I MICROCONTROLLORI PIC (Microchip) PARTE I

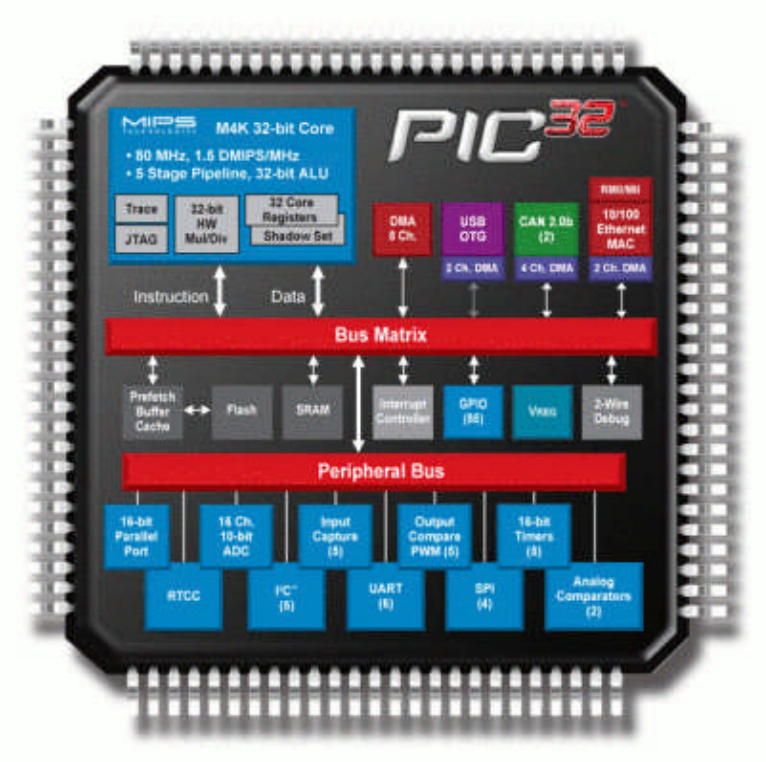
Prof. Angelo Monfroglio

Questa dispensa, ad uso degli studenti degli indirizzi Elettronica e Informatica e di tutti i progettisti e studiosi interessati, fornisce le nozioni essenziali sulla famiglia di microcontrollori PIC (acronimo per Programmable Integrated Circuit, oppure Programmable Interface Controller, o ancora Programmable Intelligent Computer come era detto dalla prima azienda costruttrice la General Instrument, nel 1975) della ditta Microchip. Come dice il prof. Sergio Conti in questo stesso sito, sul Web esiste una documentazione immensa sui PIC, sia della Micrologic, sia da parte di docenti e vari utilizzatori. Tuttavia, la quantità di pagine (decine di migliaia) e dettagli anche inutili, rende la massa di informazioni praticamente ingestibile. Ecco la motivazione di questa breve dispensa di impostazione pratica e sintetica. PIC è marchio registrato di Microchip Technology Inc.

Introduzione

I microcontrollori sono microprocessori specializzati nelle applicazioni di controllo elettronico e sono presenti in tutti gli elettrodomestici e le apparecchiature moderne. A differenza dei microprocessori di impiego generale, hanno al loro interno tutto quello che serve all'interfacciamento digitale ed analogico, cioè, ad esempio convertitori ADC e DAC, comparatori, interfacce RS232 e USB. La famiglia dei PIC della ditta Microchip (www.microchip.com) è la più completa e avanzata oggi disponibile. L'ITIS Omar è l'unica scuola nella nostra zona che insegna agli studenti tutte le fasi di lavorazione, dal progetto alla realizzazione e collaudo finali di applicazioni coi PIC. Comunque, uno studente che si impadronisce delle tecniche hardware e software sui PIC è in grado di operare su qualsiasi altro microcontrollore.

Nel 2007 la Microchip ha introdotto la famiglia di PIC32 a 32 bit che rappresenta lo stato dell'arte più avanzato di microcontrollori, con gestione integrata di Ethernet 10/100 Mbps. La famiglia più nota, il PIC16 sta per essere abbandonata e sostituita dai PIC18.



Gli ultimi nati: i PIC32

Verranno descritti in dettaglio i microcontrollori PIC16F876A, PIC18F2550, PIC18F4520, PIC16F877A. Verranno inoltre presentate (nella seconda parte) applicazioni in Assembler, Basic e C effettivamente provate in ambiente di sviluppo MPLAB8.53, MCC18 e PICSimulator.

I microcontrollori PIC adottano l'architettura Harvard, proposta da Howard Aiken per lo sviluppo dei computer Mark I, II, III e IV, appunto all'università di Harvard: usa memorie differenti per depositare dati e istruzioni. Al contrario, l'architettura Von Neumann, proposta per sviluppare l'ENIAC (Electronic Numerical Integrator And Calculator) all'università della Pennsylvania durante la seconda guerra mondiale, usa la stessa memoria per dati e programmi. Questa architettura usa meno linee ed è più economica, ma non sfrutta il parallelismo ed è quindi meno efficiente e veloce.

I PIC sono microcontrollori RISC (Reduced Instruction Set Computer): hanno un set di poche istruzioni, fra 33 e 77, e lunghe fra 12 e 16 bit. Altri microcontrollori sono invece CISC (Complex …). Ci sono vantaggi e svantaggi per i microprocessori RISC e CISC di uso generale (CPU per Personal Computer, ecc.). Tuttavia, si ritiene comunemente che i microcontrollori RISC siano più efficienti e veloci, anche se la programmazione è un po' più difficoltosa, pur se le istruzioni sono più semplici. I PIC, come tutti i moderni microcontrollori, dispongono al loro interno di watchdog (cane da guardia) che provvede a un reset automatico quando un contatore interno di guardia (indipendente dal Program Counter) raggiunge la fine. Se un programma funziona correttamente impedisce al contatore di raggiungere il massimo, azzerandolo periodicamente; se invece il programma va in stallo oppure non viene attivato per un certo tempo, il watchdog attiva il reset. Attualmente un microcontrollore senza watchdog non è più proponibile per applicazioni industriali. Anche la presenza di convertitori A/D è praticamente indispensabile.

Nei PIC18 tutte le istruzioni non di salto sono eseguite in un ciclo macchina, pari a 4 periodi di clock; quelle di salto in 2. Se quindi il clock è a 4 MHz un'istruzione dura 1 microsecondo, se il clock è 40 MHz si eseguono ben 10 milioni di istruzioni al secondo.

