SELINUS UNIVERSITY
OF SCIENCES AND LITERATURE

Panoramica delle applicazioni del microcontrollore General Purpose INTEL 80C51

By Rocco Cannizzaro

Supervised by
Prof. Salvatore Fava Ph.D.

A DISSERTATION

Presented to the Department of
Electronic Engineering
program at Selinus University

Faculty of Engineering & Technology
in fulfillment of the requirements
for the degree of
Bachelor of Science
in Electronic Engineering

2021

Al mio nipotino

INDICE

PREMESSA

Prima di iniziare a parlare del microcontrollore general purpose, è doveroso fare un richiamo al suo predecessore della quale ne ha ereditato una buona parte relativa alla CPU.

L'antenato in questione è il MICROPRCESSORE, un circuito integrato nato negli anni settanta il mod. 4004 prodotto dalla INTEL.

Precedentemente la logica digitale era di tipo hardware creata con circuiti integrati di tipo TTL(Transistor-Transistor-Logic), alimentatati a 5Vdc, e successivamente della tecnologia HMOS & CMOS. Il Microprocessore 4004 possedeva 2300 transistor, che andavano a formare una ALU, un Program Counter, uno stack e 16 registri, ma era solo un dispositivo di controllo capace di elaborare dati poco complessi. Nel 1976 nasce l'8048 questo microcontrollore era basato su un'architettura Von NEUMANN modificata capace di indirizzare memoria di tipo ROM ma anche RAM esterna al chip. Le porte di I/O erano mappate nello spazio di indirizzamento proprio della CPU, separato dallo spazio di indirizzamento della memoria del programma e di quella dei dati.

Il microcontrollore richiedeva una sola alimentazione a 5 Volt e quasi tutte le istruzioni (più di 90) occupavano 1 solo byte per tipo.

Oggi si contano più di 1000 modelli di microcontrollori la maggior parte derivati dal 80C51 della Intel. Basta sapere che nel mondo sono stati installati 11. miliardi di microcontrollori contro i 7 miliardi di microprocessori. Per comprendere quanto sia importante l'applicazione dei microcontrollori nell'industria bisogna fare una precisazione: nonostante la sua età quasi 50 anni il microcontrollore 80C51 risulta essere il più utilizzato in applicazioni generali. Forse grazie alla sua semplice configurazione hardware che è capace di comunicare con dispositivi esterni abbastanza complessi come i convertitori A/D & D/A, e dialogare con altri microcontrollori creando un primitivo multitasking.

I microcontrollori permettono la realizzazione su vasta scala e a costi veramente contenuti vere e proprie *funzioni di controllo* per diverse applicazioni, sia in ambito civile che industriale.

L'impiego dei microncotrollori è molto diffuso nella realizzazione di prodotti di uso quotidiano come, le periferiche per PC, i telefoni cellulari, le macchine fotocopiatrici, i sistemi di telefonia, gli apparati per la domotica, gli elettrodomestici etc..

Nell'ambito industriali i microcontrollori sono particolarmente impiegati nel settore dell'automazione, degli azionamenti elettrici come i DRIVER per l'avvio dei motori elettrici si Trifase che monofase, e si interfacciano con tutti i sensori e trasduttori per le diverse grandezze fisiche.

CAPITOLO 1

Sistemi a microcontrollori

1.1 *MICROCONTROLLORI CARATTERISTICHE ELETTRICHE E SOFTWARE*

Un microcontrollore che di seguito chiameremo MCU, è un microprocessore ottimizzato opportunamente per formare un vero e proprio sistema di elaborazione dati in un unico Package.
L'MCU è caratterizzato di avere implementato a bordo "on-chip" i componenti essenziali per formare un sistema completo di elaborazione, senza la necessità di richiedere ulteriori componenti esterni.
Come tutti i processori L'MCU possiede una ALU
(Artimetic Logic Unit), un sistema di memoria che forma la memoria Programma (ROM,Flash,EPROM) e la memoria dinamica (Ram statiche),(Ram dinamiche),EEprom.
A differenza dei microprocessori negli MCU sono presenti numerose unità circuitali periferici interne come le porte I/O (Input/Output di segnali digitali), i timer (Timer e Contatori) per la gestione del timing di processo, inoltre un MCU viene progettato per offrire una minore complessità Hardware, e bassi consumi elettrici, ideali per applicazioni portatili e a batterie.
Grazie ad una ampia produzione mondiale diversificata, oggi si può scegliere il MCU più adatto alla specifica applicazione, infatti ci sono MCU a 8/16/32 bit con velocità che vanno da pochi MHz fino a 600 MHz.
Gli MCUs posseggono un sofisticato codice programma che opportunamente programmato esegue istruzioni di tipo matematiche, logiche, temporizzazioni, interrupts, gestione della memoria interna ed esterna, controllo della velocità di processo grazie ai timer interni.
Il linguaggio ASSEMBLER è il linguaggio che più si avvicina al codice macchine, quando si programma un MCU nel caso specifico del MCU 80C51 occorre solo un software compilatore che codifica il linguaggio Assembler in codice macchina. Il linguaggio Assembler è molto potente anche se un pò ostico da comprendere, però è l'unico linguaggio che entra nel cuore del MCU e ne controlla tutti i suoi componenti interni a livello di registri, variabili Boleane a singolo Bit, byte e word di 16 Bit per l'indirizzamento della memoria esterna.
L'assembler permette di calcolare il timing di ogni singola istruzione, questo permette di controllare sia la quantità di memoria occupata dal codice programma sia l'ottimizzazione della velocità del codice.
Esistono altri linguaggi come il "C" il BASIC, il PASCAL questi sono si linguaggi ad alto livello ma sono anche linguaggi compilati che utilizzano molta memoria per compilare e trascrivere il codice macchina, chiaramente ove non necessita la gestione del timing e della memoria si possono usare tranquillamente i linguaggi ad alto livello compilati che risultano di più facile interpretazione da parte del programmatore.

1.2 CRITERI DI SCELTA DEI MICROCONTROLLORI

In primo luogo è necessario, prima di entrare nella progettazione vera e propria del sistema a MCU, avere ben chiaro ciò che si vuole ottenere da esso: le funzioni che dovrà svolgere, i dispositivi esterni che dovrà gestire, le informazioni che dovrà elaborare, e in generale si tratta di definire le specifiche del progetto per poi decidere su quale MCU si farà la scelta. Ovviamente non ultimo i costi di acquisto del MCU e i costi di programmazione come il software per scrivere il codice programma.
A questo punto la conoscenza dei dispositivi periferici consente di definire in linea di massima la configurazione delle porte I/O, e della linea seriale.
Una volta configurata la parte hardware si può procedere con la progettazione di uno flow chart e di uno schema a blocchi per iniziare a scrivere la configurazione del software.
Scegliere il MCU significa scegliere in primo luogo la classe di prestazioni e identificarne il modello.
La classe di prestazioni è collegata soprattutto al parallelismo del BUS, ovvero che il MCU possa essere a 4/8/16/32 Bit. Da questo dipende la velocità di elaborazione del MCU, e la capacità di indirizzamento,attenzione! in primis queste caratteristiche vanno confrontate con le esigenze del progetto.

1.3 *BENCHMARKING*

Esiste un modo per valutare le prestazioni di un MCU General Porpuse, attraverso il "Benchmarking" ovvero l'uso di un programma campione. Questo test permette di valutare globalmente le prestazione del MCU includendo anche i tempi di accesso alle memorie ROM & RAM.

1.4 *INDICI DI PRESTAZIONI*

Le prime informazione sugli indici di prestazione viene fornito dal costruttore del CHIP che emette un DATA SHEET dove sono elencate le caratteristiche del MCU ed alcuni grafici che mostrano i tempi di accesso alla memoria sia in scrittura che in lettura, inoltre si trovano anche le seguenti informazioni:

- MIPS: Milioni di istruzioni al secondo
- DMIPS: Milioni di istruzioni al secondo per un algoritmo benchmark specifico, detto Dhrystone.
- CoreMark: Indice sintetico di prestazioni ricavato dall'esecuzione di una combinazione di algoritmi standard.

- OSCILLATOR CHARACTERISTICS: è il motore del MCU tramite i due ingressi è possibile connettere un quarzo oscillatore con una frequenza conosciuta che genera all'interno del MCU la frequenza di clock. Si può eventualmente testare il comportamento alle diverse frequenze di clock, variando la frequenza di oscillazione ad esempio sostituendo al quarzo un generatore di onda quadra variabile si può verificare la gamma di lavoro del MCU e la sua risposta al variare delle frequenze di oscillazione.

- ABSOLUTE MAXIMUM RATINGS: i limiti elettrici che può supportare il MCU, in funzione di questi limiti si possono condurre i test di stress del MCU ai fini dell'affidabilità.

CAPITOLO 2
Architettura del Microcontrollor

- ***2.1 MODELLI DI ARCHITETTURA E BUS DI COMUNICAZIONE***

Per la costruzione dei MCU esistono due tipi di architettura costruttiva la VON NEUMANN e la HARWARD la prima utilizza lo stesso BUS sia per leggera la memoria programma sia per leggere e scrivere nella memoria RAM, mentre la HARWARD utilizza due bus indipendenti uno per la memoria programma e l'altro BUS dati.
Il MCU Intel 80C51 utilizza l'architettura VON NEUMANN.

VON NEUMANN ARCHITETTURA

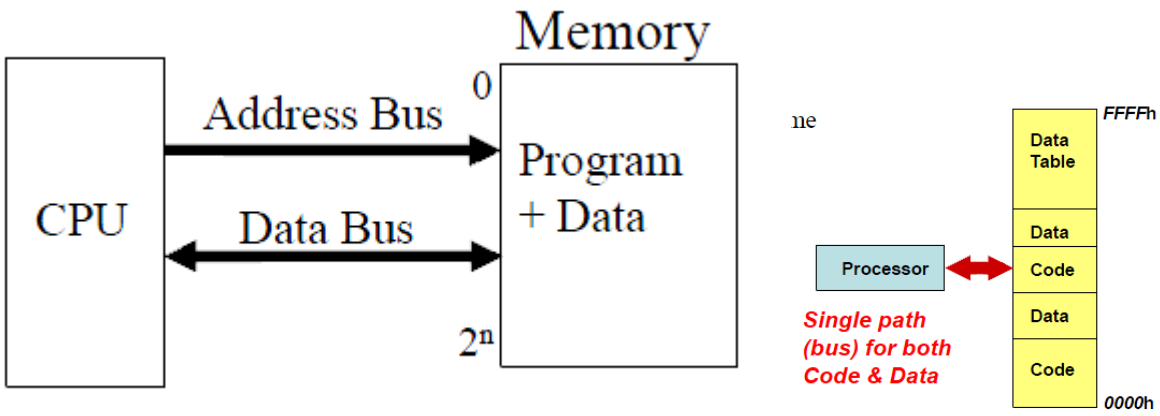


Fig. 2.1

HARWARD ARCHITETTURA

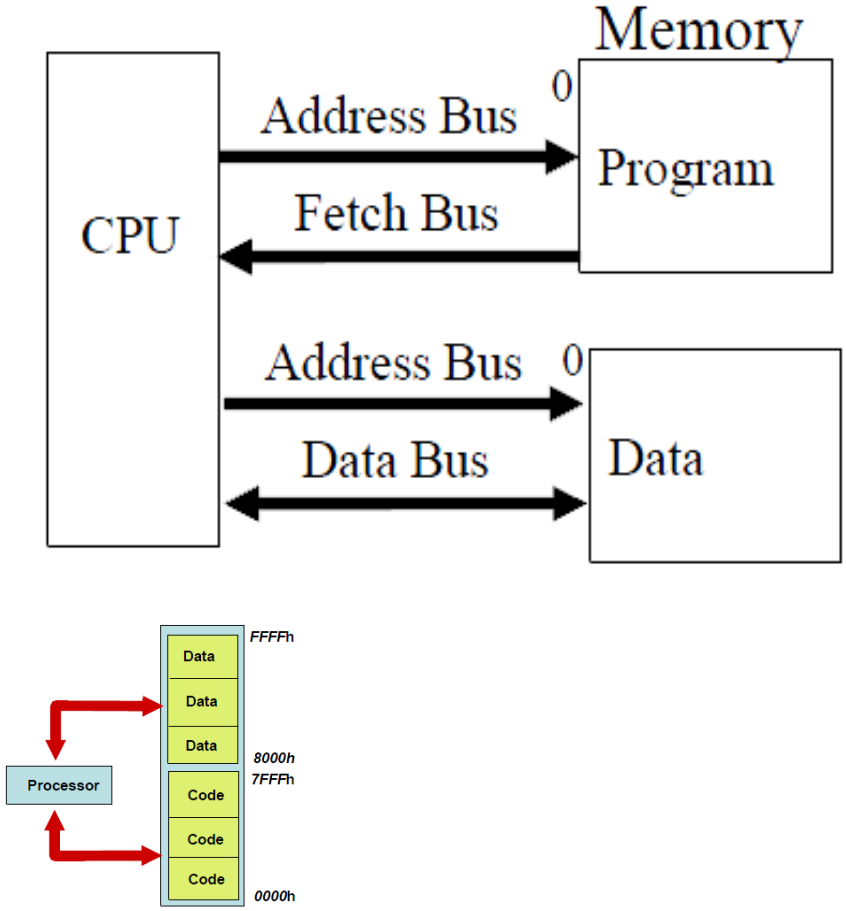


Fig. 2.2

La differenza sostanziale che c'è tra le due architetture è la seguente: Nell'architettura di Von Neumann VEDI FIG. 2.1 c'è un unico banco di memoria che racchiude Program memory e Data memory, di conseguenza la CPU è costretta a fare la fase di fetch-exec in modo sequenziale prima l'una e poi l'altra; Nell'architettura Harvard vedi fig. 2.2 possiamo invece distinguere i due banchi di memoria in modo tale che la fase di fetch può sovrapporsi a quella di exec e viceversa.
L'architettura 8051 è quella classica dei primi processori Intel, con un bus interno istruzioni/dati ad 8 bit, mentre gli indirizzi a 16 bit che permettono di ottenere la dimensione dello spazio di indirizzamento,
cioè la quantità massima di memoria accessibile dal dispositivo, che è di 64 Kb, sia per la memoria di programma che per la memoria dati. Lo spazio di indirizzamento di istruzioni e dati non è separato, vale a dire che un dato di (memoria dati) può avere lo stesso indirizzo di una istruzione di programma.
Questo ovviamente non crea problemi, in quanto le due parti di memoria sono anche fisicamente separate. Occorre però ricordare che il bus di sistema è comunque unico, pertanto l'architettura fisica dell'8051 è a tutti gli effetti una architettura Von Neumann.

2.2 ARCHITETTURE RISC & CISC

Quando nel 1950 i transistor disponibili su un solo chip erano pochi e i calcolatori venivano spesso programmati in assembler, era logico utilizzarli in modo tale da avere CPU con un set di istruzioni potenti, evolute e complesse: più queste istruzioni erano vicine alle istruzioni dei linguaggi di programmazione ad alto livello più il MCU sarebbe stato facile da programmare, e i programmi avrebbero occupato poco spazio in memoria. Le CPU progettate secondo questo approccio vengono chiamate CISC (Complex Instruction Set Computer) e possedevano unità di controllo complesse capaci di gestire al meglio pochi registri di cui i processori ne avevano dimensioni relativamente piccole.
A cavallo fra gli anni '70 e gli '80 la situazione però si è evoluta al punto che: la RAM divenne più economica e comparvero i primi compilatori moderni, ottimizzanti, in grado di generare linguaggio macchina molto efficiente. Per questo si iniziò a pensare ad un nuovo modo di progettare le CPU, considerando la possibilità di usare i transistor implementati all'interno del chip disponibili per avere invece molti registri e un set di istruzioni elementari, molto ridotto, che delegasse al compilatore il lavoro di tradurre le istruzioni complesse in serie a delle istruzioni più semplici, permettendo così di avere unità di controllo particolarmente semplici e veloci questo approccio viene chiamato RISC (Reduced Instruction Set Computer). Attualmente la distinzione fra queste due classi di architetture è venuta in gran parte meno: il numero di transistor disponibili su un solo chip è aumentato tanto da poter gestire molti registri ed anche set di istruzioni complesse.

- **Architettura CISC**

  CISC è l'acronimo di *Complex Instruction Set Computer*, tipicamente un processore di questo tipo implementa un numero relativamente basso di registri di uso generale, ed ha una unità di controllo micro programmata, la logica del programma è memorizzata in una memoria veloce situata nella parte di controllo, invece di essere espressa tramite una rete combinatoria.

  Il set di istruzioni associato a CPU di tipo CISC è molto esteso e composto in genere di alcune centinaia di codici operativi diversi che svolgono funzioni anche molto complesse, fra cui sono caratteristici i trasferimenti memoria-memoria, assenti nei RISC; le istruzioni hanno lunghezza variabile e possono presentarsi in formati diversi, e sono necessari due o più (a volte molti di più) cicli di clock per completare un'istruzione; è possibile specificare la posizione dei dati necessari alle istruzioni usando molti metodi di indirizzamento diversi. Il ridotto numero di registri interni obbliga questi processori a scrivere in memoria ogni volta che si verifica una chiamata di funzione, che si verifica un context switch o che viene salvato un registro nello stack.

  Programmare in Assembler una CPU CISC è un compito (relativamente) facile, perché le istruzioni presenti sono più vicine a quelle dei linguaggi ad alto livello.

- **Architettura RISC**

  RISC è l'acronimo di *Reduced Instruction Set Computer*. Il tipico set di istruzioni RISC è molto piccolo, che va da circa 80 a 110 istruzioni molto elementari (logiche, aritmetiche e istruzioni di trasferimento memoria-registro e registro-registro): hanno tutte lo stesso formato e la stessa lunghezza, e molte vengono eseguite in un solo ciclo di clock. La diretta

conseguenza di questa scelta progettuale è che i processori RISC posseggono una unità di controllo semplice e a bassa latenza, riservando invece molto spazio per i registri interni: un MCU RISC ha di solito da un minimo di un centinaio ad alcune migliaia di registri interni generici, organizzati in un file di registri. Il fatto di avere un formato unico di istruzione permette di strutturare l'unità di controllo come una *pipeline*, cioè una catena di montaggio a più stadi: questa innovazione ha il notevole vantaggio di ridurre il *critical path* interno alla CPU e consente ai RISC di raggiungere frequenze di clock più alte rispetto agli analoghi CISC.

Nel caso di *context-switch* o di chiamata a subroutine o comunque di uso dello stack i RISC spesso invece di accedere alla memoria di sistema usano un meccanismo chiamato *register renaming*, che consiste nel rinominare i registri in modo da usare per la nuova esecuzione una diversa zona del file di registri, senza dover accedere alla memoria ogni volta.

Praticamente sovrascrive partendo da un modo di scritture DOWN-TOP per salvare lo stato della CPU, e dopo restituire la sequenza memorizzata con una scrittura di restituzione di tipo TOP-DOWN, il tutto in pochi cicli di macchina.

- **Architettura RISC contro architettura CISC**

  La semplicità dei RISC si traduce in una minore espressività del linguaggio assembler: il numero di word necessarie per esprimere una computazione in una macchina RISC è maggiore/uguale alla controparte CISC: questo si traduce in programmi più grossi, penalità molto alta in altre epoche, in cui la RAM era una componente costosa e di bassa capacità. L'architettura CISC dipende dal compilatore più di quanto non dipenda il RISC: dato che le istruzioni prevedono più metodi di indirizzamento, e che son presenti istruzioni dalla semantica complessa, al compilatore viene prospettato un ampio ventaglio di scelte per quanto concerne la traduzione di una istruzione, e non sempre la scelta migliore è banale. Come spesso accade nei problemi di ottimizzazione, la scelta della configurazione migliore è un compito arduo, e non è pensabile utilizzare un compilatore che, per ogni istruzione, valuti la scelta migliore in base al contesto. Si conoscono solo delle buone euristiche, ma il problema dell'ottimizzazione è un problema di ricerca aperto. La stessa macchina CISC può essere dunque più o meno veloce rispetto ad una macchina comparabile RISC in base al compilatore utilizzato.

2.3 ARCHITETTURA DEL MICROCONTROLLORE INTEL 8051

L’evoluzione del progresso delle tecnologie elettroniche ha portato l’integrazione di centinaia di migliaia componenti semiconduttori all’interno di un unico CHIP, questo ha favorito lo sviluppo progettuale di dispositivi complessi, come unità periferiche sofisticate, porte seriali di comunicazione, e raccoglimento al loro interno di tutte le funzioni tipiche di un micro Computer.
Questo è stata determinata dall’esigenza di supportare la CPU con dispositivi capaci di svolgere particolari funzioni, come l’interfacciamento con altri dispositivi esterni.
Tipicamente un MCU è alloggiato in un contenitore DIL 40 PIN vedi fig.2.3

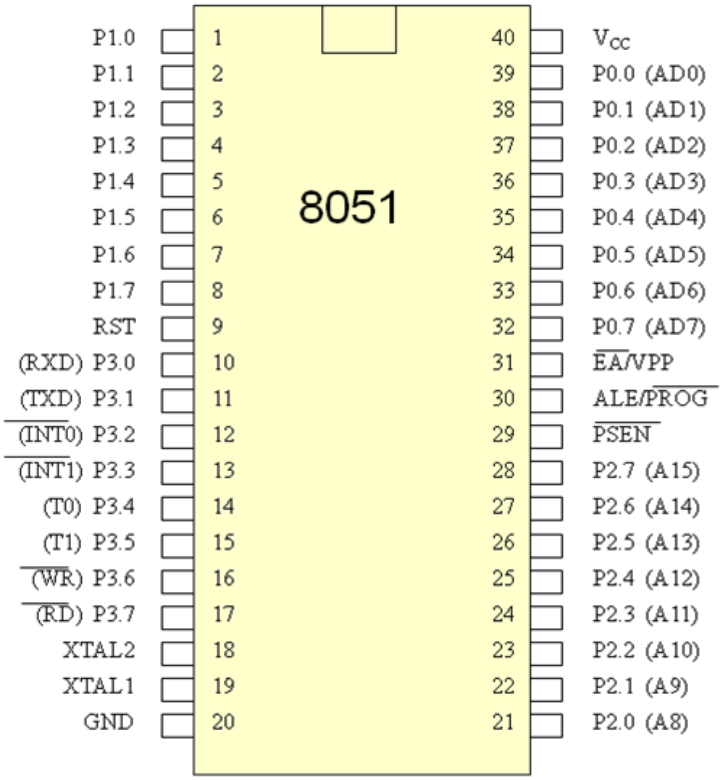


FIG. 2.3

Tipicamente L'MCU 8051 è composto da quattro porte I/O, una RAM interna da 128 byte per la manipolazione dei dati, e 128 byte per i registri speciali, una ROM da 1Kb per i programmi, due timer, ed una porta seriale di tipo UART(Universal Asynchrony Receiver transmitter).
Inoltre al suo interno sono implmentati altri dispositivi per la gestione interna delle operazioni:

- Registro Accumulatore
- Registro B
- Banco registri di lettura e scrittura
- Unità di timing e controllo
- L'oscillatore di ingresso
- La ALU (Aritmetica Logic Unit)
- Il Data Pointer DPTR
- Program Counter
- I due Timer TH1 & TH2
- Linstruction register
- L'interrupt register
- Il PSW Program Status Word

Di seguito viene mostrato nella figura 2.4 il diagramma a blocchi del MCU 80C51.

Block Diagram

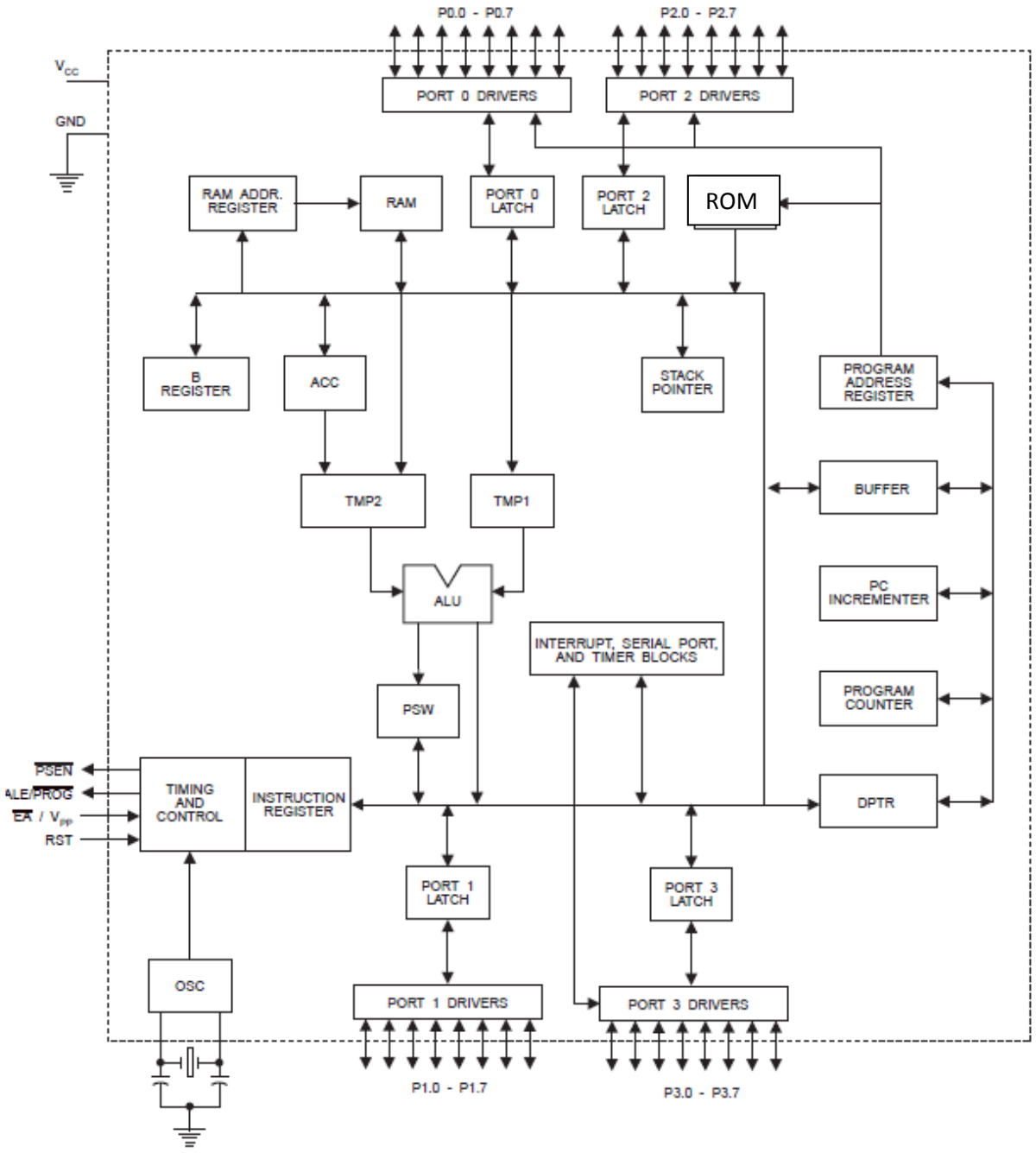


FIG. 2.4

Nella figura 2.5 si mostra il package e lo schema funzionale del MCU 8051. Come si può vedere dallo schema funzionale il BUS Address è condiviso con la memoria sia RAM che ROM mentre il BUS dati è collegato fisicamente a tutti i dispositivi interni, che provvede la Control Unit ad autorizzare la lettura e la scrittura dei dati su ogni dispositivo del MCU.

