CODICI BINARI

CODICE BCD (1° FILE)

Il nome di codice BCD deriva dall'acronimo di:

Binary Codec Decimal = codifica binaria del codice decimale.

In effetti, il codice BCD permette la codifica, mediante quattro bit primari, delle dieci cifre decimali del nostro sistema di numerazione.

DECIMALE	BCD
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001

Questo codice appartiene alla famiglia di codici solamente numerici.

Esso ha bisogno necessariamente di quattro bit per rappresentare tutte le cifre del sistema decimale, ma lascia inutilizzare ben **sei** delle possibili combinazioni dei quattro bit.

Ogni bit ha un valore in funzione della propria posizione occupata (**sistema pesato**).

Il numero di bit necessari per esprimere una cifra decimale in BCD è più elevato di quello necessario per codificarlo in binario puro.

Il codice BCD gode di un'interessante proprietà relativa alla somma:

- si sommano i quatto omologhi dei quattro bit che costituiscono le singole cifre degli addendi.
- Si potranno quindi avere dei riporti, se il risultato dovesse superare i codici esprimibili con 4 bit, dovremmo aggiungere **6 = 0110** al risultato stesso.
- Se invece non c' è riporto, ma il risultato della somma supera i quattro bit e cade in una delle sei combinazioni che non hanno significato in codice BCD, dovremmo, anche qui, aggiungere 6 = 0110 al risultato stesso.

In egual modo risolveremo le sottrazioni.

Esempi di operazioni (somma e sottrazione)

SOMMA

$(520)_{10} + (150)_{10}$

0101	**0010**	**0000**	**520 +**
0001	**0101**	**0000**	**150 =**
0110	**0111**	**0000**	**670**

$(526)_{10} + (159)_{10}$

0101	**0010**	0110	**526 +**
0001	0101	1001	**159 =**
0110	0111	1111	
		0110	
0110	1000	0101	**685**

SOTTRAZIONE

$(526)_{10} - (159)_{10}$

0101	**0010**	**0110**	**526 –**
0001	**0101**	**1001**	**159 =**
0011	**1100**	**1101**	
		0110	
0011	**0110**	**0111**	**367**

CODICE BCD (2° FILE)

Binary Code Decimal, usato nelle tastiere, nei display e nelle applicazioni gestionali.

DEC	BCD
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001

Per esempio, $\mathbf{(379)_{10} = (\ 0011\ 0111\ 1001)_{BCD}}$

Quello riportato in tabella, detto anche **codice 8421,** è un codice pesato, in quanto ogni elemento del carattere, a partire da destra (LSB) e procedendo verso sinistra, ha peso rispettivamente 1, 2, 4, 8.

Ad esempio, $(0101)_{BCD} = 0*8+1*4+0*2+1*1 = (5)_{10}$

Delle possibili combinazioni ottenibili con quattro bit, solamente dieci sono utilizzate, i restanti sei caratteri possibili **(1010, 1011, 1100, 1101, 1110, 1111)** non hanno significato nel codice BCD. Per questo motivo il codice BCD è detto **ridondante.**

Questa rappresentazione è ancora meno compatta di quella binaria, in quanto, per esempio, con un byte sono rappresentabili:

in binario da $(0)_{10}$ a $(255)_{10}$ **in esadecimale da $(0)_H$ a $(FF)_H$**

in BCD da $(0)_{10}$ a $(99)_{10}$

I numeri e le operazioni in questo codice, pur rispondendo alle normali regole aritmetiche, sono di difficile trattazione mnemonica, quando si eseguono operazioni di somma o sottrazione, a causa della ridondanza appena definita. Ad esempio:

5 + **0101 +**

8 = **1000 =**

---- **---------**

13 **1101**

Utilizzando le normali regole della somma il risultato è **1101**, però tale valore non esiste in BCD, è infatti tra i caratteri privi di significato, mentre equivale al numero 13 in binario puro.

