Luca Di Liello (Luca1995)

TERMOMETRO DIGITALE CON PIC18 PIERIN E NTC SENZA UTILIZZARE ADC

15 September 2013

Una piccola introduzione

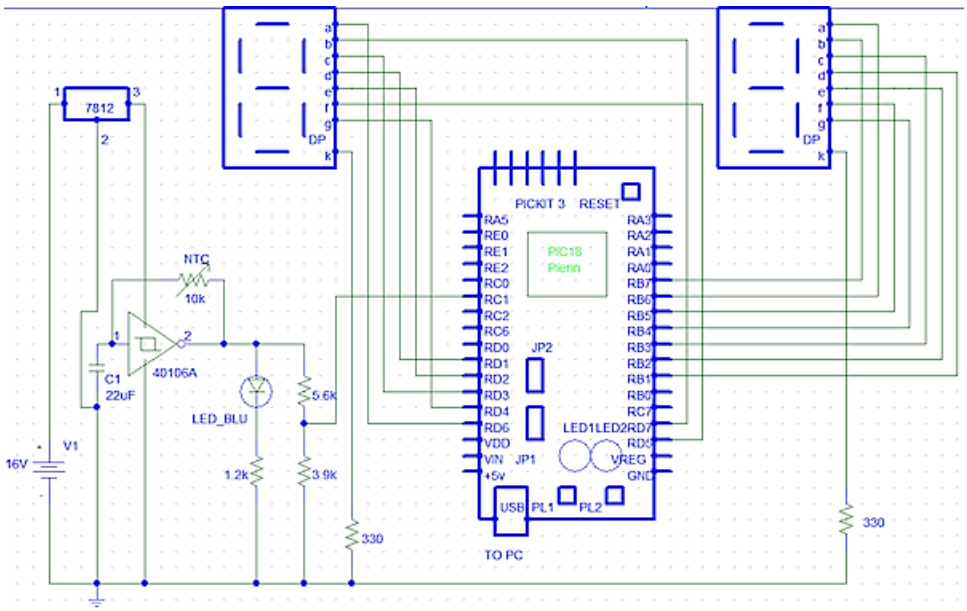
Ecco che finalmente riesco a scrivere un bell'articolo sulla mia prima esperienza con il PIC18 Pierin ed anche i microcontrollori in generale, a parte Arduino. Ringrazio anticipatamente TardoFreak, senza il quale non avrei creato un bel niente grazie al suo microcontrollore open source. Partiamo dalle basi: come si può leggere dal titolo ho realizzato un termometro digitale, che in qualche modo segnerà dei numeri su 2 piccoli display a 7 segmenti rossi; i dati provenienti dall'NTC sono stati elaborati dal PIC18 il quale infine li ha mostrati sui 2 display già citati. Siccome non ho ancora imparato bene come si utilizza l'ADC del PIC18 ho cercato una via alternativa, cioè quella di usare solo una porta del micro come ingresso (RC1).

A quel punto mi son chiesto: se non posso acquisire dati su più bit ma solo su uno come posso fare? La risposta mi è arrivata subito vedendo un led del PC lampeggiare con frequenza fissa. Così ho deciso di montare un piccolo oscillatore ad onda quadra con una porta NOT a trigger di schmitt ed inserire il mio NTC come resistenza parallela. Poi ho scelto un condensatore per avere una frequenza di 5Hz circa e dal calcolo è risultato 18uF. Valore più vicino disponibile: 22uF elettrolitico.

Progettare e montare il circuito su bread board

Per fortuna a casa ho parecchie bread board usate con i fori allargati dal tempo (le usava mio padre) e così sono riuscito ad inserirci anche il PIC18 Pierin senza tanti problemi in una di quelle. Mancando di convertitori BCD to 7 segmenti ho utilizzato l'intero registro B di I/O per il display delle decine e l'intero registro D per il display delle unità. All'uscita dell'oscillatore ho collegato anche un led per avere un'idea della frequenza a cui lavora (e mi piacciono tanto i led blu). L'alimentazione proviene da un vecchio trasformatore a 16V molto instabile; infatti con un carico piccolissimo, l'impedenza del tester digitale, quest'ultimo segnava circa 18V, mentre con un carico più pesante, una resistenza da 100 ohm, arrivava a 13V. Di norma il costruttore indica 12V così ho utilizzato un LM7812 e morta lì. Ora ho sempre 12V sull'integrato CD40106 per evitare, oltre che a bruciare il dispositivo, che la costante di carica e scarica del condensatore vari in base alla tensione e quindi cambi poi (leggermente)

anche la frequenza dell'oscillatore ad onda quadra. Ecco a voi uno schema completo del progetto



Schema

Qui l'immagine ingrandita

Montando il circuito sono passato subito alla programmazione ed alla calibrazione del termometro.....

