I Microcontrollore *PIC18FXX2*

Descrizione dei microcontrollori

I microcontrollori sono dei dispositivi elettronici di piccole dimensioni che, per molti versi, possono essere assimilati a dei veri e propri computer. Essi, infatti, contengono al proprio interno tutti i dispositivi tipici di un sistema a microprocessore, ovvero:

- Una **CPU** (**C**entral **P**rocessor **U**nit) ovvero un'unità centrale di elaborazione il cui scopo è interpretare le istruzioni di programma.
- Una memoria **FLASH** in cui sono memorizzare in maniera permanente le istruzioni del programma da eseguire.
- Una memoria **RAM** (**R**andom **A**ccess **M**emory) utilizzata per memorizzare le variabili utilizzate dal programma.
- Una serie di **LINEE DI I/O** (**I**nput/**O**utput) ovvero linee di ingresso e uscita per pilotare dispositivi esterni o ricevere impulsi da sensori, pulsanti, ecc.
- Una serie di dispositivi ausiliari al funzionamento quali generatori di clock, bus, contatori, ecc.

La presenza di tutti questi dispositivi in uno spazio estremamente contenuto, consente al progettista di avvalersi degli enormi vantaggi derivanti dall'uso di

un sistema a microprocessore, anche in quei circuiti che fino a poco tempo fa erano destinati ad essere realizzati con circuiterie tradizionali.

La memoria non volatile serve a contenere il programma software che dovrà essere eseguito ripetutamente affinché il dispositivo svolga la funzione desiderata. La memoria volatile, ha il compito di immagazzinare i dati elaborati dall'ALU; le linee di I/O, collegate a numerosi piedini che fuoriescono dalla struttura del microcontrollore, servono ad interfacciare il dispositivo con il mondo esterno. Quasi sempre questi dispositivi contengono al loro interno anche altri circuiti per comunicazioni seriali, dei comparatori e dei convertitori Analogico/Digitale. Le funzioni delle porte di I/O e i rispettivi stati logici sono direttamente controllabili dal firmware contenuto nella memoria flash del microcontrollore (interpretato dall'unita logica), questo consente, ad esempio, di collegare un pulsante ad una porta d'ingresso per poter rilevare il suo stato, e, in conseguenza di questo accendere o spegnere un LED collegato ad un piedino d'uscita.

Ovviamente, per realizzare tale funzione, dovremo scrivere un programma adatto allo scopo. Prima di procedere alla stesura del software è consigliabile descrivere "a parole" le operazioni da eseguire, tracciando cioè quello che viene chiamato "Flow chart", un diagramma di flusso della sequenza delle operazioni da impartire al nostro dispositivo.

Variando semplicemente il firmware lo stesso dispositivo che legge il pulsante e accende il LED, può essere utilizzato indifferentemente per leggere

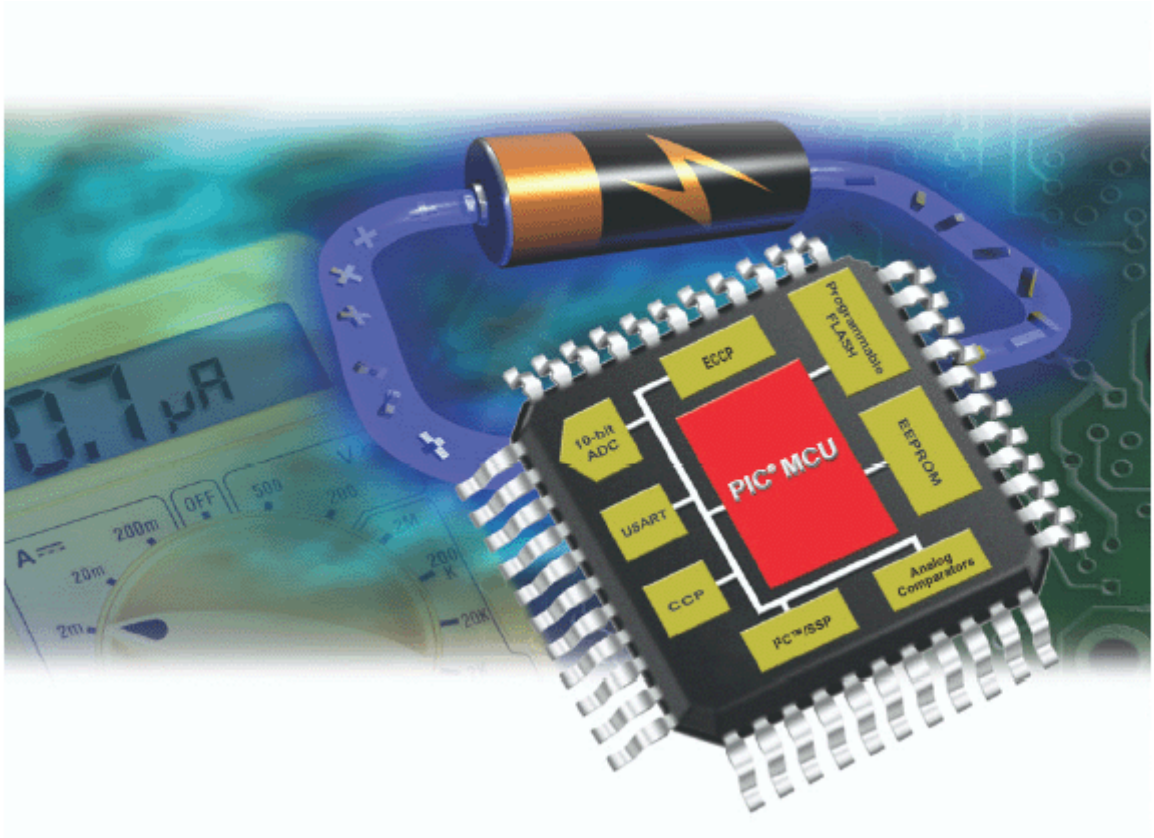
una tastiera, pilotare un display, controllare la velocità di un motore e quant'altro si possa immaginare. Il microcontrollore è dunque un dispositivo che non si può usare così come si compra, poiché, ovviamente, viene venduto senza alcun programma inserito al suo interno.

Il programma deve essere memorizzato dopo l'acquisto, allo stesso modo in cui si programma una normale memoria Eprom, utilizzando, però, un programmatore apposito, che varia a seconda della famiglia di controllori.

I programmi per i PIC vengono generalmente scritti in un linguaggio assembler o, a livello più professionale, in C, Basic o altri linguaggi di programmazione ad alto livello.

Il programma scritto in assembler [C], viene poi assemblato [compilato], ovvero tradotto in una sequenza di byte (codice macchina comprensibile dal dispositivo) che, memorizzata nella memoria interna, permette al microcontrollore di lavorare nel modo desiderato.

Esistono diverse famiglie di dispositivi in grado di svolgere queste funzioni come ad esempio lo Z80, ST6 e il più evoluto 8088; tra i più semplici e diffusi dispositivi oggi in commercio ci sono i cosi detti PIC, prodotti e distribuiti dalla Microchip.



Le famiglie di Microcontrollori

Fin dalla sua nascita è apparsa evidente l'enorme potenzialità di questi dispositivi e le case produttrici di componenti elettronici hanno investito moltissimo nella ricerca e nello sviluppo di dispositivi sempre più completi e prestanti.

Questo ha determinato la nascita di tutta una serie di dispositivi che differiscono per diverse caratteristiche funzionali. In particolare, parlando di microcontrollori, si fa generalmente riferimento alle famiglie, intendendo con questo termine dei dispositivi che hanno in comune diverse caratteristiche, fra cui ad esempio il linguaggio di programmazione (che è per tutte l'assembler,

ma le cui singole istruzioni variano a seconda dei dispositivi), l'organizzazione della memoria interna, la gestione delle periferiche e così via.

All'interno di ogni famiglia, vi sono diversi dispositivi con caratteristiche differenti da uno all'altro (più o meno memoria, velocità di funzionamento, presenza di periferiche analogiche quali convertitori e così via) per rispondere al meglio alle esigenze di un certo progetto.

E' abbastanza intuitivo, infatti, che realizzare un semplice circuito per fare accendere e spegnere dei LED richiede componenti con prestazioni ben diverse rispetto, ad esempio, ad una centralina antifurto programmabile.

Una breve panoramica sui Microcontrollori

I Microcontrollori si distinguono per il numero di bit che riescono ad elaborare contemporaneamente, per la quantità di memoria dati di cui dispongono, per la quantità di memoria riservata ai programmi, per la tipologia di quest'ultima, per la quantità di Timer a disposizione etc.

La grande maggioranza dei micro è ad 8 bit, ma esistono famiglie a 16 bit ed anche a 32 bit. Questi ultimi sono i più rari e, ovviamente, i più costosi.

Fra i dispositivi ad 8 bit, sicuramente il capostipite è stato l'8051, prodotto dall'Intel, che ha dato luogo a tutta una serie di derivati con prestazioni sempre superiori.

Un altro gigante dell'elettronica, Motorola, è ben rappresentato nel mercato degli 8 bit attraverso il 68HC11. Motorola, per inciso, è uno dei maggiori produttori di micro a 16 e 32 bit, che sono dei derivati del famoso 68000. Anche la Zilog (Z80) ha una famiglia di micro a 8 bit, gliZ8, che presentano delle caratteristiche elevate per quanto riguarda il rapporto costo/prestazioni.
SGS-Thomson produce gli ST6, una famiglia di micro che ha avuto una larga diffusione soprattutto sul mercato europeo.
Fra gli ultimi arrivati, ma diventati in breve tempo i microcontrollori più significativi ed importanti, troviamo i PIC, prodotti da un'azienda americana, l'Arizona Microchip specializzata proprio nella costruzione di microcontrollori ad 8 bit e di memorie EPROM.
Le memorie non volatili, necessarie per la memorizzazione del programma, differiscono, come gia detto, oltre che per la quantità anche per il tipo. Esistono PIC con memorie di tipo EEPROM, cioè cancellabili e riscrivibili elettronicamente, con memorie di tipo Flash, che sono da preferire nelle fasi di sperimentazione grazie al maggior numero di scritture possibili; ne esistono versioni con memorie OTP (One Time Programmable), cioè si programmano una sola volta, versioni UV con memoria che una volta scritta può essere cancellata solo mediante raggi ultravioletti.

La Famiglia Microchip PIC18FXX2

I microcontrollori PIC della famiglia 18FXX2 sono contenuti in packages con 28, 40 o 44 pin. I PIC contenuti in packages da 28 pin non hanno porte

parallele implementate e il numero di convertitori Analogico Digitale (AD) sono solo 5. I PIC della sottofamiglia 18F2XX, dove X sta ad indicare un numero 1,2,3,4… hanno 20 pin, mentre quelli appartenenti alla sottofamiglia denominata 18F4XX ne hanno 40 o 44. Tutti i dispositivi della famiglia 18FXXX possono essere contenuti in contenitori di tipo PLCC, TQFP, DIP, oppure SOIC.

Nelle tabelle seguenti sono riportate in forma abbreviata le caratteristiche salienti dei PIC della famiglia 18FXXX e le differenze tra i vari device.

DEVICE FEATURES

Features	PIC18F242	PIC18F252	PIC18F442	PIC18F452
Operating Frequency	DC - 40 MHz	DC - 40 MHz	DC - 40 MHz	DC - 40 MHz
Program Memory (Bytes)	16K	32K	16K	32K
Program Memory (Instructions)	8192	16384	8192	16384
Data Memory (Bytes)	768	1536	768	1536
Data EEPROM Memory (Bytes)	256	256	256	256
Interrupt Sources	17	17	18	18
I/O Ports	Ports A, B, C	Ports A, B, C	Ports A, B, C, D, E	Ports A, B, C, D, E
Timers	4	4	4	4
Capture/Compare/PWM Modules	2	2	2	2
Serial Communications	MSSP, Addressable USART	MSSP, Addressable USART	MSSP, Addressable USART	MSSP, Addressable USART
Parallel Communications	—	—	PSP	PSP
10-bit Analog-to-Digital Module	5 input channels	5 input channels	8 input channels	8 input channels
RESETS (and Delays)	POR, BOR, RESET Instruction, Stack Full, Stack Underflow (PWRT, OST)	POR, BOR, RESET Instruction, Stack Full, Stack Underflow (PWRT, OST)	POR, BOR, RESET Instruction, Stack Full, Stack Underflow (PWRT, OST)	POR, BOR, RESET Instruction, Stack Full, Stack Underflow (PWRT, OST)
Programmable Low Voltage Detect	Yes	Yes	Yes	Yes
Programmable Brown-out Reset	Yes	Yes	Yes	Yes
Instruction Set	75 Instructions	75 Instructions	75 Instructions	75 Instructions
Packages	28-pin DIP 28-pin SOIC	28-pin DIP 28-pin SOIC	40-pin DIP 44-pin PLCC 44-pin TQFP	40-pin DIP 44-pin PLCC 44-pin TQFP

Tabella 3.1: Caratteristiche salienti dei PIC della famiglia 18FXXX.

DEVICE DIFFERENCES

Feature	PIC18F242	PIC18F252	PIC18F442	PIC18F452
Program Memory (Kbytes)	16	32	16	32
Data Memory (Bytes)	768	1536	768	1536
A/D Channels	5	5	8	8
Parallel Slave Port (PSP)	No	No	Yes	Yes
Package Types	28-pin DIP 28-pin SOIC	28-pin DIP 28-pin SOIC	40-pin DIP 44-pin PLCC 44-pin TQFP	40-pin DIP 44-pin PLCC 44-pin TQFP

Tabella 3.2: Differenze tra i PIC della famiglia 18FXXX.

