espansione delle entrate

Tramite multiplexer

Con i due chip 74HC151 allegati a questo fascicolo (e quelli che troverai nel prossimo) potrai realizzare il circuito per l'espansione degli ingressi. Anche in questo caso, come per l'espansione delle uscite, quello che si intende fare è superare il vincolo fisico relativo al numero di ingressi, a costo però di impegnare porte logiche del microcontrollore. **A differenza della gestione degli attuatori (uscite) che, in generale, richiede poca memoria, nel caso della gestione dei sensori (ingressi) ne è tipicamente richiesta di più.** Di solito, infatti, l'uso dei sensori comporta la memorizzazione del valore rilevato, cosicché lo si possa poi riutilizzare a un dato scopo. Aumentare il numero di sensori senza modificare la quantità di memoria disponibile, perciò, potrebbe comportare una carenza di memoria, vale a dire una situazione più difficilmente superabile.

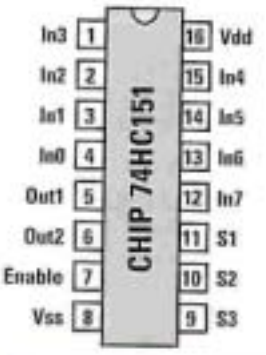
In generale, qualora il numero di ingressi e uscite non fosse sufficiente, si potrebbe affiancare, al primo, un secondo microcontrollore, con tanto di nuove porte logiche e di una nuova memoria RAM (con due controllori non ha senso, né si può ottenere, che condividano una stessa memoria RAM), guadagnando anche in velocità (un vantaggio soprattutto per l'espansione delle uscite). Tuttavia, tale soluzione è costosa (in termini economici), mentre i circuiti di espansione che stiamo studiando offrono soluzioni meno dispendiose al problema. Per quanto riguarda l'espansione degli ingressi, va detto che, comunque, resta una forte limitazione: ciò che si ottiene, infatti, è una serie di 'puri' ingressi, utilizzabili solo per la lettura di valori logici. Questo significa che **si potranno utilizzare solo componenti oppure sensori che prevedano unicamente la lettura del loro stato logico** (come un pulsante ON/OFF, che sarà infatti l'esempio cui faremo riferimento in seguito). **Sono invece esclusi dall'utilizzo con questa configurazione tutti quei sensori o componenti che, oltre alla lettura del loro stato, richiedono la possibilità di utilizzare la porta che li controlla come output** (ad esempio i fotoresistori che, per impostare il valore logico iniziale e poter poi utilizzare l'istruzione **rctime**, richiedono l'impiego di una porta logica, utilizzabile sia come input sia come output).

Nella foto. I chip 74HC151, allegati a questo fascicolo e destinati a costituire il circuito di espansione delle entrate.

Le fasi di programmazione

Anche nell'affrontare lo studio del circuito per l'espansione degli ingressi vale la pena procedere a tappe. Inizieremo perciò con un circuito in grado di gestire 8 ingressi al costo di 4 porte del microcontrollore. Il cuore di questo circuito è uno dei chip 74HC151 (a destra) di cui disponi, che è un '8-Input Multiplexer'.

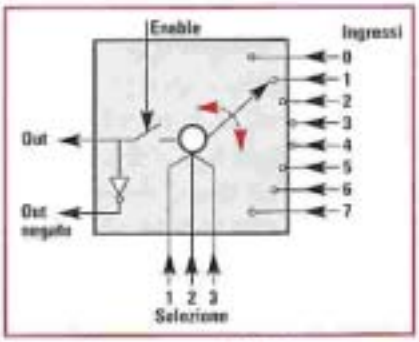
In generale, un *N*-Input Multiplexer è un componente elettronico dotato di *N* ingressi che potremmo definire 'primari' (In0, In1... In(*N*-1)), un'uscita (Out1) e alcuni altri ingressi, detti 'secondari' (S1, S2,...). In particolare, il numero di questi ultimi è funzione del numero di quelli primari (di *N*): il numero di ingressi secondari, infatti, è pari a $\log_2 N$ (logaritmo in base 2 di *N*, pari all'esponente cui si deve elevare 2 per ottenere *N*). Tramite gli ingressi secondari è possibile selezionare ogni volta l'entrata primaria da attivare, collegandola all'uscita del multiplexer.



