ISTITUTO PROFESSIONALE DI STATO PER L'INDUSTRIA L'ARTIGIANATO

63039 SAN BENEDETTO DEL TRONTO (Ascoli Piceno)

Classe 5A-5B T.I.E.N. Anno Scolastico 2001/2002

MICROCONTROLLORI ST6

Hardware

SISTEMI, AUTOMAZIONE E ORGANIZZAZIONE DELLA PRODUZIONE

Microcontrollori ST6

1.1 Introduzione

Sono trascorsi più di 20 anni da quando fece la comparsa sul mercato il primo controllore, l'8084 dell'Intel. Da allora, senza che ce ne rendessimo conto, questi componenti si sono diffusi a macchia d'olio trovando impiego praticamente dappertutto, dal televisore al computer, dalla lavatrice alla macchina fotografica, dall'automobile all'apricancello, dalla workstation grafica al cellulare.
A riprova di ciò le statistiche ci informano che nel 1993 il valore dei microcontrollori ha superato di tre volte quello dei microprocessori dai cui questi dispositivi solitamente derivano.
L'importanza di questi componenti nelle applicazioni elettroniche di ogni tipo è già enorme ed ancora di più lo sarà nel futuro. Per questo motivo si ritiene opportuno per chiunque si occupi di elettronica, non solo conoscere più a fondo questi componenti ma anche apprendere la logica di funzionamento e le tecniche di programmazione. Per venire incontro a questa improrogabile esigenza, si è preparato un corso su questo argomento che metterà in grado chiunque di programmare e utilizzare questi particolari circuiti integrati. L'obiettivo principale di questo corso è la presentazione della vasta gamma dei microcontrollori ST6, non solo a livello didattico o redazionale ma anche pratico; infatti ad esso è abbinato un kit di sviluppo che consentirà, a chi lo desidera, di mettere realmente in pratica le tecniche di programmazione apprese. Va anche detto che il corso non ha la pretesa di sostituire i manuali originali della ST-Microelectronics, ma vuole offrire un servizio di consulenza iniziale e di rendere meno complicato l'apprendimento dei concetti di base. Il corso affronterà le tecniche di programmazione partendo dai concetti elementari e si completerà con esempi pratici.

1.2 - Caratteristiche di un microcontrollore

Il termine Microcontrollore o MCU (Microcontroller Unit) identifica un particolare circuito integrato che dispone nel suo interno di almeno cinque blocchi funzionali:

1) **Una interfaccia d'ingresso:** uno o più ingressi attraverso i quali acquisire dati e informazioni dal mondo esterno; a seconda del micro, agli ingressi possono venire applicati segnali digitali (livelli logici) o analogici (tensioni, frequenze) ed anche segnali più complessi come, ad esempio, quelli video.
2) **Una interfaccia di uscita:** una o più uscite in grado di controllare attuatori di ogni genere, display, monitor, dispositivi di potenza i quali trasformano la tensione in forza meccanica (motori).
3) **Una memoria ROM o EPROM:** solitamente chiamata "memoria programma", nella quale viene caricato il programma da eseguire; i dati e le istruzioni rimangono registrati anche quando viene a mancare alimentazione.
4) **Una CPU(denominata CORE):** definita come l'unità principale di calcolo, con relativo clock, in grado di interpretare ed eseguire scrupolosamente il programma contenuto nella memoria; in base a questo programma, la CPU elabora i segnali di ingresso e controlla le uscite.
5) **Una memoria RAM o EEPROM**: chiamata "memoria dati", nella quale la CPU legge e scrive le variabili, ovvero i dati presenti sugli ingressi e sulle uscite, i risultati delle operazioni ed altre informazioni; i dati presenti nella RAM vengono presi quando viene a mancare l'alimentazione.

Oltre a questi blocchi funzionali, indispensabili per poter definire microcontrollore un determinato chip, solitamente nei micro vengono implementate altre funzioni legate alla complessità del dispositivo; tra le più comuni segnaliamo:

- timer;
- convertitori A/D e D/A;
- generatori PWM;
- driver per display;
- numerose altre periferiche specializzate.

Da quanto fin qui esposto risulta evidente quale sia la differenza tra un microprocessore e un microcontrollore: il primo contiene esclusivamente l'unità centrale di calcolo (CPU) e quindi per poter funzionare necessita di una memoria ROM esterna nella quale viene scritto il programma, di una RAM per i dati e di alcuni integrati per l'interfacciamento; nei microcontrollori tutto ciò è contenuto all'interno di un singolo chip.
Il principio di funzionamento di un microcontrollore è molto semplice, e coincide con quello di un computer o elaboratore elettronico e può essere riassunto in solo tre operazioni eseguite dalla CPU:

- legge l'istruzione contenuta nella memoria programma;
- la interpreta;
- la esegue.

Riepilogo

Il microcontrollore è un dispositivo che raggruppa su un unico chip tutto il necessario per un sistema a microprocessore.

1. CPU (core)
2. Memoria RAM
3. Memoria Eprom o EEPROM od OTP
4. Porte I/O
5. Timers e contatori
6. Uart o porte di comunicazione seriale speciali ("Bus CAN","Bus I2C")
7. Convertitori A/D

Il sistema a microcontrollore offre tutti i vantaggi dei sistemi programmabili a costi molto contenuti, si pensi che attualmente il prezzo del single-chip più economico parte da poche migliaia di lire e già un sistema di questo tipo è in grado di sostituire un dispositivo a logica cablata di dimensione fisiche e costi ben superiori.

1.3 Vantaggi del sistema a microcontrollore:

1. minor spazio occupato;
2. minore complessità del circuito stampato;
3. sistema intelligente in grado di eseguire elaborazione complesse e di comunicare con altri dispostivi.
4. riconvertibilità del progetto (riprogrammando il dispositivo);
5. protezione contro le copiature (la maggiore parte del single-chip offre la possibilità di proteggere da lettura il programma contenuto nella ROM);
6. risparmio energetico (le versioni CMOS supportano il modo di funzionamento stand-by: è possibile bloccare, *via software*, l'attività della CPU e quindi ottenere correnti di alimentazione molto basse);
7. costo contenuto.

1.4 Sistemi e apparecchiature dove è possibile trovare un microcontrollore

1) Industria:

- controllo assi (velocità, posizione);
- regolatori automatici (PID).

2) Ufficio:

- Computer e periferiche (stampanti, mouse, tastiera, tavoletta grafica, scanner, plotter);
- Fax centralini telefonici e telefoni portatili.

3) **Casa**:

- TV, videotape, telecomandi, elettrodomestici;
- sistemi di sicurezza;
- auto;
- Smart-Cards (Bancomat, Carte di credito ecc.);
- apricancello;
- regolatori riscaldamento

1.5 - Descrizione degli ST626X

I microcontrollori ST626X della ST-Microelectronics sono conosciuti soprattutto per la loro grande facilità di utilizzo e per l'elevato numero di circuiti ausiliari integrati disponibili.
Pur non essendo di tipo RISC, e non avendo quindi delle prestazioni all'altezza di concorrenti come i PIC di Microchip, essi sono il più delle volte sufficientemente veloci per un gran numero di applicazioni. Usando una frequenza di clock massima di 8MHz, il tempo medio di esecuzione delle istruzioni si aggira sui 6.5 ⇑s.
L'architettura è di tipo Von Neumann, con singolo bus (da 8 bit) per memoria programma e memoria dati e dispone di un accumulatore A e di quattro registri X,Y,V,W tutti mappati direttamente in memoria RAM.

Caratteristiche principali:

- CPU a 8 bit
- Frequenza di clock massima di 8Mhz
- Alimentazione da 3 a 6 volt
- 4 Kbyte di memoria EPROM/PROM
- 128 byte di memoria RAM
- 128 byte di memoria EEPROM
- Un timer multifunzione
- Un secondo timer con AutoReload (generatore PWM)
- Un convertitore A/D a 8 bit
- Un watchdog digitale
- 6 livelli di stack
- Ingresso di interrupt esterno NMI

I modelli ST6260 hanno:

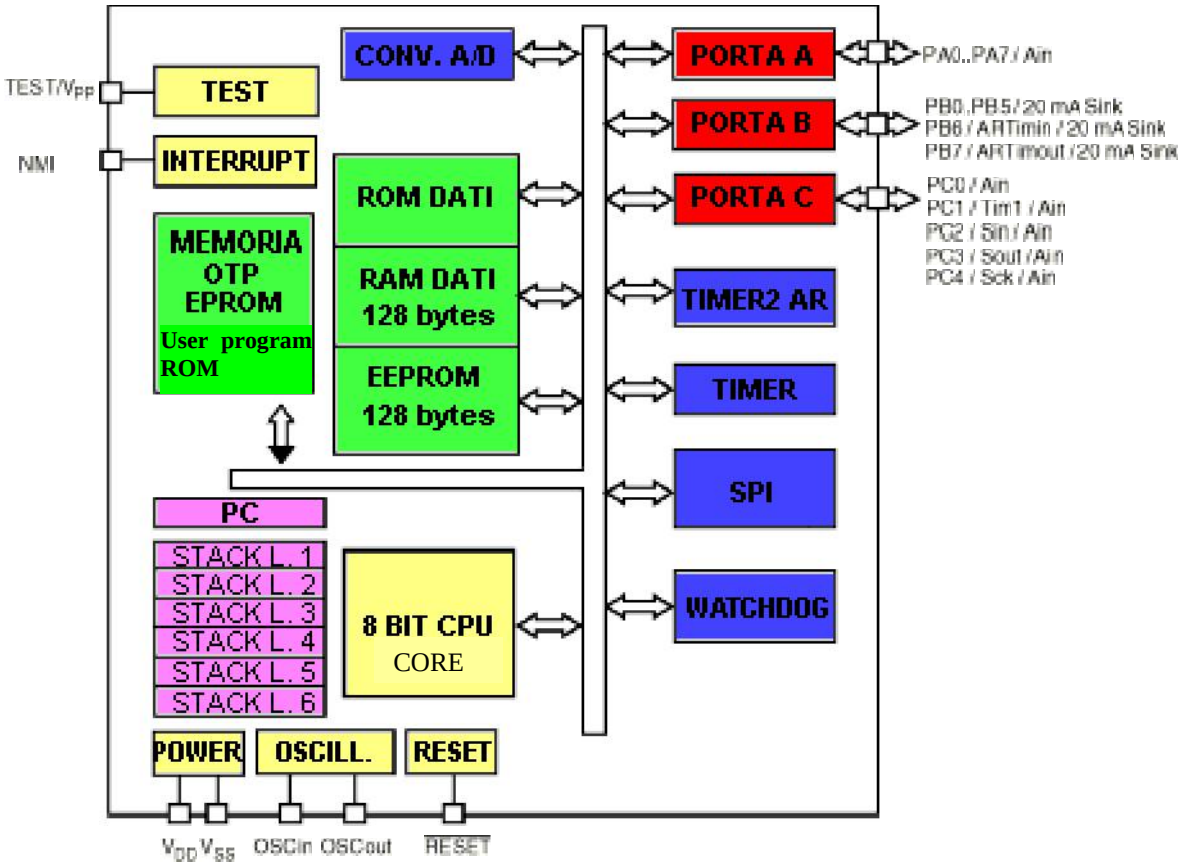
- Contenitore DIL a 20 pin
- 13 linee di I/O programmabili come:
  - Ingresso con o senza resistenza di pull-up

- Ingresso con interrupt
- Uscita open-collector o push-pull
- Ingresso analogico (solo 7 linee)

I modelli ST6265 hanno:

- Contenitore DIL a 28 pin
- 21 linee di I/O programmabili come:
  - Ingresso con o senza resistenza di pull-up
  - Ingresso con interrupt
  - Uscita open-collector o push-pull
  - Ingresso analogico (solo 13 linee)

1.6 Struttura interna



➢ Il "CORE"

La CPU rappresenta ovviamente l'elemento più importante di un microcontrollore, il "CORE" di tutto il sistema.

In base al numero di bit che il processore è in grado di elaborare è possibile individuare quattro grandi famiglie: i micro a 4 bit, quelli a 8, quelli a 16 e quelli a 32. I più diffusi sono i dispositivi a 8 bit che coprono circa il 50% del mercato mentre quelli più sofisticati e di recente produzione utilizzano la CPU di 32 bit.

Altra caratteristica molto importante dei microcontrollori (così come dei microprocessori) è la velocità di esecuzione che nei modelli più sofisticati può raggiungere alcune decine di MIPS (milione di operazioni per secondo).
La CPU contenuta nel chip è in grado di interpretare una serie di istruzioni che è specifica di quel determinato micro o della famiglia di appartenenza.
Purtroppo, non esiste un controllore uguale ad un altro; tuttavia, specie se ci riferiamo a dispositivi della fascia bassa e medio-bassa, la conoscenza di una famiglia di micro e delle relative istruzioni sono di grande aiuto nell'apprendimento della logica di funzionamento e delle istruzioni di una famiglia simile prodotta da un'altra casa costruttrice.
Il passaggio da una famiglia all'altra non rappresenta dunque un grosso problema per una persona esperta; il vero problema è un'altro.
Per programmare questi dispositivi, oltre a conoscere la logica di funzionamento ed il set di istruzioni, è necessario disporre di un emulatore o di un programmatore che sono specifici per ciascuna famiglia di integrati. Così, ad esempio, per programmare un micro della famiglia ST6 è necessario disporre del relativo programmatore della ST-Microelectronics , il quale non è adatto per un micro della famiglia PIC16XX della Microchip. Pertanto, prima di scegliere un microcontrollore, appartenente ad una famiglia, è necessario valutare accuratamente quali sono le proprie esigenze per evitare di dover passare ad un altro dispositivo, col relativo aggravio di costi dovuti al sistema di sviluppo e di programmazione. Per lo stesso motivo abbiamo dovuto fare delle scelte, individuare cioè una famiglia di microcontrollori da prendere in considerazione quando dai concetti generali si passerà alla pratica.
La scelta è caduta sulla famigliaST6 della ST-Microelectronics basata su una CPU a 8 bit: sviluppata per applicazioni nel campo auto, in quello industriale e nel settore delle telecomunicazioni, la famigliaST6 consente di realizzare numerosi dispositivi elettronici nella fascia bassa e medio-bassa.
Per impieghi più complessi, esiste una famiglia (ST9) con prestazioni più sofisticate; è relativamente facile, conoscendo la famiglia ST6, fare un salto di qualità e passare alla famiglia di rango superiore. Le ragioni della scelta della famiglia ST6 sono molteplici: l'ottimo rapporto prezzo/prestazioni dei dispositivi, la loro notevole diffusione, la disponibilità di un programmatore di basso costo. Analizzando lo schema a blocchi dell'ST6265 notiamo che la CPU indicata come "8 bit core" comunica attraverso un bus bidirezionale con tutte le risorse disponibili del chip. Tra di esse distinguiamo le principali, ovvero quelle senza le quali il micro non potrebbe funzionare, che sono la memoria ROM, la memoria RAM e le interfacce di ingresso e uscita contraddistinte dalla sigla "Port".

