Interfacciamento dei processori

Prima Parte: Interfaccia d'uscita per un dato a 8 bit

PREMESSA

L'utilizzo dei processori è una pratica ormai inderogabile: nessuno degli oggetti più comuni può farne a meno, dai *telefonini* ai *navigatori satellitari*, dagli *i-pod* ai *lettori mp3*, dalla *lavatrice di casa* all'*impianto di riscaldamento*..

In realtà questi oggetti sono dei veri e propri **microcomputer**: il **microprocessore** (o, sinteticamente, il **processore**, o ancora più in sintesi, la **CPU**, **C**entral **P**rocessing **U**nit, unità centrale di processo) è *solo una parte* dei dispositivi programmabili che li governano, per altro quella più importante, chiamata a controllare e decidere, ad eseguire ed elaborare, ma **nulla potrebbe** questo potente *cervello* se non disponesse della **memoria** e delle **porte** di ingresso e di uscita:

- la **memoria** è il luogo nel quale il processore trova conforto: la sua grande capacità, le sue enormi potenzialità *sono **niente*** senza la memoria; la prima cosa che fa, senza saperlo e senza volerlo, *per istinto*, è saltare in una locazione di memoria, alla ricerca di qualche cosa da fare... Dunque, la *memoria* è indispensabile in un computer perchè contiene il **programma** che il processore è chiamato ad eseguire e perchè si presta, eventualmente, a conservare i **dati** da esso elaborati
- nessun controllo di dispositivo potrà, poi, essere esercitato dalla CPU se non dispone di una **struttura di interscambio**, uno o più canali delegati alla trasmissione e alla ricezione delle informazioni con il *resto del mondo*.

In sintesi, per funzionare, qualunque computer deve disporre di **tutte e tre** queste componenti e, per rendere la cosa economica e competitiva, da molto tempo i vari produttori hanno provveduto ad *integrarle in un unico dispositivo*, noto come **single-chip** o **PIC** (**P**eripheral **I**nterface **C**ontroller o **P**rogrammable **I**ntelligent **C**omputer) o **microcontrollore**.

Poichè *l'appetito vien mangiando* i moderni microcomputer, oltre alla **memoria** *RAM* (per i dati) e quella *flash* (per il programma) ospitano internamente numerose **porte parallele** (programmabili sia in uscita che in ingresso) e almeno una **porta seriale** RS232, completando la dotazione con alcuni **temporizzatori** (timer), **comparatori** e **convertitori analogico digitali** (ADC): un vero lusso!!

Naturalmente il controllo dei dispositivi esterni può essere esercitato *anche* con un **computer tradizionale**, scrivendo programmi in grado di gestire le sue porte classiche, parallela e seriale, poste sulla sua parte posteriore; l'avvento delle nuove interfacce seriali (USB, Firewire, ..) rende comunque questo esercizio sempre più impegnativo.

Sono numerosi gli appassionati che si lasciano tentare da questa nuova frontiera, affidando la loro volontà e le loro speranze a un *ATMEL* o a un *MicroChip* o a un *STmicroelectronics* o chi sa che altro; una enorme disponibilità di **single-chip**, tutti con la loro *architettura*, la loro dignità e il loro dialetto; oggetti *operativamente* molto simili tra loro, ma *sostanzialmente* differenti...

L'utilizzo pratico dei microcontrollori non è difficile e non deve intimorire:

- tutti hanno bisogno di una serie di **registri** (la *memoria personale* interna con la quale rendono possibili le operazioni logiche ed aritmetiche, tipiche di ciascuno di essi); poiché essi sono inseriti in una *architettura* tipica, è necessario conoscerla bene per poterli utilizzare al meglio
- tutti hanno bisogno di un **set d'istruzioni** : esso cambia effettivamente da processore a processore ma è *sostanzialmente sempre lo stesso* su ogni piattaforma e differisce solo per lo *mnemonico* usato (per esempio taluni processori utilizzano **MOV** invece di **LD** o di **LOAD** o altro): le varie *razze* di processori **parlano la stessa lingua** ma con **dialetti diversi**!

La programmazione dei microcontrollori è dunque un'arte che va consumata con pazienza e amore; il modo più immediato ed efficiente fa riferimento all'**Assembly** (come detto, tipico di ciascuno di loro) anche se è sempre più diffusa la possibilità di fruire di ambienti di *programmazione ad alto livello* (di solito **C**-language, ma anche **Pascal** o altro).

