IL NOSTRO PRIMO PIC

In questo Workshop proseguiamo il percorso iniziato nel fascicolo precedente analizzando concretamente il primo dei microcontrollori che ci accompagnerà alla scoperta dei PICmicro: il 16F628.

Con il Workshop 55 abbiamo introdotto in modo teorico il ruolo dei microcontrollori, preannunciando l'uso della famiglia PIC a 8 bit prodotta da Microchip. Abbiamo anche parlato in modo molto generico delle performance e delle caratteristiche che accomunano la maggior parte dei dispositivi di questo tipo. Facciamo ora un passo avanti e analizziamo più nello specifico un vero microcontrollore: il 16F628.

FOCUS ON

IL 16F628: UNA SEMPLICE SCHEDA RIASSUNTIVA

In questo FOCUS ON ti presentiamo una scheda riassuntiva delle caratteristiche tecniche principali del microcontrollore PIC 16F628. Come avviene praticamente per tutti i dispositivi elettronici integrati, la documentazione completa è reperibile in forma digitale attraverso il sito internet dell'azienda produttrice, che in questo caso è www.microchip.com.

- **Numero totale pin: 18.**
- **Frequenza massima di lavoro: 20 MHz.**
- **Memoria istruzioni: 2048 istruzioni da 14 bit (c.ca 3.5 kbyte).**
- **RAM: 224 byte.**
- **EEPROM dati: 128 byte.**
- **Range di alimentazione: 3 - 5,5 V.**

FUNZIONI SPECIALI

- **Oscillatore interno per operare senza oscillatori esterni ausiliari.**
- **Supporto per oscillatori esterni di tipo R/C o a quarzo piezoelettrico.**
- **Supporto per la programmazione ICSP.**
- **Reset in caso di 'Brownout'.**
- **'Code protection' per impedire la lettura del firmware successivamente alla programmazione.**

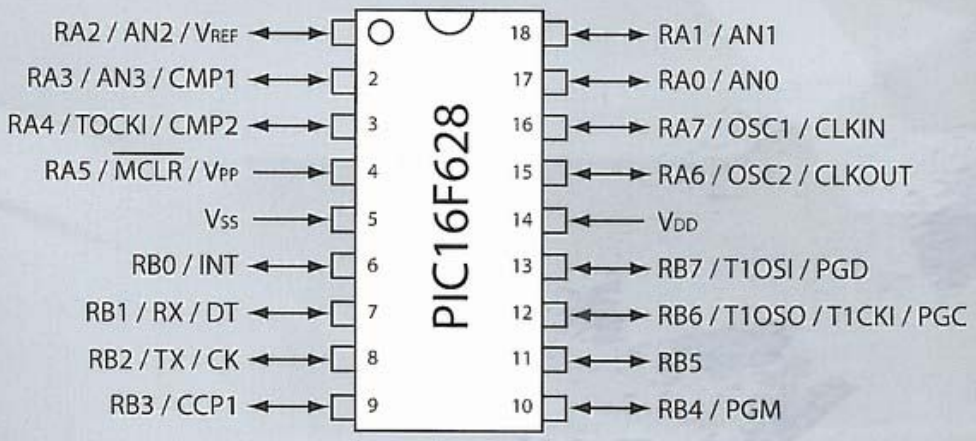
PERIFERICHE INTEGRATE

- **16 pin configurabili come I/O digitali (suddivisi in 2 porte da 8 bit).**
- **Due comparatori di tensione analogici.**
- **Tre timer integrati (uno a 16 bit, due a 8) per la temporizzazione degli eventi.**
- **Supporto per interrupt.**
- **Generatore PWM integrato a 10 bit di risoluzione.**
- **Supporto hardware per la comunicazione seriale USART.**

COS'È IL PIC 16F628 ›››

Il 16F628 è uno dei microcontrollori che più si prestano ai i nostri scopi didattici e che viene generalmente ritenuto uno degli strumenti più semplici per approcciarsi al mondo dei PIC. Storicamente il 16F628 è considerato l'evoluzione diretta del microcontrollore **16F84** (una delle MCU che negli anni passati si era maggiormente diffusa in ambito hobbistico), dal quale ha **ereditato la piedinatura**, fornendo, però, **una quantità maggiore di risorse e di periferiche interne.**