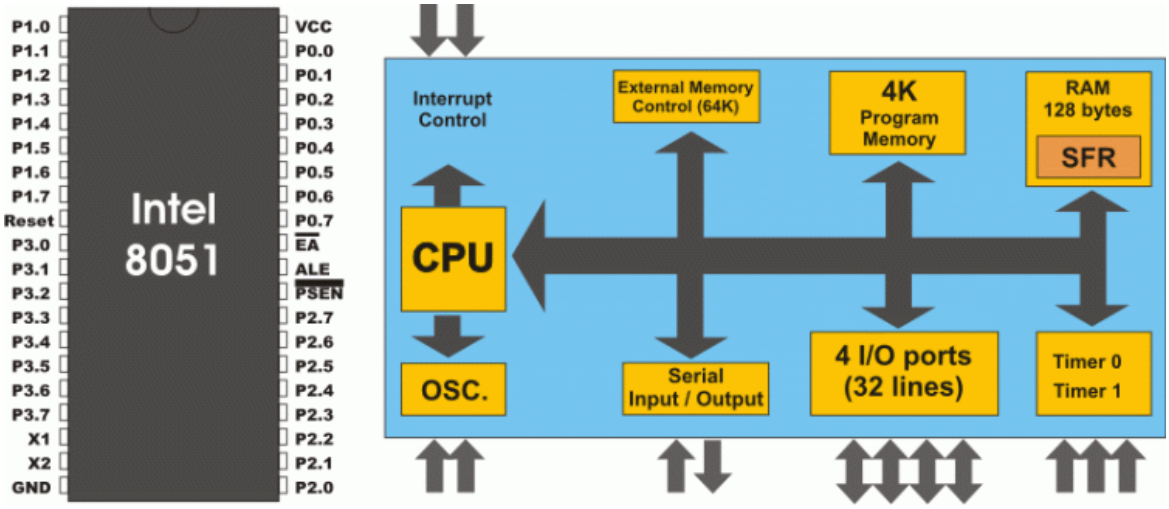


FIG. 2.5

Nel caso si utilizzasse la memoria esterna bisogna considerare il fatto che dipenderà da come o settato il Pin EA se si vorrà utilizzarre anche la memoria ROM interna.

2.4 LA RAM INTERNA

La RAM (Fig. 2.6) è composta da 4 banchi Bank0-Bank3 ognuno contenente 8 registri da 1 byte ciascuno, la memoria registri va da 00h a 1Fh . A seguire troviamo la memoria a BIT ovvero la memoria indirizzabile per ogni singolo bit che va da 20h a 2Fh, ed infine c'è il SFR Special Function Register che va da 30h a 7Fh, I registri speciali (SFR) sono aree di memoria che controllano specifiche funzionalità dell'8051.

Per esempio, 4 registri SFR consentono l'accesso alle 32 linee di input/output dell'8051. Un altro registro SFR consente di leggere e scrivere sulla porta seriale dell'8051. Altri registri SFR permettono all'utente di:
impostare il baud rate della seriale, controllare ed accedere ai timer e configurare il sistema degli interrupt dell'8051.

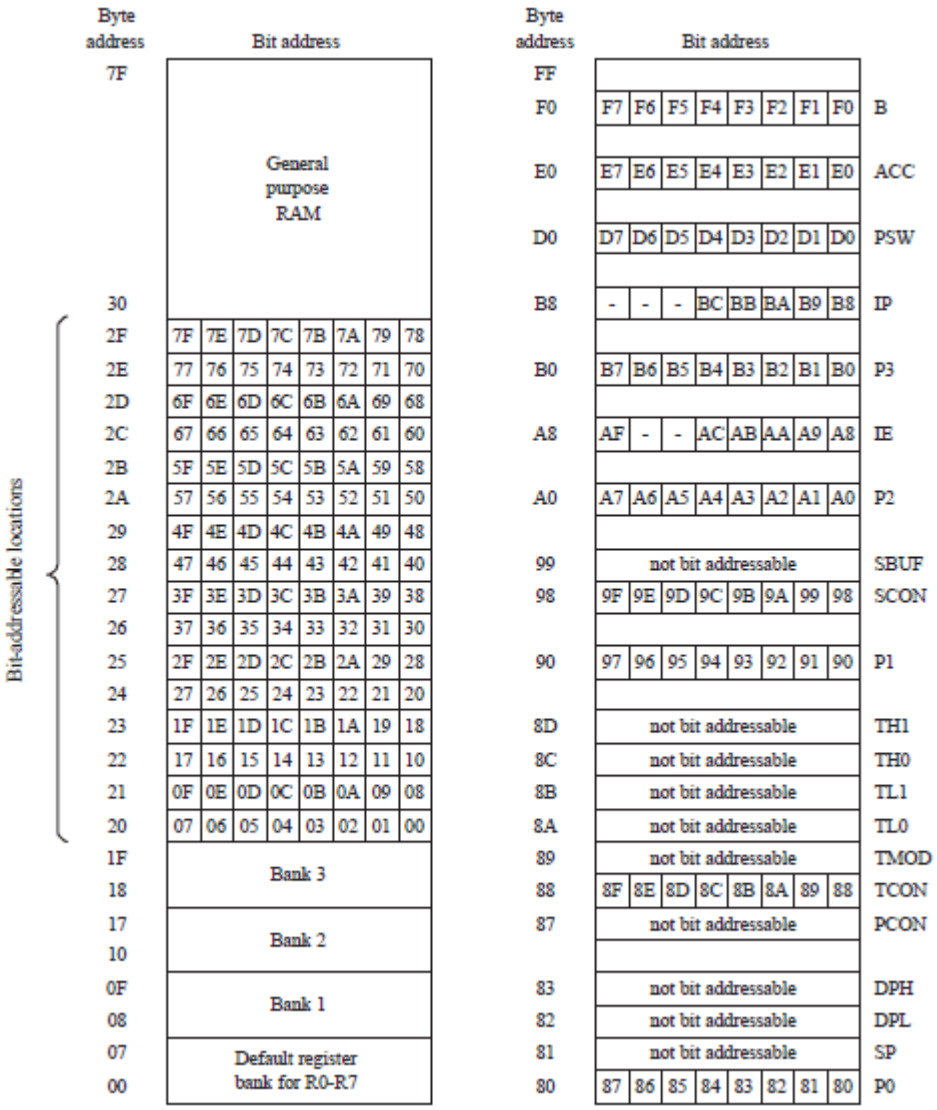


FIG.2.6

2.5 I BANCHI DI REGISTRI

Questi registri della RAM dati, formati da 4 banchi Bank0-Bank3 composti da 8 locazioni da 8 bit, sono utilizzabili come registri di uso corrente da parte del programmatore, e sono selezionabili con opportune configurazioni da inviare ai bit RS1 e RS0 al registro PSW (Program Status Word).
Ci sono 2 modi per indirizzare il registro che si vuole usare:

- Modo 1 immettere direttamente nell'istruzione la locazione di memoria in esadecimale corrispondente al alla cella di memoria RAM del registro.

Es.

ADD **A, 07h** ; Immetti nel registro accumulatore
il contenuto del registro R7 del
Bank0.

Oppure

ADD **A, 1Dh** ; Immetti nel registro accumulatore il contenuto del registro R5 del Bank3

Modo 2 Immettere direttamente nell'istruzione il nome del registro che si vuole utilizzare.

ADD **A,R0**; Immetti nel registro accumulatore il contenuto del registro R0 del Bank0.

SETB **D3h**
SETB **D4h**; in questo modo si sono settati a 1 i Bit 3 & 4 del PSW (Program Status Word register) dalla tabella di comparazione del PSW i bit settati High entrambi selezionano il Banco 3 dei registri di memoria.
Quindi:

SETB **D3h**
SETB **D4h**
ADD **A, R5**; equivale a scrivere nell'Accumulatore il valore del registro R5 del Banco 3.

PSW (Program Staus Register)

CY	AC	F0	RS1	RS0	OV	-	P

FIG.2.7

CY	PSW.7	D7h	Carry Flag
AC	PSW.6	D6h	Auxiliary Carry Flag
F0	PSW.5	D5h	Avilable to the general user
RS1	PSW.4	D4h	Register Bank selector Bit 1
RS0	PSW.3	D3h	Register Bank selector Bit 0
OV	PSW.2	D2h	Overflow Flag
-	PSW.1	D1h	User definible bit
P	PSW.0	D0h	Parity Flag

Selezione dei banchi di registri

RS1	RS2	Register bank	Address
0	0	0	00h-07h
0	1	1	08h-0Fh
1	0	2	10h-17h
1	1	3	18h-1Fh

FIG. 2.8

E’ evidente che il modo 2 occupa più cicli macchina per andare a scrivere il valore di un registro sull’accumulatore, questo però non deve sviare, in quanto, questa opzione di selezionare i banchi permette di usare gli stessi banchi come una ram dinamica, quindi scrivere e leggere dati anche a 8 bit e trasferirli direttamente sul BUS Dati.

2.6 LA MEMORIA A BIT

La memoria a bit ha la caratteristica di avere l’accesso a 128 variabili a BIT, questo significa un solo BIT può essere settato a 1 o a zero, numerate da 20h a 2Fh.

I comandi SETB & CLR permetto al programmatore di accedere direttamente alle variabili a BIT.
Es. setta a 1 i bit da 00h a 07h e cancella il bit 28.

```
SETB    20h
SETB    21h
SETB    22h
SETB    23h
SETB    24h
SETB    25h
SETB    26h
SETB    27h
CLR     1Bh
```

Per settare gli otto bit ci sono voluti 8 cicli macchina, una riduzione e ottimizzazione dei cicli macchina è possibile usando una sola istruzione:

```
MOV     20h,#0FFh
```

0FFh in decimale vale 255 in binario vale 1111 1111 quindi 8 bit settati a uno. Basta modificare una istruzione per migliorare la performance del MCU.

2.7 L’ACCUMULATORE “A”

L’Accumulatore è un registro ad 8 BIT corrispondente alla locazione di memoria RAM all’indirizzo E0h. Ad ogni reset questo registro viene sovrascritto con il valore Zero. L'Accumulatore, è usato come registro generale per accumulare i risultati di un gran numero di istruzioni. Esso può contenere un valore di 8-bit (1-byte) ed è il registro più versatile tra quelli dell'8051 per l'elevato numero di istruzioni che lo usano. Più di 130 istruzioni dell'8051 lo utilizzano. Specialmente nelle operazioni di trasferimento viene spesso utilizzato come registro appoggio dati, e anche nelle operazioni matematiche viene usato per addizionare, moltiplicare, sottrarre, e dividere etc.

Es. addizioniamo il numero 10 al numero 40:

```
MOV     R0,#10h     ; scrivi nel registro R0 il valore 10
MOV     A,#40h      ;scrivi il valore 40 nell'Accumulatore
ADD     A,R0            ; Somma R0 + A e scrivi il risultato della somma in A
```

Ora A contiene il valore 50

2.8 IL REGISTRO "B"

Il registro "B" e' molto simile all'Accumulatore, può memorizzare una parola di 8-bit.
Esso é usato soltanto in due istruzioni dell'8051: MUL AB e DIV AB. Quindi se si vuole moltiplicare o dividere facilmente e velocemente un numero con un altro, si deve memorizzare il secondo numero nel registro "B" ed usare queste due istruzioni. Oltre che alle due istruzioni MUL e DIV, eventualmente il resgistro "B" può essere usato come un registro temporaneo di appoggio dati, diventando fisicamente il nono registro della RAM Dati.

2.9 IL PROGRAM COUNTER (PC)

Il Program Counter (PC - Contatore di Programma) genera un indirizzo a 2-bytes che punta alla locazione di memoria programma ROM dove l'8051 andrà a prendere la prossima istruzione da eseguire.
Allo startup, l'8051 inizializza PC sempre al valore 0000h e lo incrementa ogni volta che un istruzione viene eseguita. Attenzione però, esistono istruzioni che impiegano solo 1 byte per incrementare il codice, ma ci sono anche istruzioni a 2 & 3 byte, quindi il PC eseguirà salti anche di 2 & 3 byte.

2.10 LO STACK POINTER (SP)

Lo Stack Pointer è uno SFR (Special Function Register) indirizzo 81h a 8 BIT punta all'ultima locazione di memoria RAM utilizzata per il salvataggio del contenuto di un registro. Lo SP opera in modo automatico, incrementa con l'istruzione di (PUSH) la RAM quando deve salvare una configurazione in seguito ad un interrupt o ad un salto di routine, decrementa invece quando deve rimettere la configurazione prelevata, il recupero avviene in modo automatico, e può essere manipolato via software con opportune istruzioni, e può essere inizializzato da un comando di reset al valore 07h. Questo registro SFR viene modificato da tutte le istruzioni che comportano operazioni sullo stack quali: **PUSH, POP, LCALL, RET, RETI**, e ogni volta che il microcontrollore esegue un'operazione di interrupt.

2.11 IL DATA POINTER REGISTER (DPTR)

Il DPTR e uno registro a 16 BIT composto dai 2 registri DPL/DPH (Data Pointer Basso/Alto, Indirizzi 82h/83h): I registri DPL e DPH lavorano insieme per formare un valore a 16-bit chiamato *Data Pointer.* Il Data Pointer è usato nelle operazioni che riguardano la memoria RAM esterna ed alcune istruzioni che riguardano il codice di programma. Comunque il DPTR è in realtà un valore a

16-bit ottenuto unendo insieme i registri DPH e DPL. Bisogna notare che quando si ha a che fare con il registro DPTR quasi sempre si usa un byte alla volta.
Per esempio, per mettere nello stack il registro DPTR (operazione di push) si deve prima effettuare il push del registro DPL e poi quello del registro DPH. Non si può semplicemente effettuare il push del registro DPTR. Quindi il DPTR è usato per indirizzare la memoria esterna essendo composto da 2 registri a 8 BIT che unendosi formano un unico registro a 16 BIT cosi si può indirizzare una memoria esterna fino a 64Kb, da 0000h fino a FFFFh cioé da (0 a 65535.

2.12 TIMER REGISTRI

L'8051 opera sulla base di un clock fornito da un quarzo esterno.
E' possibile trovare sul mercato quarzi che
oscillano su una qualsiasi frequenza in funzione dell'applicazione richiesta. Nelle applicazione dell'MCU 8051 le frequenze più utilizzate sono 12 e 11,059 Megahertz. Il valore di frequenza 11,059 MHz non è un valore intero questo valore viene utilizzato per generare agevolmente la frequenza del generatore di baud rate della porta seriale .
I Microcontrollori usano i quarzi per sincronizzare le operazioni di lettura e scrittura della seriale operazioni raed-write della memoria, la gestione degli interrupt come del resto tutte le rimanenti operazioni. L'8051 usa il quarzo proprio per questo scopo. In effetti, l'8051 opera sulla base dei cosi detti "cicli macchina", un singolo ciclo macchina è la minima quantità temporale che un'istruzione dell'8051 richiede per eseguirla.
Comunque molte istruzioni richiedono più cicli macchina.
Normalmente un ciclo macchina corrisponde a 12 impulsi dell'oscillatore a quarzo. Quindi, se un'istruzione viene completata in un ciclo macchina, essa richiede 12 impulsi dell'oscillatore. Essendo nota la frequenza del quarzo a 11.059.000 cicli al secondo, possiamo calcolare quanti cicli macchina al secondo possono essere eseguiti dall'MCU:
11.059.000 / 12 = 921.583
Questo significa che l'8051, sincronizzato con un oscillatore a 11,059 MHz puo' eseguire 921.583 cicli
macchina al secondo, del resto la maggior parte delle istruzioni dell'8051 sono a ciclo singolo, quindi possiamo ritenere per grandi linee che l'8051 esegua circa un milione di istruzioni al secondo.
Precisamente, tenendo conto che il tempo di esecuzione dipende dal un numero di cicli macchina maggiore di uno, possiamo stimare una media di circa 600.000 istruzioni al secondo.
Es., se utilizzassimo esclusivamente istruzioni a 2 cicli macchina, l'8051 eseguirebbe 460.791 istruzioni al secondo. L'8051 ha anche due istruzioni molto lente che richiedono 4 cicli. Nel caso in cui per assurdo si utilizzassero solo queste istruzioni, l'8051 eseguirebbe soltanto 230.395 istruzioni al secondo.
E' importante considerare il fatto che non tutte le istruzioni vengono eseguite con la stessa durata.
L'istruzione più veloce richiede un ciclo macchina (12 impulsi di clock), molte altre richiedono 2 cicli
macchina (24 impulsi di clock) e le due più lente istruzioni matematiche richiedono 4 cicli macchina (48 impulsi di clock).
Poiché tutte le istruzioni richiedono quantità di tempo variabile, risulta evidente che sorge la necessità di quantificare il tempo impiegato per eseguire il programma, o la routine, o la chiamata di un interrupt di tipo hardware e software. Come si può tenere traccia del tempo trascorso in una

applicazioni che richiede risposte temporali molto critiche e soprattutto molto veloci. Fortunatamente, l'8051 dispone di timer che permettono di gestire gli eventi temporali con molta precisione. Nell'80C51 il microcontrollore integra al suo interno altri due componente fondamentale per lo svolgimento del processo di elaborazione, essi sono il TIMER 0 e il TIMER 1, questi dispositivi hanno la peculiarità di poter essere usati come temporizzatori o come contatori. Come temporizzatori vengono contati gli impulsi generati all'interno con frequenza fissa f_{12}, pari ad un dodicesimo della frequenza di oscillazione del quarzo utilizzato come generatore di clock. Nel secondo caso vengo contati i cambiamenti di stato da 0 a 1 sugli ingressi TIMER 0 e TIMER 1 che fisicamente corrispondono al pin P3.4 e P3.5 della porta 3. Per evitare che i disturbi di natura elettrica e magnetica possano falsare la lettura del singolo pin del TIMER 0 e del TIMER 1 questi due piedini di ingresso vengono testati ad ogni ciclo di macchina, e solamente quando le due transazioni saranno valide ovvero se i due livelli di ingresso rimangono stabili per 2 cicli macchina sarà possibile leggere e manipolare il dato, ovviamente la massima frequenza rilevabile sarà f_{24} pari a un ventiquattresimo della frequenza di clock. La composizione di questi due TIMER e formata da 2 registri ognuno da 8 BIT, TH_0 e TL_0 del TIMER 0, e glia altri 2 registri TH_1 e TL_1 del TIMER 1, gli indirizzi delle rispettive locazioni di memoria sono:

- **TH_0 8Ch**
- **TL_0 8Ah**
- **TH_1 8Dh**
- **TL_1 8Bh**

Questi registri possono essere settati tramite softwae con valori che vanno da 00h a FFh ovvero da 0 a 255.
Con questi ti di settaggi si possono realizzare contatori a 8/13/16 BIT , tramite il MODO operativo.

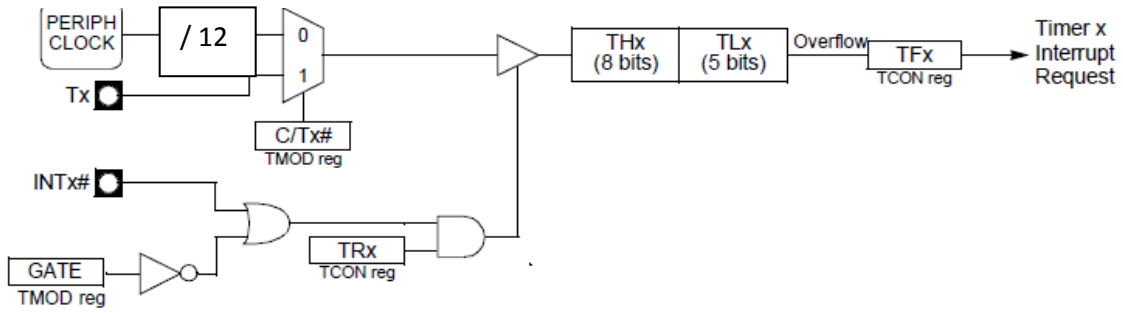


FIG 2.9

Il modo operativo dei TIMERs viene scelto attraverso i BIT C/T del registro Mode Control (TMOD) di indirizzo 89h della RAM interna SFR.

Registro TMOD 89h

Gate1	C/T	T1M1	T1M0	Gate0	CT	T0M1	T0M0

FIG. 2.10

Struttura del registro TMOD

TMOD 89h SFR

BIT	Nome	Funzione	Timer
7	GATE1	Quando questo BIT è settato a 1 il TIMER 1 è attivo solo quando il Pin della porta P3.3 si trova nello stato alto, ATTENZIONE resettando il TIMER 1 lo stesso sarà svincolato dallo stato del pin P3.3	1
6	C/T1	Quando questo bit e' settato il timer 1 conterà il numero degli eventi sul pin T1 (P3.5). Se il bit e' resettato il timer 1 verrà incrementato ogni ciclo macchina.	1
5	T1M1	BIT 1 di Modo del TIMER 1 (vedi tabella sotto)	1
4	T1M0	BIT 0 di Modo del TIMER 1 (vedi tabella sotto)	1
3	GATE0	Quando questo bit e' settato, il timer 0 e' attivo solo quando il pin P3.2 e' nello stato alto. Se tale bit e ' resettato il timer 0 sarà svincolato dallo stato del pin P3.2.	0
2	C/T0	Quando questo bit e' settato il timer 0 conterà il numero degli eventi sul pin T1 (P3.5). Se il bit e' resettato il timer 0 verrà incrementato ogni ciclo macchina.	0
1	T0M1	BIT 1 di Modo del TIMER 0 (vedi tabella sotto)	0
0	T0M0	BIT 0 di Modo del TIMER 0 (vedi tabella sotto)	0

FIG. 2.11

Tabella Modo di Funzionamento dei TIMERs

M1	M0	Modo	Descrizione
0	0	0	Timer a 13-BIT
0	1	1	Timer a 16-BIT
1	0	2	Timer a 8-BIT con Auto -reload
1	1	3	Timer in slitte mode

FIG. 2.12

Di seguito viene descritta anche la struttura del registro TCON (Timer Control Register), appartenente al complesso dispositivo per la gestione dei TIMERs.

Il TCON controlla i due timer e fornisce importanti informazioni sul loro funzionamento. Il registro TCON ha la seguente struttura:

TF1	TR1	TF0	TR0I	IE1	IT1	IE0	IT0

TCON (88h) SFR

BIT	Nome	Indirizzo	Funzione	Timer
7	TF1	8Fh	**Timer 1 Overflow**. Questo bit è settato quando il timer 1 è andato in overflow	1
6	TR1	8Eh	**Timer 1 Run**. Quando questo bit viene settato il timer 1 è abilitato, altrimenti è fermo.	
5	TF0	8Dh	**Timer 0 Overflow**.Questo bit e'settato quando il timer 0 è andato in overflow	
4	TR0	8Ch	**Timer 0 Run**. Quando questo bit viene settato il timer 0 è abilitato, altrimenti è fermo.	

FIG. 2.13

La scelta tra i due modi di funzionamento Timer & Counter è selezionabile tramite i BIT C/T del registro di Control Mode TMOD, specialmente un Zero su questo BIT pone il funzionamento del relativo Timer come Conta Tempi se invece sul BIT C/T si pone un 1 il modo di funzionamento sarà come conta eventi.
L'abilitazione al conteggio è controllata dai segnali di GATE, INT (interrupt) e TR vedi fig. 2.14, per la quale il GATE è gestibile via software inviando un opportuno valore all'omonimo BIT del registro TMOD, mentre INT è un piedino della porta PE (P3.3 per INT1 e P3.0 per INT 0) invece il TR è uno speciale BIT del registro TCON vedi fig.2.13.
Si ha la possibilità di inizializzare il contatore quando il piedino INT è a 1 e portarlo a zero, con questo cambio di stato si permette al MCU di misurare la durata dell'Impulso. Infatti, basta moltiplicare TR0 o TR1 e moltiplicare quando INT torna basso, il numero di Impulsi contati per la frequenza interna di conteggio

TR	GATE	INT	FUNZIONE
1	0	X	ABILITA
1	X	1	ABILITA
1	1	0	NON ABILITA
0	X	X	NON ABILITA

Fig. 2.14

Come si può notare dalla fig.2.11 esistono 4 modi di operare del Timer:

- Modo 0
- Modo 1
- Modo 2
- Modo 3

I due Timer dell'80C51 hanno 4 possibili modi di funzionamento, selezionabili mediante i BIT M_0 e M_1 del registro TMOD relativo ad ognuno di essi (BIT 0 e 1) per il Timer0 e (BIT 4 e 5) per il Timer1 secondo la fig. 2.11.

M_1	M_0	Modo
0	0	0
0	1	1
1	0	2
1	1	3

Fig. 2.15

MODO 0 – Ponendo i BIT M_1 e M_0 con "00" si configurano i due registri dl Timer come un contatore a 13 BIT(formato dai primi 8 BIT di TH e da 5 BIT di TL. fig. 2.16 che viene incrementato una volta abilitato, alla frequenza f_{12} fino ad avere un Overflow(ovvero da tutti i BIT 1 a tutti i bit 0) che attiva il Flip-Flop TF, questo, azzerato in modo automatico non appena viene caricata la routine di risposta cosi come anche gli omonimi BIT TF0 & TF1 del registro TCON.

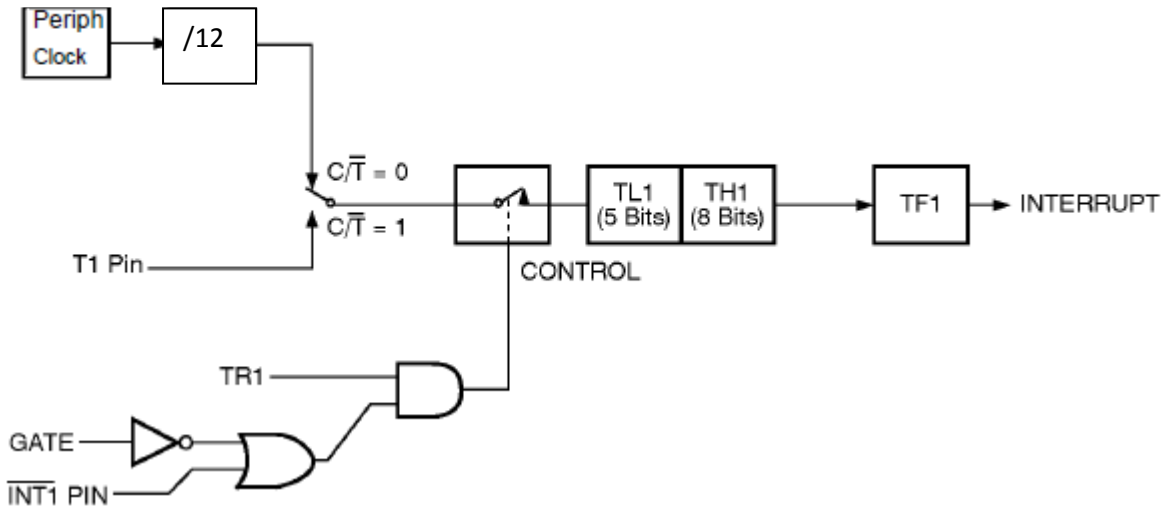


Fig. 2.16

MODO 1 – Questo modo viene selezionato inviando la configurazione di "01" nei BIT M_1 e M_0 del registro TMOD, questo settaggio utilizza tutti i BIT ovvero 16 BIT di TH & TL e si realizza cosi un contatore a 16 BIT. Questo contatore è pilotabile sia con un clock interno sia con un clock esterno a seconda di come si setta il BIT CH del TMOD register, mentre per il resto il funzionamento è analogo al Modo 0.