L'utilizzo di linguaggi evoluti, decisamente più vicini all'uomo e meno alla macchina (cioè lo stesso single-chip) è un'arma a doppio taglio: il codice così prodotto è certamente *molto più lungo*, richiedendo *maggiore memoria* per il suo stoccaggio sulla flash; il problema è però sempre meno preoccupante perché con il passare del tempo la dimensione della *memoria di programma* garantita *on-board* è sempre più grande!!

Il problema eventualmente da valutare con attenzione è legato ai *tempi di esecuzione* di un programma progettato in **C** o in **Pascal**, certamente *di gran lunga maggiori* di quelli ottenibili con un codice generato con l'**Assembly**; ma chi non ha tempo da perdere trova fantastico delegare compiti importanti (come la gestione di un visualizzatore LCD o il controllo di una periferica seriale, ..) a *procedure* belle e pronte..

Chi può dar loro torto? con buona pace di quelli (ahimè come il sottoscritto) che amano *sapere il perché* delle cose e non amano farsi *pulire il sederino* da nessuno, affrontando l'ignoto con determinazione e voglia di debellarlo.

Ma questo è romanticismo e, nel nostro mondo, qualcuno ha deciso per noi che bisogna *correre*, *produrre* e *trarre il massimo profitto* **subito e** possibilmente **gratis**, senza spendere nulla di proprio, sfruttando l'intelligenza altrui e mortificando la propria.

L'INTERFACCIAMENTO

Per **entrare nel merito** degli argomenti di questo primo inserto, qualunque computer (*microcontrollore* o *personal*) si propone di garantire il proprio servizio alle cosiddette **periferiche** (stampanti, mouse, tastiere, visualizzatori, monitor, hard disk, ecc..) e per questo deve disporre di **interfacce**, cioè di uno *strato di elettronica* in grado di mediare tra le *offerte* del primo con le *necessità* del secondo.

Il compito più banale di una **interfaccia** è quello di **assicurare i segnali elettrici adatti** da entrambe le parti, per rendere possibile lo scambio di informazioni tra di esse: è noto a tutti che l'informazione (sinteticamente **bit**, **b***inary un***it**, organizzata comunque in **byte**) altro non è che un livello di **tensione** *presente* (= "**1**") o *assente* (="**0**") su di un filo conduttore...

Tuttavia una **interfaccia** può anche essere, a sua volta, *intelligente* e *specializzata*, con l'aiuto di *componenti sofisticati* (come l'UART della seriale) o *funzionalmente indispensabili* (come buffer di memoria o buffer di corrente).
Faccio notare come, nel comune gergo dell'elettronica digitale, il termine *buffer* sia utilizzato:

- sia per indicare un *pugno di bits* disponibili a mantenere (*memorizzare*) localmente una certa quantità di livelli logici a disposizione di chi ne abbia bisogno (per esempio per mantenere acceso un *digit a sette segmenti* mentre il controllore è occupato a fare altre cose)
- sia per indicare uno *strato di elettronica* in grado di *garantire la corrente necessaria* ad una periferica, cosa decisamente impossibile per qualunque porta d'uscita del micro

In queste nostre considerazioni ci occuperemo ora della prima tipologia; di un *buffer di memoria* a 8 bit, cioè di un circuito riconoscibile come **interfaccia d'uscita** ad 8 bit.

INTERFACCIA D'USCITA A 8 BIT

Analisi del Problema

I dispositivi programmabili di controllo sono **sempre** dotati di **porte** adatte a trasferire informazioni di tipo **parallelo** ad eventuali periferiche; i *microcontrollori* dispongono di **più di una** porta, di norma a 8 bit, mentre un *personal computer* dispone (ahimè sempre con meno frequenza) di **una sola** porta a 8 bit, *tipicamente* nota come **porta parallela**.

In entrambi i casi ciascuna delle 8 linee può essere programmata per prendere (IN) o dare (OUT) un'unità di informazione (*bit*); in questo secondo caso il *byte* viene lasciato a disposizione della periferica fino a quando il dispositivo di controllo deciderà di sostituirlo con un altro.

