FrancescoFP

LET'S START! IL PRIMISSIMO PROGRAMMA 1A PARTE

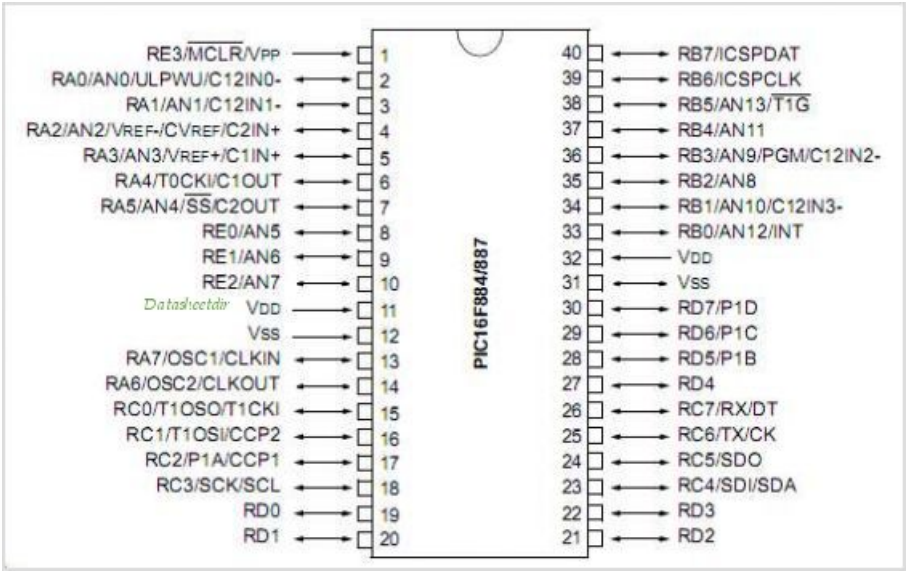
18 October 2012

Abstract

Prima di iniziare a scrivere il nostro primo programma, abbiamo bisogno di conoscere alcuni concetti di base. Ancora? Si, giusto qualcosa, in modo da avere "coscienza" di ciò che stiamo facendo.
Come detto nell'articolo precedente, il linguaggio utilizzato è il C, quindi non dovrebbe essere troppo difficile iniziare per chi è digiuno di programmazione! Bando alle ciance, vediamo cosa ci occorre!

Come è fatto un PIC

Prima di scrivere il programma e montare il circuito, dobbiamo sapere come è fatto un PIC, almeno a grandi linee. Il modo migliore per capirlo è analizzare l'immagine di un *tipico* PIC. Non c'è da preoccuparsi se all'inizio tutte quelle scritte ci confondono o non sappiamo cosa indicano, per ora concentriamoci sui componenti principali, il resto verrà da sè.



PIC16F887

Analizziamo singolarmente ogni dettaglio, capiremo così come è fatto un PIC e sapremo gestirlo a nostro piacimento. Viene preso qui in esempio il PIC16F887, per il semplice motivo che avevo questo nel cassetto :) **Qualsiasi** PIC potrebbe andare bene, in quanto quello che dobbiamo fare è *semplicissimo*!

- Il PIC ha 40 pin(piedini) disposti simmetricamente su due file da 20
- Alcuni pin sono segnati da una barra sopra($\overline{esempio}$), questo indica che sono *attivi* a livello logico *basso*, cioè per svolgere la loro funzione devono essere posti al livello logico basso, cioè al potenziale di riferimento, in quanto siamo in Logica positiva
- Il PIC presenta ben 36 pin di I/O digitale, contrassegnati dalla dicitura **R***L*x, dove *L* indica una lettera(A,B,C...) e la x un numero tra 0 e 7.
- Ogni piedino è accompagnato dalla sua descrizione. A volte più di una descrizione è presente, e per separarla dalle altre si utilizza lo *slash* Ad/ Esempio/Così. Quindi se per esempio leggiamo la descrizione del piedino 18 possiamo subito sapere che può svolgere la funzione di normale I/O(segnata come RC3), la funzione SCK e la funzione SCL. Non ci importa per ora sapere cosa vogliono dire questi acronimi, ci interessa solo capire il concetto di *multifunzionalità* di alcuni pin. Ovviamente non è possibile far svolgere al PIC 2 o più funzioni contemporaneamente
- Il piedino numero 1 è indicato da una mezzaluna, questo ci permette di riconoscerlo "al volo"
- Ogni PIC ha un piedino segnato come **MCLR**, questo piedino è importante, in quanto ci permette di resettare il PIC(far ripartire il programma dall'inizio) ponendolo al potenziale di riferimento. E' inoltre fondamentale al momento della programmazione in quanto esso riceve circa 13 Volt, necessari per cancellare e riscrivere la memoria interna del PIC.
- Il PIC ha *due coppie* di piedini segnati come VSS/VDD. Nota bene: I piedini segnati come VSS vanno posti al potenziale di riferimento, quelli segnati come VDD al potenziale maggiore(solitamente 5 Volt). **Tutti** e quattro i piedini vanno collegati, non solo due, ma tutti e quattro. In caso contrario potrebbe(sicuramente) non funzionare qualcosa, ci sarà un motivo se ne hanno fatti quattro e non due :)
- Il PIC ha due piedini *multifunzione* tra i quali spiccano le diciture "CLKIN/ OSC1" "CLKOUT/OSC2". *In questo PIC* i piedini sono il numero 13 e il numero 14, in un altro PIC potrebbero cambiare. Quello che ci interessa sapere è che a questi due piedini va collegato un quarzo(di una frequenza non superiore a un certo valore) e due condensatori *solitamente* da 22pf (PicoFarad cioè $22 \cdot 10^{-12} Farad$).
- Alcuni pin del PIC, tra le varie funzioni che hanno, ne includono una nominata **ANx** dove *x* è un numero compreso tra 0 e 13. Questa dicitura indica un **ingresso analogico**(solo ingresso, non uscita), cioè un pin che può accettare valori analogici e non solo digitali(ovviamente compresi nel range