Scrittura del programma C

Funzione per creare un ritardo

Parte del programma, come il timer software, è stata presa dal codice del modello base che per primo viene testato sul PIC18 Pierin. Molte funzioni invece le ho dovute inventare appositamente come quelle per comandare i display sui registri di I/O B e D.
Comunque ho modificato solo il file main.c, quindi fornirò in seguito solo il codice di questo senza ricopiare anche gli altri presenti quali main.h, configurazione.h, mappa_int.h e main.h.
Ho inoltre utilizzato la libreria math.h per poter usufruire della funzione logaritmo naturale.
Iniziamo ad analizzare il programma: la funzione ritardo permette di far aspettare al programma un certo lasso di tempo prima di proseguire

```
void ritardo(int tempo)
{
```

```
  timer_delay=tempo;
  while(1)
    {
      if(!timer_delay)break;
    }
}
```

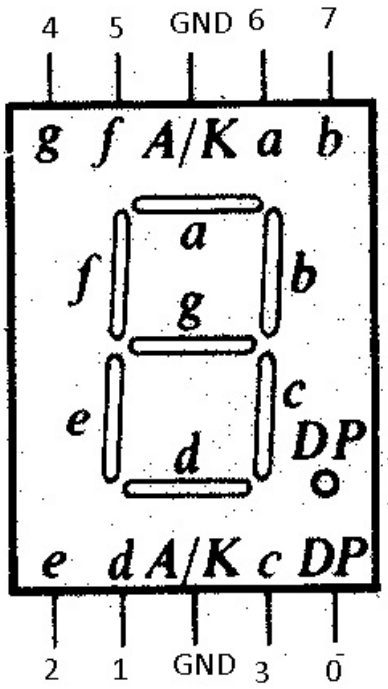
Si inizializza tempo con un valore, per esempio 1000 (il timer software di sistema è gestito da interrupt di 1ms); la variabile tempo viene copiata nel timer_delay che ogni 1ms diminuisce di 1 grazie al timer2 che genera l'interrupt; Il ciclo while permette al programma di continuare solo dopo che il timer software ha raggiunto lo 0.
Quindi la funzione ritarda l'esecuzione del programma di tanti millisecondi quanti ne vengono indicati tra parentesi nella funzione.

Gestione dei display

Per gestire i display ho creato delle semplici funzioni void che scrivono 8 bit sulla porta senza dover ogni volta ricordarmi la corrispondente uscita collegata ad ogni segmento. Con un'esempio sarà tutto più chiaro:

```
void numero0D(void)
{
LATDbits.LATD7 = 1;
LATDbits.LATD6 = 1;
LATDbits.LATD5 = 1;
LATDbits.LATD4 = 0;
LATDbits.LATD3 = 1;
LATDbits.LATD2 = 1;
LATDbits.LATD1 = 1;
}
```

Lo schema delle connessioni al display sono riportate nella seguente immagine. Nella funzione non appare lo 0 perchè non ho collegato il punto, dato che era superfluo.



display 7 segmenti

Come si può vedere dalla foto i collegamenti non sono molto logici, (per esempio "segmento a" collegato a 0, "segmento b" collegato a 1 ecc..) perchè tanto avrei dovuto scrivere lo stesso la funzione riportata sopra e in questo modo ho potuto collegare i cavetti jumper nel modo più comodo visto che non sono tutti lunghissimi. Naturalmente la funzione l'ho ripetuta per tutti i numeri (0,1,2,..,9) e sia per il display sul registro B che quello sul D. Andiamo avanti....

Funzione per numeri composti sui 2 display

Mancava ancora una funzione che mi permettesse (dato un numero maggiore o uguale a 0 e minore di 100) di scriverlo sui 2 display. Praticamente di trasformarli in un unico "schermo" che segni il numero su 2 cifre. Più che fantasia mi è costato scriverla (ho qualche lettera della tastiera consumata)... Eccola a voi nella sua semplicità:

```
int scrivi_numero(int x)
{
if (x == 0){ numero0B(); numero0D();}
if (x == 1){ numero0B(); numero1D();}
if (x == 2){ numero0B(); numero2D();}
if (x == 3){ numero0B(); numero3D();}
if (x == 4){ numero0B(); numero4D();}
if (x == 5){ numero0B(); numero5D();}
```

```
if (x == 6){ numero0B(); numero6D();}
if (x == 7){ numero0B(); numero7D();}
if (x == 8){ numero0B(); numero8D();}
if (x == 9){ numero0B(); numero9D();}
if (x == 10){ numero1B(); numero0D();}
if (x == 11){ numero1B(); numero1D();}
if (x == 12){ numero1B(); numero2D();}
if (x == 13){ numero1B(); numero3D();}
//eccetera.....................
if (x == 99){ numero9B(); numero9D();}
if (x > 99){ trattiniB(); trattiniD();}
if (x < 0){ trattiniB(); numero0D();}
}
```

Da notare le ultime 2 righe che servono per segnalare il superamento di 99 e la discesa sotto lo zero. La funzione trattini accende i segmenti centrali del diplay. Nel caso di (x > 99) si accendono entrambi, nel caso di (x < 0) appare la scritta - 0. Quindi inserendo un numero intero nella funzione scrivi_numero(int x) esso verrà visualizzato sul display a 2 cifre. Dopo tutte queste funzioni inizia ora il main vero e proprio.....

