Steve Blackbird (TardoFreak)

INIZIARE CON IL DSPIC33F

8 September 2010

Sperimentare con i PIC

Tempo addietro era quasi impossibile (per motivi di costo e di disponibilià dei componenti) **sperimentare la programmazione** e/o la realizzazione di circuiti basati su microcontrollori da parte di un pubblico hobbista. Questo perché i sistemi di sviluppo, gli eventuali **compilatori** per linguaggi ad alto livello ed i circuiti integrati per l' emulazione costavano veramente tanto. Spendere diversi milioni di vecchie lire per sviluppare un singolo microcontrollore era cosa normale. Questo relegava l' utilizzo dei microcontrollori ad un' utenza professionale e, di fatto, impediva all' hobbista la possibilità di studiare e di sperimentare. Diciamo pure che era impensabile poter utilizzare un microcontrollore per un' applicazione unica ed hobbistica.

Fortunatamente i tempi sono cambiati, gli **emulatori in-circuit** hanno oramai prezzi accessibili anche da parte degli hobbisti ed i microcontrollori si possono acquistare facilmente e ad un prezzo basso. Oggi è normale utilizzare un microcontrollore al posto di una manciata di integrati logici, e questo dà finalmente la possibilità all' hobbista di cimentarsi nella scrittura del **firmware**. Diverse case costruttrici offrono emulatori e microcontrollori a basso costo e ad alte prestazioni; fra questa spicca la **Microchip** che offre una gamma vastissima di microcontrolli ad 8, 16 e 32 bit.

Questo articolo potrà sembrare banale ma forse può fornire un' utile traccia per chi si avvicina per la prima volta a questi microcontrollori. Contiene una serie di suggerimenti e cerca di impostare un approccio graduale e costruttivo per utilizzare al meglio ed in modo semplice questi oggetti. In questi giorni mi accingo a sviluppare un' apparecchiatura audio digitale ed ho scelto un microcontrollore a 16 bit veramente potente, con funzioni di DSP. Di primo acchito sembrerebbe un poco azzardato iniziare con un microcontrollore che, oltre ad essere sicuramente potente, è anche un oggetto molto complesso. In effetti i microcontrollori di fascia media a 8 bit sono meno complessi ma è anche vero che, con il giusto approccio, complessità non significa sempre difficoltà. Se poi scegliamo di sviluppare i programmi utilizzando il linguaggio C le difficoltà diminuiscono. D' altro canto l' utilizzo di un microcontrollore potente a 16 bit con parecchia RAM, ROM e periferiche, dà la possibilità di sperimentare molto e superare in partenza le limitazioni dei microcontrollori a 8 bit, e l' uso del linguaggio C permette di essere svincolati dall'

architettura interna e lontani dalla sua complessità. Iniziamo quindi nel modo giusto e non lasciamoci spaventare dall' apparente difficoltà.

Il dsPIC33FJ128GP804

Se apriamo il datasheet del PIC in oggetto è facile rimanere disorientati dalla complessità e dalle enormi possibilita' di questo microcontrollore. Si tratta infatti di un DSP con porte di ingresso/uscita, convertitori D/A e A/D, timers, interfacce di comunicazione seriale, interrupt controller, un complesso e versatile circuito di clock, orologio in tempo reale, interfaccia per bus CAN, I2C e chi piu' ne ha piu' ne metta. Sia ben chiaro che, se io ho scelto questo microcontrollore, è perché ha le caratteristiche che il mio progetto richiede ma, volendo sperimentare facilmente i DSP, si può utilizzare un altro micro come il dsPIC30F4013. E' sempre un mostro di velocità, con un bel pò di memoria ed ha il vantaggio di essere un Dual In Line a 40 pin ottimo per le sperimentaioni. Il micro che ho scelto viene proposto in un contenitore TQFP a 44 pin, quindi per montaggio superficiale e bisogna avere un circuito stampato di adattamento per usarlo con una basetta millefori. Sono, in buona sostanza oggetti che, se presi nel loro complesso sembrerebbero difficili da utilizzare. Ma non lo sono per una serie di motivi. Innanzi tutto i vari moduli che compongono il microcontrollore, se non utilizzati, è come se non ci fossero, come se non esistessero. Questo ci permette di non prenderli in considerazione, almeno all' inizio, e di lasciare il loro studio e la loro sperimentazione più avanti nel tempo. Non abbiamo quindi bisogno di preoccuparci di inizializzare i vari moduli per fare in modo che questi se ne stiano disattivati, calmi e buoni e che non interferiscano con il primo semplice programma che implementeremo all' inizio. Per un approccio all' insegna della semplicità suggerisco di tralasciare proprio tutte le bellissime e complesse funzioni che il microcontrollore mette a disposizione, e considerarlo soltanto come un semplice micro costituito da una CPU, una ROM, una RAM e delle porte di I/O digitali. Niente di piu' e niente di meno. Le complicazioni arriveranno dopo, se e quando ce le andremo a cercare!