LE FASI DI PROGRAMMAZIONE

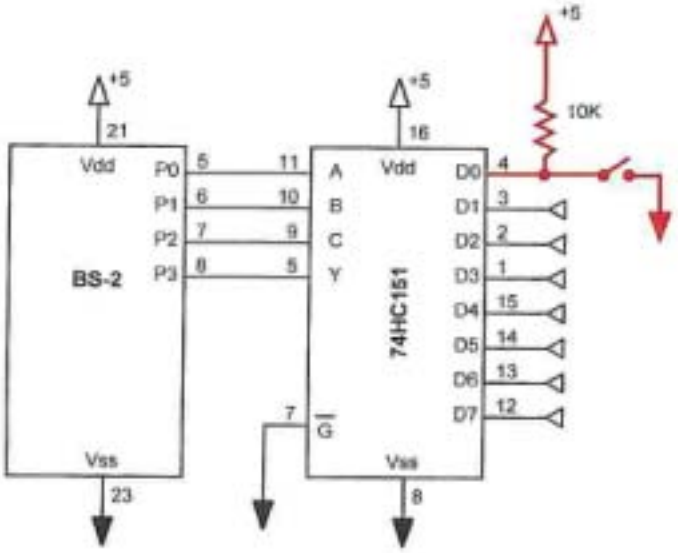
La selezione dell'ingresso primario desiderato avviene tramite la codifica (binaria) operata sugli ingressi secondari. Il nostro multiplexer, ad esempio, avendo 8 ingressi primari (In0... In7), ne avrà 3 secondari ($\log_2 8=3$): infatti, 3 bit (in ingresso, rispettivamente su S1, S2 e S3) sono sufficienti a codificare 8 numeri ($2^3=8$). Di fatto, però, in un multiplexer esiste un altro ingresso secondario, con funzione diversa: si tratta di Enable, che (come già nel flip-flop) permette di abilitare la lettura dei dati di ingresso. Nel caso del chip 74HC151, però, tale ingresso funziona all'opposto di quello del flip-flop: per abilitare il circuito, infatti, bisogna settare Enable=0. Infine, per quanto riguarda l'uscita del multiplexer, nel nostro caso è doppia: il valore logico proposto in uscita sul pin 5 del chip (Out1), infatti, risulta essere effettivamente quello dell'ingresso selezionato; la seconda uscita, invece, predisposta sul pin 6 (Out2), riporta il valore negato della precedente (questa doppia opzione non verrà sfruttata in questa sede; tuttavia può essere utile per risolvere, senza il bisogno di ulteriori componenti, particolari applicazioni che richiedano l'inversione del segnale). Riassumendo, dato il ruolo di ciascun pin del multiplexer (riassunto nella tabella in alto a destra), il suo funzionamento può essere schematizzato come in figura qui a destra: la configurazione dei tre pin di selezione (ossia il loro ricevere un valore di tensione alto o basso) è responsabile della selezione di uno degli otto ingressi che viene collegato all'uscita Out. Tale collegamento viene abilitato o meno da Enable, che funge da interruttore. La seconda uscita (Out negato), invece, consente di disporre anche del valore negato dell'ingresso selezionato, grazie alla presenza del cosiddetto 'negatore' o invertitore logico NOT (per la simbologia e la funzione delle porte logiche, vedi la sezione TECNO, a pag. 31). La tabella sotto, infine, mostra le effettive relazioni tra ingressi (vale a dire le configurazioni di quelli secondari e le possibili letture sui primari) e uscite del chip. I simboli tra parentesi vanno intesi come negati ma, mentre (Out) è davvero la negazione di Out (e rappresenta l'uscita sul pin 6, che è la negazione di quella sul pin 5), Enable non è di per sé negazione di nulla, ma funziona al contrario di quanto ci si potrebbe aspettare (e di come sappiamo funzionare lo stesso ingresso del flip-flop): tale ingresso, dunque, è indicato tra parentesi (E) solo per ricordare il suo funzionamento. Dalla tabella possiamo vedere (e ci sarà utile più avanti) che, quando Enable=1 (H=high), e quindi il multiplexer è disabilitato, l'uscita vale sempre 0 (L=low; invece * è segno di indifferenza).