Il 16F628 è, infatti, in grado di memorizzare programmi che possono raggiungere le **2048 istruzioni da 14 bit** ciascuna (circa 3,5 kbyte di memoria di programma), contro la capienza massima di 1024 istruzioni del 16F84. Cresce, inoltre, la **memoria adibita ai dati** (RAM, che passa da 68 a **224 byte**) e la memoria **EEPROM**, che raddoppia i 64 byte originari, fornendo **128 byte** di dati archiviabili dal PIC. Nel FOCUS ON della pagina precedente puoi vedere una scheda tecnica riassuntiva nella quale sono poste in risalto le principali caratteristiche tecniche del PIC in questione. Per aiutarti a comprendere meglio il significato di alcuni dei termini racchiusi in esso, puoi far riferimento alle 'parole chiave' presenti nelle ultime due pagine del Workshop numero 55. In ogni caso, con gli esperimenti che avrai la possibilità di mettere in pratica nel corso dei prossimi fascicoli, sarà molto più semplice comprendere a fondo le caratteristiche elencate.



La piedinatura del 16F628. La maggior parte dei pin ha più di un nome, poiché può svolgere ruoli differenti in base alla configurazione del microcontrollore.

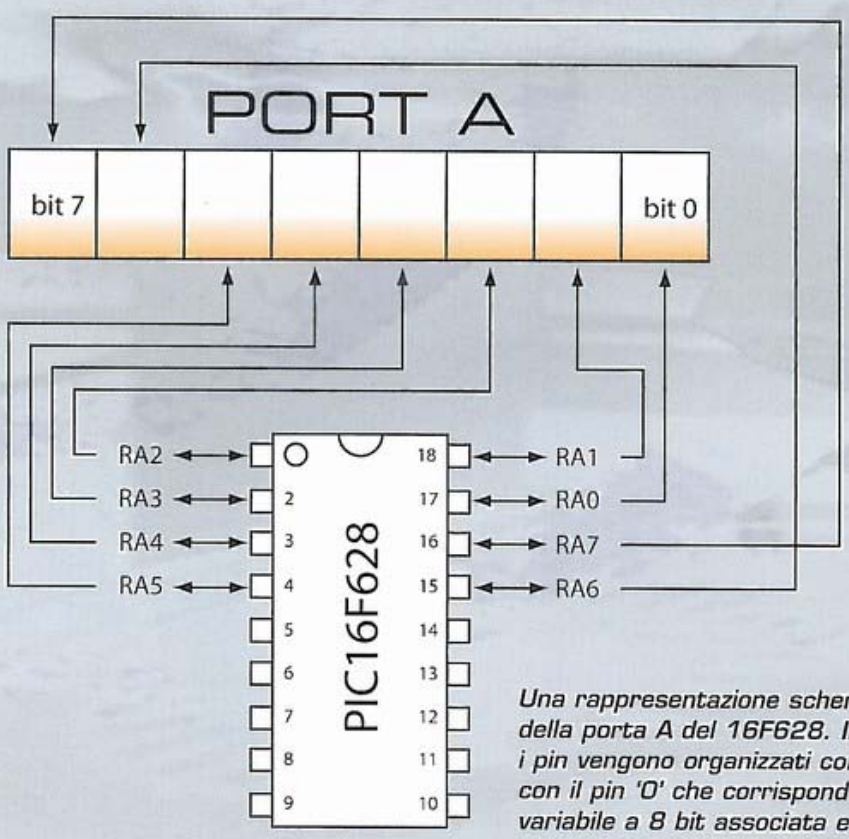
UNA PANORAMICA DEL MICROCONTROLLORE ›››

Nel paragrafo precedente abbiamo 'quantificato' le risorse di memoria messe a disposizione dal PIC. Abbiamo anche visto che il 16F628, come tutti i PIC dotati di architettura derivata Harvard, ha a disposizione due differenti tipi di memoria: **la memoria istruzioni o di programma** (generalmente di tipo 'flash', che ospita il firmware della MCU) e **la memoria dati** (di tipo RAM, in cui vengono posti i dati di elaborazione). La **EEPROM** può essere considerata a tutti gli effetti, invece, come un hard disk nel quale il PIC può immagazzinare in modo permanente e riscrivibile i dati di elaborazione da ricaricare all'avvio successivo. Come tale, quindi, **rientra nei dispositivi di I/O ausiliari e non è necessariamente presente in tutti i modelli di microcontrollori.** Diamo ora uno sguardo alla struttura tecnica del 16F628 partendo dal suo 'pinout'. Nello schema in alto puoi vedere mostrata la **piedinatura** del suddetto PIC. Come sempre, **la numerazione dei pin parte dall'alto a sinistra** (pin 1 a sinistra della 'tacca') e procede secondo il verso antiorario. Noterai, tuttavia, un'evidente differenza che rende questo schema 'anomalo' rispetto a tutti quelli visti finora: **la maggiore parte dei pin è identificata da più di un nome.