➢ Input/Output

Nel microcontrollore ST6265, ad esempio, sono disponibili tre blocchi di interfaccia ingresso/uscita, definiti anche come porte di I/O (Input/Output), siglate rispettivamente: Port A, Port B, Port C.
Ogni Port è collegato ad un determinato numero di pin del micro, per la precisione il Port A è collegato a 8 pin siglati da PA0 a PA7, il Port B è connesso ai pin da PB0 a PB7, il Port C è invece collegato a solo 5 pin siglati PC0, PC1, PC2, PC3, PC4.
Tutti i pin collegati ai Port vengono indicati con il termine "linea di I/O (ingresso/uscita)", il Port A e il Port B dispongono dunque di 8 linee ciascuno, mentre il Port C dispone di 5 linee di I/O.
Ogni singola linea può essere programmata per funzionare come ingresso o come uscita.

Ad esempio se colleghiamo un pulsante ad un pin di I/O del micro dovremo programmare questo pin come ingresso, al contrario se vi colleghiamo un relè il pin sarà settato come uscita.
Concludendo, ogni linea di I/O viene nominata ingresso se l'informazione transita dal mondo esterno alla CPU, oppure di uscita se i dati si spostano dalla CPU al mondo esterno.
I microcontrollori ST6260 e ST6265 dispongono di funzioni molto avanzate per ogni linea di I/O; sono infatti disponibili ben 5 opzioni per ogni pin d'ingresso:

1. ingresso normale;
2. ingresso con resistore di "pull-up";
3. ingresso di interruzione;
4. ingresso analogico;
5. ingresso seriale.

Per un pin di uscita sono invece disponibili 4 diversi tipi di funzionamento:

1. uscita "push-pull";
2. uscita "open-drain";
3. uscita ad alta corrente;
4. uscita seriale.

- **Il programma**

Il blocco indicato come "*User program ROM*" indica la parte del chip atta a contenere il programma.
Se, ad esempio, utilizziamo un ST6265 e lo alimentiamo applicando una tensione tra i pin Vdd e Vss potremo verificare con un tester che tutte le linee di I/O rimangono nella condizione ad alta impedenza, non succede praticamente nulla.
Il micro in oggetto dispone si di una memoria programma ma essa risulta vuota, ovvero non contiene alcun comando che può essere interpretato dalla CPU. Noi abbiamo acquistato la parte hardware che consiste fisicamente nell'integrato siglato ST6265 ma per funzionare è necessario procedere alla sua programmazione.
Da queste affermazioni nasce il concetto di programma software che rappresenta la sequenza dei comandi che la CPU deve processare ed eseguire.
Il software viene "scritto" dall'utente in funzione di ciò che si vuole far fare al micro e successivamente trasferito nella memoria programma che può essere di tipo PROM o EPROM.

> *I micro dotati di memoria programma di tipo PROM vengono definiti OTP (One Time Programmable) ovvero programmabili una sola volta e sono usati per produrre piccole o medie serie di integrati atti a svolgere sempre la stesse funzioni; se le quantità sono considerevoli (dai 5-10 mila pezzi in su), la programmazione può essere effettuata dalla stessa casa costruttrice (in questo caso si parla di micro "mascherati"); al contrario, i micro con memoria programma di tipo EPROM, possono essere cancellati e programmati più volte, vengono perciò utilizzati prevalentemente in fase di messa a punto del software.*

In ogni caso, sia per i micro di tipo PROM che di tipo EPROM, per poter lavorare è necessario disporre di un computer e di un appropriato sistema di sviluppo denominato "Starter Kit".
I comandi da impartire alla CPU vengono elaborati e simulati dapprima a computer e in un secondo tempo trasferiti nel chip tramite programmazione.

Per ora ci limitiamo a ricordare che ogni diversa famiglia di microcontrollori necessita purtroppo di un proprio sistema di sviluppo.
Ad esempio l'ST626X Starter Kit programma la sottofamiglia ST626X di microcontrollori ST-Microelectronics, mentre l'ST6220 Starter Kit gestisce le sottofamiglie ST621X e ST622X, e così di seguito l'M68HC705KICS Starter Kit programma il micro 705H1 della Motorola, il TMS320C5X Starter Kit programma il TMS320XX della Texas Instruments, ecc.
E' dunque di primaria importanza sapere esattamente qual'è il micro più adatto alle nostre esigenze e di conseguenza orientarsi sul relativo Starter Kit.

1.7 La famiglia ST6

Il mercato dei micro è in notevole espansione e tutte le principali case di semiconduttori hanno nel proprio catalogo dei microcontrollori, dalla ST-Microelectronics alla Texas Instruments, dalla Toshiba all'Hitachi, dalla Microchip alla Motorola.
Sapersi districare in questo mercato alla ricerca del micro più appropriato non è facile. Ogni micro nasce con una propria filosofia e presenta rispetto ai concorrenti sia pregi che difetti. Una ditta che prevede l'utilizzo di un micro per produrre migliaia di pezzi deve sicuramente affrontare con cura la scelta del modello più adatto, al contrario una piccola o media industria elettronica dovrebbe basarsi su altri criteri, come ad esempio la semplicità di programmazione, il basso costo del sistema di sviluppo ed infine la gamma di modelli disponibili.
Soffermiamoci su quest'ultima affermazione introducendo il concetto di "*famiglia di microcontrollori*", termine con cui si indicano micro diversi che usano lo stesso software di programmazione.
Ad esempio, la famiglia ST6 è composta da 3 sottofamiglie che sono:

1. ST621X/ST622X;
2. ST624X;
3. ST626X.

Ad ognuna di queste sottofamiglia appartengono vari modelli di microcontrollore; ad esempio della famiglia ST626X fanno parte i tipi ST6260 e ST6265.
Ogni micro indicato è disponibile sia con memoria programma di tipo EPROM che di tipo OTP. Alla famigliaST6 appartengono svariati microcontrollori che pur diversi l'un dall'altro condividono lo stesso software.
Si deduce che imparando ad utilizzare uno qualsiasi dei micro indicati si può facilmente e rapidamente passare alla programmazione di un altro micro ST6.
La famiglia ST6 utilizza una CPU ad 8 bit e consente la realizzazione di numerosi dispositivi elettronici.
Per impieghi più complessi è possibile e facile, se già si conosce l'ST6, orientarsi verso la più sofisticata famiglia ST9.
Tornando alle motivazioni che ci hanno portato alla scelta dei micro ST6 possiamo affermare che esse sono molteplici: le principali possono essere individuate nell'ottimo rapporto prezzo/prestazione, nella semplicità di programmazione, nalla vastità di modelli disponibili, nella robustezza dei micro poiché nati per il mercato automotive, nella loro notevole diffusione e infine nella disponibilità di un programma di basso costo.
La famiglia ST6 è stata progettata per soddisfare in modo facile e flessibile un grande numero di esigenze; internamente infatti il micro della ST-Microelectronics dispone di notevoli risorse sia in termini elaborativi che di completezza delle periferiche implementate,

che ci consentiranno di realizzare applicazioni anche complesse utilizzando pochi componenti esterni.

I micro ST6 comprendono diversi modelli che possiamo dividere in tre grosse categorie:

1. versioni EPROM utilizzabili in fase di sviluppo del software;
2. versioni OTP per la produzione in serie;
3. versioni ROM per produzioni di grossi quantitativi.

- Le versioni EPROM sono contraddistinte dalla lettera E (ST62EXX);
- le versioni OTP dalla lettera T (ST62TXX),;
- le versioni ROM dall'assenza di lettere (ST62XX).

1.8 La sottofamiglia ST621X/ST622/X

Possiamo ora affermare una ulteriore suddivisione in funzione della memoria disponibile:

1. ST62X10 e ST62X15, memoria programma 1828 bytes;
2. ST62X20 e ST62X25, memoria programma 3876 bytes

Dividiamo ora la famiglia ST6 in relazione agli I/O:

1. ST62X10 e ST 62X20, 12 linee I/O;
2. ST62X15 e ST 62X25, 20 linne I/O.

Suddivisione in base al tipo di WatchDog:

1. /HWD, indica che il WatchDog interno è di tipo Hardware;
2. /SWD, indica che il WatchDog interno è di tipo Software.

1.9 La sottofamiglia ST626X

Questi chip sono fisicamente identici ai precedenti e condividono le stesse istruzioni software poiché appartengono alla stessa famiglia, sia la struttura complessiva che le nozioni di base hardware sono esattamente le stesse.

In pratica l'ST6260 rappresenta l'evoluzione dell'ST6220 che a sua volta è l'evoluzione dell'ST6210.

Allo stesso modo l'ST6265 deriva dall'ST6225 che a sua volta deriva dall'ST6215.

Per evitare confusioni facciamo riferimento alla tabella comparativa della famiglia ST6 riportata di seguito. Qui possiamo effettuare subito due grosse distinsioni rappresentate dal tipo di contenitore impiegato, l'ST6210, l'ST6220 e l'ST6260 dispongono di package a 20 pin, mentre l'ST6215, l'ST6225 e l'ST6265 hanno un contenitore a 28 piedini. Da ciò deriva la prima differenza cioè il numero di linee di I/O che i micro possono gestire. La seconda differenza che possiamo riscontrare è la diversa capacità di memoria programma. In generale possiamo ricordare la seguente tabella di verità per comprendere subito con quale modello abbiamo a che fare, se trasformiamo in ST62 A B C la sigla di un qualsiasi microcontrollore indicato, possiamo affermare che: la posizione A coincide con il tipo di memoria programma ovvero sarà una E se di tipo EPROM o una T se di tipo OTP, la posizione B indica la sottofamiglia di appartenenza 1 o 2 oppure 6, la posizione

C coincide con il contenitore cioè 20 pin se 0, oppure 28 pin se 5. Ad esempio, un micro siglato ST62E25 è un ST6 in versione EPROM appartenente alla sottofamiglia 2 e in contenitore a 28 pin. La sottofamiglia 1 e 2 sono, a parità di contenitori, pin-to-pin compatibili e si differenziano tra loro solo per la capacità di memoria programma. La sottofamiglia 6 (ST6260 e ST6265) non è, a parità di contenitori, pin-to-pin compatibile con

i 4 modelli precedenti e dispone di maggiori risorse: 64 byte in più di memoria RAM, 128 byte di memoria EEPROM, un timer autoricaricabile e infine una periferica seriale.

- ***Tabella alcuni micro famiglia ST6***

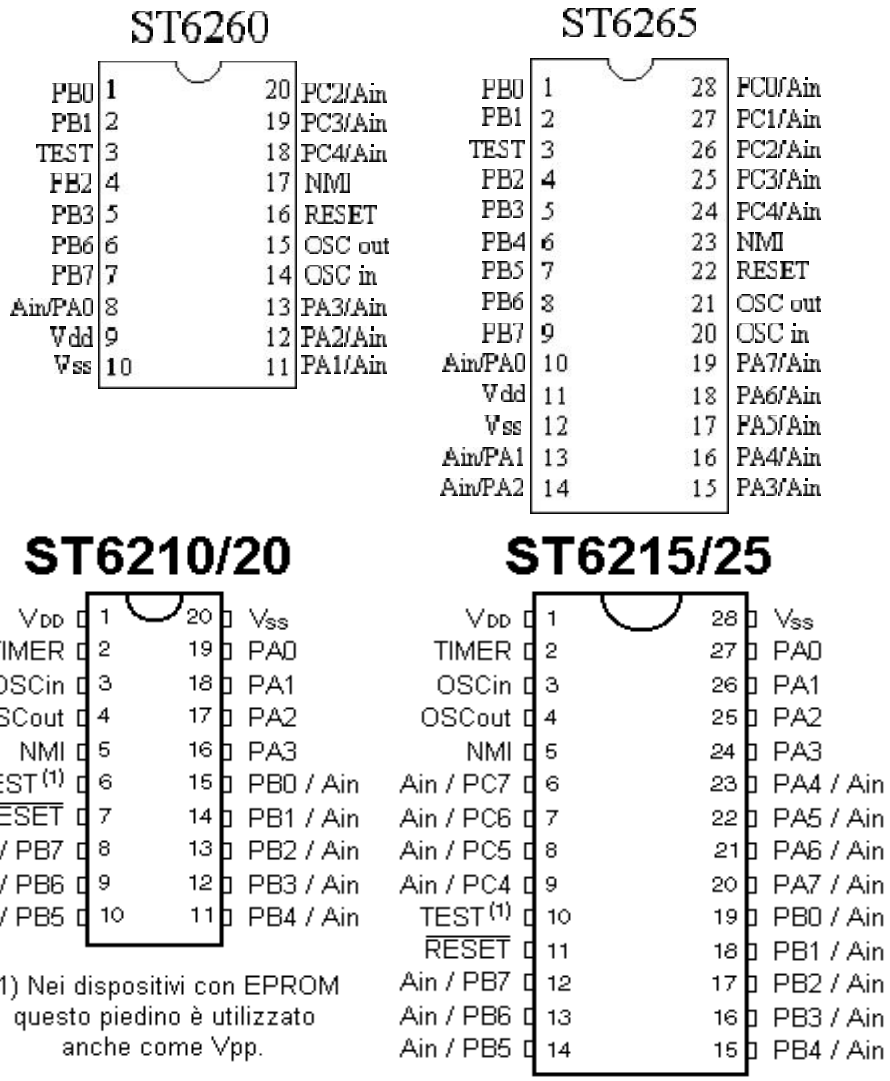
SIGLA	ROM	RAM	EEPROM	I/O	A/D	LED	TIMER	ATIMER	SPI	Contenitore
ST6210	2	64	-	12	8	4	1	-	-	DIP20
ST6215	2	64	-	20	16	4	1	-	-	DIP28
ST6220	4	64	-	12	8	4	1	-	-	DIP20
ST6225	4	64	-	20	16	4	1	-	-	DIP28
ST6260	4	128	128	13	7	6	1	1	1	DIP20
ST6265	4	128	128	21	13	8	1	1	1	DIP28

- ROM: indica la memoria programma ed è espressa in KB;
- RAM: indica i byte di memoria dati;
- EEPROM: byte disponibili di memoria dati non volatile;
- I/O: linee di I/O;
- A/D: linee analogiche;
- LED: linee ad alta corrente (20 mA)
- TIMER: numero TIMER ad 8 bit;
- ARTIMER: numero TIMER ad 8 bit autoricaricabile;
- SPI: interfaccia seriale;
- CONTENITORE: numero piedini.

La tabella sopra riportata indica una serie di microcontrollori che si differenziano l'uno dall'altro a causa delle diverse risorse disponibili, mentre, al contrario, tutti hanno in comune lo stesso set di istruzioni.
Da qui nasce la definizione di famiglia ovvero quell'insieme di micro che condividono lo stesso software, lo stesso assemblatore, e lo stesso algoritmo di programmazione, pur avendo risorse interne diverse.
Se ne deduce che imparando a lavorare con uno qualsiasi di questi chip si può facilmente e rapidamente passare ad un altro modello della stessa famiglia.

2.1 Descrizione dei piedini

Piedinatura micro ST6260 – ST6265 – ST 6210/20 – ST 6215/25



1- Alimentazione (VDD e VSS) (pin:11,12 - ST62x65)
Ingressi di alimentazione (VDD positivo e VSS negativo), su questi piedini va applicata una tensione stabilizzata compresa tra i 3 e i 6 volt (tipicamente 5 volt) e, normalmente, un condensatore elettrolitico di disaccoppiamento con capacità compresa tra 0,1 e 1 µF (fig.1).