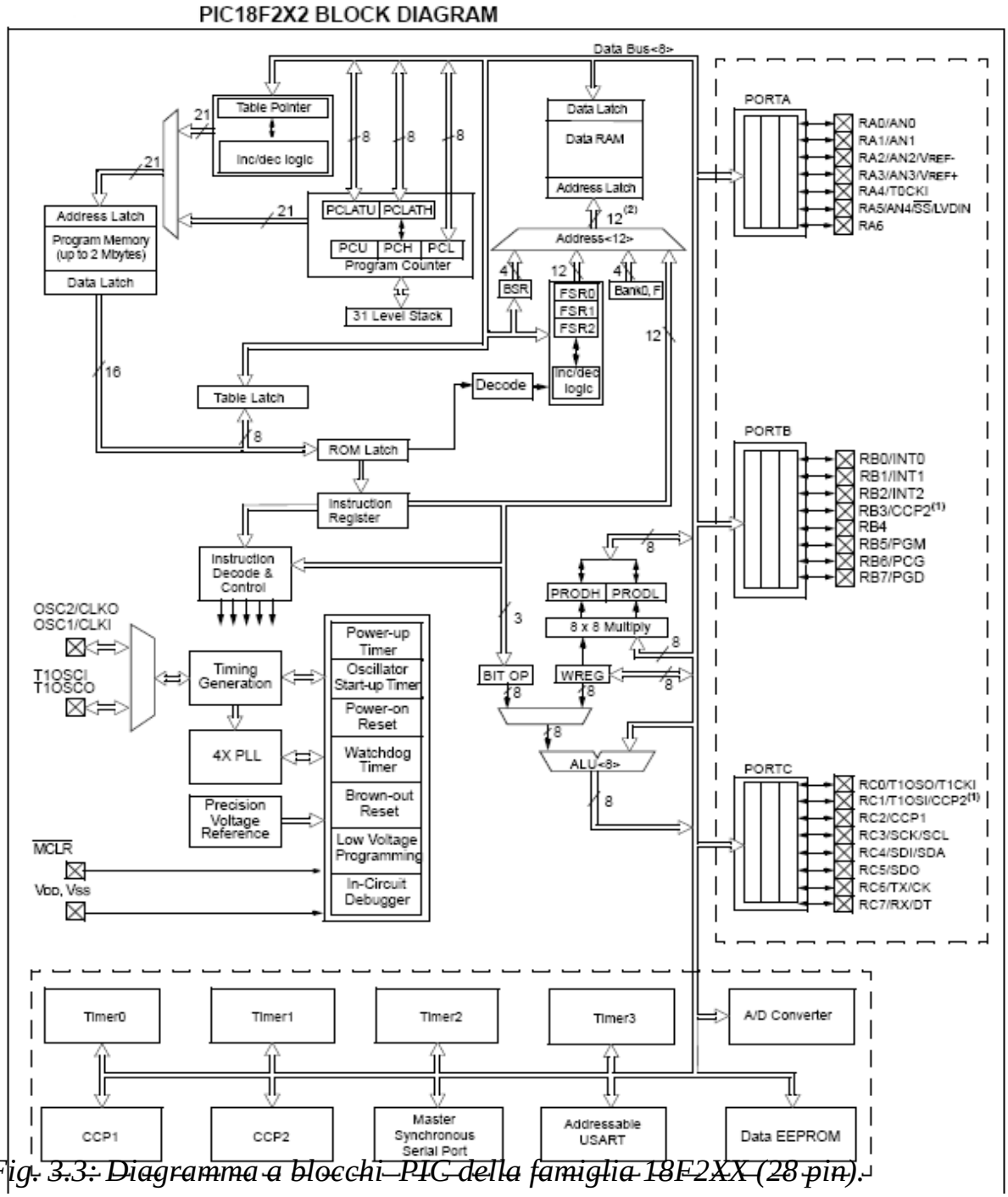


Fig. 3.3: Diagramma a blocchi PIC della famiglia 18F2XX (28 pin).

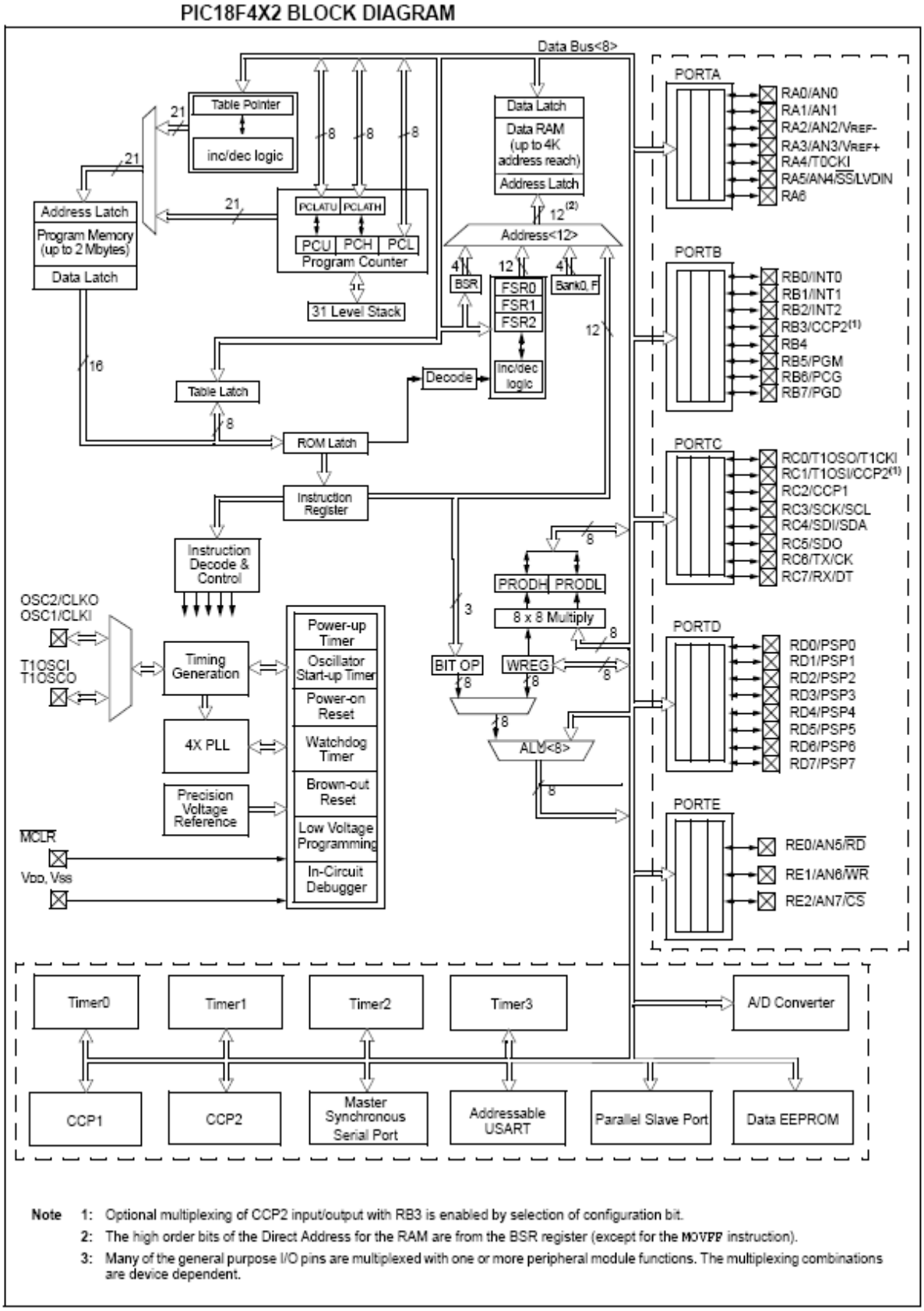


Fig. 3.4: Diagramma a blocchi PIC della famiglia 18F4XX (40 o 44 pin).

Caratteristiche della CPU

L'architettura utilizzata per il set d'istruzioni è di **tipo RISC** e ogni istruzione viene identificata da una parola di 16 bit. Questa caratteristica impone un set limitato d'istruzioni (75 istruzioni)**,** ma in ogni caso superiore a quello dei PIC delle altre famiglie, come ad esempio per la 16FXXX che, prevedono una lunghezza di parola delle istruzioni di solo 14 bit, aveva un set d'istruzioni ancora più piccolo (solo 35); mancavano quindi istruzioni come la moltiplicazione e la divisione che per poter essere svolte richiedevano più colpi di clock. Le CPU dei PIC della famiglia 18FXXX sono invece definite **High Performance RISC CPU,** infatti esse integrano al proprio interno un moltiplicatore 8 X 8 di tipo hardware, questo rende la moltiplicazione un'operazione hardware che può essere eseguita in un solo ciclo di clock. Il risultato di tale operazione è senza segno, ha una lunghezza di 16 bit e viene immagazzinato in un registro a 16 bit formato dalla coppia di registri denominati PRODH-PRODL.

L' operazione di moltiplicazione non implica l'innalzamento di nessun flag nel registro ALUSTA.

La possibilità di eseguire moltiplicazioni in un singolo colpo di clock ha i seguenti vantaggi:

1) Aumentare la capacità computazionale del dispositivo.

2)Ridurre le dimensioni del codice richiesto per implementare gli algoritmi di moltiplicazione

La tabella seguente riporta il confronto tra le prestazioni tra un device che implementa la moltiplicazione in hardware in un singolo ciclo ed uno che non la implementa.

PERFORMANCE COMPARISON

Routine	Multiply Method	Program Memory (Words)	Cycles (Max)	Time		
				@ 40 MHz	@ 10 MHz	@ 4 MHz
8 x 8 unsigned	Without hardware multiply	13	69	6.9 µs	27.6 µs	69 µs
	Hardware multiply	1	1	100 ns	400 ns	1 µs
8 x 8 signed	Without hardware multiply	33	91	9.1 µs	36.4 µs	91 µs
	Hardware multiply	6	6	600 ns	2.4 µs	6 µs
16 x 16 unsigned	Without hardware multiply	21	242	24.2 µs	96.8 µs	242 µs
	Hardware multiply	24	24	2.4 µs	9.6 µs	24 µs
16 x 16 signed	Without hardware multiply	52	254	25.4 µs	102.6 µs	254 µs
	Hardware multiply	36	36	3.6 µs	14.4 µs	36 µs

Tabella 3.3: Confronto tra le prestazioni di device con e senza moltiplicatore hardware.

Oltre alle “classiche” funzionalità le CPU di molti dispositivi della Microchip integrano al loro interno molte altre caratteristiche il cui scopo principale è quello di massimizzare l’affidabilità del sistema, minimizzare il costo di apparecchiature che integrano PIC, eliminando altri componenti hardware le cui funzioni possono essere svolte senza problemi dal PIC stesso e, in fine, garantire un consumo di corrente basso.

Le caratteristiche extra implementate nei PIC della famiglia 18FXXX sono:

- OSC Selection
- RESET
  - Power-on Reset (POR)
  - Power-up Timer (PWRT)
  - Oscillator Start-up Timer (OST)
  - Brown-out Reset (BOR)
- Interrupts
- Watchdog Timer (WDT)
- SLEEP
- Code Protection
- ID Locations
- In-Circuit Serial Programming

Tutti i dispositivi della famiglia 18FXX2 hanno un Watchdog Timer che è sempre abilitato tramite i configuration bit o tramite il software di controllo. Ci sono due timers che offrono la possibilità di avere un ritardo dopo l'accensione: Oscillator Start-up Timer (OST) e Powerup Timer (PWRT).

L' OST ha lo scopo di tenere il dispositivo in RESET fino a quando l'oscillatore appena acceso non si è stabilizzato, il PWRT garantisce un ritardo prestabilito solo al momento dell'accensione. Il suo scopo è quello di tenere il dispositivo in condizione di _RESET fino a quando la tensione d'alimentazione non si è stabilizzata.

La modalità di funzionamento denominata SLEEP è stata progettata per garantire assorbimenti di corrente molto bassi. Si può uscire dalla condizione di SLEEP mediate un RESET esterno, mediante l'operazione di wake-up da parte del Watchdog Timer oppure mediante la richiesta d'interrupt.

Sono previsti più modi per realizzare il circuito oscillante al fine di poter personalizzare il più possibile il dispositivo in base alle esigenze personali, ad esempi se si è interessati al contenimento dei costi allora si può realizzare l'oscillatore mediante un cappio RC, mentre se si vuole consumare meno corrente possibile si può optare per un circuito oscillante LP a cristallo.

Configuration Bits

I bit di configurazione (Configuration Bits) possono essere programmati o meno, per selezionare varie configurazioni del dispositivo. I bit che sono stati

programmati saranno posti a ‘0’, mentre quelli non programmati conterranno il valore ‘1’. Questi bit sono mappati in memoria (nella Program Memory) a partire dalla locazione ‘300000h’, cioè appena oltre lo spazio riservato al programma dell’utente.

La programmazione dei registri di configurazione è fatta in maniera molto simile alla programmazione della memoria FLASH. L’unica differenza consiste nel fatto che i registri di configurazione vengono scritti un byte per volta.

Watchdog Timer (WDT)

Il Watchdog Timer (WDT) è costituito da un circuito risonante di tipo RC il cui funzionamento è indipendente da quello che accade al dispositivo e, inoltre, per poter funzionare non richiede l’aggiunta di nessun componente esterno. Questo consente a tale oscillatore RC di continuare a lavorare anche se il clock principale, cioè quello esterno collegato al pin OSC1/CLCK1 o al pin OSC2/CLOCK0/RA6, è stato fermato come conseguenza di un comando, come ad esempio SLEEP.

Se il dispositivo sta funzionando regolarmente, cioè non è in modalità SLEAP, il time-out del WDT causa solo un reset del device (Watchdog Timer Reset), mentre se il device è in SLEAP mode viene riattivato dal time-out del

WDT. Il WDT può essere abilitato mediante un opportuno bit del Configuration Bits. Se tale bit è abilitato il WDT non si può disabilitare per via software se, al contrario, tale bit non è abilitato, si può procedere all'abilitazione e alla disabilitazione software del WDT.

Power-down Mode (SLEEP)

Il Power-down Mode viene attivato mediante l'istruzione SLEEP. In questa modalità si ha lo spegnimento della sorgente di clock, le porte di I/O mantengono lo stato logico in cui si trovavano prima dell'esecuzione dell'istruzione SLEEP e i pin definiti come uscite vengono portati o al valore V_{DD} o V_{SS} in funzione dello stato logico in cui si trovano. Quest'ultimo accorgimento consente di minimizzare la corrente assorbita, infatti, in questo modo nessun circuito esterno può prelevare corrente dalle porte.
Il device può uscire dalla condizione di Power-down nei seguenti modi:

1. External RESET (portando allo stato logico alto il pin MCLR).
2. Watchdog Timer Wake-up (solo se il WDT era stato abilitato).

3. Interrupt proveniente dal pin INT, cambiamenti sulla porta RB

oppure richiesta d'Interrupt da parte di una periferica.

Non tutti gli interrupt provenienti da parti periferiche del PIC possono generare il wake-up.