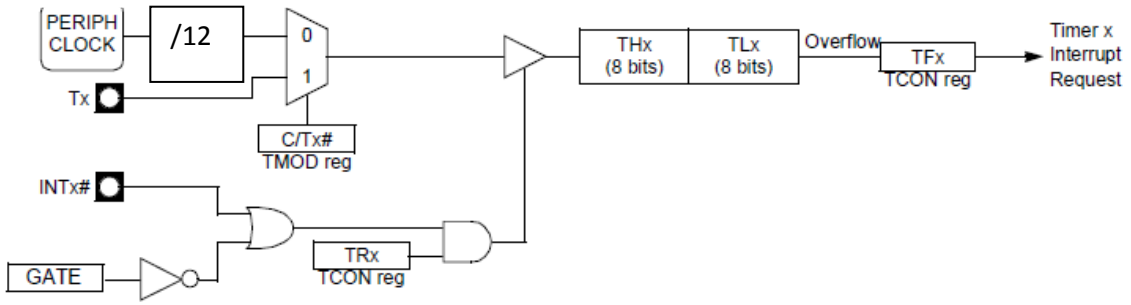


Fig.2.17

Modo 2 - Questo modo viene selezionato inviando la configurazione di "10" nei BIT M_1 e M_0 del registro TMOD,
prevede l'uso di TL come registro di conteggio, pilotato da una frequenza di clock f_{12} oppure da un segnale esterno applicato a TR & INT e caricato sia alla partenza che dopo ogni Overflow, con il valore pre-impostato via software nel registro TH, che funge così da 2° registro costante di tempo".

Questo ti di funzionamento, che non genera Interrupt, continua automaticamente finché l'abilitazione non viene annullata.

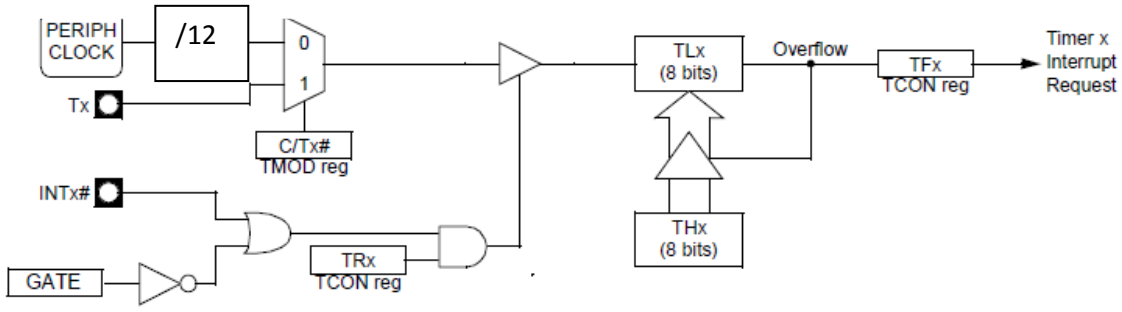


Fig. 2.18

MODO 3 - Questo modo viene selezionato inviando la configurazione di "11" nei BIT M_1 e M_0 utilizza solo i registri TH_0 e TH_1 di TIMER 0 mentre blocca il TIMER 1. In particolare TL_0 può funzionare come conta tempi con frequenza di ingresso f_{12}, o essere utilizzato come conta eventi in questo caso l'ingresso di start viene applicato al piedino T_0 esso viene controllato e gestito da C/F, GATE,TR e INT del TIMER 0 per poi attivare il filp-flop TF 0 e l'omonimo BIT di TCON in caso di overflow.
TH0, invece, può funzionare solo come contatempi f_{12} controllato dal BIT TR del TIMER 1 (TR1) di cui utilizza anche il flip-flop di interrupt (TF1), ed il relativo Flag.

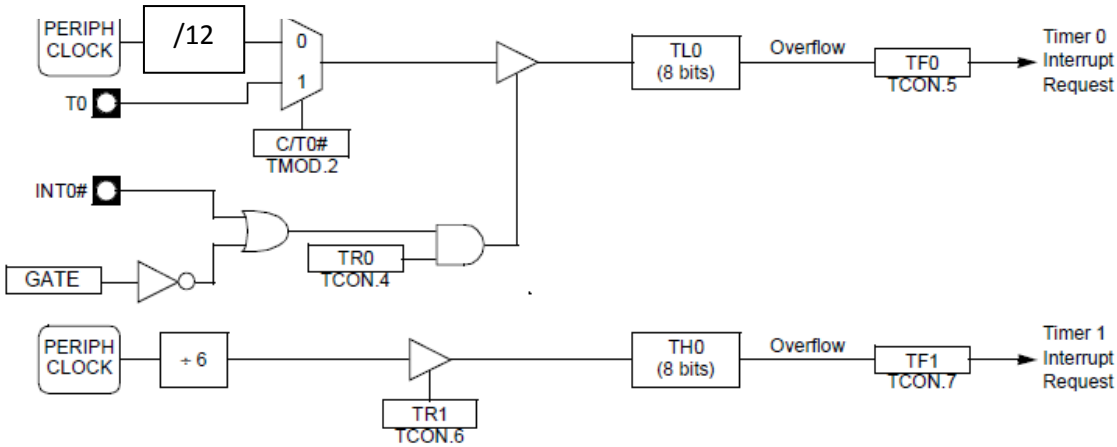


Fig. 2.19

CAPITOLO 3
Comunicazione con dispositivi esterni

3.1 PORTE I/O

Nell'80C51 sono disponibili 4 porte bidirezionali a 8 BIT, indicate con le sigle P0,P1,P2,P3, utilizzati BIT per BIT, per lo scambio di dati con l'esterno tramite 4 registri speciali associati a particolari locazioni di memoria RAM , vedi fig. 2.6 e sono indirizzabili a livello di BIT.

Indirizzi dei registri P0-P3

Registro	Indirizzo
P0	80h
P1	90h
P2	0Ah
P3	0Bh

Fig. 3.1

Configurazione hardware delle porte P0-P1-P2-P3

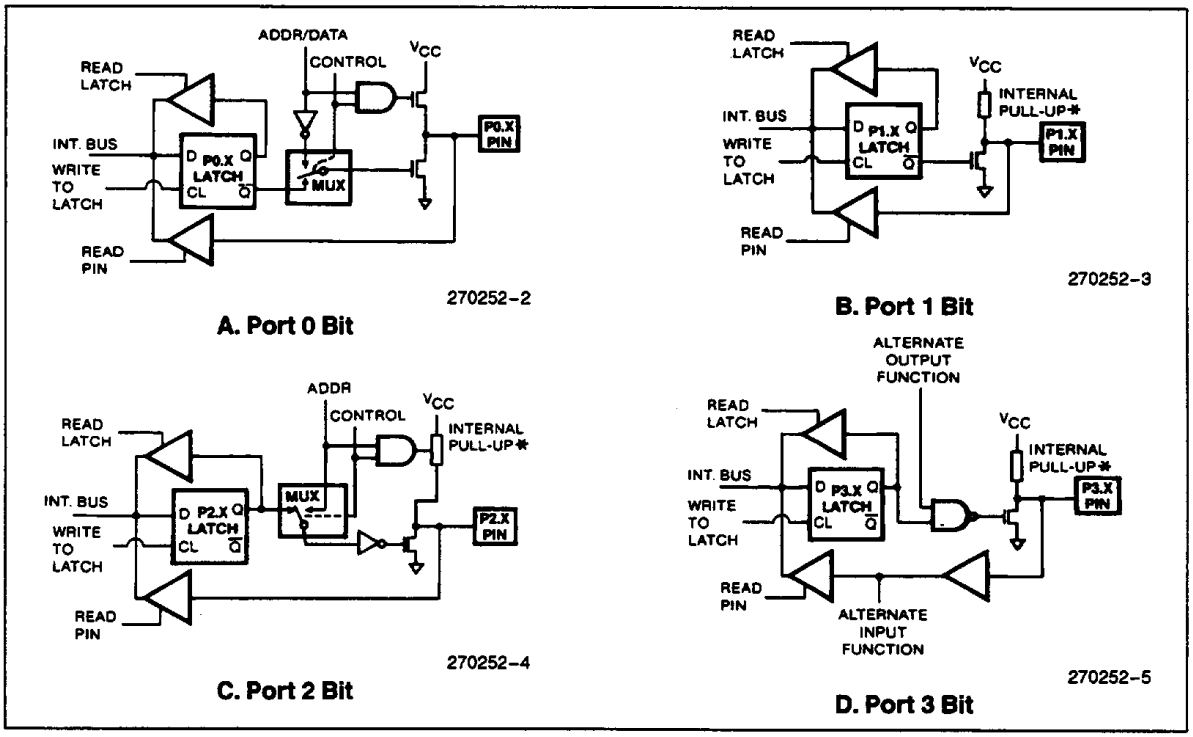


Fig. 3.2

Come si vede da fig. 3.2 è prevista la possibilità di leggere il livello logico presente nelle singole linee del latch(in alternativa alla lettura del più sterno) in modo da evitare livelli ambigui dovuti a disturbi e ai carchi esterni.

Es. se si collega un Transistor di tipo NPN con la base collegata ad una linea della porta, inviando un "1" in uscita la basa del transistor conduce, per cui le eventuali letture successive sul piedino della porta verrebbe letto il valore della base V_{be} che sarebbe interpretato come livello logico basso,

quindi si avrebbe una lettura sfalsata, ma leggendo il dato dal latch della porta, verrà ritrovato la stato logico precedentemente impostato, questo grazie al FET che fa da driver separatore tra Latch e piedino.
Per definire una linea come ingresso bisogna forzare con un istruzione del tipo ***MPV Px,dato*** un "1"sul latch corrispondente prima di iniziare una operazione di lettura dall'esterno.
Per le operazioni di uscita non occorre nessuna forzatura basta inviare il dato al latch della porta scrivendo l'omonimo registro.

- Porta 0
  La porta P0 può essere utilizzata sia come porta di I/O del tipo generico che come supporto per il BUS Dati e Bus Indirizzi. Nel secondo caso le 8 linee vengono prima utilizzate come uscita per i BIT A_0 – A_7 dell'indirizzo esterno, e poi configurate come BUS Dati.
  Con riferimento allo schema di fig. 3.2 tenuto presente che i FET del circuito conducono con il gate a "1" e che il multiplexer è collegato a ""
  Quando CONTROL = 0 e ADD/DATA = 0, si può notare che la linea d'usci $\overline{Q}$ la resistenza dei FET T1 come pull-up interno solo quando viene mandat[T d'indirizzo di valore "1" altrimenti si comporta come Open-Collector.

Possibili stati dei piedino della porta 0

Q	A	C	Condizione PIN
1	X	0	0
0	X	0	Floating
X	0	1	0
X	1	1	1

Fig. 3.3

Per avere il valore "1"sul pin, è necessario utilizzare una resistenza di pull-up.

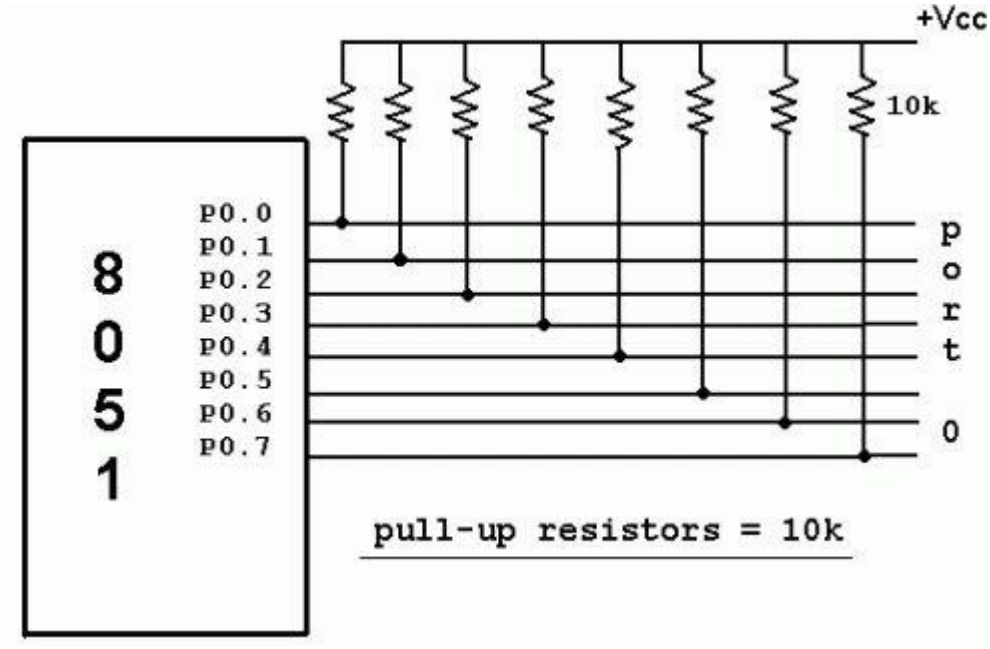


Fig.3.4

- **Porta 1**
  Porta bidirezionale con pull-up interno, utilizzabile per I/O generico. Va comunque tenuto conto che la resistenza riportata nella fig. 3.2 è solo simbolica in quanto il circuito di carico del FET di uscita, il sistema all'interno è molto complesso. Nella versione del MCU 8751, questa porta è usata come supporto per la parte meno significativa del BUS-ADDRESS ($A_0 – A_7$).

- **Porta 2**
  Anche questa porta con resistenza interna di pull-up, può essere utilizzata sia per I/O generico che come supporto per gli 8 BIT più significativi del BUS-Adrress verso le memorie esterne.
  Dalla fig. 3.2 risulta presente anche in questo caso un segnale interno di controllo che collega il multiplexer al dato o all'indirizzo, a seconda che si tratti di un generico I/O o di un accesso alla memoria. $\overline{Q}$
  Significativo risulta il funzionam[illegible]uesta porta quando si usa una memoria esterna DATI con indirizzi di 8 BIT ($A_0 – A_7$), in quanto essa, durante gli accessi, mette in uscita il contenuto dei suoi latch (registro P2) il che consente una gestione in pagine di più blocchi da 256 locazioni.

- **Porta 3**
  Questa è una porta bidirezionale utilizzabile per I/O generico o come supporto per il collegamento con l'esterno dei blocchi funzionali come (Timer, Interrupt Controller, porta seriale UART) presenti all'interno del MCU. Per predisporre il funzionamento nella seconda modalità, bisogna inviare degli "1" nei latch della porta (registro P3).

3.2 COMUNICAZIONE SERIALE UART

All'interno del chip è disponibile un blocco funzionale per trattamento dei dati seriale di 8 – 9 BIT, attraverso le linee TXD & RXD (P3.1 & P3.0) con velocità che vanno d 110 a 31Kb, che utilizza per lo scambio dati tra la CPU e un unico registro (SUBF), allocato all'indirizzo di memoria RAM al 99h.
Sono previsti 4 possibili modi di funzionamento:

- *Modo zero* *(Shift Mode)*
- *Modo uno* *(8 BIT Uart Mode)*
- *Modo due* *(9 BIT Uart Mode)*
- *Modo tre* *(9 BIT Uart Mode)*

Questi modi sono selezionabile e controllabili mediante il registro speciale SCON allocato alla memoria RAM interna di 98h.

Registro SCON 98h

SMO	SM1	SM2	REN	TB8	RB8	TI	RI

Fig. 3.6

SMO,SM1. Servono a selezionare uno dei quattro modi di funzionamento.

SM1	SM0	MODO
0	0	0
0	1	1
1	0	2
1	1	3

SM2. Posto a "1" , abilita il blocco di ricezione a chiedere un interrupt ponendo a "1" RI, se:
RB8 è uguale a 1 (Modo 2 e 3) il BIT di stop del dato ricevuto è corretto, vale cioè "1". Questo vale nel Modo 1 per il quale il BIT di stop è copiato in RB8, mentre nel modo 0 invece bisogna sempre imporre SM2=0.

REN. Posto a "1" abilità la ricezione.

TB8. Utilizzato nel Modo 2 & 3, il suo valore viene inviato in trasmissione come NONO BIT di dato, con particolari funzioni di controllo per sistemi multiprocessori.

RB8. Nel funzionamento in Modo 1 ricopia il valore del BIT di stop dei singoli dati ricevuti. In Modo 2 & 3 viene caricato con il NONO BIT del dato ricevuto (TB8). Nel Modo 0 non ha nessuna operatività.

TI. Utilizzato come flag di interrupt da parte del blocco di trasmissione, viene posto a "1" per chiedere alla CPU un nuovo dato da trasmettere e deve essere resettato dalla routine di risposta.

RI. Controllato dal blocco di ricezione, e posto a "1" per chiedere un interrupt:
a)quando è disponibile un dato per la CPU nello SUBF(Modo 0)
b) se il BIT di stop, copiato in RB8 è corretto (RB8=1), nel caso che SM2 sia stato posto a "1", oppure quando è disponibile un dato nel SUBF, nel caso SM2 sia stato posto a "0" (Modo 0).
c) Quando è disponibile un dato nel SUBF(SM2=0), oppure se il BIT 9 del dato ricevuto (TB8 del dato trasmesso), copiato in RB8, risulta "1", nel caso che SM2 sia stato posto a "1" (Modo 2&3). La routine di risposta deve provvedere poi al Reset.

Modo 0
Prevede la trasmissione e la ricezione di dati di 8 BIT, trattati, a partire dal meno significativo, con frequenza fissa e pari 1/12 della frequenza di clock.
La trasmissione inizia in corrispondenza S_6P_2 con un'operazione di scrittura del dato da trasmettere nello speciale SBUF (Serial Buffer) che abilità il blocco di controllo TX Control, e carica automaticamente a "1" il nono BIT (D_8) del registro di trasmissione.
Un ciclo macchina dopo il comando SEND, abilita la porta che collega l'uscita dello shift register di trasmissione al piedino RXD (P3.0) e contemporaneamente fa passare su TXD (P3.1) lo shift clock per eventuali sincronismi esterni. In questa prima fase si ha in uscita D_1 poi di seguito tutti gli altri. Man mano che i BIT del dato escono dalla destra dello shift register, dalla sinistra entrano degli Zeri generati dal flip-flop D di figura 3.7 Dopo sette shift, il registro di trasmissione conterrà la configurazione seguente:

D_7								
8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	1	D_7

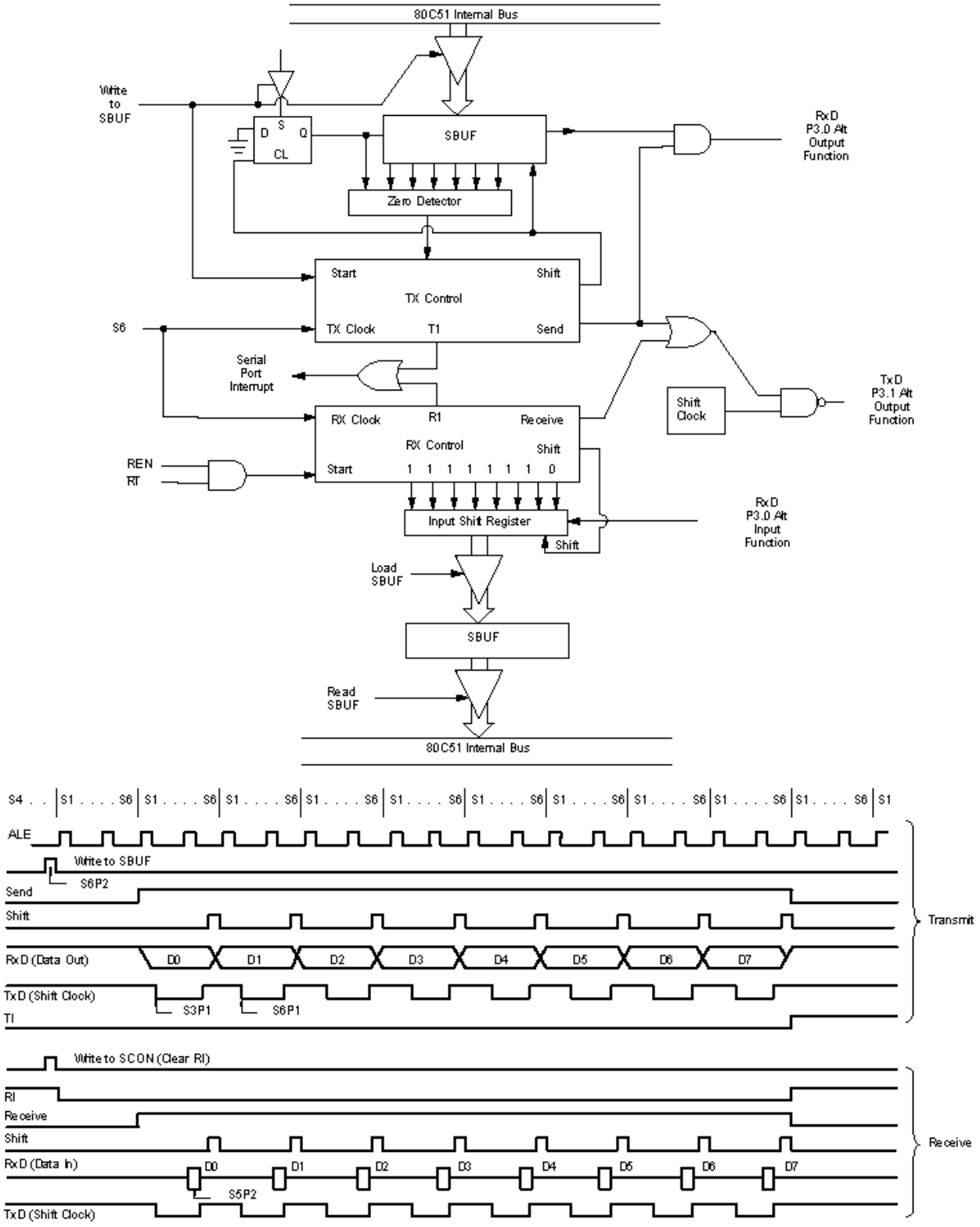


Fig. 3.5

Il blocco di controllo invierà, un altro impulso di shift, cosi disattiverà SEND e attiverà la richiesta di interrupt alla CPU per avere un altro dato. La ricezione abilitata da REN=1 quando RI è stato resettato, inizia con il caricamento della configurazione 1111 11110h(D_8 – D_0) nello shift di

ricezione mentre il comando RECEIVE dal blocco di controllo RX Control, collega lo shift clock a TXD.
Successivamente, in corrispondenza della stato S_5 P_2 di ogni ciclo macchina viene letto il valore presente su RXD ad in S_6 P_2 viene slittato verso sinistra il contenuto del registro d'ingresso. Quando lo zero, caricato inizialmente nel Bit meno significativo, raggiunge la posizione 7 nel registro d'ingresso:
In questo modo i dati, con il formato a 10 Bit vengono trasmessi con velocità ricavata dalla frequenza di Overflow del Timer 1 utilizzato come conta tempi in un qualsiasi dei tre possibili modi, secondo la formula:

$$\text{Baud rate} = \frac{2^{SMOD}}{32} \times f_1 \text{ (overflow)}$$

In cui l'esponente SMOD può essere programmato a 0 o 1 tramite l'omonimo Bit(Bit 7) del registro speciale PCON (Power Control), associato alla locazione di memoria 97h della RAM interna, in modo da dividere f_1 (Overflow) per 16 (SMOD=1) o 32 (SMOD=0)

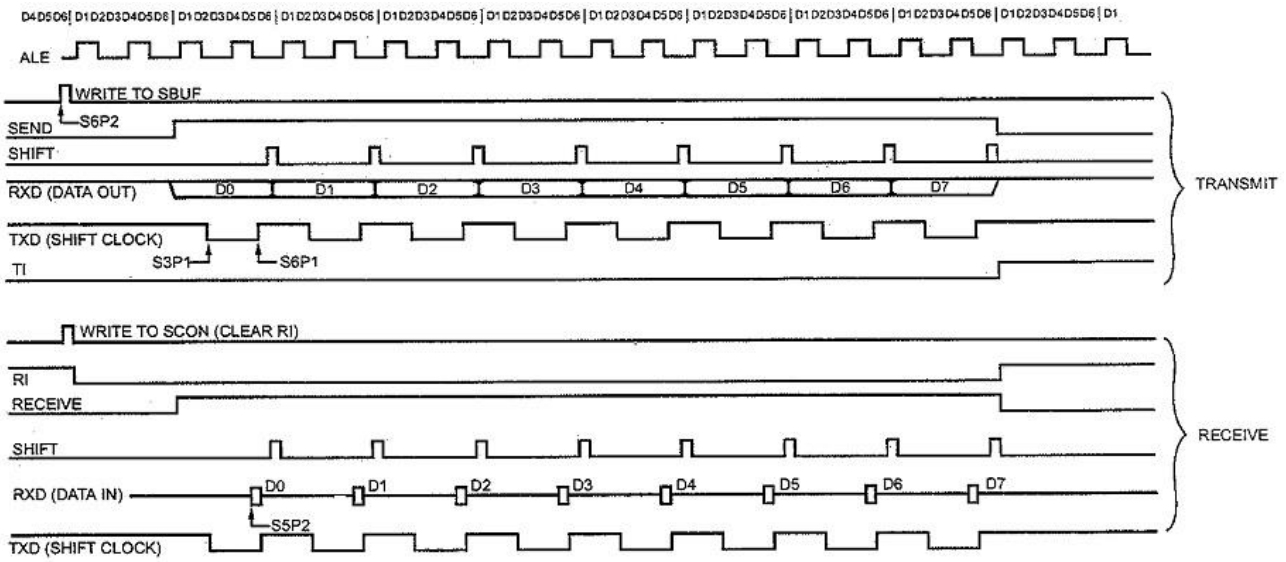


Fig. 3.6

3.3 MEMORIA

Memoria Esterna

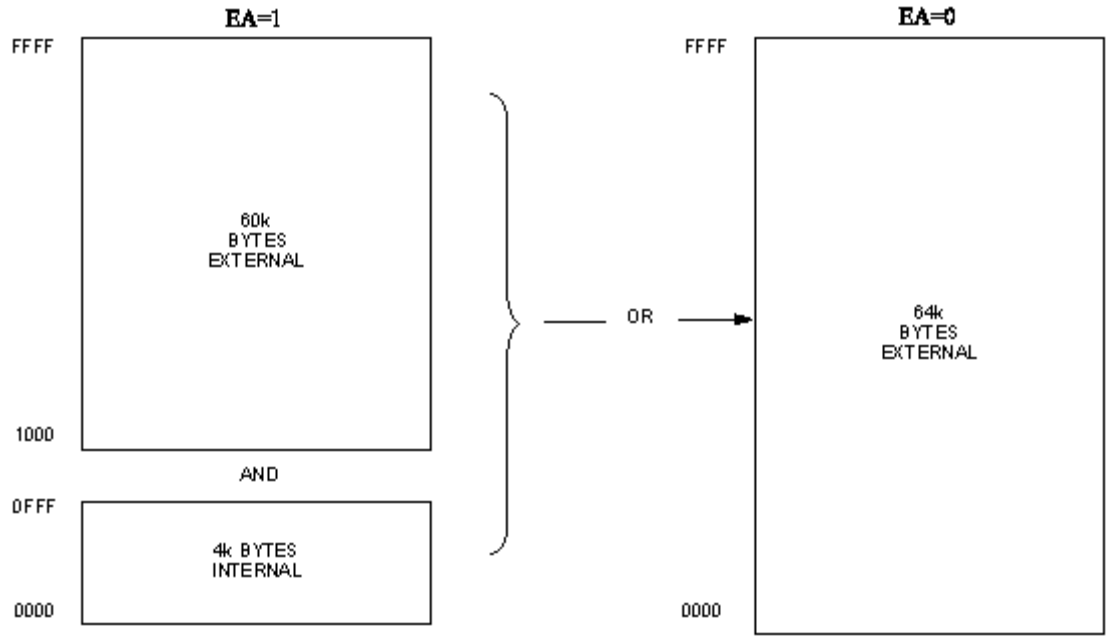
L'8051 gestisce la memoria in due segmenti separati :

- Memoria programma
- Memoria dati

La CPU può effettuare il fetch solo nella memoria di programma

Memoria di programma

La massima capacità della memoria di programma è di 64K di cui 4K possono risiedere dentro il chip.
Nel modo di funzionamento a memoria interna (EA=1) la CPU opera nei primi 4 K di indirizzamento, esegue il fetch nella memoria interna, e per indirizzi superiori 0FFFH il fetch viene effettuato nella memoria esterna
Nel modo di funzionamento a memoria esterna la CPU esegue sempre il fetch, per tutta l'area di indirizzamento
(0000....FFFFH), della memoria esterna.
Durante l'accesso alla memoria esterna la, Cpu sfrutta P0 come parte bassa del bus indirizzi o bus dati e P2 come parte alta del bus indirizzi, di conseguenza non è più possibile sfruttarle come porte I/O.