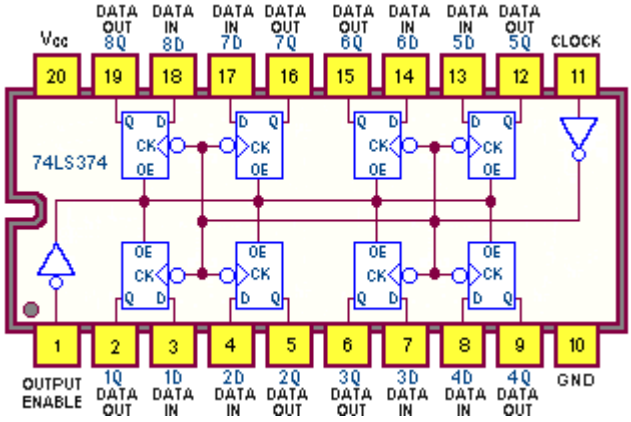
In questo caso, dunque, la presenza di hardware (interfaccia) aggiuntivo esterno **non è necessaria**: il processore che governa la porta (nell'ambito del *PC* o del *single-chip*) sarà chiamato ad eseguire *un paio di istruzioni* (.. in assembly), sufficienti per imprimere *in tempo reale* (qualche nanosecondo) un ben preciso valore binario su ciascuna delle 8 linee

d'uscita, lasciandolo a disposizione fino a nuovo intervento e continuando, subito dopo, ad occuparsi di altre attività.

La presentazione potrebbe dunque finire qui, in attesa di documentare le prossime interfacce a 16, 32 o 64 bit...; invece ho pensato di cogliere l'occasione per descrivere il modo di **memorizzare 8 bit** su una struttura esterna al computer.

Memorizzazione con il componente 74LS374

Il componente più versatile, in questo caso, è il **74LS374**, Ottuplo flip-flop **D-Type** con uscite 3-state; il suo **pin-out** è illustrato dal seguente schema:



Esso contiene **8 elementi di memoria (*flip-flop*)** di tipo ***D-Type***, controllati *contemporaneamente* da un unico piedino di abilitazione (pin **11**, *Clock* CK).

La caratteristica dei flip-flop **D-Type** è quella di agire **sul fronte** della linea di comando, CK:

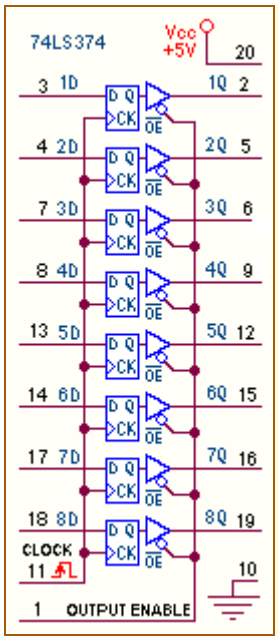
- quando il segnale di controllo (di solito un impulso *attivo basso*) passa dal livello basso **0** a quello alto **1** (*fronte di salita*) l'uscita **Q** *tiene* (memorizza) il valore logico predisposto in precedenza sull'ingresso corrispondente **D**; proprio per questo suo comportamento i manuali definiscono questo componente ***Octal Edge-Triggered Flip-Flops***
- in ogni altro istante della forma d'onda del clock CK su ogni uscita **Q** rimane disponibile il valore logico della *precedente* operazione di memorizzazione

I **D-Type** di questo componente sono del tutto uguali agli elementi di memoria che costituiscono la *cache* del nostro computer, cioè di **Static RAM**, **SRAM**.

Osservando il dettaglio funzionale del componente si capisce perchè la **SRAM** sia più costosa e più veloce della **DRAM** (Dynamic RAM) di solito usata per la memoria convenzionale: ogni bit è realizzato con un piccolo circuito elettronico bistabile, appunto ***flip-flop***.

In aggiunta alla sua tipica funzione di **memoria** il componente offre il servizio di ***buffer di corrente***, cioè dispone di uscite **amplificate *non* invertenti** di tipo ***3-state***, controllate contemporaneamente da un ulteriore piedino di abilitazione (pin **1**, *Out Enable*, OE), *attivo basso*; la funzione di OE è del tutto simile a quella esercitata dalle abilitazioni G**x** del buffer **74LS244**:

- quando è *attiva* (cioè a massa, **0**) consente il passaggio in uscita del livello logico presente sull'uscita interna del *flip-flop*
- quando *non è attiva* (cioè scollegata o posta a **1**) mette in ***alta impedenza*** le 8 uscite esterne del componente; il valore presente sulle uscite dei rispettivi *flip-flop* rimane così *congelato*, in attesa di tempi migliori....



