8 Microcontrollori PIC

8.1 Descrizione generale

Un microcontrollore è un dispositivo elettronico che opportunamente programmato è in grado di svolgere diverse funzioni in modo autonomo. Essenzialmente gestisce delle linee di input e di output in relazione al programma in esso implementato.

Esistono diverse famiglie di dispositivi in grado di svolgere queste funzioni come ad esempio lo Z80, ST6 e il più evoluto 8088; per i nostri scopi ci si è affidati ai Microcontrollori PIC della Microchip® in quanto questi:

- risultano essere molto economici;
- sono di piccole dimensioni;
- hanno performance elevate in termini di velocità;
- sono facili da programmare.

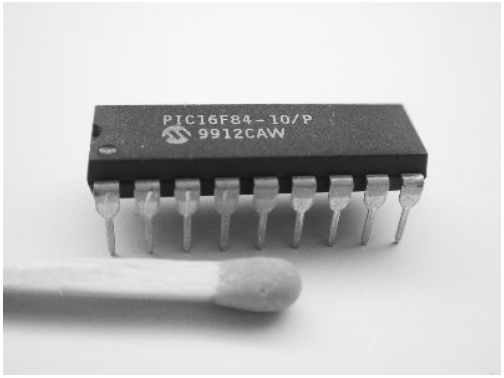
Questi dispositivi implementano al loro interno un vero e proprio microprocessore completo di CPU, ALU, RAM, Timer e numerose linee di IN/OUT. A differenza dei microprocessori più evoluti, nei PIC il programma è contenuto all'interno in un'apposita area di memoria (non volatile) e viene eseguito ciclicamente; la RAM viene utilizzata per i dati volatili (le variabili). Alcuni dispongono di aree dati non volatili e riscrivibili (EEPROM), generalmente di piccole dimensioni. Lo stadio di IN/OUT è già implementato ed alcuni sono dotati di interfacce analogico/digitali per comparatori o per comunicazioni seriali. I microcontrollori PIC sono dei dispositivi di tipo RISC, dispongono di un numero ridotto di istruzioni, e si programmano in Assembler (codice macchina).

Le versioni con memoria Flash sono programmabili più volte (e quindi idonei alla sperimentazione) al contrario delle versioni OTP (One Time Programmable) che si possono programmare solamente una volta (utilizzati per la realizzazione degli strumenti definitivi). Le versioni UV possono essere riprogrammati dopo la cancellatura con raggi ultravioletti.

I PIC sono disponibili in un’ampia gamma di modelli per meglio adattarsi alle esigenze di progetto specifiche, differenziandosi per numero di linee I/O e per dotazione di dispositivi. Si parte dai modelli più piccoli identificati dalla sigla **PIC12C5xx** dotati di soli 8 pin, fino ad arrivare ai modelli più grandi con sigla **PIC17Cxx** dotati di 40 pin.

8.2 Modello 16F84

Il **PIC16F84** da noi scelto è un caso intermedio (18 pin); si tratta di un microcontrollore ad 8 bit, ad alte prestazioni e basso costo, realizzato in tecnologia CMOS, il quale può essere programmato in linguaggio Assembler con un set di sole 35 istruzioni.



L'architettura Harvard consente di operare su dati a 8 bit e codificare le istruzioni a 14 bit, poiché il Bus Dati ed il Bus Istruzioni sono separati. Il sistema di pipeline a 2 stadi consente di eseguire tutte le istruzioni in un singolo ciclo[1], ad eccezione delle istruzioni di salto che ne richiedono 2.

Il PIC16F84 dispone di 68 byte di **RAM**, 64 byte di memoria dati di tipo **EEPROM** (Electrical Erasable Programmable Read Only Memory, ovvero memoria a sola lettura programmabile, cancellabile elettricamente) e 1024 word (2048 byte) di memoria di tipo **FLASH** nella quale è contenuto il programma in Assembler.

8.2.1 Altre caratteristiche tecniche

Frequenza del clock

Il microcontrollore può lavorare ad una frequenza massima di 10Mhz; inoltre la progettazione *fully-static* permette di utilizzare anche frequenze molto basse dell'ordine dell'unità di Hz.

Gli oscillatori utilizzabili si differenziano a seconda della loro velocità:

- **RC:** per soluzioni a basso costo (velocità minima)
- **LP:** per minimizzare la potenza consumata
- **XT:** per gli oscillatori al quarzo
- **HS:** per i cristalli ad elevata velocità.

Interrupt

Il microcontrollore 16F84 dispone di quattro sorgenti di interrupt:

- Pin INT esterno
- Overflow del TIMER0
- Cambio di stato sulla porta B<4..7>
- Fine ciclo di scrittura nella EEPROM dati

[1] **Ciclo**: per ciclo si intendono 4 fronti di clock

Microcontroller Power Requirement

- < 2 mA @ 5 V, 4 MHz
- 1,5 μA @ 2 V, 32 KHz

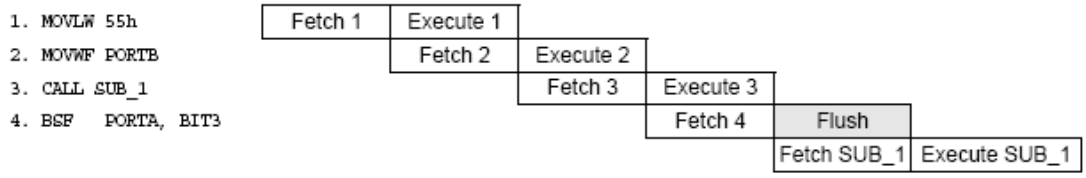
E' disponibile anche la modalità **SLEEP**, cioè si può far "addormentare" il dispositivo in attesa di un Interrupt, in modo da ottenere un consumo ridotto di corrente (si pensi ai casi in cui debba essere alimentato con delle batterie). In tal caso il consumo si riduce a meno di 1μA @ 2 V.

