di GIORGIO OBER (GIOBE2000)

l'angolo di Mr A.KEER

(parte undicesima)

PROGETTARE con le PORTE LOGICHE

Decoder "con memoria"

Con i Decoder "con memoria" si conclude la rassegna dei principali dispositivi in grado di controllare i visualizzatori formati da digit "a 7 segmenti": scopriamone i segreti e il modo migliore per utilizzarli.

In chiusura della puntata precedente abbiamo constatato la concreta realtà di disporre di super decoder "a 7 segmenti" (dispositivi combinatori) dotati di uno *strato sequenziale*, cioè della possibilità di fruire su ciascuno dei propri ingressi di un flip-flop (un elemento di memoria) e di uno *strato di potenza* (driver) tale da garantire alle rispettive uscite le migliori condizioni elettriche per pilotare un led.

A questa categoria appartengono i sofisticati **4511** e **9368**, oggetto della nostra analisi attuale.

Decoder da BCD a 7 segmenti (CMOS): 4511

Come tutti i decoder per digit analizzati finora, anche il **4511** (unico disponibile nella serie **CMOS** per il controllo di digit a led) appartiene alla categoria dei **BCD to 7-segment Decoder** caratterizzata dalla capacità di interpretare solo 10 delle possibili 16 parole del *codice binario a 4 bit* proposto sulle sue 4 linee d'ingresso.

Come il TTL **74LS48** (col quale è in parte *pin-out compatibile*, escludendo 3 piedini di controllo) il Decoder **4511** converte il ***Codice BCD*** proposto sulle sue linee d'ingresso generando sulle linee d'uscita dei segnali *attivi alti*: i segmenti del digit ad esso collegato si accenderanno dunque con un **1** logico per cui questo componente è adatto a digit a *catodo comune*; il suo *pin-out* è illustrato in *Figura 1*.

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 3 funzioni in una: a) consente di congelare in una memoria a 4 bit le informazioni fornite sulle linee d'ingresso; b) garantisce ovviamente il suo intrinseco servizio di Decoder; c) è dotato di Drivers su ciascuna delle 7 linee d'uscita, al fine di fornire le migliori condizioni di pilotaggio ai led ad esse collegati.

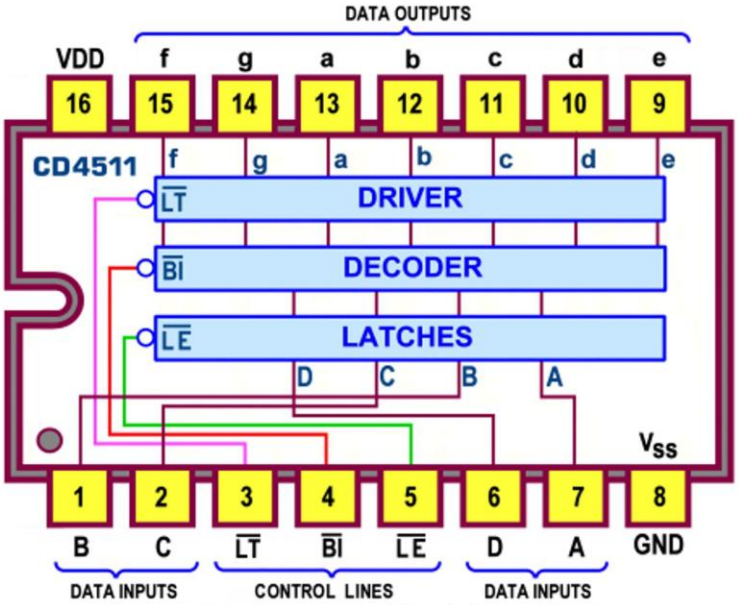

Figura 1 - BCD to 7-segment Latch/Decoder/Driver 4511: pinout

Va sottolineata l'impossibilità di gestire più componenti **4511** in cascata con la tecnica ***Ripple Blanking*** garantita invece dal **74LS48** (per altro chiamato a svolgere lo stesso servizio combinatorio), descritta la volta scorsa e ripresa più avanti per il **9368**: la disponibilità del segnale di controllo **RBO** (*Ripple Blanking Output*) è stata sacrificata per far posto a quello **LE** (*Latch Enable*) necessario per governare una memoria.

Lo strato di memoria - Prima di raggiungere il Decoder ciascuna delle 4 linee d'ingresso attraversa un flip-flop di tipo *D-Latch*; l'utilizzo di questo tipo di memoria non è molto diffuso nel progetto digitale e questo è uno dei casi più peculiari; i suoi 4 elementi sono attivati *sul livello* della linea di sincronismo (**LE**, *Latch Enable*, *pin5*), 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) per tutto il tempo nel quale la linea **LE** è tenuta a livello **1**, fino a quando verrà riportata a **0**.

La notazione logica 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 *pin5* 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 eventuale conteggio in ingresso.