Collegamenti della memoria programmi esterna

La memoria programmi esterna può essere collegata come in figura (in questo schema la memoria per i programmi esterna viene abilitata solo quando il micro esegue i fetch al di sopra dell'indirizzo 0FFFH, EA=1).

Indirizzi	Memoria
0000H, 0FFFH	ROM interna
1000H,1FFFH	EPROM esterna

Gli indirizzi per la memoria programmi vengono emessi dalle porte PO e P2 (rispettivamente parte bassa e parte alta degli indirizzi). La porta PO serve anche per ricevere e trasmettere i dati: la porta PO costituisce quindi I'interfaccia di un bus indirizzi/dati multiplexato. La parte bassa dell'indirizzo viene emessa prima del trasferimento del dato; deve essere memorizzata temporaneamente fino al completamento dell'operazione di trasferimento (si faccia riferimento ai diagrammi temporali di lettura e scrittura sulla memoria esterna).

Andamento temporale di un ciclo di lettura di memoria di programma esterna.

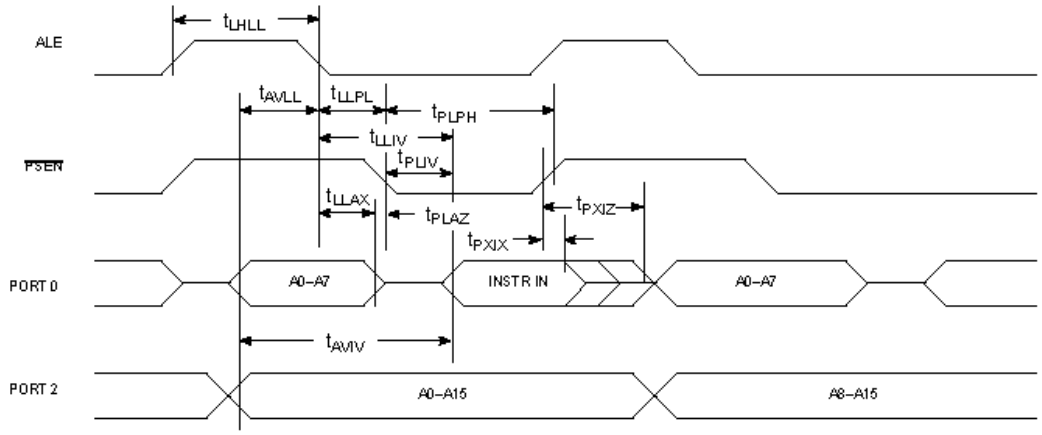


Fig. 3.7

I1 segnale *ALE* (Address Latch Enable) indica quando è presente un indirizzo e quindi può essere utilizzato come segnale di clock per il latch che congela temporaneamente la parte bassa dell'indirizzo.
I1 segnale #PSEN indica che I'operazione di lettura avviene con la memoria esterna e viene utilizzato come segnale di abilitazione delle uscite della ROM collegata. Si noti che il segnale #PSEN è attivo tutte le volte che 1'8051 deve effettuare un fetch del codice di un'istruzione dalla memoria esterna per i programmi.

- **MOMORIA RAM**

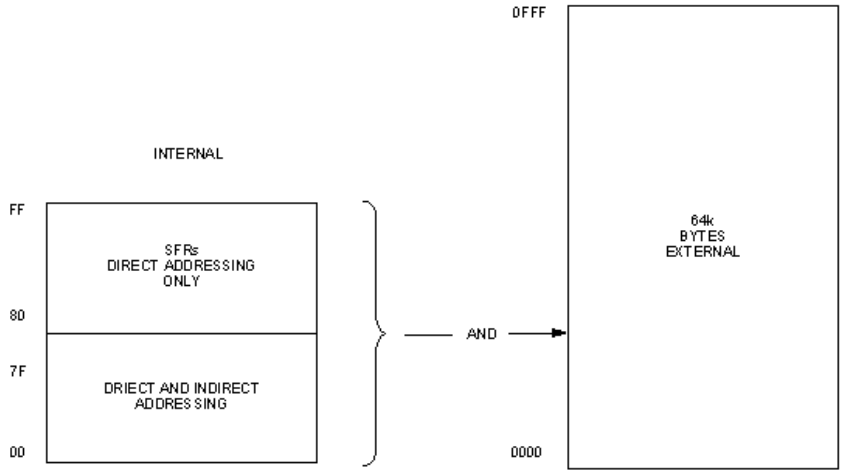


Fig. 3.7

I registri interni della CPU de11'8051 sono mappati in un'area di RAM interna che va dall'indirizzo 80H all'indirizzo OFFH detta **SFR area** (Special Function Register area).
I 128 byte di RAM interna utilizzabili come area per i dati sono mappati invece dall'indirizzo 00H all'indirizzo 7FH.
Non tutte le locazioni della SFR area (da 80H a OFFH) sono utilizzate per contenere i registri. Le rimanenti locazioni sarebbero teoricamente utilizzabili per contenere dati, ma la Intel raccomanda di non utilizzarle per questo scopo poiché il risultato sarebbe la lettura di dati casuali e la scrittura in tali locazioni non avrebbe alcun effetto.
Queste locazioni sono riservate alla ditta costruttrice del chip per futuri sviluppi.

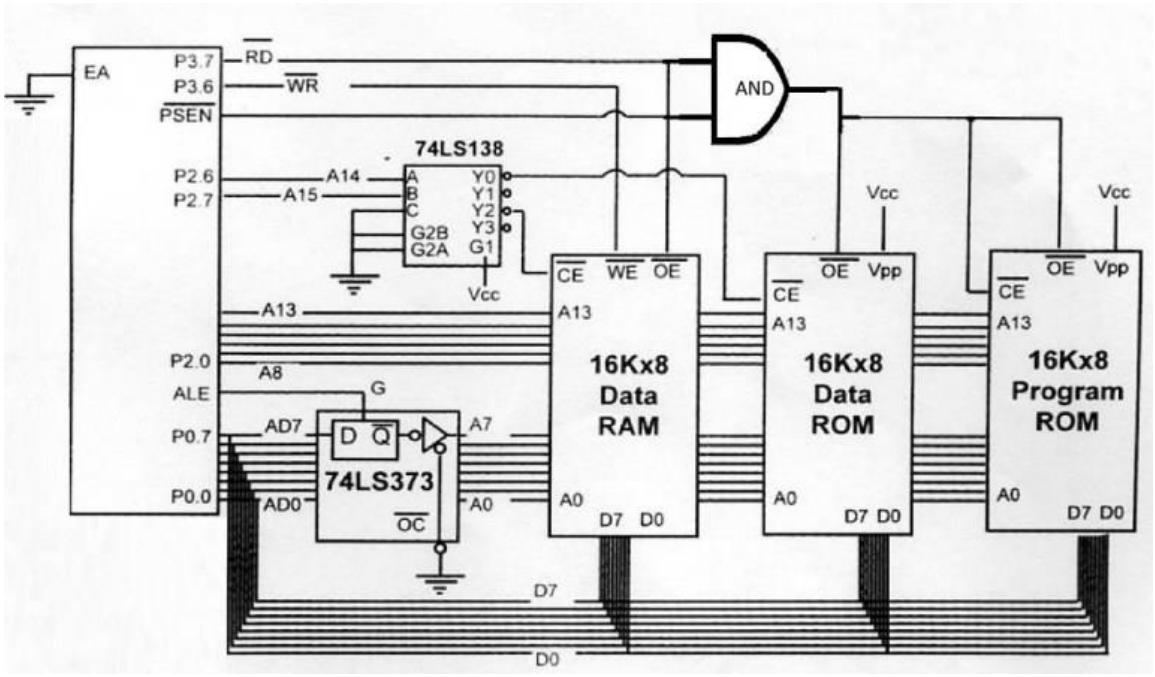


Fig.3.8

In figura viene mostrato il collegamento con una memoria RAM per i dati. Anche qui si utilizzano le porte PO e P2 come bus indirizzi e dati. Vengono inoltre utilizzati i segnali P3.7 (RD)e P3.6

(WR) della porta P3 per l'abilitazione in lettura e in scrittura della RAM. I1 segnale PSEN non viene utilizzato poiché la sua funzione è quella di abilitare in lettura una memoria esterna per i programmi (si noti inoltre che EA viene forzato a Vcc in modo da utilizzare per i programmi i 4 k di ROM interna).
In ambedue i collegamenti vengono sacrificate le due porte PO e P2 e parte della porta P3 per generare tutti i segnali necessari alle memorie esterne.
Si tenga comunque presente che, ad esempio, se si utilizza una memoria RAM esterna più piccola di 64 k non è necessario utilizzare tutti i bit della porta P2 (lo stesso dicasi per la memoria programmi).
La memoria esterna per i programmi e per i dati può essere combinata applicando il segnale PSEN e RD ad una porta AND a due ingressi e usando l'uscita della porta come segnale di lettura per la memoria dati/programmi.

Andamento temporale di un ciclo di lettura su memoria dati esterna.

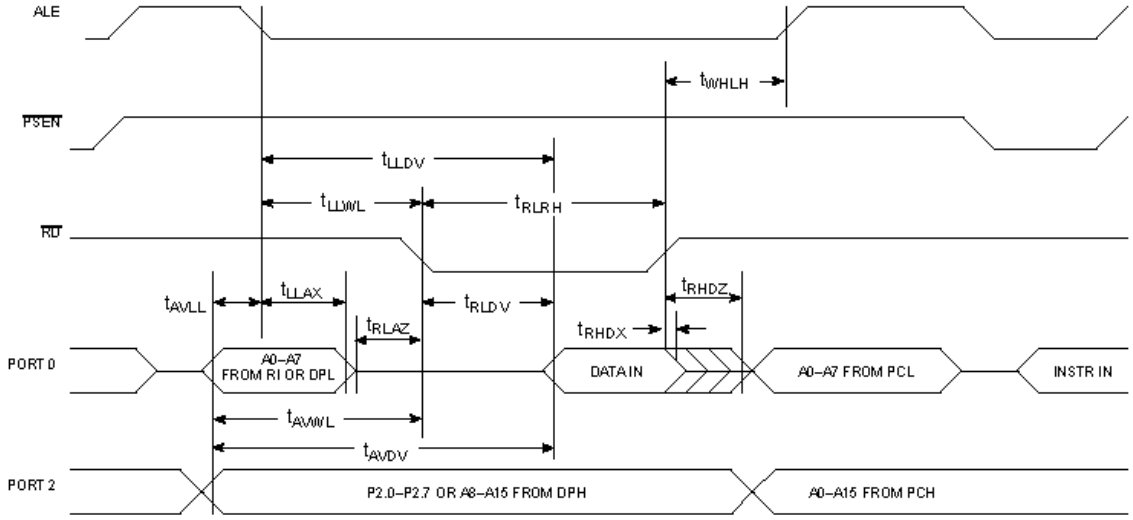


Fig.3.9

Andamento temporale di un ciclo di scrittura su memoria dati esterna.

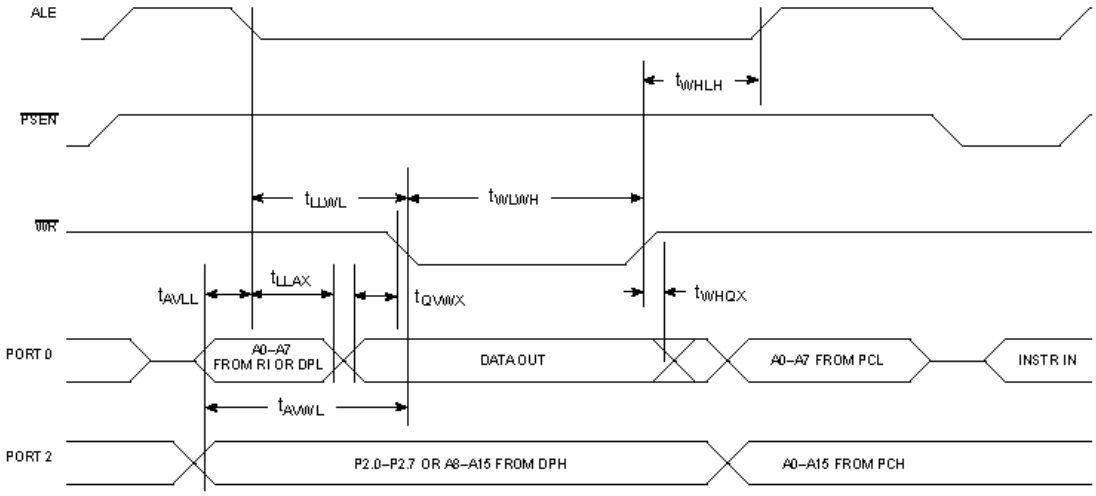


Fig.3.10

3.4 INTERRUPTS

Come implicito nel nome stesso, un **interrupt** è un evento che interrompe la normale esecuzione di un programma.
Praticamente il flusso di un programma e sempre di tipo sequenziale e può essere alterato da particolari istruzioni che intendono modificare il flusso del programma stesso.
Gli interrupt forniscono un meccanismo di congelamento del flusso del programma in corso, eseguono una subroutine (sottoprogramma) e ripristinano il normale funzionamento del programma come se nulla fosse accaduto.
Queste subroutine,denominate "interrupt handler" (gestore di interrupt) viene eseguita soltanto quando si verifica un determinato evento che attiva l'interrupt. L'evento può essere:

- il timer che va in overflow
- la ricezione di un byte dalla seriale, la fine della trasmissione di un byte dalla seriale
- uno o due eventi esterni.

L'8051 può essere configurato per gestire un interrupt handler per ognuno di questi eventi.
La capacità di interrompere la normale esecuzione di un programma quando un particolare evento si verifica, rende il programma stesso molto più efficiente nel gestire processi asincroni. Qualora non si dispone degli interrupt e si volesse gestire più eventi dal programma principale, bisognerebbe scrivere un codice poco elegante e difficile da capire.
Per esempio, si suppone di avere una grande memoria programma di 16k che esegue molte subroutine per
eseguire molti lavori. Si suppone inoltre che il programma debba automaticamente far cambiare lo stato del pin P3.0 ogni volta che il timer 0 va in overflow.
Il codice per effettuare tale azione non e' tanto difficile:

```
JNB TF0,SKIP_TOGGLE
CPL P3.0
CLR TF0
SKIP_TOGGLE: ...
```

Poiché il flag TF0 è settato ogni volta che il timer 0 va in overflow, il codice di sopra farà cambiare lo stato di P3.0 di conseguenza. Questo parte di programma non è efficiente. L'istruzione JNB consuma 2 cicli d'istruzione per determinare che il flag non è attivo e saltare al codice non necessario. Quando il timer va in overflow le due istruzioni CPL e CLR richiedono due cicli d'istruzione per essere eseguite. Per semplificare i calcoli ammettiamo che il resto del codice nel programma richieda 98 cicli d'istruzione. Quindi, in totale il codice consuma 100 cicli d'istruzione. Se il timer e' configurato in modo 16-bit, esso andrà in overflow ogni 65536 cicli macchina. In tutto questo tempo dovremmo aver eseguito 655 JNB test per un totale di 1310 cicli d'istruzione ed altri 2 per eseguire il vero codice utile. Per ottenere lo scopo si è speso il
2,002% del tempo per controllare quando cambiare lo stato di P3.0. Il codice è brutto perché si dovrebbe ripetere questa operazione di controllo per ogni ciclo del loop del programma principale.
Per fortuna, questo non serve. Gli interrupt permettono di dimenticare di controllare la condizione che scatena l'interrupt stesso. Il microcontrollore eseguirà il controllo automaticamente e quando la condizione

sarà vera, richiamerà la subroutine (chiamata interrupt handler), eseguirà il codice in esso contenuta e ritornerà. In questo caso la subroutine dovrebbe ridursi a:

CPL P3.0
RETI

Prima di tutto va notato che l'istruzione CLR TF0 e' sparita. Questo perché, quando l'8051 esegue la routine dell'interrupt del timer 0, cancella automaticamente il flag TF0. Inoltre invece della normale istruzione RET si è usato l'istruzione RETI. Quest' ultima fa le stesse cose della precedente ma informa l'8051 che la routine di interrupt è finita.
Bisogna sempre terminare un interrupt handler con RETI.
Tornando all'esempio precedente, ogni 65536 cicli d'istruzione eseguiremo le istruzioni CPL e RETI che
richiedono solo 3 cicli. Come risultato finale il nostro codice è 437 volte più efficiente del precedente senza contare che è più facile da leggere e da capire. Bisogna solo predisporre l'interrupt e dimenticarsi, fiduciosi che l'8051 eseguirà il codice quando serve.
Lo stesso concetto si applica alla ricezione dei dati dalla porta seriale. Una possibilità è quella di controllare continuamente lo stato del flag RI in un loop infinito. Oppure si potrebbe controllare il flag all'interno del più grande loop del programma principale. Nel peggiore dei casi si potrebbe correre il rischio di perdere il carattere ricevuto. Se un carattere viene ricevuto subito dopo che e' stato effettuato il controllo del flag RI, nella prosecuzione del resto del programma, prima di controllare di nuovo RI, un nuovo carattere potrebbe arrivare e si sarebbe perso il precedente. Usando l'interrupt, l'8051 ferma il programma e richiama l'interrupt handler per servire la ricezione del carattere. Così non si sarà costretti a scrivere un antiestetico controllo nel codice e nemmeno correre il rischio di perdere i caratteri ricevuti.

Quali sono gli eventi possono attivare un interrupt.

Possiamo configurare l'8051 affinché ognuno di questi eventi possa causare un interrupt:
Overflow del Timer 0
Overflow del Timer 1
Ricezione/trasmissione di un carattere dalla seriale
Evento esterno 0 (INT 0).
Evento esterno 1 (INT 1).
Ovviamente è necessario poter distinguere tra vari interrupt ed eseguire dei codici differenti per ognuno di
loro. Quando l'interrupt viene attivato, il microcontrollore salterà a degli indirizzi di memoria predefiniti come mostrato nella fig.3.11 che segue:

Interrupt	Flag	Indirizzo dell'interrup handler
INT 0	IE0	0003h
Timer 0	TF0	000Bh
INT 1	IE1	0013h
Timer 1	TF1	001Bh
Seriale	RI/TI	0023H

Fig.3.11

Se si consulta la tabella, si può affermare che, se il timer 0 va in overflow, il programma principale sospenderà momentaneamente il programma e salterà all'indirizzo 000Bh dove si dovrà scrivere l'interrupt handler opportuno.

- **Configurare gli interrupt**

Al power up, per default, tutti gli interrupt sono disabilitati, questo significa che, anche se per esempio il flag TF0 è attivo, l'interrupt non verrà accettato.
Il programma deve specificatamente dire all'8051 che
intende abilitare la gestione degli interrupt e indicare quali interrupt sono abilitati ad essere serviti.
Il programma può abilitare o disabilitare gli interrupt mediante il registro IE (A8h):

REGISTRO IE (A8h)

Bit	Nome	Indirizzo Bit	Funzione dell'interrupt Handler
7	EA	AFh	Abilita globalmente gli interrupt
6	-	AEh	Non definito
5	-	ADh	Non definito
4	ES	ACh	Abilita l'interrupt della seriale
3	ET1	ABh	Abilita l'interrupt del Timer 1
2	EX1	AAH	Abilita l'interrupt esterno INT1
1	ET0	A9h	Abilita l'interrupt del Timer 0
0	EX0	A8h	Abilita l'interrupt esterno INT1

Fig.3.12

Come si può notare, ciascun interrupt dell'8051 dispone si un suo bit nel registro IE. Si può abilitare un
determinato interrupt, settando il corrispondente bit. Per esempio, se si deve abilitare l'interrupt del Timer
1, si possono usare ambedue le seguenti istruzioni:

MOV IE,#08h
Oppure

SETB ET1

Ambedue le istruzioni settano il bit 3 di IE, e abilitano l'interrupt del Timer 1. Affinché l'interrupt del Timer 1
(e questo vale anche per gli altri) sia effettivamente abilitato, è necessario settare anche il bit 7 di IE (Attivazione degli interrupt globale), se tale
bit rimane a zero, nessun interrupt anche abilitato sarà attivo. Porre ad uno il bit 7 di IE abilita globalmente
tutti gli interrupt i cui bit sono allo stato on.
Questo bit è utile durante l'esecuzione di un programma che abbia una porzione di codice con criticità
temporali. Allora, per evitare che quella parte di codice possa essere interrotta da un qualsiasi interrupt, sarà
sufficiente resettare il bit 7 di IE e settarlo di nuovo quando la sezione critica è terminata.

Sequenza di Polling

Durante ogni istruzione, l'8051 verifica che non vi sia un interrupt da servire. Durante tale controllo, esso
segue il seguente ordine:

- Interrupt esterno INT 0
- Interrupt Timer 0
- Interrupt esterno INT 1
- Interrupt Timer 1
- Interrupt della seriale

Questo significa che, se sul'interrupt della seriale si attiva nello stesso momento quello esterno INT 0,
quest'ultimo verrà servito per primo e quello relativo alla porta seriale immediatamente dopo.

Priorita' degli Interrupt

L'8051 dispone di due livelli di priorità degli interrupt: **alto e basso**. Usando tali priorità potete cambiare
l'ordine con il quale gli interrupt vengono serviti.
Per esempio, se si ha abilitato sia l'interrupt del Timer 1 che quello della porta seriale e si ritiene che la
ricezione di un carattere sia molto più importante della gestione del timer, si può assegnare alla seriale un
priorità alta, in maniera da cambiare l'ordine standard con il quale il micro serve normalmente gli interrupt.

La priorità degli interrupt è controllata dal registro
IP (B8h).
Esso ha il seguente formato:

Bit	Nome	Indirizzo Bit	Funzione
7	-	-	Non definito
6	-	-	Non definito
5	-	-	Non definito
4	PS	BCh	Priorità Interrupt Seriale
3	PT1	BBh	Priorità Interrupt Timer1
2	PX1	BAh	Priorità Interrupt esterni INT1
1	PT0	B9h	Priorità Interrupt Timer0
0	PX0	B8h	Priorità Interrupt esterni INT0

Fig.3.13

Quando consideriamo la priorità degli interrupt, vanno applicate le seguenti regole:

- Nulla può interrompere un interrupt ad alta priorità, nemmeno un altro dello stesso tipo.
- Un interrupt ad alta priorità può invece interromperne uno a bassa priorità.
- Un interrupt a bassa priorità può essere iniziato soltanto quando nessun altro interrupt è attivo.
- Quando si verificano due richieste contemporanee di interrupt, quello a priorità più alta viene servito per primo. Se ambedue gli interrupt hanno la medesima priorità allora viene scelto secondo l'ordine della
  sequenza di **polling**.

Cosa succede quanto si verifica una richiesta d'interrupt?
Al momento che un interrupt viene richiesto, il microcontrollore automaticamente esegue le seguenti
operazioni:

- Il valore del Program Counter viene salvato nello stack, a partire dal byte meno significativo.
- Tutti gli interrupt della medesima o più bassa priorità sono bloccati.
- Se l'interrupt riguarda o un timer o una richiesta esterna, viene disattivato il flag corrispondente.
- L'esecuzione del programma salta all'indirizzo dell'interrupt handler corrispondente.

Una speciale attenzione è richiesta circa il terzo passo nel quale il flag viene automaticamente cancellato.
Ciò significa che non è necessario farlo all'interno del codice.

Che succede quando un interrupt finisce?
Un interrupt termina quando viene eseguita l'istruzione RETI (Return from Interrupt). A quel punto il
microcontrollore esegue i seguenti passi:

- Due byte vengono prelevati dallo stack (operazione di pop) e messi nel Program Counter per tornare al
  programma principale.
- Lo stato dell' interrupt è rimesso nella condizione precedente all'inizio dell'interrupt stesso.