$VSS \div VDD$). Questo significa che possiamo ad esempio fornirgli in ingresso 1.7 Volt, cosa non "desiderabile" se l'ingresso è digitale. Questi ingressi possono pure essere configurati come ingressi/uscite digitali, tramite particolari impostazioni.

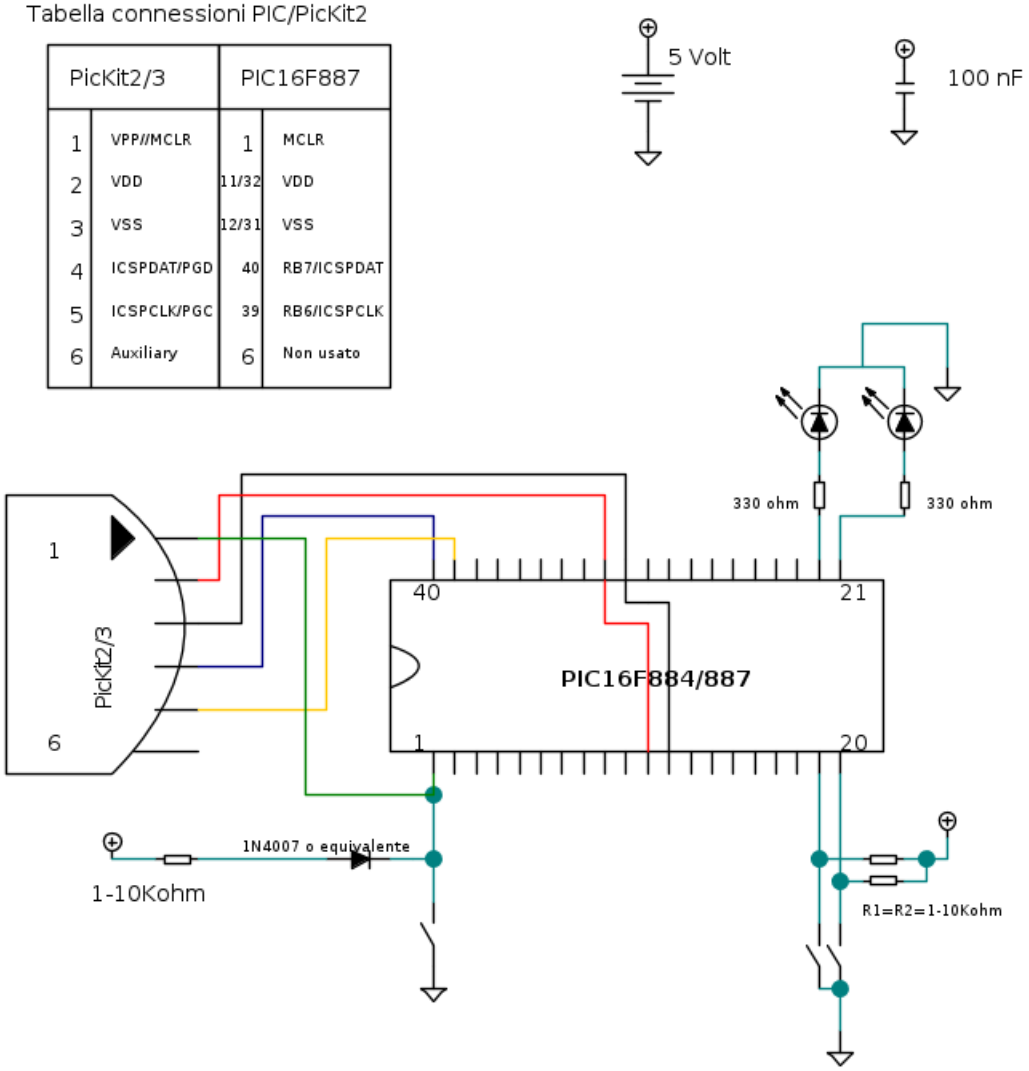
Ora sappiamo, a grandi linee, come è fatto un PIC. Possiamo iniziare a montare il circuito di base, con il quale sperimenteremo i concetti di: *registro*,*ciclo infinito*,*operatore logico* ecc... Non sono concetti difficili, li impareremo man mano che li utilizzeremo, non c'è da preoccuparsi!