INPUT			OUTPUT
OE	CLOCK CK	xD	xQ
0	↑	1	1
0	↑	0	0
0	0	X	Qo
1	X	X	HI-Z

1 livello logico **alto**
0 livello logico **basso**
Q_o livello logico **assunto dall'ingresso** sul fronte di **salita**

X valore logico **indifferente**
HI-Z alta impedenza, **circuito aperto**

E' importante capire cosa significa mettere un'uscita in ***alta impedenza***; supponiamo di collegare una tensione ad un resistore:

- la corrente che lo attraversa è soggetta alla Legge di Ohm per cui, a parità di tensione applicata, più grande è la sua resistenza e minore è la corrente che passa.
- se la resistenza elettrica è infinita la corrente è nulla, cioè un resistore con questa caratteristica si comporta come un ***circuito aperto***, come se esso stesso fosse sparito, lasciando scollegati i punti ai quali era collegato..

Dunque, la condizione di alta impedenza delle uscite **scollega *praticamente*** l'oggetto ad esse collegato dall'ingresso, **anche se *fisicamente*** la cosa **non si vede**!

Tecnicamente si dice che le uscite di questo componente hanno ***3 stati logici***: i consueti **1** e **0** e il terzo stato, detto **Hi-Z**, appunto "*alta impedenza*".

Nel normale utilizzo come ***memoria*** la possibilità disporre dell'*alta impedenza* delle uscite *non è necessaria*, per cui OE si collega **fisso** a massa (il che assicura comunque la gradita azione di ***buffer di corrente*** su ciascuna linea d'uscita).

Va invece sottolineata la sostanziale differenza con il **74LS373**, Ottuplo flip-flop **D-Latch** con uscite 3-state, con il quale è spesso (a torto) *tecnicamente confuso* e *scambiato*: ne riparleremo *tra un po'.*

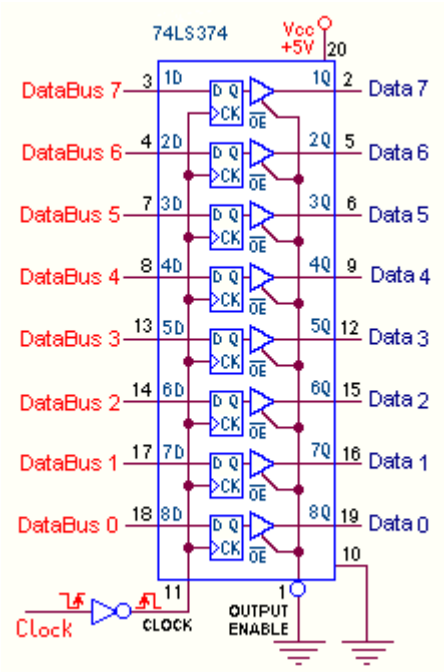
Infine è istruttivo evidenziare un aspetto spesso sottovalutato: il segnale di sincronismo (*clock*) disponibile per la memorizzazione è quasi sempre un segnale **alto** a riposo con un breve impulso **basso** generato al momento desiderato per la memorizzazione; ricordando che la natura **D-Type** (ram statica) dei singoli *flip-flop* del **74LS374** li rende **attivi solo su uno dei due fronti**, *in questo caso* va sottolineato che:

- la memorizzazione avviene *comunque* sul **fronte di salita**.
- in nessun altro istante dell'impulso sarà possibile imporre cambiamenti alle uscite, per cui esse sono *del tutto indifferenti ad*

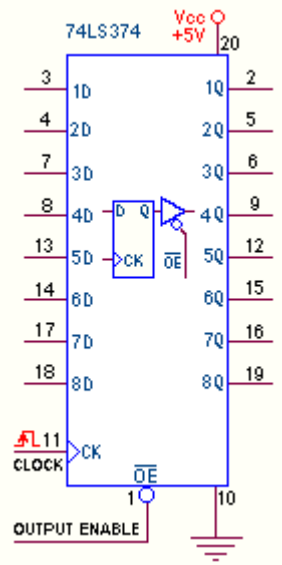
ogni variazione dei rispettivi ingressi, e la *durata* dell'impulso è del tutto **irrilevante**.