Resta da valutare l'utilità di questa memoria; se questo componente è collegato a valle di un contatore (per esempio una Decade CMOS **4029** Binary/Decade Up/Down Counter) 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 (per cui *sente* un **1**) il numero sul digit non subirà più aggiornamenti, sebbene il conteggio in ingresso continui regolarmente.

Intervenendo in questo modo sulla linea **LE** (per esempio con un *controllo a fotocellula* posto in un dato punto di un percorso sportivo, nel progetto di un cronometro digitale) è possibile generare l'effetto *intertempo*, congelando il tempo corrente per qualche istante.

Mantenendo invece stabilmente *alto* l'ingresso **LE**, ogni volta che si fornisce su di esso un *breve impulso basso* il numero sul digit cambierà *istantaneamente* in funzione del codice presente in quel momento sugli ingressi DCBA, rimanendo invariato fino al successivo impulso; questo artificio torna molto utile in alcune applicazioni, per esempio per consentire la visualizzazione del valore misurato da uno strumento digitale (come 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.

La funzione intrinseca di Decoder - Come i suoi simili TTL anche questo componente accetta solo le prime 10 combinazioni, da $(0000)_2$ a $(1001)_2$, appartenenti al ***Codice BCD***; se, nonostante il divieto, si fornisce una delle ultime 6 combinazioni vietate, i componenti TTL visti la scorsa puntata davano corpo a simboli sostanzialmente casuali, dovuti alle scelte imposte dal progetto combinatorio del componente; con questo invece si otterrà come effetto quello di spegnere il digit controllato dal Decoder.

Da notare che le stesse scelte di progetto impongono anche al **4511** la visualizzazione del 6 $(0110)_2$ e del 9 $(1001)_2$ rispettivamente priva del trattino in alto o in basso, tipica del TTL **74LS48**, preso sempre come oggetto di confronto. La *Figura 2* riassume l'aspetto del digit collegato al decoder, in ogni situazione.

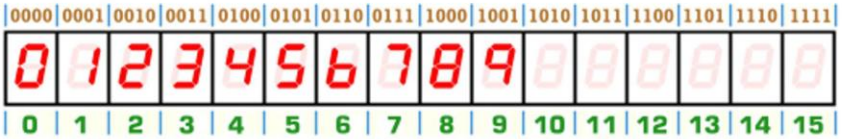

Figura 2 - BCD to 7-segment Latch/Decoder/Driver 4511: simboli visualizzati con FND500

Su questa parte circuitale agisce l'influenza del segnale d'ingresso **BI** (*Blanking Input*, *pin4*), *attivo basso*; se su di esso si forza uno **0** 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.

Curiosamente nel **74LS48** succede esattamente l'opposto: **BI** è prioritario rispetto a **LT** (e ad ogni altro ingresso).

Anche con questo componente è possibile controllare l'intensità luminosa del suo digit con la tecnica di ***Blanking*** (*spegnimento*) descritta per i suoi simili TTL nella puntata precedente: basta applicare sul *pin4* un'onda quadra non simmetrica e modificare il suo *duty cycle* (rapporto tra la durata della *parte alta* e il *periodo* del segnale) purché la frequenza sia maggiore di **45** Hz (per i digit con Led Rossi, all'Arseniuro Fosfuro di Gallio, GaAsP) al fine di garantire una sequenza di spegnimenti (imposti da **BI** *attivo*, cioè a **0**) e accensioni (imposti da **BI** *disattivo*, cioè a **1**) sufficientemente veloce da essere percepita come variazione di luminosità per effetto della persistenza ottica dell'immagine sulla retina dall'occhio umano.

La *Figura 3* (già proposta la volta scorsa) ricorda l'aspetto dei segnali necessari rispettivamente per una discreta (*duty cycle* pari a 86%) e una bassa (*duty cycle* del 14%) accensione dei digit; l'effetto ottenuto è legato al valore medio del segnale imposto su **BI** , più vicino a **1** logico nel primo caso e allo **0** nel secondo.

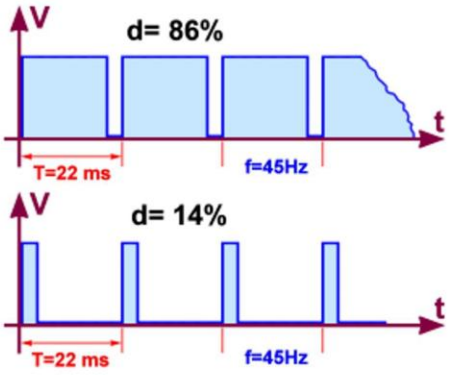

Figura 3 - BCD to 7-segment Latch/Decoder/Driver 4511: Effetto Blanking

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 realizzato con un transistor NPN configurato come *inseguitore d'emettitore* (o *a collettore comune*), una presenza decisamente inconsueta tra i componenti di tipo CMOS; le caratteristiche di questo stadio sono quelle di garantire: a) una notevole amplificazione di corrente (intrinseca dei transistor); b) un'amplificazione di tensione pressoché uguale a 1 (leggermente minore di 1), ragione per la quale lo stadio è detto anche *inseguitore di tensione* (*voltage follower*); c) una elevata impedenza di ingresso; d) una bassa resistenza d'uscita.