Il circuito di base

Eh si, prima di scrivere qualsiasi istruzione/programma abbiamo bisogno di un circuito in cui inserire il nostro PIC. Il circuito di base che ci servirà per le nostre prove non è affatto complesso, ed è alla portata di tutti! Per disegnarlo ho utilizzato l'**ottimo** fidocadj di DarwinNE, un leggero quanto potente tool di grafica *vettoriale* scritto in java, che non richiede installazione(dunque è portabile) e *multipiattaforma*. E' questo il programma per disegnare circuiti utilizzato sul forum, in quanto è possibile modificare schemi altrui con pochi semplici click, invece di impazzire a ridisegnare il circuito per cambiare una resistenza o un condensatore. Come ho detto prima è un programma di grafica vettoriale, ciò vuol dire che non *sgrana* le immagini se ingrandite, e come se tutto ciò non bastasse, è semplicissimo da usare e *Open Source*!
Questo è il nostro circuito di base:

Tabella connessioni PIC/PicKit2

PicKit2/3		PIC16F887	
1	VPP//MCLR	1	MCLR
2	VDD	11/32	VDD
3	VSS	12/31	VSS
4	ICSPDAT/PGD	40	RB7/ICSPDAT
5	ICSPCLK/PGC	39	RB6/ICSPCLK
6	Auxiliary	6	Non usato



Ma quanti colori ci hai messo dentro un disegno da 4 soldi?
Forse ho esagerato, ma volevo essere sicuro che le connessioni risultassero chiare, assegnando a ogni colore una diversa funzione. Forse ho ottenuto l'effetto contrario, ma spero di no. Dal prossimo articolo in poi userò un solo colore, per questa volta però, visto che è la prima, ho abbondato pesantemente! Inziamo con l'analizzare questo circuito semplice semplice, in modo da passare alla scrittura del nostro primo programma!
In alto a sinistra, troviamo una tabella che ci spiega le connessioni da effettuare tra il PIC16F887 e il PicKit2/3. Queste connessioni molto probabilmente non saranno uguali a quelle di un altro PIC, in quanto potrebbe cambiare il *pinout*(disposizione dei pin). Per sapere la disposizione dei pin del vostro PIC, basta cercarne il datasheet(foglio tecnico), che oltre a contenere un'immagine simile a quella sopra, contiene **TUTTO** ciò che c'è da sapere su quel PIC. E' il libro sacro di ogni