Solo le seguenti fanno uscire il dispositivo dalla modalità Power-down:

1. PSP read or write.
2. TMR1 interrupt. Timer1 must be operating as an asynchronous counter.
3. TMR3 interrupt. Timer must be operating as an asynchronous counter.
4. CCP Capture mode interrupt.
5. Special event trigger (Timer1 in Asynchronous mode using an external clock).
6. MSSP (START/STOP) bit detect interrupt.
7. MSSP transmit or receive in Slave mode (SPI/I2C).
8. USART RX or TX (Synchronous Slave mode).
9. A/D conversion (when A/D clock source is RC).
10. EEPROM write operation complete.
11. LVD interrupt

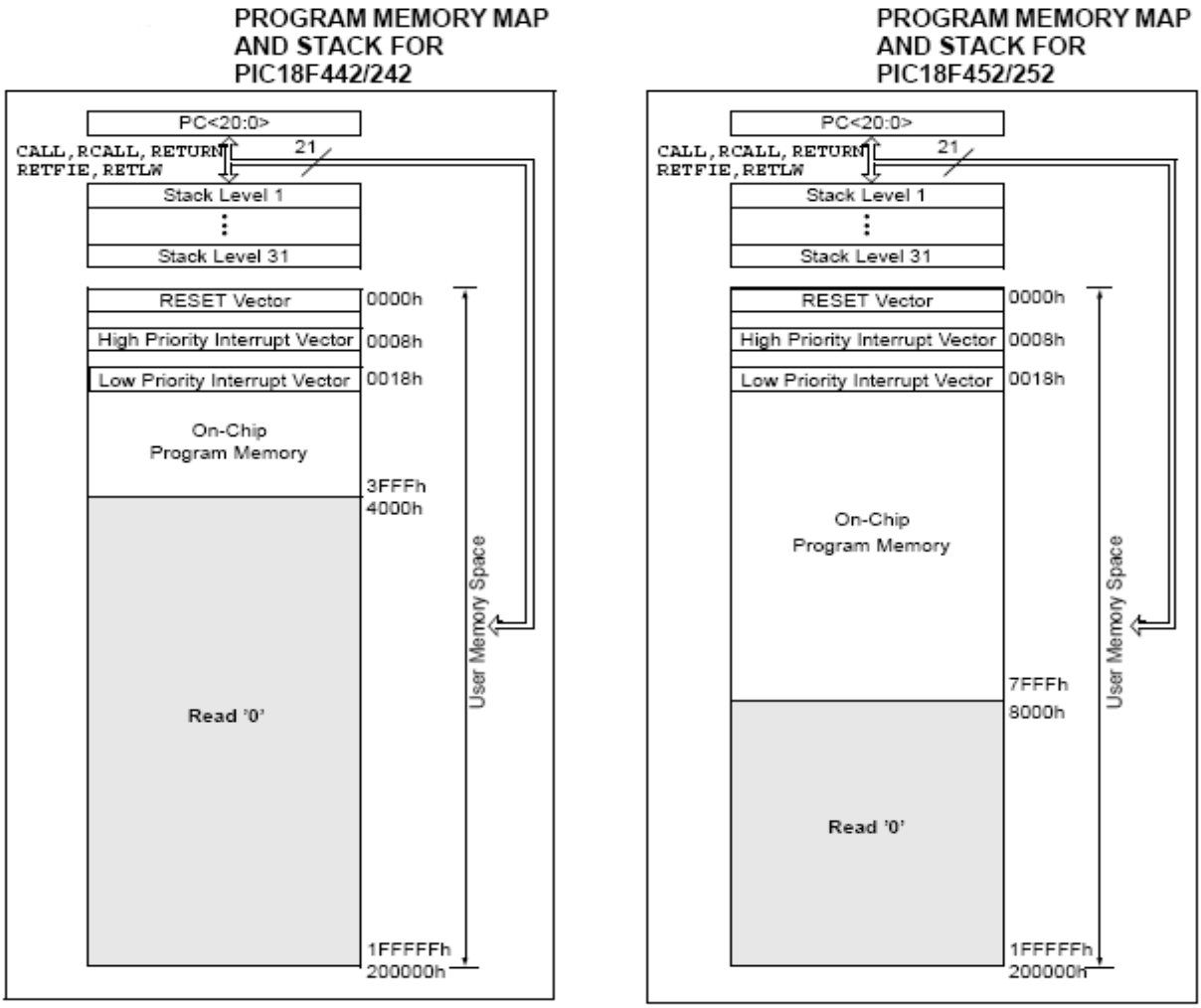
Organizzazione della memoria

Questo tipo di PIC prevede l'organizzazione della memoria in tre blocchi che sono i seguenti:

- Program Memory
- Data RAM
- Data EEPROM

La Data Memory e la Program Memory prevedono dei bus separati al fine di garantire un acceso concorrente a questi blocchi.
Viene utilizzato un contatore a 21 bit Per indicizzare 2 Mega Byte di Program Memory. Se si accede in lettura alle locazioni di memoria comprese tra quella fisicamente implementata (32 MB per la famiglia 18FX52) e quella indicizzabile, si leggeranno tutti zero (A NOP INSTRUCTION).
I PIC 18F252 e 18F452 hanno entrambi 32 Kilo bytes di memoria flash, mentre i PIC 18F242 e 18F442 ne hanno 16 Kilo bytes. Questo significa che i device della famiglia 18FX52 possono contenere un programma di lunghezza massima pari a sedicimila word (16K Word Instruction), mentre i device della famiglia 18FX42 possono contenere al massimo un programma di 8K Word Instruction. Il vettore RESET è mappato in memoria all'indirizzo '0000h' mentre i vettori delle interrupt sono mappati agli indirizzi '0008h' e '0018h'.

La figura seguente mostra l'architettura della Program Memory:



Struttura della Program Memory per PIC 18FX42 e 18FX52.

Lo stack di ben trentuno Return Address permetta una qualsiasi combinazione di chiamate a sottoprogrammi e interruzioni. Il PC (Program Counter) viene posto pari al valore contenuto in uno dei trentuno valori dello stack quando

viene eseguita una chiamata a sottoprogramma o viene servita una interrupt mediante le funzioni CALL o RCALL. Il valore del PC viene spostato da quello dello stack mediante la chiamata delle funzioni RETURN, RETLW o RETFIE.

Lo spazio riservato allo stack dei puntatori non fa parte del programma o dei dati. il puntatore allo stack può essere sia letto che scritto e l'indirizzo all'inizio dello stack può essere letto e scritto mediante il registro SFR.

La memoria Data Memory è implementata mediante l'uso di una memoria RAM. Ogni registro della Data Memory ha un indirizzo costituito da 12 bit, consentendo così, di avere una memoria dati indirizzabile di 4096 bytes. La mappa della memoria dati viene suddivisa in 16 banchi da 256 bytes ciascuno (16 X 256 bytes = 4096 bytes). I quattro bit meno significativi del Bank Select Register (BSR <3:0>) indicano quale banco della memoria accedere. Gli altri quattro bit del BSR, quelli più significativi (BSR <7:4>) non vengono utilizzati. La Data Memory contiene indirizzi specifici, Special Function Registers (SFR), e registri generici, General Purpose Registers (GPR). GLI Special Function Registers sono utilizzati per effettuare operazioni di controllo, per contenere informazioni riguardanti lo stato del controllore e delle periferiche, mentre i General Purpose Registers vengono usati per immagazzinare i dati elaborati dal microcontrollore.

I Special Function Registers iniziano dall'ultima locazione di memoria del quindicesimo banco ('0XFFF') e si estendono verso il l'alto. Tutto lo spazio

rimanente nel banco di memoria oltre agli Special Function Registers può essere usato per registri generici. I General Purpose Registers partono dal primo indirizzo del banco di memoria ‘0’ e si sviluppano verso il basso. Se si prova a leggere ad una locazione di memoria non implementata si otterrà come risultato uno zero (‘0’).

L‘ intera memoria dati può essere acceduta sia direttamente che indirettamente. L’ indirizzamento diretto può richiedere l’uso del BSR Register. L’ indirizzamento indiretto richiede l’uso di un File Select Register (FSRn) e di un corrispondente Indirect File Operand (INDFn). Ogni File Select Register contiene un indirizzo di 12 bit che può essere usato per indirizzare ad una qualsiasi locazione della Data Memory senza alcun bisogno di conoscere a quale banco appartiene.

L'insieme d’istruzioni e l’architettura permettono operazioni su tutti i banchi di memoria. Questo può essere fatto mediante indirizzamento indiretto oppure mediante l'uso dell'istruzione di MOVFF.

L'istruzione di MOVFF è un'istruzione di two-word/two-cycle che permette spostare un valore da un registro ad un altro.

Il File Select Register (FSRn) è allocato nella prima metà del quindicesimo banco di memoria, dall’indirizzo ‘0XF80’ all’indirizzo ‘0XFF’.

Di seguito viene riportato uno schema che illustra l’organizzazione della Data Memory:

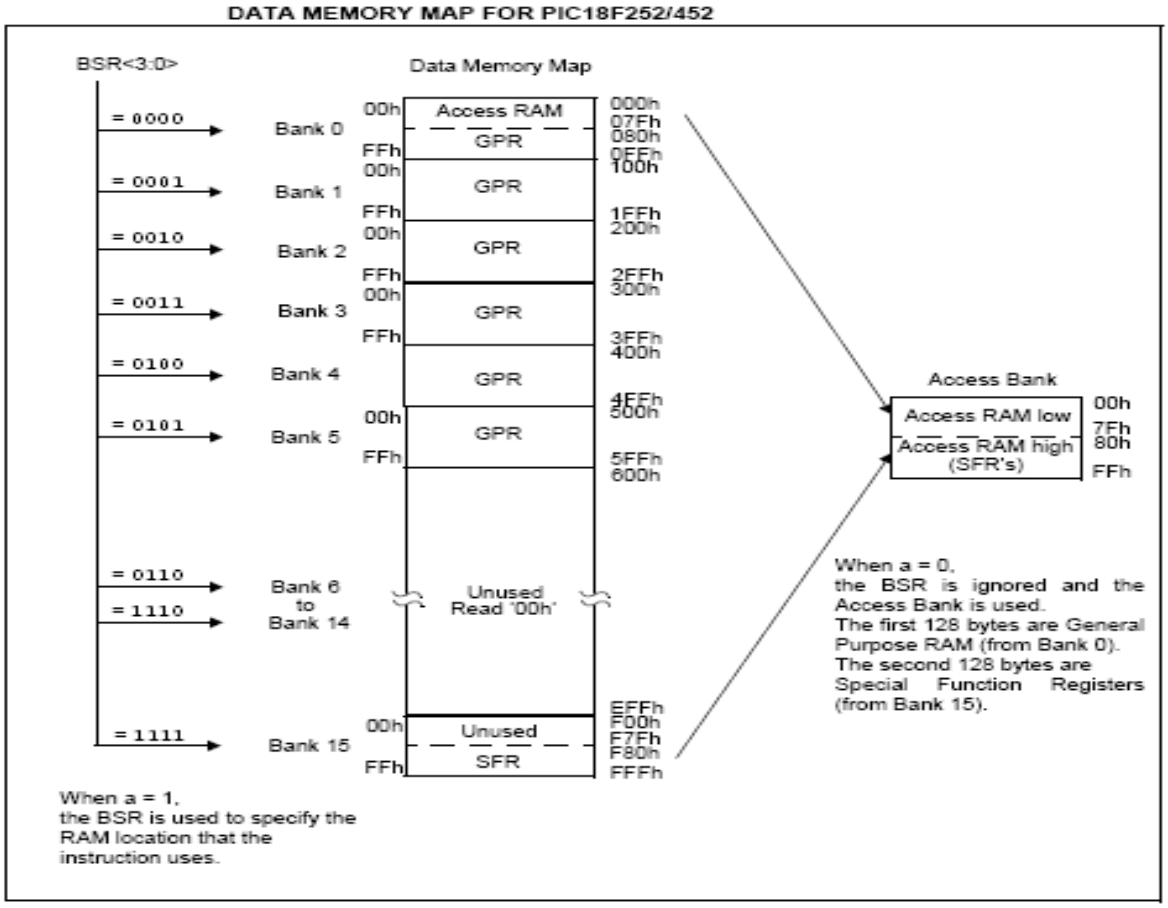


Fig. 3.6: Struttura della Data Memory dei PIC 18F252 e 18F452.

Gli Special Function Registers sono registri utilizzati dalla CPU e dalle unita periferiche per controllare il funzionamento del dispositivo. Essi possono essere classificati in due categorie:

1) Special Function Registers associati alle funzionalità interne.
2) Special Function Registers associati alle funzionalità periferiche.

I registri speciali, che servono per la gestione delle periferiche, sono implementati nelle vicinanze delle stesse periferiche a cui fanno riferimento.

Una tabella con la mappa degli SFR e riportata di seguito:

SPECIAL FUNCTION REGISTER MAP

Address	Name	Address	Name	Address	Name	Address	Name
FFFh	TOSU	FDFh	INDF2(3)	FBFh	CCPR1H	F9Fh	IPR1
FFEh	TOSH	FDEh	POSTINC2(3)	FBEh	CCPR1L	F9Eh	PIR1
FFDh	TOSL	FDDh	POSTDEC2(3)	FBDh	CCP1CON	F9Dh	PIE1
FFCh	STKPTR	FDCh	PREINC2(3)	FBCh	CCPR2H	F9Ch	—
FFBh	PCLATU	FDBh	PLUSW2(3)	FBBh	CCPR2L	F9Bh	—
FFAh	PCLATH	FDAh	FSR2H	FBAh	CCP2CON	F9Ah	—
FF9h	PCL	FD9h	FSR2L	FB9h	—	F99h	—
FF8h	TBLPTRU	FD8h	STATUS	FB8h	—	F98h	—
FF7h	TBLPTRH	FD7h	TMR0H	FB7h	—	F97h	—
FF6h	TBLPTRL	FD6h	TMR0L	FB6h	—	F96h	TRISE(2)
FF5h	TABLAT	FD5h	T0CON	FB5h	—	F95h	TRISD(2)
FF4h	PRODH	FD4h	—	FB4h	—	F94h	TRISC
FF3h	PRODL	FD3h	OSCCON	FB3h	TMR3H	F93h	TRISB
FF2h	INTCON	FD2h	LVDCON	FB2h	TMR3L	F92h	TRISA
FF1h	INTCON2	FD1h	WDTCON	FB1h	T3CON	F91h	—
FF0h	INTCON3	FD0h	RCON	FB0h	—	F90h	—
FEFh	INDF0(3)	FCFh	TMR1H	FAFh	SPBRG	F8Fh	—
FEEh	POSTINC0(3)	FCEh	TMR1L	FAEh	RCREG	F8Eh	—
FEDh	POSTDEC0(3)	FCDh	T1CON	FADh	TXREG	F8Dh	LATE(2)
FECh	PREINC0(3)	FCCh	TMR2	FACh	TXSTA	F8Ch	LATD(2)
FEBh	PLUSW0(3)	FCBh	PR2	FABh	RCSTA	F8Bh	LATC
FEAh	FSR0H	FCAh	T2CON	FAAh	—	F8Ah	LATB
FE9h	FSR0L	FC9h	SSPBUF	FA9h	EEADR	F89h	LATA
FE8h	WREG	FC8h	SSPADD	FA8h	EEDATA	F88h	—
FE7h	INDF1(3)	FC7h	SSPSTAT	FA7h	EECON2	F87h	—
FE6h	POSTINC1(3)	FC6h	SSPCON1	FA6h	EECON1	F86h	—
FE5h	POSTDEC1(3)	FC5h	SSPCON2	FA5h	—	F85h	—
FE4h	PREINC1(3)	FC4h	ADRESH	FA4h	—	F84h	PORTE(2)
FE3h	PLUSW1(3)	FC3h	ADRESL	FA3h	—	F83h	PORTD(2)
FE2h	FSR1H	FC2h	ADCON0	FA2h	IPR2	F82h	PORTC
FE1h	FSR1L	FC1h	ADCON1	FA1h	PIR2	F81h	PORTB
FE0h	BSR	FC0h	—	FA0h	PIE2	F80h	PORTA

Tabella 3.4: Mappa degli Special Function Registers.

Lo Status Register

Lo STATUS Register ha lo scopo di contenere lo stato aritmetico dell'ALU; esso può essere la destinazione per alcune istruzioni, proprio come accade per tutti gli altri registri di uso comune.