Il valore 13 espresso in BCD è invece: 0001 0011 (4 bit per ogni cifra decimale)

La discordanza tra i due valori, dovuta alla ridondanza del codice, sta nel fatto che al risultato della somma, in BCD, occorre aggiungere $\mathbf{(6)_{10}}$ = $\mathbf{(0110)_{BCD}}$ **ogni volta che viene superato il numero 9**, in quanto di sei posizioni è il salto che occorre fare per evitare i caratteri privi di significato in quel codice.

È appunto il caso dell'esempio precedente, pertanto per correggere il risultato occorre eseguire la seguente operazione:

1 1 0 1 + (primo risultato)

0 1 1 0 =

----

0 0 0 1 0 0 1 1

Questo accade tutte le volte che con la somma, nel sistema decimale, si ottiene un riporto, poiché in questo caso la cifra del risultato supera il nove.

35 + **0011 0101 +**

5 = **0101 =**

----- **----------**

40 **0011 1010 +**

0110 =

0100 0000

Analogamente, per le sottrazioni in BCD, si pone il problema della correzione del risultato ogni volta che, nell'eseguire una differenza decimale, occorre un prestito dalla cifra precedente.

Oltre alla complicazione già evidenziata nelle operazioni aritmetiche, che per altro normalmente sono eseguite da automatismi logici, il codice BCD presenta lo svantaggio di richiedere un maggior numero di bit rispetto al codice binario puro.

CODICE GRAY (1° file)

Il **codice Gray** è un codice binario, composto da un qualsiasi numero di bit maggiore di due, nel quale il passaggio da un numero rappresentato al successivo (o precedente) comporta **la variazione di un solo bit**.

Decimale	Binario	Gray
0	0000	0000
1	0001	0001
2	0010	0011
3	0011	0010
4	0100	0110
5	0101	0111
6	0110	0101
7	0111	0100
8	1000	1100
9	1001	1101
10	1010	1111
11	1011	1110
12	1100	1010
13	1101	1011
14	1110	1001
15	1111	1000

Se realizziamo una tabella di confronto tra una numerazione binaria ed una Gray possiamo vedere con facilità in cosa consista questa codifica.

Se prendiamo ad esempio il passaggio da 1 a 2 vediamo che gli equivalenti binari sostituiscono DUE cifre contemporaneamente:

Decimale	Binario	Gray
1	01	01
2	10	11

Nel codice Gray, invece, **solamente una cifra viene cambiata**.

E vediamo questo applicato a tutti i successivi numeri della sequenza.

Questa caratteristica permette al progettista di eseguire il controllo degli errori in operazioni con dispositivi elettronici che trattano sequenze binarie provenienti da dispositivi esterni al calcolatore, come ad esempio codificatori elettromeccanici, encoder,

trasmissioni seriali di dati, ecc.

La codifica diventa essenziale per trattare sequenze di valori binari, dato che il passaggio da un numero al successivo deve presentare **una sola variazione di bit ed è quindi possibile identificare se si è verificato un errore di input** al momento dell'acquisizione della sequenza.

Convertire il binario in Gray

Per **convertire un numero binario in codice Gray** è molto semplice: si utilizza la funzione **XOR**, che è definibile come una somma in modulo 2.

Si effettua l' XOR tra il numero binario e se stesso shiftato di una posizione verso destra. Ad esempio:

1011

1011

1110

Da notare che la prima cifra del codice Gray è la stessa del numero binario.

CODICE GRAY (2° file)

Non è conveniente realizzare encoder assoluti che utilizzino una codifica binaria perché, l'uso di questa codifica, associato all'imperfetto allineamento dei rilevatori ottici, è causa di errori molto rilevanti.

La lettura non avviene simultaneamente provocando errori come mostrato in questo esempio:

Es. Supponiamo di avere un encoder dotato di 4 rilevatori ottici. Un simile encoder è pertanto a 4 bit.