- con un impulso di sincronismo *tipico* (cioè attivo basso) se fosse fornito **direttamente** sul pin **11** di *clock* (CK) l'effettiva memorizzazione avverrebbe **con un ritardo** pari alla durata dell'impulso fornito; **è consigliabile** dunque **inserire un *inverter*** sul pin di *clock* (CK) per garantire la memorizzazione esattamente nell'istante in cui il sincronismo *va basso*.
- naturalmente **se** la **durata** dell'impulso è effettivamente **breve** la presenza dell'*inverter* esterno può tranquillamente essere ritenuta un lusso inutile..

Una tipica applicazione (con uscite 3-state trasparenti, cioè con OE a massa) è presentata in figura; le **linee d'ingresso** sono presumibilmente quelle del *Bus Dati* di un microprocessore o di *una porta* di un single-chip o della *porta parallela* di un PC, mentre le **linee d'uscita** rendono disponibile un byte d'informazione utile per accendere *8 led,* o pilotare *8 relè*, o controllare i *7 segmenti (e il decimal point) di un digit*, e così via.



Il disegno precedente suggerisce un possibile **schema a blocchi** da utilizzare nei nostri schemi, sostituibile a piacere con la seguente versione, meno dettagliata ma più compatta:

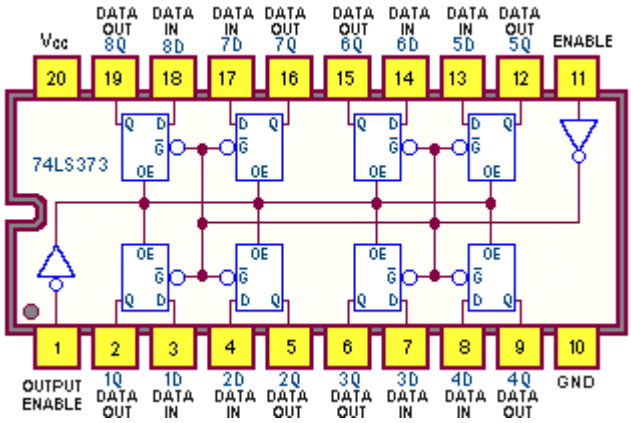


La tabella raccoglie le principali **caratteristiche elettriche** del componente (i tempi sono stati rilevati con carico di 667ohm/ 45 pF):

Caratteristiche Elettriche	Valori
potenza dissipata	**225 mW** (**massima** con ingressi a "**0**" e con uscita **Hi-Z**, assorbe **45 mA** a **5V**)
corrente erogata **tipica** in uscita	**2,6 mA** (con uscita a "**1**")
corrente assorbita **tipica** in uscita	**24 mA** (tipica buffer con uscita a "**0**")
tempo di propagazione **massimo** da ***clock*** a uscita	**28 ns** (verso uscita a "**1**") e **28 ns** (verso uscita a "**0**")
tempo di abilitazione **massimo** da ***out enable*** a uscita	**28 ns** (verso uscita a "**1**") e **28 ns** (verso uscita a "**0**")
tempo di disabilitazione **massimo** da ***out enable*** a uscita	**20 ns** (da uscita a "**1**") e **25 ns** (da uscita a "**0**")

Memorizzazione con il componente 74LS373

Il componente **74LS373**, Ottuplo flip-flop **D-Latch** con uscite 3-state, offre una interessante alternativa al **74LS374**; il suo **pin-out** è illustrato dal seguente schema:



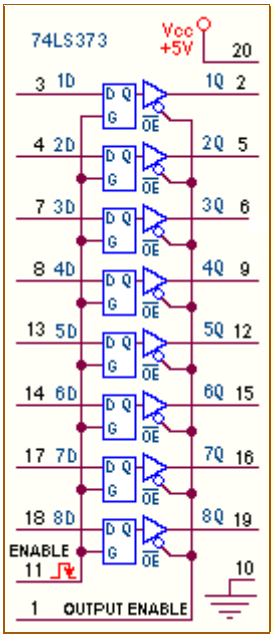
Esso contiene **8 elementi di memoria (*flip-flop*)** di tipo ***D-Latch***, controllati contemporaneamente da un unico piedino di abilitazione (pin **11**, *Enable* G).

La caratteristica dei flip-flop **D-Latch** è quella di agire **sul livello** della linea di comando, G:

- quando G è a livello **1**, scollegato o fissato al positivo dell'alimentazione, il dato **D passa inalterato** sull'uscita corrispondente **Q**, in modo completamente asincrono: ciascun elemento di memoria si comporta come una porta aperta (***l'uscita insegue l'ingresso***), cosicché si dice essere *trasparente* (... come non ci fosse!); proprio per questo suo comportamento i manuali definiscono questo componente ***Octal Transparent Latches***.
- quando G è forzata a massa, **0**, l'uscita **Q** *tiene* (memorizza) il valore logico che aveva l'ingresso corrispondente **D** nel momento del passaggio da un livello (alto) all'altro (basso), cioè sul *fronte di discesa*.