2 – Clock (OSCin e OSCout) (pin:20,21 - ST62x65)
Questi piedini sono collegati all'oscillatore interno del micro che serve a generare il *clock* di sistema. Esistono 3 possibili configurazioni:

1. Quando si seleziona l'opzione "*Quartz/Ceramic resonator*", tramite l'Option Byte, è possibile collegare un quarzo, un risonatore ceramico o un segnale di clock esterno con frequenza compresa tra i 2 e gli 8 MHz a questi due piedini. Nel caso più comune, quello del quarzo (fig.1), vanno collegati anche due condensatori ceramici di capacità compresa tra 12pF e 22 pF fra i piedini e la massa.
2. Selezionando invece l'opzione "*RC oscillator*" è sufficiente collegare fra il piedino OSCout e la massa una resistenza (OSCin:NC). Sui microcontrollori sprovvisti di Option Byte l'impostazione predefinita è "Quartz/Ceramic resonator".
3. onda quadra stabile di frequenza F_{ck}=8 MHz, applicata direttamente al pin OSCin (OSCout:NC).

3-RESET (pin:22 - ST62x65)

Quando questo piedino viene cortocircuitato a massa si causa un reset del microcontrollore facendo ripartire il programma dall'inizio.

Normalmente va mantenuto a livello logico alto (VDD) tramite una resistenza da 100 Kohm collegata al positivo e un condensatore elettrolitico da 1 µF collegato a massa, questa rete RC provoca un reset automatico ogni volta che il circuito viene alimentato, garantendo così il corretto funzionamento del micro. Nei micro della famiglia ST6 esistono 3 differenti condizioni di RESET:

1. Rete RC esterna (fig.1);
2. comando generato dal WatchDog (dall'interno); quando si azzera il micro va in RESET;
3. POR (Power On Reset), il segnale si attiva al momento della prima alimentazione.

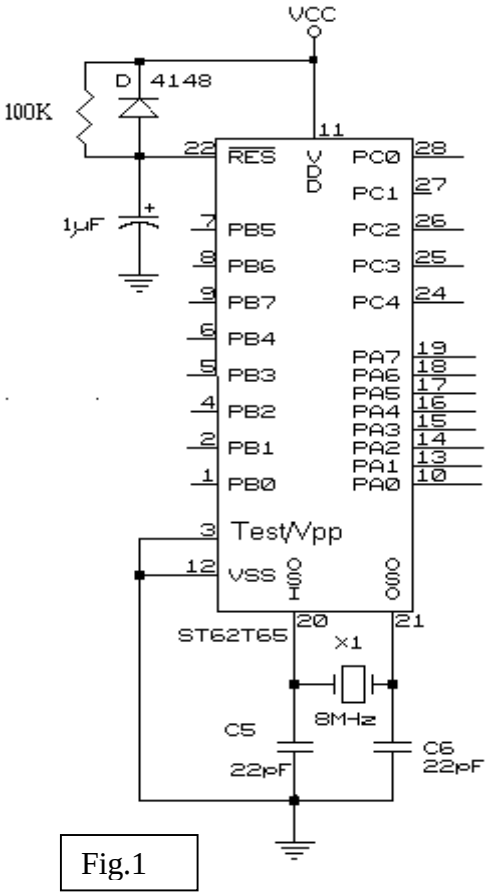


Fig.1

Per quest'ultima condizione (POR) occorre premettere che al momento in cui viene fornita tensione sul piedino VDD, occorre un certo periodo di tempo prima che la tensione si stabilizza, quando l'alimentazione, sale ad un livello sufficiente (1,2V), l'oscillatore parte a funzionare e viene generato un ritardo interno fino a che l'oscillatore non è a regime.

La condizione di RESET generata da POR sussiste fino a 2048 cicli dopo l'entrata a regime dell'oscillatore; fino a quel momento le porte del micro sono configurate tutte come ingressi con resistenza di *Pull-Up* interna.

In fig.2 i diagrammi temporali.

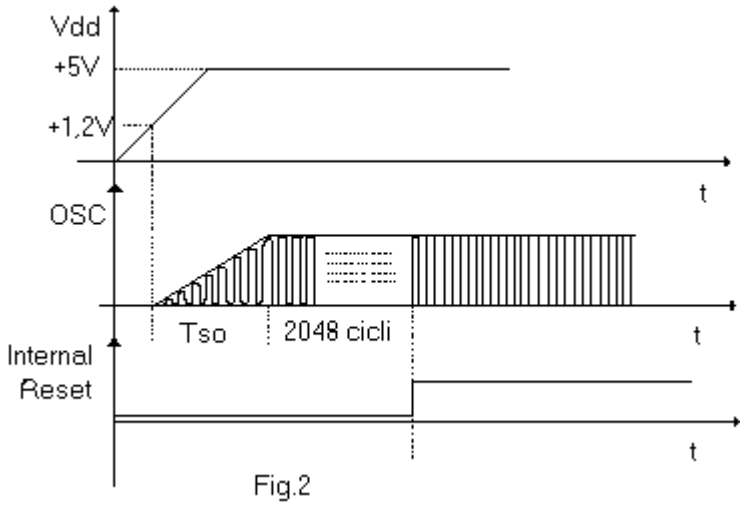


Fig.2

4-TEST/Vpp (pin:3 - ST62x65)

Durante il funzionamento normale questo piedino va mantenuto a massa (VSS), sebbene ciò avvenga in automatico (tramite una resistenza di *pull-down* interna al micro), è tuttavia consigliabile collegarlo fisicamente al negativo di alimentazione onde evitare il danneggiamento del programma contenuto in memoria. Soltanto durante la programmazione

Starter-Kit), il piedino si trova a livello logico 1 (VDD) e viene portato a +12,5 volt (VPP) dal programmatore per trasferire i dati all'interno del micro.

5- NMI (pin:23 - ST62x65):
Ingresso asincrono associato alla gestione dell'*Interrupt non mascherabile*. (fig.3)
Applicando un fronte di discesa a questo piedino (segnale negativo), si obbliga il microcontrollore a sospendere lo svolgimento del programma principale (indipendentemente da quello che sta facendo), e ad eseguire una *subroutine* (sottoprogramma) opportunamente programmata.
Se non utilizzato, questo piedino va collegato al positivo d'alimentazione (VDD); nei microcontrollori dotati di Option Byte è comunque possibile attivare la resistenza interna di *pull-up*.

6- AR TIMER (autoricaricabile**)(ARTIMin/PB6, ARTIMout/PB7) (pin:8, 9 - ST62x65):**

- ARTIMin (pin 8) (fig.3): questo pin viene gestito via software, può essere utilizzato come ingresso per far partire il conteggio della periferica AR TIMER;
- ARTIMout (pin 9) (fig.3): questo pin viene gestito via software, può essere utilizzato come uscita per attivare un dispositivo allo scadere del tempo impostato nel AR TIMER. Su questo piedino, è anche possibile prelevare dei segnali ad onda quadra con frequenze impostate da software (utilissimo per creare dei generatori PWM). Anche per questo piedino è possibile attivare la resistenza interna di *pull-up* utilizzando l'Option Byte, sui microcontrollori sprovvisti di questa possibilità la resistenza (quando necessaria) deve essere montata all'esterno.

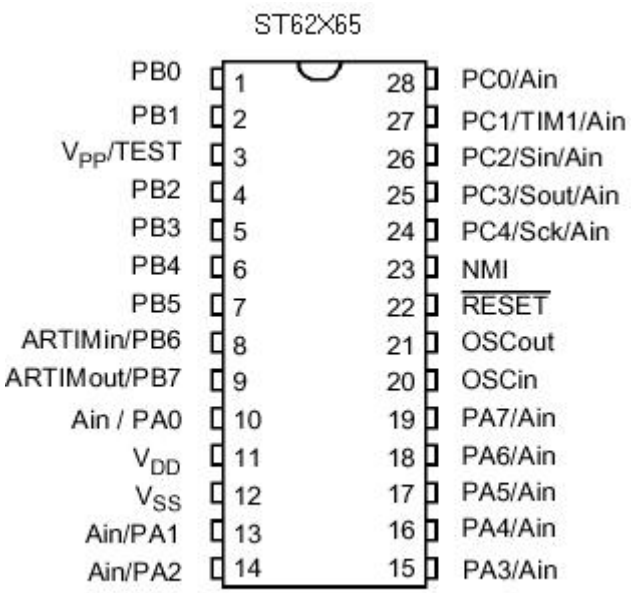


Fig.3

7- TIMER (PC1/TIM1/Ain) (pin:27 - ST62x65):
Questo piedino (fig.3), come suggerisce il nome, è collegato al contatore interno denominato per l'appunto *Timer*. Il suo uso dipende da come viene configurato via software, può essere utilizzato come ingresso per far partire il conteggio del timer o come uscita per attivare un dispositivo allo scadere del tempo stabilito. Su questo piedino, è anche possibile prelevare dei segnali ad onda quadra con frequenze impostate da software (utilissimo per creare dei generatori PWM). Anche per questo piedino è possibile attivare la resistenza interna di *pull-up* utilizzando l'Option Byte, sui microcontrollori sprovvisti di questa possibilità la resistenza (quando necessaria) deve essere montata all'esterno

8- I/O (PORT A, 8 linee - PORT B, 8 linee - PORT C, 5 linee):

Le 21 linee di I/O (fig.4) sono utilizzate dal microcontrollore per dialogare con il mondo esterno. Sono configurabili via software in sei modi diversi ed in maniera indipendente l'uno dall'altro (anche più volte durante lo svolgimento del programma). I possibili modi di funzionamento sono:

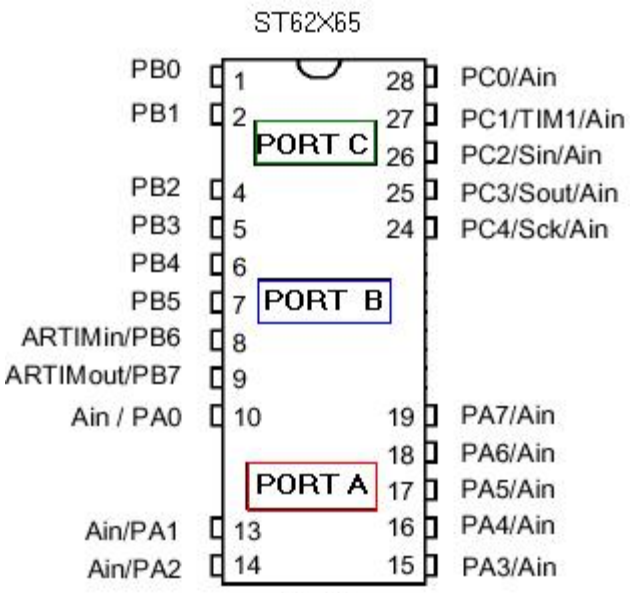


Fig. 4

1. ingresso con resistenza di pull-up; (default);
2. ingresso normale senza pull-up;
3. ingresso con pull-up e interruzione;
4. ingresso analogico (collegato al convertitore A/D) (solo alcune linee);;
5. uscita *open-drain*.
6. uscita *push-pull;*

Nel capitolo “Comunicazione con il mondo esterno”, verrà descritto in dettaglio il funzionamento e la configurazione delle linee I/O.

9- I/O Seriale (PC2/Sin, PC3/Sout, PC4/Sck) (pin:26,25,24 ST62x65)

Queste 3 linee (fig.5) (Sin, Sout, Sck) permettono di trasferire i dati serialmente.

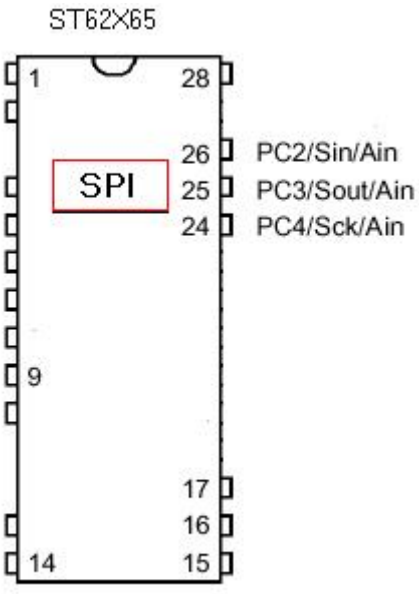


Fig. 5

Osservando lo schema a blocchi interno (pag.4), questi pin fanno capo ad una periferica denominata SPI, i dati in uscita vengono presentati al mondo esterno attraverso il pin Sout (pin:25), mentre i dati in ingresso sono letti sul pin Sin (pin:26). La comunicazione viene sincronizzata con il clock interno, oppure dal pin siglato Sck (pin:24).

La comunicazione seriale viene utilizzata per il trasferimento dati tra due micro oppure tra un micro ed un Personal Computer.

Nel capitolo “Comunicazione con il mondo esterno”, verrà descritto in dettaglio il funzionamento della periferica seriale

3.1 CPU (core)

La periferica più importante contenuta all'interno di un ST6 è la CPU (detta anche "*Core*"), essa è in pratica il cervello di tutto il sistema, è quella che, comunicando con le varie periferiche interne attraverso dei canali chiamati "*bus*", si fa carico di eseguire il programma ed elaborare i dati

Il suo compito principale è di **leggere**, **interpretare** ed **eseguire** le istruzioni presenti nella memoria programma.

Per poter svolgere questi compiti, la CPU deve essere collegata ad altre periferiche, da un buffer bidirezionale.

Le periferiche principali sono:

1. Program Counter (PC) e 6 livelli di Stack
2. ALU
3. Flag
4. Controller

In fig. 1 lo schema a blocchi della CPU

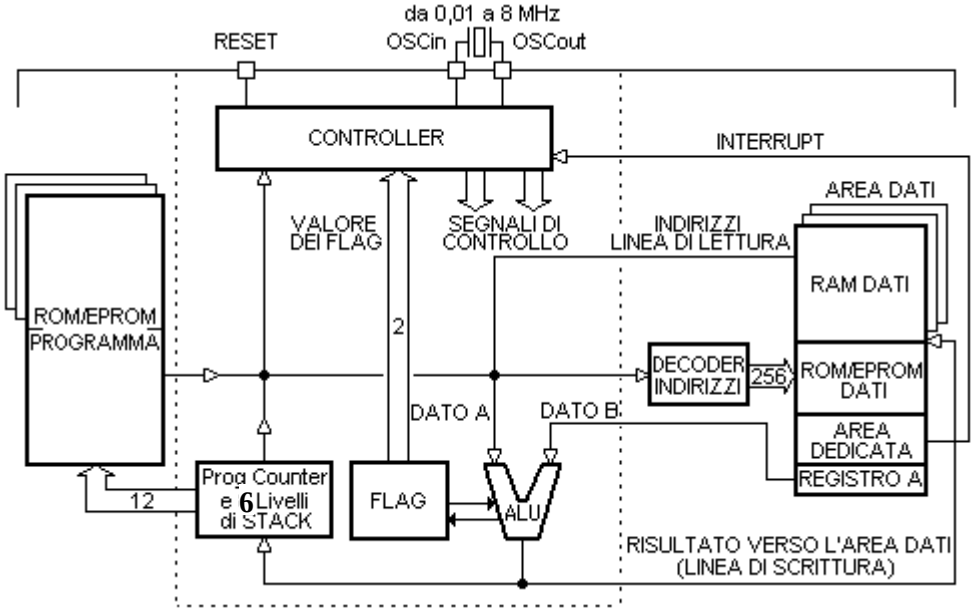


Fig.1- *Schema a blocchi della CPU.*

1. Program Counter (PC) e 6 livelli di Stack

Il PC contiene l'indirizzo del byte di memoria programma, in cui è scritta l'istruzione da eseguire, esso risulta il componente che tiene il conteggio delle istruzioni che la CPU deve processare una dopo l'altra.