Se ad esempio un'istruzione ha effetto sui bit Z, DC, C, OV ed N dello Status Register, allora viene disabilitata la possibilità di scrivere o cancellare questi cinque bit. Essi vengono settati o cancellati in base alla logica del dispositivo. Per questo motivo un'istruzione sullo Status Register può avere un risultato diverso da quello previsto. Rifacendoci, ad esempio, al caso precedente in cui i cinque bit meno significativi sono già riservati per l'esecuzione di un'istruzione, se viene dato il comando "CLRF STATUS", che dovrebbe cancellare tutti gli otto bit dello Status Register, esso cancellerà i primi tre bit (quelli non ancora riservati all'esecuzione di alcuna istruzione) e setterà il bit Z al valore uno.

Avremo così lo Status Register nella seguente Forma:

0 0 0 U U U 1 U U

Dove U sta per Unchanged (Invariato).

Per questo motivo è raccomandabile usare solo istruzioni come BCF, RSF, SWAPF, MOVFF e MOVWF per alterare lo Status Register, in quanto non hanno alcun effetto sui bit Z, C, DC, OV e N.

Nella figura seguente viene fornita una rappresentazione dello Status Register:

STATUS REGISTER

U-0	U-0	U-0	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
—	—	—	N	OV	Z	DC	C
bit 7							bit 0

bit 7-5 **Unimplemented**: Read as '0'

bit 4 **N**: Negative bit
This bit is used for signed arithmetic (2's complement). It indicates whether the result was negative (ALU MSB = 1).
1 = Result was negative
0 = Result was positive

bit 3 **OV**: Overflow bit
This bit is used for signed arithmetic (2's complement). It indicates an overflow of the 7-bit magnitude, which causes the sign bit (bit7) to change state.
1 = Overflow occurred for signed arithmetic (in this arithmetic operation)
0 = No overflow occurred

bit 2 **Z**: Zero bit
1 = The result of an arithmetic or logic operation is zero
0 = The result of an arithmetic or logic operation is not zero

bit 1 **DC**: Digit carry/$\overline{\text{borrow}}$ bit
For `ADDWF`, `ADDLW`, `SUBLW` and `SUBWF` instructions
1 = A carry-out from the 4th low order bit of the result occurred
0 = No carry-out from the 4th low order bit of the result

Note: For $\overline{\text{borrow}}$, the polarity is reversed. A subtraction is executed by adding the two's complement of the second operand. For rotate (`RRF`, `RLF`) instructions, this bit is loaded with either the bit 4 or bit 3 of the source register.

bit 0 **C**: Carry/$\overline{\text{borrow}}$ bit
For `ADDWF`, `ADDLW`, `SUBLW` and `SUBWF` instructions
1 = A carry-out from the Most Significant bit of the result occurred
0 = No carry-out from the Most Significant bit of the result occurred

Note: For $\overline{\text{borrow}}$, the polarity is reversed. A subtraction is executed by adding the two's complement of the second operand. For rotate (`RRF`, `RLF`) instructions, this bit is loaded with either the high or low order bit of the source register.

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

Fig. 3.7: Rappresentazione dello Status Register.

Gli Interrupt

I PIC della famiglia 18FXX2 hanno molte possibili sorgenti d'interrupt e un meccanismo che consente di assegnare a ciascuna possibile sorgente d'interrupt un livello di priorità alto o basso.

Il vettore degli interrupt ad alta priorità si trova mappato in memoria all'indirizzo '000008h', mentre il vettore delle interrupt a bassa priorità all'indirizzo '000018h'.

Il verificarsi di un'interrupt con alta priorità prevarrà su tutte le interrupt a bassa priorità che possono già essere state accolte.

Esistono dieci registri usati per il controllo dell'operazione d'interrupt e sono:

- RCON
- INTCON
- INTCON2
- INTCON3
- PIR1, PIR2
- PIE1, PIE2
- IPR1, IPR2

Ogni sorgente d'interrupt, tranne INTO, prevede l'uso di tre bit per il controllo, ognuno dei quali ha una specifica funzione:

- Flag bit per indicare il verificarsi di un evento d' interrupt
- Enable bit che consente all'esecuzione del programma di saltare all'indirizzo del vettore d'interrupt quando è

stato settato il Flag bit

- Priority bit per selezionare un livello di priorità alto o basso

Il livello di priorità viene abilitato settando il bit IPEN (bit 7 di RCON). Quando il livello di priorità è abilitato, ci sono due bit che abilitano gli interrupt globalmente. Settando il bit GEH (bit 7 di INTCON) si abilitano tutti gli interrupt che hanno il bit di priorità fissato, mentre settando il bit GIEL (bit 6 di INTCON) si abilitano gli interrupt che non hanno priorità fissata.

Quando sono settati il Flag bit, l'Enable bit e uno dei bit d'abilitazione globale degli interrupt, il vettore d'interrupt viene direttamente indirizzato all'indirizzo '000008h' oppure '000018h' in base al livello di priorità. Gli interrupt singoli possono essere disabilitati mediante il loro corrispondente bit d'abilitazione.

Quando viene accolta una richiesta d'interrupt, il bit di abilitazione globale viene cancellato per disabilitare ulteriori interrupt. Il valore di ritorno viene memorizzato nello stack e l'indirizzo del vettore d'interrupt ('000008h' oppure '000018h') è caricato nel PC (Program Counter).

Una volta entrati nella routine di servizio dell'interrupt, le sorgenti dell'interrupt possono essere determinate interrogando i flag bit. I flag bit

dell'interrupt devono essere cancellati prima di riabilitare gli interrupt per evitare interrupt ricorsivi.

L'istruzione di ritorno dell'interrupt, "RETFIE", esce dalla routine d'interrupt e setta il bit GIE (GIEH o GIEL se sono usati i livelli di priorità), che riabilita gli interrupt.

Viene di seguito riportata una figura relativa alla logica degli interrupt:

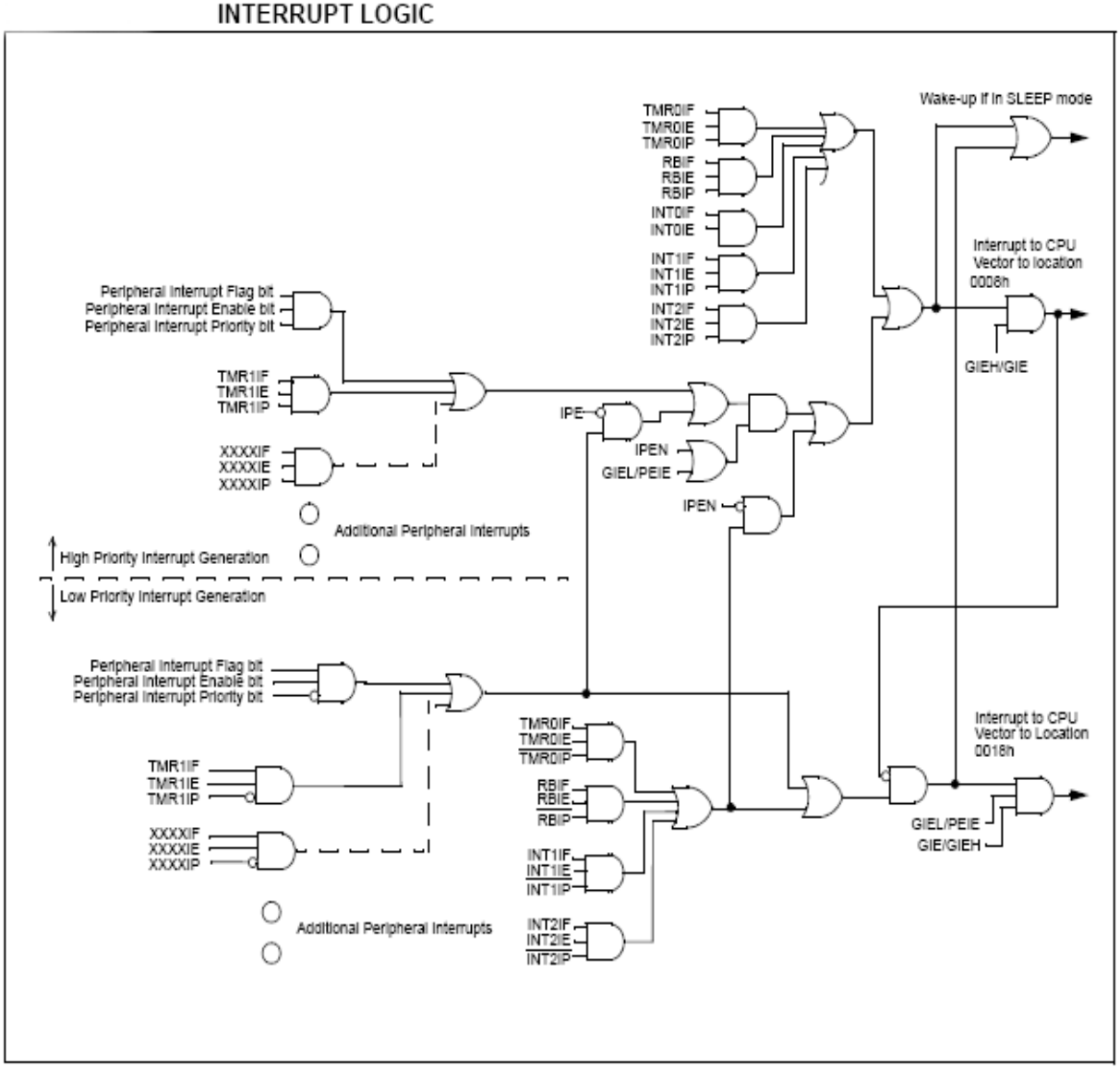


Fig. 3.8: Logica degli Interrupt.

Porte di I/O

Consideriamo le interfacce di I/O del PIC. Innanzi tutto le considereremo per comodità come semplici porte logiche in cui i valori a cui si possono portare sono 0 e 1 logici. In realtà il valore logico 1 è relativo alla tensione +5V mentre lo 0 ai 0V. Sapendo questo si può pensare di collegare opportunamente ai piedini di output un led che si accenderà quando ai piedini arriva un 1 logico, oppure un interruttore ai piedini di input che comunicheranno uno 0 logico quando si chiuderà il contatto.

In funzione del dispositivo scelto il numero delle porte disponibili varierà; per i PIC della famiglia 18FXXX si va da un massimo di cinque ad un minimo di tre. Alcuni pin delle porte di I/O sono multiplexati con varie funzioni dalle varie funzionalità periferiche presenti nel dispositivo.

Ogni porta possiede tre registri per gestire le sue funzionamento:

- TRIS register (data direction register)
- PORT register (reads the levels on the pins of the device)
- LAT register (output latch)

Il TRIS register serve per tenere le informazioni sullo stato d'utilizzo della porta, cioè se i singoli piedini sono usati come Input (indicati nel registro attraverso un 1) o come Output (indicati con uno 0); il PORT register

mantiene traccia del dato presente nella porta, leggendo il suo contenuto, infatti, si legge lo stato del pin ad esso associato, scrivendo in esso si setta lo stato alto o basso del pin corrispondente. Il LAT register sono anche mappati in memoria per cui operazioni di lettura, scrittura o cancellazione di tali registri equivale ad operare direttamente con i pin ad essi associati.

Ad esempio, se il dato presente nel registro TRISB risulta essere ' b.00001111', avremo che i piedini della porta B saranno settati metà come Output (0) e metà come Input (1): nel caso in esame, i piedini RB4, RB5, RB6, RB7 saranno output, mentre RB0, RB1, RB2, RB3 saranno Input.

Considerando la stessa porta si può vedere che se il dato presente nel registro PORTB fosse 'b.00001111' avremo che il PIC sta ricevendo, dai piedini d'ingresso, degli "1" e sta trasmettendo degli"0".

Di seguito viene riportata una tabella riassuntiva con le principali caratteristiche delle porte presenti su un PIC18F452 o simili:

Porta	Numero piedini	Registro definizione della direzione del dato (I/O)	Registro contenente il dato presente sulla porta	LAT Register
PORTA	**7**	**TRISA**	**PORTA**	**LATA**
PORTB	**8**	**TRISB**	**PORTB**	**LATB**
PORTC	**8**	**TRISC**	**PORTC**	**LATC**
PORTD	**8**	**TRISD**	**PORTD**	**LATD**
PORTE	**3**	**TRISE**	**PORTE**	**LATE**

Tabella 3.5: Porte di I/O e rispettivi registri.

Le cinque porte del PIC rappresentano le uniche risorse che questo dispositivo ha a disposizione per comunicare con il mondo esterno, cioè con tutti gli altri dispositivi elettronici ad esso collegati. Ogni porta del dispositivo per funzionare ha bisogno di componenti hardware abbastanza complessi; molto spesso i singoli pin possono essere programmati per svolgere più funzionalità. Vengono riportate di seguito tutte e cinque le porte del dispositivo, ponendo l'attenzione su quelle che sono le possibili funzionalità dei pin.

Porta A

Name	Bit#	Buffer	Function
RA0/AN0	bit0	TTL	Input/output or analog input.
RA1/AN1	bit1	TTL	Input/output or analog input.
RA2/AN2/VREF-	bit2	TTL	Input/output or analog input or VREF-.
RA3/AN3/VREF+	bit3	TTL	Input/output or analog input or VREF+.
RA4/T0CKI	bit4	ST	Input/output or external clock input for Timer0. Output is open drain type.
RA5/$\overline{SS}$/AN4/LVDIN	bit5	TTL	Input/output or slave select input for synchronous serial port or analog input, or low voltage detect input.
OSC2/CLKO/RA6	bit6	TTL	OSC2 or clock output or I/O pin.

Legend: TTL = TTL input, ST = Schmitt Trigger input

Tabella 3.6: Funzionalità della porta A.

Porta B

Name	Bit#	Buffer	Function
RB0/INT0	bit0	TTL/ST(1)	Input/output pin or external interrupt input0. Internal software programmable weak pull-up.
RB1/INT1	bit1	TTL/ST(1)	Input/output pin or external interrupt input1. Internal software programmable weak pull-up.
RB2/INT2	bit2	TTL/ST(1)	Input/output pin or external interrupt input2. Internal software programmable weak pull-up.
RB3/CCP2(3)	bit3	TTL/ST(4)	Input/output pin or Capture2 input/Compare2 output/PWM output when CCP2MX configuration bit is enabled. Internal software programmable weak pull-up.
RB4	bit4	TTL	Input/output pin (with interrupt-on-change). Internal software programmable weak pull-up.
RB5/PGM(5)	bit5	TTL/ST(2)	Input/output pin (with interrupt-on-change). Internal software programmable weak pull-up. Low voltage ICSP enable pin.
RB6/PGC	bit6	TTL/ST(2)	Input/output pin (with interrupt-on-change). Internal software programmable weak pull-up. Serial programming clock.
RB7/PGD	bit7	TTL/ST(2)	Input/output pin (with interrupt-on-change). Internal software programmable weak pull-up. Serial programming data.