Certamente questa è 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 4* (di proprietà **Philips Semiconductors**, estratta dal *datasheet* del suo HEF**4511**B) 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** (*Lamp Test*, *pin3*), *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 l'efficienza; da notare che questo ingresso è prioritario rispetto a tutti gli altri, compresi **LE** e **BI**.

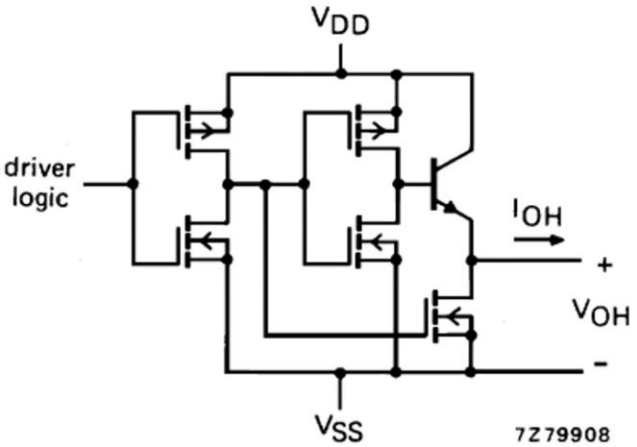

Figura 4 - BCD to 7-segment Latch/Decoder/Driver 4511: Stadio d'uscita

L'analisi della Tabella di verità (*Figura 5*, *Function Table*) è, come al solito, un importante esercizio, capace di sintetizzare il funzionamento del dispositivo:

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

H livello logico alto
L livello logico basso
X livello logico irrilevante

Figura 5 - BCD to 7-segment Latch/Decoder/Driver 4511: Tabella di verità

a) l'ultima riga evidenzia la forza prioritaria di **LT**: quando esso è *attivo* (*basso*, **L**ow level) tutte le uscite di segmento passano a **1** indipendentemente dal valore di ogni altro ingresso, accendendo di fatto il digit (a catodo comune) collegato al decoder;

b) la penultima riga conferma che, dopo **LT**, comanda **BI**: se questo ingresso è *attivo* (*basso*, **L**), con **LT** disattivo, tutte le uscite di segmento passano a **0** (basse, **L**) spegnendo di fatto il digit;

c) la prima riga mostra l'effetto di **LE**: se esso è *attivo* (*basso*, **L**) la memoria è *trasparente* (la parola DCBA d'ingresso passa sul decoder senza alcun intoppo) ma nel passaggio da **0** a **1** (fronte di salita) e per tutto il tempo in cui rimane a **1** la memoria interviene e blocca sulle sue uscite il valore dell'ingresso corrente.

Le caratteristiche logiche di questo componente sono riassunte in dettaglio dallo *schema funzionale* (*Figura 6*) e sinteticamente dallo *schema pratico* (*Figura 7*): le linee d'ingresso possono essere quelle del Bus Dati di un microprocessore o le uscite 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.

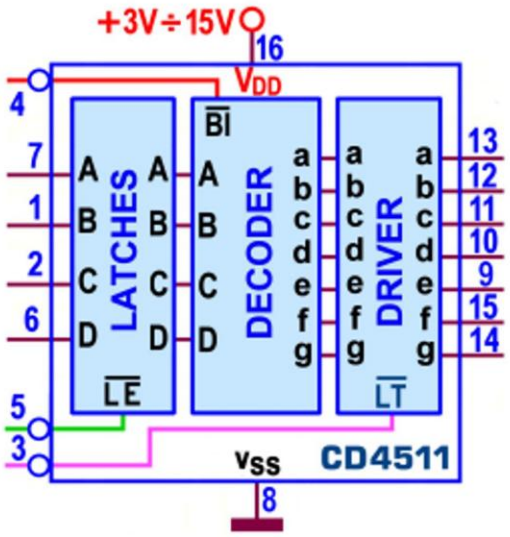

Figura 6 - BCD to 7-segment Latch/Decoder/Driver 4511: schema funzionale

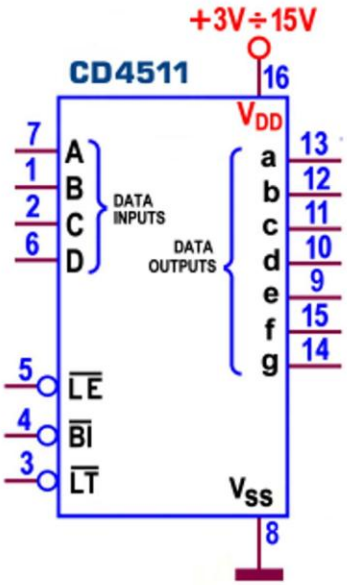

Figura 7 - BCD to 7-segment Latch/Decoder/Driver 4511: schema pratico

Per le scelte di progetto non va dimenticato che sulle uscite di segmento sono presenti valori $\mathbf{V_{OH}}$ (tensione d'uscita a livello alto) tipici CMOS, tipicamente tendenti a quello dell'alimentazione $\mathbf{V_{DD}}$; 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.

