

Premessa.....	2
Codice Binario a 4 bit .....	2
Codice Ottale (Octal) a 3 bit .....	3
Codice BCD Decimale codificato in binario.....	4
Codice HEX Esadecimale a 4 bit .....	5
<i>Esercizio 1:</i> .....	5
<i>Esercizio 2:</i> .....	5
Codice Eccesso 3.....	6
Rappresentazione in complemento a 1 .....	6
<i>Esempio :</i> .....	6
Rappresentazione in complemento a 2 .....	6
<i>Esercizio 1:</i> .....	7
<i>Esercizio 2:</i> .....	7
Complemento a 9 di un dato decimale.....	7
CODICE ASCII (American Standard Code for Information Interchange) .....	8
Caratteri di controllo del codice ASCII.....	9
Significato dei caratteri di controllo del codice ASCII .....	9
Esadecimale codificato ASCII .....	11
S- Records.....	11
Codice Gray .....	12
Codice Baudot.....	14
Cenni storici e caratteristiche .....	14
Il Codice Fiscale .....	16
Cos'è il Codice Fiscale.....	16
Struttura del Codice Fiscale.....	16
Un codice fiscale di esempio .....	16
L'algoritmo di calcolo.....	16
Cognome.....	16
Giorno .....	18
CODICI MECCANOGRAFICI di alcuni COMUNI della prov. di Brescia .....	18
CODICI MECCANOGRAFICI di alcuni STATI ESTERI .....	19
Il codice di controllo .....	20
IL CODICE MORSE (Alfabeto MORSE).....	21
Alfabeto fonetico NATO .....	23
Codice EBCDIC.....	24
Codice dei Colori RGB (R red – G green – B blue).....	24
Tabella dei principali colori RGB .....	25
Codice ISBN (International Standard Book Number).....	25
Formato .....	26
ISBN-10 e ISBN-13.....	26
Che cosa è il codice ISBN .....	26
Codice a barre .....	27
Tipi di codici a barre.....	28
Elementi di cifratura di un messaggio ASCII.....	28
Controllo e recupero errori.....	29
Controllo di Parità .....	30
Check di ridondanza longitudinale LRC (Longitudinal Redundancy Check) .....	30
Checksum.....	30

## Codici-formati

prof. Cleto Azzani  
 IPSIA Moretto Brescia  
 (Ottobre 2007)

## Premessa

Ho raccolto in questo fascicolo una serie di informazioni riguardanti i codici. Fra i codici ho ritenuto opportuno inserire quelli più importanti o semplici da capire che comunque un tecnico diplomato deve conoscere; volutamente ho tralasciato tutte le nozioni teoriche riguardanti le problematiche relative alla correzione degli errori mentre ho ritenuto opportuno trattare anche sommariamente le problematiche relative alla rilevazione degli errori. Ho inserito alcuni dei codici che hanno fatto storia nel mondo delle telecomunicazioni anche se, ormai oggi obsoleti (Morse e codice delle telescriventi). Ho inserito il Codice Fiscale, il codice dei colori RGB (schede grafiche).

## Codice Binario a 4 bit

Con 4 bit posso rappresentare 16 combinazioni binarie da (0000) a (1111) a cui corrispondono valori decimali compresi fra 0 (0000) e 15 (1111).

PESI CIFRE				VALORE
8	4	2	1	DECIMALE
0	0	0	0	0
0	0	0	1	1
0	0	1	0	2
0	0	1	1	3
0	1	0	0	4
0	1	0	1	5
0	1	1	0	6
0	1	1	1	7
1	0	0	0	8
1	0	0	1	9
1	0	1	0	10
1	0	1	1	11
1	1	0	0	12
1	1	0	1	13
1	1	1	0	14
1	1	1	1	15

Esistono metodi generali per convertire quantità numeriche decimali nell'equivalente rappresentazione binaria (metodo dei resti delle successive divisioni x 2) e per fare la conversione in senso opposto. Ripassiamole rapidamente con due esempi. Poniamo ad esempio di dover convertire la quantità 1237. Il metodo da seguire è schematizzato nel prospetto che segue:

**1237**

Quozienti	Resti
div. per 2	
618	1
309	0
154	1
77	0
38	1
19	0
9	1
4	1
2	0
1	0
0	1

Effettuando le successive divisioni x 2 da 1237 si passa a 618 con resto 1 poi a 309 con resto 0 e così via dicendo fino a 0 con resto 1. Ora si leggono le cifre binarie della colonna resti partendo dal basso verso l'alto si ottiene il valore binario 100 1101 0101 costituito da 11 bit.