dispositivo elettronico(non solo PIC, ma anche integrati/transistor/diodi ecc...), quindi va letto **attentamente** per capire cosa si può/non può fare con quel PIC, integrato o dispositivo. *Ogni buon progetto non può prescindere dallo studio attento e accurato dei datasheet dei componenti impiegati.*
Detto questo, analizziamo la parte restante del circuito. In alto a destra troviamo il simbolo di una batteria, un terminale + e un triangolino verso il basso, che indica l'*equipotenzialità* del circuito. Perchè l'ho disegnata? Per praticità. Perchè così al posto di disegnare tutta la batteria mi basta posizionare il terminale + o il triangolino a seconda di cosa voglio indicare. Ora il "pezzo forte", il PIC tutto colorato!
Iniziamo col dire che il colore rosso indica il potenziale maggiore(5 Volt) mentre il nero l'equipotenzialità. In questo modo è facile capire quali sono i pin di alimentazione del PIC a "colpo d'occhio". Il fatto che queste linee passino "dentro" il PIC indica che i piedini sono collegati insieme, e per evitare di fare tutto il giro del PIC, li ho fatti passare sotto, semplice questione di ordine visivo del circuito. Il piedino numero 1 è collegato al piedino numero 1 del PicKit2/3 ma ha un'altra connessione: è collegato con una resistenza ai 5 Volt, e pure con un bottone "normalmente aperto" che *chiude* verso il riferimento. Mi spiego meglio:
Il piedino 1, quando l'interruttore è aperto, è collegato sia ai 5 Volt tramite una resistenza(Resistenza di pull up) sia al piedino del PicKit2/3. In questo modo se, con un Multimetro, misuriamo la tensione tra il potenziale di riferimento(equipotenzialità) e il piedino 1 troviamo 5 Volt(in realtà qualcosina in meno). Ora, quando l'interruttore viene chiuso il piedino viene posto al livello logico 0 in quanto viene collegato al potenziale di riferimento. Questo avviene perchè il collegamento con il potenziale di riferimento è realizzato tramite un bottone e del filo, e non tramite una resistenza(come per i 5 Volt), quindi la corrente scorre nella strada(ramo di circuito) nella quale "incontra" minor resistenza, quella del bottone. Quel grosso cerchio riempito di grigio indica una connessione tra due o più elementi. Ad esempio, sotto la tabella in alto a sinistra, la linea rossa e la linea nera non sono connesse(e NON devono esserlo), in quanto manca quell'enorme punto. Invece nel pin 1, è presente il simbolo della connessione in quanto sono collegate tra loro la linea verde e quella grigia. In generale, se due linee passano sopra tra loro in uno schema, se non vi è questo punto quelle linee non sono collegate tra loro.
I simboli "strani" racchiusi in un cerchio sono diodi LED, componenti elettronici(diodi) che emettono luce se attraversati da corrente elettrica **E nello stesso tempo** la tensione ai loro capi è della giusta polarità, cioè l'anodo ha il potenziale maggiore(di almeno 0.7 Volt) del catodo. Se inseriti al contrario non si illuminano, e potrebbero anche danneggiarsi.
E' chiaro che i pin 19 e 20 verranno utilizzati come ingressi digitali, i pin 21/22 come uscite digitali. Non c'è un particolare motivo che mi ha spinto a utilizzare questi pin, li ho scelti e basta! I più attenti avranno notato che poco fa ho parlato di quarzo, l'oscillatore(molto preciso) che ci permette di far funzionare il PIC. "Ma tu prima non ci hai detto di comprarlo!" Questa potrebbe essere l'obiezione di qualcuno, che ha giustamente ragione. Non vi ho detto che era necessario prima perchè questo

PIC(come quasi tutti ormai) ha un oscillatore **interno**(di solito più lento del quarzo) che ci permette di non comprare il quarzo, almeno agli inizi e se realizziamo progetti in cui non è richiesta un'elevata potenza di calcolo ed un'elevata velocità.
Ora che sappiamo come è fatto un PIC, iniziamo a scrivere il nostro programma.

Il programma

Un programma in C è generalmente strutturato così:
Inclusione files header(files .h) tramite la "direttiva" **#include**
Definizione di costanti e valori tramite la direttiva **#define**
Definizione delle variabili globali(se presenti). E' però sconsigliato usare per problemi di *portabilità del codice*
Inizio del programma tramite la funzione **void main(void)**
Ciclo infinito in cui eseguiamo le istruzioni e le funzioni
Chiusura del programma, effettuata con la parentesi graffa che chiude quella aperta nel main(). Per scrivere la parentesi graffa digitare **alt+123**({) o **alt+125**(}) nel tastierino numerico, oppure **shift+alt gr+é** o **shift+alt gr+***.
Ogni programma deve essere accompagnato da un flowchart(diagramma di flusso) e da un'analisi scritta(o visuale) del problema, altrimenti si rischia di scrivere codice non funzionante, difficile da mantenere e aggiornare(e molto spesso anche da capire). Per questa volta, dato che il problema è MOLTO semplice evito di disegnare il flowchart, ma di norma dovrei farlo anche stavolta.
Dopo aver montato il circuito correttamente, aver capito come si struttura un programma in C, possiamo *fissare i nostri obbiettivi e i nostri vincoli*, in modo da averceli **ben chiari** durante la risoluzione del problema stesso. Riassumiamo in una semplice tabella il problema da risolvere.

Obbiettivi	Vincoli
Accendere il led 1	solo alla pressione del pulsante 1
Accendere il led 2	solo alla pressione del pulsante 2
Accendere entrambi i led	solo alla pressione di entrambi i pulsanti

La tabella è un ottimo strumento per sintetizzare in maniera chiara concetti che bisogna poi tenere a mente, per questo la preferisco a righe e righe di spiegazioni.
Per ora pubblico la prima parte, la seconda parte è in lavorazione e spero di pubblicarla a breve, 1 o 2 giorni al massimo.

Stay tuned :)