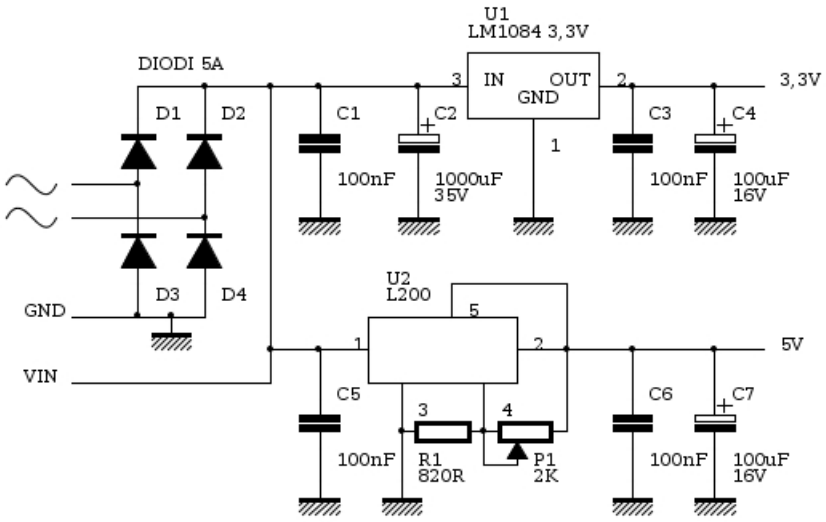
L' applicazione più semplice

Da sempre, quando mi avvicino ad un nuovo microcontrollore, cerco di implementare il programma piu' semplice che si può implementare in un microcontrollore: fargli modulare un pin di uscita (o fargli accendere un LED). Questo tipo di programma è il classico "Hello world" in versione micro. Il buono di un programma del genere è che, una volta implementato e fatto funzionare, servirà come base di partenza per tutti gli altri programmi che si svilupperanno. In ogni caso questo è il primo passo e fatto il primo passo si proseguirà verso una vera applicazione. Quello che ci serve per fare una cosa del genere e' avere un sistema di sviluppo minimo. La Microchip fornisce gratis un ambiente di sviluppo chiamato MPLAB ed un compilatore ANSI C per la famiglia dei PIC24 e dsPIC a 16bit chiamato MPLAB-C30. La versione gratuita

è la versione accademica e non ha limitazioni di codice. Quello che la differenzia dalle versioni a pagamento è la generazione del codice. In effetti il codice compilato è piu' lungo e lento di quello prodotto dalle versioni a pagamento ma è ugualmente efficace. Se però pensiamo che il PIC che andiamo ad utilizzare è un mostro da 40MIPS e che ha a disposizione ben 128KBytes di memoria di programma, l' avere un codice meno ottimizzato e piu' lento non rappresenta un problema, per fortuna aggiungo io! Abbiamo anche bisogno di un emulatore in-circuit che dovremmo comprare. Il PicKit3 fornito dalla stessa Microchip costa relativamente poco, meno di un pieno di benzina e ci permette di emulare praticamente tutte le famiglie di PIC, dagli 8 bit di fascia bassa ai 32 bit. Infine dobbiamo realizzare un circuito su millefori o su piastra sperimentale per avere fisicamente il nostro oggetto funzionante e, da questo, partire e collegarci gli altri compomenti elettronici per le nostre sperimentazioni.

