Interfacciamento dei processori

Quinta Parte: Le periferiche di visualizzazione a 8 bit [3 di 3]

PREMESSA

Nella ***prima puntata*** ci siamo occupati delle **interfacce d'uscita per un dato a 8 bit** e abbiamo concluso che qualunque dispositivo programmabile (*microcontrollore* o *personal computer*) dispone ***intrinsecamente*** di almeno **almeno una porta** in grado di assicurare questo servizio

Nella ***seconda puntata*** abbiamo studiato le **caratteristiche elettriche tipiche di una porta d'uscita** , analizzando le specifiche di quelle **TTL** compatibili (per altro le più diffuse) e proponendo una tecnica d'*analisi elettronica* facilmente generalizzabile anche per eventuali altre tecnologie. Abbiamo affrontato il concetto di **carico**, e proposto i primi due schemi d'interfaccia, uno con l'utilizzo dei ***buffer di corrente non* invertenti 74LS244** e uno con l'impiego delle **memorie esterne a 8 bit 74LS374** (*batterie* di 8 flip-flop **D-Type** *in parallelo*).

Nella ***terza puntata*** abbiamo affrontato la necessità di disporre di **periferiche adatte ad esprimere l'infomazione**, presentando la **batteria di 8 led** e i **digit a sette segmenti** nelle sue due forme funzionali alternative (a *catodo* e a *anodo comune*); per entrambe è stata introdotta la **Tabella di associazione logica**, una struttura indispensabile per facilitare la creazione del codice di gestione.

Nella ***quarta puntata*** abbiamo descritto le caratteristiche delle prime **interfacce** basate sulla presenza di *dispositivi* in grado di tradurre un **Codice BCD**, fornito loro in ingresso, nei segnali necessari per accendere con sufficiente gradimento i segmenti di **periferiche costituite da due digit** (a *catodo* o ad *anodo comune*); in particolare abbiamo progettato le prime strutture, basate sulla presenza di **Decoder TTL da *BCD* a *sette segmenti***, tipicamente il **74LS47** e il **74LS48**, affrontando e risolvendo anche i problema di evitare di accendere gli eventuali zero a sinistra o a destra del numero proposto sul visualizzatore.

PERIFERICHE DI VISUALIZZAZIONE A 2 DIGIT A 7 SEGMENTI *(segue)*

Premessa

In questa puntata riprendiamo il discorso dell'**interfacciamento** di due **digit**, occupandoci dei due **Decoder** non ancora trattati, il **CMOS 4511** (ancora molto diffuso) e il **TTL 9368** (ahimè disponibile con sempre maggiore difficoltà).

Rispetto a quelli trattati la volta scorsa, entrambi sono dotati di *due **speciali** caratteristiche*: possono **memorizzare il dato** che li attraversa e sono corredati da un importante **driver** d'uscita in grado di assicurare le condizioni ottimali per ciascuno dei **Led** ad essi collegato:

- la prima opzione, nell'ambito in cui il ***decoder*** sarà chiamato ad operare, è sostanzialmente inutile poiché esso assumerà i 4 bit (necessari per esercitare il controllo del suo **digit**) dalla *porta d'uscita* di un *microcontrollore* o dalla *porta parallela* di un *PC*, esse stesse intrinsecamente "**Memoria**"
- decisamente utile, invece, la presenza di uno *stadio di potenza* integrato su ciascuna delle sue linee d'uscita, in grado di assicurare ai *segmenti-led* la *corrente* ideale per accenderli al meglio

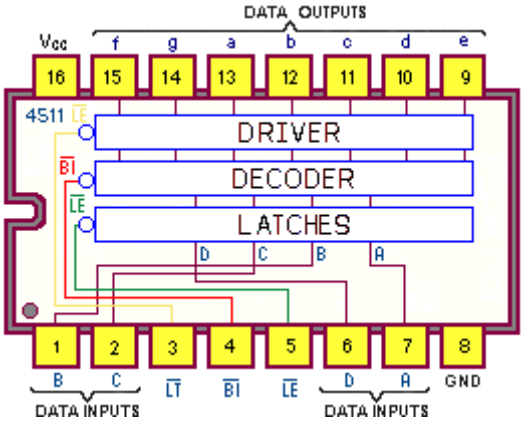
Prima di entrare nel dettaglio di ciascuno di essi, dopo aver evidenziato le loro importanti *specificità* comuni, mi piace sottolineare lo loro *diversità*, altrettanto importanti per le scelte di progetto che saremo chiamati a fare:

- la loro *tecnologia costruttiva* è **diversa** e dovrà essere tenuta presente per garantire un corretto interfacciamento con le *porte* chiamate a supportarli: la natura **TTL** di queste ultime le rende immediatamente compatibili con il **TTL 9368** ma non con il **CMOS 4511**, per il quale sarà necessario qualche intervento aggiuntivo
- il loro modo di interpretare la quaterna di bit assunta dai rispettivi *4 ingressi* offre al ***decoder* TTL** (**9368**) delle *caratteristiche funzionali* **esclusive**: esso è l'unico a poter gestire un ***codice binario puro a 4 bit*** e a garantire (di conseguenza) la possibilità di visualizzare sul suo **digit** tutti e 16 i *simboli* del sistema di numerazione esadecimale (i *numerici* da **0** a **9** e gli *alfabetici* da **A** a **F**); il **CMOS 4511** si comporta invece come tutti gli altri trattati in precedenza, accettando un **Codice BCD** e mostrando quindi incapacità a gestire le *sei combinazioni* più significative di 4 bit

CMOS 4511: DECODER CON MEMORIA DA BCD A SETTE SEGMENTI

Decoder 4511: Caratteristiche generali

Il **Decoder CMOS 4511** converte il **Codice BCD** proposto sulle sue linee d'ingresso generando sulle sue linee d'uscita delle sequenze *attive alte*: i segmenti del **digit** ad esso collegato si accenderanno dunque con uno "**1**" logico per cui questo componente è adatto a **digit** a **catodo comune** (il **Decoder TTL 9368** è una versione ad esso simile, ma in grado di decodificare un ***Codice binario puro a 4 bit***); il suo **pin-out** è illustrato dal seguente schema:



Osservando la figura appare subito evidente che si tratta di un componente speciale, in grado di dare molto di più di quanto ci si aspetta, cioè di svolgere ***tre*** *funzioni in una*:

- consente di *congelare* in una **Memoria** *a 4 bit* le informazioni fornite sulle linee d'*ingresso*
- garantisce ovviamente il suo intrinseco servizio di **Decoder**
- è dotato di **Drivers** su ciascuna delle sette linee d'*uscita*, al fine di fornire le migliori condizioni di pilotaggio ai led ad esse collegati

Decoder 4511: lo strato di Memoria

Prima di raggiungere il **Decoder** le linee d'*ingresso* attraversano una **Memoria *D-Latch***; l'utilizzo di questo tipo di memoria non è molto diffuso e questo è uno dei casi più peculiari; già dalla prima puntata sappiamo che i suoi quattro elementi (*flip-flop*) sono *attivati* **sul livello** della linea di sincronismo (**LE**, ***L**atch **E**nable*, pin**5**), che li controlla contemporaneamente:

- quando **LE** è a *livello* **0**, fissato al negativo dell'alimentazione, i *4 bit* del **Codice BCD** applicato in ingresso **passano inalterati** a valle della memoria, che risulta essere *trasparente* (... come non ci fosse) nei loro confronti, così come fosse una porta aperta (*l'uscita **insegue** l'ingresso*)
- nell'istante in cui **LE** passa da un *livello basso* ad uno *alto* (cioè sul *fronte di salita* di **LE**) le uscite ***D-Latch*** "*scattano*" sul *codice binario* presente su di esse in quel momento, *tenendolo bloccato* (memorizzandolo) fino a quando la linea **LE** verrà riportata a *livello* **0**

Da notare la *notazione logica* che mette in evidenza queste specifiche (graficamente, con la *presenza del pallino* sulla linea d'ingresso, e formalmente, *soprasegnando* il segnale **LE**): in condizioni di funzionamento normale (oppure se l'azione della memoria non è desiderata) questo pin**5** *va posto **a massa***! Si tratta di uno dei più frequenti errori dei giovani virgulti che, dimenticandolo *scollegato*, si disperano perché il **digit** collegato al **Decoder** rimane *bloccato* anche quando è attivo un *conteggio* in ingresso!!