Pin 74HC151	Ruolo	Specifiche
1	In3	Ingresso primario
2	In2	Ingresso primario
3	In1	Ingresso primario
4	In0	Ingresso primario
5	Out1	Uscita
6	Out2	Uscita negata
7	Enable	Enable negato (attivazione per ENABLE=0, opposta al flip-flop)
8	Vss	Alimentazione (a massa)
9	S3	Ingresso secondario
10	S2	Ingresso secondario
11	S1	Ingresso secondario
12	In7	Ingresso primario
13	In6	Ingresso primario
14	In5	Ingresso primario
15	In4	Ingresso primario
16	Vdd	Alimentazione (a 5 volt)



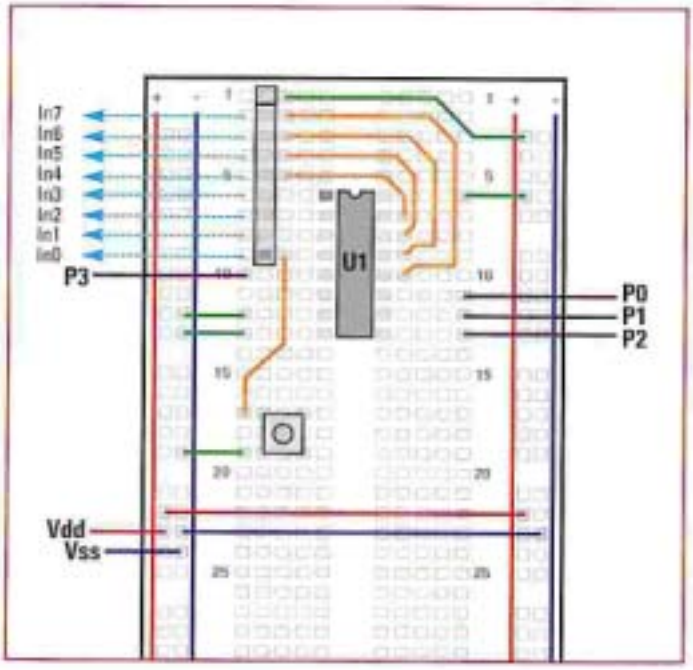
Enable	Ingressi secondari			Ingressi primari								Uscita	
(E)	S1	S2	S3	In0	In1	In2	In3	In4	In5	In6	In7	Out	(Out)
H	*	*	*	*	*	*	*	*	*	*	*	L	H
L	L	L	L	L	*	*	*	*	*	*	*	L	H
L	L	L	L	H	*	*	*	*	*	*	*	H	L
L	L	L	H	*	L	*	*	*	*	*	*	L	H
L	L	L	H	*	H	*	*	*	*	*	*	H	L
L	L	H	L	*	*	L	*	*	*	*	*	L	H
L	L	H	L	*	*	H	*	*	*	*	*	H	L
L	L	H	H	*	*	*	L	*	*	*	*	L	H
L	L	H	H	*	*	*	H	*	*	*	*	H	L
L	H	L	L	*	*	*	*	L	*	*	*	L	H
L	H	L	L	*	*	*	*	H	*	*	*	H	L
L	H	L	H	*	*	*	*	*	L	*	*	L	H
L	H	L	H	*	*	*	*	*	H	*	*	H	L
L	H	H	L	*	*	*	*	*	*	L	*	L	H
L	H	H	L	*	*	*	*	*	*	H	*	H	L
L	H	H	H	*	*	*	*	*	*	*	L	L	H
L	H	H	H	*	*	*	*	*	*	*	H	H	L

Da un punto di vista hardware, per implementare il multiplexer sulla breadboard è necessario realizzare il circuito in questa pagina. Nello schema circuitale (sotto) noterai che i tre ingressi di selezione (indicati con A, B e C) sono collegati sulle porte logiche P0, P1 e P2 del microcontrollore (BS-2), mentre l'uscita (Y) è su P3 (delle due uscite, utilizziamo quella con valore logico uguale a quello di ingresso, non negato, ossia l'uscita del pin 5). Dal momento che l'ingresso Enable (G) è collegato direttamente a massa (Vss=0), il multiplexer risulta sempre abilitato (Low).