I PIC sfruttano la tecnica moderna del pipelining (letteralmente oleodotto, o catena di montaggio): mentre il micro controllore interpreta un'istruzione (fase di fetch = interpretazione), contemporaneamente ne esegue un'altra che era stata prima interpretata. In questo modo, sfruttando il parallelismo, è possibile incrementare il numero di istruzioni eseguite a parità di clock.

L'oscillatore che sincronizza tutte le operazioni, cioè il clock, può essere un circuito RC (si collegano ai 2 piedini una resistenza e un condensatore) o meglio un circuito al quarzo (XTAL e due condensatori dell'ordine dei pico Farad) più stabile e preciso, o, infine, un clock esterno.

Il PIC memorizza in memoria interna non volatile EEPROM alcuni bit di configurazione: tipo di oscillatore, Watchdog attivato o disattivato, protezione memoria programma e dati, specifiche per il reset e l'alimentazione. I PIC possono essere collocati in modalità dormiente (sleep mode) con risparmio energetico, ed essere risvegliati all'occorrenza.

I PIC sono caratterizzati anche dall'uso distinto della memoria Dati (RAM volatile e/o EEPROM) e memoria programma (per lo più FLASH).

Famiglie di PIC

I PIC si possono collocare in 3 famiglie: PIC a basso, medio e alto livello a 8 bit (il livello è dato dal numero di istruzioni); PIC 24 a 16 bit e PIC32 a 32 bit. Purtroppo, le sigle 16, 18, ecc. non ci dicono né quanti piedini il chip ha, né quali sono le caratteristiche, né se è a 8, 16 (PIC24) o 32 bit (PIC32), neppure il livello di complessità, il numero di porte e dispositivi come ADC, seriali, ecc., e il numero di istruzioni. I PIC a 8 bit trattano direttamente dati lunghi 8 bit. Quelli a 16 dati a 16 bit, ecc.

Vediamo ora qualcosa di più sulle sotto famiglie.

PIC a basso livello

Hanno 33 istruzioni lunghe 12 bit ciascuna. La memoria di programma arriva fino a 2 K parole di 12 bit. I piedini sono 6, 8, 10, 20, 28. Le sigle PIC16X5xx, PIC12X5xx, PIC10Fxxx. I PIC16X5xx possono usare come memoria programmi EPROM o FLASH. I PIC12X5xx hanno anche un convertitore A/D.

PIC a medio livello

Hanno 35 istruzioni ognuna lunga 14 bit. La memoria programmi arriva a 8 K, organizzata in pagine di 2 K parole. La memoria dati è organizzata in banchi di 120 registri a 8 bit. Gestiscono interrupt interni fissi e un interrupt esterno. Le sigle sono PIC16 (eccetto PIC16X5xx che abbiamo già visto) e PIC12X6xx che usano solo 8 piedini. Hanno diverse porte (A, B, C,ecc.), fino a 3 timer, porte seriali, parecchi convertitori A/D, comparatori e altre interfacce.

Il Program Counter (PC) ha 13 bit e quindi può indirizzare fino a 8k word di memoria Programma. Poiché la memoria è suddivisa in pagine di 2k, il bit 12 e il bit 11 del PC indicano una delle 4 pagine, mentre i bit da 0 a 10 danno l'indirizzo all'interno della pagina.

Pagina 0 Pagina 1 Pagina 2 Pagina 3

0h	800h	1000h	1800h
…..	….	….	….
7FFh	FFFh	17FFh	1FFFh

Il PIC16F876A

Il PIC16F876A ha 28 piedini, di cui alcuni multifunzione:

1	MCLR	Vpp	Reset (attivo basso) – ingresso tensione programmazione
2	RA0	AN0	I/O digitale PortA-Ingresso analogico 0 A/D
3	RA1	AN1	I/O digitale PortA –Ingresso analogico 1 A/D
4	RA2	AN2, Vref-, CVref	idem + tensione riferimento A/D e uscita comparatore
5	RA3	AN3, Vref+	
6	RA4	T0CKI, C1out	idem + ngress clock Timer 0, uscita comparatore 1 (open drain)
7	RA5	AN4, C2out	idem + uscita comparatore 2
8	VSS		GND (massa)
9	OSC1	CLKIN	Ingresso oscillatore a cristallo o clock esterno
10	OSC2	CLKOUT	Uscita oscillatore a cristallo o clock esterno
11	RC0	T1CKI, T1OSO	I/O digitale 0 PortC, ingresso clock Timer 1, uscita osc. Timer1
12	RC1	CCP2, T1OSI	I/O digitale PortC, Input Capture 2, Input osc. TMR1
13	RC2	CCP1	I/O ditale PortC, Input Capture 1
14	RC3	SCL	I/O digitale PortC, Seriale CL
15	RC4	SDI	I/O digitale Portc, seriale Dati input
16	RC5	SDO	I/O digitale PortC, seriale dati output
17	RC6	TX	I/O digitale PortC, trasmissione seriale
18	RC7	RX	I/O digitale PortC, ricezione seriale
19	VSS		GND
20	VDD		+ di alimentazione
21	RB0	INT	I/O digitale PortB, ingresso interrupt esterno
22	RB1		I/O digitale PortB

23	RB2		I/O digitale PortB
24	RB3	PGM	I/O digitale PortB, ingresso programmazione
25	RB4		I/O digitale PortB
26	RB5		I/O digitale PortB
27	RB6	PGC	I/O digitale PortB, clock programmazione seriale
28	RB7	GND	I/O digitale PortB, dati programmazione seriale

Le periferiche integrate

5 canali A/D a 10 bit

Comparatori analogici con soglia programmabile

Timer

Capture, compare

PWM (Pulse Width Modulation) per il commando di motori

Porta USART (seriale sincrona e asincrona)

Organizzazione della memoria del PIC16F876A (e PIC16F877A che ha 2 porte in più)

Sono di fatto 3 le memorie da considerare

Memoria programma (es. FLASH)	Memoria dati non volatile (EEPROM)	Memoria volatile RAM
Da 0000h in su	Da 00h in su	Da 000h in su
STACK (pila) livello 1 STACK livello 2 ….. STACK livello 8 VETTORE Reset VETTORE Interrupt Programma utente		FILE REGISTER: SFR (Special Function Registers) RAM uso generale: GPR

Ci sono 4 banchi di memoria, 192 General Purpose Register (GPR) e 50 Special Function Register (SFR) nel PIC16F877A (40 piedini).