Supponiamo che in un dato istante esso legga il valore associato alla posizione 7 che nel caso di codifica binaria corrisponde al numero (0111). E' ovvio che con la rotazione il successivo segnale dovrà rilevare la posizione del settore numero 8 che in binario vale (1000).

Supponiamo che, a causa dei problemi di allineamento, il bit più significativo (il bit 3 della tabella) commuti prima dei restanti quattro.

Questo vuol dire che per alcuni istanti il valore letto dall' encoder sarà (1111) ovvero l' encoder legge il valore 15 che è **totalmente errato**.

	Bit 3	Bit 2	Bit 1	Bit 0
Valore iniziale	**0**	**1**	**1**	**1**
Il bit 3 commuta prima	**1**	**1**	**1**	**1**
Alla fine commutano tutti	**1**	**0**	**0**	**0**

Questi spiacevoli inconvenienti possono essere eliminati sostituendo al codice binario il codice Gray.

La caratteristica di questo codice è quella che nel passaggio da un dato valore al successivo (o al precedente) cambia un solo bit evitando il problema delle differenti velocità di commutazione che si possono verificare quando cambiano più bit contemporaneamente.

Il codice Gray è descritto da questa tabella:

Valore Decimale	Codice Binario				Codice Gray			
1	**0**	**0**	**0**	**1**	**0**	**0**	**0**	**1**
2	**0**	**0**	**1**	**0**	**0**	**0**	**1**	**1**
3	**0**	**0**	**1**	**1**	**0**	**0**	**1**	**0**
4	**0**	**1**	**0**	**0**	**0**	**1**	**1**	**0**
5	**0**	**1**	**0**	**1**	**0**	**1**	**1**	**1**
6	**0**	**1**	**1**	**0**	**0**	**1**	**0**	**1**
7	**0**	**1**	**1**	**1**	**0**	**1**	**0**	**0**
8	**1**	**0**	**0**	**0**	**1**	**0**	**0**	**0**
9	**1**	**0**	**0**	**1**	**1**	**0**	**0**	**1**
10	**1**	**0**	**1**	**0**	**1**	**0**	**1**	**1**
11	**1**	**0**	**1**	**1**	**1**	**0**	**1**	**0**
12	**1**	**1**	**0**	**0**	**1**	**1**	**1**	**0**
13	**1**	**1**	**0**	**1**	**1**	**1**	**1**	**1**
15	**1**	**1**	**1**	**0**	**1**	**1**	**0**	**1**
15	**1**	**1**	**1**	**0**	**1**	**1**	**0**	**0**

Questo codice è detto anche codice riflesso perché ogni riga si ottiene mediante opportune riflessioni.

Conversione da binario a Gray

La conversione da codice binario a codice Gray si effettua confrontando il bit binario B_i con il bit immediatamente superiore B_{i+1}. Se B_i e B_{i+1} risultano uguali allora il corrispondente bit del codice Gray, G_i , sarà uguale a zero.

In caso contrario G_i sarà uguale ad uno.

In altre parole si effettua l'operazione booleana XOR:

XOR		
B_i	B_{i+1}	G_i
0	0	0
0	1	1
1	0	1
1	1	0

Riportiamo di seguito un esempio chiarificatore:

Es. Si consideri il numero binario:

B = 10110

Il numero non cambia se aggiungo uno zero alla sua sinistra:

B = 010110

$B_5 = 0$	$B_4 = 1$	$B_3 = 0$	$B_2 = 1$	$B_1 = 1$	$B_0 = 0$

Per quanto detto:

$G_4 = B_5$ XOR $B_4 = 0$ XOR 1 = 1

$G_3 = B_4$ XOR $B_3 = 1$ XOR 0 = 1

$G_2 = B_3$ XOR $B_2 = 0$ XOR 1 = 1

$G_1 = B_2$ XOR $B_1 = 1$ XOR 1 = 0

$G_0 = B_1$ XOR $B_0 = 1$ XOR 0 = 1

Quindi il codice Gray corrispondente è:

G = 11101