Resta da valutare l'utilità di questa **Memoria**; se questo componente è collegato *a valle* di un **Contatore** potrà disporre in ingresso di un *codice a 4 bit* riproposto ciclicamente dal valore minimo, $(0000)_2$, a quello massimo, $(1001)_2$, con cadenza imposta dalla frequenza dell'*onda quadra* applicata, per esempio ogni secondo se il *clock* è di 1 Hz:

- mantenendo *a massa* l'ingresso **LE** del **Decoder** il **digit** ad esso collegato mostrerà le corrispondenti *cifre decimali*, variabili con la stessa cadenza
- non appena **LE** viene *scollegato da massa* il numero sul **digit** non subirà aggiornamenti, ma il conteggio in ingresso continuerà regolarmente!!
- mantenendo *alto* questo ingresso, ogni volta che si fornisce un *breve impulso basso* il numero sul **digit** cambierà in funzione del codice

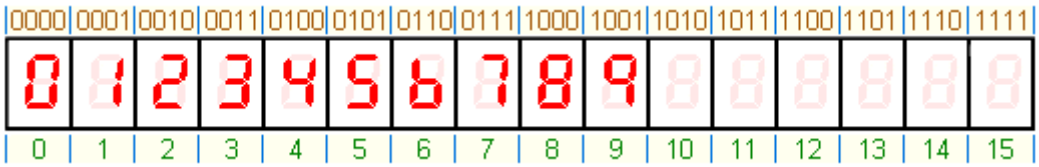
presente in quel momento sugli ingressi **DCBA**, rimanendo invariato fino al successivo *impulso*
- naturalmente il **Contatore** coinvolto nella prova deve essere una **Decade CMOS**, per esempio un **4029**, Binary/Decade up/down counter

Questo artificio torna molto utile in alcune applicazioni, per esempio:

- per generare l'*effetto* ***intertempo*** nel progetto di un ***cronometro digitale***, congelando il *tempo corrente* per qualche istante
- per consentire la *visualizzazione del* ***valore misurato*** da uno ***strumento digitale*** (per esempio un *frequenzimetro*) attivandola solo alla fine di ogni ciclo del conteggio: in questo modo si evita di renderla illeggibile, come sarebbe se il valore fosse mostrato mentre il conteggio è in corso (in sostanza la *misura della frequenza* si esegue contando i fronti attivi del segnale da misurare in un intervallo di grande precisione fissato dalla base dei tempi e mostrandone il valore solo alla fine)

Decoder 4511: la funzione intrinseca di Decoder

Come i suoi simili **TTL** anche questo componente accetta solo le *10 combinazioni* (*codici binari a 4 bit*) appartenenti al **codice BCD**; se, nonostante il divieto, si fornisce una delle *ultime 6 combinazioni vietate*, si otterrà come effetto quello di **spegnere** il **digit** controllato dal **Decoder** (i componenti **TTL** visti finora davano invece corpo a *simboli* sostanzialmente casuali, comunque dovuti alle scelte imposte per il loro progetto). L'aspetto delle uscite è dunque quello mostrato in figura:

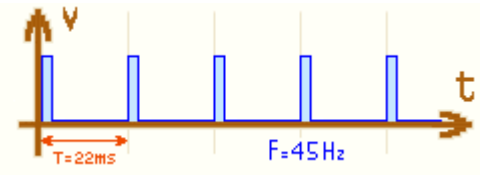


Su questa parte circuitale agisce l'influenza del segnale d'*ingresso* **BI** (***B****lanking* ***I****nput*, pin**4**), *attivo basso*; se si forza uno **0** su **BI** tutte le linee d'*uscita* di segmento sono forzate a **0**, causando lo spegnimento di tutti i

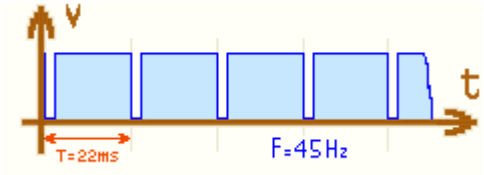
segmenti del **digit**; questa linea di controllo ha un livello di *priorità* secondo solo al segnale d'ingresso **LT** per cui potrà produrre l'effetto descritto solo se **LT** non è contemporaneamente attivo.

Anche con questo componente è possibile controllare l'intensità luminosa del suo **digit** con la tecnica di ***B**lanking* (*spegnimento*) descritta per i *suoi simili* nella puntata precedente: è sufficiente applicare sul pin**4** un'*onda quadra* non simmetrica e modificare il suo *duty cycle*, cioè il rapporto tra la durata della *parte alta* e il periodo del segnale, purché la frequenza sia maggiore di 45 Hz (per il rosso) al fine di evitare la percezione di una sequenza di spegnimenti (imposti da **BI** *attivo*, cioè a **0**) e accensioni (imposti da **BI** *disattivo*, cioè a **1**) e di consentire al nostro occhio (per effetto della *persistenza* dell'immagine sulla retina) di percepire invece una *variazione di luminosità*.

Le figure mostrano il segnale necessario (a) per accendere il **digit** quasi al massimo e (b) per tenerlo quasi spento.



a) *duty cycle* molto basso (es **5**%) treno d'impulsi di *breve durata* molto distanziati tra loro



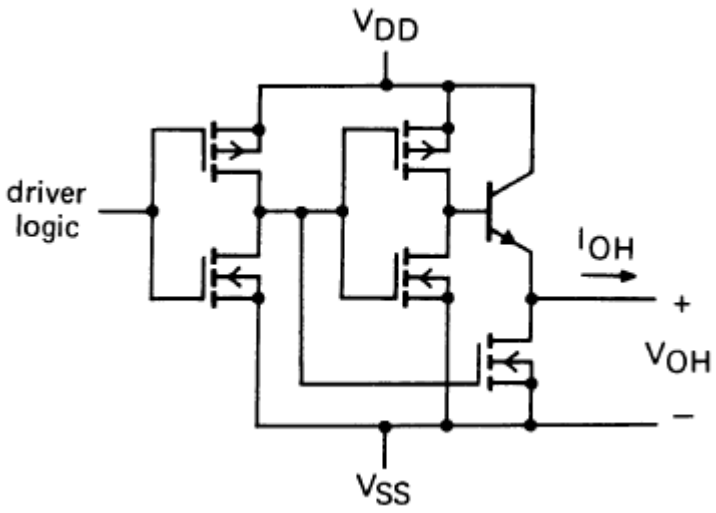
b) *duty cycle* molto alto (es **95**%) treno d'impulsi *molto lunghi* quasi attaccati l'uno all'altro

Decoder 4511: lo strato del Driver

Il terzo strato di hardware è proprio *una gran cosa*: ciascuna delle uscite interne del **Decoder** è dotata di uno ***stadio d'uscita*** realizzato con un ***transistor NPN*** in configurazione *inseguitore d'emettitore* (o *a collettore comune*), una presenza molto *inconsueta* tra i componenti di tipo **CMOS**; le caratteristiche di questo stadio sono quelle di garantire:

- una notevole amplificazione di corrente (intrinseca dei transistor)
- una amplificazione di tensione pressoché uguale a (leggermente minore di) uno, ragione per la quale lo stadio è detto anche *inseguitore di tensione* (*voltage follower*)
- una elevata impedenza di ingresso
- una bassa resistenza d'uscita

Certamente la *situazione ideale* per non ***caricare*** il (***assorbire*** *corrente* dal) circuito **CMOS** che lo pilota e per garantire le migliori condizioni d'uscita, assicurando, *a livello alto* (**1** *logico*), l'***erogazione*** fino a 25mA, più che sufficienti per pilotare direttamente il ***carico*** (ciascuno dei *segmenti-Led* del **digit**).
La figura (estratta dal *data sheet* di HEF**4511**B *MSI BCD to 7-segment latch/decoder/driver*, di proprietà **Philips** *Semiconductors Product*) mostra lo schema di questo ***stadio d'uscita*** evidenziando la presenza del transistor appena descritto:



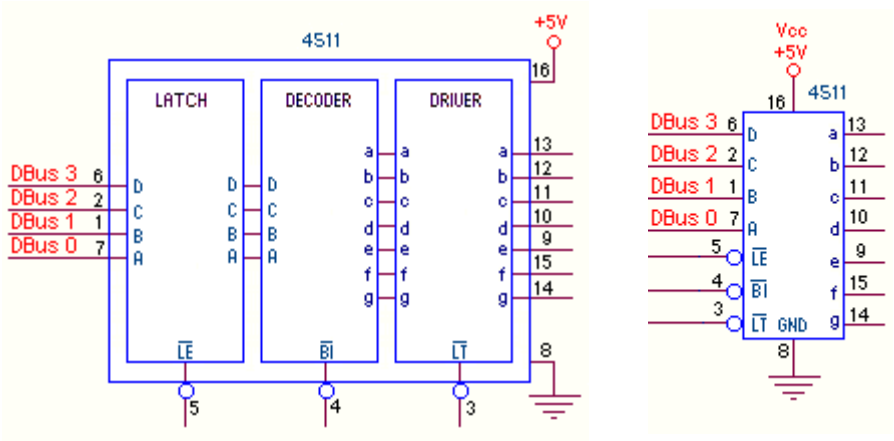
Su questo settore agisce l'influenza del segnale d'*ingresso* **LT** (***L****amp* ***T****est*, pin**3**), *attivo basso*; se si forza uno **0** su **LT** tutte le linee d'*uscita* di segmento sono forzate a **1**, causando l'accensione di tutti i segmenti del **digit**, al fine di verificarne il funzionamento; da notare che questo ingresso è *prioritario* rispetto a tutti gli altri, compresi **LE** e **BI**.