Ora proviamo a riconvertire in Decimale il valore binario 100 1101 0101 per fare ciò poniamo sopra ogni cifra binaria il corrispettivo peso partendo da destra e verso sinistra  $2^0=1$   $2^1=2$   $2^2=4$   $2^3=8$   $2^4=16$  fino a  $2^9=512$   $2^{10}=1024$ . Effettuiamo i prodotti fra peso e cifra binaria ed effettuiamo la somma che come si nota da il risultato 1237.

1024	512	256	128	64	32	16	8	4	2	1	
<b>1</b>	<b>0</b>	<b>0</b>	<b>1</b>	<b>1</b>	<b>0</b>	<b>1</b>	<b>0</b>	<b>1</b>	<b>0</b>	<b>1</b>	
1024	0	0	128	64	0	16	0	4	0	1	<b>1237</b>

### Codice Ottale (Octal) a 3 bit

Con 3 bit posso rappresentare 8 combinazioni binarie da (000) a (111) a cui corrispondono valori decimali compresi fra 0 (000) e 7 (111).

PESI CIFRE			VALORE DECIMALE
4	2	1	
0	0	0	0
0	0	1	1
0	1	0	2
0	1	1	3
1	0	0	4
1	0	1	5
1	1	0	6
1	1	1	7

Esistono metodi generali per convertire quantità numeriche decimali nell'equivalente rappresentazione ottale (metodo dei resti delle successive divisioni x 8) e per fare la conversione in senso opposto. Ripassiamole rapidamente con due esempi. Poniamo ad esempio di dover convertire la quantità 1237. Il metodo da seguire è schematizzato nel prospetto che segue:

**1237**

Quozienti	Resti
div. per 8	
154	<b>5</b>
19	<b>2</b>
2	<b>3</b>
0	<b>2</b>

Effettuando le successive divisioni x 8 da 1237 si passa a 154 con resto 5 poi a 19 con resto 2 e così via dicendo fino a 0 con resto 2. Ora si leggono le cifre ottali della colonna resti partendo dal basso verso l'alto si ottiene il valore ottale 2 3 2 5 costituito da 4 cifre ottali.

Ora proviamo a riconvertire in Decimale il valore ottale 2 3 2 5 per fare ciò poniamo sopra ogni cifra binaria il corrispettivo peso partendo da destra e verso sinistra  $8^0=1$   $8^1=8$   $8^2=64$   $8^3=512$ . Effettuiamo i prodotti fra peso e cifra ottale ed effettuiamo la somma che come si nota da il risultato 1237.

512	64	8	1	
<b>2</b>	<b>3</b>	<b>2</b>	<b>5</b>	
1024	192	16	5	<b>1237</b>

Da ultimo sostituiamo alla rappresentazione ottale 2 3 2 5 la corrispondente “versione binaria” ottenuta sostituendo ad ogni cifra ottale il corrispondente valore binario otteniamo 010 011 010 101 ossia la rappresentazione di 1237 binaria a 12 bit perfettamente uguale a quella precedentemente ricavata a pag.3.

### Codice BCD Decimale codificato in binario

La codifica **Binary-coded decimal (BCD)** è un modo comunemente utilizzato in informatica ed elettronica per rappresentare le cifre decimali in codice binario.

valore decimale	combinazioni 4 bit	simboli codice BCD
0	0000	<b>0</b>
1	0001	<b>1</b>
2	0010	<b>2</b>
3	0011	<b>3</b>
4	0100	<b>4</b>
5	0101	<b>5</b>
6	0110	<b>6</b>
7	0111	<b>7</b>
8	1000	<b>8</b>
9	1001	<b>9</b>
10	1010	
11	1011	
12	1100	
13	1101	
14	1110	
15	1111	

In questo formato ogni cifra di un numero è rappresentata da un codice binario di quattro bit, il cui valore è compreso tra 0 (0000) e 9 (1001). Le restanti sei combinazioni tra (1010) e (1111) non sono utilizzate dal codice BCD. Per esempio il numero 127 è rappresentato in BCD come 0001, 0010, 0111.