Il PC viene salvato nello stack quando la CPU abbandona la normale sequenza di istruzioni, e si sposta in un'altra parte della memoria programma (esempio gestione subroutine di interrupt).

Nei micro della famiglia ST6 esistono 6 livelli di stack, ciò significa, che si possono avere un numero massimo di spostamenti pari a 6, quantità più che sufficiente anche per le applicazioni più complesse.

2. ALU (**Arithmetic Logic Unit**)

Tutte le operazioni matematiche vengono svolte dall'unità ALU alla quale spetta anche l'aggiornamento del Carry flag e dello Zero flag.

3.Flag

L' ST6 dispone di tre gruppi di due flag, Carry (C) e Zero (Z). (fig.2)
I tre gruppi possono essere associati a tre modi di funzionamento:

- Normal mode (CN, ZN);
- Interrupt mode (CI, ZI)
- NMI mode (CNMI, ZNMI)

I flag Zero e Carry sono due registri ad 1 bit che vengono scritti dall' ALU al termine di un'operazione matematica, e che possono essere letti dal programma per sapere se due numeri sono uguali o diversi, o se un numero è negativo o positivo. Ad esempio il flag di Carry viene settato ad 1 se l'operazione genera un riporto.

4.Controller

All'interno dell'ST6 la sequenza degli eventi o meglio la velocità con cui questi avvengono è regolata dal clock di sistema, il controllore di sistema ne gestisce i segnali e li comunica alla CPU.
Un altro compito del controller è anche quello di ricevere eventuali segnali di reset che possono arrivare da più fronti (rete RC esterna, azzeramento del WATCHDOG, il PORT).

3.2 I Registri della CPU:

Il Core dei micro ST6 è dotato di 5 registri ad 8 bit e tre gruppi di due flag. Questi sono mostrati nella figura a lato (fig.2):

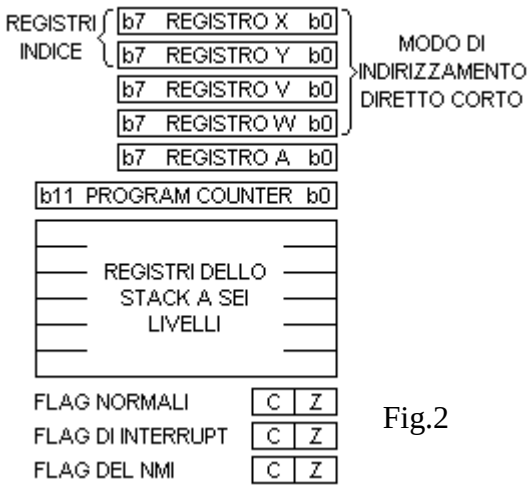


Fig.2

1. **ACCUMULATORE-REGISTRO A**
2. **REGISTRI X e Y**
3. **REGISTRI V e W**

1. **REGISTRO ACCUMULATORE** (registro A): l'accumulatore è il registro più importante all'interno della CPU e quasi tutti i comandi utilizzano per lo svolgimento questo registro.
2. **REGISTRI X e Y**: vengono definiti anche registri indiretti poiché il loro compito principale è quello di indirizzare in modo indiretto la memoria dati.
3. **REGISTRI V e W**: i registri V e W vengono detti " short register " e sono utilizzati per l'indirizzamento diretto della memoria dati.

Tutti i registri sono ad 8 bit, ad ognuno corrisponde una locazione RAM.

TABELLA REGISTRI DELLA CPU

REGISTRI	ABBREVIAZIONE	LOCAZIONE RAM ST 6260 - ST 6265
ACCUMULATORE	A	0FFh
X REGISTER	X	080h
Y REGISTER	Y	081h
V REGISTER	V	082h
W REGISTER	W	083h

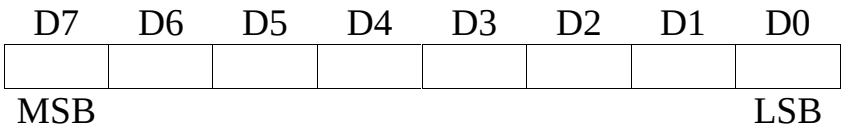
4.1 Memorie e Registri

L'area di memoria è suddivisa in 2 zone ben distinte:

1. ROM / EPROM ► Memoria programma
2. RAM / EEPROM ► Memoria dati. e registri

- La memoria ROM/EPROM contiene il programma vero e proprio, ovvero la sequenza di istruzioni che il " core " deve processare una dopo l'altra.
- La memoria RAM contiene i dati del programma, i registri ed altre variabili.

Tutte le locazioni sono ad 8 bit (1 byte).



4.2 Memoria Programma (ROM/EPROM)

Tale memoria viene gestita dal Program Counter, i microcontrollori ST 622X e ST 626X presentano una memoria programma di 4Kbyte (4096 locazioni da 8 bit).
Questa memoria è un insieme di celle della capacità di 1 byte, ad ogni cella viene associato un numero (esadecimale: 0000h-0FFFh) che ne identifica la posizione, tale numero viene denominato indirizzo o locazione.
In figura è riportata la mappa della memoria programma.

MAPPA MEMORIA PROGRAMMA

0000h 007Fh	RESERVED	
0080h		0080h Inizio programma
	USER PROGRAM MEMORY 3872 BYTES	
0F9Fh		0F9F Ultima locazione riservata al programma
0FA0h 0FEFh	RESERVED	
0FF0h 0FF7h	INTERRUPT VECTOR	
0FF8h 0FFBh	RESERVED	
0FFCh 0FFDh	NMI VECTOR	
0FEEh 0FFFh	USER RESET VECTOR	

La memoria programma è suddivisa in 3 parti:

1. **Riservata alla sola CPU: (0000h-007Fh; 0FA0h-0FEFh; 0FF8h-0FFBh):** Questa parte è riservata alla CPU, non può essere né letta né scritta.
2. **Programma (0080h-0F9Fh):** questa zona (3872 bytes: ST626X) contiene il programma vero e proprio. Quest'area viene letta dal Program Counter della CPU e scritta dalla scheda di programmazione dello Starter Kit. Durante la fase di sviluppo del software si attribuisce alla prima istruzione la locazione 0080h, mentre l'ultima istruzione possibile sarà allocata alla 0F9Fh.
3. **Vettori di Interrupt (0FF0h-0FF7h; 0FFCh-0FFFh):** questa zona contiene i vettori di interrupt, spazio di memoria il cui contenuto viene letto dalla CPU solo durante un interrupt; mentre le normali istruzioni vengono processate una dopo l'altra, le istruzioni contenute nei vettori vengono eseguite in tempo reale dalla CPU quando avviene una precisa richiesta hardware. I vettori disponibili nei micro sT626X, sono sei e ognuno di essi occupa uno spazio di due byte. Ogni vettore viene indicato con #0, #1, #2, #3, #4 e vettore di reset

In tabella la mappa dei vettori di interrupt per ST626X

Sorgente di interrupt	Vettore associato	Indirizzo del vettore (Hex)
Pin NMI	Vettore #0	FFCh - FFDh
Pin Port A e B	Vettore #1	FF6h – FF7h
Pin Port C e SPI	Vettore #2	FF4h – FF5h
Periferica AR-TIMER	Vettore #3	FF2h – FF3h
Periferica ADC – TIMER	Vettore #4	FF0h – FF1h
Pin RESET	Vettore di RESET	FFEh - FFFh

1. **Vettore #0**: le istruzioni contenute nel vettore #0 vengono eseguite quando la causa dell'interrupt è un segnale presente sul pin d'ingresso NMI
2. **Vettore #1:** vettore associato alle periferiche di ingresso/uscita Port A e Port B.
3. **Vettore #2:** il contenuto vettore #2, viene letto quando la causa dell'interrupt è la periferica Port C di I/O oppure la periferica seriale.
4. **Vettore #3:** il contenuto vettore #3 viene processato allo scadere del tempo impostato nel TIMER autoricaricabile (AR-TIMER).
5. **Vettore #4**: Il contenuto del vettore #4 viene letto quando scade il tempo impostato nel TIMER 1, oppure al termine della conversione analogica/digitale (ADC).
6. **Vettore RESET**: il contenuto del vettore RESET viene letto dalla CPU all'atto della prima accensione o quando viene prelevato uno 0 logico sul piedino di RESET.

4.3 Memoria Dati (RAM/EEPROM)

I modelli ST626X, dispongono di 128 byte di RAM e 128 byte di EEPROM. La memoria è composta da 256 locazioni, ognuna capace di contenere 1 byte di dato e contraddistinta da un preciso indirizzo.(in tabella la mappa della memoria dati).

La memoria dati (RAM/EEPROM) può essere suddivisa in quattro aree distinti:

DATA RAM (084h- 0BFh): composta da 60 byte che vengono utilizzati dalla CPU per la memorizzazione temporanea di dati e variabili.

REGISTRI (0C0h–0FFh): celle di memoria dati, fisicamente collegate a specifiche sezioni hardware (es.:registro indice X:locazione 080h)

ZONA RISERVATA ALLA CPU: non può essere né letta né scritta, non è gestibile dall'utente. (0C3h, 0CAh, 0CBh, 0CfF, 0DEh, 0DFh, 0E3h-0E7h, 0Ebh-0FEh)

DATA ROM WINDOWS (040h-07Fh): serve per prelevare alcune istruzioni depositate nella memoria programma in caso di particolari operazioni da far svolgere al micro.

Table 2. ST62T55C, ST62T65C ST62E65 Data Memory Space

RAM and EEPROM	000h 03Fh
DATA ROM WINDOW AREA	040h 07Fh
X REGISTER	080h
Y REGISTER	081h
V REGISTER	082h
W REGISTER	083h
DATA RAM 60 BYTES	084h 0BFh
PORT A DATA REGISTER	0C0h
PORT B DATA REGISTER	0C1h
PORT C DATA REGISTER	0C2h
RESERVED	0C3h
PORT A DIRECTION REGISTER	0C4h
PORT B DIRECTION REGISTER	0C5h
PORT C DIRECTION REGISTER	0C6h
RESERVED	0C7h
INTERRUPT OPTION REGISTER	0C8h*
DATA ROM WINDOW REGISTER	0C9h*
RESERVED	0CAh 0CBh
PORT A OPTION REGISTER	0CCh
PORT B OPTION REGISTER	0CDh
PORT C OPTION REGISTER	0CEh
RESERVED	0CFh
A/D DATA REGISTER	0D0h
A/D CONTROL REGISTER	0D1h
TIMER PRESCALER REGISTER	0D2h
TIMER COUNTER REGISTER	0D3h
TIMER STATUS CONTROL REGISTER	0D4h
AR TIMER MODE CONTROL REGISTER	0D5h
AR TIMER STATUS/CONTROL REGISTER1	0D6h
AR TIMER STATUS/CONTROL REGISTER2	0D7h
WATCHDOG REGISTER	0D8h
AR TIMER RELOAD/CAPTURE REGISTER	0D9h
AR TIMER COMPARE REGISTER	0DAh
AR TIMER LOAD REGISTER	0DBh
OSCILLATOR CONTROL REGISTER	0DCh*
MISCELLANEOUS	0DDh
RESERVED	0DEh 0DFh
SPI DATA REGISTER	0E0h
SPI DIVIDER REGISTER	0E1h
SPI MODE REGISTER	0E2h
RESERVED	0E3h 0E7h
DATA RAM/EEPROM REGISTER	0E8h*
RESERVED	0E9h
EEPROM CONTROL REGISTER	0EAh
RESERVED	0EBh 0FEh
ACCUMULATOR	0FFh

* WRITE ONLY REGISTER

4.4 RAM and EEPROM (000h – 03Fh)

La zona compresa tra 000h - 03Fh è composta da pagine da 64 Byte (2 pagine EEPROM e 1 pagina RAM).

Per usare la EEPROM e gli altri 64 Byte di RAM è necessario ricorrere al registro DRBR (DATA RAM / EEPROM BANK REGISTER) che si trova all'indirizzo 0E8h (registro di sola scrittura).

4.5 Registro DRBR (indirizzo:0E8h)

(DATA RAM / EEPROM BANK REGISTER (solo scrittura)

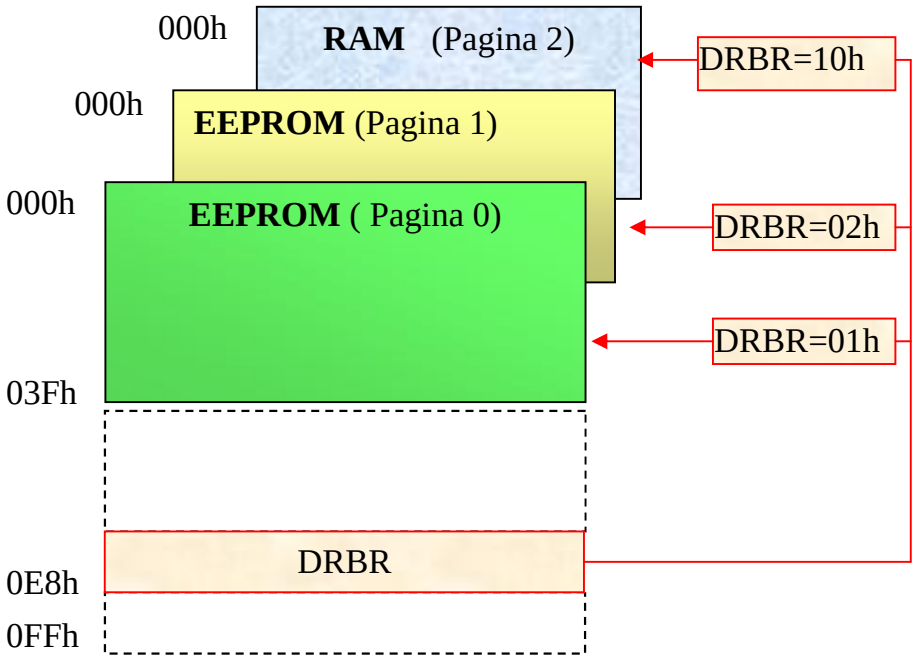
7							0
-	-	-	DRBR4	-	-	DRBR1	DRBR0

BIT 7-6-5: non usati
BIT 4: **DRBR4** = selezione pagina 2 di RAM
BIT 3-2: non usati
BIT 1: **DRBR1** = selezione pagina 1 di EEPROM
BIT 0: **DRBR0** = selezione pagina 0 di EEPROM

In questo registro deve essere settato un solo bit alla volta, in altre parole deve essere selezionato un solo banco di memoria alla volta. Per accedere a questo registro non si devono usare le istruzioni SET e RES, ma solo istruzioni LD e LDI.