** Questo dipende dal fatto che **la funzione dei singoli pin non è fissata a priori, ma viene configurata attraverso apposite sequenze di istruzioni che devono essere introdotte nel firmware.** Così, ad esempio, il **pin numero 17** (etichettato con la sigla **RA0/AN0**) può essere impiegato sia come **pin di I/O digitale**, sia come **pin di input analogico. Tutti i pin che includono nel nome la dicitura ANx, infatti, sono utilizzabili come ingressi analogici**, per mezzo dei

convertitori analogico/digitali presenti all'interno dei PIC. Questi piedini possono quindi essere sfruttati per **acquisire dati analogici relativi a valori di tensione** per rilevare, ad esempio, l'output di un sensore. Come detto in precedenza, però, il pin 17 ha anche una modalità **funzionamento alternativa, di tipo digitale**. Più in generale, tutti i pin che riportano un nome del tipo **RXn** (con **'X'** lettera e **'n'** numerico) vanno interpretati come **'n-esimo pin di I/O della porta digitale X'**. Per quanto appena detto, di conseguenza, la sigla **RA0** deve essere letta come **'pin di I/O digitale numero 0 della porta A'**. Il significato del concetto 'porta' era già stato anticipato all'interno dell'elenco incluso nel Workshop precedente, tuttavia proviamo ad approfondirlo ulteriormente. Come puoi osservare dalla piedinatura del 16F628, se escludiamo i pin 5 (VSS) e 14 (VDD), tutti gli altri piedini, che includono nel nome una sigla RXn, sono suddividibili nei due gruppi distinti **RA** (da RA0 a RA7) e **RB** (da RB0 a RB7). Proprio tali gruppi formano le due porte digitali a 8 bit del microcontrollore (chiamate semplicemente PORTA e PORTB). Come sperimenteremo più avanti, ogni porta è associata a un ben preciso registro di 8 bit, che tratteremo in modo praticamente analogo alle comuni variabili del linguaggio C. L'importante è ricordare che la numerazione dei pin della porta è concorde alle convenzioni con cui si scrivono i numeri binari: **il pin '0' è associato al bit meno significativo del registro, mentre il pin '7' a quello più significativo** (vedi schema in fondo alla pagina). Ovviamente, nel caso in cui un pin supporti più di una modalità di funzionamento, è indispensabile impostare quella prescelta in fase di scrittura del firmware del microcontrollore. Non è possibile, infatti, che un pin svolga più ruoli differenti allo stesso tempo poiché si genererebbe un conflitto funzionale (non si può fare in modo, ad esempio, che un piedino abbia funzione sia di ingresso digitale, sia di ingresso analogico di un comparatore).