Interrupt seriali
Gli interrupt seriali sono leggermente diversi da tutti gli altri. Ciò è dovuto al fatto che ci sono due flag di interrupt: **RI e TI**. Se uno dei due è attivo allora anche l'interrupt della seriale diventerà attivo. Questo vuol
dire che, al momento di una richiesta di interrupt sulla seriale, non si conosce quali dei due o tutti e due i
flag sono attivi. Allora, l'interrupt handler deve verificare lo stato dei flag e comportarsi di conseguenza. Inoltre deve anche cancellare i flag poiché l'8051 volutamente non lo fa in maniera automatica.
Es.:

```
                JNB
INT_SERIAL:
                RI,CHECK_TI     ;       Se il flag RI non e' attivo
                        vai a CHECK_TI
                MOV A,SBUF      ;       leggi il buffer di ricezione
                CLR RI          ;       Cancella il flag RI
CHECK_TI: JNB TI,EXIT_INT ;             Se il flag TI non e'
                                        attivo vai a EXIT_INT
                CLR TI                  Cancella il flag TI prima
                                        di inviare un nuovo carattere
                MOV SBUF,#?A?   ;       Copia il nuovo carattere da inviare nel buffer di
                                        trasmissione.
EXIT_INT: RETI
```

Una Importante Considerazione sugli interrupt: Preservare il valore di un Registro

Una regola importante deve essere applicata da tutti gli interrupt handler: Ogni interrupt deve lasciare il
processore nel medesimo stato che esso aveva al momento di iniziare l'interrupt stesso.
Il programma principale non tiene conto del fatto che i vari interrupt sono eseguiti d*i* nascosto.
Si prenda in considerazione la seguente porzione di codice:

```
                CLR C           ;       Cancella il carry
                MOV A,#25h      ;       Carica 25h nell'accumulatore
                ADDC A,#10h     ;       Aggiungi 10h, tenendo conto
                        del carry
```

Al termine dell'esecuzione di queste istruzioni, l'accumulatore conterrà il valore 35h.
Ma cosa accadrebbe se appena conclusa l'istruzione MOV viene servito un interrupt e questo cambia sia il
carry che il valore dell'accumulatore ponendolo ad uno? Quando l'interrupt restituisce il controllo al programma principale, l'istruzione ADDC addizionerà 10h a 40h e aggiungerà 1h perché troverà il bit di carry settato. In questo caso l' accumulatore conterrà il valore 51h invece di 35h.
Quello che è successo nella realtà, è che l'interrupt handler non ha preservato il valore del registro che ha usato. Però, Intel dice : ***Ogni interrupt deve lasciare il processore nel medesimo stato che esso aveva al momento di iniziare l'interrupt stesso.***
Vuol dire che, se l'interrupt usa l'accumulatore, deve assicurare che il valore di esso rimanga inalterato; ciò viene ottenuto con delle operazioni di PUSH e POP. Per esempio:

```
PUSH ACC
PUSH PSW
MOV A,#0FFh
ADD A,#02h
POP PSW
POP ACC
RETI
```

Le istruzioni dell'interrupt handler sono MOV e ADD che modificano sia l'accumulatore che il bit di carry;
allora è necessario salvare nello Stack Pointer sia l'accumulatore che il registro PSW con due operazioni di PUSH.
Una volta eseguite le due istruzioni proprie della routine di interrupt dovranno essere ripristinati i valori dei due registri con due operazioni di POP (Attenzione all'ordine inverso con il quale devono essere eseguite). A questo punto è possibile terminare l'interrupt con l'istruzione RETI.
Il programma principale troverà la situazione dei suoi registri immutata e quindi non si accorgerà minimamente dell'interruzione.
In generale, la routine di interrupt deve preservare il contenuto dei seguenti registri:

- PSW
- DPTR (DPH/DPL)
- PSW
- ACC
- B
- Registers R0-R7
-

In particolare fare attenzione al registro PSW, che contiene i flag di stato del processore. E' buona norma
salvare sempre il contenuto di PSW mediante le operazioni di push e pop all'inizio e alla fine della routine
d'interrupt, *a meno che non si è assolutamente sicuri* di non modificare alcun bit di PSW.
Notare che l'assembler non permette di eseguire la seguente istruzione:
PUSH R0
Questo è dovuto al fatto che l'indirizzo di R0 dipende dal banco di registri "R" selezionato e potrebbe riferirsi alla ram interna in posizione 00h o 08h o 10h oppure 18h. Perciò, se si sta usando un registro "R" all'interno della routine di interrupt, effettuare l'operazione di push facendo riferimento all'indirizzo assoluto del registro stesso.
Per esempio, invece di usare PUSH R0, dovreste usare:
PUSH 00h
Naturalmente l'istruzione è corretta solo se si sta utilizzando il set di registri di default, altrimenti si deve fare il PUSH dell'indirizzo assoluto corrispondente al registro che effettivamente si sta usando.

Problemi Comuni con l'uso degli Interrupt
Gli interrupt sono degli strumenti molto potenti, ma se usati in maniera non corretta, sono una fonte di ore spese a trovare problemi nel programma.
Gli errori sulle routine di interrupt sono spesso molto difficili da diagnosticare e correggere.
Se si sta utilizzando degli interrupt e il programma va in crash oppure si hanno dei comportamenti strani con
risultati di tipo random, si faccia ammenda di verificare che le regole appena esposte non siano state violate.
In generale i problemi più comuni derivano da:

- **Non aver preservato il valore di un registro che viene usato nella routine di interrupt.** Controllare che tutti i registri in uso siano preventivamente salvati nello stack con un'operazione di push
- **Non aver ripristinato il valore del registro prima dell'uscita dalla routine dell'interrupt.** Controllare di aver effettuato lo stesso numero di operazioni di pop in ordine inverso
- **Aver usato l'istruzione RET invece di RETI oppure non aver usato alcuna istruzione di return.** In questo caso l'interrupt non termina e quindi verrà eseguito una volta soltanto e poi misteriosamente sembrerà come se esso fosse bloccato in realtà l'MCU entra in un LOOP.
- **Non aver chiuso la routine di interrupt con l'istruzione RETI**.

Alcuni simulatori dell'8051,hanno delle speciali caratteristiche che notificano il fatto che se si è dimenticato di preservare il valore di un registro oppure commesso un qualche errore riguardante l'uso improprio della routine di interrupt grazie al software si potranno vedere e trovare

3.5 RESET

Di seguito viene descritto il modo RESET, ovvero riportare tutti i valori degli SFR a 0000 0000b (b sta per numerazione binaria).
Il reset vedi S1 nella figura 3.14 impone al MCU di azzerare tutti gli stati dei registri e re-iniziare dal codice programma alla posizione 0000h

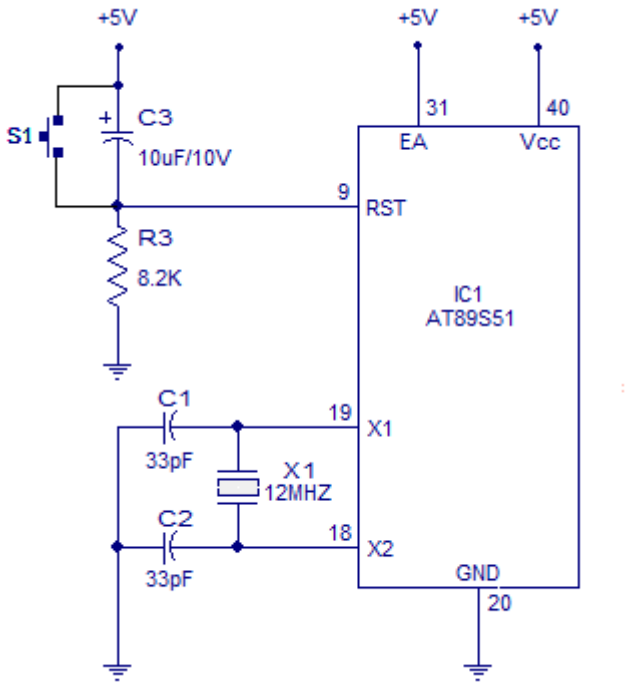


Fig.3.14

3.6 INPUT OSCILLATORE

Le famiglie di microcontrollori Intel (MCS-51,) contengono un circuito comunemente denominato "oscillatore su chip". Il circuito su chip non è di per sé un oscillatore, ovviamente, ma un amplificatore adatto per essere utilizzato come parte di un oscillatore di feedback.
Con una corretta selezione di componenti fuori chip, questi circuiti dell'oscillatore funzioneranno meglio di ogni altro tipo di oscillatore di clock.
I circuiti suggeriti sono semplice, economici, stabili e affidabili.
L'uso del doppio cristallo di quarzo da 50 ohm. Intel
non "garantirà il funzionamento" con cristalli da 50 ohm. In effetti, Intel garantisce
solo ciò che è incorporato all'interno di un prodotto Intel. Rispetto a come si dovrebbe ottimizzare il clock dell'MCU la loro selezione dovrebbe essere valutata in ambito progettuale. Si dovrebbe ridurre al minimo il tempo di avvio e massimizzare la stabilità della frequenza. Normalmente in molte applicazioni, né il tempo di avvio né la stabilità della frequenza sono particolarmente critici e le "raccomandazioni" di INTEL limitano il sistema solo a tolleranze non necessarie.

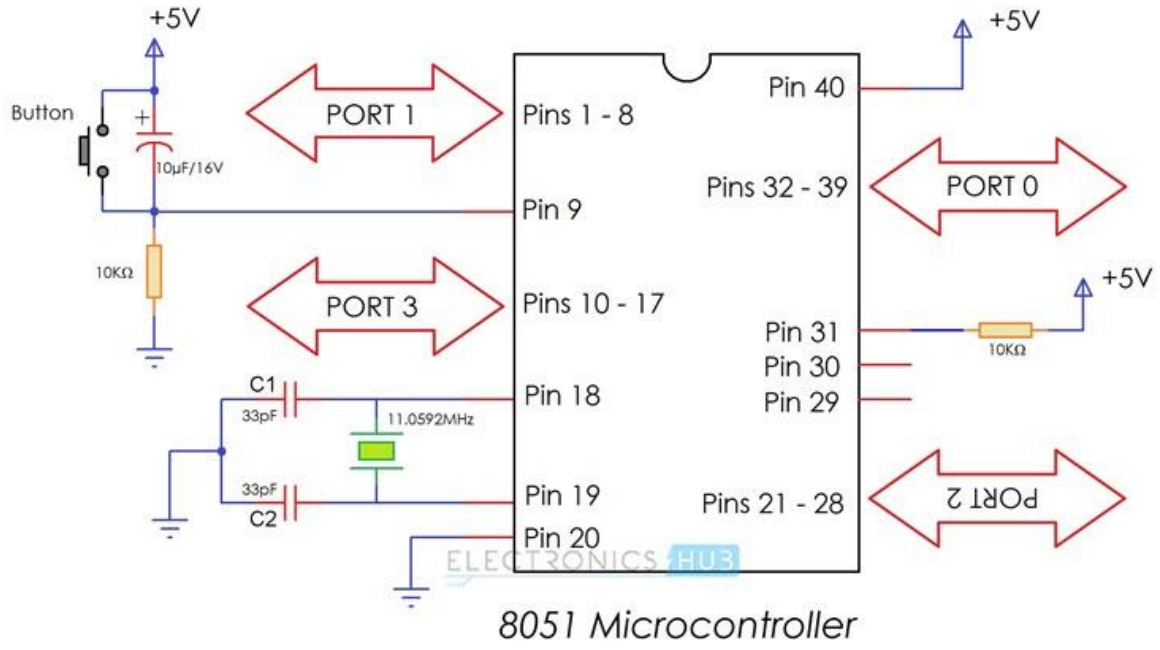


Fig.3.15

Ogni microcontrollore ha una certa frequenza di clock. In generale, un cristallo di quarzo viene utilizzato per realizzare il circuito di clock. La connessione è mostrata nella figura 3.16 e prende nota delle connessioni a XTAL 1 e XTAL 2. In alcuni casi, vengono utilizzate sorgenti di clock esterne e si possono vedere le varie connessioni sopra. I limiti della frequenza di clock (massimo e minimo) possono cambiare da dispositivo a dispositivo. La pratica standard è quella di utilizzare la frequenza di 12 MHz. Se sono coinvolte comunicazioni seriali, è meglio usare la frequenza di 11,0592 MHz.

Lo schema seguente mostra che il circuito del clock 8051.

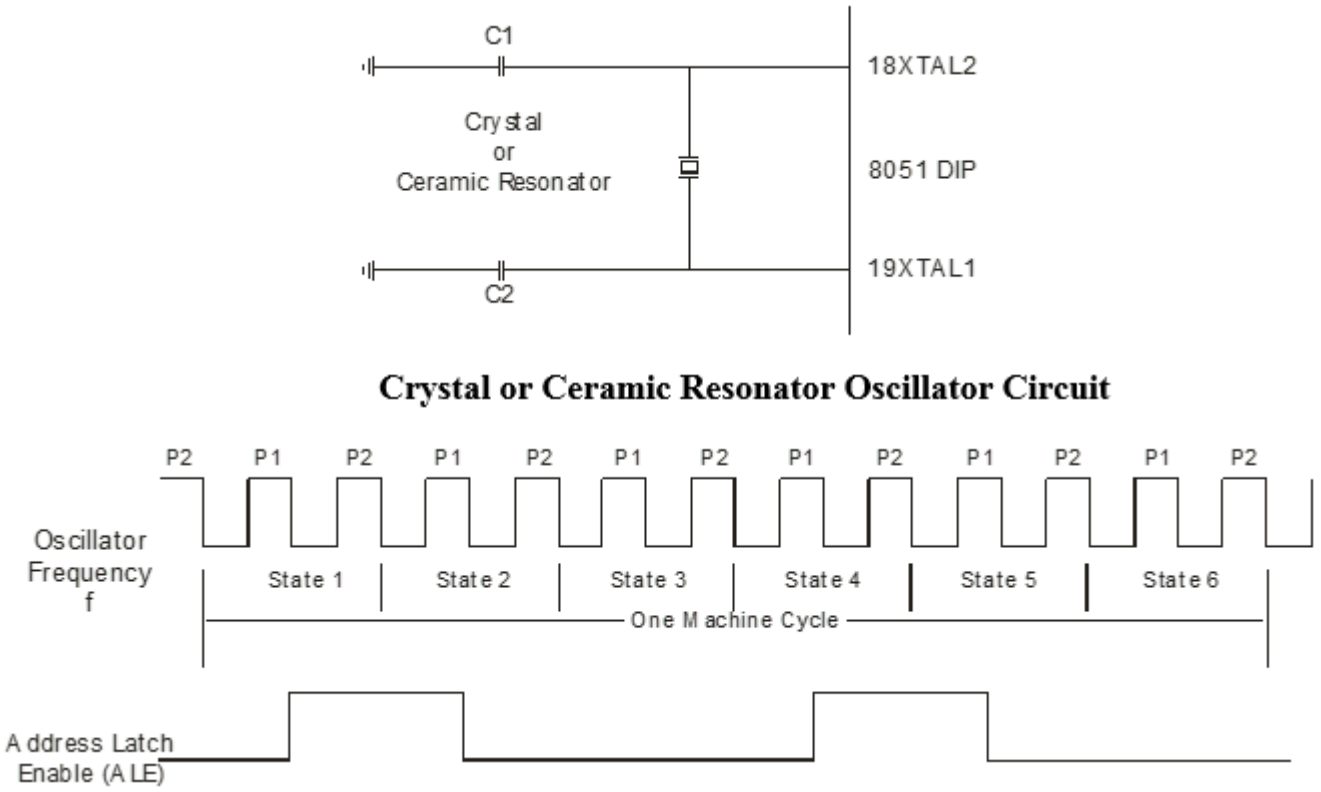


Fig.3.16

Sono disponibili modelli di 8051 che possono funzionare a frequenze massime e minime specificate, tipicamente da 1 MHz a 16 MHz. Le frequenze minime implicano che alcune memorie interne sono dinamiche e devono sempre funzionare al di sopra di una frequenza minima altrimenti i dati andranno persi.

La comunicazione seriale dei dati spesso determina la frequenza dell'oscillatore a causa del requisito che i contatori interni debbano dividere la frequenza di clock di base per produrre velocità di comunicazione standard in Bit al secondo (baud). Se la frequenza di clock di base non è divisibile, la frequenza di comunicazione risultante non essere standard.

L'oscillatore formato dal cristallo, dal condensatore e da un inverter su chip genera un treno di impulsi alla frequenza del cristallo come nella figura sopra.

- Stati:

Uno stato è l'intervallo di tempo di base per il funzionamento discreto dell'MCU come il recupero di un byte di codice operativo, la decodifica di un codice operativo o la scrittura di un byte di dati. Due impulsi dell'oscillatore definiscono ogni stato della ,macchina.

- Ciclo macchina:

L'intervallo di tempo minimo richiesto per eseguire qualsiasi istruzione semplice o parte di un'istruzione complessa. Un ciclo macchina stesso è composto da 6 stati do clock.

Le istruzioni di programma possono richiedere l'esecuzione di uno, due o quattro cicli macchina, a seconda del tipo di istruzione. Le istruzioni vengono recuperate ed eseguite automaticamente dall'MCU, a partire dalle istruzioni che si trovano all'indirizzo di memoria ROM 0000H al momento in cui il microcontrollore viene resettato per la prima volta o avviato a freddo. Se si

conosce il numero di cicli macchina necessari per eseguire una particolare istruzione, allora è possibile determinare il tempo necessario per eseguire tale istruzione. Il tempo per eseguire quell'istruzione si trova quindi moltiplicando C per 12 e dividendo il prodotto per la frequenza del cristallo. Tinst = (C * 12d) / frequenza del cristallo. Ad esempio, se la frequenza del cristallo è di 16 MHz, il tempo necessario per eseguire un ADD A, R1 è 1. Questa istruzione è di un'istruzione di ciclo, quindi C = 1.2. Tinst = (1 * 12d) / 16 MHz = 750 microsecondi. Un cristallo da 12 MHz fornisce il tempo conveniente di 1 microsecondo per ciclo. Un cristallo da 11,0592 MHz, sebbene apparentemente un valore dispari, produce una frequenza di ciclo di 921,6 KHz, che può essere divisa equamente per le velocità di trasmissione standard di 19200, 9600, 4800, 1200 e 300 Hz.
Nella figura sopra, ci sono due impulsi ALE per ciclo macchina. Questo impulso ALE è stato utilizzato principalmente come impulso di temporizzazione per l'accesso alla memoria esterna. Questo impulso indica quando ogni byte di istruzione viene recuperato. Dalla figura sopra, si arriverà a sapere che una singola istruzione di due byte può essere prelevata ed eseguita in un ciclo macchina. Ma non significa che le istruzioni a byte singolo vengano eseguite a metà ciclo.

3.7 SPECIAL FUNCTION REGISTERS

L'8051 è un microcontrollore flessibile con un discreto numero di modi operativi. Il programma può leggere e/o modificare il modo operativo dell'8051 cambiando i valori dei registri SFR.
Si accede ai registri SFR come se fossero una parte della normale RAM interna. L'unica differenza è che la
RAM interna è compresa fra gli indirizzi da 00h a 7Fh, mentre i registri SFR risiedono nell'area dall'indirizzo
80h all'indirizzo FFh.
Ogni registro SFR ha un proprio indirizzo ed un proprio nome. La tabella sottostante fornisce: una rappresentazione grafica dei registri SFR dell'8051, la denominazione e l'indirizzo.

TABELLA SFR (SPECIAL FUNCTION REGISTER)

00	P0	SP	DPL	DPH					PCON	07
08	TCON	TNOD	TLO	TL1	TH0	TH1				0F
89	P1									97
98	SCON	SUBF								9F
A0	P2									A7
A8	IE									AF
B0	P3									B7
B8	IP									B9
C0										C7
C8										CF
D0	PSW									D7
D8										DF
E0	ACC									E7
E9										EF
F0	B									F7
F9										FF

Fig.3.17

Come si può osservare, anche se il range degli indirizzi da 80h a FFh offre 128 possibili bytes, ci sono solo 21 registri SFR nell'8051 standard. Tutti gli altri indirizzi sono considerati non validi. Leggere o scrivere in questi registri può produrre un valore indefinito oppure un comportamento anomalo.
Si raccomanda di non leggere o scrivere nei registri SFR non assegnati Ciò può provocare un comportamento anomalo e può rendere il programma incompatibile
Con altri derivati dell'8051 che usano quei registri SFR per altri scopi.

- **Tipi di SFR**

Come risulta evidente dalla tabella, i registri SFR che hanno uno sfondo blu sono relativi a operazioni di I/O.
L'8051 dispone di quattro porte di I/O ad 8 bit per un totale di 32 linee di I/O. Per porre una linea di I/O e' allo stato alto o basso o leggere il suo valore vengono usati i registri SFR in rosso.

I registri SFR con lo sfondo in giallo servono a controllare il modo operativo o la configurazione dell'8051.
Per esempio il registro **TCON** controlla i timer, mentre il registro **SCON** controlla la porta seriale.
I restanti registri SFR a sfondo verde, sono registri ausiliari nel senso che essi non configurano direttamente il modo operativo del microprocessore, ma l'8051 non potrebbe operare senza di essi. Per esempio, una volta che la porta seriale è stata configurata usando il registro **SCON** il programma può leggere o scrivere sulla porta seriale usando il registro **SBUF**.
I registri SFR il cui nome appare scritto in azzurro nella fig. 2.6 sono quelli ai quali si può accedere con operazioni bit a bit (p.e. usando le istruzioni **SETB** e **CLR).** Gli altri registri SFR non possono essere usati nella modalità Bit a Bit. Come potete notare, tutti i registri SFR i cui indirizzi sono multipli interi di 8 possono essere usati con modalità Bit a Bit.

- **Descrizione dei registri SFR**

Questa sezione fornirà una breve panoramica dei registri SFR standard che si trovano nella mappa di fig. xxx nella mappa.

- P0 (Porta 0, Indirizzo 80h, Indirizzabile a bit): Questa è la porta P0 di input/output. Ciascun bit di questo registro SFR corrisponde ad un pin del microcontrollore. Per esempio, il Bit 0 è il pin P0.0, il Bit 7 è il pin P0.7. Settare un Bit di questo registro SFR equivale a forzare ad un livello alto il corrispondente pin di I/O, mentre resettarlo equivale a forzarlo ad un basso livello.

Da notare. Siccome l'8051 dispone di 4 porte di I/O (P0, P1, P2 e P3), se il hardware usa della RAM esterna oppure della ROM per il codice, le porte P0 a P2 non possono essere utilizzate. Questo è dovuto al fatto che in questa modalità di funzionamento, l'8051 usa queste porte per indirizzare la memoria esterna. In questo caso si possono usare solo le porte P1 e P3.

- SP (Stack Pointer, Indirizzo 81h): Questo è lo stack pointer del microcontrollore. Questo registro SFR punta al prossimo valore dello stack nella RAM interna. Se si carica un valore nello stack (operazione di push) tale valore sarà memorizzato all'indirizzo SP+1. Supponiamo che SP contenga il valore 07h, una istruzione di PUSH caricherà il valore nello stack all'indirizzo 08h. Questo registro SFR viene modificato da tutte le istruzioni che comportano operazioni sullo stack quali: PUSH, POP, LCALL, RET, RETI, e ogni volta che il MCU esegue un'operazione di interrupt.
Da notare. Lo stack pointer allo start up, è inizializzato a 07h. Questo significa che lo stack parte da 08h e si espande in alto nella RAM interna. Poiché i banchi di registri 1, 2 e 3 come pure le variabili a Bit occupano la RAM interna da 08h a 2Fh, è necessario inizializzare diversamente lo stack pointer del programma se si intende usare i banchi di registri e/o la memoria a Bit. Non è una cattiva idea inizializzare lo stack pointer ad un valore pari a 2Fh nella prima istruzione di ogni programma a meno che non si è sicuri al 100% di non dover usare i registri a banchi e le variabili a Bit.

- DPL/DPH (Data Pointer Basso/Alto, Indirizzi 82h/83h):
I registri DPL e DPH lavorano insieme per formare un valore a 16-bit chiamato *Data Pointer*. Il Data Pointer è usato nelle operazioni che riguardano la memoria,
RAM esterna ed alcune istruzioni che riguardano il codice di programma. Essendo un valore intero

unsigned, esso può rappresentare valori da 0000h a FFFFh (da 0 a 65535 decimale).
DPTR è in realtà un valore a 16-Bit ottenuto raggruppando insieme i registri DPH e DPL. In realtà quando si ha che fare con il registro DPTR quasi sempre si usa un byte alla volta. Per esempio, per mettere nello stack il registro DPTR (operazione di push) si deve prima effettuare il push del registro DPL e quindi quello del registro DPH. Non si può semplicemente effettuare il push del registro DPTR. Eccezionalmente esiste un'istruzione per "incrementare il DPTR". Quando si riesegue questa istruzione, i due byte operano come se fossero un solo registro a 16-bit. Comunque non esiste un'istruzione per decrementare DPTR. Se si vuole decrementare DPTR si deve scrivere un sezione di codice apposta.

- PCON (Power Control, Indirizzo 87h): Il registro PCON (Power Control) è usato per controllare il power
mode dell'8051. In alcune modalità di funzionamento è permesso all'8051 di mettersi nello stato "sleep" che richiede molta minor potenza. Questa modalità di funzionamento è controllata dal registro PCON. In più, uno dei Bit di PCON è usato per raddoppiare il baud rate effettivo della porta seriale dell'8051.

- TCON (Timer Control, Indirizzo 88h, Indirizzabile a Bit): Il registro TCON (Timer Control) è usato per
configurare e modificare il modo di funzionamento dei timer dell'8051. Questo registro abilita o disabilita il funzionamento dei due timer e contiene un flag per indicare quando un timer va in overflow. Alcuni Bit del registro TCON non sono relativi all'uso del timer ma servono a configurare gli interrupt esterni e contengono dei flag che segnalano che un interrupt esterno è stato attivato.

- TMOD (Timer Mode, Indirizzo 89h): Il registro TMOD (Timer Mode) è usato per configurare il modo di funzionamento di ciascuno dei due timer. Mediante tale registro e' possibile far funzionare un timer come un timer a 16-bit, oppure un timer a 8-bit con auto-caricamento, oppure come un timer a 13 bit o come dure timer separati. Inoltre e' possibile configurare il timer come contatore quando un pin esterno e' attivo oppure come "contatore di eventi" segnalati da un apposito pin esterno.
TL0/TH0 (Timer 0 Basso/Alto, Indirizzi 8Ah/8Bh): Questi due registri uniti insieme danno lo stato del timer 0. Il loro comportamento è determinato da come è stato configurato il timer 0 nel registro TMOD; in ogni caso essi contano sempre in modo ascendente. Quello che è possibile configurare è la maniera e il momento nel quale essi devono incrementare il valore.
TL1/TH1 (Timer 1 Basso/Alto, Indirizzi 8Ch/8Dh): Questi due registri uniti insieme danno lo stato del timer
1. Il loro comportamento e' determinato da come è stato configurato il timer 1 nel registro TMOD; in ogni caso essi contano sempre in avanti. E' possibile configurare sia la maniera che il momento nel quale il timer deve incrementare il suo valore.

P1 (Porta 1, Indirizzo 90h, Indirizzabile a bit): Questa è la porta P1 di input/output. Ciascun Bit di questo registro SFR corrisponde ad un pin dell'MCU. Per esempio, il Bit 0 è il pin P1.0, il Bit 7 è il pin
P1.7. Settare un bit di questo registro SFR equivale a forzare a livello alto il corrispondente pin di I/O, mentre resettarlo equivale a forzarlo a livello basso.**SCON (Serial Control, Indirizzo 98h, Indirizzabile a bit):** Il registro SCON(Serial Control) e' usato per configurare il comportamento

della porta seriale dell'8051. Questo registro controlla: il baud rate della porta seriale, l'abilitazione della ricezione e contiene dei flag per indicare quando un byte è stato trasmesso o ricevuto con successo.
Suggerimento per il programmatore: Per usare la porta seriale dell'8051 è generalmente necessario inizializzare i seguenti registri SFR: SCON, TCON e TMOD. Ciò è dovuto al fatto che il registro SCON è usato solo per controllare la porta seriale. Nella maggior parte dei casi il programma prevede di usare uno
dei timer come generatore di baud rate. In questo caso va configurato il timer corrispondente mediante i registri TCON e TMOD.
SBUF (Serial Buffer, Indirizzo 99h): Il registro SBUF (Serial Buffer) è usato per inviare e ricevere i dati dalla porta seriale dell'8051. Ogni valore scritto nel registro SBUF è inviato sulla porta seriale dal pin TXD. Allo stesso modo, ogni valore ricevuto dalla seriale attraverso il pin RXD è reso disponibile al programma applicativo dell'utente mediante una lettura del registro SBUF. In altre parole il registro SBUF serve da porta di uscita quando viene scritto e da porta d'ingresso quando viene letto.
P2 (Porta 2, Indirizzo A0h, Indirizzabile a bit): Questa e' la porta P2 di input/output. Ciascun bit di questo registro SFR corrisponde ad un pin del microcontrollore. Per esempio, il bit 0 e' il pin P2.0, il bit 7 e' il pinP2.7. Settare un bit di questo registro SFR equivale a forzare ad un livello alto il corrispondente pin di I/O, mentre resettarlo equivale a forzarlo ad un basso livello.