La prima entrata (D0, ossia In0) è collegata a un cosiddetto 'partitore di tensione' costituito da un resistore da 10 KΩ e un interruttore (visibili nella porzione colorata del circuito): se l'interruttore è aperto, il partitore è escluso dal circuito e l'ingresso D0 si collega direttamente a Vdd (5 volt, ossia valore logico alto); se invece l'interruttore è chiuso, dal momento che la resistenza associata all'interruttore è minima rispetto ai 10 KΩ, l'ingresso va a massa (0 volt, ossia valore basso). Soffermiamoci ora su quest'ultimo valore di resistenza: invece di impiegare un resistore da 10 KΩ, useremo un cosiddetto 'array resistivo' (allegato al prossimo fascicolo), ossia un componente particolare che realizza una rete resistiva da 10 KΩ. Tale rete non è nient'altro che un parallelo di 8 resistori integrati, in modo tale da ridurre lo spazio che occupano sulla breadboard. L'array (a destra, il rettangolo) dispone di 9 pin, di cui 8 relativi ai resistori (quello più scuro è il pin 1) e 1, collegato a massa o a Vdd, per alimentare l'intero parallelo (a destra, questo pin è isolato dagli altri; sull'involucro nero del componente fisico, invece, è indicato da un pallino grigio). Vedremo tra poco in un esempio che ciascuno degli 8 pin di cui sopra va collegato a un ingresso del multiplexer.

Oltre alla compattezza del circuito, in questo modo si ottiene una protezione degli ingressi rispetto al rumore ambientale (ad esempio, un disturbo di natura elettromagnetica): in pratica, mantenendo costante il valore logico di ciascun ingresso (anche quando non fosse utilizzato o selezionato), i resistori filtrano l'eventuale rumore dovuto a interazioni esterne. In particolare, collegando la rete resistiva a Vdd (come nel nostro caso), il valore logico si mantiene alto a tutti gli ingressi, compresi quelli non utilizzati; viceversa, collegando la rete a massa si mantiene il valore logico basso. Per quanto riguarda l'interruttore, invece, puoi utilizzare quello presentato a pag. 129. Il suo funzionamento è identico a quello visto per il tasto di reset (integrato direttamente sulla scheda madre). Tale interruttore è munito di 4 pin, due dei quali possono essere utilizzati per implementarne la funzione di pulsante ON/OFF. Vediamo dunque un semplice programma (nella pagina successiva) per capire la gestione software del circuito implementato (illustrato qui sotto). Poiché il programma si limiterà a verificare il corretto funzionamento del circuito, utilizzeremo un solo componente (il pulsante, rappresentato da un quadrato contenente un cerchio) collegato a uno degli ingressi (In0) del multiplexer (U1). Come puoi notare, uno dei due pin del pulsante è collegato a In0 mentre l'altro è collegato a massa (la colonna blu della porzione laterale della breadboard). Il pin collegato a In0, attraverso l'array resistivo, è inoltre collegato a Vdd. Se il pulsante non viene premuto, manca il collegamento a massa, pertanto su In0 si ha un valore logico alto. Viceversa, premuto il pulsante, si stabilisce il collegamento a massa, cosicché all'ingresso In0 compare un valore logico basso.



LE FASI DI PROGRAMMAZIONE

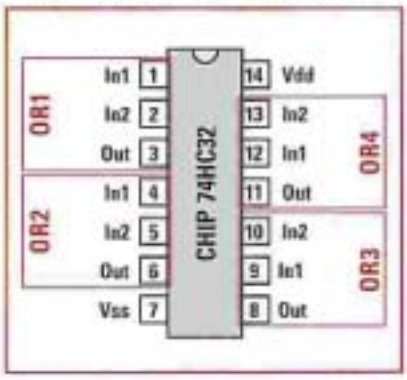
Vediamo allora il programma (a destra) che mostrerà il valore logico registrato all'ingresso In0 (e quindi lo stato del pulsante) nella finestra di debug. Per farlo, bisogna anzitutto selezionare come ingresso l'entrata In0, configurando i tre pin di selezione a 0 (le tre istruzioni low su S1_, S2_ e S3_), in base alla tabella di pag. 230 (in basso). Segue un loop (ciclo:... goto ciclo) in cui, semplicemente, si visualizza in debug il valore dell'uscita (OUT_, che è un alias della variabile speciale in3, collegata a P3). Come puoi notare, la gestione dell'espansione delle entrate è molto più semplice rispetto a quella dell'espansione delle uscite: intanto qui è possibile selezionare solo un ingresso, quello desiderato; inoltre, sebbene la lettura sia più lenta rispetto a quella diretta su una porta del BS-2, tale operazione risulta però relativamente veloce. Nel caso peggiore del circuito con 8 uscite erano infatti necessarie 25 istruzioni (24 di inserimento e 1 di aggiornamento del flip-flop), rispetto all'unica istruzione necessaria per gestire direttamente le porte del BS-2; inoltre, anche utilizzando ripetutamente la stessa uscita, non si aveva nessun vantaggio in termini di tempo. Nel circuito con 8 ingressi, invece, si utilizzano al massimo 4 istruzioni (3 per selezionare l'ingresso e, eventualmente, 1 per abilitare il multiplexer) al posto dell'unica diretta sul BS-2; inoltre, se si volesse utilizzare ripetutamente uno stesso ingresso, basterebbe 1 sola istruzione (una volta che l'ingresso è selezionato, non è necessario riselezionarlo per una seconda lettura).