Consultando il *datasheet* è possibile ricostruire una interessante tabella (*Figura 8*) relativa alle uscite in condizioni di funzionamento tipiche.

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

Figura 8 - BCD to 7-segment Latch/Decoder/Driver 4511: Dati di progetto

Con tensione d'alimentazione di **+5V**, supponendo di imporre una corrente erogata di **15mA** risulta disponibile una tensione $\mathbf{V_{OH}}$ 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**; la *Figura 9* mostra l'impiego del Decoder **4511** con un normale digit a *catodo comune* nelle condizioni appena suggerite; nel normale funzionamento, il piedino di controllo **LE** (*pin5*) della memoria deve essere posto a massa mentre i rimanenti 2, **BI** (*pin4*) e **LT** (*pin3*), possono essere lasciati inattivi, collegandoli al positivo dell'alimentazione, data la natura CMOS del componente, che non ammette pin fluttuanti!

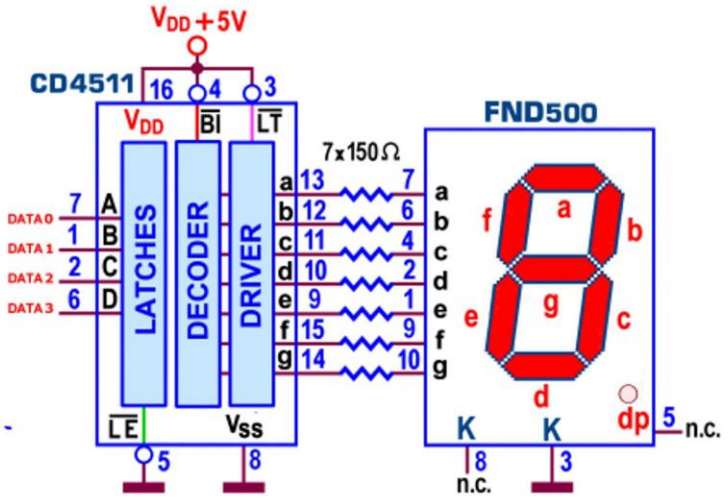

Figura 9 - BCD to 7-segment Latch/Decoder/Driver 4511: progetto con FND500

Come per gli altri componenti della famiglia CMOS la *potenza dissipata* è trascurabile mentre il *ritardo di propagazione* (*Propagation Delay Time*, $\mathbf{t_{PHL}}$ e $\mathbf{t_{PLH}}$), rilevato con carico di **50**pF, mediamente si dimezza ad ogni incremento di **5V** della tensione di alimentazione $\mathbf{V_{DD}}$; per un componente così sofisticato i punti di riferimento sono numerosi: per esempio per disporre di uscite stabili, dopo l'applicazione di un codice in ingresso, sono necessari da **150ns** ($\mathbf{V_{DD}}$=**5V**) a **40ns** ($\mathbf{V_{DD}}$=**15V**); nei confronti delle uscite dopo l'applicazione del segnale **LE** è necessario lo stesso tempo, mentre ne basta la metà dopo l'azione di **LT**.

In chiusura voglio citare (per amore di completezza) altri integrati della serie CMOS che prevedono la disponibilità di Decoder da BCD a 7 segmenti; si tratta dei componenti **4055**, **4056**, **4513** e **4543** (Latch/Decoder/Driver espressamente progettati per visualizzatori a Cristalli liquidi, LCD) e dei componenti **4026** e **4033** (Contatori/Divisori Decadici con uscite decodificate).

Decoder da binario a 7 segmenti (TTL): 9368

Questa lunga rassegna termina con il decoder TTL **9368**, un componente eccezionale, unico in grado di *decodificare* tutte le 16 combinazioni del ***Codice Binario Puro a 4 bit*** fornito sui suoi ingressi; sebbene il suo *datasheet* non preveda per esso una classificazione specifica, lo chiamerei **Binary to 7-segment Decoder**, in contrapposizione con quelli finora trattati, tutti appartenenti alla categoria destinata a decodificare solo un ***Codice BCD***.

Esso è sostanzialmente *pin-out compatibile* con il TTL **74LS48** (esclusa la funzione associata al *pin3*) e funzionalmente simile al CMOS **4511**, per altro (come detto) entrambi destinati a decodificare solo 10 codici su 16; il suo *pin-out* è illustrato in *Figura 10*.