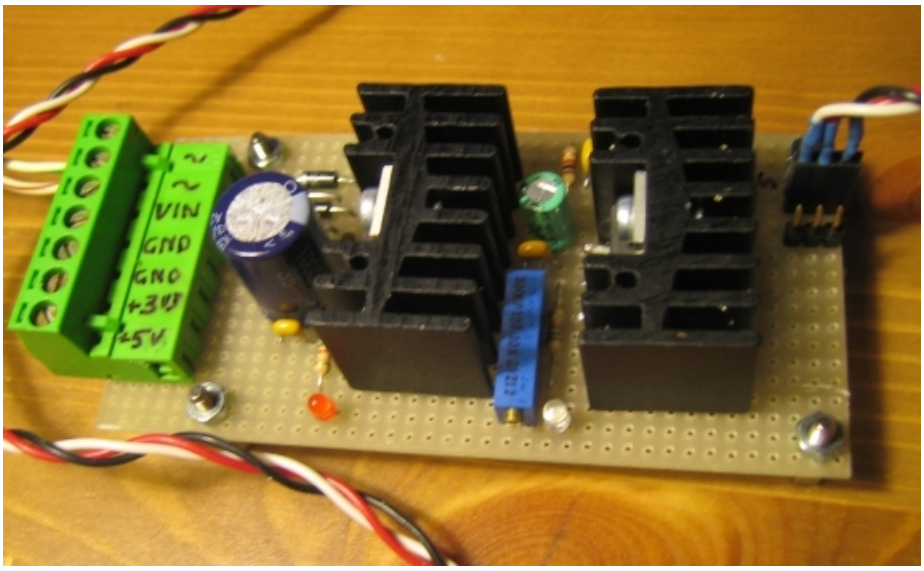
L' alimentatore

L' alimentazione dei PIC può essere di 3,3V o di 5V. Dipende dal modello. Alcuni le permettono entrambe, altri no. Ho pensato che fosse bene avere un circuitino di alimentazione fatto apposta per alimentare questa scheda di sviluppo ma che andasse bene anche per altre applicazioni. Si puo' benissimo usare un alimentatore da laboratorio ma io preferisco avere a disposizione le mie tensioni fisse in modo pratico e veloce. Ho quindi realizzato un alimentatore che mi fornisce le due tensioni e che ha la possibilità di erogare anche una discreta corrente. Mi e' capitato diverse volte, e mi capita tutt' oggi, di avere bisogno del 3,3V per il micro, del 5V per il display LCD alfanumerico e del 12V per i relè. Con questo alimentatore collegato a quello di laboratorio regolato a 12V posso averle tutte insieme. Questo e' il circuito:



AlimPIC.jpg

Per generare i 3,3V ho usato un regolatore in grado di erogare fino a 5A, non tanto perche' ce ne sia bisogno ma perché ... costano poco ed avere una buona corrente non e' mai un male. In questo modo posso alimentare diversi circuiti contemporaneamente circuiti senza problemi. Per il 5V ho usato un L200 che è in grado di fornire fino a 2A di corrente. L' alimentatore prevede l' ingresso di una tensione alternata da un trasformatore o da una sorgente in continua da regolare. Per il collegamento fra l' alimentatore e la scheda di sviluppo utilizzo dei semplici connettori per scheda a cui saldo i fili. Questi semplici connettori sono visibili nella parte destra della basetta. Preferisco usare questo sistema perché in questo modo posso saldarne diversi, utilizzare sempre gli stessi cavetti per il collegamento e collegare e scollegare velocemente piu' schede.