prog1.bs2

```
'{$STAMP BS2}

'-------- Variabili ----------

OUT_        var   in3

'-------- Costanti ----------

S1_         con   0
S2_         con   1
S3_         con   2

'------ Programma Principale --------

low S1_             'Setta come
                    'ingresso l'entrata 0
low S2_
low S3_

ciclo:

      debug ? OUT_

      pause 100

goto ciclo
```

Come annunciato, vediamo come espandere il circuito per ottenere 16 ingressi. Per farlo, ci serve il secondo multiplexer (allegato a questo fascicolo), ma non solo.

Quello che ci proponiamo di fare è ottenere un circuito che, allo stesso tempo, permetta a due multiplexer di condividere la stessa uscita e di escludersi reciprocamente. Volendo, basterebbe replicare il circuito precedente anche per il secondo multiplexer, impegnando 6 porte del BS-2; la scelta progettuale proposta qui, invece, consente di utilizzarne solo 5. Per farlo, però, a differenza di quanto visto per l'espansione delle uscite, non è sufficiente mettere semplicemente in parallelo i due multiplexer. In effetti, per quel che riguarda solo i 3 pin di selezione non ci sarebbero problemi, che restano però in merito al pin di uscita. Nel caso dell'espansione delle uscite, il canale DATA era collegato a entrambi i flip-flop che, dunque, ricevevano in ingresso la stessa serie di bit prodotta dal registro; poi, quello abilitato (eventualmente entrambi) veniva aggiornato rispetto a tutte le 8 uscite. Nel caso dell'espansione degli ingressi, invece, una cosa del genere non ha senso, perché il microcontrollore può ricevere solamente un ingresso alla volta.



Ciò che vogliamo ottenere, allora, è che le uscite dei due multiplexer coesistano su un unico ingresso del BS-2, senza però interferire tra loro: come vedremo ora, ci serviranno due diversi dispositivi che realizzino altrettante porte logiche (in particolare, una porta OR e una NOT). Prima di procedere, però, torniamo alla tabella di pag. 230, relativa al funzionamento del multiplexer, per ribadire che quando un multiplexer è disabilitato (Enable=H, sulla prima riga), la sua uscita vale sempre 0. Come vedremo, questa informazione si rivelerà preziosa per risolvere il nostro problema. Ma procediamo per gradi. Consideriamo anzitutto l'operazione logica OR, schematizzata a destra con la sua tabella di verità. Noterai che lo 0 è un valore cosiddetto 'di indifferenza', dal momento che, se uno degli ingressi vale 0, l'uscita assume lo stesso valore dell'altro ingresso: osservando la tabella, infatti, quando l'ingresso A vale 0, l'uscita assume i valori di B e viceversa. Dunque, se collegassimo le uscite dei due multiplexer agli ingressi di un dispositivo che realizzi una porta logica OR e se, inoltre, riuscissimo a garantire che uno degli ingressi sia senz'altro pari a 0, ma non entrambi (e lo faremo con una porta NOT), ecco che, allora, raggiungeremmo il nostro scopo: i due multiplexer condividerebbero l'uscita e, allo stesso tempo, abilitandone uno si escluderebbe l'altro. Da un punto di vista circuitale, per implementare la funzione OR utilizzeremo il chip 74HC32 (illustrato a sinistra e allegato al prossimo fascicolo).