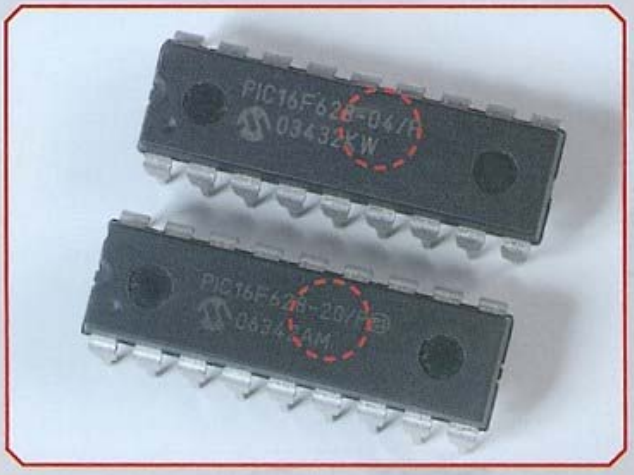
LE MODALITÀ DI FUNZIONAMENTO DELL'OSCILLATORE

Sospendiamo, per ora, l'analisi a livello di piedinatura del 16F628 e iniziamo a gettare le basi che ci condurranno alla sua programmazione vera e propria. Nel fascicolo precedente, nel paragrafo in cui abbiamo comparato i microcontrollori e i personal computer, abbiamo utilizzato come termine di comparazione per le prestazioni **la frequenza di lavoro** del dispositivo. Questo parametro, in termini più 'tecnici', altro non è che **la frequenza del 'clock', ossia dell'onda quadra di temporizzazione, che 'scandisce' il lavoro dei circuiti sequenziali interni**



Una rappresentazione schematica dell'organizzazione della porta A del 16F628. In modalità di I/O digitale, i pin vengono organizzati come una variabile di 8 bit con il pin '0' che corrisponde al bit meno significativo della variabile a 8 bit associata e il pin 7 a quello più significativo.

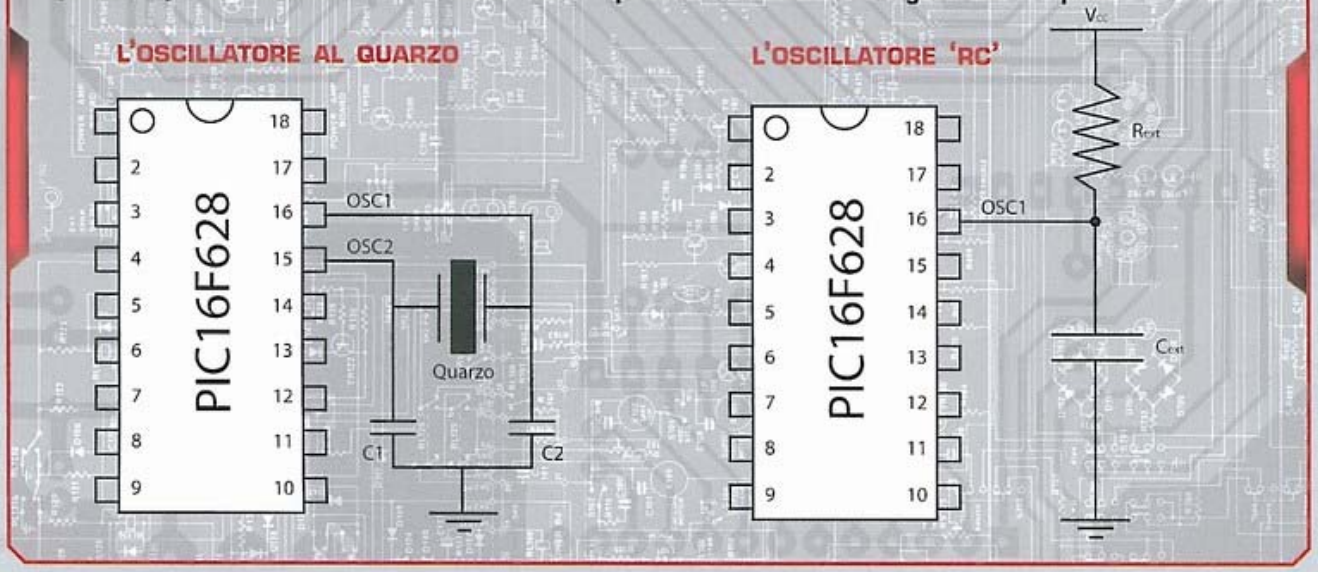
al microprocessore. La frequenza di clock non è altro che il numero di 'step' eseguiti in un singolo secondo dai circuiti di elaborazione del processore, step che però non corrispondono necessariamente a singole operazioni di programma. Per avere un esempio concreto, nei PIC ogni singola istruzione assembly richiede generalmente 4 colpi di clock per essere eseguita, con la conseguenza che se un microcontrollore di questo tipo lavora a 20 MHz (ossia 20 milioni di step al secondo), può eseguire un massimo di 5 milioni di istruzioni per unità di tempo. È quindi sbagliato affermare in termini assoluti che tra due microprocessori con architetture interne diverse il più 'performante' è quello con la frequenza di lavoro maggiore; è invece vero che aumentando la frequenza di clock di un microprocessore aumentano le sue prestazioni, salvo il raggiungimento di determinati limiti fisici legati all'elettronica del dispositivo, oltre i quali non è possibile andare senza incorrere in malfunzionamenti o 'bruciare' irreparabilmente l'integrato. Per il funzionamento interno del 16F628 (ma ciò vale per tutti i microcontrollori e i microprocessori) è quindi indispensabile una sorgente di clock. Come ottenerla? Generalmente i PIC offrono diverse possibilità di generazione dell'onda quadra di clock. Il 16F628, in particolare, può operare in 8 distinte modalità, chiamate rispettivamente: LP (Low Power Crystal), XT (Crystal), HS (High Speed Crystal),