I **D-Latch** di questo componente, con un'immagine figurata, si possono dunque ritenere *porte con memoria*: con l'abilitazione *alta*, sulle uscite passa ogni variazione degli ingressi, fino all'arrivo di un *fronte di discesa*, in corrispondenza del quale l'ultimo dato presente viene mantenuto in uscita per tutto il tempo in cui l'abilitazione rimane *bassa*.

Anche questo componente (come il **74LS374**) aggiunge, alla sua tipica funzione di **memoria**, il servizio di ***buffer di corrente***, dotando le proprie uscite di **amplificatori *non* invertenti** di tipo ***3-state***, controllati da un piedino di abilitazione (pin **1**, *Out Enable*, OE), *attivo basso*; valgono le stesse considerazioni di prima...



INPUT			OUTPUT
$\overline{OE}$	ENABLE G	xD	xQ
0	1	1	1
0	1	0	0
0	0	X	Qo
1	X	X	HI-Z

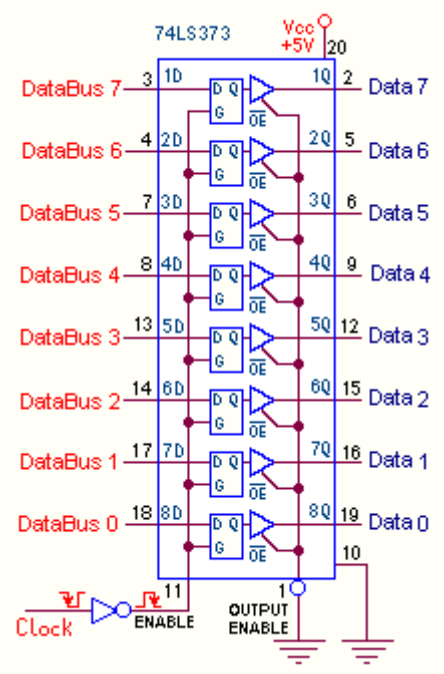
1 livello logico **alto**
0 livello logico **basso**
Q_o livello logico **assunto dall'ingresso** sul fronte di **discesa**

X valore logico **indifferente**
HI-Z alta impedenza, **circuito aperto**

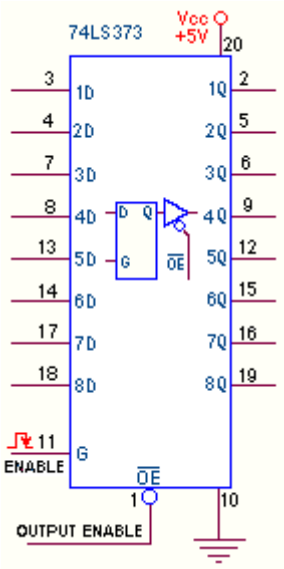
E' certamente importante ed istruttivo evidenziarne l'aspetto funzionale; sempre nell'ipotesi di utilizzare, per la memorizzazione, un *tipico* segnale di sincronismo, **alto** a riposo, portato **basso** per un breve impulso al momento opportuno, ricordando che la natura **D-Latch** dei singoli *flip-flop* del **74LS373**, lo rende **attivo sui livelli**, *in questo caso* va sottolineato che:

- la memorizzazione avviene sul **fronte di discesa**.
- per tutto il tempo in cui l'impulso rimane ***alto*** le uscite sono fortemente *sensibili ad ogni variazione* dei rispettivi ingressi, per cui è assolutamente indispensabile che l'impulso sia **alto e breve** e che i dati in ingresso siano **stabili**, almeno per un tempo un po' più grande dell'impulso positivo (cioè siano forniti *poco prima* e tolti *poco dopo* la sua presenza).
- per entrambe le precedenti ragioni, con impulsi di sincronismo *tipici* (attivi bassi) per un corretto funzionamento del componente **è necessario inserire un *inverter*** direttamente sul pin **11** di *enable* (G).
- la presenza dell'inverter esterno ritarda comunque l'effettiva memorizzazione del tempo pari alla durata (di norma trascurabile) dell'impulso fornito...