Operatore logico OR		
Ingressi		Uscita
A	B	
0	0	0
0	1	1
1	0	1
1	1	1

Pin 74HC32	Ruoli		Collegamento
1	In1	OR1	Out multiplexer 1
2	In2		Out multiplexer 2
3	Out		P3 (BS-2)
4	In1	OR2	- (non usato)
5	In2		- (non usato)
6	Out		- (non usato)
7	Vss		Alimentazione (a massa)
8	Out	OR3	- (non usato)
9	In1		- (non usato)
10	In2		- (non usato)
11	Out	OR4	- (non usato)
12	In1		- (non usato)
13	In2		- (non usato)
14	Vdd		Alimentazione (5 volt)

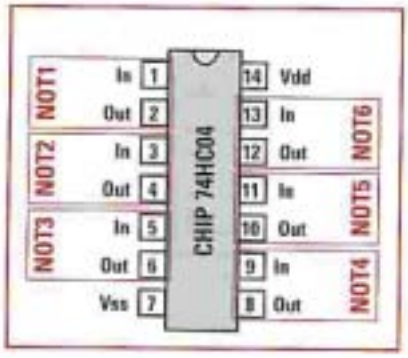
Per come è fatta una porta OR, imponendo due valori logici come ingressi su due pin, si ottiene in uscita, su un altro pin, il valore logico corrispondente alla funzione OR dei due ingressi: in generale, dunque, sono sufficienti 3+2 pin (2 ingressi e 1 uscita, oltre ovviamente a Vdd e Vss). Come puoi constatare dallo schema del chip e dalla tabella a sinistra, il 74HC32 dispone però di 14 pin. Questo chip, infatti, può essere utilizzato per implementare fino a quattro OR diversi. Osserva lo schema: fatta eccezione per i pin 7 e 14 che vanno collegati rispettivamente a massa e a Vdd, gli altri 12 sono raggruppati in 4 gruppi di 3 pin contigui: ciascun gruppo è una porta OR costituita da due ingressi (In1 e In2) e da un'uscita (Out) che restituisce il valore OR dei primi due. Per il nostro scopo sarà dunque sufficiente collegare le uscite dei due multiplexer agli ingressi di una delle porte OR disponibili e l'uscita di quest'ultima alla porta P3 del BS-2. Nell'esempio in tabella abbiamo scelto di utilizzare il primo OR (pin 1, 2 e 3), tuttavia è di fatto possibile utilizzarne uno qualunque dei quattro disponibili, indifferentemente.

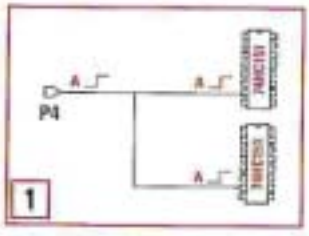
L'ultimo passo che resta da compiere riguarda la porta NOT (realizzata dal chip 74HC04, allegato al prossimo fascicolo) e la gestione dei due Enable che, dovendosi escludere l'un l'altro, non possono essere collegati semplicemente a massa: vogliamo, perciò, comandarne l'abilitazione con una sola porta del BS-2 (nell'esempio, P4).

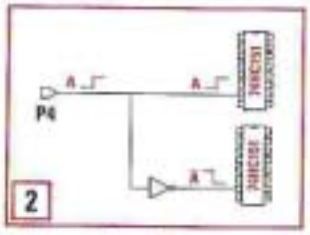
Mentre nel caso dell'espansione delle uscite si utilizzavano due diverse porte del BS-2 (in modo da poter comandare separatamente l'abilitazione dei due flip-flop, anche contemporanea), nel caso degli ingressi vogliamo invece che sia abilitato sempre e solo un multiplexer alla volta (per sfruttare l'operazione OR appena vista). Per questo, dunque, entrambi i multiplexer potrebbero dipendere da una stessa porta del microcontrollore, che ne abiliti ora uno, ora l'altro. Per inciso, così facendo, risparmieremo una porta del BS-2. Tuttavia, per poterlo fare, dobbiamo utilizzare un nuovo componente che, in particolare, realizzi una porta NOT e che sia posizionato prima dell'ingresso di uno dei due multiplexer (nel nostro caso il secondo). In sostanza, quello che dobbiamo costruire per i due multiplexer è un circuito cosiddetto 'di mutua esclusione' che, abilitando un multiplexer, disabiliti automaticamente l'altro.