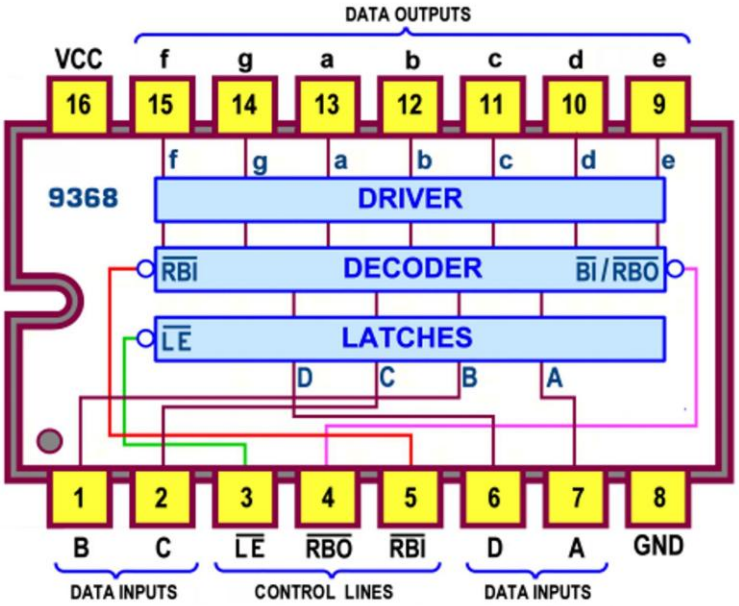

Figura 10 - Binary to 7-segment Latch/Decoder/Driver 9368: pinout

La figura evidenzia la stessa struttura logica del CMOS **4511**, in grado di assicurare le sue stesse 3 mansioni in un unico componente: lo *strato* di memoria a 4 bit sulle linee d'ingresso, la funzione di Decoder (ora estesa a tutte e 16 le possibili combinazioni) e la disponibilità di Drivers su ciascuna delle 7 linee d'uscita.

Va subito anticipato che le descrizioni relative al funzionamento del **9368** sono in gran parte simili a quelle fornite per il il **4511**, fatta salva la puntualizzazione di alcuni piccoli particolari;

di certo questo straordinario componente ne migliora i servizi in comune, rendendo possibile (tra l'altro) la tecnica ***Ripple Blanking***, che riprenderemo in dettaglio tra poco.

Lo strato di memoria e il suo controllo è identico a quello già descritto, tenendo presente che ora il segnale di sincronismo **LE**, *Latch Enable*, è sul *pin3*; in sintesi: ciascuna delle 4 linee d'ingresso attraversa un flip-flop di tipo *D-Latch* e il codice ad esse relativo può essere congelato a valle degli elementi di memoria sul *fronte di salita* di **LE** e mantenuto costante (in ingresso al decoder) fino a quando la linea **LE** verrà riportata a livello **0**; per tutto il tempo nel quale **LE** è a livello **0** il *codice binario puro a 4 bit* applicato in ingresso passa inalterato a valle della memoria, come se essa non ci fosse.

Naturalmente le applicazioni suggerite per il **4511** (intertempo per cronometro e gestione di frequenzimetro) sono assolutamente legittime anche con il **9368**, ora però da associare a contatori TTL come il *Decade Counter* **74LS90** o il *Binary Counter* **74LS93**: se l'ingresso **LE** è mantenuto a massa il digit ad esso collegato mostrerà le corrispondenti cifre (rispettivamente decimali o esadecimali) variabili con la stessa cadenza imposta dalla frequenza di *clock* applicata al contatore; non appena **LE** viene scollegato da massa il simbolo sul digit, corrispondente al codice presente in quel momento sugli ingressi DCBA, non subirà più aggiornamenti (sebbene il conteggio fornito in ingresso continui a variare regolarmente) rimanendo invariato fino a quando **LE** sarà riportato a massa.

La funzione di Decoder - Unico tra tutti i Decoder conosciuti, questo componente interpreta tutte le possibili 16 combinazioni del ***Codice binario a 4 bit*** proposto in ingresso, assicurando per ciascuna di esse il corrispondente simbolo del *sistema di numerazione esadecimale*; l'aspetto del digit collegato al decoder è dunque quello mostrato in *Figura 11*.

Figura 11 - Binary to 7-segment Latch/Decoder/Driver 9368: simboli visualizzati con FND500

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); tra l'altro nella visualizzazione del **6** e del **9** è previsto il famoso trattino, rispettivamente in alto o in basso.

Su questa parte circuitale agisce l'influenza dei segnali **BI**/**RBO** (*Blanking Input*/*Ripple Blanking Output*, *pin4*) e **RBI** (*Ripple Blanking Input*, *pin5*), entrambi *attivi bassi*, gli stessi descritti la puntata precedente per il Decoder TTL **74LS48** che, come già anticipato, da questo punto di vista è del tutto identico a quello sotto test; ne ricordiamo le principali considerazioni.