Decoder 4511: considerazioni finali

Le considerazioni sul modo di attivare le *linee di controllo* sono tratte dalla **Tabella di verità** (***Function Table***) di questo **Decoder**, un importante strumento che è bene *saper consultare* e *capire* per la sua alta capacità di sintetizzarne il funzionamento; essa è proposta qui di seguito:

	INGRESSI								USCITE (*attive alte*)						
n°	**LE**	**BI**	**LT**	**D**	**C**	**B**	**A**		**a**	**b**	**c**	**d**	**e**	**f**	**g**
dato	**1**	1	1	X	X	X	X		**dato**, memorizzato sul *fronte di salita* di **LE**						
0	0	1	1	0	0	0	0		1	1	1	1	1	1	0
1	0	1	1	0	0	0	1		0	1	1	0	0	0	0
2	0	1	1	0	0	1	0		1	1	0	1	1	0	1
3	0	1	1	0	0	1	1		1	1	1	1	0	0	1
4	0	1	1	0	1	0	0		0	1	1	0	0	1	1
5	0	1	1	0	1	0	1		1	0	1	1	0	1	1
6	0	1	1	0	1	1	0		0	0	1	1	1	1	1
7	0	1	1	0	1	1	1		1	1	1	0	0	0	0
8	0	1	1	1	0	0	0		1	1	1	1	1	1	1
9	0	1	1	1	0	0	1		1	1	1	0	0	1	1
10	0	1	1	1	0	1	0		0	0	0	0	0	0	0
11	0	1	1	1	0	1	1		0	0	0	0	0	0	0
12	0	1	1	1	1	0	0		0	0	0	0	0	0	0
13	0	1	1	1	1	0	1		0	0	0	0	0	0	0
14	0	1	1	1	1	1	0		0	0	0	0	0	0	0
15	0	1	1	1	1	1	1		0	0	0	0	0	0	0
LT	X	X	**0**	X	X	X	X		1	1	1	1	1	1	1
BI	X	**0**	1	X	X	X	X		0	0	0	0	0	0	0

Lo **schema funzionale** (proposto in due versioni) riassume le caratteristiche logiche di questo componente; le **linee d'ingresso** possono essere 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 disponibili i sette valori logici *attivi alti* necessari per controllare i *7 segmenti* di un **digit** a **catodo comune**:

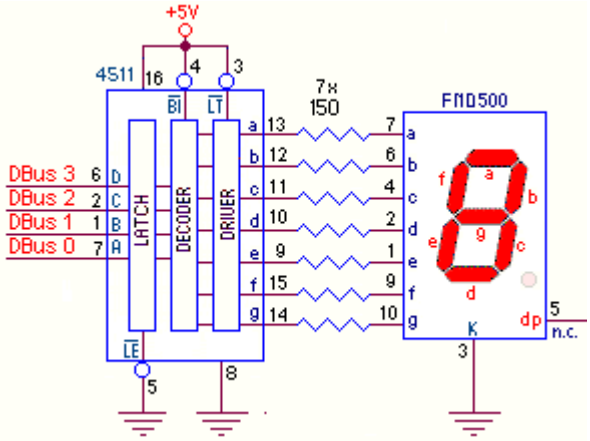


Per le scelte di progetto non va dimenticato che sulle uscite di segmento sono presenti valori **V_{OH}** (*tensione d'uscita* a *livello alto*) tipici della natura **CMOS** del **Decoder 4511**; collegando su esse un normale **digit** a **catodo comune** è consigliabile la presenza di **resistori** in serie ai *segmenti-led*, necessari per limitare la corrente a valori tali da garantirne una buona accensione in sicurezza; il valore della loro *resistenza* dipende dalla *tensione d'alimentazione* e dalla *corrente* ***erogata***.

V_{CC}	I_{OH}	V_{OH}	R (ohm)
5V	0mA	4,40V	
	5mA	4,20V	480
	10mA	4,05V	225
	15mA	4,00V	**147**
	20mA	3,80V	100
	25mA	3,70V	76
10V	0mA	9,40V	
	5mA	9,20V	1,48k
	10mA	9,10V	730
	15mA	9,00V	167
	20mA	9,00V	360
	25mA	8,90V	284
15V	0mA	14,40V	
	5mA	14,20V	2,48k
	10mA	14,10V	1,23k
	15mA	14,00V	813
	20mA	14,00V	610
	25mA	14,00V	488

L'interessante tabella della pagina precedente, relativa alle uscite *in condizioni* di funzionamento *tipiche*, è stata ricostruita consultando il *data sheet*.

Con *tensione d'alimentazione* di +5V, supponendo di imporre una *corrente* ***erogata*** di 15mA risulta disponibile una tensione V_{OH} di 4V che, al netto della caduta di 1,8V sul **Led**, porta ad un valore di *resistenza* pari a 147ohm , normalizzato a quello standard di 150ohm:



Nel normale funzionamento, il *piedino di controllo* **LE** (pin**5**) della ***memoria latch*** deve essere *a massa* mentre i rimanenti due, **BI** (pin**4**) e **LT** (pin**3**), possono essere *inattivi*, collegandoli al positivo dell'alimentazione, data la natura **CMOS** del componente, che non ammette pin *fluttuanti*!

Può essere, infine, istruttivo consultare le pagine dei ***data sheet*** *originali*; una copia è disponibile qui: **4511** [Philips].

Per amore di completezza, la serie **CMOS** prevede altri integrati con disponibilità di **Decoder** da **BCD** a ***7 segmenti***:

- i **4026** e **4033** (**4026 e 4033** [Texas]) sono *contatori/divisori decadici* con uscite decodificate
- i **4055** e **4056** (**4055 e 4056** [Texas]) e il **4543** (**4543** [Texas]) sono *latch/decoder/driver* espressamente progettati per visualizzatori a Cristalli liquidi (LCD)

INTERFACCIA CON DECODER CMOS PER VISUALIZZATORE A 2 DIGIT CON INGRESSO BCD

Il **Decoder CMOS 4511** dispone di linee d'uscita *attive alte*, adatte al controllo dei *7 segmenti* di **digit** a **catodo comune**; la disponibilità di uscite bufferizzate (per la presenza di **driver** su di esse) consiglia la presenza di **resistori** in serie da 150ohm, per limitare la *corrente **erogata*** a 15mA (con alimentazione di +5V).

Le due linee *d'ingresso* (entrambe *attive basse*) **LT** (***L**amp **T**est*, che accende di tutti i segmenti) e di **BI** (***B**lanking **I**nput*, che invece li spegne tutti incondizionatamente) non sembrano particolarmente utili in un ambito programmabile, per cui sono rese *inattive* collegandole a +5V: se per la logica **TTL** si tratta di una buona abitudine, per i pin *fluttuanti* **CMOS** diventa un obbligo!