Gli SFR si possono raggruppare in:

STATUS	Seleziona i banchi; contiene Flag
OPTION	Prescaler; fronte impulsi; Pull-up per portB
PCL, PCLATCH	Program Counter
FSR	Indirizzamento indiretto
INTCON, PIR1, PIE1, PIR2, PIE2	Gestione interrupt
PORTA, PORTB, PORTC,(PORTD, PORTE)	Porte parallele
TRISA, TRISB,TRISC,(TRISD,TRISE)	Selezione Input/Output porte parallele
TMR0, OPTION, INTCON	Gestione Timer0
TMR1H, TMR1L, T1CON, PIR1	Gestione Timer1
TMR2, PR2, T2CON, PIR2	Gestione Timer2
CCPRxH, CCPRxL, CCPxCON	Gestione moduli (x=1,2,3) compare / capture / PWM
TXREG, TXSTA, RCREG, RCSTA	Gestione seriale asincrona
SSPSTAT, SSPCON, SSPBUF, SSPADD	Gestione seriale sincrona
ADRESH, ADRESL, ADCON0, ADCON1	Gestione convertitore A/D
EEADRH, EEADR, EEDATH, EEDATA	Gestione memoria dati EEPROM e memoria FLASH
EECON1, EECON2	idem

I banchi di memoria: quadro sintetico (PICF877A, 40 piedini). Il PIC16F876A (28 piedini) ha solo le Porte A,B,C.

Banco 0		Banco 1		Banco 2		Banco 3	
00h	INDF	80h	INDF	100h	INDF	180h	INDF
	TMR0						
	PCL		PCL		PCL		PCL
	STATUS		STATUS		STATUS		STATUS

	FSR		FSR		FSR		FSR
	PORTA		TRISA				
	PORTB		TRISB		PORTB		TRISB
	PORTC		TRISC				
	PORTD		TRISD				
	PORTE		TRISE				
	PCLATH		PCLATH		PCLATH		PCLATH
	INTCON		INTCON		INTCON		INTCON
	PIR1		PIE1		EEDATA		EECON1
	PIR2		PIE2		EEADR		EECON2
	TMR1L		PCON		EEDATH		
	TMR1H				EEDARH		
	T1CON						
	TMR2		SSCON2				
	T2CON		PR2				
	SSPBUF		SSPADD				
	SSPCON		SSPADD				
	CCPR1L						
	CCPR1H						
	CCP1CON						
	RCSTA						
	TXREG						
	RCREG						
	CCPR2L						
	CCPR2H						
	CCP2CON						
	ADRESH		ADRESL				

	ADCON0		ADCON1				
	96 GPR	A0h	96 GPR		GPR mappati in banco 0		GPR mappati in banco 1
		FFh		17Fh		1FFh	

Banco0 NOME	Indirizzo	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
INDF	00h	Se usato, FSR punta a memoria dati	idem						
TMR0	01h	TIMER 0	idem						
STATUS (FLAGs)	03h	IRP	RP1	RP0	TO	PD	Z	DC	C
FSR	04h	Puntatore indir. indiretto							
PORTA	05h	-	-	RA5	RA4	RA3	RA2	RA1	RA0
PORTB	06h	RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0
PORTC	07h	RC7	RC6	RC5	RC4	RC3	RC2	RC1	RC0
INTCON	0Bh	GIE	PEIE	TMR0IE	INTIE	RBIE	MR0IF	INTF	RBIF
PIR1	0Ch	PSPIF	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	MR1IF
PIR2	0Dh	-	CMIF	-	EEIF	BCLIF	-	-	CCP2IF
TMR1L	0Eh	LSB Timer0							
TMR1H	0Fh	MSB							
T1CON	10h	-	-	T1CKPS1	T1CKPS0	T1OSCEN	T1SYNC	TMR1CS	TMR1ON
TMR2	11h	Timer2							
T2CON	12h	-	TOUTPS3	TOUTPS2	TOUTPS1	TOUTPS0	TMR2ON	T2CKPS1	T2CKPS0
CCPR1L	15h	LSB modulo 1 capture/compare/PWM							
CCPR1h	16h								
CCP1CON	17h	-	-	CCP1X	CCP1Y	CCP1M3	CCP1M2	CCP1M1	CCP1M0
CCPR2L	1Bh	LSB modulo 2							
CCPR2H	1Ch	MSB modulo 2							
CCP2CON	1Dh	-	-	CCP2X	CCP2Y	CCP2M3	CCP2M2	CCP2M1	CCP2M0
ADRESH	1Eh	MSM risultato convertitore A/D							

ADCON0	1Fh	ADSC1	ASCS0	CHS0	CHS1	CHS0	CO/DONE	-	ADON
Banco 1									
OPTION	81h	RBPU	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0
TRISA	85h	-	-	I/O Porta A					
TRISB	86h	I/O Porta B							
TRISC	87h	I/O Porta C							
PIE1	8Ch	-	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMT1IE
PIE2	8Dh	-	CMIE	-	EEIE	BCLIE	-	-	CCP2IE
PCON	8Eh							POR	BOR
PR2	92h	Periodo Timer 2							
CMCON	9Ch	C2OUT	C1OUT	C2INV	C1NV	CIS	CM2	CM1	CM0
CVRCON	9Dh	CVREN	CVROE	CVRR	-	CVR3	CVR2	CVR1	CVR0
ADRESL	9Eh	LSB risultato A/D							
ADCON1	8Fh	ADFM	-	-	-	PCFG3	PCFG2	PCFG1	PCFG0
Banco 2									
EEDATA	10Ch	Registro Dati EEPROM							
EEADR	10Dh	Registro Indirizzi EEPROM							
EEDATH	10Eh	-	- Parte alta registro dati EEPROM						

EEADRH	10Fh	-	-	-	Parte alta registro indirizzi EEPROM				
Banco 3									
EECON1	18Ch	EEPGD	-	-	-	WRERR	WREN	WR	RD

Nota: Quelli sopra riportati sono i registri SFR del PIC16F876A (28 piedini)

Registro dei Flag (STATUS): significato

IRP	indirizzamento indiretto: 0 significa banchi 0 e 1, 1 banchi 2 e 3
RP1, RP0	seleziona un banco fra 0 (==) e 3 (11)
TO	0 quando il Watchdog va in Time Out
PD	0 quando il micro controller va in Sleep
Z	Flag di Zero (1 se zero)
DC	Digital Carry: 1 se c'è riporta dal bit 3 (quarto bit) al 4 (quinto bit)
C	Carry (riporto): 1 se c'è riporto dal bit più significativo