La tipica applicazione è quella già proposta per il **74LS374** (con uscite 3-state *trasparenti*, cioè con OE a massa):



Lo **schema a blocchi** del precedente disegno può essere sostituito, nei nostri schemi, con la seguente versione più compatta:



La tabella raccoglie le principali **caratteristiche elettriche** del componente (i tempi sono stati rilevati con carico di 667ohm/ 45 pF):

Caratteristiche Elettriche	Valori
potenza dissipata	**200 mW** (**massima** con ingressi e enable a "**0**" e con uscita **Hi-Z**, assorbe **40 mA** a **5V**)
corrente erogata **tipica** in uscita	**2,6 mA** (con uscita a "**1**")
corrente assorbita **tipica** in uscita	**24 mA** (tipica buffer con uscita a "**0**")
tempo di propagazione **massimo** da ingresso **Dato** a uscita	**18 ns** (fronte di salita) e **18 ns** (fronte di discesa)
tempo di propagazione **massimo** da ***enable*** a uscita	**30 ns** (fronte di salita) e **30 ns** (fronte di discesa)
tempo di abilitazione **massimo** da ***out enable*** a uscita	**28 ns** (verso uscita a "**1**") e **36 ns** (verso uscita a "**0**", 5 pF)
tempo di disabilitazione **massimo** da ***out enable*** a uscita	**20 ns** (da uscita a "**1**") e **25 ns** (da uscita a "**0**", 5 pF)

Considerazioni finali

La lettura attenta degli *ultimi due schemi* di entrambi i componenti, **74LS374** e **74LS373**, usati come **interfaccia**, mette in evidenza la necessità di disporre di almeno una *ulteriore linea di uscita*, in aggiunta alle 8 necessarie per assicurare il dato alla **periferica**, per garantire il segnale di sincronismo necessario alla memorizzazione del dato, da fornire sul pin **11** di entrambi i componenti e inteso come *Clock* (CK) dai **D-Type** del **74LS374** e come *Enable* (G) dai **D-Latch** del **74LS373**.

Essa dovrà essere resa disponibile utilizzando ***un*** *bit* di una seconda *porta* (di un *single-chip*) o ***uno*** dei *segnali di controllo* (della *porta parallela* di un *PC*); modulando *da software* il valore logico presente su questa linea dovrà essere ricostruita la forma d'onda necessaria, in funzione delle specifiche richieste dal componente utilizzato per la memoria, discusse e descritte in precedenza.

In particolare il programma dovrà provvedere a tenere questa linea a *livello basso* (***0*** *logico*) fino al momento nel quale è richiesta la memorizzazione, nel quale verrà portata a ***1*** *logico* per un istante (*impulso*) *molto breve*, per rendere trascurabile il fatto che per il **D-Type 74LS374** sarà attiva sul *fronte di salita* mentre per il **D-Latch 74LS373** sarà attiva sul *fronte di discesa*.

La gestione *da software* del segnale di sincronismo rende quindi inutile la presenza dell'***inverter*** consigliata nella gestione *da hardware* per via del fatto che di solito, per questo scopo, sono disponibili impulsi attivi bassi.

La prossima scheda si occuperà di presentare le migliori e più utili **periferiche** d'uscita, da collegare direttamente alle interfacce descritte in questa occasione e in preparazione per le prossime.

Questo Tutorial è **del tutto originale**, creato e pensato per gli amici di **Grix**
e articolato in numerose *puntate*...

Puoi scaricare qui la versione **PDF** della **PRIMA PARTE**

Alcuni argomenti di questa parte sono **estatti** (ed **adattati** per essa)
dalla *monografia* "**Data Sheet**" della *sezione* "**Dentro il Computer**" del sito

Giobe2000

una ricca raccolta di dispositivi e progetti
pensata per aiutarti a comprendere i segreti del tuo computer
[sia *Personal* che *microcontrollore*]

In particolare sono citati gli argomenti relativi
alla descrizione dei *componenti di memoria* **74LS373** e **74LS374**
http://www.giobe2000.it/HW/DataSheet/pag/**74374**_1.htm
http://www.giobe2000.it/HW/DataSheet/pag/**74373**_1.htm

per i quali sono disponibili anche i relativi Manuali stampabili Gratuiti
(4 pagine, 208 kBytes)
http://www.giobe2000.it/Manuali/DataSheet.htm