SELEZIONE DEL BANCO DI MEMORIA RAM / EEPROM

DRBR	ST 6260 / 65	D7	D6	D5	D4	D3	D2	D1	D0
00h	NESSUNO								
01h	PAGINA 0 EEPROM				**0**			**0**	**1**
02h	PAGINA 1 EEPROM				**0**			**1**	**0**
08h	NON DISPONIBILE								
10h	PAGINA 2 RAM				**1**	0		**0**	**0**
ALTRI	NON USARE								



4.6 – Uso pagina 2 RAM (indirizzo:000h – 03Fh)

Per usare il banco di 64 bytes di ram (pag.2) e' sufficiente usare questa istruzione:

```
ldi     drbr,010h          ; attiva pagina 2 RAM
```

A questo punto si può accedere normalmente a questi 64 byte di ram nell'area di memoria che va da **000h** a **03Fh**. All'interno di questa area si possono definire anche delle etichette con la normale direttiva .def

```
spazio .def    000h        ; definisce la variabile spazio
tempo .def     020h        ; definisce la variabile tempo
```

Se nello stesso programma viene usata anche la EEPROM, queste etichette corrisponderanno anche agli indirizzi 00h e 020h della EEPROM

4.7 Lettura della EEPROM (indirizzo:000h – 03Fh)

La lettura della EEPROM avviene nello stesso modo della RAM. E' sufficiente selezionare il banco di EEPROM desiderato, e leggere i dati sempre nell'area 000h-03Fh. Prima però e' necessario rendere attiva la EEPROM, il registro di controllo si chiama EECTL (indirizzo:0EAh)

```
ldi     eecr,00h           ; attiva la memoria EEPROM
ldi     drbr,01h           ; attiva pagina 0 EEPROM
```

Al termine della lettura si può disattivare nuovamente la EEPROM per evitare scritture accidentali e per ridurre il consumo di corrente.

```
ldi     eecr,040h          ; disattiva la memoria EEPROM
```

4.8 Programmazione della EEPROM (indirizzo:000h – 03Fh)

La memorizzazione di dati nella EEPROM richiede una procedura leggermente più complessa, ma comunque si tratta solo di alcune semplici istruzioni. Va detto anche che, a differenza della RAM, la scrittura della EEPROM e' molto più lenta e non può essere fatta un numero illimitato di volte (il costruttore dichiara 1 milione di cicli di scrittura sulla EEPROM).
La programmazione può avvenire in modalità BYTE o in modalità PARALLELA, utile per ridurre i tempi di scrittura nel caso in cui devono essere scritti più bytes contemporaneamente.
Nel capitolo “Memoria EEPROM”, verrà descritta la memorizzazione dati nella memoria EEPROM

4.9 REGISTRI ST626X (35 registri)

Con il termine " registri " si indicano determinate celle di memoria fisicamente collegate ad una specifica sezione hardware.
Nell'st626x, tutti i registri, sia quelli del " CORE " che delle varie periferiche sono collegati all'interno della memoria dati e possono quindi essere gestiti con estrema facilità, come normali celle RAM.
Utilizzeremo i registri per passare dei comandi alla CPU e alla periferiche del micro, oppure leggeremo i registri per apprendere informazioni che ci vengono fornite dalla CPU e dalle periferiche.
I registri disponibili sono **35** ed ognuno è contraddistinto da un numero esadecimale che ne identifica la precisa locazione all'interno della memoria e da una sigla mnemonica. Possiamo raggruppare i registri a seconda della funzione cui sono destinati:

Periferica	**N. registri**
CORE	5
Porte Input/Output	9
Convertitore ADC	2
TIMER 1	3
TIMER autoricaricabile (AR TIMER)	6
Periferica seriale (SPI)	4
Memoria EEPROM	2
Controllo area programma (Data ROM WINDOWS AREA)	1
Opzioni di interruzione	1
Controllo oscillatore	1
WATCHDOG	1
Totale registri	**35**

In tabella il riepilogo di tutti i registri con abbreviazioni e indirizzi.

Periferica	Registro	Abbreviazione	Locazione RAM (Hex)
Core	Accumulator	A	0FF
	X Register	X	080
	Y Register	Y	081
	V Register	V	082
	W Register	W	083
Porte Input/Output	Port A Data Register	DRA	0C0
	Port B Data Register	DRB	0C1
	Port C Data Register	DRC	0C2
	Port A Direction Register	DDRA	0C4
	Port B Direction Register	DDRB	0C5
	Port C Direction Register	DDRC	0C6
	Port A Option Register	ORA	0CC
	Port B Option Register	ORB	0CD
	Port C Option Register	ORC	0CE
ADC	A/D Control Register	ADCR	0D1
	A/D Data Register	ADR	0D0
TIMER 1	TIMER 1 Status Control Register	TSCR1	0D4
	TIMER 1 Counter Register	TCR1	0D3
	TIMER 1 Prescaler Register	PSC1	0D2
AR-TIMER	AR Timer Mode Control Register	ARMC	0D5
	AR Timer Status/Control Register 1	ARSC0	0D6
	AR Timer Status/Control Register 2	ARSC1	0D7
	AR Timer Load Register	ARLR	0DB
	AR Timer Reload/Capture Register	RELCAP	0D9
	AR Timer Compare Register	ARCP	0DA
SPI	SPI Mode Register	SPIMOD	0E2
	SPI Divider Register	SPIDIV	0E1
	SPI Data Register	SPIDSR	0E0
	Miscellaneous	MISC	0DD
EEPROM	Data RAM/EEPROM Register	DRBR	0E8
	EEPROM Control Register	EECTL	0EA
Controllo area prg	Data ROM Window Register	DWR	0C9
Interrupt	Interrupt Option Register	IOR	0C8
Oscillatore	Oscillator Control Register	OSCR	0DC
Watchdog	Digital Watchdog Register	DWDR	0D8

4.10 Data RAM (indirizzo:084h- 0BFh)

All'interno della memoria dati oltre ai numerosi registri di controllo, di cui già abbiamo indicato le sigle e gli indirizzi (vedi tabella), notiamo la presenza dalla locazione 084h alla locazione 0BFh, di uno spazio di 60 Byte denominata DATA RAM.
Questi Byte di memoria RAM che possiamo chiamare con il termine celle rappresentano lo spazio dove il programma andrà a memorizzare i dati e le variabili; possiamo immaginare questo spazio di Byte come una periferica di memorizzazione temporanea.
Con particolari istruzioni software possiamo scrivere in queste celle dati, variabili ed altre informazioni. Con altre istruzioni possiamo invece leggere le informazioni contenute in queste celle.
La memoria RAM presente all'interno del micro ST6 risulta più che sufficiente per soddisfare la maggior parte delle esigenze; all'atto pratico gestiamo queste celle attraverso un nome associato: ad esempio se intendiamo realizzare un programma che conti il trascorrere di 10 secondi, utilizzeremo una cella di RAM in cui di volta in volta andremo a scrivere il numero di secondi trascorsi. Il fatto interessante è che potremo chiamare questa cella " tempo ", oppure "secondi", ovvero nel modo che più ci aggrada.
La memoria RAM è di tipo volatile o temporanea, i dati rimangono registrati fino a quando permane la tensione di alimentazione del micro; l'applicazione specifica della RAM è quindi la memorizzazione di variabili, utilizzeremo invece la memoria ROM o di programma per memorizzare le costanti. Queste ultime vengono definite in fase di programmazione attraverso delle particolari istruzioni denominate " EQUATE " e contraddistinte dalla sigla " EQU ".

Comunicazione con il mondo esterno

5.1 introduzione

Il microcontrollore scambia informazioni con il mondo esterno attraverso degli appositi piedini a cui diamo il nome di porte o linee di Input/Output.
La porta viene detta di Input quando i dati vengono trasferiti dall'esterno alla CPU del micro, viceversa viene definita di uscita quando i dati transitano dalla CPU verso l'esterno (fig.1)

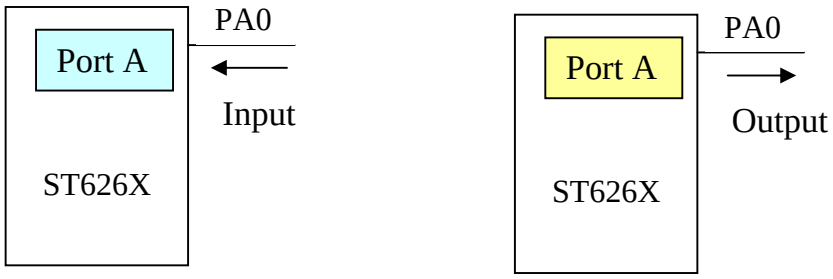


Fig.1

Nel microcontrollore ST6265, ad esempio, sono disponibili tre blocchi di interfaccia ingresso/uscita, definiti anche come porte di I/O (Input/Output), siglate rispettivamente: Port A, Port B, Port C (fig.2).

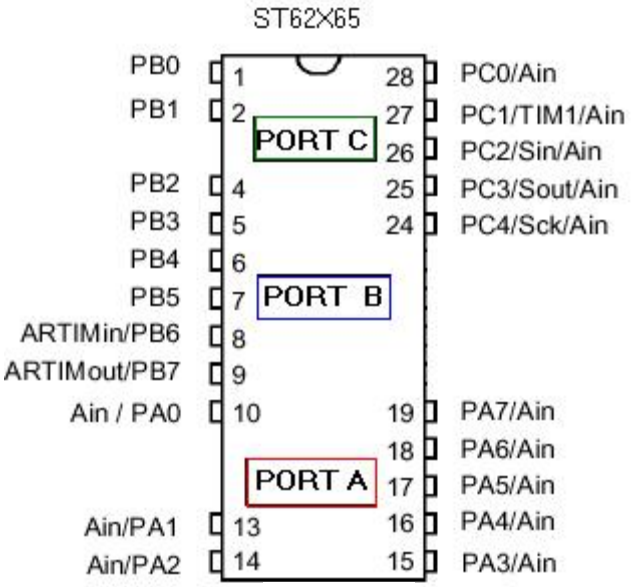


Fig.2

Il microcontrollore ST626x dispone di 21 linee di I/O programmabili, ogni Port è collegato ad un determinato numero di pin del micro:

1. Port A : 8 linee
2. Port B: 8 linee;
3. Port C: 5 linee.

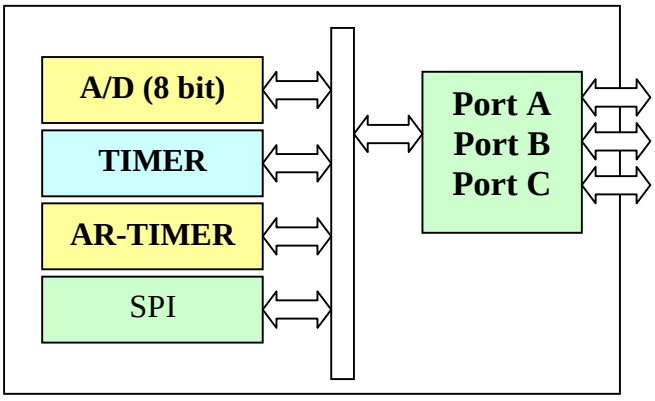


Fig.3

Alcune linee vengono utilizzate per la comunicazione con specifiche sezioni hardware interne (fig.3):

1. Convertitore A/D: 13 linee
2. TIMER: 1 linea
3. AR-TIMER: 2 linee
4. Interfaccia seriale SPI: 3 linee

5.2 Porte Input/Output (I/O)

Le periferiche di I/O contenute all'interno del micro ST6260 e ST6265 sono tre (Port A, Port B, Port C).
Ogni linea può essere programmata da software per funzionare come Input o come Output indipendentemente dallo stato associato alle altre linee, inoltre la funzione di ogni linea può essere variata a piacere in ogni punto del programma.
I micro ST626X dispongono di periferiche aggiuntive, non presenti nelle famiglie inferiori, e di conseguenza le opzioni di alcuni pin di I/O risultano diverse poiché sono diversi i collegamenti tra questi ultimi e le nuove periferiche.
In generale, possiamo affermare che all'atto della prima accensione ogni pin di I/O viene automaticamente impostato dall'hardware come ingresso ad alta impedenza, successivamente il software utente può modificare la configurazione di ogni singolo pin abilitandolo a funzionare in un diverso modo.
Le porte di I/O dei micro ST6 vengono gestite da tre registri per ogni porta:

1. DR Data Register
2. DDR Data Direction Register
3. OR Option Register

Le tre sigle indicate (DR, DDR, OR) sono sempre seguite dalla lettera che indica la porta (A, B, C) (Esempio Data Register della Port A: DRA).
In tabella abbreviazioni e locazioni RAM relative a questi registri.

	Port **A**	
Registro	**Abbreviazione**	**Locazione RAM (Hex)**
Data **R**egister	DRA	0C0
Data **D**irection **R**egister	DDRA	0C4
Option **R**egister	ORA	0CC

	Port **B**	
Registro	**Abbreviazione**	**Locazione RAM (Hex)**
Data **R**egister	DRB	0C1
Data **D**irection **R**egister	DDRB	0C5
Option **R**egister	ORB	0CD

	Port **C**	
Registro	**Abbreviazione**	**Locazione RAM (Hex)**
Data **R**egister	DRC	0C2
Data **D**irection **R**egister	DDRC	0C6
Option **R**egister	ORC	0CE

Ogni piedino di I/O del micro appartiene ad una sola periferica ed è contraddistinto dalla lettera P (Port) seguita dal nome della periferica di appartenenza (A, B, C) e dalla relativa posizione nei registri (0 – 7).