IE (Interrupt Enable, Indirizzo A8h): Il registro IE (Inetrrupt Enable) è usato per abilitare e disabilitare gli interrupt. I 7 bit meno significativi di IE sono usati per abilitare o disabilitare gli interrupt individualmente,mentre il bit più significativo e' usato per abilitare o disabilitare tutti gli interrupt contemporaneamente. Per cui, se tale bit e' nello stato 0 tutti gli interrupt sono disabilitati anche se il corrispondente bit e' attivo.

P3 (Porta 3, Indirizzo B0h, Indirizzabile a bit): Questa è la porta P3 di input/output. Ciascun bit di questo registro SFR corrisponde ad un pin del microcontrollore. Per esempio, il bit 0 e' il pin P3.0, il bit 7 è il pin P3.7. Settare un bit di questo registro SFR equivale a forzare ad un livello alto il corrispondente pin di I/O,mentre resettarlo equivale a forzarlo ad un basso livello.

IP (Interrupt Priority, Indirizzo B8h, Indirizzabile a bit): Il registro IP (Interrupt Priority) è usato per cambiare la priorità di ciascun interrupt. Sull'8051 un interrupt può essere a bassa priorità (0) oppure ad alta priorità (1). Un interrupt può interrompere solo un altro interrupt a priorità più bassa. Per esempio, se configurate l'8051 in modo tale che tutti gli interrupt siano a bassa priorità eccetto quello relativo alla porta seriale, esso può sempre sospendere il sistema anche se c'è un altro interrupt attivo. Nel momento in cui è attivo l'interrupt della porta seriale, però, nessun altro interrupt è in grado di sospendere la routine in corso poichè essa ha priorità più elevata.

PSW (Program Status Word, Indirizzo D0h, Indirizzabile a bit): Il registro PSW (Program Status Word) è usato per memorizzare alcuni Bit importanti che vengono aggiornati nell'esecuzione delle istruzioni dell'8051. Il registro PSW contiene: il carry flag, il carry flag ausiliario, il flag di overflow ed il flag di parità. Il registro PSW contiene anche i flag utilizzati per selezionare il banco di registri "R" attivo.

Suggerimento: Se si scrive una routine di interrupt (interrupt handler), è buona norma *salvare sempre* lo stato del registro PSW nello stack all'inizio della routine e di ripristinarlo appena prima di ritornare al programma principale. Molte istruzioni modificano lo stato dei Bit del registro PSW, per cui se la vostra routine di interrupt non preserva lo stato del registro PSW, il programma avrà dei comportamenti strani e imprevedibili e sarà difficile trovare il problema a causa del comportamento non deterministico del microcontrollore.

ACC (Accumulatore, Indirizzo E0h, Indirizzabile a bit): L'accumulatore e' il registro dell'8051 più usato poiché esso viene coinvolto in quasi tutte le istruzioni. Esso risiede all'indirizzo E0h, ciò significa che l'istruzione **MOV A,#20h** in realtà corrisponde all'istruzione **MOV E0h,#20h.** E' comunque meglio usare la prima istruzione poiché richiede un solo byte mentre la seconda ne richiede almeno due.

B (Regitro B, Indirizzo F0h, Indirizzabile a Bit): Il registro B è utilizzato in due istruzioni: la moltiplicazione e la divisione. Esso è anche comunemente usato dai programmatori come registro ausiliario per memorizzare dei valori temporanei.

Altri registri SFR

La tabella precedente e' un sommario di tutti i registri SFR di un 8051 standard. Tutti i microcontrollori derivati dall'8051 devono avere questi registri come base per mantenere la compatibilità indietro con lo standard MSC51.

E' pratica comune, delle fabbriche di semiconduttori che vogliano sviluppare un nuova versione di 8051, aggiungere altri registri SFR per supportare delle nuove funzioni nel chip derivato.

Per esempio, il microcontrollore Dallas Semiconductor DS80C320 e' compatibile verso l'alto con l'8051. Ciò significa che un qualsiasi programma sviluppato per un 8051 standard dovrebbe girare senza modifiche sul DS80C320. In questo caso tutti i registri SFR che abbiamo finora menzionato devono esistere anche nel componente della Dallas.

Il DS80C320, però, fornisce molte nuove funzionalità in più rispetto all'8051 standard che devono in qualche mode essere gestite. Ciò viene realizzato aggiungendo dei nuovi registri SFR a quelli già menzionati. Per esempio, poiché il DS80C320 (Dallas Semiconductors) dispone di due porte seriali (invece di una soltanto come nell'8051) sono stati aggiunti due nuovi registri SFR denominati rispettivamente SBUF2 e SCON2. In aggiunta ai normali registri, il DS80C320 riconosce come validi anche questi due nuovi registri SFR e usa il loro valore per determinare il modo operativo della seconda porta seriale. Ovviamente, a questi due nuovi registri devono essere assegnati degli indirizzi non utilizzati nell'8051 originale. In questa maniera, un nuovo chip derivato dall'8051 può essere sviluppato con la capacità di poter interpretare i programmi dell'8051 esistenti senza modificare il codice.

Qualora state scrivendo un programma che utilizza dei nuovi registri SFR che sono specifici di un chip derivato e non sono contemplati nell'elenco dei registri che abbiamo analizzato, ricordate che esso non girerà su un 8051 standard. Perciò usate i registri SFR non standard solo se ritenete di dover usare il vostro programma su quel microcontrollore specifico. Nella stessa maniera, qualora forniate tale programma ad una terza parte, avvisateli che il vostro codice usa dei registri SFR non standard, per evitargli dei possibili rompicapo.

CAPITOLO 4
Interfacciamenti con dispositivi esterni

4.1 CONVERITORE ANALOGICO-DIGITALE
In un convertitore analogico/digitale, il problema principale consiste nello stabilire la corrispondenza tra la grandezza analogica di ingresso da convertire (che ha andamento continuo nel tempo) e la grandezza digitale d'uscita grandezza convertita (che, in quanto numerica, ha andamento discreto, per gradini) La corrispondenza deve essere univoca e il grado di dettaglio raggiunto nella discretizzazione del segnale da esaminare determina l'incertezza della conversione.

-Campionamento e Quantizzazione
La digitalizzazione di un segnale analogico coinvolge due processi di discretizzazione: un processo di discretizzazione nel dominio del tempo (campionamento) e un processo di discretizzazione in ampiezza (quantizzazione)
Una delle fasi più critiche della trasformazione di un segnale analogico in uno digitale è quella del campionamento che consiste nel convertire una grandezza variabile nel tempo in modo continuo in una discreta che rappresenti la prima in modo univoco e con la introduzione di errori trascurabili Un segnale analogico che varia in funzione del tempo viene moltiplicato per un segnale di tipo impulsivo di ampiezza costante (ad esempio, unitaria) che si ripete con frequenza preordinata e pure costante, detta frequenza di campionamento Il periodo TS corrispondente alla citata frequenza è detto intervallo di campionamento
Il risultato che si ottiene dal prodotto è rappresentato ovviamente da una serie di impulsi modulati in ampiezza che rappresentano in una certa misura il segnale analogico assegnato

Gli ADC o convertitori analogico-digitale, convertono i valori di tensione in ingresso nel numero corrispondente espresso in binario. La Risoluzione (R) di un convertitore A/D è definita come la minima variazione della grandezza analogica in ingresso che provoca una variazione di un LSB (Least Significant Bit= bit meno significativo) nel numero di uscita: tale variazione è definita come quanto (Q). La risoluzione R di un ADC coincide, dunque, col quanto Q.

$$Q = \frac{V_{FS}}{2^n}$$

con V_{FS}=Tensione di fondo scala; essa è una tensione di riferimento fornita al convertitore: essa individua il massimo valore in ingresso convertibile in binario. In figura è rappresentato il comportamento di un ADC a 3 bit; in uscita, dovranno essere rappresentati $N=2^n=2^3=8$ numeri compreso lo 0.

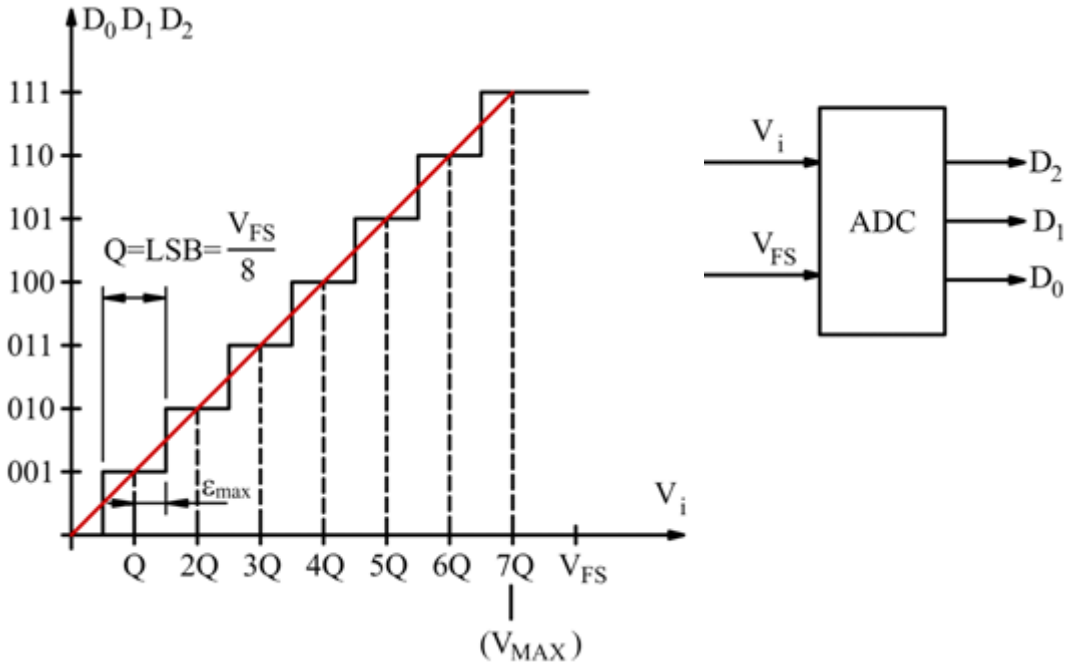


Fig.4.1

Si nota come un valore di tensione in ingresso V_i=Q venga convertito in uscita col valore $(001)_2$, così come tutti i valori compresi fra Q/2e 3Q/2. Viene così introdotto **l'errore di quantizzazione ε_{max}.**

$$\varepsilon_{max} = \pm\frac{Q}{2}$$

Il massimo valore di tensione in ingresso che può essere convertito in binario con errore ε=0:

$$V_{max} = (2^n - 1) \cdot Q = V_{FS} - Q$$

Convertitore ADC Flash

Chiamato anche convertitore con priorità, ha il compito di individuare la soglia di valore di tensione maggiore, tra quelle superate dalla tensione di ingresso per poi generare la corrispondente codifica binaria.

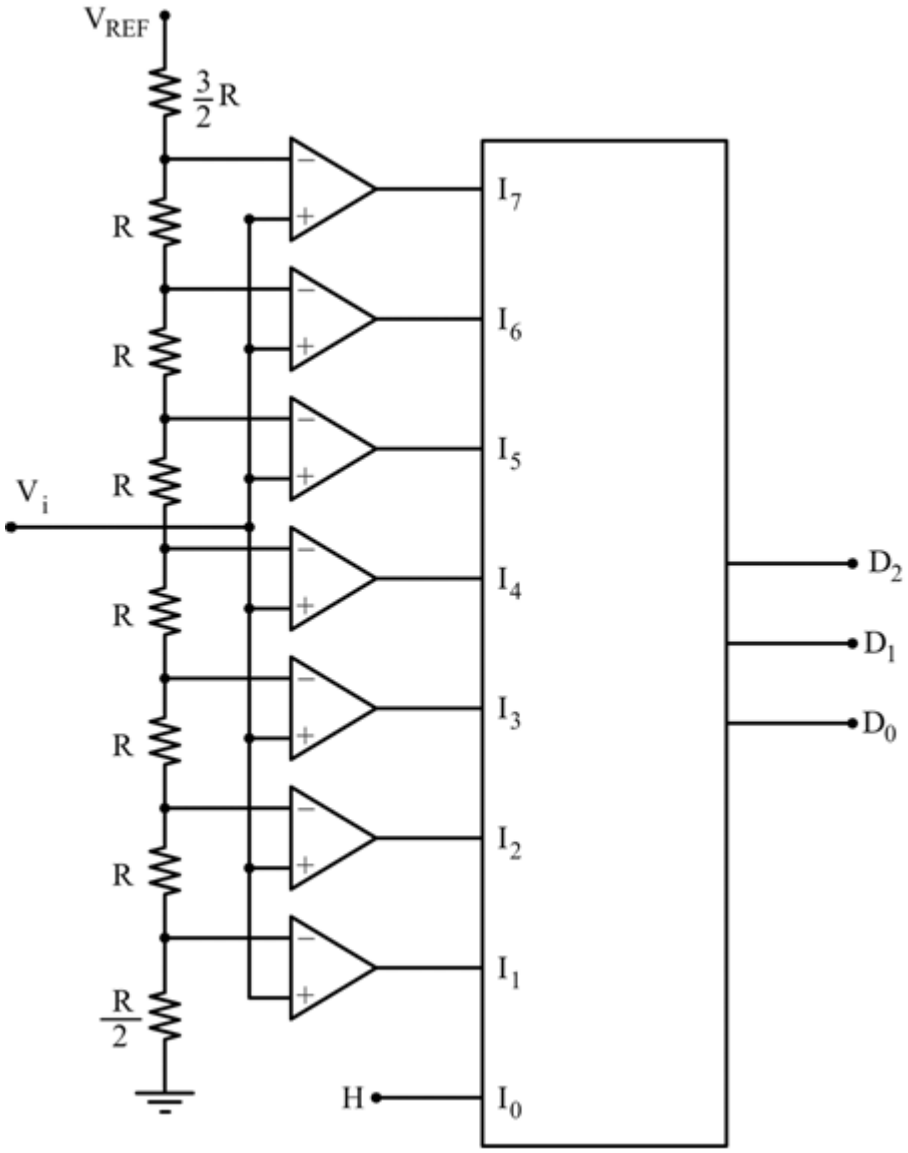

V_{REF}
$\frac{3}{2}R$
R
R
R
V_i
R
R
R
$\frac{R}{2}$
H
I_7
I_6
I_5
I_4
I_3
I_2
I_1
I_0
D_2
D_1
D_0

Fig.4.2

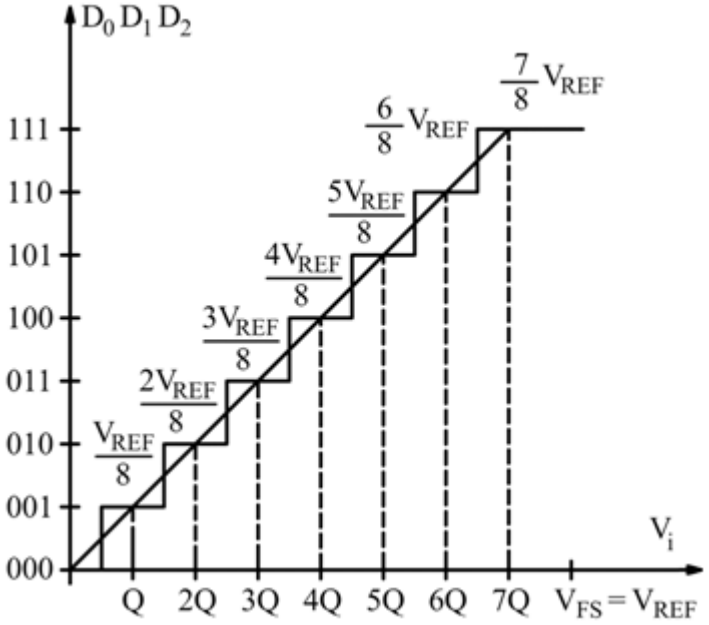


Fig.4.3

Qui è invece riportato un convertitore (minimale) a 2 bit. Ad ogni resistenza che si trova al morsetto non invertente di un operazionale si ha un salto di tensione pari al valore del quanto Q. Il circuito combinatorio usato per la codifica è molto semplice. In questo caso:

$$Q = \frac{V_{FS}}{2^n} = \frac{12}{2^2} = 3V = V_{i\,min}$$

- Questa è la variazione minima della tensione di ingresso per permettere una variazione del bit meno significativo (LSB) in uscita.

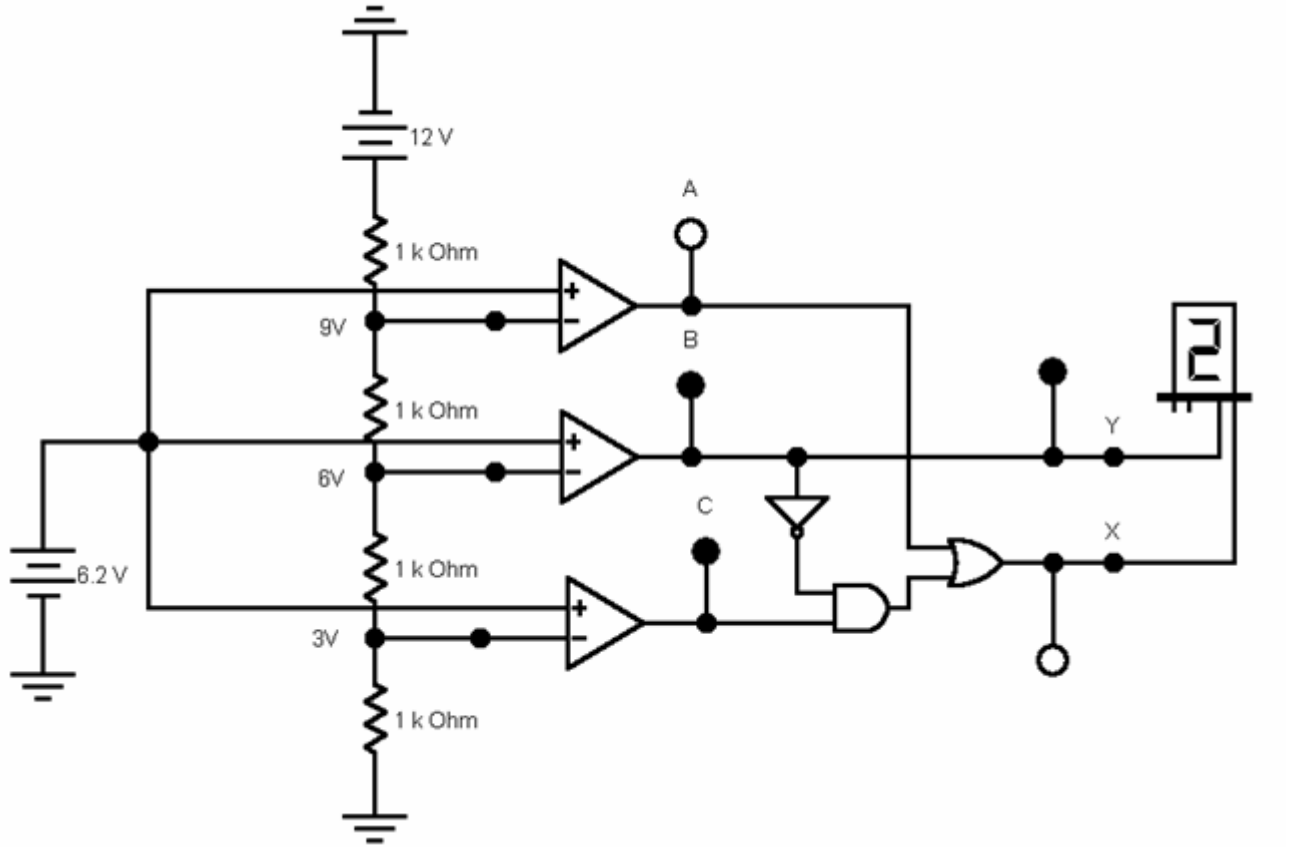


Fig.4.4

Ad esempio quando la tensione di ingresso è di 9,5V avremo:

$$NV_{FS} = V_i 2^n \quad \rightarrow \quad N = \frac{V_i 2^2}{V_{FS}} = \frac{9{,}5 \cdot 4}{12} = 3{,}1\bar{6} = (3)_{10} = (11)_2$$

infatti la massima tensione che può essere convertita in binario è:

$$V_{i\,max} = V_{FS} - Q = 12 - 3 = 9\,V$$

Il circuito restituisce 0 in uscita se V_i<3V restituisce 1 se 3V < V_i < 6V, restituisce 2 se 6V < V_i <9V restituisce 3 se V_i>9V. Non ha un grande precisione, ma la tecnica usata è la stessa anche per convertitori di maggior precisione.

Il campionamento e la conversione dei segnali

In figura è rappresentato un sistema di acquisizione e ricostruzione per un singolo segnale analogico, i blocchi tratteggiati non sono sempre necessari .

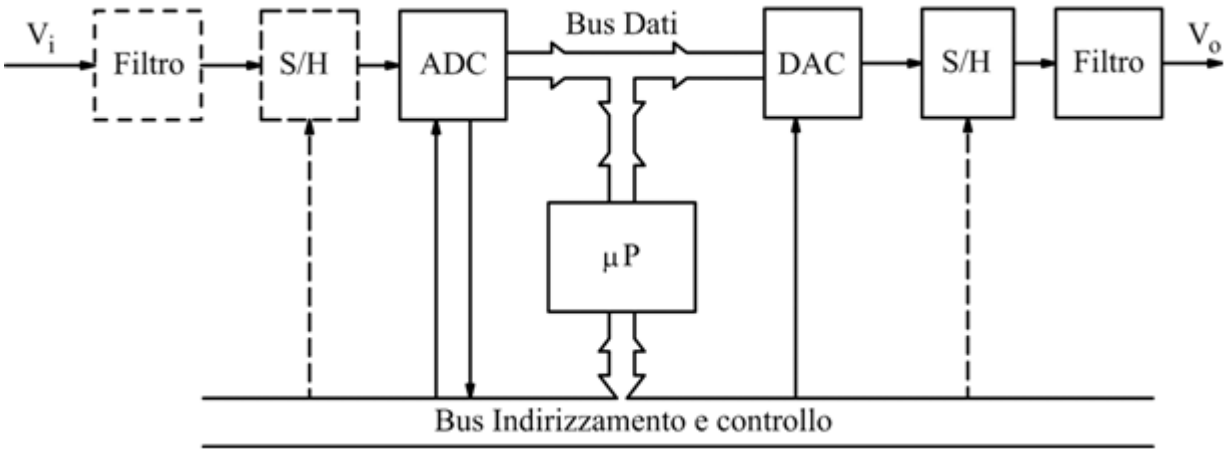


Fig.4.5

Per fare in modo che l'informazione acquisita durante il processo non venga deteriorata dalla conversione in digitale occorre risolvere qualche problema:

Se il segnale analogico all'ingresso dell'ADC varia molto velocemente durante la conversione, il valore della tensione a cui è associato il risultato potrebbe non essere individuabile con precisione ; a seconda della velocità di variazione del segnale e della risoluzione del convertitore può essere necessario inserire un circuito Sample and Hold (S/H) a monte dell'ADC.

Il segnale all'uscita del DAC è a gradini; per "arrotondare" la forma d'onda e renderla uguale all'originale è generalmente necessario porre dopo il DAC un FILTRO PASSA-BASSO.

- Dato che la conversione richiede un certo tempo (T_c), non è possibile convertire in digitale i valori assunti in ogni istante dal segnale analogico d'ingresso, ma solo quelli in corrispondenza di una successione discreta di istanti di tempo. Il problema sta nell'individuare il massimo intervallo di tempo tra una conversione e quella successiva , che garantisce ancora la ricostruzione del segnale originale; la risposta a quest'ultimo quesito è data dal:

- Teorema di Shannon

Se f_s è la frequenza di campionamento del modulo S/H ed f_{max} è la frequenza massima del segnale da campionare, il segnale in questione può essere ricostruito se è soddisfatta la condizione:

$$f_s \geq 2 f_{max}$$

- Sample and hold (S/H)

Sample and hold significa "campiona" e "mantieni"; lo scopo di questo circuito è quello di acquisire la tensione di ingresso in un determinato istante (campionamento) e di mantenerlo invariato all'uscita fino ad una nuova lettura.

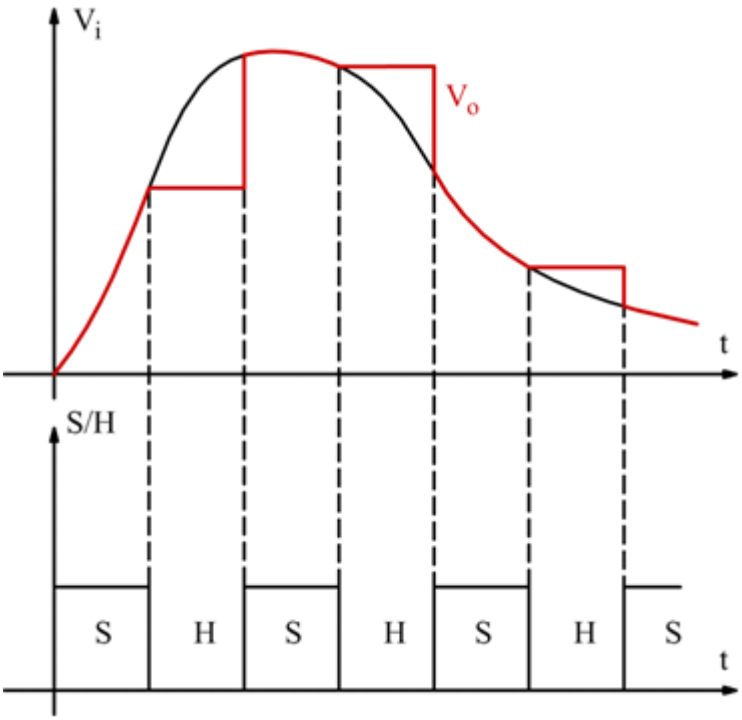


Fig.4.6

Il circuito che implementa tale funzione è il seguente:

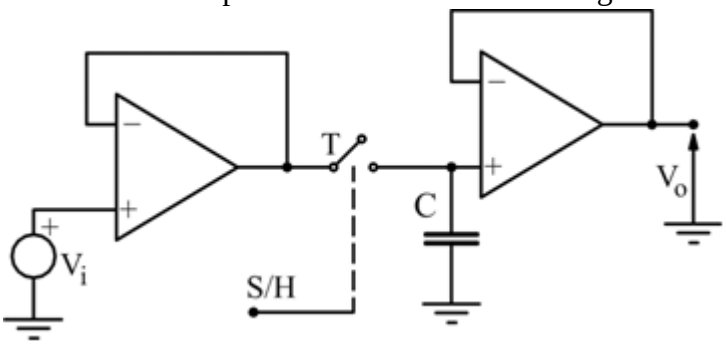
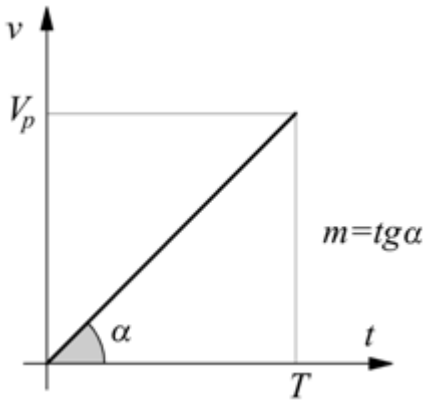


Fig.4.6

Nella catena di conversione analogico-digitale il circuito S/H precede il convertitore. Il circuito S/H non è sempre indispensabile, la sua presenza dipende dalla velocità di variazione del segnale da convertire. La regola che deve essere rispettata è:

$$\left(\frac{dv}{dt}\right)_{max} \leq \frac{V_{FS}}{2^n T_c} = \frac{Q}{T_c}$$

La massima velocità di variazione del segnale deve essere inferiore o uguale al rapporto fra il quanto e il tempo di conversione T_c; n è il numero di bit del convertitore.
Il tempo di conversione è l'intervallo di tempo che passa fra l'istante in cui in ingresso si presenta un valore di tensione stabile e quello in cui sulle uscite appare il corrispondente valore binario.