In virtù di una particolare circuiteria interna (basata su un resistore di *pull-up*, adatta a realizzare una logica di tipo *wired-AND* con ogni uscita *open collector*) la linea collegata al *pin4* può essere utilizzata, senza produrre danni, sia come *uscita* (rendendo disponibile il segnale **RBO**) sia come *ingresso* (consentendo di forzare su di essa il segnale **BI**).

Usata come ingresso **BI** permette (se attivata, cioè forzata a **0**) di spegnere incondizionatamente il digit collegato al Decoder, con qualunque codice presente sugli ingressi DCBA; nel funzionamento normale **BI** deve essere lasciato scollegato (o collegato al **+5V**); da notare che (come per il **74LS48**) l'ingresso **BI** è prioritario, cioè quando esso è attivo contemporaneamente ad ogni altro ingresso la funzione di questi ultimi 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.

Usata come uscita, la linea **RBO** serve (insieme all'ingresso **RBI** sul *pin5*) per gestire il meccanismo di ***Ripple Blanking*** atto ad 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); sebbene esso sia stato già descritto la puntata scorsa, per il **74LS48**, è bene riprenderne i concetti, anche per il fatto che gli schemi applicativi sono diversi, per l'assenza della linea di *Lamp Test* (**LT**) ora sostituita da quella di *Latch Enable* (**LE**):

a) l'ingresso **RBI** consente (se attivato, cioè forzato a **0**) di spegnere (*blanking*) il digit collegato al Decoder *solo se* contemporaneamente il codice presente sui suoi 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 l'eventuale cifra zero è di lasciare l'ingresso **RBI** scollegato (o meglio collegato al +**5V**), cioè disattivo (forzato a livello logico **1**).

b) l'uscita **RBO** serve per reiterare questo servizio anche sul decoder di un eventuale digit meno significativo, posto a destra del primo: collegato all'ingresso **RBI** del Decoder successivo produrrà lo stesso effetto, tenendo spento il digit ad esso collegato *solo se* sui suoi ingressi DCBA è presente il codice $(0000)_2$.

c) collegando in cascata le uscite **RBO** di ciascun decoder (a cominciare da quello associato al digit più significativo) con l'ingresso **RBI** del decoder successivo (subito a destra) è dunque possibile evitare di accendere tutti gli antiestetici e inutili *zero a sinistra* di un numero: anche nella pratica comune sembra logico scrivere 236 piuttosto che 0236.

Va per altro notato che se il **9368** è utilizzato nel pieno della sua specificità (quella 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 $\mathbf{0AF5}_H$ o come $\mathbf{001E}_H$) sia per l'improbabilità di dover mostrare un numero esadecimale frazionario (rendendo così inutile anche il servizio per gli zero a destra).