Ad esempio, il pin 15 del micro ST6265 è contraddistinto dalla sigla PA3 poiché viene controllato dalla periferica di I/O siglata A, e precisamente dal bit 3 dei registri DRA, DDRA e ORA. Per modificare il funzionamento di un pin di I/O dovremo quindi agire sui tre relativi bit dei registri della periferica di appartenenza (fig.4)

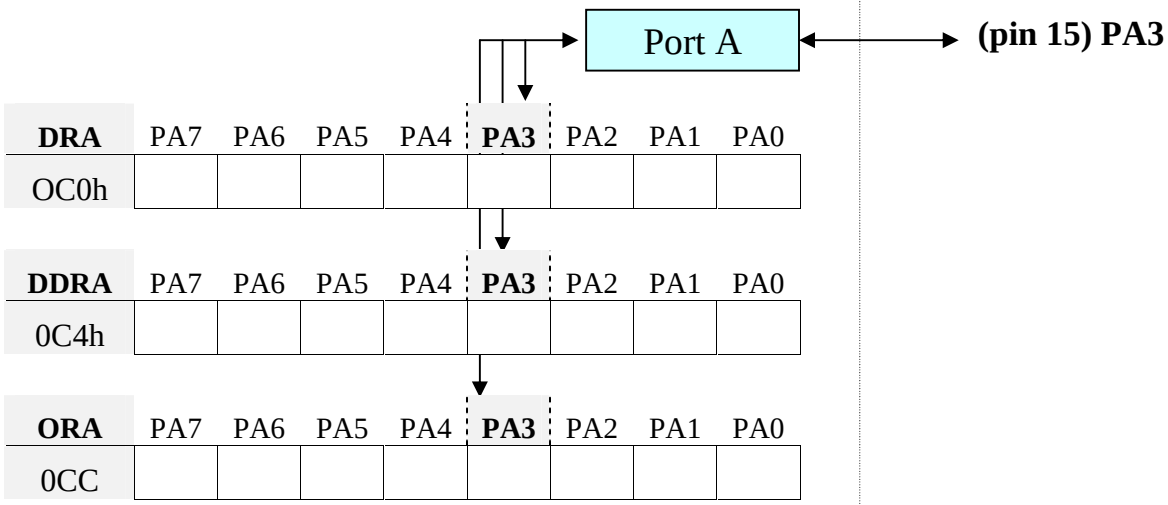


Fig.4

In figura (fig.5), lo schema a blocchi di una linea di Input/Output

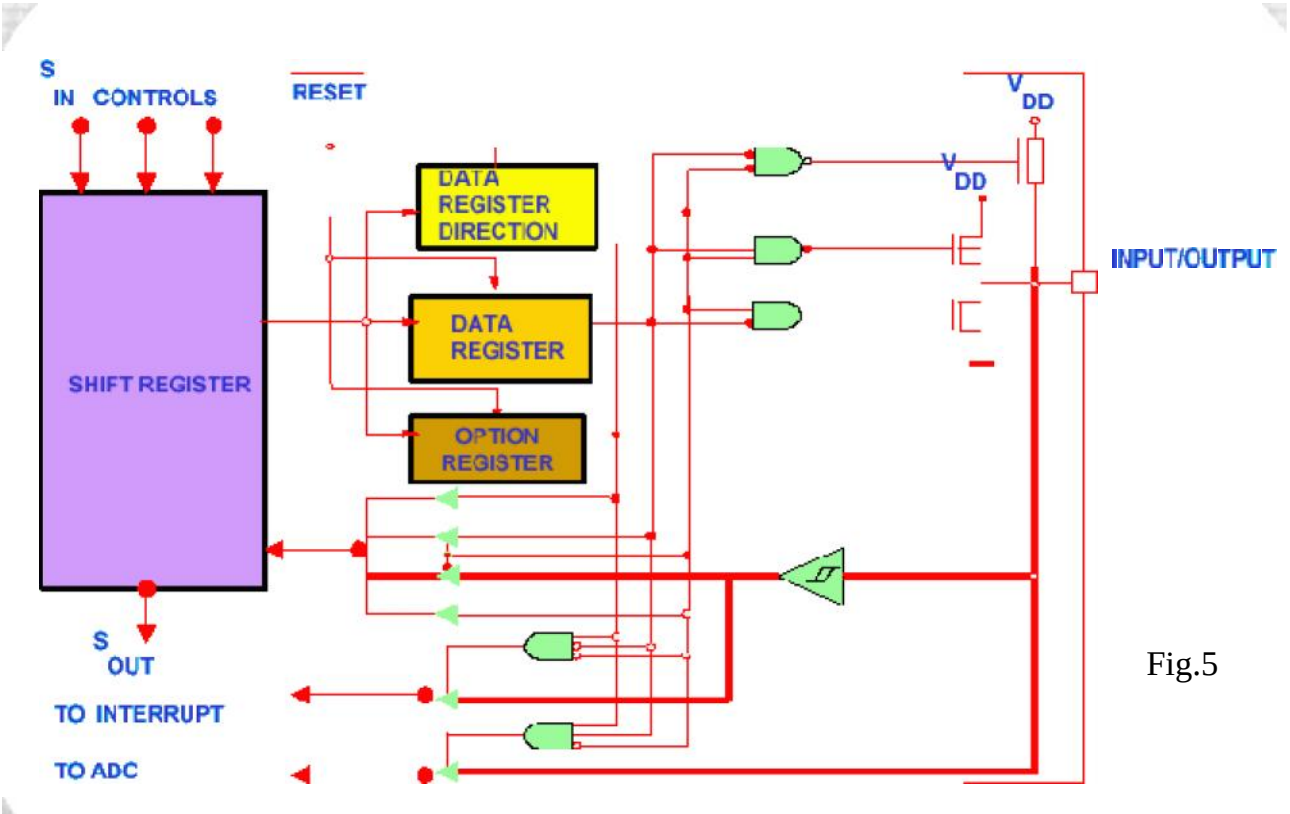


Fig.5

I registri di controllo delle porte sono sempre ad 8 bit, ma in realtà in alcuni casi, non tutti i bit di questi registri vengono usati perché le corrispondenti linee di I/O non sono fisicamente presenti nel chip (esempio Port C)

5.3 Configurazione delle Porte I/O

Le porte di I/O dei micro ST6 possono essere configurate con le seguenti modalità:

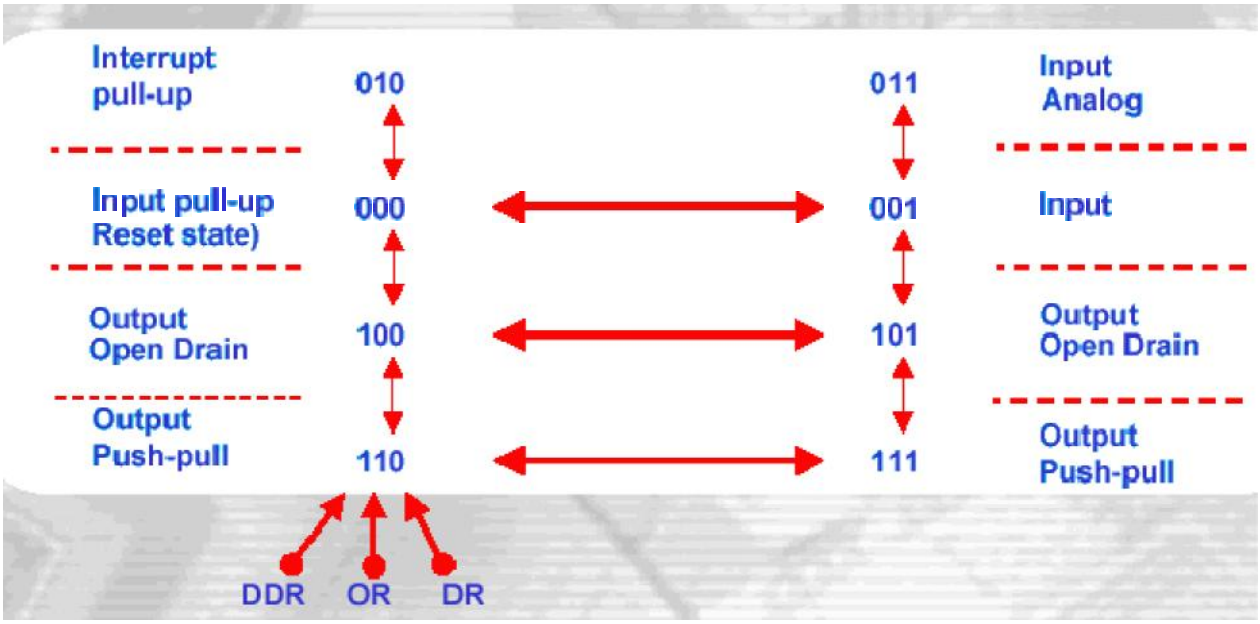
1. ingresso con resistenza di PULL-UP, senza interrupt (default);
2. ingresso normale senza PULL-UP; senza interrupt
3. ingresso con PULL-UP e interruzioni;
4. ingresso analogico (solo alcune linee);
5. uscita OPEN-DRAIN (collettore aperto)
6. uscita PUSH-PULL.

Per attivare queste modalità si devono impostare i bit dei registri delle porte di I/O secondo la seguente tabella

DDR	OR	DR	Funzionamento del pin	Modo
0	0	0	ingresso con resistenza di PULL-UP, senza interrupt(default)	Input
0	0	1	ingresso normale senza PULL-UP, senza interrupt	Input
0	1	0	ingresso con PULL-UP e interrupt	Input
0	1	1	ingresso analogico (solo alcune linee);	Input
1	0	X	uscita OPEN-DRAIN (collettore aperto)	Output
1	1	X	uscita PUSH-PULL	Output

X= non è importante

Per passare da una delle sei configurazioni sopra riportate ad un'altra configurazione, occorre seguire un particolare criterio onde evitare anomalie nel funzionamento del microcontrollore.
Le variazioni di configurazione debbono cioè essere eseguite seguendo una particolare sequenza di transizione di seguito riportata.

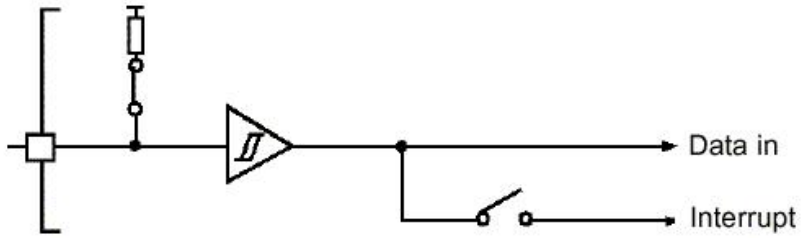


le transizioni possono avvenire sia nel senso di lettura sia nel senso inverso, è sufficiente che la transizione successiva sia quella adiacente, i valori riportati corrispondono ai relativi bit dei registri DDR, OR, DR, relativi ad una singola porta.

5.4 Descrizione delle opzioni disponibili sulle porte di I/O

1) Ingresso con resistenza di PULL-UP, senza interrupt (default) DDR=0, OR=0, DR=0

Abilitando questa opzione, il micro inserisce al suo interno una resistenza, tra il piedino selezionato e la tensione positiva di alimentazione.
Questo è particolarmente utile in molte applicazioni, in quanto consente di ridurre il numero

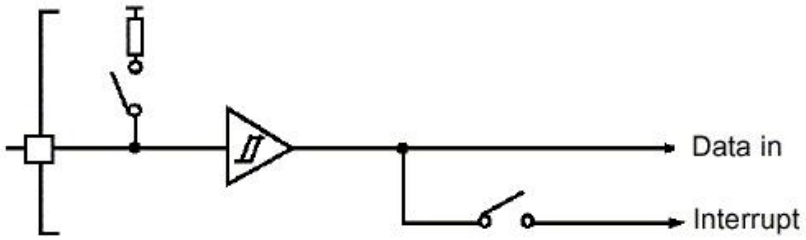


Port	Modo
PA0-PA7 PB0-PB5 PB6-PB7 PC0-PC4	Ingresso con Pull-Up, senza interrupt

di componenti esterni necessari. Pensiamo ad esempio alla gestione di un pulsante collegato all'ingresso: abilitando l'opzione "PULL-UP" sarà sufficiente collegare il pulsante, tra il piedino del micro prescelto e la massa per generare i due stati logici che ne individuano la posizione del pulsante.
Quando la CPU legge un livello logico uguale a 1 significa che il pulsante è rilasciato, viceversa, quando il livello logico sul pin è uguale a zero, significa che il pulsante è premuto.

2) Ingresso normale senza PULL-UP, senza Interrupt DDR=0, OR=0, DR=1

Nella configurazione senza resistori di "PULL-UP", il circuito collegato all'esterno deve

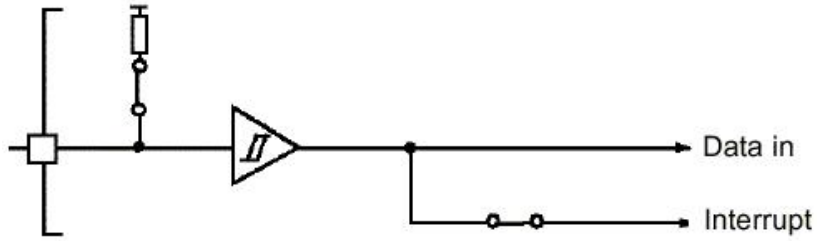


Port	Modo
PA0-PA7 PB0-PB5 PB6-PB7 PC0-PC4	Ingresso normale, senza Pull-Up, senza interrupt

essere in grado di fornire autonomamente i due stati logici riconosciuti (1-0) senza lasciare fluttuante, il piedino collegato, il che provocherebbe mal funzionamenti sul riconoscimento dello stato logico.

3) Ingresso con resistenza di PULL-UP e Interrupt
DDR=0, OR=1, DR=0

Con questa selezione un determinato segnale sul pin di ingresso, obbliga la CPU a leggere le istruzioni contenute in una particolare mappa di memoria (Vettori di interrupt)



Port	Modo
PA0-PA7 PB0-PB5 PB6-PB7 PC0-PC4	Ingresso con Pull-Up e interrupt

PORT A e PORT B- Vettore #1- FF6h-FF7h (Memoria programma)
PORT C - Vettore #2- FF4h-FF5h (Memoria programma)

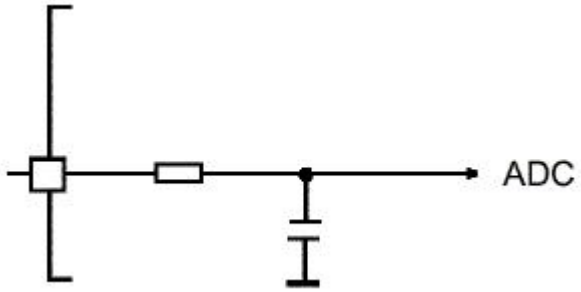
Nella maggior parte delle applicazioni, l'ingresso non viene gestito via INTERRUPT ma via software con un metodo detto di "POLLING" . Esso consiste nel verificare in continuazione lo stato dell'ingresso per agire quando tale ingresso risulta variato rispetto alla condizione precedente.

Possono però capitare, applicazioni in cui il programmatore non può permettersi la gestione dell'ingresso in "POLLING", in questo caso sarà sufficiente selezionare per quel pin l'opzione interrupt. Ad esempio supponiamo di inizializzare la linea denominata PA0 come ingresso dotato di interruzione, quando avviene l'evento esterno associato al PA0 la periferica di I/O forza la CPU ad abbandonare il suo normale corso di istruzioni e va ad eseguire l'istruzione contenuta nella locazione FF6h definita come vettore di interrupt della porta A (Vettore#1).

4) Ingresso analogico (solo alcune linee, 13 lineee ST6265)
DDR=0, OR=1, DR=1

Abilitando l'opzione, ingresso analogico, la tensione presente sul piedino, viene applicata al convertitore A/D interno che lo trasforma in un numero, ad essa proporzionale (la tensione analogica, può variare, tra VSS e VDD in genere 0V-5V tale intervallo viene convertito nell'intervallo numerico 0-255).

Questa particolare funzione verrà descritta in seguito nelle trattazione della periferica A/D,

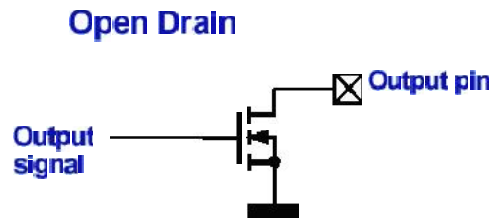


Port	Modo
PA0-PA7 PC0-PC4	Ingresso analogico

solo un piedino per volta può essere abilitato a funzionare come ingresso analogico, poiché il micro controllore della famiglia ST6 dispone di un'unica periferica A/D.