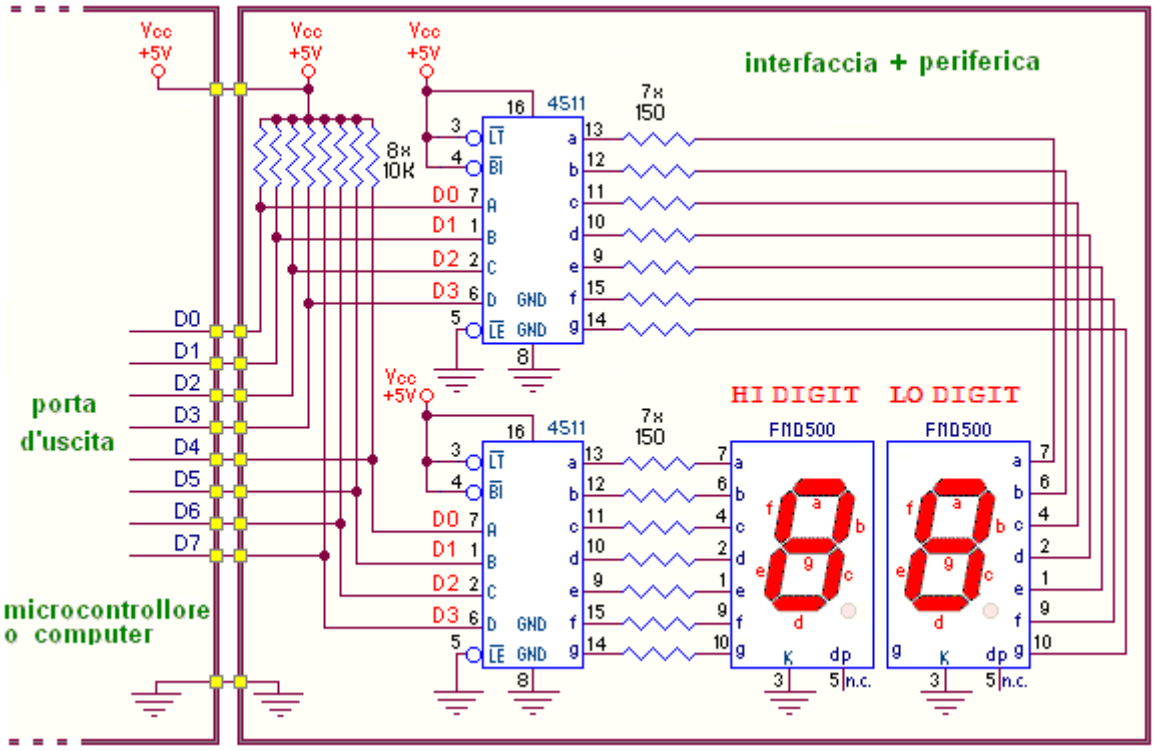
Stesso trattamento per la linea *d'ingresso* **LE**, ***L**atch **E**nable*: certamente la presenza di questa **Memoria** è del tutto irrilevante, in questa applicazione, essendo *un doppione* ... (la stessa *porta d'uscita* di un *micro* o del *PC* è una **Memoria**); per renderla *trasparente* è sufficiente mantenerla *a massa*.

Rispetto agli schemi d'**interfaccia** presentati precedentemente c'è ora un fatto nuovo: sia la *porta d'uscita* di un *microcontrollore* o la *porta parallela* di un *PC* è di natura **TTL** mentre il **Decoder** intermediario è **CMOS**; questo fatto introduce incompatibilità logica e va risolto sulla base di queste considerazioni:

- i segnali *a livello **basso*** in *uscita* dalla *porta* **TTL non creano problemi** agli *ingressi di dato* del **Decoder CMOS**: tipicamente la $V_{OL(TTL)}$ è di 0,2V, **inferiore** alla $V_{IL(CMOS)}$, compresa tra 0V e Vcc/3 (= 1,66V con alimentazione di +5V)
- i segnali *a livello **alto*** in *uscita* dalla *porta* **TTL possono essere incompatibili** con gli *ingressi di dato* del **Decoder CMOS**: tipicamente la $V_{OH(TTL)}$ è compresa tra 2,4V e 4V, mentre la $V_{IH(CMOS)}$ richiesta per il riconoscimento del *livello alto* è compresa tra 2Vcc/3 (= 3,33V con alimentazione di +5V) e Vcc

Per avere la certezza di un corretto interfacciamento si rende necessaria la presenza di **resistori** di *pull-up* su ciascuno degli 8 ingressi **CMOS**: a *livello alto* ciascun **resistore**, non essendo attraversato da corrente, non subisce *caduta di tensione* ai suoi capi, cioè anche il suo terminale opposto

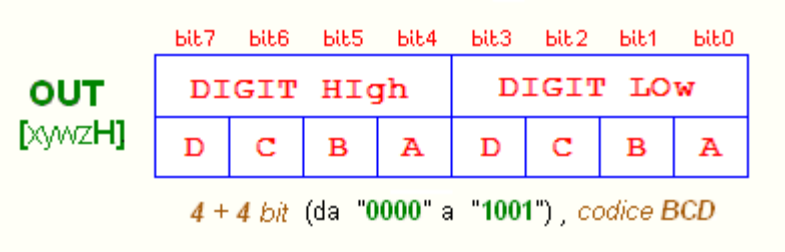
all'alimentazione è a +5V, risolvendo il problema del riconoscimento del livello alto da parte degli *ingressi* **CMOS**.



Il programma che controllerà questa **interfaccia** dovrà farsi carico di formattare in **BCD** i dati binari da spedire sulla *porta principale*: ognuna delle ***6*** *combinazioni* non appartenente al **codice** [da $(1010)_2$ a $(1111)_2$] provoca in questo caso lo spegnimento di tutti i segmenti del digit; sulla base di questa caratteristica si rende disponibile l'inattesa possibilità di controllare il ***B****lanking* da software, senza la necessità di dover intervenire da hardware su **BI**.

Da notare che questo progetto non consente la **gestione diretta** dei *punti decimali* dei due **digit**; all'occorrenza è comunque possibile associarne il controllo a due linee di una seconda *porta d'uscita* (con *microcontrollore*) o due linee delle quattro linee del **Registro 037AH / 027AH** (con la *porta parallela* del *PC*, vedi http://www.giobe2000.it/HW/Parallela/Pag/RegistriSPP4.htm).

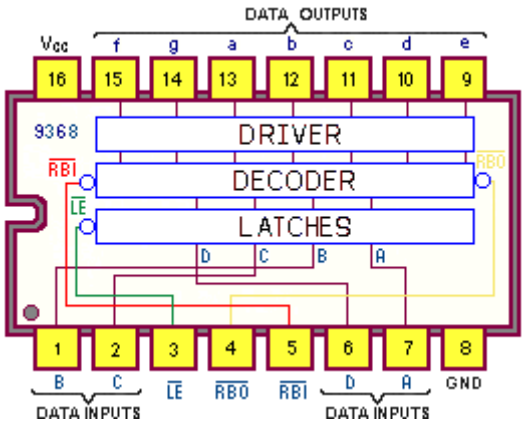
La seguente **tabella di associazione logica** è chiamata a *virtualizzare* l'utilizzo del progetto proposto:



TTL 9368: DECODER CON MEMORIA DA BINARIO A 4 BIT A SETTE SEGMENTI

Decoder 9368: Caratteristiche generali

Il **Decoder TTL 9368** converte il ***codice binario puro a 4 bit*** proposto sulle sue linee d'ingresso generando sulle sue linee d'uscita delle sequenze *attive alte*: i segmenti del **digit** ad esso collegato si accenderanno dunque con uno "**1**" logico per cui questo componente è adatto a **digit** a **catodo comune**.
Esso è *pin-out compatibile* con il **Decoder TTL 74LS48** ed è *funzionalmente* simile al **Decoder CMOS 4511**, entrambi per altro destinati a decodificare un **Codice BCD**; il suo **pin-out** è illustrato dal seguente schema:



Osservando la figura appare subito evidente che si tratta di un componente speciale, in grado di dare molto di più di quanto ci si aspetta, cioè di svolgere ***tre*** *funzioni in una*:

- consente di *congelare* in una **Memoria** *a 4 bit* le informazioni fornite sulle linee d'*ingresso*
- garantisce ovviamente il suo intrinseco servizio di **Decoder**
- è dotato di **Drivers** sulle sette linee d'*uscita*, al fine di fornire le migliori condizioni di pilotaggio ai led ad esse collegati

NB: Le descrizioni proposte in seguito per questo componente sono solo apparentemente simili a quelle fornite, in precedenza, per il il **4511**: l'impostazione dei testi è sostanzialmente la stessa ma differisce in numerosi piccoli particolari; per evitare imprecisioni ho ritenuto più corretto ripetere i principali concetti, adattandoli alla nuova realtà, piuttosto che fare dei rimandi *alla fine* inopportuni.