Legend: TTL = TTL input, ST = Schmitt Trigger input

Tabella 3.7: Funzionalità della porta B.

Porta C

Name	Bit#	Buffer Type	Function
RC0/T1OSO/T1CKI	bit0	ST	Input/output port pin or Timer1 oscillator output/Timer1 clock input.
RC1/T1OSI/CCP2	bit1	ST	Input/output port pin, Timer1 oscillator input, or Capture2 input/ Compare2 output/PWM output when CCP2MX configuration bit is set.
RC2/CCP1	bit2	ST	Input/output port pin or Capture1 input/Compare1 output/PWM1 output.
RC3/SCK/SCL	bit3	ST	RC3 can also be the synchronous serial clock for both SPI and I^2C modes.
RC4/SDI/SDA	bit4	ST	RC4 can also be the SPI Data In (SPI mode) or Data I/O (I^2C mode).
RC5/SDO	bit5	ST	Input/output port pin or Synchronous Serial Port data output.
RC6/TX/CK	bit6	ST	Input/output port pin, Addressable USART Asynchronous Transmit, or Addressable USART Synchronous Clock.
RC7/RX/DT	bit7	ST	Input/output port pin, Addressable USART Asynchronous Receive, or Addressable USART Synchronous Data.

Legend: ST = Schmitt Trigger input

Tabella 3.8: Funzionalità della porta C.

Porta D

Name	Bit#	Buffer Type	Function
RD0/PSP0	bit0	ST/TTL(1)	Input/output port pin or parallel slave port bit0.
RD1/PSP1	bit1	ST/TTL(1)	Input/output port pin or parallel slave port bit1.
RD2/PSP2	bit2	ST/TTL(1)	Input/output port pin or parallel slave port bit2.
RD3/PSP3	bit3	ST/TTL(1)	Input/output port pin or parallel slave port bit3.
RD4/PSP4	bit4	ST/TTL(1)	Input/output port pin or parallel slave port bit4.
RD5/PSP5	bit5	ST/TTL(1)	Input/output port pin or parallel slave port bit5.
RD6/PSP6	bit6	ST/TTL(1)	Input/output port pin or parallel slave port bit6.
RD7/PSP7	bit7	ST/TTL(1)	Input/output port pin or parallel slave port bit7.

Legend: ST = Schmitt Trigger input, TTL = TTL input

Note 1: Input buffers are Schmitt Triggers when in I/O mode and TTL buffer when in Parallel Slave Port mode.

Tabella 3.9: Funzionalità della porta D.

Porta E

Name	Bit#	Buffer Type	Function
RE0/$\overline{RD}$/AN5	bit0	ST/TTL(1)	Input/output port pin or read control input in Parallel Slave Port mode or analog input: $\overline{RD}$ 1 = Not a read operation 0 = Read operation. Reads PORTD register (if chip selected).
RE1/$\overline{WR}$/AN6	bit1	ST/TTL(1)	Input/output port pin or write control input in Parallel Slave Port mode or analog input: $\overline{WR}$ 1 = Not a write operation 0 = Write operation. Writes PORTD register (if chip selected).
RE2/$\overline{CS}$/AN7	bit2	ST/TTL(1)	Input/output port pin or chip select control input in Parallel Slave Port mode or analog input: $\overline{CS}$ 1 = Device is not selected 0 = Device is selected

Legend: ST = Schmitt Trigger input, TTL = TTL input

Note 1: Input buffers are Schmitt Triggers when in I/O mode and TTL buffers when in Parallel Slave Port mode.

Tabella 3.10: Funzionalità della porta E.

Convertitore A/D

Il convertitore Analogico/Digitale prevede cinque ingressi per i PIC della famiglia 18F2X2 e otto per la famiglia 18F4X2. Questo convertitore A/D a 10 bit consente di convertire un segnale analogico presente su queste porte in un corrispondente valore digitale a 10 bit.

Di seguito viene riportato uno schema a blocchi del convertitore:

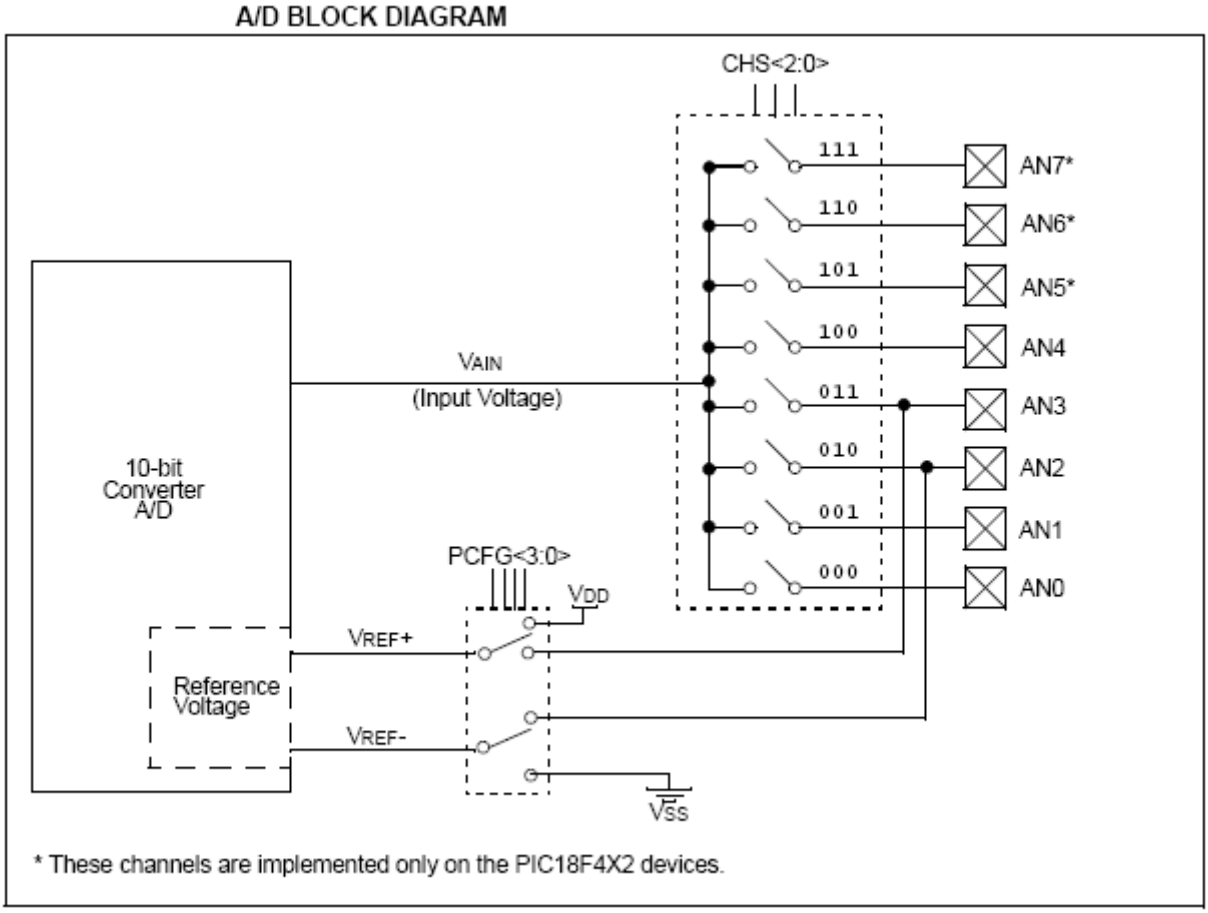


Fig. 3.9: Schema a blocchi del convertitore A/D.

Come possiamo notare il convertitore è unico, mentre a monte di esso è presente un MUX analogico necessario per settare gli ingressi analogici opportuni (5 ingressi per il PIC a 28 pin 8 ingressi per il PIC a 40 pin).

Il convertitore A/D per poter funzionare necessita di quattro registri:

- A/D Result High Register (ADRESH)
- A/D Result Low Register (ADRESL)
- A/D Control Register 0 (ADCON0)

• A/D Control Register 1 (ADCON1)

Il registro ADCON0, mostrato sotto, controlla il funzionamento del convertitore.

ADCON0 REGISTER

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	U-0	R/W-0
ADCS1	ADCS0	CHS2	CHS1	CHS0	GO/$\overline{\text{DONE}}$	—	ADON
bit 7							bit 0

Fig. 3.10: Registro ADCON0.

I bit sette e sei, denominati “ADCS1” e “ADCS0”, insieme al bit sei del registro ADCON1, servono a selezionare una delle otto possibili sorgenti di clock da usare per la conversione, secondo la tabella riportata sotto:

ADCON1 <ADCS2>	ADCON0 <ADCS1:ADCS0>	Clock Conversion
0	00	Fosc/2
0	01	Fosc/8
0	10	Fosc/32
0	11	FRC (clock derived from the internal A/D RC oscillator)
1	00	Fosc/4
1	01	Fosc/16
1	10	Fosc/64
1	11	FRC (clock derived from the internal A/D RC oscillator)

Tabella 3.11: Corrispondenza bit ADCS1&ADCS0_sorgente di clock.

I bit da cinque a tre (“CHS2”, “CHS1” e “CHS0”) servono a stabilire quali pin della porta debbano essere convertiti:

CHS2:CHS0: Analog Channel Select bits
000 = channel 0, (AN0)
001 = channel 1, (AN1)
010 = channel 2, (AN2)
011 = channel 3, (AN3)
100 = channel 4, (AN4)
101 = channel 5, (AN5)
110 = channel 6, (AN6)
111 = channel 7, (AN7)

Fig. 3.11: Bit CHS2-CHS0 per la selezione dei canali analogici da convertire.

Il bit due di ADCON0 (‘GO/DONE’) indica lo stato del convertitore A/D:

GO/DONE=1: la conversione analogico/digitale e in fase d’esecuzione (settando a 1 questo bit parte la conversione). Il bit viene portato a zero tramite hardware quando la conversione è finita.

GO/DONE=0: la conversione non è in esecuzione.

Il bit zero ‘ADON’ se posto ad 1 accende il convertitore A/D, se posto a zero lo spegne. Il pin uno non ha alcuna funzione.

Il registro ADCON1 configura le funzioni dei pin della porta.

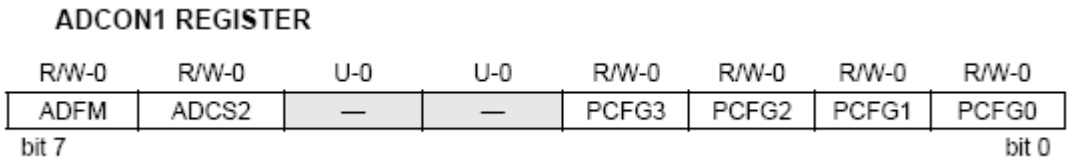


Fig. 3.12: Registro ADCON1.

Il bit sette 'ADFM' consente di selezionare il formato del risultato della conversione:

ADFM=1: giustificato a desta. I sei bit meno significativi di ADRESH

sono letti come '0'

ADFM=0: giustificato a sinistra. I sei bit meno significativi di

ADRESL sono letti come '0'

Il bit sei 'ADCS2', come gia detto prima, insieme ai bit 'ADCS1' e ADCS0' di ADCON0 servono a selezionare la sorgente di clock per il convertitore.

I bit da tre a zero, ‘PCFG3’, ‘PCFG2’, ‘PCFG1’ e ‘PCFG0’, controllano la configurazione delle porte connesse all’A/D converter secondo la seguente tabella:

PCFG3:PCFG0: A/D Port Configuration Control bits

PCFG <3:0>	AN7	AN6	AN5	AN4	AN3	AN2	AN1	AN0	VREF+	VREF-	C / R
0000	A	A	A	A	A	A	A	A	VDD	VSS	8 / 0
0001	A	A	A	A	VREF+	A	A	A	AN3	VSS	7 / 1
0010	D	D	D	A	A	A	A	A	VDD	VSS	5 / 0
0011	D	D	D	A	VREF+	A	A	A	AN3	VSS	4 / 1
0100	D	D	D	D	A	D	A	A	VDD	VSS	3 / 0
0101	D	D	D	D	VREF+	D	A	A	AN3	VSS	2 / 1
011x	D	D	D	D	D	D	D	D	—	—	0 / 0
1000	A	A	A	A	VREF+	VREF-	A	A	AN3	AN2	6 / 2
1001	D	D	A	A	A	A	A	A	VDD	VSS	6 / 0
1010	D	D	A	A	VREF+	A	A	A	AN3	VSS	5 / 1
1011	D	D	A	A	VREF+	VREF-	A	A	AN3	AN2	4 / 2
1100	D	D	D	A	VREF+	VREF-	A	A	AN3	AN2	3 / 2
1101	D	D	D	D	VREF+	VREF-	A	A	AN3	AN2	2 / 2
1110	D	D	D	D	D	D	D	A	VDD	VSS	1 / 0
1111	D	D	D	D	VREF+	VREF-	D	A	AN3	AN2	1 / 2

A = Analog input D = Digital I/O
C/R = # of analog input channels / # of A/D voltage references

Tabella 3.12: Bit PCFG3-PCFG0 per la configurazione dei pin d’ingresso.

I registri ADRESH e ADREL contengono il risultato del processo di conversione.

Quando un processo di conversione viene ultimato il risultato del processo viene caricato in ADRESH e ADREL, il bit due di ADCON0 (‘GO/DONE’) si abbassa e il flag degli interrupt generati dal convertitore A/D (ADIF) viene alzato.

Il convertitore A/D è composto, internamente, da un sample&hold con in cascata un convertitore ad approssimazioni successive, la cui risoluzione ottimale è di 10 bit. La capacità del circuito di sample&hold è di 120pF. Nella figura seguente abbiamo il modello analogico della sezione d'ingresso.

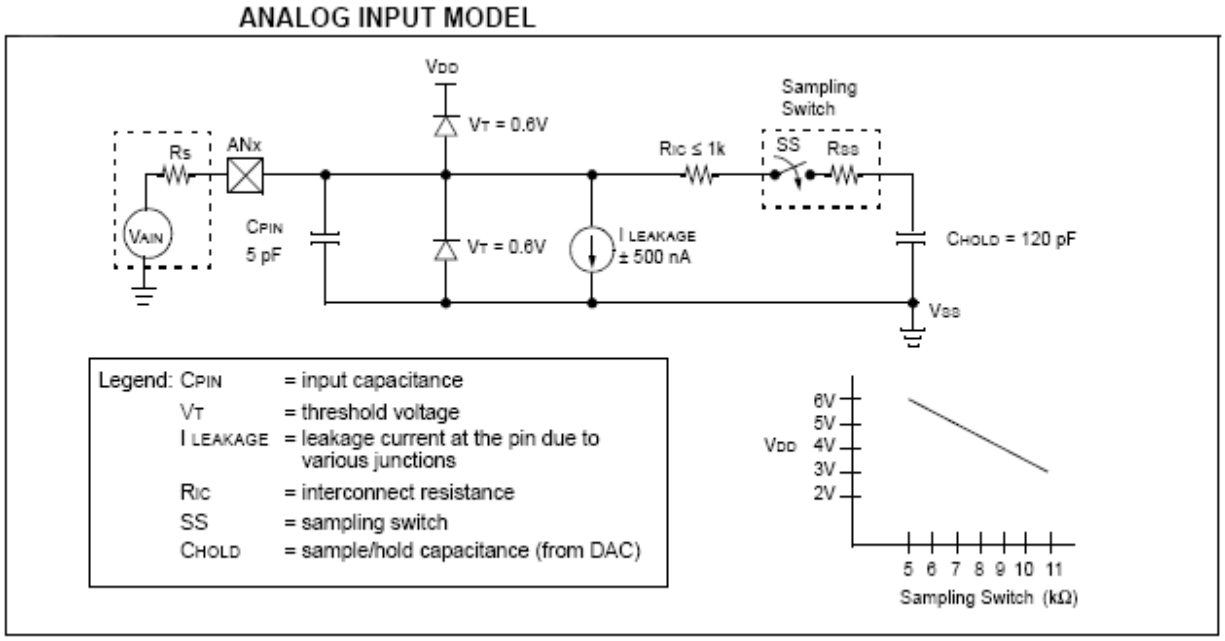


Fig. 3.13: Modello analogico d'ingresso dell'A/D converter.