Nel caso di un segnale a rampa è piuttosto semplice, osservando che

$$\left(\frac{dv}{dt}\right) = \left(\frac{dv}{dt}\right)_{max} = \frac{V_p}{T} = m$$

nel caso di un segnale sinusoidale del tipo $v = V_p \sin(2\pi \cdot f \cdot t)$ si ha:

$$\frac{dv}{dt} = 2\pi \cdot V_p \cos(2\pi \cdot f \cdot t)$$

il massimo (la massima variazione) è::

$$\left(\frac{dv}{dt}\right)_{max} = 2\pi \cdot f \cdot V_p$$

$$\left(\frac{dv}{dt}\right)_{max} \le \frac{V_{FS}}{2^n T_c} = \frac{Q}{T_c} \quad \rightarrow \quad 2\pi\, f\, V_p \le \frac{V_{FS}}{2^n T_c} \quad \rightarrow \quad \omega\, V_p \le \frac{V_{FS}}{2^n T_c}$$

Quindi per i segnali sinusoidali se vale la condizione:

$$\omega V_p \le \frac{V_{FS}}{2^n T_c}$$

si può evitare di usare un modulo S/H.

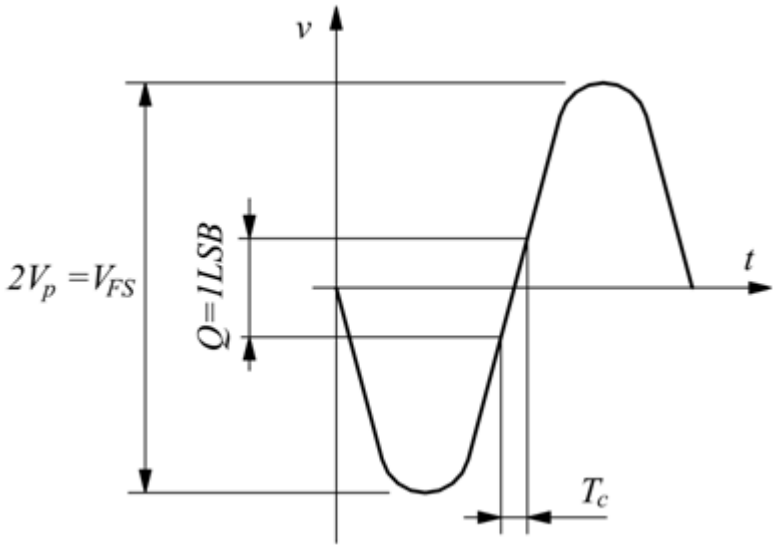


Fig.4.7

Se si ipotizza che il valore picco-picco della tensione sinusoidale da convertire abbia il massimo valore convertibile, cioè $2V_p=V_{FS}$ si avrà:

$$\omega \cdot V_p = 2\pi \cdot f \cdot V_p = \pi \cdot f \cdot V_{FS} \leq \frac{V_{FS}}{2^n T_c}$$

si avrà:

$$f \leq \frac{1}{2^n \pi T_c}$$

Si può evitare l'uso di un modulo S/H per l'acquisizione dei segnali sinusoidali solo se vengono rispettate queste condizioni.
Esiste l'eventualità che si debbano campionare più segnali simultaneamente con lo stesso convertitore, si applica, usualmente, una tecnica di time-sharing che utilizza un multiplexer analogico.

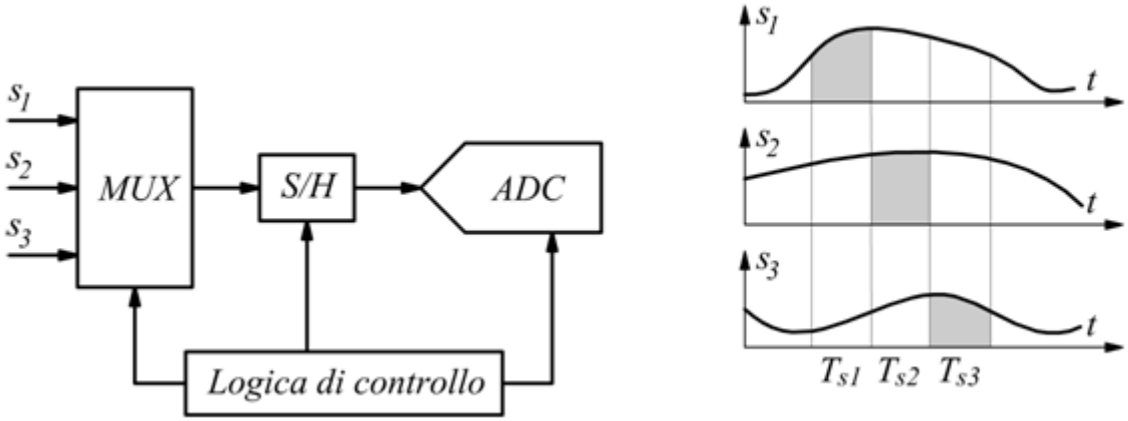


Fig.4.8

ad intervalli di tempo regolari i segnali analogici in ingresso vengono smistati verso il sistema di acquisizione e campionati . Se gli intervalli di campionamento sono gli stessi:

$$T_{s1} = T_{s2} = \ldots = T_{sN} = T_s$$

deve essere soddisfatta la relazione:

$$T_s = NT_c$$ con

T_s=tempo di campionamento
T_c=tempo di conversione
N=numero di canali

Anche se esistono cinque tipi principali di ADCs, nel dominio dei moderni sistemi DAQ, si trovano in pratica solo due:
- Approssimazioni successive e sigma-delta.
Gli altri tipi sono comunque validi ma più orientati ad applicazioni non-DAQ. Per esempio"**dual-slope ADCs**"sono abbastanza lenti ed utilizzati nella maggior parte dei Multimetri.

Poi ci sono i"**flash ADCs**"che offrono una frequenza di campionamento altissima ma la risoluzione è troppo bassa per le tipiche applicazioni DAQ. I Pipeline converter ADCs combinano più convertitori flash per migliorare la risoluzione, ma il loro impiego è limitato.

In definitiva esiste un convertitore A/D per qualsiasi applicazione, considerando chiaramente le caratteristiche elettriche e dinamiche della conversione.

4.2 MODULAZIONE PWM

In questo paragrafo verranno introdotti il concetto di modulazione della larghezza d'impulso (PWM) della tensione d'uscita di un inverter e verranno illustrate le principali tecniche di PWM quali la sotto-oscillazione e il metodo degli angoli memorizzati. La tecnica della modulazione della larghezza d'impulso, fornisce una soluzione ai principali problemi che si incontrano col funzionamento in onda quadra quali la presenza di armoniche di elevata ampiezza a bassa frequenza nella forma d'onda della tensione d'uscita e conseguenti correnti assai distorte e relativi effetti collaterali indesiderati che possono essere di elevata portata. Questa tecnica consiste nel frazionare l'onda quadra con impulsi di larghezza variabile cercando di approssimare un andamento sinusoidale col valor medio variabile degli impulsi. Da cui il nome "Pulse Width Modulation" o 'PWM'. Diversamente da quanto accade nell'onda quadra, modulando opportunamente gli impulsi, si ha che le armoniche di ampiezza maggiore sono a frequenza molto più elevata della fondamentale. Questa situazione è particolarmente favorevole in quanto, nel caso di carico con componente induttiva (ad esempio motori), una tensione PWM produce una corrente pressoché sinusoidale data l'energica azione di filtro passa-basso del circuito RL. La presenza di buchi nella forma d'onda della tensione di uscita consente inoltre di variare l'ampiezza della componente fondamentale senza dover variare la tensione continua che alimenta l'inverter. Da ciò si deduce che l'alimentazione dell'inverter può essere realizzata semplicemente con un ponte a diodi senza la necessità di ricorrere a un ponte controllato con vantaggio economico spesso rilevante. In definitiva con questa tecnica si hanno i seguenti molteplici vantaggi:

1. Una semplificazione della sezione di potenza dell'azionamento.
2. Una sensibile riduzione della potenza reattiva e distorcente in rete rispetto al caso di alimentazione con raddrizzatore controllato a tiristori.
3. Un minor ritardo nella risposta della sezione di potenza ad un comando di tensione, con benefici sulle prestazioni dinamiche dell'azionamento (il ritardo è valutabile a metà del periodo medio degli impulsi).

Gli svantaggi rispetto al funzionamento in onda quadra sono:

1. L'elevato numero di commutazioni nel periodo e il conseguente aumento delle perdite di commutazione.
2. Utilizzo di dispositivi a semiconduttore a commutazione forzata.
3. Il circuito di comando risulta inevitabilmente più complesso.

Il numero delle commutazioni deve ovviamente essere il più alto possibile compatibilmente coi limiti imposti dai circuiti di commutazione dei dispositivi di commutazione, dalle perdite ammissibili e per riflesso dal rendimento ammesso dell'azionamento. Per la scelta della posizione delle commutazioni esistono numerose tecniche di cui le seguenti due sono le fondamentali:

1. 'Metodo della sotto oscillazione': le commutazioni avvengono in corrispondenza delle intersezioni di due segnali a frequenza diversa; è una tecnica essenzialmente analogica, ma che ha trovato facile implementazione nelle periferiche dei microntrollori.

2. ‘Metodo degli angoli di commutazione memorizzati’: gli istanti di commutazione sono calcolati mediante procedimenti analitici intesi a conseguire determinate prestazioni dal sistema inverter-motore (ad esempio la minima distorsione di corrente); è una tecnica essenzialmente digitale, anche se non di immediata implementazione sui microncontrollori.

CAPITOLO 5
Modi operativi del Microcontrollore

5.1 IDLE

In modalità Idle, la CPU si mette in stop mentre tutte
le periferiche su chip rimangono attive. La modalità è
richiamata dal software. Il contenuto del chip
come la RAM e tutti i registri delle funzioni speciali rimangono invariati durante questa modalità. La modalità inattiva può essere terminata da qualsiasi interrupt abilitato o da un hardware di ripristino.
Va notato che quando Idle viene terminato da un file
ripristino hardware, il dispositivo riprende normalmente il programma di esecuzione, da dove si era interrotta, fino a due cicli macchina prima che l'algoritmo di ripristino interno riprende il controllo.
L'MCU NELLO STATO DI idle impedisce l'accesso alla RAM interna in questo caso, ma ha accesso ai pin di interrupt della porta non è inibito. Per eliminare la possibilità di un file scrittura imprevista su un pin della porta quando Idle viene terminato da reset, l'istruzione successiva a quella che l'MCU richiama di nuovo l'Idle cosi non dovrebbe essere scritto niente sul pin di una porta o un accesso alla memoria esterna.

5.2 POWER DOWN MODE

Nella modalità Power Down l'oscillatore è fermo,
e l'istruzione che richiama Power Down è
l'ultima istruzione eseguita. La RAM su chip e
i registri di funzioni speciali mantengono i propri valori fino a che la modalità di spegnimento viene terminata.
Il Reset in questo caso ridefinisce gli SFR ma non modifica il file della RAM su chip. Il ripristino non deve essere attivato prima che VCC viene ripristinato al suo normale livello operativo e deve essere tenuto attivo abbastanza a lungo da consentire l'oscillatore per poter riavviare e stabilizzare il normale funzionamento.

CAPITOLO 6
Linguaggio di programmazione ASSEMBLER

6.1 TIPI DI INDIRIZZAMENTO

Esistono diversi di indirizzare una locazione di memoria, o un registro, o una periferica, ci sono modi che indirizzano direttamente utilizzando 1 ciclo macchina, mentre ci modi come quello di puntare alla memoria esterna che richiede 4 cicli macchina, di seguito sono descritti i vari tipi di indirizzamento.

- **Indirizzamento diretto**:
  I'operando è specificato nell'istruzione da un campo indirizzo ad 8 bit. Solo i dati della RAM interna e della SFR area possono essere indirizzati direttamente.

  Es.:

  ADD A,7FH

  il contenuto dell'accumulatore viene sommato al contenuto della locazione 7FH della RAM interna e il risultato posto ancora nell'accumulatore.

- **Indirizzamento indiretto**:
  nell'istruzione viene specificato un registro che contiene I'indirizzo dell'operando. Sia la RAM interna sia quella esterna possono essere indirizzate indirettamente. I registri d'indirizzo per indirizzi ad 8 Bit possono essere RO,R1,R2,R3 del banco di registri selezionato, oppure lo stack pointer SP. I1 registro d'indirizzo per gli indirizzi a 16 bit può essere solo il registro DPTR (Data PoinTeR).

  Es.:

  ADD A,@RO

  Il contenuto dell'accumulatore viene sommato al dato che si trova nella RAM interna all'indirizzo specificato dal registro RO. II risultato viene ancora posto nell'accumulatore.

- **Indirizzamento a registro**:
  alcune istruzioni possono accedere ai banchi di registri che contengono i registri RO - R7. Questo è possibile grazie a 3 bit che selezionano uno degli 8 registri (RO - R7) interni al codice operativo. Questo metodo di indirizzamento è efficiente dal punto di vista del codice macchina poiché elimina il byte del campo indirizzo. I1 banco utilizzato dipende dallo stato dei bit RSO e RS1 del registro di stato PSW.
  Es.:

ADD A,R7

La somma viene effettuata tra il contenuto del registro A e il contenuto del registro R7 il risultato sarà contenuto nell'accumulatore A

.

- **Indirizzamento immediato**:
in alcune istruzioni il byte che segue il codice operativo può rappresentare il valore di una costante e non di un indirizzo.

Es.:

ADD A,#136

La somma viene effettuata tra il contenuto dell'accumulatore A e il valore costante 136.

- **Indirizzamento indicizzato**: è possibile utilizzare l'indirizzamento indicizzato solo con la memoria dei programmi (memoria a sola lettura). Questo metodo di indirizzamento è stato previsto per permettere l'accesso a tabelle di ricerca nella memoria dei programmi. Un registro a 16 bit (DPTR oppure il program counter) punta alla base della tabella e I'accumulatore viene utilizzato come indice all'interno della tabella. L'indirizzo effettivo del dato nella tabella viene ottenuto sommando a DPTR o PC il contenuto dell'accumulatore. Un altro tipo di indirizzamento indicizzato viene utilizzato nell'istruzione di salto con selezione multipla. In questo caso I'indirizzo di salto viene calcolato sommando alla base il contenuto dell'accumulatore.

Es.:

MOVC A, @A + DPTR

Mette in A il contenuto della locazione di memoria di programma ricavata da A + DPTR

- **Indirizzamento implicito**:

alcune istruzioni sono specifiche di alcuni registri per cui nel codice operativo è implicitamente specificato di quale registro si tratta

Es.:

DIV AB

Dividi i due registri A/B ed il risultato andrà nell'accumulatore A.

6.2 LISTA ISTRUZIONI

NOTAZIONI

A - Accumulatore (Registro "A")
AB - Tra l'Accumulatore ed il registro "B"
DPTR - Data Pointer
C - Carri bit
Rn - Registro (R7-R0)del banco di registri "R" attualmente selezionato
@Ri - Indirizzamento indiretto tramite un registro R (R0 o R1) nella RAm interna da 0 a 255.
@DPTR - Indirizzamento diretto tramite DPTR
@A+DPTR - Indirizzamento indiretto tramite la somma di DPTR piu' "A"
@A+PC - Indirizzamento indiretto tramite la somma di PC piu' "A"
iram addr - Indirizzo della RAM interna
#data - Indirizzamento immediato
bit addr -indirizzo del bit
/bit addr - indirizzo del bit (prendi il bit negato)
reladdr - Indirizzo relativo (da +128 a -127)
page i - pagina i (0 -7)
code addr - Indirizzo di programma

ISTRUZIONI ARITMETICHE

ADD, ADDC: Somma l'Accumulatore (con il riporto)
DA: Aggiusta la parte decimale
DEC: Decrementa il registro
DIV: Dividi l'Accumulatore per il registro B
INC: Incrementa il registro
MUL: Moltiplica l'Accumulatore per il registro B
SUBB: Sottrai dall'Accumulatore con il prestito

ISTRUZIONI LOGICHE

ANL: AND logico
CLR: Azzera il registro
CPL: Complementa il registro
ORL: OR logico
RL: Ruota l'accumulatore a sinistra
RLC: Ruota l'Accumulatore a sinistra attraverso il Carry
RR: Ruota l'accumulatore a destra
RRC: Ruota l'Accumulatore a destra attraverso il Carry
SWAP: Scambia i nibble dell'Accumulatore
XRL: OR esclusivo logico

ISTRUZIONI DI TRASFERIMENTO DATI
MOV: Trasferisci un byte
MOVC: Trasferisci un byte della memoria programma
MOVX: Trasferisci un byte della memoria estesa
POP: Prendi l'Accumulatore dallo stack
PUSH: Metti l'accumulatore nello stack
XCH: Scambia i byte
XCHD: Scambia i digit
ANL: AND per variabili a bit
CLR: Azzera il bit
CPL: Complementa il bit
JB: Salta se il bit e' a uno
JBC: Salta se il bit e' a uno e resettalo
JC: Salta se il Carry e' a uno
JNB: Salta se il bit non e' a uno
JNC: Salta se il Carry non e' a uno
MOV: Trasferisci un bit
ORL: OR a bit
SETB: Poni il bit a uno

ISTRUZIONI DI SALTO
ACALL: Chiamata a subroutine con indirizzo assoluto
AJMP: Salto asssoluto
CJNE: Compara e salta se non uguale
DJNZ: Decrementa il registro e salta se il risultato non e' nullo
JMP: Salta all'indirizzo
JNZ: Salta se l'Accumulatore non e' nullo
JZ: Salta se l'Accumulatore e' nullo
LCALL: Chiamata a subroutine con indirizzo a 16-bit
LJMP: Salto con indirizzo a 16-bit
NOP: Nessuna operazione
RET: Torna da subroutine
RETI: Torna da interrupt
SJMP: Salto con indirizzo relativo a 8-bit

ELENCO ALFABETICO DELLE ISTRUZIONI – 8051
ACALL: Chiamata a subroutine con indirizzo assoluto
ADD, ADDC: Somma l'Accumulatore (con il riporto)
AJMP: Salto asssoluto
ANL: AND logico
ANL: AND per variabili a bit
CJNE: Compara e salta se non uguale
CLR: Azzera il registro

CLR: Azzera il bit
CPL: Complementa il registro
CPL: Complementa il bit
DA: Aggiusta la parte decimale
DEC: Decrementa il registro
DIV: Dividi l'Accumulatore per il registro B
DJNZ: Decrementa il registro e salta se il risultato non e' nullo
INC: Incrementa il registro
JB: Salta se il bit e' a uno
JBC: Salta se il bit e' a uno e resettalo
JC: Salta se il Carry e' a uno
JMP: Salta all'indirizzo
JNB: Salta se il bit non e' a uno
JNC: Salta se il Carry non e' a uno
JNZ: Salta se l'Accumulatore non e' nullo
JZ: Salta se l'Accumulatore e' nullo
LCALL: Chiamata a subroutine con indirizzo a 16-bit
LJMP: Salto con indirizzo a 16-bit
MOV: Trasferisci un byte
MOV: Trasferisci un bit
MOVC: Trasferisci un byte della memoria programma
MOVX: Trasferisci un byte della memoria estesa
MUL: Moltiplica l'Accumulatore per il registro B
NOP: Nessuna operazione
ORL: OR logico
ORL: OR a bit
POP: Prendi l'Accumulatore dallo stack
PUSH: Metti l'accumulatore nello stack
RET: Torna da subroutine
RETI: Torna da interrupt
RL: Ruota l'accumulatore a sinistra
RLC: Ruota l'Accumulatore a sinistra attraverso il Carry
RR: Ruota l'accumulatore a destra
RRC: Ruota l'Accumulatore a destra attraverso il Carry
SETB: Poni il bit ad uno
SJMP: Salto con indirizzo relativo a 8-bit
SUBB: Sottrai dall'Accumulatore con il prestito
SWAP: Scambia i nibble dell'Accumulatore
XCH: Scambia i byte
XCHD: Scambia i digit
XRL: OR esclusivo logico
http://www.8052.it/set8051.htm

SET D'ISTRUZIONI: TRASFERIMENTO DATI
MOV
Istruzione: MOV

Funzione: Trasferisci un byte di Memoria
Sintassi: MOV *Operando1,Operando2*

6.3 GESTIONE DEGLI INTERRUPTS

Il MCU 80C51 riesce a controllare diversi dispositivi periferici in due modi.

- Polling
- Interrupts Polling

E comunicare attraverso più interconnessioni con un singolo processore, in questo caso viene utilizzato il metodo di polling. Il MCU chiama periodicamente ogni circuito logico esterno o dispositivo periferico ed esamina se ha richiesto o meno il servizio. Se il circuito logico esterno o i dispositivi periferici non richiedono assistenza dall'MCU, o se uno di essi richiede assistenza, il MCU commuta per eseguire il programma di servizio del rispettivo circuito logico esterno. In altre parole, il polling è il processo di chiamata del client in modo che i dispositivi periferici possano inviare dati dopo essere stati chiamati dal processore. Se il dispositivo periferico client dispone di dati, li invia dopo il processo di polling e, se non sono presenti dati, il dispositivo periferico client risponde negativamente e il MCU chiama il client successivo. Quando alcuni dispositivi richiedono la manutenzione dal MCU, la richiesta viene inviata al MCU impostando la linea SRQ su bassa. Dopo che il controller ha ricevuto una richiesta dal client per il servizio, il microcontrollore chiama tutti i dispositivi sul bus per trovare il dispositivo che ha inviato quella richiesta. Il metodo di polling utilizza semplicemente una sezione di codice che controlla uno o più flag particolari, per conoscere lo stato di tutte le operazioni eseguite. Il metodo di polling fa sempre parte del codice principale e non fa parte di un ISR (Interrupt Service Routine). Il metodo di polling non implica alcuna priorità per servire il dispositivo periferico. Anche la sezione del codice verrà sempre eseguita all'interno di un file di programma prefissato, in ordine di tempo e in una sequenza fissa. È facile da eseguire il debug e non ha alcun effetto sull'esecuzione di altre sezioni di codice. Infatti si può vedere che, nel metodo polling, i dispositivi esterni dipendono dal MCU, e solo il micro ha il diritto di ottenere l'accesso alle informazioni di cui ha bisogno.
Nel metodo degli interrupt, la sezione di controllo del MCU risponde solo quando si verifica un'interruzione da parte del dispositivo periferico. L'Interrupt è il segnale ricevuto dal microcontrollore per marcare l'evento che richiede attenzione immediata da parte del dispositivo esterno quindi esegue un codice speciale richiesto dal dispositivo. Ogni volta che un dispositivo necessita del suo servizio dal MCU, il dispositivo avvisa il microcontrollore inviandogli un segnale di interrupt.

- Struttura dell'interrupt

Il microcontrollore 8051 ha principalmente cinque sorgenti di interruzioni. Nel microcontrollore 8051 su cinque sorgenti di interrupt, ci sono 2 interrupt esterni come INT0 e INT1, 2 interrupt timer TF0, TF1 e 1 interrupt seriale come RI o TI. Gli interrupt esterni INT0 e INT1 possono essere attivati sia dal livello che dal fronte. Ciò dipende dai bit IT0 e IT1 del registro TCON. I flag che

generano questo tipo di interrupt sono i bit IE0 e IE1 del registro IE. INT0 e INT1 sono interrupt esterni su P3.2 e P3.3 rispettivamente. TF0 e TF1 sono interrupt di overflow del timer rispettivamente per il timer 0 e 1. Gli interrupt seriali RI e TI possono essere configurati per attivarsi alla trasmissione o alla ricezione di un byte durante la comunicazione seriale.

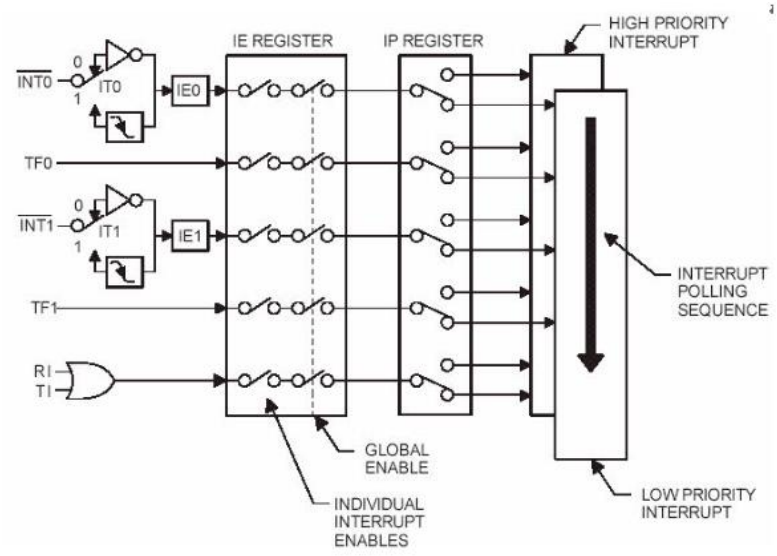


Fig.5.1

L'interrupt della porta seriale è generato dall'OR logico dei registri RI / TI. Questi flag non vengono cancellati dall'hardware, quando la routine di servizio è vettoriale. È la routine di servizio che deve determinare se è stato RI o TI a generare l'interrupt. Quando un interrupt esterno viene generato dal dispositivo periferico, il flag che ha generato l'interruzione viene cancellato dall'hardware su chip quando la routine di servizio viene vettorizzata nella posizione ISR. Questo accade solo quando l'interrupt è di tipo edge-triggered. Se l'interrupt è di tipo attivato dal livello, il dispositivo periferico esterno controlla il flag richiesto, piuttosto che l'hardware su chip. IE e IP sono i registri utilizzati per abilitare e impostare la priorità del sistema di interrupt nel microcontrollore 8051. In base alle priorità degli interrupt, gli interrupt con la priorità più bassa non vengono serviti fino a quando non lo è il microcontrollore che ha finito con quelli con priorità più alta.