Decoder 9368: lo strato di Memoria

Come per il **CMOS 4511** le linee d'*ingresso* incontrano anzitutto una **Memoria *D-Latch***, prima di raggiungere il **Decoder** vero e proprio; i suoi quattro elementi (*flip-flop*) sono *attivati* **sul livello** della linea di sincronismo (**LE**, ***L****atch* ***E****nable*, pin**3**) che li controlla contemporaneamente:

- quando **LE** è a *livello* **0**, fissato al negativo dell'alimentazione, i *4 bit* del ***codice binario puro*** applicato in ingresso **passano inalterati** a valle della memoria, che risulta essere *trasparente* (... come non ci fosse) nei loro confronti, come fosse una porta aperta (*l'uscita* ***insegue*** *l'ingresso*)
- nell'istante in cui **LE** passa da un *livello basso* ad uno *alto* (cioè sul *fronte di salita* di **LE**) le uscite ***D-Latch*** "*scattano*" sul *codice binario* presente su di esse in quel momento, *tenendolo bloccato* (memorizzandolo) fino a quando la linea **LE** verrà riportata a *livello* **0**

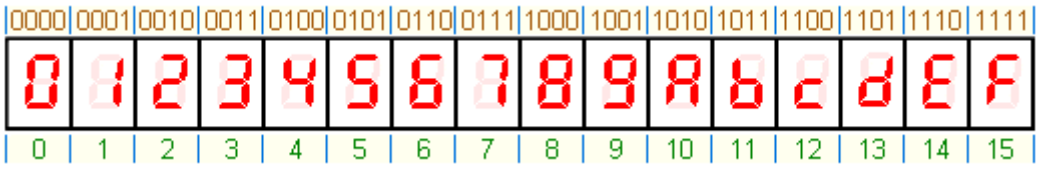
La *notazione logica* (la *presenza del pallino* sulla linea d'ingresso e il *soprasegno* nel nome del segnale. **LE**) mette in evidenza le condizioni del suo funzionamento: se l'azione della memoria non è desiderata questo pin**3** *va posto* ***a massa***! Ho già ricordato che, dimenticandolo *scollegato*, qualora il **Decoder** sia coinvolto in un conteggio il **digit** ad ed esso collegato rimane inesorabilmente *bloccato*, generando panico in chi è incorso in questa disattenzione!

L'utilità di questa **Memoria** si può evidenziare collegando questo componente *a valle* di un **Contatore** in grado di proporre in ingresso un *codice a 4 bit* riproposto ciclicamente, dal valore minimo $(0000)_2$ a quello massimo $(1111)_2$, con cadenza imposta dalla frequenza dell'*onda quadra* applicata, per esempio ogni secondo se il *clock* è di 1 Hz:

- mantenendo *a massa* l'ingresso **LE** del **Decoder** il **digit** ad esso collegato mostrerà le corrispondenti *cifre esadecimali*, variabili con la stessa cadenza
- non appena **LE** viene *scollegato da massa* il numero sul **digit** non subirà aggiornamenti, ma il conteggio in ingresso continuerà regolarmente
- mantenendo *alto* questo ingresso, ogni volta che si fornisce un *breve impulso basso* (tipicamente di durata di almeno di 30ns) il numero sul **digit** cambierà in funzione del codice presente in quel momento sugli ingressi **DCBA**, rimanendo invariato fino al successivo *impulso*
- naturalmente il **Contatore** coinvolto nella prova deve essere **Binario TTL**, per esempio un **74LS93**, high-speed 4-bit ripple type counters

Decoder 9368: la funzione intrinseca di Decoder

Unico tra tutti i **Decoder** conosciuti, questo componente interpreta tutte le possibili *16 combinazioni* (*codici binari a 4 bit*) assicurando per ciascuna di esse un *simbolo significativo*; l'aspetto delle uscite è dunque quello mostrato in figura:



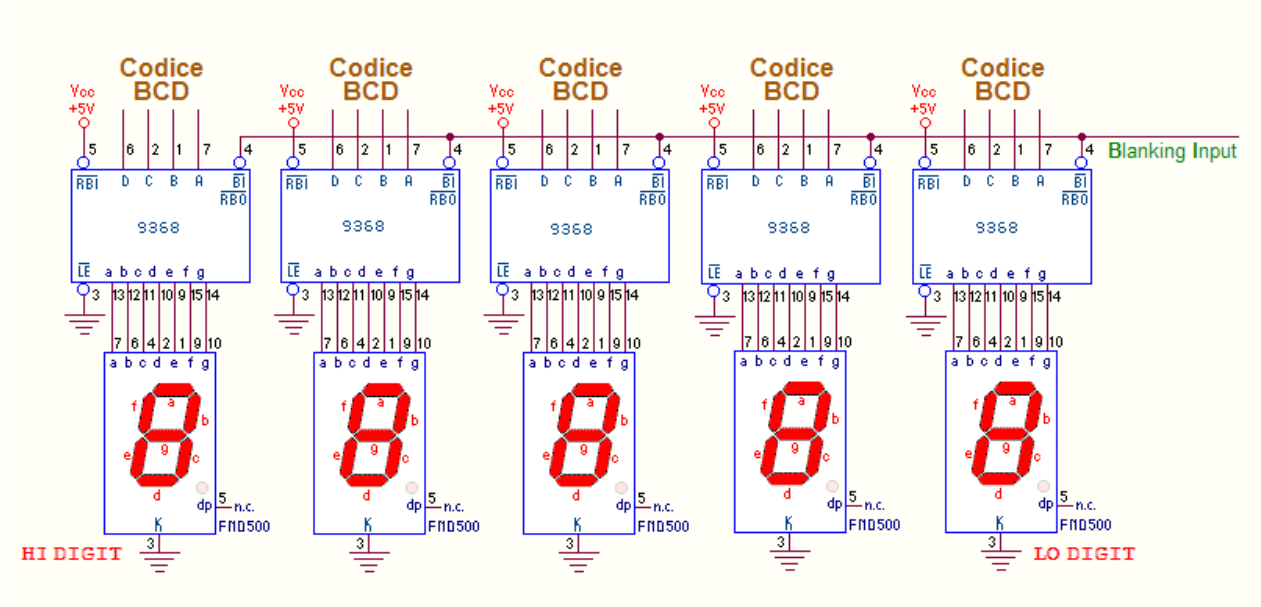
La grande novità consiste nella visualizzazione dei *simboli alfabetici* da **A** a **F** (proposte con lettere minuscole o maiuscole) in aggiunta ai *simboli numerici* da **0** a **9** (già disponibili con gli altri **Decoder**).

Su questa parte circuitale agisce l'influenza dei segnali **BI** / **RBO** (***B**lanking **I**nput* / ***R**ipple **B**lanking **O**utput*, pin**4**) e **RBI** (***R**ipple **B**lanking **I**nput*, pin**5**), entrambi *attivi bassi*.

La linea collegata al pin**5** è decisamente *d'ingresso* mentre quella collegata al pin**4** dispone di una circuiteria interna (una logica cablata di tipo ***wired-AND***) che permette di disporre sia del *segnale d'uscita* **RBO** che di forzare su essa un *segnale d'ingresso* **BI**, senza produrre danni.

Questo componente permette di **spegnere incondizionatamente** il **digit** collegato al **Decoder**, con qualunque codice presente sugli ingressi **DCBA**, *attivando* (forzando *a* **0**) la linea collegata al pin**4**, utilizzata come ingresso (**BI**, ***B****lanking* ***I****nput*), in modo identico a quello descritto la puntata scorsa per il **Decoder TTL 74LS48**, come detto con esso *pin-out compatibile* anche se *funzionalmente* molto diverso. Per motivi di chiarezza sono qui riproposte le medesime considerazioni.

Nel funzionamento normale l'ingresso **BI** deve essere lasciato *scollegato* (o collegato al +5V), cioè *disattivo* (forzato *a* **1**); la figura mostra la predisposizione per il controllo di **BI** su un visualizzatore a 5 cifre, con tutte le memorie *trasparenti* (cioè con tutti gli ingressi **LE** (***L****atch* ***E****nable*, pin**3**) *a massa*; da notare che l'ingresso **BI** è *prioritario* rispetto a tutti gli altri, cioè quando esso è *attivo* contemporaneamente ad ogni altro ingresso la funzione degli altri viene ignorata.



Anche in questo caso, applicando un'*onda quadra* non simmetrica sull'ingresso **BI** e modificandone il *duty cycle*, è possibile controllare l'intensità luminosa del **digit** ad esso collegato, come descritto in precedenza.

L'ingresso RBI e l'uscita RBO

L'utilizzo di queste due linee di controllo è irrilevante se il **Decoder TTL 9368** è chiamato ad operare con la *porta d'uscita* di un *micro* o di un *PC*; la volontà di spiegarne comunque il funzionamento è frutto della consapevolezza di affrontare un argomento spesso difficile da reperire.

In sostanza si tratta dello stesso meccanismo (già descritto la puntata scorsa per il più economico **74LS48**) per **evitare di accendere** tutti gli **zero non significativi** nell'ambito di un visualizzatore **decimale** con numerose cifre, sia **a sinistra** che **a destra** (se il numero mostrato è frazionario); va subito notato che se il **9368** è utilizzato nel pieno della sua *specificità* (unico componente in grado di mostrare *numeri* **esadecimali**) questo servizio è comunque *illogico*, sia per il fatto che ogni cifra del numero mantiene la sua necessità d'esistere (anche gli *zero a sinistra*, basti pensare ad un indirizzo o un dato a 16 bit, come **2AF5** o come **001E**) sia per l'improbabilità di dover mostrare un numero esadecimale frazionario (rendendo così inutile anche il servizio per gli *zero a destra*).