Affinché la conversione sia fatta correttamente bisogna attendere che la capacità (CHOLD) venga caricata fino al valore di tensione in ingresso. Il tempo di carica della capacità CHOLD viene influenzato sia dalla resistenza della sorgente analogica (Rs) che dall'impedenza interna del circuito di Sampling Switch (Rss). Quest'ultima cambia in funzione della temperatura e

della tensione Vdd. L'impedenza della sorgente analogica influenza il valore della tensione di offset in ingresso, a causa della corrente di Leakage.

Se si considera il massimo valore della resistenza della sorgente analogica da convertire pari a 2,5 KΩ, si può calcolare il tempo minimo di conversione. Si assume che il massimo errore accettabile affinché la risoluzione sia quella indicata, sia pari a 1/2 LSB. In queste condizioni si ha che il tempo minimo d'acquisizione è pari a 11,86 μs.

Il tempo necessario alla conversione di un singolo bit viene definito come T_{AD}; per eseguire un'operazione di conversione a 10 bit sono necessari 12 T_{AD}.

Sono previste sette sorgenti di clock da utilizzare per la conversione:

- 2 TOSC
- 4 TOSC
- 8 TOSC
- 16 TOSC
- 32 TOSC
- 64 TOSC
- Internal A/D module RC oscillator (2-6 μs)

Affinché l'operazione di conversione avvenga in modo corretto è necessario selezionare il clock del convertitore A/D in modo da assicurare un T_{AD} (tempo necessario alla conversione di un bit) superiore a 1,6 μs.

La tabella seguente riassume la corrispondenza tra una delle sette possibili sorgenti di clock selezionabili e il corrispondente T_{AD}:

TAD vs. DEVICE OPERATING FREQUENCIES

AD Clock Source (TAD)		Maximum Device Frequency	
Operation	ADCS2:ADCS0	PIC18FXX2	PIC18LFXX2
2 Tosc	000	1.25 MHz	666 kHz
4 Tosc	100	2.50 MHz	1.33 MHz
8 Tosc	001	5.00 MHz	2.67 MHz
16 Tosc	101	10.00 MHz	5.33 MHz
32 Tosc	010	20.00 MHz	10.67 MHz
64 Tosc	110	40.00 MHz	21.33 MHz
RC	011	—	—

Tabella 3.13: Corrispondenza tra le sorgenti di clock selezionate e T_{AD}.

Il risultato dell'operazione di conversione viene caricato nella coppia di registri ADRESH – ADRESL; ognuno di questi registri ha otto bit per un totale di 2 X 8 = 16 bit.

I PIC consentono di scegliere se il risultato debba essere caricato nella coppia di registri giustificandolo a destra o a sinistra. Nella figura seguente sono riportati due esempi, uno con giustificazione a destra e l'altro a sinistra. In entrambi i casi tutti i bit non utilizzati per contenere il risultato dalla conversione vengono posti a '0'.

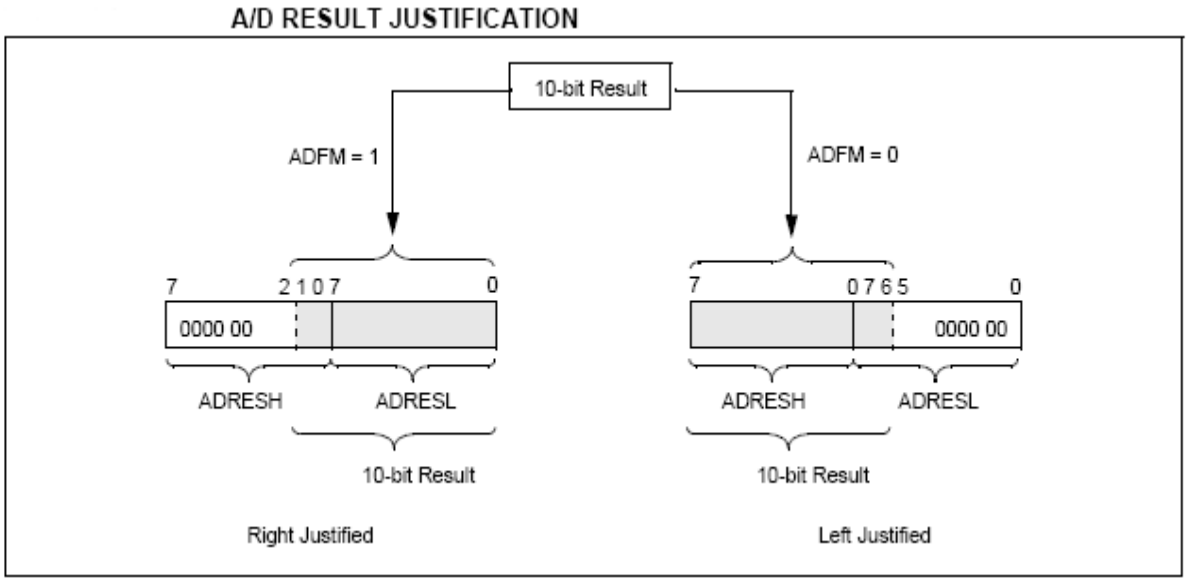


Fig. 3.14: Risultato della conversione con giustificazione a destra o a sinistra.

Comunicazione Seriale

I PIC della Microchip appartenenti alla famiglia 18FXX2 contengono al loro interno un modulo elettronico capace di implementare una comunicazione seriale. Tale modulo prende il nome di Universal Synchronous Asynchronous Receiver Transmitter (USART) e viene spesso indicato anche come Serial Communications Interface (SCI).

L' USART può essere configurata come un'interfaccia asincrona full-duplex o come un'interfaccia sincrona half-duplex.

Quindi i tre possibili modi di configurare l'USART sono i seguenti:

- Asynchronous (full-duplex)
- Synchronous - Master (half-duplex)
- Synchronous - Slave (half-duplex)

Se si configura come Asynchronous (full-duplex) allora il PIC può usare l'USART per comunicare con un qualsiasi device periferico che implementa tale standard, come ad esempio un comune PC. Bisogna però ricordare che la porta RS232, che comunemente viene usata sui PC per comunicazioni seriali, associa al livello logico alto il valore +12V e al livello logico basso il valore 0V (per maggiori dettagli si rimanda alle specifiche dello standard), mentre le porte del pic, anche quelle utilizzate per la comunicazione seriale, associano al livello logico alto il valore +5V e al valore logico basso 0V.

Quindi risulta praticamente impossibile collegare direttamente un PC tramite la propria RS232 all'interfaccia seriale implementata dal microcontrollore. Esistono in commercio dei dispositivi integrati (ad esempio il MAX232) che si occupano di convertire i segnali RS232 di 12V in segnali di 5V, compatibili con le porte logiche in tecnologia TTL che realizzano l'interfaccia seriale dei pic.

Per configurare il pin RC6/TX/CK ed il pin RC7/RX/DT come USART bisogna seguire il seguente procedimento:

• bit SPEN (RCSTA<7>) deve essere settato (= 1)

• bit TRISC<6> deve essere abbassato (= 0)

• bit TRISC<7> deve essere settato (=1)

Il registro TXSTA serve per controllare la trasmissione e per contenere informazioni sul suo stato.

Il registro RCSTA per controllare la ricezione e per contenere informazioni sul suo stato.

TXSTA: TRANSMIT STATUS AND CONTROL REGISTER

R/W-0	R/W-0	R/W-0	R/W-0	U-0	R/W-0	R-1	R/W-0
CSRC	TX9	TXEN	SYNC	—	BRGH	TRMT	TX9D
bit 7							bit 0

bit 7 **CSRC:** Clock Source Select bit
Asynchronous mode:
Don't care
Synchronous mode:
1 = Master mode (clock generated internally from BRG)
0 = Slave mode (clock from external source)

bit 6 **TX9**: 9-bit Transmit Enable bit
1 = Selects 9-bit transmission
0 = Selects 8-bit transmission

bit 5 **TXEN**: Transmit Enable bit
1 = Transmit enabled
0 = Transmit disabled
Note: SREN/CREN overrides TXEN in SYNC mode.

bit 4 **SYNC**: USART Mode Select bit
1 = Synchronous mode
0 = Asynchronous mode

bit 3 **Unimplemented:** Read as '0'

bit 2 **BRGH**: High Baud Rate Select bit
Asynchronous mode:
1 = High speed
0 = Low speed
Synchronous mode:
Unused in this mode

bit 1 **TRMT**: Transmit Shift Register Status bit
1 = TSR empty
0 = TSR full

bit 0 **TX9D:** 9th bit of Transmit Data
Can be Address/Data bit or a parity bit.

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

Fig. 3.15: Registro TXSTA per il controllo e la gestione della trasmissione.

RCSTA: RECEIVE STATUS AND CONTROL REGISTER

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R-0	R-0	R-x
SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D
bit 7							bit 0

bit 7 **SPEN:** Serial Port Enable bit
1 = Serial port enabled (configures RX/DT and TX/CK pins as serial port pins)
0 = Serial port disabled

bit 6 **RX9**: 9-bit Receive Enable bit
1 = Selects 9-bit reception
0 = Selects 8-bit reception

bit 5 **SREN**: Single Receive Enable bit
Asynchronous mode:
Don't care
Synchronous mode - Master:
1 = Enables single receive
0 = Disables single receive
This bit is cleared after reception is complete.
Synchronous mode - Slave:
Don't care

bit 4 **CREN**: Continuous Receive Enable bit
Asynchronous mode:
1 = Enables receiver
0 = Disables receiver
Synchronous mode:
1 = Enables continuous receive until enable bit CREN is cleared (CREN overrides SREN)
0 = Disables continuous receive

bit 3 **ADDEN**: Address Detect Enable bit
Asynchronous mode 9-bit (RX9 = 1):
1 = Enables address detection, enable interrupt and load of the receive buffer when RSR<8> is set
0 = Disables address detection, all bytes are received, and ninth bit can be used as parity bit

bit 2 **FERR**: Framing Error bit
1 = Framing error (can be updated by reading RCREG register and receive next valid byte)
0 = No framing error

bit 1 **OERR**: Overrun Error bit
1 = Overrun error (can be cleared by clearing bit CREN)
0 = No overrun error

bit 0 **RX9D:** 9th bit of Received Data
This can be Address/Data bit or a parity bit, and must be calculated by user firmware.

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

Fig. 3.16: Registro RCSTA per il controllo e la gestione della ricezione.

Low Voltage Detect

Quando ci si trova a progettare un'apparecchiatura elettronica si sente speso la necessità di avere a disposizione un dispositivo che sia in grado di verificare se la tensione d'alimentazione V_{DD} sia scesa al di sotto di un valore prestabilito. I PIC della famiglia 18FXX2 implementano al loro interno questa funzionalità che può risultare molto utile; ad esempio quando ci si accorge che la tensione d'alimentazione sta scendendo, prima che raggiunga un valore al di sotto del quale il dispositivo non può più funzionare, è possibile settare il dispositivo in modo che consumi il meno possibile in modo da prolungare il suo funzionamento.

Il modulo Low Voltage Detect dei pic e completamente programmabile via software quindi è molto flessibile. Quando il valore della tensione d'alimentazione scende al di sotto del valore impostato viene attivato un flag d'interrupt. Se gli interrupt sono abilitati l'esecuzione del programma può saltare all'indirizzo del vettore degli interrupt e servirla.

Nella figura seguente viene riportata una situazione in cui è possibile utilizzare il Low Voltage Detect (LVD). Si presuppone che il dispositivo sia alimentato da delle batterie e che la tensione d'alimentazione stia scendendo, cioè che le batterie si stiano scaricando. Quando la tensione arriva al valore V_A impostato, il LVD se n'accorge e genera una richiesta d'interrupt; quest'interrupt può, ad esempio, spegnere il dispositivo prima che la tensione raggiunga il valore V_B che è critico per il funzionamento dell'apparecchiatura.

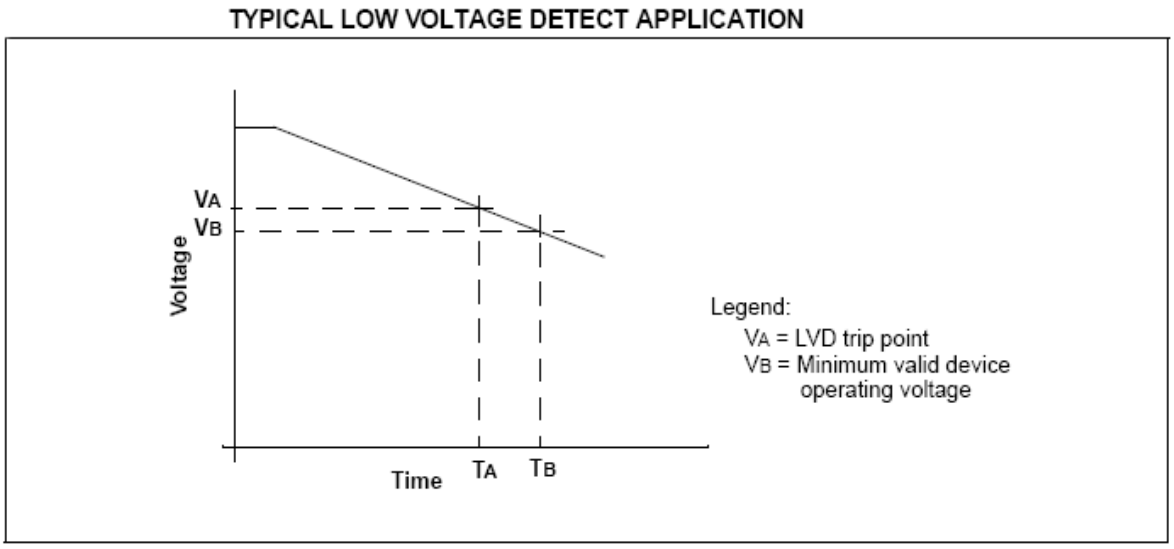


Fig. 3.17: Esempio d'utilizzo del Low Voltage Detect.