Due modelli di PIC 16F628 a confronto. I due numeri alla fine della sigla (evidenziati in rosso) indicano la frequenza massima di funzionamento del microcontrollore.

IL FUNZIONAMENTO DELL'OSCILLATORE AL QUARZO E CON RESISTORE ESTERNO

Il PIC 16F628 (ma questo vale praticamente per tutti i tipi di PIC) può ottenere l'onda di clock per la temporizzazione del microcontrollore in 8 modalità differenti di funzionamento. Alcune di queste (LP, XT, HS, RC) richiedono l'utilizzo di componenti esterni da collegare ai pin dell'integrato. Sotto puoi vedere come collegare il quarzo per le modalità LP, XT e HS (a sinistra) e il resistore (a destra) nella modalità RC. La scelta dei componenti da utilizzare è legata alla frequenza di clock.



DIMENSIONARE I COMPONENTI PER L'OSCILLATORE AL QUARZO

Il microcontrollore 16F628 può sfruttare diverse tipologie di oscillatori per la generazione dell'onda quadra di clock. Tra le modalità in cui può operare, tre (LP, XT, HS) richiedono l'impiego di un particolare elemento 'risonante', il quarzo. Nel Workshop non ci addentreremo nel funzionamento fisico degli oscillatori al quarzo, tuttavia per poter utilizzare correttamente i microcontrollori è indispensabile accoppiare i quarzi a condensatori aventi valori appropriati. I datasheet forniscono dei range ottimali dei valori di capacità da adoperare in funzione della frequenza prodotta dal quarzo che si vuole impiegare (i quarzi, infatti, vengono scelti per imporre una frequenza di funzionamento al microcontrollore).

MODALITÀ	FREQUENZA QUARZO	OSC1 (C1)	OSC2 (C2)
LP	32 kHz	15 - 30 pF	15 - 30 pF
	200 kHz	0 - 15 pF	0 - 15 pF
XT	100 kHz	68 - 150 pF	150 - 200 pF
	2 MHz	15 - 30 pF	15 - 30 pF
	4 MHz	15 - 30 pF	15 - 30 pF
HS	8 MHz	15 - 30 pF	15 - 30 pF
	10 MHz	15 - 30 pF	15 - 30 pF
	20 MHz	15 - 30 pF	15 - 30 pF