5) Uscita Open-Drain
DDR=1, OR=0, DR=X

Abilitando l'uscita open-drain alcune linee sono in grado di supportare, una corrente massima di 20 mA quindi, sono in grado di pilotare senza l'interposizione di un BUFFER,

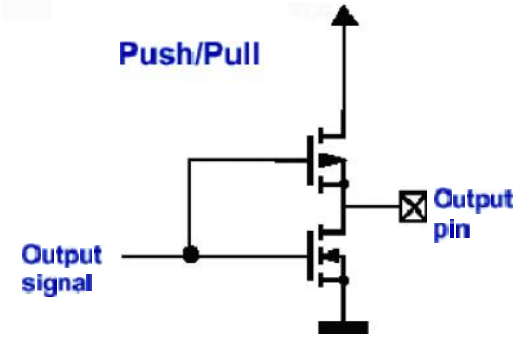


Port	Modo
PA0-PA7 PC0-PC4	Uscita Open-Drain - 5 mA
PB0-PB5 PB6-PB7	Uscita Open-Drain - 20 mA

un diodo led, oppure un piccolo relè, o anche un triac. Nei micro ST626X, solo la porta B può essere attivata, come uscita open-drain a 20 mA, le altre porte (PORT A e PORT B) possono essere abilitate come uscite open-drain a 5mA.

6) Uscita Push-Pull
DDR=1, OR=1, DR=X

Questa uscita è caratterizzata dalla possibilità di generare due livelli di uscita in funzione di



Port	Modo
PA0-PA7 PC0-PC4	Uscita Push-Pull - 5 mA
PB0-PB5 PB6-PB7	Uscita Push-Pull 20 mA

uno stato di un particolare bit, esso è contenuto all'interno del registro dati denominato DR (data register). Ogni linea o porta di uscita dispone del relativo bit all'interno del registro DR, associando a tale bit il valore logico 1 avremo sulla linea selezionata, una tensione uguale a VDD e viceversa, se al bit associamo il valore logico zero, la tensione sulla porta diventa VSS.
Nei micro ST626X solo il PORT B può essere attivato come uscita PUSH-PULL a 20mA, le altre porte possono essere abilitate come PUSH-PULL a 5mA.

All'atto della prima accensione, o dopo l'impulso di reset, l'hardware del micro configura tutte le porte di input-output come resistori di pull-up. Il programma utente configurerà attraverso gli appositi registri, le linee di ingresso-uscita nel modo desiderato; utilizzando opportune istruzioni software possiamo trasformare a piacere, le porte d'ingresso in porte di uscita e viceversa, ovviamente rispettando l'hardware esterno collegato al micro. Nella configurazione di una porta come ingresso, difficilmente potremo causare danni al

microcontrollore, qualunque siano i dispositivi collegati a tale porta; mentre nella configurazione di una porta come uscita, dovremo rispettare le caratteristiche elettriche della ditta costruttrice del micro evitando, ad esempio, di prelevare una corrente troppo alta. Ad eccezione di alcuni pin, che come detto, possono erogare fino a 20mA nella configurazione open-drain (PORT B); per le altre linee di input output (PORT A e PORT C) la corrente massima prelevabile è di circa 5mA.

I registri disponibili per comunicare con le periferiche di I/O risultano essere nove:
DDRA, DDRB, DDRC, ORA, ORB, ORC, DRA, DRB, DRC, e ognuno di essi occupa uno spazio nella memoria RAM del micro pari ad un byte.
Il registro di direzione (DDR) consente di inizializzare la periferica come ingresso o come uscita:

- DDR=0 Input
- DDR=1 Output

Il registro opzioni (OR) e il registro dati (DR) sono usati in fase di inizializzazione per selezionare le opzioni della periferica, inoltre il registro dati (DR) è usato durante lo svolgimento del programma per leggere lo stato di un ingresso oppure per settare o resettare lo stato di una uscita..
Ogni linea viene controllata da tre bit presenti nei tre registri della periferica di appartenenza, esempio la linea di I/O PA3 viene controllata e gestita tramite il bit 3 del registro DDRA, il bit 3 del registro ORA, ed il bit 3 del registro DRA.(fig4, pag.26). Modificando questi tre bit (tabella pag.27 e transizioni pag.27) inizializziamo la porta a funzionare in uno dei 6 modi disponibili (4-input, 2-output)
Per associare ad un bit il valore logico “1” si usa l’istruzione SET (Set Bit), mentre per associare al bit il valore logico 0 si usa l’istruzione RES (Reset Bit).
Queste istruzioni possono essere utilizzate solo se la periferica è programmata per funzionare come uscita in tutte le sue linee.Nella maggior parte delle applicazioni non è sempre possibile soddisfare questa esigenza, la periferica viene spezzata in due, una parte funzionerà come uscita ed un’altra parte funzionerà come ingresso. In quest’ultimo caso, l’hardware del micro non è in grado di supportare le istruzioni SET e RES, e poiché non ve ne sono altre disponibili ricorreremo ad un piccolo stratagemma.
Esso consiste nel creare in una cella di memoria RAM una copia del registro dati, ovvero in fase di inizializzazione riportiamo tutte le modifiche effettuate sul registro dati anche in questa cella RAM.
Utilizziamo, le istruzioni SET e RES solo sulla cella RAM per poi copiarne il contenuto all’interno del registro dati.

I micro ST626X dispongono di alcuni pin che si utilizzano, sia come appena descritto, sia in abbinamento alle nuove periferiche implementate (fig 3, pag.24); per la precisione questi piedini sono siglati:

1) PC3 (Uscita seriale) Sout
2) PC2 (Ingresso seriale) Sin
3) PC4 (Clock della periferica seriale) Sck
4) PC1 (Input-Output del Timer 1) Tim1
5) PB6 (Ingresso Timer Autoreload) ARTIMin
6) PB7 (Uscita Timer Autoreload) ARTIMout

Queste opzioni vengono gestite con altri registri

5.5 CONVERTITORE A/D

I micro ST6 dispongono di un versatile convertitore Analogico/Digitale ad 8 bit (periferica 8 bit A/D Converter) che consente di collegare direttamente al chip dispositivi esterni, i quali generano segnali che abbiano un'escursione di tensione compresa tra Vdd e Vss.
Il segnale analogico viene letto attraverso un piedino di I/O abilitato al funzionamento analogico, convertito in un numero digitale dalla periferica ADC, e memorizzato all'interno di un registro denominato "ADR" (A/D Converter Data Register).
La periferica di conversione analogico/digitale è caratterizzata da tre parametri fondamentali:

1) Risoluzione;

2) Precisione;

3) tempo di conversione.

Risoluzione: coincide con il minimo valore di tensione che la periferica può distinguere se espressa in maniera analogica, oppure coincide con il numero di bit che compongono il risultato della conversione A/D se espressa digitalmente.Nei micro della famiglia ST6 la risoluzione digitale dell'ADC è di 8 bit, il valore decimale del risultato della conversione può quindi spaziare da 0 a 255. La risoluzione analogica si calcola dividendo la tensione di alimentazione del micro per il numero di combinazioni decimali disponibili nel risultato:

Esempio: Vdd=5V, Vss=0V Risoluzione digitale 8 bit

Risoluzione analogica $Ra = \frac{Vdd - Vss}{2^8} = \frac{5 - 0}{256} = 19{,}53 mV$

Precisione: La presione del'ADC dell'ST6 è ≤ 2bit LSB, il risultato della conversione, contenuto nel registro ADR, può differire rispetto al valore reale di 2 bit LSB in più o 2 bit LSB in meno. Considerando una tensione di alimentazione di 5V, il risultato della conversione A/D può differire al massimo del valore reale di ≤ 39,06 mV

Tempo di conversione: Intervallo di tempo tra l'inizio della conversione e il termine della conversione, ed è uguale tipicamente a 70 μs.

L'ingresso analogico della periferica ADC può essere fisicamente collegato ai seguenti pin:

Micro	**N.linee**	**Pin Port**
ST6260	7	Port A (PA0, PA1, PA2, PA3) Port C (PC2, PC3, PC4)
ST6265	13	Port A (PA0, PA1, PA2, PA3, PA4, PA5, PA6, PA7) Port C (PC0, PC1, PC2, PC3, PC4)

E' importante ricordare che solo un ingresso alla volta può essere abilitato al funzionamento analogico, attraverso opportune istruzioni software, ed essere fisicamente collegato all'ingresso della periferica ADC.

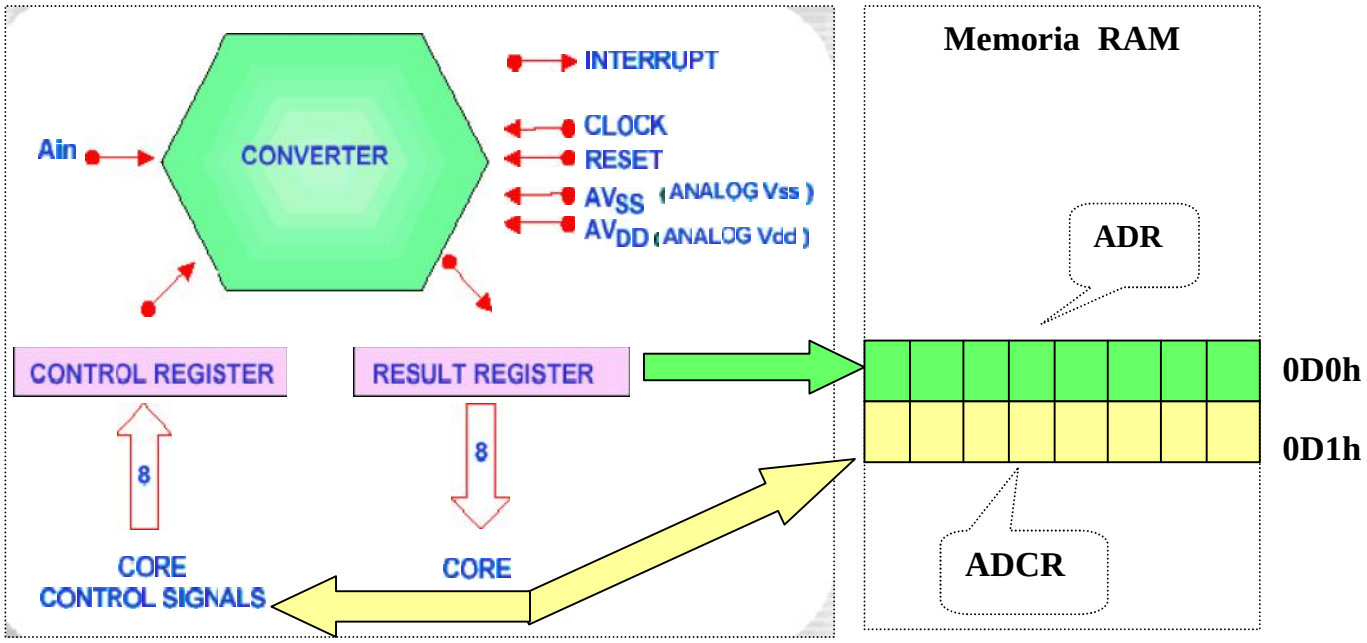
La procedura corretta per la gestione di un ingresso analogico è la seguente:

1) abilitare l'ingresso prescelto al funzionamento analogico;
2) inizializzare la periferica ADC e dare avvio alla conversione;
3) attendere il termine della conversione;
4) leggere il valore all'interno del registro dati ADR (indirizzo:0D0h);
5) disabilitare la periferica ADC e disabilitare l'ingresso utilizzato.

Il software comunica con la periferica di conversione A/D attraverso due registri:

- **Registro Dati : ADR (indirizzo loc. RAM: 0D0h)**
- **Registro di Controllo : ADCR (indirizzo loc. RAM: 0D1h)**

Il registro dati (ADR) (registro ad 8 bit di sola lettura), contiene il risultato dell'ultima conversione.
Il registro di controllo (ADCR ad 8 bit di lettura/scrittura), permette la gestione dei segnali di controllo dalla periferica A/D: i bit 0, 1, 2, 3 sono riservati alla CPU, mentre i bit 4, 5, 6, 7 sono utilizzati dall'utente.



D7	D6	D5	D4	D3	D2	D1	D0
EAI	**EOC**	**STA**	**PDS**				

Registro ADCR (A/D Converter Control Register, Read/Write, indirizzo:0D1h

Bit 7: EAI (**E**nable **A**/D **I**nterrupt): permette di abilitare l'interrupt della periferica ADC, settando questo bit, ovvero associando al bit il valore logico "1", abilitiamo la periferica a generare un inerrupt al termine della conversione A/D. con EAI uguale a "1" la periferica forza la CPU ad abbandonare il suo normale ciclo di istruzione (la posizione in cui si trova la CPU viene copiata all'inerno del registro di Stack) e a processare l'istruzione contenuta nella locazione FF0h (vettore interrupt #4).

Bit 6: EOC (**E**nd **O**f **C**onversion): questo bit viene automaticamente posto a "1" dall'hardware del micro quando la periferica ha terminato la conversione, mentre viene posto a "0" quando inzia una conversione.

Bit 5: STA (**STA**rt of Conversion): consente di attivare la conversione, questo bit non può essere letto ma solo scritto, associando al bit STA il valore logico "1" la conversione ha inizio e l'hardware del micro resetta automaticamente il bit EOC. Se il bit STA viene posto a "1" mentre si sta svolgendo una conversione A/D, si ottiene che la conversione in atto viene bloccata e viene avviata una nuova conversione.

Bit 4: PDS (**P**ower **D**own **S**election): consente di abilitare o disabilitare la periferica di conversione A/D: (PDS=1, periferica A/D abilitata; PDS=0, periferica A/D disabilitata). Quando il bit PDS è uguale a "0" la periferica A/D entra in uno stato di funzionamento denominato "*idle mode*", esso permette di annullare la corrente assorbita dalla periferica e quindi di ridurre i consumi del micro. In applicazioni a basso consumo, ad esempio quando il micro è alimentato da una batteria, è consigliabile mantenere il bit PDS=1 solo durante la lettura di un ingresso analogico.

Bit 3,2,1,0: (Reserved CPU), sono riservati alla CPU

Specifiche tecniche del convertitore A/D

Symbol	Parameter	Test conditions	Value			Unit
			Min	Typ	Max	
Res	Resolution			8		bit
A_{Tot}	Total Accuracy	Fosc>1,2 MHZ Fosc>32KHZ		{2 {4		LSB
Tc	Conversion Time			70		⇑s
V_{AN}	Conversion Range		Vss		Vdd	V
ZIR	Zero Input Reading	Conversion result when Vin=Vss	00			Hex
FSR	Full Scale Reading	Conversion rsult when Vin=Vdd			FF	Hex
AD	Analog Input Current During Conversion	Vdd=4,5V			1,0	⇑A
AC_{IN}	Analog Input Capacitance			2	5	pF
ASI	Analog Source Impedence	Analog Channel switched just before conversion start			30	Kϑ

La scansione del tempo

6.1 Il Clock

Il microcontrollore, per eseguire tutti i suoi compiti necessita di un segnale di sincronismo, tale segnale è chiamato **clock** (orologio)

All'interno del microcontrollore troviamo un oscillatore che determina il periodo del ciclo macchina

Per funzionare l'oscillatore necessita di pochissimi componenti esterni: un quarzo o un risonatore ceramico, con frequenza di oscillazione compresa solitamente tra 4MHz e 8 MHz, e due condensatori con capacità compresa tra 12pF e 22 pF (fig.1)

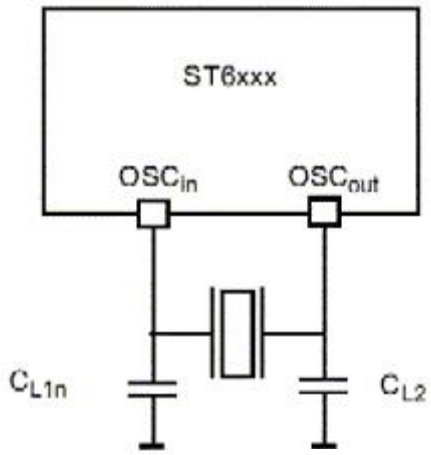


Fig.1

All'atto della prima accensione, trascorso il ritardo di reset(POR), la CPU inizierà a lavorare "mossa" dal dock generato dall'oscillatore.