Ciononostante voglio riprendere i concetti già descritti per evitare di dover cercarli altrove, in caso di necessità; tra l'altro gli schemi sono ***diversi*** rispetto a quelli proposti per il economico **74LS48**, per l'assenza della linea di **Lamp Test** ora sostituita da quella di **Latch Enable**.

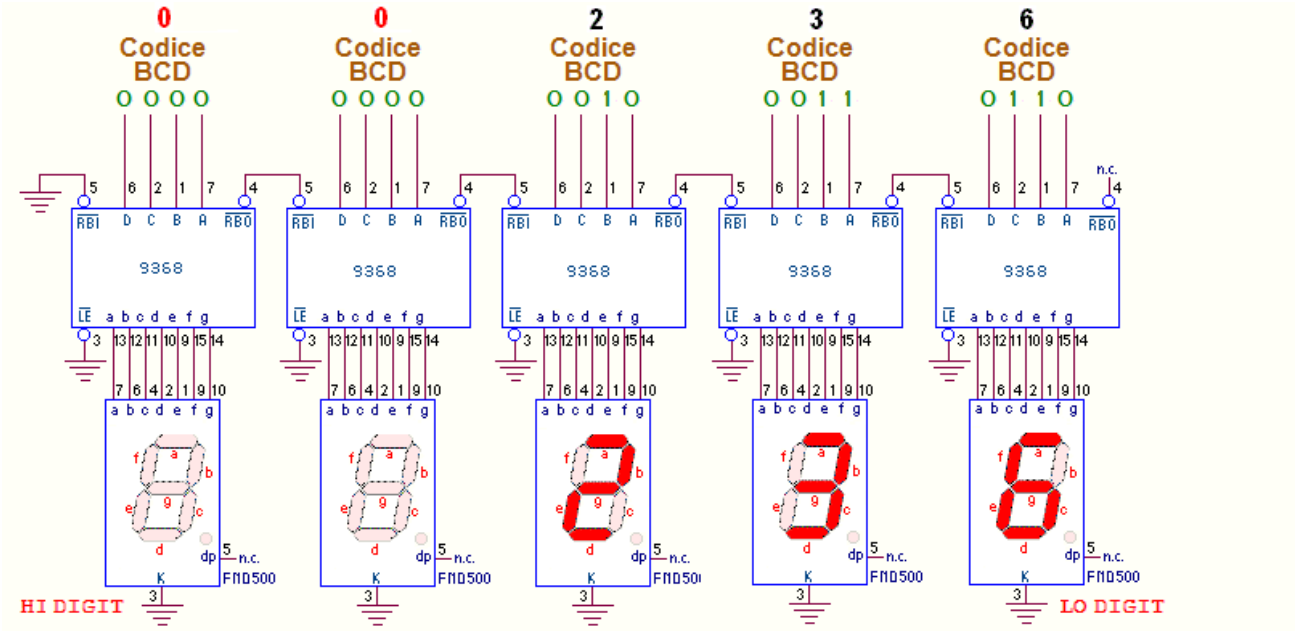
Ricordo che l'ingresso **RBI**, se *attivato* (forzato *a livello logico* **0**):

- consente di **spegnere** (*blanking*) il **digit** collegato al **Decoder** solo se contemporaneamente il codice presente sugli ingressi **DCBA** è $(0000)_2$; in queste condizioni forza a **0** anche l'uscita **RBO**
- se il codice presente sugli ingressi **DCBA** è diverso da $(0000)_2$ il **digit** mostra regolarmente i *simboli* corrispondenti e l'uscita **RBO** rimane *disattiva* (cioè *a livello logico* **1**)

La condizione per poter visualizzare comunque la cifra ***zero*** è di lasciare l'ingresso **RBI** *scollegato* (o meglio collegato al +5V), cioè *disattivo* (forzato *a livello logico* **1**).

L'uscita **RBO** è il meccanismo che consente di **evitare di accendere** tutti gli **zero a sinistra** di un numero, decisamente antiestetici e inutili; anche nella comune aritmetica sembra logico scrivere **236** piuttosto che **00236**; il metodo per raggiungere questo scopo è quello di collegare a massa

l'ingresso **RBI** del **Decoder** collegato al **digit** *più significativo* e di collegare in cascata l'uscita **RBO** di ciascun digit (a cominciare da quello *più significativo*) con l'ingresso **RBI** del digit *successivo* (subito a *destra*):

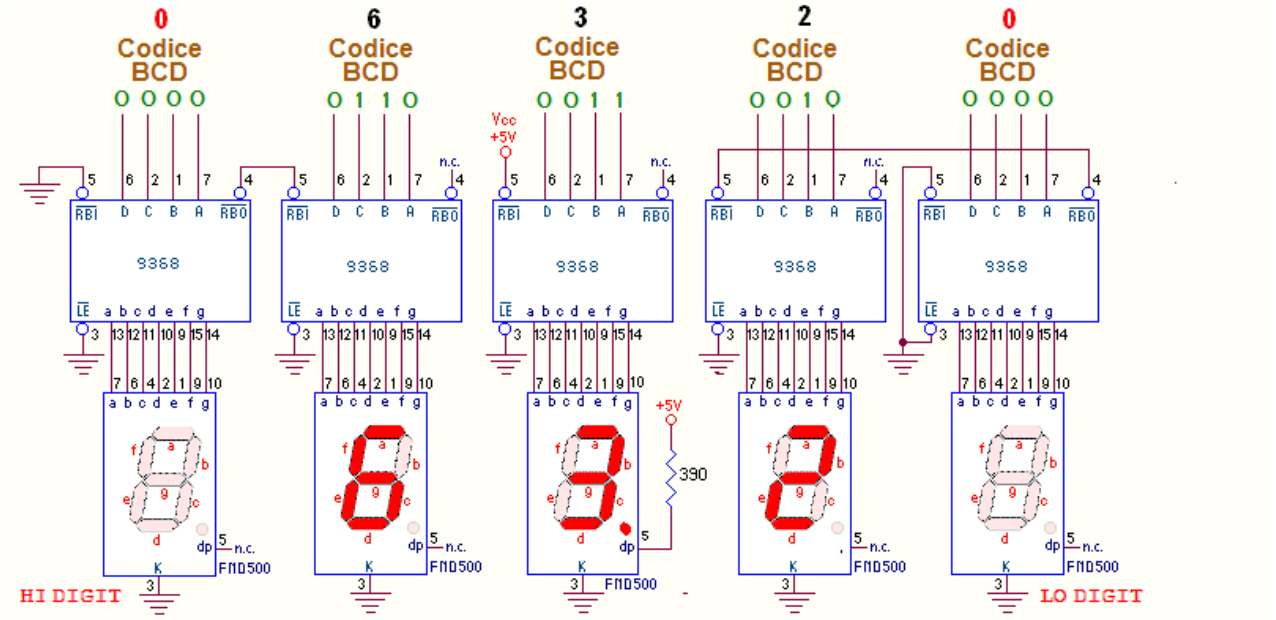


Nell'ipotesi di utilizzare il **9368** nell'ambito di un visualizzatore **decimale** è scontato ritenere che l'informazione fornita in ingresso ai cinque **Decoder** sia di tipo **BCD**, anche se essi sarebbero in grado di processare tutte le 16 possibili combinazioni; la figura precedente mostra un visualizzatore chiamato a mostrare il *numero intero* **00235**:

- fissando *a massa* l'ingresso **RBI** del **Decoder** collegato al **digit** *più significativo* se (come nel nostro esempio) sui suoi ingressi **DCBA** è presente del codice $(0000)_2$, si evita la visualizzazione dello 0 su di esso (***B**lanking* della cifra)
- questa situazione forza *a* **0** anche l'uscita **RBO**; essa a sua volta forza *a massa* l'ingresso **RBI** del **Decoder** collegato al **digit** *successivo* (subito a *destra*) evitando la visualizzazione dello 0 se sui suoi ingressi **DCBA** è presente del codice $(0000)_2$
- poiché il nostro esempio prevede *proprio questo* anche il secondo **digit** (subito a *destra* del primo) rimane spento e, in coerenza con la regola, il **Decoder** ad esso associato forza *a* **0** anche la sua uscita **RBO**; essa a sua volta forza *a massa* anche l'ingresso **RBI** del **Decoder** collegato al **digit** *successivo* (il terzo da sinistra)

- poiché il codice presente sugli ingressi **DCBA** di questo **Decoder** ***non*** è $(0000)_2$ il fatto che il suo ingresso **RBI** sia *a livello logico* **0** è ora del tutto irrilevante e, da questa posizione in poi ogni cifra verrà visualizzata, anche gli eventuali 0 interni