Quando due o più interrupt arrivano allo stesso tempo, il microcontrollore li accoda secondo la loro priorità. Tutti gli interrupt del microcontrollore vengono disabilitati e quindi tutti questi interrupt devono essere abilitati da istruzioni software. Tutti gli interrupt del MCU 8051 possono essere impostati o cancellati da un Bit di registro di funzioni speciali come Interrupt Enabled (IE), e questo a sua volta dipende anche dalla priorità, che viene eseguita da IP, ovvero registro di priorità degli interrupt. Il registro IE ha anche un bit di disattivazione globale, che può essere cancellato per disabilitare tutti gli interrupt contemporaneamente. Ogni sorgente di interrupt può anche essere impostata individualmente su uno dei due livelli di priorità impostando o cancellando un bit in IP, ad esempio il registro di priorità degli interrupt. In generale, un interrupt a bassa priorità può essere interrotto da un interrupt ad alta priorità, ma non da un altro a bassa priorità. Se interrupt ad alta priorità non può essere interrotto da nessun'altra sorgente di interrupt. le richieste dello stesso livello di priorità vengono ricevute simultaneamente dal MCU 8051, una sequenza di polling interna determina quale richiesta deve essere servita.

Quando il MCU 8051 riceve un segnale di interrupt, salva l'indirizzo dell'istruzione successiva utilizzando il contatore del programma, ad esempio PC e puntatore di stack, ovvero i registri SP

(Stack Pointer). Significa che il valore del contatore del programma (PC) del programma interrotto viene memorizzato (inserito) nella memoria dello stack. Quindi salta a una posizione fissa in memoria, chiamata tabella vettoriale di interrupt. Questa tabella contiene l'indirizzo dell'ISR, ovvero la routine del servizio di interruzione. Ogni interrupt all'interno del MCU 8051 ha il proprio indirizzo ISR. Il MCU ottiene quindi questo indirizzo dell'ISR dalla tabella del vettore di interrupt e salta a quell'indirizzo per servire l'interrupt richiedente. Inizia a eseguire la subroutine del servizio di interrupt fino a quando non raggiunge l'ultima istruzione della subroutine nota come istruzione RETI, ovvero ritorno dall'interruzione.

- ISR o gestore degli interrupt

Gli interrupt sono eventi o segnali di emergenza che sospendono temporaneamente il programma principale, trasferiscono il controllo del programma ad altre subroutine o sorgenti ed eseguono il loro compito.
Dopo la selezione di uno specifico interrupt, il passo successivo è specificare al MCU cosa fare quando si verifica una richiesta di interrupt. È anche noto come il gestore di interrupt viene chiamato automaticamente quando si verifica una richiesta di interruzione. Ogni interrupt ha il suo numero di subroutine specifico. ISR non è altro che una sezione di codice di programmazione che viene eseguita nel caso in cui il MCU identifichi un qualsiasi interrupt durante il suo normale funzionamento. Un ISR o un gestore di interrupt gestisce la richiesta di interrupt e quindi la invia alla CPU e dice al processore di intraprendere l'azione appropriata per il file iniziando dalla richiesta di interrupt.

- Descrizione degli interrupt del MCU 8051 All'attivazione dei pin di interrupt, il MCU 8051 salta alla tabella vettoriale di interrupt per eseguire la routine di servizio di interrupt.
- Reset Interrupt (RESET)

Quando viene ricevuto un RESET interrupt dal MCU 8051, il microcontrollore 8051 riavvia l'esecuzione del codice del programma dalla posizione 0000H. Un interrupt di ripristino è un interrupt non mascherabile. Quindi non può essere evitato quando viene chiamato o ricevuto. Poiché nessuna combinazione di Bit in alcun registro può interrompere o mascherare l'azione di ripristino, a differenza di altri interrupt. Questo interrupt è noto anche come interrupt Power on Reset (POR).
Il processo di Reset si verifica quando al pin RST viene fornito un impulso positivo della durata di almeno 2 cicli macchina o 24 cicli di clock dell'oscillatore a cristallo. Quindi il MCU genera un segnale di ripristino interno che cancella tutti gli SFR, ad eccezione dei registri SBUF. Il puntatore allo stack e lo stato delle prime due porte non sono definiti. Mentre il valore FF viene scritto sulle porte configurando tutti i loro pin come ingressi. Il pin RST è solitamente collegato a un pulsante o circuito di ripristino dell'accensione o a entrambi.

- Interrupt timer (TF0 e TF1)

Nel MCU 8051, ci sono due interrupt interni dei timer chiamati timer 0 o TF0 e timer 1 o TF1. La posizione della tabella vettoriale di interrupt 000BH e 001BH appartiene rispettivamente agli interrupt TFO e TF1. Gli interrupt del timer 0 e del timer 1 vengono generati quando i registri corrispondenti T0 (TH0 e TL0) e T1 (TH1 e TL1) vengono rispettivamente fatti ruotare.
Ogni volta che il timer va in overflow, vengono impostati i flag di overflow del timer come TF0 e TF1. Il flag del timer TF viene portato a valore alto quando il timer si ribalta, cioè raggiungono il valore programmato e il microcontrollore salta quindi alla routine di servizio di interruzione del

timer. Senza timer con interruzione dobbiamo interrogare la TF e attendere che la TF venga sollevata. Pertanto il microcontrollore è bloccato in attesa che TF venga sollevato e non può fare nient'altro. Quindi usando il timer con interrupt possiamo risolvere questo problema ed evita di legare il microcontrollore. Se il flag di interruzione del timer nel registro IE è abilitato, quindi ogni volta che il timer si sposta, la TF viene aumentata. Pertanto, il MCU 8051 viene interrotto in qualunque processo stia eseguendo e salta alla tabella vettoriale di interrupt per servire l'ISR.

- Interrupt esterni (INT0 e INT1)

Il MCU 8051 ha due interrupt hardware esterni come INT0 e INT1. La posizione 0003H e 0013H della tabella vettoriale di interrupt appartiene rispettivamente agli interrupt INTO e INT1. Gli interrupt generati da una fonte esterna sono definiti interrupt esterni. Gli interrupt esterni sono gli interrupt ricevuti dai dispositivi periferici esterni che sono interfacciati con il microcontrollore 8051. Sono ricevuti ai pin INT0 e INT1 del microcontrollore 8051. Queste interruzioni possono essere di entrambi i livelli attivato o edge triggered. Negli interrupt attivati dal livello, l'interrupt può essere abilitato a un impulso basso sui pin INT0 e INT1. Mentre in caso di interrupt attivati dal fronte, l'interrupt può essere abilitato da un impulso alto a un impulso basso sui pin INT0 e INT1. La condizione di interrupt attivata dal fronte o dal livello viene decisa dal registro TCON. Un interrupt esterno informa il microcontrollore 8051 che un dispositivo esterno necessita della sua routine di servizio di interrupt. Gli interrupt esterni INT0 e INT1 possono essere attivati sia dal livello che dal fronte. Ciò dipende dai bit IT0 e IT1 forniti nel registro TCON. I flag che generano questo tipo di interrupt sono i bit IE0 e IE1. Gli interrupt esterni sono gli interrupt ricevuti dalle periferiche esterne che si interfacciano con il microcontrollore 8051. Sono ricevuti ai pin INT0 e INT1 del microcontrollore. Quando viene generato un interrupt esterno, il flag che ha generato l'interrupt viene cancellato dall'hardware quando la routine di servizio viene vettorizzata nella posizione ISR. Ciò accade solo se l'interrupt è stato attivato dal fronte. Questi interrupt hardware esterni possono essere programmati per essere attivati a livello o attivati dalla transizione impostando o cancellando il bit IT1 o IT0 del registro TCON.

- Se IT0 / 1 = 0, l'interrupt esterno viene attivato da un livello basso rilevato sul pin INT0 / 1.
- Se IT0 / 1 = 1, l'allarme esterno viene attivato sul fronte. Se l'interruzione è stata attivata dal livello, la sorgente richiedente esterna controlla il flag richiesto, piuttosto che l'hardware su chip. Entrambi questi interrupt sono attivi bassi Se l'interrupt esterno è attivato dal livello, la sorgente esterna deve mantenere attiva la richiesta fino a quando l'interrupt richiesto non viene effettivamente generato. Quindi deve disattivare la richiesta prima che la routine del servizio di interruzione sia completata, altrimenti verrà generato un altro interrupt. In tale situazione, i successivi campioni del pin INT0 / 1 mostrano una logica

alto in un ciclo e logico basso nel ciclo successivo, quindi verrà impostato il flag di interrupt esterno IE0 / 1 nel registro TCON. Pertanto si dice che il microcontrollore sia interrotto. Poiché i pin di interrupt esterni vengono campionati una volta in ogni ciclo della macchina, un ingresso alto o basso deve essere mantenuto per almeno 12 periodi dell'oscillatore per garantire il campionamento. Se l'interrupt esterno è attivato dalla transizione, la sorgente esterna deve mantenere il pin di richiesta di interrupt con logica alta attiva per almeno un ciclo macchina, quindi tenerlo a logica bassa attiva per almeno un ciclo macchina per garantire che l'interruzione flag di richiesta IE0 e IE1 verranno impostati. I flag IE0 e IE1 verranno automaticamente cancellati dal microcontrollore quando viene chiamata la routine di servizio di interrupt. L'interrupt esterno ha due tipi di livello di attivazione come:

1. Livello attivato (l'interruzione si verifica al rilevamento di alto / basso livello)

2. Fronte attivato (si verifica un interrupt al rilevamento del fronte di salita / discesa)

Per impostazione predefinita, gli interrupt esterni vengono attivati ad un determinato livello. Nell'attivazione del livello, l'interrupt diventerà attivo quando l'impulso di clock è su un livello particolare, cioè alto o basso. Tale livello può essere definito dal programmatore. A livello negativo l'attivazione dell'interrupt sarà attivo quando il segnale di clock è basso. In un livello positivo l'attivazione dell'interrupt sarà attivo quando il segnale di clock è alto. Quando i pin di interrupt esterni come INT0 e INT1 ricevono un segnale di basso livello per una durata minima di quattro cicli macchina, si verifica un fenomeno di interrupt. Questo segnale di basso livello deve essere trasferito a un livello alto prima che il microcontrollore raggiunga l'istruzione RETI nell'ISR. Se il segnale rimane basso dopo l'istruzione RETI, interromperà nuovamente il MCU. A causa di ciò, il microcontrollore può rimanere bloccato in un ciclo di interruzioni. Pertanto, in modalità interrupt con trigger di livello, per garantire l'attivazione dell'interrupt hardware ai pin INT0 e INT1, si deve assicurarci che la durata del segnale di basso livello sia di circa 4 cicli macchina ma non di più.
Durata minima richiesta per rilevare interrupt attivato dal livello = 1 ciclo macchina = 1,085 µS
Pertanto, 4 cicli macchina = 4 × 1.085 µS
Nella modalità attivata dal livello, i pin INT0 e INT1 sono normalmente ad alto livello per impostazione predefinita. Se viene applicato un segnale "0" di livello basso o logico, allora attiva gli interrupt INT0 e INT1. Quando il MCU riceve tale interruzione sui suoi pin interni come INT0 e INT1, interrompe i processi che sta attualmente eseguendo e salta alla tabella del vettore di interrupt per fornire il servizio a quell'interrupt. L'importante è ricordare che, il segnale "0" logico o di basso livello ai pin INT0 e INT1 deve essere rimosso prima dell'esecuzione dell'ultima istruzione dell'ISR come RETI. In caso contrario, verrà generato un altro interrupt durante tale situazione. In altre parole, se il segnale di interrupt di basso livello non viene rimosso prima che l'ISR sia terminato, può essere interpretato come un altro interrupt e il microcontrollore 8051 salta alla tabella vettoriale di interrupt per eseguire nuovamente l'ISR.
In caso di modalità edge triggering, l'interrupt sarà attivo quando il fronte negativo o positivo del segnale di clock viene applicato ai pin INT0 e INT1. Ad esempio, se l'interrupt è attivato dal fronte positivo, riceverà l'input da sorgenti esterne esattamente nel momento in cui il segnale di clock passa da impulso basso a impulso alto. Allo stesso modo, se l'interrupt è innescato dal fronte negativo, riceverà input da sorgenti esterne esattamente nel momento in cui il segnale di clock passa da impulso alto a impulso basso. Per fare in modo che INT0 e INT1 siano interrupt attivati dal fronte, i Bit del registro TCON devono essere programmati di conseguenza. Il registro TCON ha i Bit flag come IT0, ovvero TCON.0 e IT1 per determinare la modalità triggerata dal livello o dal fronte degli interrupt hardware esterni. Per fare in modo che INT0 e INT1 siano interrupt attivati dal fronte, i Bit flag TCON.0 e TCON.2 devono essere a impulso alto.
Si può fare tale disposizione utilizzando istruzioni come "SETB TCON.0" e "SETB TCON.2". Quindi gli interrupt hardware esterni come INT0 e INT1 sono detti interrupt attivati da edget. Pertanto, il MCU 8051 viene interrotto e il controllo del programma è costretto a saltare alla posizione di interruzione come 0013H nella tabella vettoriale di interruzione per servire l'ISR. Pertanto, in modalità edge triggered interrupt, per garantire l'attivazione dell'interrupt hardware ai pin INT0 e INT1, si deve sempre assicurare che la durata del segnale sorgente esterna debba essere mantenuta ad impulso alto per almeno un ciclo macchina, e quindi mantenuto a impulsi bassi per

almeno un ciclo macchina. Il fronte di discesa dei pin INT0 e INT1 sono bloccati dal MCU 8051 e sono trattenuti dal TCON.1 cioè IE0 e TCON.3 cioè IE1 Bit del registro TCON.
Durata minima richiesta per rilevare interrupt innescato dal fronte = 1 ciclo macchina = 1,085 µS a impulso alto
= 1,085 µS a bassa pulsazione.

- Interrupt seriali (RI o TI)

Il MCU 8051 dispone di una porta di comunicazione seriale. Quindi questi interrupt vengono utilizzati per la comunicazione seriale. Questa porta ha flag di interrupt seriale come TI o RI. La posizione 0023H della tabella vettoriale di interrupt appartiene ai flag di interrupt seriale come TI o RI. Se uno di questi flag è impostato, si verifica un interrupt seriale. Questi flag di interrupt vengono utilizzati per la ricezione e la trasmissione di dati seriali durante la comunicazione seriale. L'RI indica che un byte di dati è stato ricevuto ed è disponibile per la lettura nel buffer di ingresso. Il TI indica che il byte di dati precedente è stato inviato in serie al dispositivo periferico e un nuovo byte di dati può essere scritto sulla porta seriale del microcontrollore. Entrambi i flag come RI o TI possono essere impostati, in base all'operazione di lettura o scrittura con il buffer seriale come il registro SBUF.

I buffer seriali di input e output sono distinti ma si trovano allo stesso indirizzo. I flag RI o TI non vengono cancellati automaticamente quando viene gestito un interrupt. Pertanto, la routine del servizio di interruzione deve cancellare tali flag prima di ritornare nello stato normale. Quando viene trasmesso con successo l'ultimo bit o bit di stop di un byte di dati, viene impostato il flag TI e quando viene ricevuto con successo l'ultimo bit o bit di stop della ricezione del byte di dati, viene impostato il flag RI. Quando entrambi i flag sono abilitati, notifica al microcontrollore se è stato un byte di dati ricevuto o trasmesso.
Per utilizzare la comunicazione seriale con interrupt, dobbiamo impostare il Bit ES insieme al Bit EA.

- 8051 Programma in linguaggio assembly per l'invio del carattere "A" alla porta seriale.
SERIALE:

CHECK_TI: EXIT:
JNB RI, CHECK_TI
MOV A, SBUF
CLR RI
JNB TI, ESCI
CLR TI
MOV SBUF, # ’A’ RETI

- ✓ Se il flag RI non è impostato, passare all'etichetta CHECK_TI; Copia byte di dati dalla porta seriale in A
- ✓ Cancella bit RI, dopo averlo ricevuto con successo
- ✓ Se il flag TI non è impostato, passare all'etichetta EXIT; Cancella bit TI prima di inviare un altro carattere; Invio di un altro carattere alla porta seriale; Ritorno dall'interruzione

Se sono stati impostati entrambi i flag come RI e TI, verranno eseguite entrambe le sezioni di codice. Notare inoltre che ogni sezione di codice cancella il flag di interrupt corrispondente. Se ci si

dimentica di cancellare i Bit di interrupt di RI e TI, l'interrupt seriale verrà eseguito più e più volte fino a quando non si cancella il Bit. Quindi è molto importante cancellare sempre i flag di interrupt in un interrupt seriale.

- Registro IE

Questo è un registro a 8 Bit utilizzato per abilitare o disabilitare gli interrupt. È bit indirizzabile. Questo registro ha un indirizzo in byte come "A8h". Tutti i Bit che generano interrupt possono essere impostati o cancellati da software o hardware. Per abilitare uno qualsiasi degli interrupt, prima il bit EA(Enable Interrupt) deve essere impostato a 1. Dopodiché vengono abilitati i Bit corrispondenti agli interrupt desiderati. Il livello logico "1" applicato ai bit all'interno del registro IE attiva l'interrupt e lo 0 "logico" applicato disabilita l'interrupt. La struttura del registro IE è mostrata di seguito.

Bit	Nome	Indirizzo Bit	Funzione dell'interrupt Handler
7	EA	AFh	Abilita globalmente gli interrupt
6	-	AEh	Non definito
5	-	ADh	Non definito
4	ES	ACh	Abilita l'interrupt della seriale
3	ET1	ABh	Abilita l'interrupt del Timer 1
2	EX1	AAH	Abilita l'interrupt esterno INT1
1	ET0	A9h	Abilita l'interrupt del Timer 0
0	EX0	A8h	Abilita l'interrupt esterno INT1

Fig: - Formato registro IE

EA (Bit n. 7 o IE.7)

- ✓ EA = 0, disabilita tutti gli interrupt
- ✓ EA = 1, Abilita interrupt specifici.

Il bit n. 6 e il bit n. 5 sono bit riservati.

ES (Bit n. 4 o IE.4)

- ✓ES = 0, disabilita l'interruzione della porta seriale.
- ✓ES = 1, Abilita l'interrupt della porta seriale.

ET1 (bit n. 3 o IE.3)

- ✓ET1 = 0, disabilita l'interrupt del timer 1, cioè TF1.
- ✓ET1 = 1, Abilita l'interrupt 1 del timer, cioè TF1.

EX1 (bit n. 2 o IE.2)

- ✓ EX1 = 0, disabilita l'interrupt esterno 1, ovvero INT1.
- ✓ ET1 = 1, Abilita l'interrupt esterno 1, ovvero INT1.

ET0 (bit n. 1 o IE.1)

- ✓ ET0 = 0, disabilita l'interrupt del timer 0, cioè TF0.
- ✓ ET0 = 1, Abilita l'interrupt del timer 0, ovvero TF0.

EX0 (bit n. 0 o IE.0)

- ✓ EX0 = 0, disabilita l'interrupt esterno 0, ovvero INT0.
- ✓ ET0 = 1, abilita l'interrupt esterno 0, ovvero INT0.

- Attivazione e disattivazione degli interrupt

Nel MCU 8051 l'abilitazione degli interrupt o il registro IE viene utilizzato per abilitare o disabilitare gli interrupt. All'accensione del sistema nel MCU, tutti gli interrupt sono disabilitati per impostazione predefinita. Ad esempio, se il bit TF0 è impostato, il microcontrollore 8051 non eseguirà tale interrupt. Gli interrupt devono essere abilitati o disabilitati dal programma software affinché il microcontrollore risponda ad essi utilizzando il registro IE.

Si considerano le seguenti istruzioni per abilitare o disabilitare gli interrupt,

MOV IE, # 01H; Enable INT 0 Interrupt
MOV IE, # 02H; Enable TF 0 Interrupt
MOV IE, # 80H; Enable Specific Interrupts
MOV IE, # 10H; Enable Serial Interrupt
MOV IE, # 10010100B; Enable Serial Interrupt,INT1 Interrupt
SETB IE.1; Enable Timer 0 i.e. TF0 Interrupt
SETB IE.2; Enable INT1 Interrupt
SETB IE.4; Enable Serial Interrupt
CLR IE.1; Disable Timer 0 i.e. TF0 Interrupt
CLR IE.7; Disabilita tutti gli interrupt
SETB IE.7; Abilita specifici interrupt
SETB EA; Abilita specifici interrupt
SETB EX0; Abilita INT0 Interrupt
SETB EX1; Abilita INT1 Interrupt
SETB ET0; Abilita Timer 0 Interrupt
SETB ET1; Abilita Timer 1 Interrupt
SETB ES; Abilita porta seriale

Questo è un registro IP a 8 Bit. è noto come registro di priorità degli interrupt. Il microcontrollore 8051 ha due livelli di priorità di interrupt alta o bassa. Questo registro consente di controllare l'ordine in cui verranno serviti più interrupt. Questo registro ha un indirizzo in byte indirizzo come "B8H".

Anche questo registro è indirizzabile a bit. La priorità degli interrupt può essere modificata utilizzando i Bit di controllo nel registro IP.
Per fornire una priorità più alta a uno qualsiasi degli interrupt, dobbiamo rendere il Bit corrispondente nel registro IP con un "1" logico o attivo alto. In alcune situazioni, quando due o più bit di interrupt nel registro IP sono impostati su alti, verrà servito per primo l'interrupt che ha la priorità più alta.

Formato registro IP (B8h)

--	--	--	PS	PT1	PX1	PT0	PXO

Il bit n. 7, il bit n. 6 e il bit n. 5 sono bit riservati.

EA (bit n. 4 o IP.4)

- ✓ PS = 0, Assegna priorità bassa all'interrupt seriale.
- ✓ PS = 1, Assegna alta priorità all'interrupt seriale.

PT1 (bit n. 3 o IP.3)

- ✓ PT1 = 0, Assegna priorità bassa a Timer1, ovvero interrupt TF1.
- ✓ PT1 = 1, Assegna alta priorità a Timer1, ovvero interrupt TF1.

PX1 (bit n. 2 o IP.2)

- ✓ PX1 = 0, Assegna bassa priorità all'interrupt esterno, ad esempio INT1.
- ✓ PX1 = 1, Assegna alta priorità all'interrupt esterno, ovvero INT1.

PT0 (bit n. 1 o IP.1)

- ✓ PT0 = 0, Assegna priorità bassa al Timer 0, ovvero interrupt TF0.
- ✓ PT0 = 1, Assegna alta priorità al timer 0, ovvero interrupt TF0.

PX0 (bit n. 0 o IP.0)

- ✓ PX0 = 0, Assegna una priorità bassa all'interrupt esterno, ad esempio INT0.
- ✓ PX0 = 1, Assegna alta priorità all'interrupt esterno, ovvero INT0.
  In caso di priorità degli interrupt, si applicano le seguenti regole.
- ✓ Mentre è in esecuzione un gestore di interrupt a bassa priorità, se arriva un interrupt ad alta priorità, il gestore di interrupt verrà interrotto e verrà eseguito il gestore di alta priorità.
- ✓ Quando il gestore ad alta priorità ha completato con l'istruzione "RETI", il gestore a bassa priorità riprenderà. Quando questo gestore ha completato con l'istruzione "RETI", il controllo del programma viene restituito al programma principale.

- ✓ Se è in esecuzione un interrupt ad alta priorità, non può essere interrotto da nessun'altra sorgente.
- ✓ Un gestore di interrupt a bassa priorità verrà eseguito solo se nessun altro interrupt è già in esecuzione. Anche in questo caso, l'interrupt a bassa priorità non può consentire un altro interrupt a bassa priorità, anche se l'ultimo è più alto in ordine di sequenza.
- ✓ Se si verificano due interrupt contemporaneamente, verrà eseguito per primo l'interrupt con priorità più alta. Se entrambi gli interrupt hanno la stessa priorità, verrà eseguito per primo l'interrupt che è più alto nell'ordine di sequenza.

6.4 PROGRAMMI

SEND_1.A51

```
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
; Trasmette il carattere ascii 'A' continuamente utilizzando la porta seriale 8051.
; Utilizza dati a 9 bit a 9600 baud. Il bit di parità esiste ma non viene calcolato.
; Utilizza il funzionamento POLLED, non guidato da interrupt
; Rev. 0.1 Rocco Cannizzaro 28 Gennaio 1997
; xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
ORG 0000h; indirizzo di ingresso per 8051 RESET
LJMP MAIN; MAIN inizia oltre lo spazio vettoriale di interrupt
ORG 0100h

PRINCIPALE:; impostare il timer / contatore 1 per
        pilotare 9600 baudrate
MOV TMOD, # 00100000B; il timer / contatore 1 è impostato
per la modalità 2 TIMER 8 Bit
MOV TH1, # 0FDh; il timer / contatore 1 è programmato per
9600 baud
SETB TR1; il timer / contatore 1 è abilitato e funzionerà liberamente
;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
; Inizializza la porta seriale per la modalità 3: operazione

MOV SCON, # 11010000B

INVIARE:
MOV SBUF, # 41h; ascii 'A' a SBUF

CICLO CONTINUO:
JNB TI, LOOP; loop test TI per sapere quando i dati vengono
inviati
  CLR TI; cancellare TI
  LJMP SEND; torna per inviare di nuovo "A"
```

END

INTER.A51

xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
; Programma guidato da interruzione semplice per controllare la temperatura di un forno. Set a 200C
; il sensore si abbassa INT1 si interrompe provocando lo spegnimento del resistenza da parte di ISR1. Se
; Il sensore 190C diventa basso INT0 interrompe provocando l'accensione del riscaldatore da parte di ISR0.
; La porta 1, bit0, ovvero P1.0 si collega al riscaldatore.
;
; Rev. 0.0 Rocco Cannizzaro 19 luigio 97
; xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
ORG 0000h; indirizzo di ingresso per 8051 RESET
LJMP MAIN; MAIN inizia oltre lo spazio vettoriale interrupt
ORG 0003h; indirizzo del vettore per l'interrupt 0
LJMP ISR0; salta all'inizio di ISR0
ORG 0013h; indirizzo del vettore per l'interrupt 1
LJMP ISR1; salta all'inizio dell'ISR1
; xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
; MAIN abilita gli interrupt e definisce l'operazione di trigger negativo.
; Il riscaldatore è acceso e programma solo i loop lasciando che gli ISR facciano il lavoro.
; xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
ORG 0100h; definisce dove inizia

MAIN ..

PRINCIPALE:

MOV IE, # 10000101B; abilitare gli interrupt esterni IE0, IE1
SETB IT0; trigger sul fronte negativo per interrupt 0
SETB IT1; trigger sul fronte negativo per interrupt 1

; Inizializzare il riscaldatore su ON

SETB P1.0; il riscaldatore è acceso

CICLO CONTINUO:
LJMP LOOP; fare un giro senza eseguire nulla!
; xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
; ISR0 accende semplicemente il riscaldatore
; xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
ISR0:
SETB P1.0; accendere il riscaldatore

```
 RETI; ritorno dall'interruzione
; xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
; ISR1 spegne semplicemente il riscaldatore
; xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
ISR1:
 CLR P1.0; spegnere il riscaldatore
 RETI; ritorno dall'interruzione
END
```

BIBLIOGRAFIA:

1) Intel Data Sheet MCS80C51

2) Intel User Manual MCS8051

3) 8051 Microcontrollers: Hardware, Software and Applications

by David Calcutt (Author), Frederick Cowan (Author), Hassan Parchizadeh (Author)

4) Principles and Applications of Microcomputers: 8051 Microcontroller Software, Hardware, and Interfacing: Vol. I 8051 Assembly-Language Programming Paperback – September 5, 2016

by Ming-Bo Lin (Author)

Authors: Gimenez, Salvador Pinillos

5) 8051 MICROCONTROLLER: HARDWARE, SOFTWARE & APPLICATIONS Paperback – 1 July 2017
(Authors)
V.Udayashankara, M.S. Mallikarjunaswamy