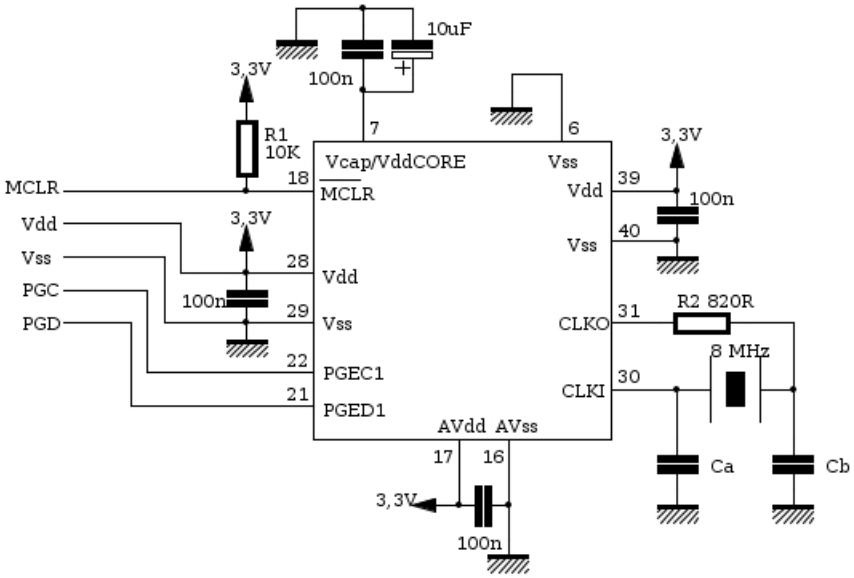
Alimentatore.jpg

Le tensioni di uscita sono anche disponibili sul connettore a sinistra della basetta. Due generosi dissipatori mi garantiscono un funzionamento senza problemi di sorta e due LED, uno rosso in basso a sinistra ed uno blu in basso vicino al trimmer indicano la presenza delle tensioni.

La scheda di sviluppo

Per la realizzazione dello schedino su cui svilupperò la mia applicazione ho fatto riferimento ad un paragrafo del datasheet intitolato "Guidelines for getting started with 16-bit digital signal controllers". Il titolo è molto chiaro ed il paragrafo fornisce utilissime informazioni per iniziare. Innanzi tutto mostra uno schema e delle indicazioni per il collegamento minimo del micro indicando quali alimentazioni collegare per farlo funzionare e i vari condensatori di filtro necessari per evitare problemi. In particolare c'è un piedino indicato come "Vcap/Vddcore" che è

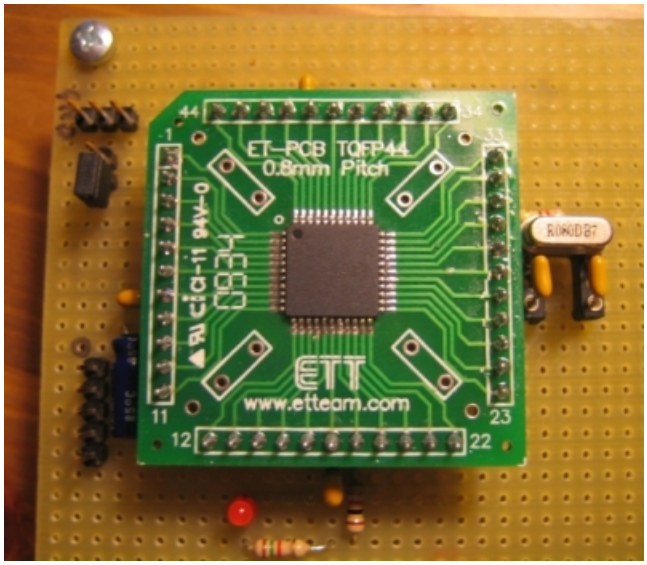
estremamente importante e non va assolutamente collegato alla Vdd. Infatti è l' uscita del regolatore interno del micro e l' hanno messo per essere collegato ad un condensatore di filtro da 10uF al tantalio. Non avendo condensatori al tantalio l' ho collegato al parallelo di un condensatore elettrolitico con un condensatore ceramico da 100nF. Il piedino di MCLR (il reset del micro) serve anche al programmatore/debugger. Nelle specifiche del PicKit3 e dell' ICD3 (quello che uso io) è scritto di non collegarci nessun condensatore o diodo e che una resistenza da 10K è sufficiente per un buon funzionamento. Quindi, vada per la 10K verso la Vdd! Questo è lo schema dello schedino di sviluppo.