La velocità con cui la CPU svolge le sue mansioni è ovvia mente correlata alla frequenza di oscillazione del quarzo esterno: quanto più alta sarà la frequenza del quarzo, tanto più alta sarà la velocità di lavoro del micro e il consumo di corrente, al contrario, abbassando la frequenza del quarzo, andremo a diminuire la velocità del micro ed anche il suo consumo di corrente.

Il circuito formato dal quarzo, dai condensatori, e dall'oscillatore interno, genera il cosiddetto clock di sistema, ovvero un segnale ad onda quadra che determina la sequenza degli eventi all'interno del micro.

La CPU processa, ovvero esegue, le istruzioni presenti nella memoria programma (ROM/EPROM) sempre con la stessa velocità; il micro ST6 interpreta ed esegue sempre la medesima istruzione con un uguale numero di cicli macchina. Utilizziamo il termine “machine cycle” o ciclo temporale di macchina come unità di misura della velocità del micro; ad ogni singola istruzione, corrisponde un numero di cicli macchina (fissati dalla casa costruttrice) che ne quantifica il tempo di esecuzione.

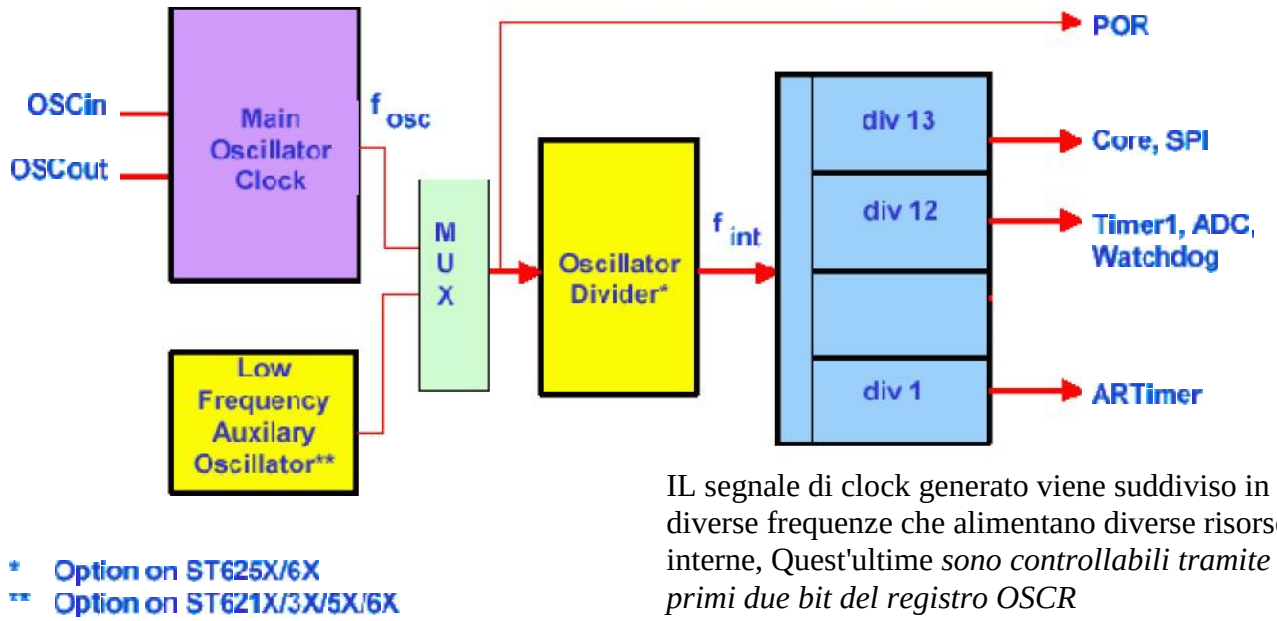
Ad esempio, l'istruzione **LDI A,nn** (trasferisci il numero ad otto bit **nn** all'interno del registro accumulatore) richiede per essere eseguita 4 cicli macchina, ovvero devono trascorrere quattro cicli macchina fra l'istante in cui la CPU del microcontrollore legge nella memoria l'istruzione **LDI A,nn** e l'istante in cui il numero nn viene trasferito all'interno del registro accumulatore.

Il numero di cicli macchina necessari per eseguire una istruzione varia da un minimo di uno ad un massimo di cinque.

Le risorse interne dei microcontrollori ST6 non vengono guidate tutte dalla stessa frequenza, alcune lavorano più velocemente ed altre meno; per questo motivo, per adeguare le diverse velocità di esecuzione, occorre dividere la frequenza del clock più volte per ottenere varie sottofrequenze di lavoro.

Le frequenze che vengono applicate alle diverse unità sono (fig.2)

- CORE e periferica seriale SPI uguale a clock diviso 13;
- TIMER 1, WATCHDOG e convertitore A/D clock diviso 12;
- AR TIMER (timer autoricaricabile) viene pilotato direttamente con la frequenza di clock.



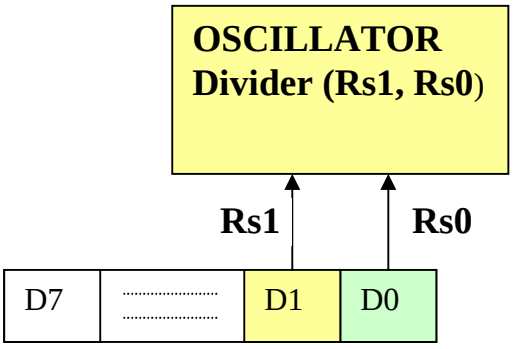
IL segnale di clock generato viene suddiviso in diverse frequenze che alimentano diverse risorse interne, Quest'ultime *sono controllabili tramite i primi due bit del registro OSCR*

Fig.2 (schema a blocchi dell'oscillatore e del divisore di frequenze)

In tabella è riportata la frequenza del segnale applicato al core (Fcore) ed il periodo (Tcore) con quarzo a 6 MHz e 8 MHz, **un Tcore corrisponde ad un ciclo macchina**

Clock (MHz)	Fcore (MHz)	Tcore («s)
6	0,462	2,167
8	0,615	1,625

Tornando all'esempio precedente, se vogliamo calcolare il tempo di esecuzione dell'istruzíone LDI A,nn (supponendo di collegare esternamente al micro un quarzo da 8 MHz), dalla tabella si determina $T_{LDI\ A,nn}$=4*1,625⇑s =6,5⇑s, in quanto detta istruzione necessita di quattro cicli macchina.
E' possibile dividere ulteriormente 1e frequenze applicate alle varie unità agendo sul registro di controllo dell'oscillatore **OSCR** (Oscillator Control Register, locazione RAM: 0DCh) (fig.3), tale registro può essere solo scritto, vengono usati solo due bit, il DO e il D1, per selezionare il fattore di divisione dell'oscillatore in funzione della seguente tabella:



D1	D0	Fattore di divisione
0	0	1
0	1	2
1	0	3
1	1	4

Fig.3 Registro OSCR (0DCh)

6.2 Il watchdog (Loc. RAM 0D8h, Read/Write)

Il watchdog consiste in un circuito contatore al quale è associato un byte di memoria, denominato registro DWDR (loc. RAM 0D8h), che ad intervalli regolari viene decrementato di una unità. Quando si lavora con programmi molto lunghi può capitare che si vada "*fuori-programma*": determinate condizioni possono portare il software, che non è stato provato in tutte le sue possibilità, ad eseguire sempre la stessa sequenza di istruzioni senza possibilità di uscita; può anche capitare che un impulso spurio esterno, metta il core del micro in condizione di "stallo", con la CPU che continua ad eseguire sempre la stessa istruzione.
In entrambi i casi l'intervento del watchdog causa un restart del micro, ovvero forza la CPU ad eseguire la prima istruzione presente in ROM come se qualcuno dall'esterno toglie per un istante la tensione di alimentazione.
Il watchdog utilizza, come già accennato, un registro denominato DWDR (Digital Watch Dog Register) (fig.4), esso è composto da otto bit e svolge due differenti funzioni:

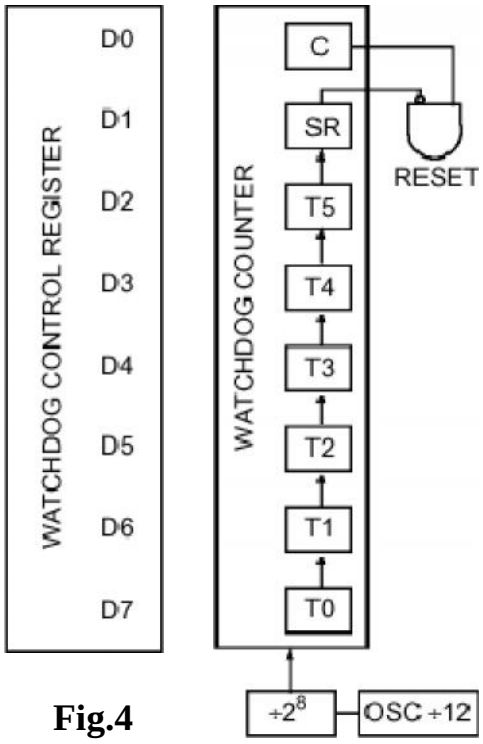


Fig.4

1. attivazione/disattivazione;
2. richiesta di reset.

Il comando di attivazione e disattivazione avviene mediante il bit 0 denominato "**C**", settando il quale (scrivendo "1" nel bit) si attiva il watchdog; ovviamente portando a zero questo bit la funzione viene disattivata. Questa operazione può essere effettuata solamente in alcuni micro, precisamente in quelli la cui sigla termina con l'estensione: "/SWD". I micro con estensione "/HWD", invece, non permettono la gestione del bit 0 del registro DWDR in quanto tale bit è settato automaticamente dall'hardware e non può essere modificato da software.

La richiesta di reset è implementata nel secondo bit denominato "SR" (Software Reset Bit): quando questo bit viene resettato ed il bit "C" è allo stato logico 1, il watchdog genera un impulso di reset.

Durante il normale funzionamento il registro DWDR viene decrementato con una frequenza legata al clock del micro. Tale frequenza è uguale a quella del quarzo divisa per 12 e per 2^8 (12 x 256 = 3072). (fig.4); per il decremento tale registro utilizza 6 bit (D7, D6, D5, D4, D3, D2), corrispondenti ai Flip-Flop T: T0, T1, T2, T3, T4, T5).
Se utilizziamo, ad esempio, un quarzo esterno da 8 MHz il contatore DWDR viene decrementato ogni 384 microsecondi: infatti:

Fosc=8MHZ $F_{DWDR} = \frac{Fosc}{3072} = 2{,}604 KHz$ quindi $T_{DWDR} = \frac{1}{F_{DWDR}} = \frac{1}{2604} = 384 \mu s$

A questo punto agendo sui bit di conteggio del registro DWDR (D7, D6, D5, D4, D3, D2; 6bit; 2^6=64 combinazioni) possiamo generare un reset con tempo variabile da 384 microsecondi a 24,576 millisecondi.
Se, ad esempio, scriviamo nel registro del watchdog il valore esadecimale FF, abilitiamo questa periferica a generare un reset dopo 24,576 millisecondi. In tabella un esempio di calcolo del tempo di ritardo del reset generato dal Watchdog.

Valore nel DWDR (hex)	**VD_{10} =Valore nel contatore (base 10)**	**Ritardo reset 384 «s*(VD_{10}+1)**
00	0	384 ⇑s
A4	41	16,128 ms
FF	63	24,576 ms

Per evitare che il circuito generi l'impulso di reset dovremo, prima dello scadere del tempo impostato, ricaricare il registro con il valore FF o con un altro dato.

In fig 5 lo schema a blocchi del circuito watchdog

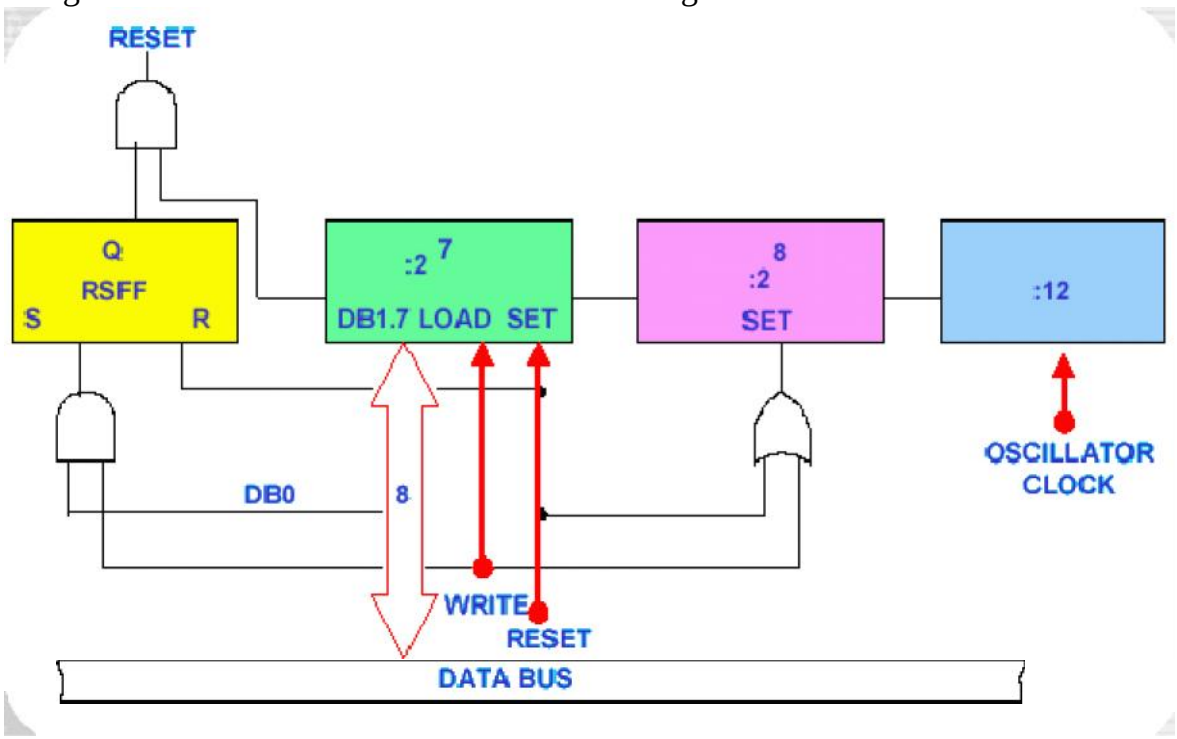


Fig.5