Registro opzioni (OPTION): significato

RBPU	disabilita o abilita i resistori di pull-up sulla porta B
INTEDG	selezione il fronte (1=salita/0=discesa) del segnale di interrupt su RB0/INT
T0CS	selezione la sorgente per il clock del Timer 0
T0SE	fronte segnale clock del Timer su RA4/T0CKI
PSA	prescaler per Timer 0 o Watchdog (WDT)
PS2	prescaler (fattore di divisione) di cui sopra
PS1	idem
PS0	idem

INTCON: significato

GIE abilitazione generale di tutti gli interrupt

PEIE abilitazione interrupt periferiche

T0IE abilitazione dell'interrupt per il Timer 0

INTE abilita interrupt linea RB0/INT

RBIE abilita l'interrupt per il cambio di livello sulle linee RB7-RB4

T0IF segnalazione raggiungimento overflow Timer 0

INTF segnalazione richiesta di interrupt sulla linea RB0/INT

RBIF segnalazione di cambio livello sulle linee RB/-RB4

Il convertitore A/D: funzionamento e registri

Il PIC16F876A ha 5 canali di conversione A/D a 10 bit. Il risultato della conversione è posto in ADRESL e ADRESH . I registri di controllo sono ADCON0 (banco 0, indirizzo 1Fh) e ADCON1 (banco 1, 9F). Il convertitore A/D è ad approssimazioni successive e ha anche il Sample-Hold (o Track-Hold). Con il registro ADCON1 si seleziona:

. l'abbinamento con I piedini RA0, ecc.

-il tipo di tensione di riferimento (ad esempio VDD)

-la modalità di salvataggio del risultato (nei registri ADRESL e ADRESH).

Vediamo i loro bit:

ADCON1

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
ADFM	-	-	-	PCFG3	PCFG2	PCFG1	PCFG0

ADCON0

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
ADCS1	ADCS0	CHS2	CHS1	CHS0	GO/DONE	-	ADON

I Timer

TIMER0: è a 8 bit¸si può usare con clock esterno su RA4, o interno (frequenza oscillatore / 4) ed abbinare un prescaler a 8 bit. Se TMR0 passa da FFh a 00h viene generato un interrupt.

TIMER1: è un timer a 16 bit (conta da 0 a 65535). Può essere usato con clock esterno su RC0 o interno. Come contatore può funzionare in modo asincrono e sincrono (sincronizzato dal clock di sistema).

TIMER2: a 8 bit con registro PR2 associato a 8 bit. Funziona con clock interno (F/4) e con prescaler.

I moduli Capture/Compare/PWM

Nel Capture , quando un evento (cambio di livello) si manifesta sull'ingresso RC2/CCP1 il registro associato CCPR1 (8 + 8 bit) cattura il contenuto di TMR1 che (timer).

In modo Compare, il contenuto del registro CPR1 (16 bit) è confrontato con quello di TMR1. Se c'è uguaglianza si agisce su RC2/CCP1 (preventivamente impostato come uscita) in vari modi:

-viene portato alto

-viene portato basso

-si genera un interrupt.

Il modo PWM

PWM (Pulse Width Modulation) è un meccanismo per comandare alcuni motori di impiego comune. Qui viene generato un segnale a duty cicle variabile, cioè in cui si può variare il rapporto fra il tempo in cui l'uscita del PIC è alta rispetto al tempo in cui è bassa.

Le istruzioni (35) del PIC16F876A, e in generale nei PIC16

(Il segno + significa che il Flag è modificato)

Codice	Operandi	Significato	Flag Z	Flag C	Flag DC
ADDWF	f,d	WREG + f => W o f	+	+	+
ANDWF	f,d	WREG and f	+		
CLRW		Azzera WREG	+		
COMF	f,d	F negato => in WREG o f	+		

DECF	f,d	Decrementa f	+		
DECFSZ	f,d	Decr. E salta l'istruz. Successiva se zero			
INCF	f,d	Incrementa f	+		
INCFSZ	f,d	Incr. F e salta se zero			
IORWF	WREG	WREG or f	+		
MOVF	f,d	Copia f in WREG (F => WREG)	+		
MOVWF	f	WREG => f			
NOP		No operation			
RLF	f,d	Ruota a sinistra f (bit 7 in carry)		+	
RRF	f,d	Ruota a destra con carry		+	
SUBWF	f,d	F – WREG => WREG oppure f (d=1)	+	+	+
SWAPF	f,d	Scambia i due nibble di f			
XORWF	f,d	WREG xor f	+		
BCF	f,b	Azzera il bit b di f			
BSF	f,b	Pone a 1 il bit bi-esimo di f			
BTFSC	f,b	Test sul bit b di f e salta se zero			
BTFSS	f,b	Test sul bit b di f e salta se 1			
ADDLW	k	Somma k a WREG (indirizzamento immediato)	+	+	+
ANDLW	k	WREG and k => WREG	+		
CALL	k	Chiama subroutine di indirizzo k			
CLRWDT		Azzera il watchdog			
GOTO	k	Salta all'indirizzo k			
IORLW	k	WREG or k => WREG	+		
MOVLW	k	Carica k in WREG			
RETFIE		Ritorna dalla routine di gestione interrupt			
RETLW	k	Ritorna dalla subroutine e poni k in WREG			
RETURN		Ritorna da subroutine			

SLEEP		Poni in standby			
SUBLW	k	WREG – k => WREG	+	+	+
XORLW	k	WREG xor k => WREG	+		

Istruzioni aggiuntive (valide per alcuni PIC16)

ADDCF	f,d	f + carry => WREG se d=0 o f	+		
ADDDCF	f,d	F + digital carry	+		
B	k	Salta all'indirizzo k (branch)			
BC	k	Salta se carry			
BDC	k	Salta se digital carry			
BNC	k	Salta se non carry			
BNDC	k	Salta se non digital carry			
BNZ	k	Salta se non zero			
BZ	k	Salta se è settato il flag di zero			
CLRC		Azzera carry			
CLRDC		Azzera digital carry			
CLRZ		Azzera flag di zero			
LCALL	n	Chiamata (lunga) a subroutine n			
LGOTO	n	Salto (lungo)			
MOVFW	f	Carica f in WREG			
NEGF	f,d	Complementa f e metti in WREG (d=0) o f	+		
SETC		Pone a 1 il carry			
SETDC		Pone a 1 il digital carry			
SETZ		Pone a 1 il flag di zero			
SKPC		Salta l'istruz. successiva se carry			
SKPDC		Salta se digital carry			

SKPNC		Salta se non carry			
SKPNZ		Salta se non zero			
SKPZ		Salta se zero			
SUBCF	f,d	f– carry => WREG oppure => f	+		
SUBDCF	f,d	f– digital carry e come sopra	+		
TSTF	f	Testa il file register f	+		
TRIS	f	Pone WREG nel registro di configurazione (I/O) della porta			

Alcuni esempi di istruzioni

Caricamento:

MOVLW n

Carica il literal (alla lettera), cioè l'argomento immediato in accumulatore (WREG): n => WREG.