SchSvil.jpg

A sinistra ci sono tutti i segnali da collegare al programmatore/debugger. Questi è anche in grado di fornire l' alimentazione al micro ma la stessa Microchip sconsiglia di fare questo per non caricare il programmatore. Io, per sicurezza e comodità, ho messo un jumper in serie al collegamento con l' alimentatore in modo da poter scegliere se alimentare il circuito tramite l' alimentatore o il programmatore. Ho anche montato un quarzo da 8Mhz con i suoi due condensatori ed una resistenza da 820R che può essere eventualmente ponticellata ma che è necessaria per alcuni tipi di quarzo e, visto che il quarzo lo potrei sostituire (è montato su zoccolo proprio per questo) la resistenza è praticamente d' obbligo. Ho scelto il quarzo da 8Mhz perchè rientra nei limiti indicati dal paragrafo in questione e con questo quarzo i condensatori sono da 22pF. In pratica, con un quarzo da 8Mhz e tenendo i valori di default di configurazione del micro (i cosiddetti "fuses") per l' oscillatore primario con PLL, la frequenza di lavoro sarà di 50Mhz. Un ottimo valore perchè permette di far lavorare il micro a 25 MIPS e senza che questi si trasformi in una piccola stufetta. In

effetti, a 25 MIPS, il micro assorbe intorno ai 40 mA mentre a 40 MIPS il consumo sale fino a 60 mA o più. In pratica a 25 MPIS è freddo mentre a 40 si sente scaldare.



Chip.jpg

La foto mostra il dettaglio della scheda dove ho montato il micro. Sulla destra c'è il quarzo con i due condensatori montati sullo zoccolo, in alto a sinistra il connettorino a 3 vie per l' alimentazione e, sul lato destro, il connettore per il programmatore/ debugger. Sotto al connettore dell' alimentazione si trova il jumper che ho inserito per scollegare il micro dall' alimentatore. In basso il mio solito LED rosso per tenere sott' occhio l' alimentazione.

Fuoco alle polveri!

Ed ora cerchiamo di far funzionare il nostro primo programma. Per fare questo dobbiamo, innanzi tutto, creare un nuovo progetto su MPLAB. Dò per scontato che abbiamo MPLAB ed il compilatore C30 già installati nel PC quindi chiamiamo MPLB e, tramite il "project wizard" (Project -> Project Wizard) creiamo un nuovo progetto selezionando opportunamente il microcontrollore, la toolsuite "Microchip C30 Toolsuite" la cartella dove risiederà il progetto, e diamo un nome al nostro progetto.

Una volta creato il progetto scegliamo il debugger cliccando su "Debugger" e scegliendolo. Per fare la prova possiamo utilizzare l' alimentazione fornita dallo stesso. Per fare questo andiamo in "Debugger" -> "Settings" e selezioniamo l' opzione che ci permette di alimentare il circuito attarverso il debugger. A questo punto possiamo collegarlo alla nostra scheda senza paura di arrostire il microcontrollore. Nella mia schedina ho inserito il LED con la resistenza sull' alimentazione proprio per avere l' indicazione della presenza della Vdd, consiglio di montarlo, costa poco ed è

molto utile. Quello che ci serve è un file (di solito chiamato "main.c" che contenga il programmino. Lo creiamo e lo riempiamo con il sorgente che potete trovare a questo indirizzo. Ed ora premiamo il tasto **<F10>** per compilare il tutto. La compilazione non dovrebbe dare errori. L' unico errore possibile potrebbe essere dato dal fatto che il compilatore non trova il file **p33FJ128GP804.h**. In questo caso è sufficiente istruire il compilatore cliccando su "Project" -> "Buil Options" -> "Project", nella finestra "directories" selezionare dal menù a tendina "Include Search Path", premere il tasto "New" e selezionare la cartella che contiene il file. Ora deve funzionare. Una volta compilato il programma lo carichiamo nel micro, lo facciamo partire, colleghiamo l' oscilloscopio a massa e al piedino 34 del micro e, con una base tempi di 2us/div. vedremo la nostra bella onda quadra comparire.