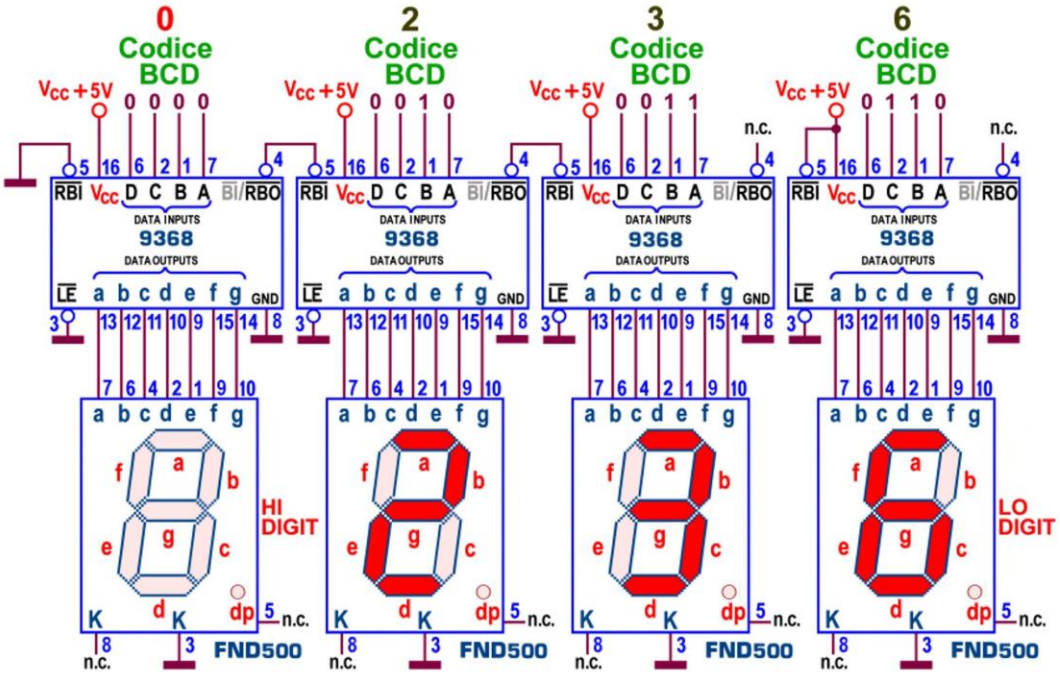

Figura 12 - Visualizzatore a 4 digit FND500 con 9368: Effetto Ripple Blanking

Ma è pur vero che questo dispositivo si comporta benissimo anche nell'ambito di un *visualizzatore decimale*, dando in questo caso per scontato che l'informazione fornita in ingresso ai Decoder sia *di tipo **BCD***, anche se essi sarebbero in grado di processare tutte le 16 possibili combinazioni; la situazione è mostrata in *Figura 12*, relativa ad un visualizzatore con 4 digit **FND500**, ciascuno controllato da un **9368**, nell'ipotesi che i ***codici BCD*** in ingresso ai 4 Decoder siano quelli corrispondenti alle cifre 0236; con questa predisposizione (ingresso **RBI** del Decoder collegato al *digit più significativo* forzato a massa) lo "**0**" corrispondente al codice $(0000)_2$, fornito sui suoi ingressi DCBA, non viene mostrato (il digit rimane spento).

Questa situazione forza a **0** anche l'uscita **RBO** che, a sua volta, forza a massa l'ingresso **RBI** del Decoder collegato al digit successivo (subito a destra): 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.

In caso contrario anche il secondo digit rimarrebbe spento e, in coerenza con la regola, il Decoder ad esso associato forzerebbe a **0** la sua uscita **RBO** e, di conseguenza, anche l'ingresso **RBI** del Decoder collegato al digit successivo (il terzo da sinistra); il meccanismo si applica per tutte le cifre del display e, se fosse mantenuto inalterato per tutti i decoder successivi al primo, fornendo 4 codici BCD "nulli", terrebbe spento tutto il visualizzatore.

Sebbene la cosa sia possibile non sembra comunque una buona soluzione: se sugli ingressi DCBA di ogni decoder viene fornito il codice $(0000)_2$ è ragionevole mostrare almeno lo zero associato al digit più a destra; per questa ragione l'ingresso **RBI** del Decoder ad esso relativo viene mantenuto a **1**, spezzando la catena di controllo e vanificandone l'effetto.

Nella scorsa puntata abbiamo analizzato anche lo schema necessario per gestire gli zero non significativi della parte frazionaria, per esempio come nel numero 63,20; poichè la tecnica rimane sostanzialmente identica evitiamo di appesantire ulteriormente questa trattazione, rimandando il lettore a quanto già pubblicato.

Lo strato del Driver - La terza parte di questo componente è decisamente interessante ed esclusiva: 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 *datasheet*) ma, con un led ad essa direttamente collegato, si adatterà al valore della caduta ai suoi capi, cioè da **1,75V** a **1,85V**.

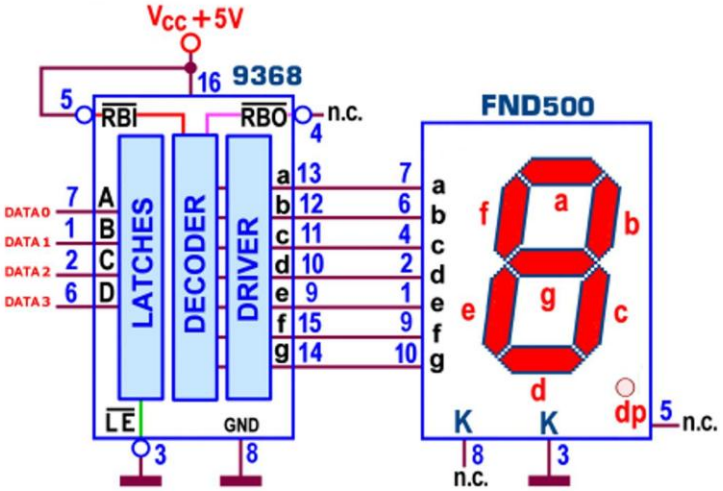

Figura 13 - Binary to 7-segment Latch/Decoder/Driver 9368: progetto con FND500

Nell'impiego con un normale digit a catodo comune la presenza di resistori di limitazione corrente in serie ai segmenti-led non è dunque necessaria; la *Figura 13* propone questa situazione, sottolineando che nel normale funzionamento, il piedino di controllo **LE** (*pin3*) della memoria latch deve essere a massa mentre le 2 linee di controllo del *Ripple Blanking*, **RBO** (*pin4*) e **RBI** (*pin5*), irrilevanti in questo caso, possono essere lasciate inattive, lasciando scollegata la prima e scollegata (o meglio collegandola a +**5V**) la seconda.