Di seguito viene riportato lo schema a blocchi del LVD. In esso si possono riconoscere vari elementi:

- un partitore resistivo con 16 livelli (i possibili livelli impostabili per il LVD)
- un comparatore
- un riferimento di tensione interno (1,2 V)
- un multiplexer a 16 ingressi
- un registro di controllo per selezionare il livello di soglia del LVD

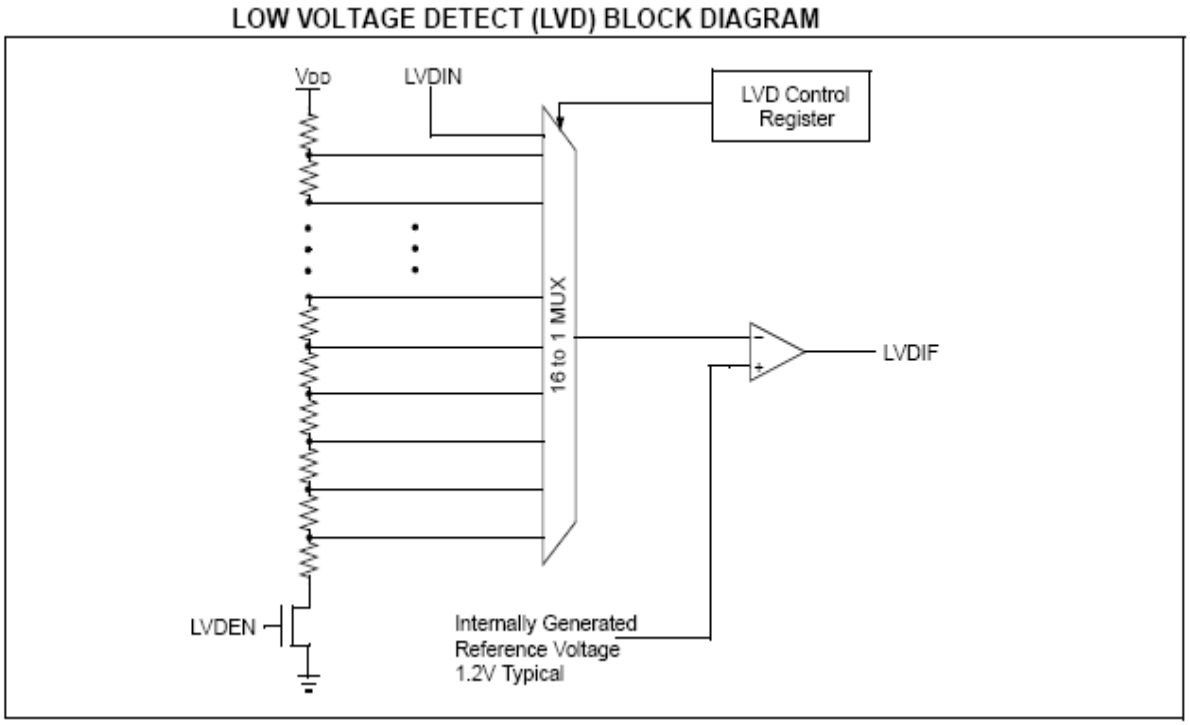


Fig. 3.18: Schema a blocchi del Low Voltage Detect.

La tensione d'alimentazione V_{DD} viene mandata ad un partitore resistivo che genera a6 livelli di tensione da inviare al multiplexer a 16 ingressi. Tramite il LVD Control Register si può scegliere quale di questi 16 riferimenti debba essere inviato al comparatore. Quando il riferimento scelto diventa più basso della tensione di riferimento (1,2V) l'uscita del comparatore si alza (LVDIF) settando l'opportuno flag d'interrupt.

Il LVD prevede anche la possibilità di utilizzare come riferimento per il comparatore un ingresso supplementare esterno. Risulta quindi possibile impostare il valore del LVD a proprio piacimento.

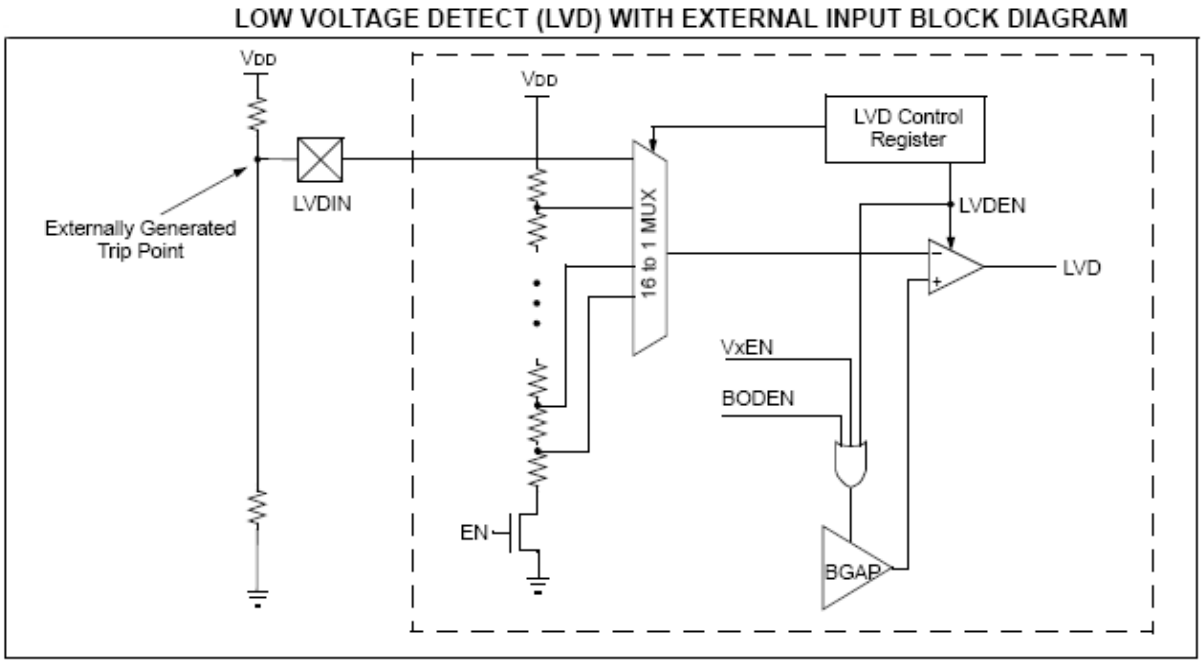


Fig. 3.19: Schema a blocchi del Low Voltage Detect con riferimento esterno.

Set d'Istruzioni

Nei precedenti paragrafi è stata descritta la struttura hardware dei microcontrollori PIC appartenenti alla famiglia 18FXX2; in questo paragrafo analizzeremo in dettaglio le istruzioni disponibili per scrivere un programma in assembler per i PIC; affronteremo quelli che sono i passi necessari per la stesura di un programma e analizzeremo come il microcontrollore si comporta di fronte alle varie istruzioni.

Il linguaggio che viene normalmente utilizzato per i microcontrollori è l'assembler; tutti i dispositivi elettronici digitali interpretano e lavorano tramite un linguaggio proprio definito dallo standard binario.

I dispositivi elettronici digitali sono progettati per riconoscere solamente due condizioni: “1”, ovvero presenza di tensione e “0”, assenza di tensione. Nello specifico lo stato logico alto on (‘1’) corrisponde nei PIC (che implementano una logica definita positiva) al valore di + 5V e lo stato logico basso off (‘0’) al valore 0V.

Dovendo scrivere un programma tramite questo tipo di linguaggio è facile immaginare a quale difficoltà si va incontro nel compilarlo e nell’interpretarlo; ecco quindi che viene in nostro soccorso un’interfaccia, il linguaggio assembler, che ci semplifica di molto il lavoro di programmazione.

Un programma scritto in assembler (che varia a seconda del tipo di microcontrollore) è costituito da una serie di frasi, definite statement (dichiarazioni), ciascuna delle quali può rappresentare una serie d’informazioni: etichette (Labels), codice operativo (spesso denominato anche mnemonico), che rappresenta in pratica le istruzioni che il PIC è in grado di eseguire, operandi, cioè gli elementi (registri, locazioni di memoria) su cui le istruzioni devono andare ad agire, commenti, cioè indicazioni che non vengono eseguite dal micro, ma che aiutano chi legge il programma ad interpretarne il significato.

Generalmente all’inizio di ogni listato viene inserita una presentazione descrittiva contenente alcune informazioni quali il nome del file, la data di realizzazione, una descrizione sommaria del contenuto, l’autore eccetera.

Le labels sono delle parole che vengono utilizzate come “rimandi” nel programma oppure delle costanti che saranno sostituite dal compilatore nella generazione del codice macchina.

Generalmente, la prima parte di un programma scritto in assembler contiene diverse labels che serviranno per semplificare la scrittura del programma stesso.

La parola EQU non è un’istruzione del PIC, ma una direttiva del compilatore, che dice appunto di associare ad una label un certo valore.

Quando il programma viene compilato dall’assemblatore, per generare il codice finale in linguaggio macchina, ogni volta che il compilatore incontra la parola a cui è associato EQU, sostituisce a questa il suo valore effettivo.

In pratica il dichiarare delle costanti in questo modo, anziché scrivere direttamente il loro valore all’interno del programma, ha due sostanziali vantaggi:

1. risulta molto più facile scrivere un programma, in quanto è più semplice utilizzare una label che ricorda il significato di una certa costante piuttosto che il suo valore numerico;

2. diventa più facile e veloce effettuare modifiche al programma.

Una parte importante del programma è costituita dai commenti: per aggiungere dei commenti, è sufficiente porre un punto e virgola prima del commento stesso; il compilatore automaticamente ignorerà tutto ciò che è scritto dopo il punto e virgola.

I commenti, anche se non servono direttamente al programma, sono tuttavia d'estrema importanza per la comprensione del programma stesso.

E' buona norma inserire commenti frequenti, ad esempio per le costanti usate nel programma, per le routine e così via. Dopo la prima parte relativa alla dichiarazione delle costanti, vi è il programma vero e proprio, che comincia dalla label INIT.

La prima istruzione che si trova è ORG 0000h. Questa, come la EQU già vista, non è un'istruzione del PIC ma una direttiva dell'assembler; in pratica "dice" all'assemblatore che la parte di programma che segue dovrà essere compilata in memoria a partire dalla locazione 0000h (in esadecimale).

Questo perché quando il PIC viene alimentato, o quando esce da una situazione di reset, il program Counter parte dalla prima locazione di memoria, che nel 16F877 è rappresentata dalla locazione di indirizzo 0.

In tale locazione è quindi necessario inserire un rimando all'inizio, per così dire, vero, del programma. A volte si fa riferimento a tale rimando parlando di vettore di reset del micro.

Dopo la ORG inizia il programma vero e proprio, con diverse istruzioni.

Quasi tutte le istruzioni sono costituite da due parti: l'istruzione vera e propria, che dice al microcontrollore il tipo d'operazione che deve essere eseguita, e gli operandi, cioè in pratica ciò su cui l'istruzione deve andare ad agire.

Prima di scrivere la sequenza d'istruzioni che compone un programma, si descrive il programma che il micro dovrà eseguire attraverso un diagramma di flusso, o flow chart, per esprimere il tipo di operazioni che il micro dovrà eseguire.

Vediamo adesso in maniera più approfondita, il set d'istruzioni del PIC, che rappresenta l'insieme d'istruzioni che il micro è in grado di eseguire e che è quindi possibile scrivere in un programma assembler.

Il set d'istruzioni del 16FXX2 è costituito da 75 istruzioni che, anche se possono sembrare poche, sono molte di più di quelle implementate dalle altre famiglie di microcontrollori PIC.

Le istruzioni possono essere così raggruppate:

- Byte-oriented operations
- Bit-oriented operations
- Literal operations
- Control operations

La maggior parte delle istruzioni ha una lunghezza di una singola word di memoria (16-bits), ma esistono tre istruzioni speciali che richiedono due locazioni di memoria (due word 32-bits).

Tutte le istruzioni che hanno lunghezza pari ad una word sono eseguite in un solo ciclo d'istruzione tranne nel caso in cui risulti vero un test condizionale o che il Program Counter (PC) vari come conseguenza dell'istruzione stessa. In questi casi l'istruzione per essere eseguita completamente richiede due cicli d'istruzione. Le tre istruzioni speciali con lunghezza due word richiedono sempre due cicli d'istruzione.

Ogni ciclo d'istruzione è costituito da quatto periodi di clock. Se ad esempio abbiamo una frequenza di clock pari a 4 MHz (T = 250 ns) il tempo di esecuzione di un'istruzione che richiede un solo ciclo d'istruzione è pari a 1 μs (4 X 250 ns = 1 μs). Se risulta vera un test condizionale o il program counter viene incrementato come risultato dell'istruzione, il tempo d'esecuzione dell'istruzione diventa doppio, cioè 2 μs (8 X 250 ns = 2 μs).

Di seguito vengono riportate due tabelle; la prima contiene una descrizione sintetica di tutti i campi che si possono trovare nel formato generale di un'istruzione (opcode), il cui schema viene riportato nella seconda tabella.

OPCODE FIELD DESCRIPTIONS

Field	Description
a	RAM access bit a = 0: RAM location in Access RAM (BSR register is ignored) a = 1: RAM bank is specified by BSR register
bbb	Bit address within an 8-bit file register (0 to 7)
BSR	Bank Select Register. Used to select the current RAM bank.
d	Destination select bit; d = 0: store result in WREG, d = 1: store result in file register f.
dest	Destination either the WREG register or the specified register file location
f	8-bit Register file address (0x00 to 0xFF)
fs	12-bit Register file address (0x000 to 0xFFF). This is the source address.
fd	12-bit Register file address (0x000 to 0xFFF). This is the destination address.
k	Literal field, constant data or label (may be either an 8-bit, 12-bit or a 20-bit value)
label	Label name
mm	The mode of the TBLPTR register for the Table Read and Table Write instructions. Only used with Table Read and Table Write instructions:
*	No Change to register (such as TBLPTR with Table reads and writes)
*+	Post-Increment register (such as TBLPTR with Table reads and writes)
*-	Post-Decrement register (such as TBLPTR with Table reads and writes)
+*	Pre-Increment register (such as TBLPTR with Table reads and writes)
n	The relative address (2's complement number) for relative branch instructions, or the direct address for Call/Branch and Return instructions
PRODH	Product of Multiply high byte
PRODL	Product of Multiply low byte
s	Fast Call/Return mode select bit. s = 0: do not update into/from shadow registers s = 1: certain registers loaded into/from shadow registers (Fast mode)
u	Unused or Unchanged
WREG	Working register (accumulator)
x	Don't care (0 or 1) The assembler will generate code with x = 0. It is the recommended form of use for compatibility with all Microchip software tools.
TBLPTR	21-bit Table Pointer (points to a Program Memory location)
TABLAT	8-bit Table Latch
TOS	Top-of-Stack
PC	Program Counter
PCL	Program Counter Low Byte
PCH	Program Counter High Byte
PCLATH	Program Counter High Byte Latch
PCLATU	Program Counter Upper Byte Latch
GIE	Global Interrupt Enable bit
WDT	Watchdog Timer
$\overline{TO}$	Time-out bit
$\overline{PD}$	Power-down bit
C, DC, Z, OV, N	ALU status bits Carry, Digit Carry, Zero, Overflow, Negative
[ ]	Optional
( )	Contents
→	Assigned to
< >	Register bit field
∈	In the set of
italics	User defined term (font is courier)

Tabella 3.14: Descrizione dei campi delle istruzioni.