Il firmware

Il firmware è, come si può notare, estremamente semplice. La prima linea valida è la direttiva che include il file di defiinizione della macchina. E' un file **estremamente importante** e ad esso faremo riferimento in futuro per ... diciamo tutto. Contiene infatti le definizioni e gli indirizzi delle porte, delle periferiche, in pratica di tutto quello che serve per utilizzare in toto il micro. Subito dopo troviamo le linee che servono per programmare i fuses del micro. In pratica l' unica cosa importante che ho selezionato (e che serve per questo programma) è il tipo di oscillatore. Per conoscere a fondo le opzioni bisogna leggere il datasheet specifico del micro o, meglio anora, il reference manual della famiglia dsPIC33. Possiamo però farci una rapida idea del significato aprendo il file di definizione della macchina. Al fondo sono elencate tutte le defines ed il significato delle varie opzioni. Ho volutamente scritto tutte le opzioni per avere una configurazione completa sicuro di non tralasciare nulla. In futuro basterà cambiare opportunamente il/i fuses per adattarlo alle esigenze delle varie applicazioni. Subito dopo incontriamo la funzione **main**. Le prime tre linee di programma servono per selezionare il clock a 80MHz. Come ho detto prima io uso un quarzo da 8MHz e, senza queste 3 linee, il micro funziona a 50 MHz. di clock. Se avessi usato un quarzo a 4MHz (il minimo richiesto dalle specifiche) la frequenza di clock risulterebbe dimezzata. Queste tre linee possono essere benissimo eliminate e lasciare la condizione iniziale. Io le ho lasciate nel listato per dare la possibilità di poter modificare facilmente la frequenza di clock. E' chiaro che per fare le cose ben fatte è necessario leggere attentamente il reference manual alla sezione "oscillator". L' oscillatore di questo micro è complessoo ma estremamente versatile e permette di scegliere le frequenze con grande accuratezza. Subito dopo troviamo l' inizializzazione delle porte. Per prima cosa ho selezionato tutti gli ingressi come digitali e poi, mettendo a 0 tutti i bits del TRISA, ho selezionato la PORTA come tutti output. In questa sezione del main dovrebbero trovare posto tutte le inizializzazioni perchè nel loop infinito del main trova posto il nocciolo dell' applicazione vera e propria. Quello che segue e' un loop infinito che incrementa la variabile "c" e mette il suo valore nella PORTA.

Il passo successivo

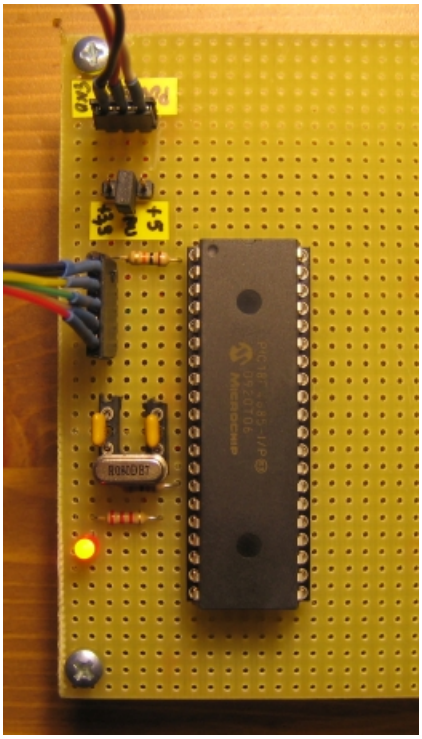
Da qui in poi bisogna studiare, leggersi attentamente il datasheet, il reference manual, la documentazione del compilatore, le librerie ed incominciare a sperimentare partendo dalle cose più semplici per aumentare man mano la difficoltà. Se la nostra intenzione è quella di sperimentare possiamo lasciar perdere, per ora, le funzionalità DSP e concentrarci sulle periferiche. Se invece le intenzioni sono quelle di utilizzare al meglio le funzionalita DSP ... probabilmente non state leggendo questo articolo! Sarebbe bene, quando si sperimentano le periferiche, andarsi a cercare le funzioni di libreria per la loro gestione o scriverle di proprio pugno seguendo le indicazioni del datasheet. E' un ottimo esercizio ed è anche utile, se fatto con metodo, per crearsi i propri moduli software con la prospettiva di utilizzarli per le future, eventuali applicazioni. A dire il vero la Microchip non è molto generosa in termini di documentazione, lo è per le applicazioni. E' possibile scaricare dal sito ufficiale un application framework che spazia in molti settori ed applicazioni. E' molto complesso da capire e necessita di lavoro e di attenzione ma può servire. Altra cosa utile è quella di analizzare gli esempi e cercare di implementarli. Gli esempi di utilizzo delle periferiche sono estremamente utili. Se siete agli inizi e non avete esperienza di microcontrollori vi suggerisco di evitare approcci tipo "vorrei realizzare un controllore PWM per un motore in continua, che rileva la temperatura tramite un ingresso analogico, che si collega al PC tramite linea seriale e mi faccia da data-logger". Prima magari proverei a far funzionare una linea seriale facendogli scrivere sul terminale "Prova di comunicazione" per poi provare a leggere una tensione qualsiasi, magari utilizzando un potenziometro, ed inviare il valore sulla linea seriale. Insomma, andare per gradi partendo dalla cosa più semplice con la sola volontà di imparare. Le applicazioni vere arriveranno dopo. Non è consigliabile cercare di bruciare le tappe utilizzando i micro. Questi signori hanno le spalle larghe, quindi occhio a non sbatterci il mento!

