato inserendo i jumper "JP1" e "JP2".

2) Non avete scritto i fuse, vi siete dimenticati di cliccare su "write" per il "Low Fuse" nel loader eXtreme Burner.

3) Non avete premuto il tasto di "RESET" sulla demoboard dopo aver inserito i jumper per collegare il quarzo.

4) Vi siete dimenticati il martello per solleccitare il vostro ATmega8 ..... SCHERZO!! NON PRENDETELO A MARTELLATE!! Domani sono sicuro che sarete più rilassati e riuscirete a farlo funzionare! ;)