Pin 74HC04	Ruoli		Collegamento
1	In	NOT1	P4 (BS-2)
2	Out		Enable (multiplexer 2)
3	In	NOT2	- (non usato)
4	Out		- (non usato)
5	In	NOT3	- (non usato)
6	Out		- (non usato)
7	Vss		Alimentazione (a massa)
8	Out	NOT4	- (non usato)
9	In		- (non usato)
10	Out	NOT5	- (non usato)
11	In		- (non usato)
12	Out	NOT6	- (non usato)
13	In		- (non usato)
14	Vdd		Alimentazione (5 volt)

Se ci limitassimo a collegare il pin Enable dei due multiplexer alla medesima porta (ad esempio P4), in parallelo, essi sarebbero comandati dallo stesso segnale e, quindi, risulterebbero avere sempre lo stesso stato (entrambi abilitati o entrambi disabilitati), come illustrato nel disegno 1 (sotto). Per il nostro scopo invece è necessario che un multiplexer sia comandato da un segnale e l'altro dalla negazione di quello stesso segnale. E qui entra in gioco il chip 74HC04 che, realizzando una porta NOT prima del pin di Enable di uno dei due multiplexer, funge da invertitore del segnale in ingresso sull'altro, ottenendo così la mutua esclusione (come illustrato sotto, nel disegno 2): inserendo in serie al secondo multiplexer un invertitore (NOT), a sua volta in parallelo con il primo multiplexer, si ottiene che entrambi i dispositivi siano sì comandati dalla stessa porta logica (P4) ma tramite segnali opposti, visualizzati infatti con porzioni d'onda quadra di andamento inverso: ossia da basso ad alto uno (⎍); da alto a basso l'altro (⎏). Nella tabella a sinistra sono stati riassunti i vari pin, il loro ruolo e i collegamenti che realizzeremo: il chip 74HC04 può realizzare ben 6 invertitori (vale a dire NOT logici), noi però ci limiteremo a usare solo il primo.