Livelli di tensione delle porte di I/O

Symbol	**Buffer**[2]	**V_{min}**	**V_{max}**
V_{IL}	TTL	Vss	0,16 Vdd
	ST	Vss	0,2 Vdd
V_{IH}	TTL	0,48 Vdd	Vdd
	ST	0,45 Vdd	Vdd
V_{OL}	-	-	0,6 Vdd
V_{OH}	-	Vdd – 0,7	-

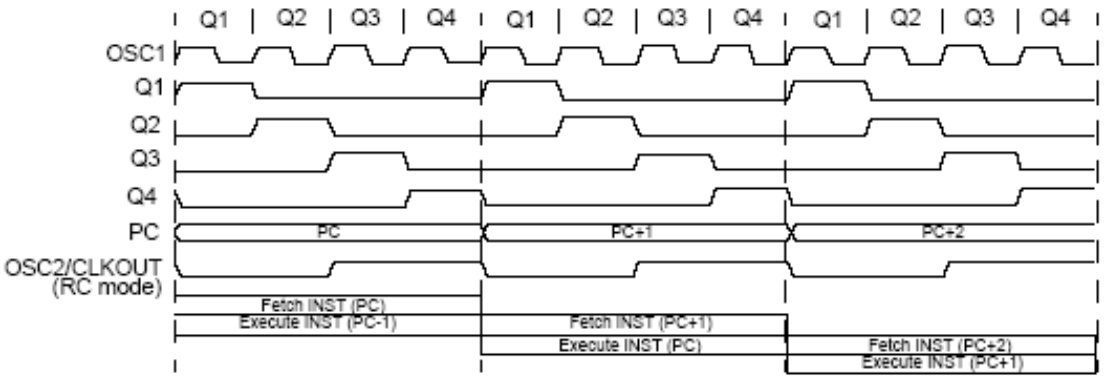
8.2.2 Architettura RISC

Il microcontrollore utilizza una "pipeline" costituita da sole due fasi: Fetch e Execute. Ciò che avviene ad ogni ciclo di istruzione e rappresentato in figura:



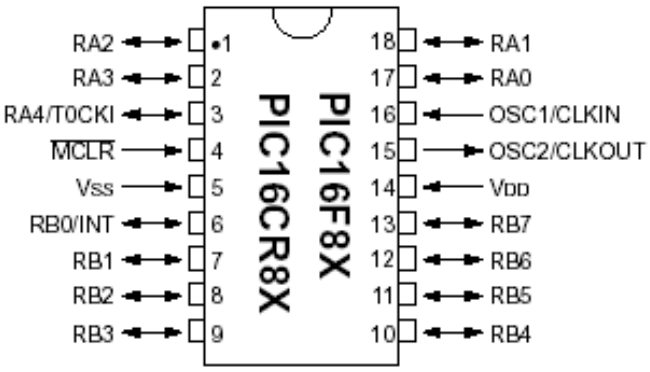
Ad ogni ciclo di istruzione (4 colpi di clock) viene effettuato il *fetch* dell'istruzione cui punta il Program Counter e contemporaneamente viene eseguita l'istruzione precedente. Le istruzioni di salto sono un po' più complesse e necessitano di un ciclo aggiuntivo (rappresentato dal blocco *Flush*).

[2] **ST**: input a trigger di Schmitt



Durante la fase di *execute*, l'istruzione viene caricata nell'Instruction Register durante il clock Q1, gli operandi vengono caricati in memoria durante Q2 e il risultato viene scritto durante Q4.

8.2.3 Piedinatura



Pin	**Buffer**	**Descrizione**
RA0…RA3	TTL	Porte I/O bidirezionale
RA4/TOCK1	ST	▪ Porta I/O bidirezionale ▪ Ingresso per il clock esterno del Timer0
RB0/INT	TTL ST	▪ Porta I/O bidirezionale ▪ Ricezione di un interrupt esterno
RB1…RB7	TTL/ST	Porte I/O bidirezionali : se opportunamente programmate le porte B4, B5, B6, B7 possono generare un interrupt hardware.
OSC1/CLKIN	ST CMOS	▪ Oscillatore ▪ Clock esterno

Pin	Buffer	Descrizione
OSC2/CLKOUT	---	▪ Oscillatore ▪ Se il PIC ha in ingresso un clock esterno su questo pin c'è un clock di frequenza ¼ rispetto a quello di ingresso
MCLR	ST	Reset, attivo basso, del microcontrollore

8.3 Programmazione del PIC

Per i nostri scopi programmare direttamente in linguaggio Assembler si rivelava essere piuttosto problematico; per questo in commercio esistono compilatori ad un livello più alto rispetto al codice macchina che permettono una programmazione più snella e comprensibile, anche in previsione di modifiche per eventuali sviluppi futuri. Si è deciso di utilizzare il **Proton+ Compiler** prodotto dalla Crownhill: si tratta sostanzialmente di un linguaggio molto simile al Basic, dedicato per lo sviluppo di programmi per microcontrollori, a basso costo (poco più di 10$) ed estremamente *user friendly*.

Per trasferire il programma all'interno del PIC si è utilizzato un programmatore universale della Advantech: il **LabTool-48**, con il software per il download fornito in dotazione. Si tratta di un dispositivo molto versatile che permette di programmare qualunque tipo di PIC oltre a EEPROM, PLD, Microcontrollori, etc..; si collega al PC tramite la porta parallela.

Figura 1: LabTool Advantec-48

Il trasferimento del programma dal LabTool al PIC è piuttosto veloce (nel nostro caso una ventina di secondi), ma dipende fortemente dal tipo di PIC utilizzato.