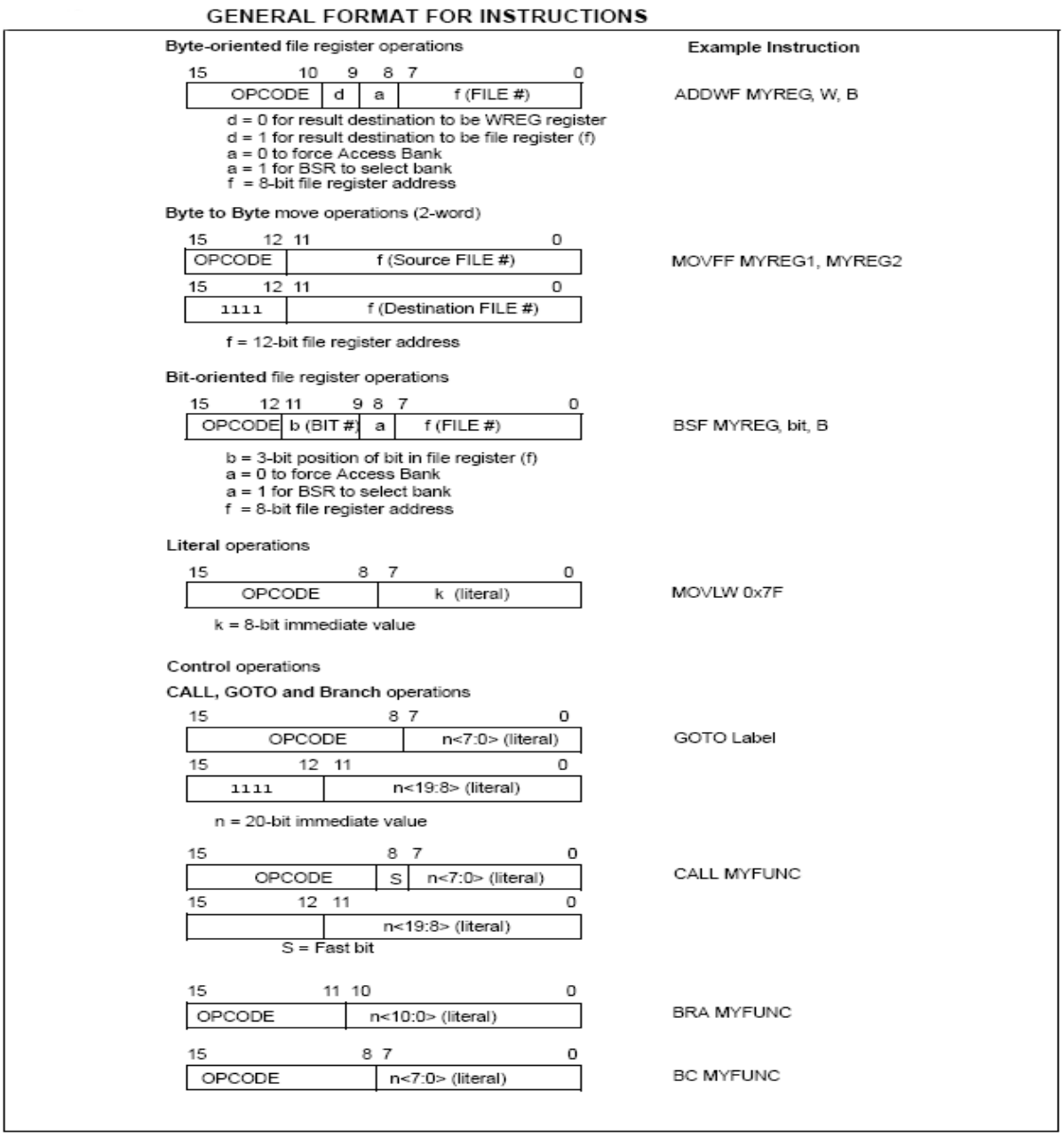


Fig. 3.20: Formato generale delle istruzioni.

Byte-oriented instruction

La maggior parte delle Byte-oriented instruction ha tre operandi:

1. file register (identificato con ‘f’)
2. destination register (identificato con ‘d’)
3. acceded memory(identificato con ‘a’)

Il file register che, come abbiamo detto sopra, viene segnalato con la lettera ‘f’ indica quale file register deve essere utilizzato dall’istruzione; il destination register indica il registro di destinazione, cioè dove deve essere posto il risultato dell’istruzione. Se tale operando è lasciato vuoto il risultato dell’istruzione viene salvato nel registro WREG, altrimenti, se posto ad uno, il risultato viene collocato nel file register specificato nell’istruzione.

Di seguito viene riportata una tabella con tutte le funzioni Byte-oriented:

Mnemonic, Operands		Description	Cycles	16-Bit Instruction Word				Status Affected	Notes
				MSb			LSb		
BYTE-ORIENTED FILE REGISTER OPERATIONS									
ADDWF	f, d, a	Add WREG and f	1	0010	01da0	ffff	ffff	C, DC, Z, OV, N	1, 2
ADDWFC	f, d, a	Add WREG and Carry bit to f	1	0010	oda	ffff	ffff	C, DC, Z, OV, N	1, 2
ANDWF	f, d, a	AND WREG with f	1	0001	01da	ffff	ffff	Z, N	1,2
CLRF	f, a	Clear f	1	0110	101a	ffff	ffff	Z	2
COMF	f, d, a	Complement f	1	0001	11da	ffff	ffff	Z, N	1, 2
CPFSEQ	f, a	Compare f with WREG, skip =	1 (2 or 3)	0110	001a	ffff	ffff	None	4
CPFSGT	f, a	Compare f with WREG, skip >	1 (2 or 3)	0110	010a	ffff	ffff	None	4
CPFSLT	f, a	Compare f with WREG, skip <	1 (2 or 3)	0110	000a	ffff	ffff	None	1, 2
DECF	f, d, a	Decrement f	1	0000	01da	ffff	ffff	C, DC, Z, OV, N	1, 2, 3, 4
DECFSZ	f, d, a	Decrement f, Skip if 0	1 (2 or 3)	0010	11da	ffff	ffff	None	1, 2, 3, 4
DCFSNZ	f, d, a	Decrement f, Skip if Not 0	1 (2 or 3)	0100	11da	ffff	ffff	None	1, 2
INCF	f, d, a	Increment f	1	0010	10da	ffff	ffff	C, DC, Z, OV, N	1, 2, 3, 4
INCFSZ	f, d, a	Increment f, Skip if 0	1 (2 or 3)	0011	11da	ffff	ffff	None	4
INFSNZ	f, d, a	Increment f, Skip if Not 0	1 (2 or 3)	0100	10da	ffff	ffff	None	1, 2
IORWF	f, d, a	Inclusive OR WREG with f	1	0001	00da	ffff	ffff	Z, N	1, 2
MOVF	f, d, a	Move f	1	0101	00da	ffff	ffff	Z, N	1
MOVFF	f_s, f_d	Move f_s (source) to 1st word f_d (destination) 2nd word	2	1100 1111	ffff ffff	ffff ffff	ffff ffff	None	
MOVWF	f, a	Move WREG to f	1	0110	111a	ffff	ffff	None	
MULWF	f, a	Multiply WREG with f	1	0000	001a	ffff	ffff	None	
NEGF	f, a	Negate f	1	0110	110a	ffff	ffff	C, DC, Z, OV, N	1, 2
RLCF	f, d, a	Rotate Left f through Carry	1	0011	01da	ffff	ffff	C, Z, N	
RLNCF	f, d, a	Rotate Left f (No Carry)	1	0100	01da	ffff	ffff	Z, N	1, 2
RRCF	f, d, a	Rotate Right f through Carry	1	0011	00da	ffff	ffff	C, Z, N	
RRNCF	f, d, a	Rotate Right f (No Carry)	1	0100	00da	ffff	ffff	Z, N	
SETF	f, a	Set f	1	0110	100a	ffff	ffff	None	
SUBFWB	f, d, a	Subtract f from WREG with borrow	1	0101	01da	ffff	ffff	C, DC, Z, OV, N	1, 2
SUBWF	f, d, a	Subtract WREG from f	1	0101	11da	ffff	ffff	C, DC, Z, OV, N	
SUBWFB	f, d, a	Subtract WREG from f with borrow	1	0101	10da	ffff	ffff	C, DC, Z, OV, N	1, 2
SWAPF	f, d, a	Swap nibbles in f	1	0011	10da	ffff	ffff	None	4
TSTFSZ	f, a	Test f, skip if 0	1 (2 or 3)	0110	011a	ffff	ffff	None	1, 2
XORWF	f, d, a	Exclusive OR WREG with f	1	0001	10da	ffff	ffff	Z, N	

Tabella 3.15: Byte-oriented instruction.

Bit-oriented instruction

Tutte le istruzioni bit-oriented hanno tre operandi:

1. file register (identificato con ‘f’)
2. bit nel file register (indicato con ‘b’)
3. acceded memory(identificato con ‘a’)

Il secondo campo, bit nel file register, indica su quanti bit indica il numero di bit su cui l’operazione avrà effetto, mente il campo file register indica il numero del file in cui il bit si trova.

Mnemonic, Operands		Description	Cycles	16-Bit Instruction Word				Status Affected	Notes
				MSb			LSb		
BIT-ORIENTED FILE REGISTER OPERATIONS									
BCF	f, b, a	Bit Clear f	1	1001	bbba	ffff	ffff	None	1, 2
BSF	f, b, a	Bit Set f	1	1000	bbba	ffff	ffff	None	1, 2
BTFSC	f, b, a	Bit Test f, Skip if Clear	1 (2 or 3)	1011	bbba	ffff	ffff	None	3, 4
BTFSS	f, b, a	Bit Test f, Skip if Set	1 (2 or 3)	1010	bbba	ffff	ffff	None	3, 4
BTG	f, d, a	Bit Toggle f	1	0111	bbba	ffff	ffff	None	1, 2

Tabella 3.16: Bit-oriented instruction.

Le funzioni bit-oriented sono:

Literal instruction

Le literal instruction possono far uso di alcuni dei seguenti operandi:

1. Un valore letterale che deve essere caricato in un file register (indicato con ‘k’)
2. L’ FSR register in cui caricare il valore letterale (indicato con ‘f’)
3. Nessun operando richiesto (indicato con ‘–’)

le literal instruction sono:

Mnemonic, Operands		Description	Cycles	16-Bit Instruction Word MSb			LSb	Status Affected	Notes
LITERAL OPERATIONS									
ADDLW	k	Add literal and WREG	1	0000	1111	kkkk	kkkk	C, DC, Z, OV, N	
ANDLW	k	AND literal with WREG	1	0000	1011	kkkk	kkkk	Z, N	
IORLW	k	Inclusive OR literal with WREG	1	0000	1001	kkkk	kkkk	Z, N	
LFSR	f, k	Move literal (12-bit) 2nd word to FSRx 1st word	2	1110 1111	1110 0000	00ff kkkk	kkkk kkkk	None	
MOVLB	k	Move literal to BSR<3:0>	1	0000	0001	0000	kkkk	None	
MOVLW	k	Move literal to WREG	1	0000	1110	kkkk	kkkk	None	
MULLW	k	Multiply literal with WREG	1	0000	1101	kkkk	kkkk	None	
RETLW	k	Return with literal in WREG	2	0000	1100	kkkk	kkkk	None	
SUBLW	k	Subtract WREG from literal	1	0000	1000	kkkk	kkkk	C, DC, Z, OV, N	
XORLW	k	Exclusive OR literal with WREG	1	0000	1010	kkkk	kkkk	Z, N	

Tabella 3.17: Literal instruction.

Control instruction

Le control instruction possono far uso di alcuni dei seguenti operandi:

1. Un indirizzo della program memory (indicato con ‘n’).
2. La modalità dell’istruzione di Call o Return (indicato con ‘s’).
3. La modalità dell’istruzione di Table Read e Table Write (indicato con ‘m’).
4. Nessun operando richiesto (indicato con ‘–’).

Le control operation sono:

Mnemonic, Operands		Description	Cycles	16-Bit Instruction Word MSb			LSb	Status Affected	Notes
CONTROL OPERATIONS									
BC	n	Branch if Carry	1 (2)	1110	0010	nnnn	nnnn	None	
BN	n	Branch if Negative	1 (2)	1110	0110	nnnn	nnnn	None	
BNC	n	Branch if Not Carry	1 (2)	1110	0011	nnnn	nnnn	None	
BNN	n	Branch if Not Negative	1 (2)	1110	0111	nnnn	nnnn	None	
BNOV	n	Branch if Not Overflow	1 (2)	1110	0101	nnnn	nnnn	None	
BNZ	n	Branch if Not Zero	2	1110	0001	nnnn	nnnn	None	
BOV	n	Branch if Overflow	1 (2)	1110	0100	nnnn	nnnn	None	
BRA	n	Branch Unconditionally	1 (2)	1101	0nnn	nnnn	nnnn	None	
BZ	n	Branch if Zero	1 (2)	1110	0000	nnnn	nnnn	None	
CALL	n, s	Call subroutine1st word	2	1110	110s	kkkk	kkkk	None	
		2nd word		1111	kkkk	kkkk	kkkk		
CLRWDT	—	Clear Watchdog Timer	1	0000	0000	0000	0100	$\overline{TO}$, $\overline{PD}$	
DAW	—	Decimal Adjust WREG	1	0000	0000	0000	0111	C	
GOTO	n	Go to address1st word	2	1110	1111	kkkk	kkkk	None	
		2nd word		1111	kkkk	kkkk	kkkk		
NOP	—	No Operation	1	0000	0000	0000	0000	None	
NOP	—	No Operation	1	1111	xxxx	xxxx	xxxx	None	4
POP	—	Pop top of return stack (TOS)	1	0000	0000	0000	0110	None	
PUSH	—	Push top of return stack (TOS)	1	0000	0000	0000	0101	None	
RCALL	n	Relative Call	2	1101	1nnn	nnnn	nnnn	None	
RESET		Software device RESET	1	0000	0000	1111	1111	All	
RETFIE	s	Return from interrupt enable	2	0000	0000	0001	000s	GIE/GIEH, PEIE/GIEL	
RETLW	k	Return with literal in WREG	2	0000	1100	kkkk	kkkk	None	
RETURN	s	Return from Subroutine	2	0000	0000	0001	001s	None	
SLEEP	—	Go into Standby mode	1	0000	0000	0000	0011	$\overline{TO}$, $\overline{PD}$	

Tabella 3.18: Control instruction.