IL numero caricato può essere a base 10, 2, 16. Se per default la base è 10, non occorre un'ulteriore specificazione. Si specifica invece quando il numero è a base 2, mettendo in fondo B, oppure a base 16:

0x davanti, o h dopo.

Consideriamo il numero

100.

Lo abbiamo scelto perché potrebbe essere a base 10, 2 o 16. Se è a base 10, scriveremo

MOVLW 100.

100 in base 2 è

1100100

Scriveremo:

MOVLW 1100100B.

In base 16 100 decimale è 64, scriveremo

MOVLW 0x64

Oppure MOVLW 64h.

MOVWF 100

Copia il contenuto dell'accumulatore (WREG) nella locazione di memoria f(di indirizzo 100 decimale o 64 esadecimale). Cioè WREG => f. Si noti che nel codice W viene prima di F (file register). Questo tipo di indirizzamento è comunemente chiamato diretto.

Possiamo anche dare un nome all'indirizzo che vogliamo usare

RIS1 EQU 100

E

MOVWF RIS1.

Occorre fare attenzione al banco di memoria selezionato e al fatto che i diversi PIC possono avere diverse memorie RAM a disposizione.

MOVF f,d copia f nell'accumulatore (o in se stesso se d = destinazione è 1)

SWAPF f,d scambia i nibble di f. Nibble è un semi byte

CLRF f azzera f e pone a 1 il flag di zero

CRRW azzera l'accumulatore e pone a 1 il flag di zero

Si noti che con MOVLW 0 si azzera WREG ma non viene toccato il flag di zero.

RLF f,d ruota a sinistra passando per il carry: il bit 7 di f cioè va a finire nel carry e il valore che aveva il carry entra in bit 0 di f.

RRF f,d lo stesso ma a destra

BCF f,b azzera il bit b di f

BSF f,b pone a 1 il bit b di f

Istruzioni di controllo del flusso

Le istruzioni di controllo (salto, ciclo, subroutine, ecc.) sono quelle qualificanti del microcontrollore. Negli anni 1970 sono state proposte regole per una buona programmazione professionale: la cosiddetta programmazione strutturata. Il teorema di Jacopini-Boehm assicura che per scrivere un qualsiasi programma sono sufficienti 3 strutture base:

-(1)l'elaborazione (istruzioni aritmetiche, logiche, di caricamento, ecc.)

-(2)la scelta binaria (vero / falso). E' la classica scelta dicotomica IF (Vero/Falso) THEN…ELSE…ENDIF

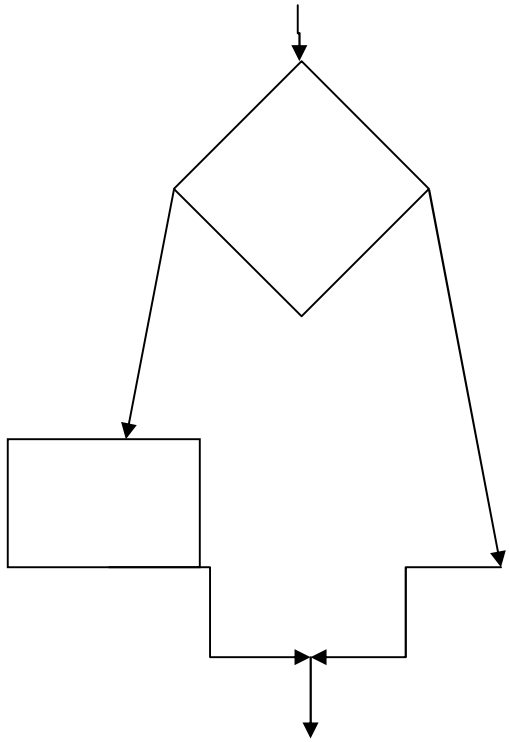
-(3)il ciclo iterativo (del TIPO FOR…NEXT. DO … .WHILE (Vero/Falso), WHILE (Vero/Falso) DO…

Si noti che non si devono usare salti del tipo GOTO al di fuori di queste strutture perché rendono il programma pasticciato e facile agli errori.

Pertanto, dobbiamo vedere come implementare la (2) e la (3) nel PIC. Per i PIC16, in generale, disponiamo di:

BTFSC f,b se il bit b di f è zero salta l'istruzione successiva.

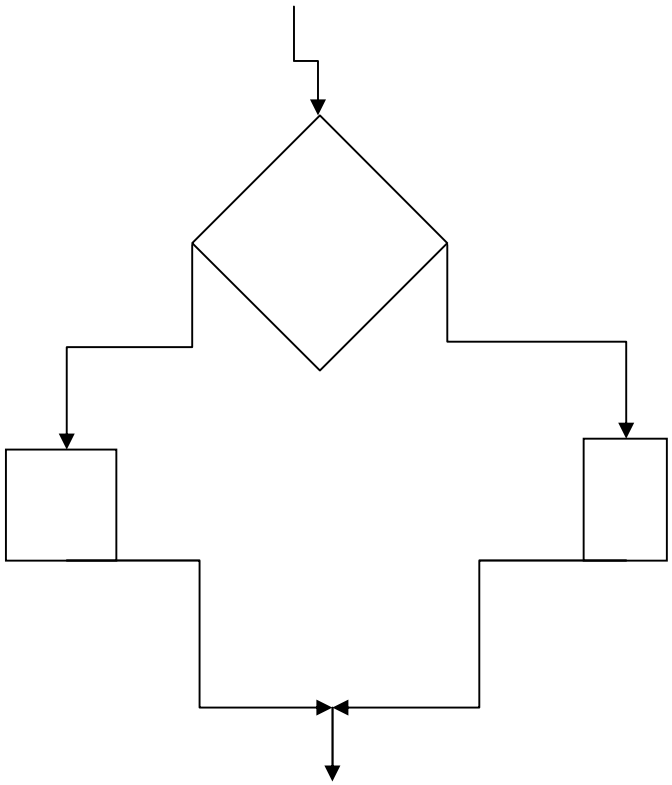
Si vede subito che questa istruzione non implementa la scelta binaria completa prevista da Jacopini-Boehm ma:



Se il bit b di f è diverso da zero esegue l'istruzione successiva, altrimenti salta l'istruzione successiva. Quasi sempre abbiamo bisogno di eseguire due alternative (con più istruzioni) a seconda del valore del bit. Siamo quindi costretti a mettere dopo la BTFSC (Bit Test su F, SKIP if Clear) un'istruzione GOTO, ad esempio:

```
        BTFSC  semaforo,0
        GOTO else
        RLF LUCE,F
        GOTO ENDIF
ELSE:   RRF LUCE,F
ENDIF.
```

In questo modo si ottiene la desiderata struttura del tipo IF…THEN…ELSE…, dove dopi i due GOTO si possono mettere le istruzioni dei due pacchetti alternativi:



I GOTO si devono usare solo all'interno di questa struttura e della struttura iterativa che andiamo adesso ad esaminare.

Per quanto riguarda i cicli (struttura iterativa) occorre osservare le differenze fondamentali fra le 3 possibilità. Il ciclo di tipo FOR…NEXT si sceglie solo quando si sa a priori quante volte deve essere eseguito il pacchetto di istruzioni del ciclo. Si può anche non usare mai: Jacopini-Boehm lo sconsigliano e invitano ad usare le altre due strutture iterative. Bisogna anche capire bene la differenza fra DO..WHILE e WHILE…DO. La prima è detta 1-ciclo perché il test è fatto in fondo e il pacchetto di istruzioni è eseguito almeno 1 volta. Il ciclo chiamato 0-ciclo, invece, cioè WHILE…DO effettua il test prima e il ciclo può non essere eseguito neanche una volta. La scelta va fatta caso per caso a seconda dell'applicazione.

I cicli del tipo FOR trovano un tipico impiego quando si deve scrivere una subroutine per creare un ritardo di n micro o milli secondi, usando anche opportune istruzioni NOP (No Operation) per far passare un tempo noto. Da ricordare che nei PIC, la NOP (e molte istruzioni) durano esattamente 1 ciclo macchina, pari a 4 periodi di clock di sistema.

Vediamo un esempio di realizzazione di ciclo FOR:

```
        MOVLW 100                ;WREG = 100
        MOVWF CONTATORE ;        ;CONTATORE = 100
NEXT:                            ; mettere istruzioni del pacchetto
                                 ;…
```

```
        MOVLW  1                ;passo del ciclo
        SUBWF CONTATORE,F       ;CONTATORE  = CONTATORE – WREG
        BTFSS   STATUS,Z        ;se contatore  = 0 salta perché fine ciclo
        GOTO NEXT
```

E' possibile usare una sola istruzione per il decremento (se è di 1) , il test e il salto:

```
        DECFSZ CONTATORE,0
        GOTO NEXT
```

Rendendo il programma più conciso.

Anche nel caso di IF…THEN…ELSE…, nel caso semplice di una sola istruzione da eseguire dopo THEN e una sola dopo ELSE, si può ottenere un programma più breve, senza etichette e senza i GOTO:

```
        BTFSC   SCELTA,0
        RRF CASO,F              ;istruzione dopo THEN (alternativa 1)
        BTFSS  SCELTA,0
        RLF CASO,F              ;istruzione dopo ELSE (alternativa 2).
```

Esiste poi un'altra soluzione, veramente professionale, anche nel caso di più di una istruzione dopo then e/o dopo ELSE: l'uso delle chiamate a subroutine che eliminano del tutto i GOTO. Ad esempio:

```
        BTFSC   SCELTA,0
        CALL  routine1
        BTFSS SCELTA,0
        CALL routine2
        …
routine1:
                ;istruzione
                ;…
        RETURN
        …

routine2:
```

```
        ;istruzione
        ;…
    RETURN
```

Esercizio1

Scrivere le istruzioni per realizzare la struttura DO…WHILE

Esercizio 2

Scrivere le istruzioni per realizzare la struttura WHILE…DO.

Terminiamo questi primi esempi parlando dell'indirizzamento indiretto tramite puntatore. I PIC dispongono del registro FSR che è appunto il puntatore, ad esempio:

```
MOVLW 100       ;WREG = 100
MOVWF FSR       ;FSR = 100
MOVLW 255       ;WREG = 255
MOVWF INDF      ;scrive 255 nella cella di indirizzo FSR, cioè la cella 100
```

E anche, per la lettura con indirizzamento indiretto:

```
MOVLW 100
MOVWF FSR
MOVF INDF,W     ;legge il valore della cella puntata.
```

PIC ad alto livello

Comprendono i PIC17 a 58 istruzioni e i PIC18 a 76 istruzioni, tutte lunghe 16 bit. Possono gestire svariati interrupt esterni.

PIC24 a 16 bit

Avere un PIC a 16 bit significa trattare dati da 16 bit: spesso i dati da trattare, ad esempio provenienti dai sensori, richiedono più di 8 bit di precisione, e quindi la scelta si sposta a questa famiglia.

I PIC24 hanno:

-real time clock e calendario

-controllo errori a CRC (Cyclic Redundancy Check)

-gestione USB

-convertitori A/D anche a 12 bit

-convertitori D/A a 10 e anche 16 bit

-Direct Memory Access (DMA)

-controllo PWM di motori

Configurazione dell'oscillatore

Sono possibili diverse configurazioni da selezionare per l'oscillatore che fornisce il clock al PIC

LP	Con cristallo a bassa frequenza	da 5 a 200kHz
XT	Con quarzo a media frequenza	da 0,1 a 4 MHz
HS	Con quarzo ad alta frequenza	da 4 a 25 MHz
RC	Circuito resistore condensatore	da 0 a 4MHz
EC	Clock esterno	da 0 a 40 MHz.