La figura seguente mostra invece lo stesso visualizzatore chiamato a mostrare il *numero frazionario* **063,20**:



Ripetiamo l'analisi del visualizzatore precedente, ora organizzato per visualizzare la *parte intera* di un numero sui primi tre **digit** e la *parte fazionaria* sugli ultimi due; la situazione è evidenziata dalla presenza del *punto decimale* acceso permanentemente sul **digit** centrale. L'ipotesi di lavoro è che i **codici BCD** in ingresso ai cinque **Decoder** siano ora quelli corrispondenti alle cifre **06320**; la gestione della *parte intera* è identica a quella già descritta:

- il **Decoder** associato al **digit** *più significativo* ha l'ingresso **RBI** *a massa* perciò se (come nel nostro esempio) sui suoi ingressi **DCBA** è presente del codice $(0000)_2$, la visualizzazione dello 0 viene evitata (***B****lanking* della cifra) e la sua uscita **RBO** viene forzata *a massa*
- poiché la citata uscita **RBO** è collegata all'ingresso **RBI** del **Decoder** associato al **digit** *successivo* (subito a *destra*), l'eventuale presenza

sui suoi ingressi **DCBA** del codice $(0000)_2$ eviterà anche per esso la visualizzazione dello 0
- il **Decoder** del **digit** *centrale* (quello con *punto decimale* acceso) non è soggetto a nessun controllo **RBI** per cui mostrerà qualunque cifra decimale, compreso lo 0

La gestione della *parte fazionaria* riflette la stessa logica ma, dovendo evitare l'accensione degli **zero a destra** del numero, il controllo dovrà essere esercitato a partire dal **digit** *meno significativo*:

- fissando *a massa* l'ingresso **RBI** del **Decoder** collegato al **digit** *meno significativo* se (come nel nostro esempio) sui suoi ingressi **DCBA** è presente del codice $(0000)_2$, la visualizzazione dello 0 su di esso viene evitata
- questa situazione forza *a* **0** anche l'uscita **RBO** che, a sua volta, forza *a massa* l'ingresso **RBI** del **Decoder** collegato al **digit** *precedente* (subito a *sinistra*) evitando la visualizzazione dello 0 se sui suoi ingressi **DCBA** è presente del codice $(0000)_2$

Decoder 9368: lo strato del Driver

La terza parte di questo componente è decisamente interessante: ciascuna delle uscite interne del **Decoder** è dotata potente **Driver** in grado di ***erogare*** verso i *segmenti-led* una *corrente* $\mathbf{I_{OH}}$ da 16mA (*minima*) a 22mA (*massima*), senza bisogno di **resistori** per limitare la corrente; la *tensione nominale* d'uscita *a livello alto*, $\mathbf{V_{OH}}$, è 3,4V (secondo il *data sheet*) ma, con un **led** ad essa direttamente collegato, si adatterà al valore della caduta ai suoi capi, cioè da 1,75V a 1,85V.

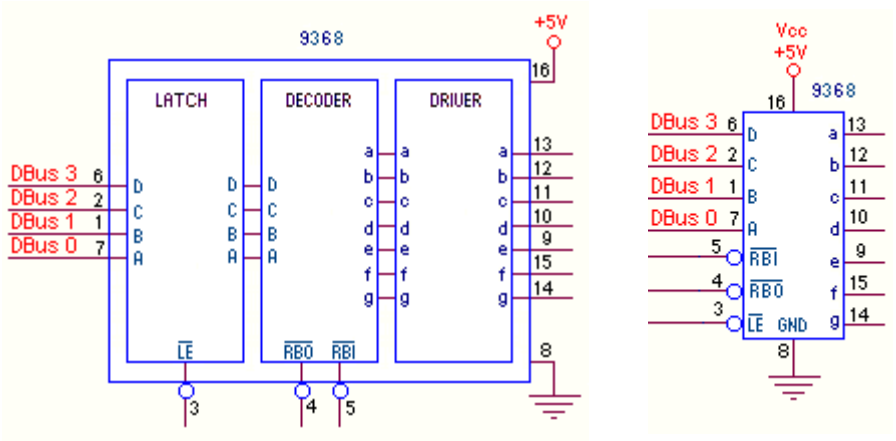
Decoder 9368: considerazioni finali

Le considerazioni sul modo di attivare le *linee di controllo* sono tratte dalla **Tabella di verità** (***Function Table***) di questo **Decoder**: ribadisco

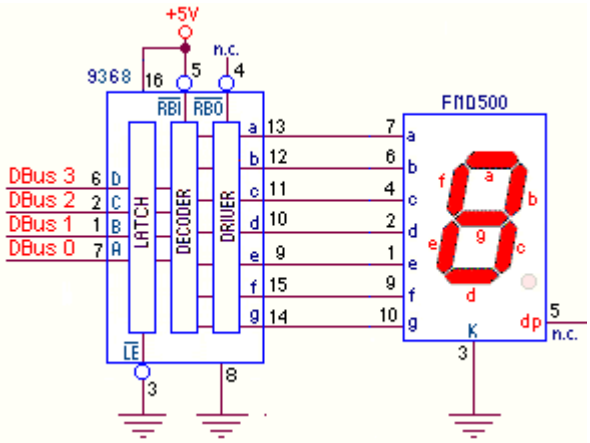
l'importanza di *saper consultare* e *capire* questo importante strumento, proposto qui di seguito:

	INGRESSI						USCITE (*attive alte*)							
n°	**LE**	**RBI**	**D**	**C**	**B**	**A**	**RBO**	**a**	**b**	**c**	**d**	**e**	**f**	**g**
dato	**1**	X	X	X	X	X	1	**dato**, memorizzato sul *fronte di salita* di **LE**						
0	0	**0**	0	0	0	0	0	0	0	0	0	0	0	0
		1	0	0	0	0	1	1	1	1	1	1	1	0
1	0	X	0	0	0	1	1	0	1	1	0	0	0	0
2	0	X	0	0	1	0	1	1	1	0	1	1	0	1
3	0	X	0	0	1	1	1	1	1	1	1	0	0	1
4	0	X	0	1	0	0	1	0	1	1	0	0	1	1
5	0	X	0	1	0	1	1	1	0	1	1	0	1	1
6	0	X	0	1	1	0	1	0	0	1	1	1	1	1
7	0	X	0	1	1	1	1	1	1	1	0	0	0	0
8	0	X	1	0	0	0	1	1	1	1	1	1	1	1
9	0	X	1	0	0	1	1	1	1	1	0	0	1	1
10	0	X	1	0	1	0	1	1	1	1	0	1	1	1
11	0	X	1	0	1	1	1	0	0	1	1	1	1	1
12	0	X	1	1	0	0	1	1	0	0	1	1	1	0
13	0	X	1	1	0	1	1	0	1	1	1	1	0	1
14	0	X	1	1	1	0	1	1	0	0	1	1	1	1
15	0	X	1	1	1	1	1	1	0	0	0	1	1	1
BI	X	X	X	X	X	X	**0**	0	0	0	0	0	0	0

Lo **schema funzionale** (proposto in due versioni) riassume le caratteristiche logiche di questo componente; le **linee d'ingresso** possono essere 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 disponibili i sette valori logici *attivi alti* necessari per controllare i *7 segmenti* di un **digit** a **catodo comune**:



Come preannunciato, nell'impiego del **Decoder 9368** con un normale **digit** a **catodo comune** la presenza di **resistori** in serie ai *segmenti-led* non è necessaria:



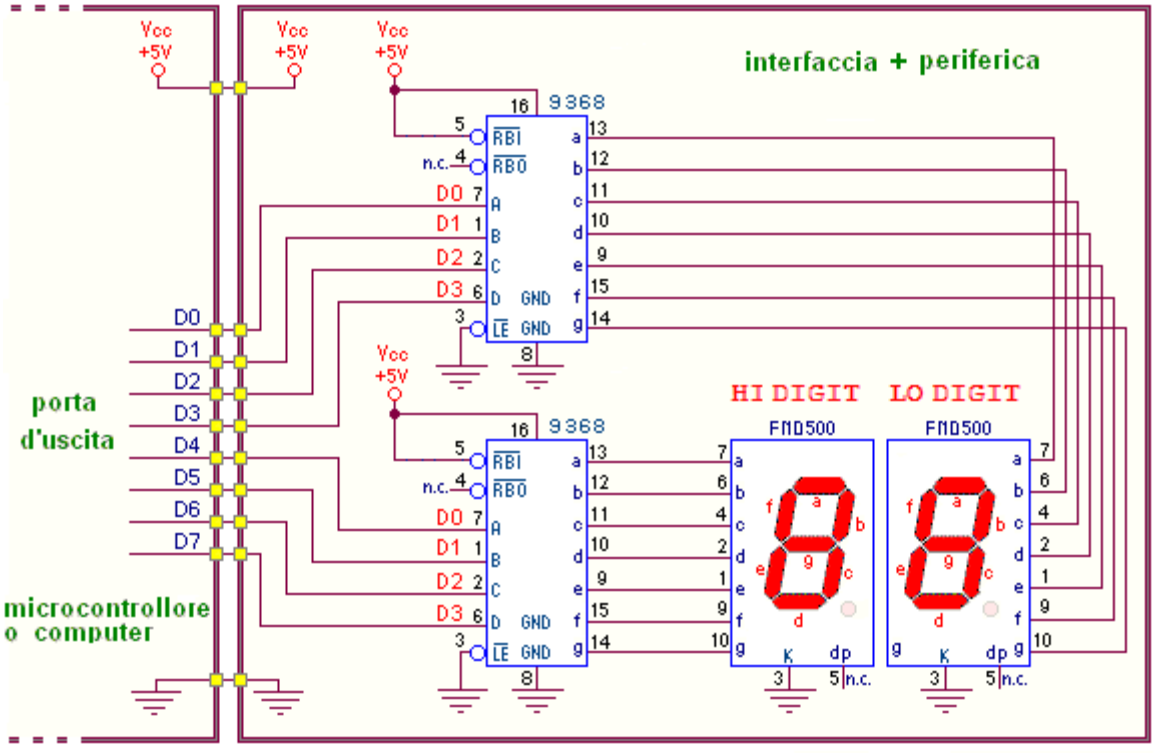
Nel normale funzionamento, il *piedino di controllo* **LE** (pin**3**) della ***memoria latch*** deve essere *a massa* mentre i rimanenti due, **RBO** (pin**4**) e **RBI** (pin**5**), possono essere *inattivi*, lasciandoli *scollegati* o meglio collegandoli a +5V.

Può essere, infine, istruttivo consultare le pagine dei ***data sheet*** *originali*; una copia è disponibile qui: **9368** [Fairchild].

INTERFACCIA CON DECODER TTL PER VISUALIZZATORE A 2 DIGIT CON INGRESSO BINARIO PURO A 4 BIT

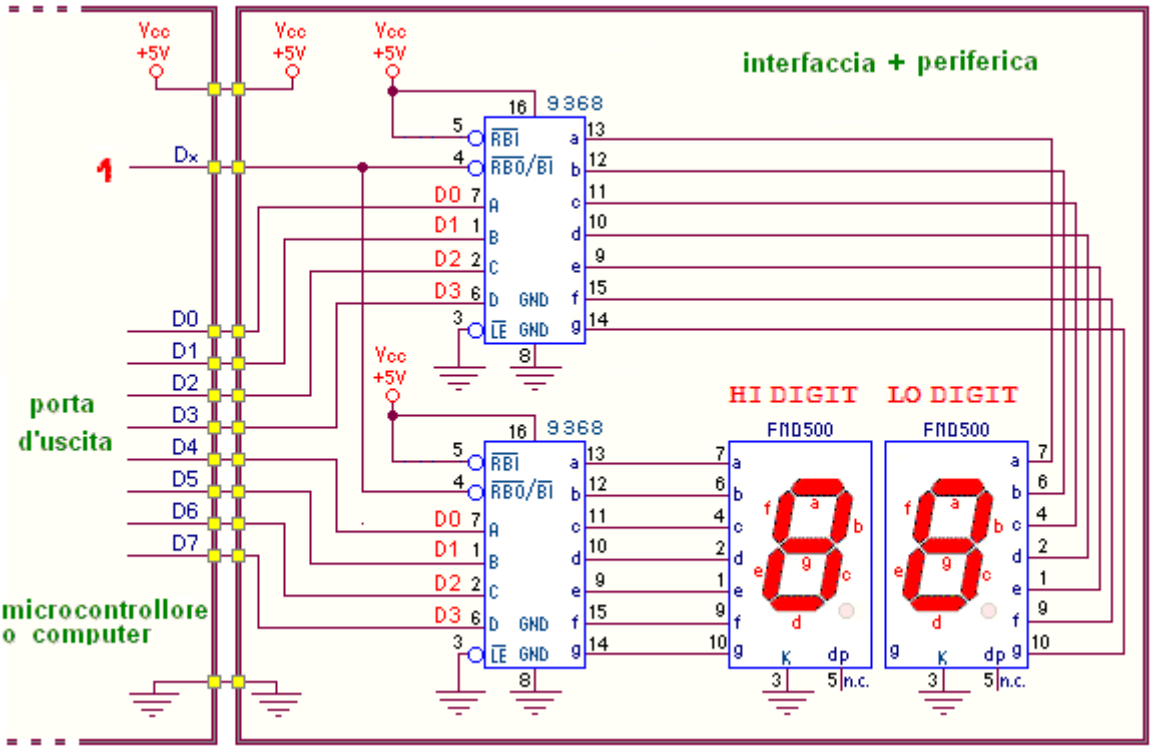
Il **Decoder TTL 9368** dispone di linee d'uscita *attive alte*, adatte al controllo dei *7 segmenti* di **digit** a **catodo comune**; la presenza di un ottimo **Driver** su ciascuna delle sue uscite rende non necessaria la presenza di **resistori** in serie per limitare la *corrente*.

La linea *d'ingresso* (*attiva bassa*) **RBI** (***R**ipple **B**lanking **I**nput*, che spegne tutti i segmenti se contemporaneamente gli ingressi **DCBA** sono a $(0000)_2$) non sembra particolarmente utile in un ambito programmabile, per cui è stata resa *inattiva*, collegandola a +5V. Lo stesso trattamento è stato riservato alla linea *d'ingresso* **LE**, ***L**atch **E**nable*: certamente la presenza di questa **Memoria** è del tutto irrilevante, in questa applicazione, essendo una **Memoria** anche la stessa *porta d'uscita* di un *micro* o del *PC*; per renderla *trasparente* è sufficiente mantenerla *a massa*.



Più interessante è la linea **BI** / **RBO** (***B**lanking **I**nput* / ***R**ipple **B**lanking **O**utput*) la cui circuiteria interna permette di forzare su essa il segnale *d'ingresso* **BI** (*attivo basso*) in grado di spegnere tutti i segmenti incondizionatamente.

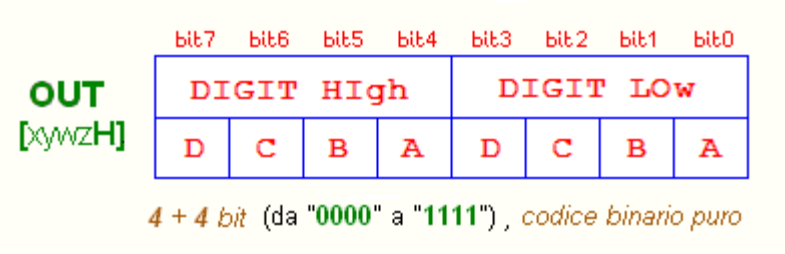
In questo caso si tratta dell'unico modo per fruire di questo servizio, perché questo componente accetta sugli ingressi **DCBA** tutte le possibili ***16*** *combinazioni* a 4 bit e quindi non esiste la possibilità di *intervenire da software* come abbiamo suggerito per tutti gli altri **Decoder** descritti in precedenza; in particolare, la *sequenza* $(11111111)_2$ offerta alla coppia invece di produrre il ***B****lanking* mostrerà FF sul digit; la variante circuitale mostra come esercitare questo controllo utilizzando una linea di una seconda *porta d'uscita* (con *microcontrollore*) o una linea delle quattro del **Registro 037AH** / **027AH** (con la *porta parallela* del *PC*, vedi http://www.giobe2000.it/HW/Parallela/Pag/RegistriSPP4.htm):



Il programma che controllerà questa **interfaccia** dovrà farsi carico di inizializzare a **1** la linea di controllo per **BI**.

Da notare che questo progetto non consente la **gestione diretta** dei *punti decimali* dei due **digit**; all'occorrenza è comunque possibile associarne il controllo ad altre due linee della seconda *porta d'uscita*.

La seguente **tabella di associazione logica** è chiamata a *virtualizzare* l'utilizzo di entrambe le soluzioni proposte:



Nella prossima puntata riprenderemo il discorso dell'**interfacciamento** dei **digit** e di altre periferiche.

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

Gli argomenti **citati** in questa puntata sono **disponibili** anche sul sito

Giobe2000

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