RC (External Resistor, in due modalità), **INTOSC** (Internal Oscillator, in due modalità) ed **EC** (External Clock). Le prime tre (**LP**, **XT** e **HS**) permettono di attivare un **oscillatore al quarzo**, per il quale sono richiesti **tre componenti ausiliari esterni: un quarzo piezoelettrico** (che determina la frequenza di lavoro) e **una coppia di condensatori**, posti in concomitanza dei pin 15 e 16 (modo di funzionamento OSC2 e OSC1, come mostrato nello schema a sinistra del box in fondo alla pagina precedente). In questa modalità i pin 15 e 16 non possono, ovviamente, essere impiegati come piedini di I/O della porta A. La modalità **RC** permette di sfruttare **un circuito interno al PIC accoppiandolo a un ramo R/C esterno (collegato tra il pin 16 e la massa)** che regola la temporizzazione dell'oscillatore. In questo caso, **la frequenza operativa del PIC dipende dal valore del resistore utilizzato** (i datasheet del 16F628 forniscono una tabella riassuntiva di alcuni valori tipici). La modalità **EC**, invece, prevede semplicemente **l'utilizzo di un clock esterno.** In pratica, **l'onda quadra viene generata da un circuito esterno al PIC e portata in ingresso tramite il pin 16.** In ultimo, il modo di funzionamento **INTOSC** permette di **utilizzare come generatore di clock un circuito R/C interno all'integrato (l'oscillatore può operare a 4 MHz o 37 kHz, in base alla configurazione impostata).** Il vantaggio dell'utilizzo dell'oscillatore interno, quindi, è che **non sono richiesti componenti esterni al microcontrollore**, con la conseguenza che **tutti i pin del PIC sono a disposizione del progettista.** Lo svantaggio, tuttavia, è che **l'oscillatore INTOSC interno ha una precisione fortemente inferiore rispetto alle temporizzazioni a quarzo.** Il suo utilizzo, quindi, è sconsigliabile nelle applicazioni fortemente sensibili a eventuali anomalie del clock. Ma come selezionare quale modalità di funzionamento utilizzare? La configurazione dell'oscillatore avviene durante la fase di programmazione, impostando alcuni bit dedicati tra i 'fuse' del PIC. Non appena avremo basi sufficienti a realizzare il primo semplice esperimento con il 16F628, potrai sperimentare di persona come configurare l'oscillatore ma, soprattutto, potrai applicare quanto hai potuto vedere nel corso dei Workshop di programmazione alla realizzazione di alcuni semplici software embedded.

FOCUS ON

I PROGRAMMATORI: HARDWARE E SOFTWARE

La programmazione dei microcontrollori avviene utilizzando appositi strumenti hardware chiamati 'programmatori'.
I programmatori non sono altro che schede più o meno complesse equipaggiate con uno o più socket, nei quali vengono inseriti gli integrati su cui si vuole copiare il firmware.
Esiste una grande varietà di modelli di programmatori (molti dei quali costruibili anche artigianalmente a partire da schemi elettronici pubblici facilmente reperibili tramite internet), che si differenziano per complessità, microcontrollori supportati, tipi di connessioni utilizzate e necessità o meno di alimentazione ausiliaria esterna.
I programmatori JDM, ad esempio, sono tra i più semplici da realizzare, vista la loro struttura elementare. Di contro, poiché i computer moderni sono spesso privi di porta seriale (indispensabile per la connessione con i JDM) è necessario ricorrere ad adattatori USB, che però nella maggior parte dei casi non consentono al JDM di funzionare correttamente.
In alternativa si può, comunque, ricorrere a programmatori paralleli (ossia collegabili alla porta parallela, come i ProPic2, foto sopra e i relativi 'cloni') o USB, come i PicKit2, prodotti dalla stessa Microchip.
Una volta deciso il programmatore hardware, è necessario scegliere il software che si occuperà di trasferire il firmware.
Come per l'hardware, la scelta del programma non è univoca: sono, infatti, numerosi i software reperibili in rete (molti dei quali gratuiti e liberamente scaricabili) adibiti a questo scopo. Una delle applicazioni storicamente più note è IC-Prog (prima immagine a sinistra in alto), ormai giunto alla versione 1.06B. Molto diffuso, anche grazie alla grande quantità di dispositivi supportati, è WinPic800 (immagine a sinistra in basso), progettato per funzionare con l'apposito programmatore commerciale GTP-USB, ma utilizzabile tranquillamente anche con la maggior parte dei programmatori seriali e paralleli più diffusi.
La scelta della tecnologia, quindi, è solo del progettista, che può valutare in base alle necessità tecniche e in base al 'gusto' personale. Nel corso dei prossimi Workshop, avrai la possibilità di sperimentare di persona le procedure di programmazione utilizzando questi sistemi software e hardware.