Il set di istruzioni dei PIC18

f = file register

d indica la destinazione: se 0 in accumulatore WREG, se 1 in file register

a indica la modalità di accesso alla memoria: a= 0 primi 128 bytes del banco 0 e ultimi 128 del banco 15

a=1 nel banco assegnato dal registro BSR

f = indirizzo in memoria del file register

b indica il bit (da 0 a 7) su cui si deve operare

+ significa che il Flag (bandierina) è modificato (settato)

k = numero di 4 bit (kk 8 bit, kkk 12 bit)

Si noti che il sorgente dell'operazione viene sempre indicata prima del destinatario sorg => dest, a differenza di altri microcontrollori dove avviene il contrario. Tuttavia esiste il parametro d che specifica dove sarà memorizzato il risultato: nell'accumulatore o nel registro, e questo complica il discorso.

Da notare anche che la maggior parte dei testi chiama spostamento un'operazione che si dovrebbe chiamare copia: ad esempio dall'accumulatore a un registro, perché il valore rimane ancora pure nel sorgente dell'operazione.

Il Flag Z (zero) è posto a 1 se il risultato dell'operazione è 0

Il Flag C (carry = riporto o prestito) è posto a 1 se c'è un riporto dal bit 7 (più significativo)

Il Flag OV è il flag di Overfflow

Il Flag N è il flag di segno (1 = negativo)

Il Flag DC (Digital Carry) è posto a 1 se c'è riporto dal bit 3 (nibble meno significativo) al 4

F nel codice esadecimale (HEX) è l'indirizzo del registro f da aggiungere dopo il codice dell'operazione. Il codice si riferisce al caso di parametri default.

Codice operativo	Operandi	Descrizione	Flag Z	Fag C	Overflow	Negativo	Flag DC	Codice HEX
ADDWF	f,d,a	Somma WREG e f, => WREG (d=0) o f	+	+	+	+	+	27f
ADDWFC	f,d,a	Come sopra + carry	+	+	+	+	+	23f
ANDWF	f,d,a	AND	+			+		17f
CLRF	f,a	Azzera f	+					6Bf
COMF	f,d,a	Complementa f	+			+		1Ff
CPFSEQ	f,a	Se f=WREG salta istruzione successiva						63f
CPFSGT	f,a	Se >						65f
CPFSLT	f,a	Se <						61f
DECF	f,d,a	Decrementa f	+	+	+	+	+	07f
DECFSZ	f,d,a	Dec f e salta se zero						2Ff

DECFSNZ	f,d,a	Dec f e salta se non zero						4Ff
INCF	f,d,a	Incrementa f	+	+	+	+	+	2Bf
INCFSZ	F,d,a	Incr f e salta se zero						3Ff
IORWF	f,d,a	OR	+			+		13f
MOVF	f,d,a	Copia f in WREG o f	+			+		53f
MOVFF	f',f"	Copia f' in f"						Cf'Ff"
MOVWF	f,a	Copia WREG in f						6Ff
MULWF	f,a	WREG per f, ris. in PRODH e PRODL						03f
NEGF	f,a	Complemento a 2 di f	+	+	+	+	+	6Df
RLCF	f,d,a	Ruota a sinistra con carry (il bit 7 va in riporto)	+	+		+		37f
RLNCF	f,d,a	Senza carry	+			+		47f
RRCF	f,d,a	Ruota a destra con carry (il carry va nel bit 7)	+	+		+		33f
RRNCF	f,d,a	Senza carry	+			+		43f
SETF	f,a	Tutti 1 inf						69f
SUBFWB	f,d,a	WREG – f – C => destinazione (sottrazione con prestito)	+	+	+	+	+	57f
SUBWF	f,d,a	f – WREG => destinazione	+	+	+	+	+	5Ff
SUBWFB	f,d,a	f – WREG – C => destinazione	+	+	+	+	+	5Bf

SWAPF	f,d,a	Scambia il nibble di destra con quello di sinistra e viceversa						3Bf
TSTFSZ	f,a	Se f=0 salta l'istruzione successiva						67f
XORWF	f,d,a	OR esclusivo	+			+		1Bf
BCF	f,b,a	Azzera il bit b di f						91f
BSF	f,b,a	Pone a 1 il bit b di f						81f
BTFSC	f,b,a	Se il bit b di f = 0 salta						B1f
BTFSS	f,b,a	Salta se 1						A1f
BTG	f,b,a	Bit toggle : inverte il bit di f						71f
BC	n	Salta (branch) a n se carry 1 (riporto). -128 <= n <= 127						E2n
BN	n	Salta se negativo						E6n
BNC	n	Salta se non c'è riporto						E3n
BNN	n	Salta se non negativo						E7n
BNOV	n	Salta se non overflow						E5n
BNZ	n	Salta se non zero						E1n
BOV	n	Salta se overflow						E4n
BRA	n	Salto						D'0'n

		incondizionato a n						
BZ	n	Salta se zero (flag Z = 1)						E0n
CALL	n,s	Chiama la subroutine n. Se s = 1 i registri STATUS, WREG e BSR vengono salvati in ausiliari						ECkkFkkk
CLRWDT		Azzera il watchdog timer						0004
DAW		Aggiusta in decimale l'accumulatore WREG						0007
GOTO	n	Va a n (20 bit) con 0<=n<=1048575						EFkkFkkk
NOP		Non fa nulla						0000
NOP		Non fa nulla						Fxxx
POP		Il valore in cima alla Pila viene fatto uscire e il top è posto un indirizzo più in basso (TOS – 1 => TOS)						0006
PUSH		PC + 2 => TOS						0005
RCALL	n	Chiama subroutine a n						D'1'n
RESET		Reset software						00FF
REFIE	s	Ritorno da routine di interrupt						0010

RETLW	kk	Ritorno da subroutine con kk in accumulatore						0Ckk
RETURN	s	Ritorno da subroutine						0012
SLEEP		Pone in standby						0003
ADDLW	kk	Somma kk all'accumulatore	+	+	+	+	+	0Fkk
ANDLW	kk	WREG and kk => WREG	+			+		oBkk
IORLW	kk	OR	+			+		09kk
LFSR	kk	Copia kk (8 bit meno significativi del literal kkk) in r						EErk F0kk
MOVLB	k	Copia i bit da 0 a 3 di kk nel registro BSR						010k
MOVLW	kk	Copia kk in WREG						0Ekk
MULLW	kk	WREG per kk => PRODH PRODL						0Dkk
SUBLW	kk	Kk – WREG => WREG	+	+	+	+	+	08kk
XORLW	kk	WREG xor kk => WREG	+			+		0Akk
TBLRD*		Legge la memoria di programma						0008
TBLRD*+		Legge la memoria di programma e incrementa il puntatore TBLPTR dopo la						0009

		lettura						
TBLRD*-		Legge e decrementa dopo						000A
TBLRD+*		Incrementa TBLPTR e poi legge						000B
TBLWT		Scrive sulla memoria						000C
TBLWT*+		Scrive e incrementa TBLPTR						000D
TBLWT*-		Scrive e decrementa TBLPTR						000E
TBLWT+*		Incrementa TBLPTR e scrive						000F