Complicarsi la vita

Con i microcontrollori è molto facile, con i DSP lo è ancora di più. Un ottimo sistema per farlo è quello di cercare di sfruttare al massimo il micro programmandolo in linguaggio assembly. Auguro ai temerari buona fortuna ma, per quanto mi riguarda, mi astengo dal farlo. Se il problema è la velocità il mio approccio è quello di scegliere un micro di categoria superiore, se il problema è la quantità di memoria scelgo un micro con più memoria. Ma sono scelte molto personali e non esiste una scelta "giusta". Altro modo per complicarsi la vita è quello di implementare su un DSP un programma da ... DSP. E qui entriamo nel campo dei filtri digitali, delle trasformate di Fourier e compagnia bella. Ci sarebbe tanto da sperimentare. A voi la scelta!

Conclusioni

La differenza fra i micro a 8 bit e quelle a 16 bit esiste, ed è innegabile sia in termini di prestazioni che di complessità. In effetti è più semplice implementare applicazioni in linguaggio assembly sugli 8 bit piuttosto che sui 16 bit. Fortunatamente i linguaggi ad alto livello semplificano la vita e permettono, come detto in precedenza, di svincolarci dalla struttura interna della macchina. Riuscire a sperimentare con macchine potenti è però più divertente, si possono fare molte cose in più. Non per questo gli 8 bit devono essere messi nel dimenticatoio anzi, e comunque sono da considerarsi un ottimo entry level per iniziare esperienze con i micrcocontrollori. Il modo di realizzare un sistemino analogo a quello presentato in questo articolo, utilizzante un micro a 8 bit è sostanzialmente identico. La foto ritrae la parte di circuito per la sperimentazione con un micro a 8 bit. Nello specifico vorrei sviluppare un' apparecchiatura che utilizza display ed integrati che funzionano a 5V.



PIC_8bit.jpg

Usero' quindi un micro che puo' funzionare a quella tensione per evitare circuiti di interfaccia. In questo caso non ho bisogno di grande velocità e di tanta memoria RAM e l' apparecchiatura sarà montata in un circuito stampato di tipo tradizionale, senza componenti SMD. Ho invece bisogno di molta memoria di programma ed il micro in questione (PIC18F4685) ha una ROM da 96KB mentre il 40 pin dsPIC30F4013 ha solo 48KB di ROM, insufficienti per la mia applicazione, altrimenti avrei optato sul dsPIC. L' alimentatore che utilizzo è quello illustrato in questo articolo. In alto

a sinistra trova posto il connettorino per l' alimentazione, a sinistra il connettore che si collega al debugger. Che dire? Le differenze sono minime, lo scheletro del firmware è identico, mancano solo le tre linee per la scelta della frequenza di clock perchè gli 8 bit non hanno circuiti versatili come quelli dei fratelli maggiori. Di solito parto sempre da un micro a 32bit dato che la differenza di costo non è poi così drammatica (il costo di un cappuccino e brioche) a meno di dover far fronte ad esigenze particolari (contenitore, tensione di alimentazione etc). Quindi non abbiate timore di utilizzare micro potenti perchè non mordono e non sono per niente difficili da utilizzare. Semmai sono più complicati, questo è vero, ma andando per gradi, con meticolosità ed attenzione le complicazioni si superano per lasciare posto a prestazioni esaltanti.

E con questo auguro buone sperimentazioni a tutti i lettori di ElectroPortal.

Download

http://www.electroportal.net/users/files/Main.c

Estratto da "http://www.electroyou.it/mediawiki/index.php?title=UsersPages:Tardofreak:pic33f"