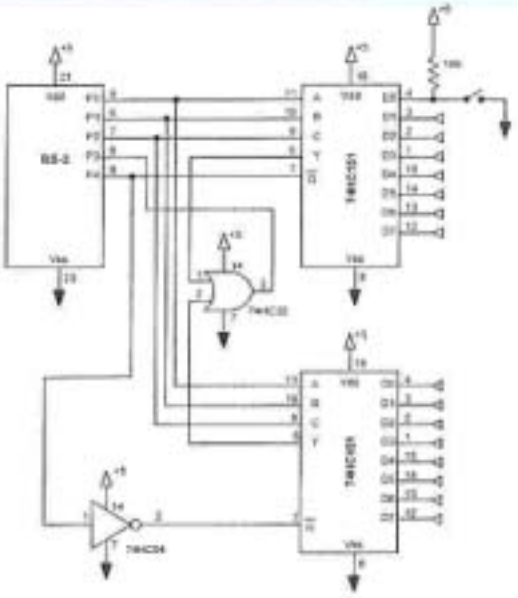


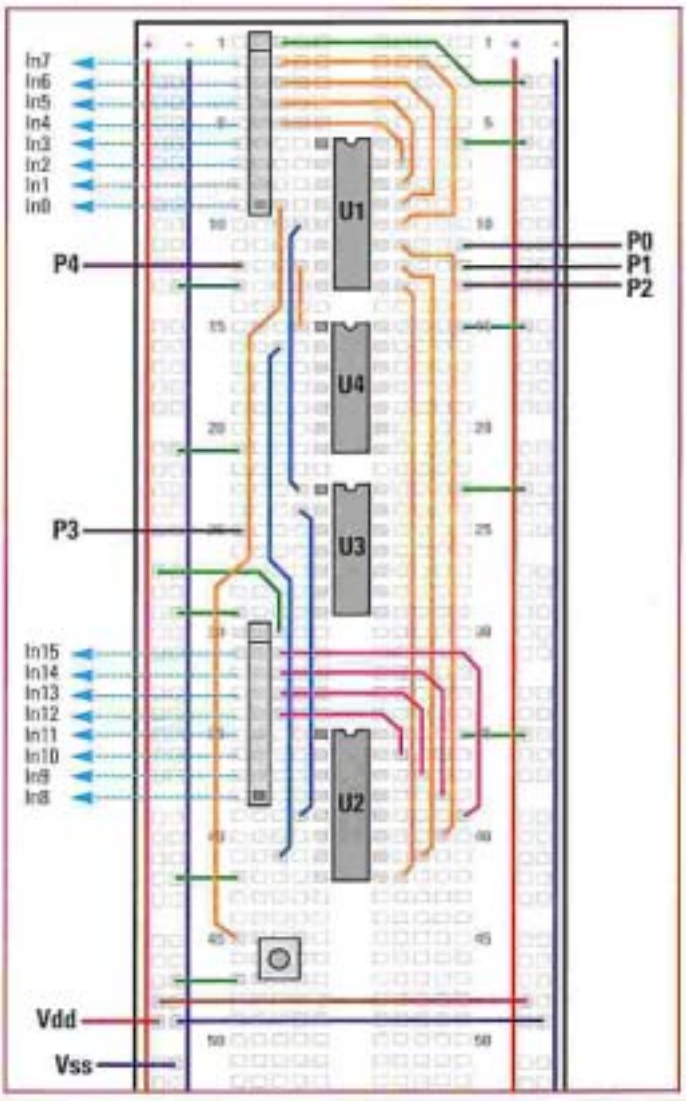




LE FASI DI PROGRAMMAZIONE

Come preannunciato, si tratta ora di collegare in parallelo l'Enable del primo multiplexer e il NOT che, a sua volta, va collegato in serie all'Enable del secondo multiplexer, come illustrato nello schema circuitale (a destra) e, come sempre, realizzando un circuito sulla breadboard (sotto: U1 e U2 sono i multiplexer 74HC151; U4 è il chip 74HC04 e U3 è il chip 74HC32). Così facendo, quando P4 avrà, ad esempio, un valore logico alto (1), anche l'Enable del primo multiplexer varrà 1 (e il dispositivo sarà perciò disabilitato), mentre quello del secondo multiplexer, grazie alla presenza del NOT, varrà 0 (sarà cioè abilitato): abilitando al massimo un multiplexer alla volta, sarà quindi possibile leggere l'uscita corretta. Ecco dunque raggiunto definitivamente il nostro scopo. Per inciso, dei 6 NOT logici realizzati dal chip 74HC04, si è scelto di utilizzare il primo NOT, ma è possibile utilizzarne uno qualunque, indifferentemente. Inoltre, avrai notato (dallo schema e dalla tabella della pagina precedente) che la struttura del 74HC04 è simile a quella vista per il chip 74HC32 (quello che realizzava le porte OR): anche i pin che





realizzano ogni porta NOT, infatti, sono raggruppati, ma a due a due: il primo funziona da ingresso (In) mentre l'altro ripropone in uscita (Out) il valore logico negato del primo. A questo punto, non resta che vedere cosa cambia a livello di gestione software degli ingressi. In merito, vediamo il listato sotto: innanzitutto è stata dichiarata la costante ENABLE_ che fa riferimento alla porta P4 e controlla i due multiplexer. Per quanto riguarda la selezione dell'ingresso, la procedura richiede che, configurando il valore logico della porta relativa a Enable, si selezioni il multiplexer da utilizzare (in questo caso, il primo). Quindi, come visto in precedenza, attraverso le porte P0, P1 e P2, si seleziona l'ingresso desiderato del multiplexer abilitato. Dato questo procedimento generale per selezionare un ingresso, è chiaro che la sequenza e il numero delle operazioni (e istruzioni) dipendono da come si intende sfruttare gli ingressi: se, dopo averne abilitato uno, si vuole passare a un altro ingresso dello stesso multiplexer, non è ovviamente necessario ripetere l'istruzione di selezione di quest'ultimo; analogamente, se il successivo ingresso da selezionare ha la stessa codifica del precedente, ma si trova sull'altro multiplexer, è sufficiente invertire il valore di Enable, senza ripetere le istruzioni relative alla selezione dell'ingresso.

prog2to2

```
'{$STAMP BS2}

'-------- Variabili ----------------

OUT_        var   in1

'-------- Costanti ----------------

ENABLE_     con   4

S1_         con   0
S2_         con   1
S3_         con   2

'-------- Programma Principale ------

low ENABLE_             'Seleziona il primo
                        'multiplexer

low S1_                 'Setta come ingresso
low S2_                 'l'entrata 0
low S3_

ciclo:

      debug ? OUT_

      pause 100

goto ciclo
```