Il main

Dichiaro un paio di variabili:

```
double a;
double durata = 0;
double duratas = 0;
double logaritmo1 = 0;
double resistenza;
double temperaturaK = 0;
double temperaturaC = 0;
double resistenza_at_25C = 11740;
double costanteB = 2704;
double temperatura_base = 295;
```

Serviranno per il calcolo di una temperatura a partire da un tempo. Lo scopo è questo: misurare la durata di una oscillazione positiva e contare ogni millisecondo per 2 (in questo modo trovo subito il periodo invece che il semi-periodo, visto che in realtà si misura solo la semi-oscillazione positiva e non tutto), dividere il periodo per 1000 perchè era in millisecondi, trovare la resistenza ed infine la temperatura. Ecco un po di formule..... Data una certa durata:

$$Durata(s) = Durata(ms) \ / \ 1000$$

Trovo ora la resistenza dato che:
$T = 1{,}4 \cdot R \cdot C$
$R = T/(1{,}4 \cdot C)$

E ora, partendo dalla resistenza, di ricava la temperatura:

Temperatura(K) = 1/(((ln(R_T/R_0))/B)+(1/T_0))

R_0 sta per la resistenza a T_0 che vale 25°C. La B è un parametro caratteristico di ogni NTC che varia tra un minimo di 2500K ad un massimo di 5700K. Nel mio caso vale 2704. Si può calcolare invertendo la formula soprastante a patto che si abbiano a disposizione i dati a due temperature e le relative resistenze.

Infine, si convertono i gradi Kelvin in gradi celsius

$$Temperatura(C) = Temperatura(K) - 273$$

Le istruzioni che nel main controllano il timer2, i suoi interrupt e il PLL sono rimaste invariate. Con prescaler che divide per 16, postscaler che divide per 5 e comparatore a 150 si ha un ritardo di 1ms esatto. La relazione tra queste grandezza è:
$Ritardo(s) = (PR2 \cdot POST \cdot PRES \cdot 4)/(48.000.000)$
Gli I/O sono stati dichiarati nel seguente modo:

```
// Inizializza la PORTD
TRISD = 0x00;
// Inizializza la PORTB
TRISB = 0x00;
//Inizializza la PORTC
TRISCbits.TRISC1 = 1;


// Mette a 0 tutte le uscite
LATD = 0;
LATB = 0;
```

Infine riporto il ciclo for infinito che permette al progamma di continuare a calcolare le temperature e mostrarle all'osservatore.....

```
for(;;)
{
//Trovo la durata dell'oscillazione per poi arrivare alla temperatura.
//Il primo ciclo while serve a "sintonizzarsi" col circuito in modo che il conteggio
//non inizi mentre l'oscillatore è già ad un livello alto. Inizializo sempre la durata
```

```
//a 0 altrimenti continuerebbe da dove si era fermato il ciclo precedente


durata=0;


while(!PORTCbits.RC1){}


while(PORTCbits.RC1)
    {
durata = durata + 2;
ritardo(1);
    }


//Trasformo il ritardo in secondi dopo averlo acquisito in ms
duratas = durata/1000;


//Trovo la resistenza partendo dalla solita formula dell'oscillatore a not triggerate:
//.....dati i condensatori che posseggo della prima guerra mondiale e gli integrati anc
//...ho trovato che 15uF nella formula funzionano meglio di 22uF durante la taratura.

resistenza = (duratas / (1.4*0.000015));


//Trovo subito il logaritmo per poi poterlo inserire già calcolato nella formula succes
//..serve solo per una maggiore leggibilità del programma anche per me.




logaritmo1 = log(resistenza/resistenza_at_25C);
//Eseguo prima il logaritmo per poi inserirlo come variabile nel calcolo successivo....
//base (25°C) è espressa in Kelvin. La costante B è un particolare numero
//compreso tra 2500k e 5700k che è caratteristico di ogni singolo NTC. Esso serve nella
//temperatura e resistenza.



a = logaritmo1/costanteB + (1/temperatura_base);
```

```
//Faccio l'inverso per trovare la temperatura in Kelvin;
temperaturaK=1/a;

//Trovo la temperatura in gradi celsius sottraendo 273K.
temperaturaC = temperaturaK - 273;

//Uso la funzione scrivi numero per digitare la temperatura sui 2 display a 7 segmenti.
scrivi_numero(temperaturaC);

//Chiudo il for(;;)}
```

La parte finale consiste in alcuni calcoli e l'uso della funzione scrivi_numero(int x) per mostrare il risultato sul display a 2 cifre. Se avete domande o correzioni da eseguire non esitate a commentare.
Per scaricare il progetto MPLAB IDE v8.92 completo clicca qua.
Spero che l'articolo sia stato di vostro gradimento alla prossima, Luca
Ecco a voi un breve video----------

Termometro digitale Pic18 Pierin NTC no ADC youtbe

Estratto da "http://www.electroyou.it/mediawiki/index.php?title=UsersPages:Luca1995:termometro-digitale-con-pic18-pierin-e-ntc-senza-utilizzare-adc"