Il PIC18F4520 (e il PIC18F2520)

Caratteristiche

-Memoria Flash (32 Kbytes) avanzata con 100000 cicli di lettura e scrittura e 1000000 per la EEPROM

-auto programmazione: all'interno del programma è possibile programmare la stessa memoria di programma

-8 nuove istruzione per l'indirizzamento indicizzato

-il modulo PWM è migliorato per la gestione diretta dei drive a ponte dei motori

-USART completa per seriale RS232

-Watchdog avanzato

-convertitore A/D più flessibile e con 13 canali a 10 bit

Piedini (40, contenitore PDIP):

Nota: le porte A,B,C,D,E sono bidirezionali. Sottolineato significa attivo basso

1 MCLR / Vpp / RE3 Master Clear (reset attivo basso), Tensione programmazione, Ingresso digitale

13 OSC1 / CLKI /RA7 Ingresso oscillatore, Ingresso Clock esterno, I/O digitale

14 OSC2 / CLKO / RA6 Uscita oscillatore, uscita clock esterno, I/O digitale

2 RA0/AN0 I/O digitale / Ingresso analogico (convertitore A/D)

3 RA1,AN1 I/O digitale / Ingresso analogico

4 RA2/AN2/VREF-/CVREF I/O digitale/Ingresso analogico/tensione riferimento A/D -/t. rifer.comp.

5 RA3/AN3/VREF+ I/O digitale/Ingresso analogico/tensione riferimento A/D +

6 RA34/T0CKI/C1OUT I/O digitale/ingresso clock Timer 0/ uscita comparatore 1

7 RA5/AN4/SS/HLVDINC2OUT I/O digitale/ingr.analogico/SPI selez.in/rileva tensione H-I/uscita comparatore 2

33 RB0/INT0/FLT0/AN12 I/O digitale/ingresso analogico/ interrupt esterno 0

34 RB1/AN1/INT1 I/O digitale/ingresso analogico/Interrupt esterno 1

35 RB2/AN8/INT2 I/O digitale/ingresso analogico/interrupt esterno 2

36 RB3/AN9/CCP2 I/O digitale/ingresso analogico/ingresso Capture 2/uscita Compare 2/uscita PWM2

37 RB4/AN11/KBI0 I/O digitale/ingresso analogico/Interrupt su cambiamento di livello 0

38 RB5/KBI1/PGM I/O digitale/Interrupt su cambiamento di livello 1/abilitazione ICSP Low-Voltage

39 RB6/KBI2/PGC I/O digitale/Interrupt su cambiamento di livello 2/Clock per ICSP e debugger

40 RB7/KBI3/PGD I/O digitale/Interrupt su cambio livello 3/Dati ICSP e debugger

15 RC0/T1OSO/T13CKI I/O digitale/uscita oscillatore Timer 1/ingresso clock esterno per i Timer 1,3

16 RC1/T1OSI/CCP2 I/O digitale/ingresso oscillatore Timer 1/ingresso Capture 2/uscita Compare 2/uscita PWM2 (alternativa al piedino 36)

17 RC2/CCP1/P1A I/O digitale/ingresso Capture 1/uscita Compare 1/uscita PWM 1

18 RC3/SCK/SCL I/O digitale/clock input-output sincrono per SPI/idem per I^2C

23 RC4/SDI/SDA I/O digitale/dati per SPI/dati I/O per I^2C

24 RC5/SDO/ I/O digitale/Uscita dati SPI

25 RC6/TX/CK I/O digitale/trasmissione asincrona (extended USART)/USART Clock modo sincrono

26	RC7/RX/DT	I/O digitale/ricezione asincrona (EUSART)/Dati modo sincrono
19	RD0/PSP0	I/O digitale/ Porta parallela per interfacciamento
20	RD1/PSP1	I/O digitale/ Porta parallela per interfacciamento
21	RD2/PSP2	I/O digitale/Porta parallela come sopra
22	RD3/PSP3	I/O digitale/Porta parallela
27	RD4/PSP4	I/O digitale/Porta parallela
28	RD5/PSP5/P1B	I/O digitale/Porta parallela/Uscita CCP1
29	RD6/PSP6/P1C	I/O digitale/Porta parallela/Uscita CCP1
30	RD7/PSP7/P1D	I/O digitale/Porta parallela/Uscita CCP1
8	RE0/RD/AN5	I/O digitale/Controllo lettura porta parallela/ingresso analogico
9	RE1/WR/AN6	I/O digitale/Controllo scrittura porta parallela/ingresso analogico
10	RE2/CS/AN7	I/O digitale/Chip Select porta parallela/ingresso analogico
12	VSS	Massa (- alimentazione)
11	VDD	+ alimentazione

Strumenti di sviluppo

Hardware utilizzato:

-programmatore di PIC attraverso ISP e porta RS232 (seriale verso PC), con autodect (riconoscimento automatico dell'integrato) per PIC a 18, 28, 40 piedini

-scheda di sviluppo e prova per PIC a 40 piedini, con LED, seriale, ISP, ecc.

Software utilizzato:

-MPLAB (free) Microchip 8.53

-Compilatore C (Student edition, free) Microchip C32

-PIC Simulator e PIC18 Simulator, con compilatore Basic per PIC

Bibliografia

-Fernando E. Valdes-Peres, Ramon Pallas-Areny, Microcontrollers, Fundamentals and Applications with PIC, CRC Press, Boca Raton, Florida, USA, 2009

-A. De Santis, M. Cacciaglia, C. Saggese, Sistemi2 e 3, Calderini, Milano, 2005

Sul Web:

-naturalmente, www.microchip.com

-e www.itiomar.it (area studenti, documentazione area elettronica, PIC, Conti)

-il riferimento famoso è quello di Sergio Tanzillli, autori di corsi sui PIC (ad esempio Pic by Example)

Tuttavia, la cosa migliore che potete fare è venire all'Omar e iscrivervi all'indirizzo Elettronica, oppure Informatica, e seguire le lezioni (e i laboratori di progettazione) sui PIC del prof. ing. Giuseppe Peretti, del prof. Sergio Conti e del prof. Angelo Garro.