Anche in questo caso la Tabella di verità (*Function Table*, *Figura 14*) offre un eccellente sintesi del funzionamento del dispositivo, per cui è importante saperla consultare e capire: a) l'ultima riga evidenzia la forza prioritaria di **BI**, sul *pin4* usato come ingresso: quando esso è *attivo* (*basso*, **L**) tutte le uscite di segmento passano a **0** indipendentemente dal valore di ogni altro ingresso, spegnendo di fatto il digit collegato al decoder; b) la prima riga mostra l'effetto di **LE**: se esso è tenuto a **1** (*alto*, **H**) la memoria mantiene sulle uscite il valore presente sugli ingressi DCBA nel suo passaggio da **0** a **1** (*fronte di salita*); riportata a **0** la memoria è *trasparente* (la parola DCBA d'ingresso passa sul decoder senza alcun intoppo); c) la seconda riga conferma l'influenza dell'ingresso **RBI** sul simbolo mostrato sul digit e su **RBO** (*pin4* usato come uscita): se forzata a **0** (*attiva bassa*, **L**) essa produce effetto solo quando sugli ingressi DCBA è presente il codice $(0000)_2$ ed è irrilevante in tutti gli altri casi.

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

H livello logico alto
L livello logico basso
X livello logico irrilevante

Figura 14 - Binary to 7-segment Latch/Decoder/Driver 9368: Tabella di verità

Le caratteristiche logiche di questo componente sono riassunte in dettaglio dallo *schema funzionale* (*Figura 15*) e sinteticamente dallo *schema pratico* (*Figura 16*).

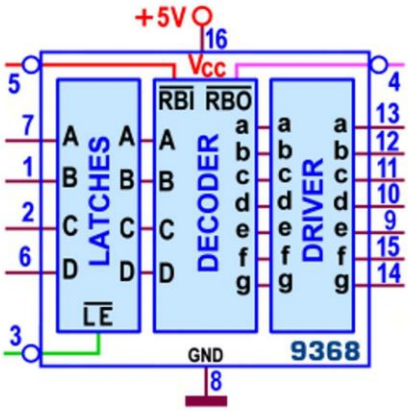

Figura 15 - Binary to 7-segment Latch/Decoder/Driver 9368: schema funzionale

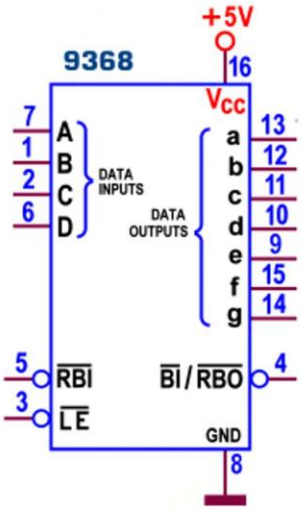

Figura 16 - Binary to 7-segment Latch/Decoder/Driver 9368: schema pratico

In conclusione: l'utilizzo del **9368**, prodotto dalla **Fairchild Semiconductor** da più di vent'anni, è molto versatile in ogni applicazione ma risulta particolarmente interessante quando è chiamato ad operare nell'ambito dei microprocessori o dei microcontrollori; essendo l'unico decoder in grado di organizzare la visualizzazione dei simboli esadecimali, indispensabili per tradurre in modo umano le comunicazioni binarie, si presta in modo eccellente per visualizzare le informazioni tipicamente presenti sulle porte d'uscita di un micro o di un PC, come *indirizzi* (delle locazioni di memoria, visualizzabili su 4 digit a partire dai loro 16 bit) o *dati* (per esempio il contenuto di registri della CPU o di una locazione della memoria, visualizzabili su 2 digit a partire dai loro 8 bit).

La sua disponibilità di un stato di memoria rende inoltre possibile l'interfacciamento diretto con le le linee del *Bus Dati* di un processore o di un single-chip, controllando il sincronismo di memorizzazione (**LE**) a partire dai *Bus Address* e *Bus di Controllo*, con opportune reti di decodifica binaria.

L'assorbimento tipico è di **335 mW** e il *ritardo di propagazione* (*Propagation Delay Time*, rilevato con carico di **100**ohm/**15**pF) per disporre di uscite stabili è pari $\mathbf{t_{PHL}}$=**75ns** e $\mathbf{t_{PLH}}$=

50ns, dopo l'applicazione di un codice in ingresso, e leggermente maggiore dopo l'applicazione del segnale **LE**.

Per finire possiamo citare altri 2 componenti, del tutto *pinout compatibili* con il **9368** ma dotati di uscite *attive basse*, adatte quindi a digit ad anodo comune, come l'**FND507**: il **9370**, versione con uscite *open collector* sempre prodotta dalla **Fairchild Semiconductor**, funzionalmente identico e capace di gestire tutti e 16 i codici a 4 bit proposti in ingresso, ma che ha bisogno di resistori esterni; e il **8374**, prodotto dalla **NTE electronics**, capace di assicurare **15mA** in uscita ma limitato ad interpretare solo ingressi di tipo BCD.