Poiché i computer memorizzano i dati in byte di otto bit, è possibile memorizzare una cifra per byte e riempire i restanti quattro bit con zeri o uno (come nel codice EBCDIC), oppure mettere due cifre per byte, modalità chiamata *packet BCD*. I numeri packet BCD normalmente terminano con un codice di segno, solitamente 1100 per il + e 1101 per il meno. La cifra 127 si rappresenta 11110001, 11110010, 11110111 in EBCDIC e 00010010, 01111100 in packet BCD.

Il codice BCD è molto usato in elettronica, specialmente in circuiti digitali privi di microprocessore, perché facilita la visualizzazione di lunghe cifre su display a sette segmenti, infatti ad ogni display fisico corrisponde esattamente una cifra. Esistono appositi circuiti integrati (decoder 9368P TTL e 4511 CMOS) che effettuano la conversione da BCD nella corrispondente combinazione di accensione dei segmenti. Anche l'esecuzione di semplici calcoli aritmetici è più semplice da effettuarsi su cifre BCD per circuiti logici combinatori.

Il BIOS dei personal computer memorizzano la data e l'ora in formato BCD, presumibilmente per ragioni storiche.

## Codice HEX Esadecimale a 4 bit

La codifica **Esadecimale** (HEX) è un modo comunemente utilizzato in informatica ed elettronica per rappresentare in forma compatta quantità binarie.

valore decimale	combinazioni 4 bit	simboli codice HEX (esadecimale)
0	0000	<b>0</b>
1	0001	<b>1</b>
2	0010	<b>2</b>
3	0011	<b>3</b>
4	0100	<b>4</b>
5	0101	<b>5</b>
6	0110	<b>6</b>
7	0111	<b>7</b>
8	1000	<b>8</b>
9	1001	<b>9</b>
10	1010	<b>A</b>
11	1011	<b>B</b>
12	1100	<b>C</b>
13	1101	<b>D</b>
14	1110	<b>E</b>
15	1111	<b>F</b>

In questo caso, a differenza della codifica BCD, vengono sfruttate tutte e 16 le combinazioni binarie a 4 bit; le prime 10 da 0 a 9 rappresentano quantità decimali pertanto i simboli usati sono gli stessi del codice BCD le 10 cifre da 0 a 9 alle combinazioni da 10 a 15 vengono associate le lettere dell'alfabeto da A ad F: A (1010), B (1011), C (1100) e così via fino ad F (1111).

Per distinguere una sequenza di cifre decimali da una analoga sequenza di cifre esadecimale queste ultime devono essere precedute dal simbolo \$ (notazione Motorola) oppure seguite dal simbolo H (notazione Intel).

### *Esercizio 1:*

Rappresentare in Codice Esadecimale la quantità 1237. Abbiamo a suo tempo visto che risulta:  $1237_{10} = (100\ 1101\ 0101)_2$  Raggruppando le cifre binarie in gruppi da 4 bit da destra a sinistra ed interpretando le combinazioni a 4 bit si ottiene la equivalente rappresentazione esadecimale \$ 4D5 = 4D5 H.

### *Esercizio 2:*

Convertire direttamente in Codice Esadecimale la quantità 1237 usando il metodo dei resti delle successive divisioni per 16. Passare dalla rappresentazione esadecimale alla rappresentazione binaria equivalente.

### Codice Eccesso 3

Nella rappresentazione dei valori decimali esiste la possibilità di utilizzare il codice "Eccesso 3" in pratica la quantità 0 viene codificata come si trattasse del valore  $0 + 3 = 3$ ; la quantità 1 viene codificata con il valore 4 ed infine la quantità 9 viene codificata come si trattasse del valore 12 ossia \$C in binario (1100).

valore decimale	codice eccesso 3	simboli codice BCD
3	0011	<b>0</b>
4	0100	<b>1</b>
5	0101	<b>2</b>
6	0110	<b>3</b>
7	0111	<b>4</b>
8	1000	<b>5</b>
9	1001	<b>6</b>
10	1010	<b>7</b>
11	1011	<b>8</b>
12	1100	<b>9</b>

### Rappresentazione in complemento a 1

Nella rappresentazione di valori binari si incontra talora la necessità di effettuare la conversione "in complemento a 1" in pratica si tratta di cambiare nella stringa binaria ogni 1 in 0 e ogni 0 in 1. Il circuito elettronico che effettua tale operazione è una o più porte NOT.

*Esempio :*

Rappresentare "in complemento ad 1" la "stringa binaria "0100 1101 0101. Effettuando l'operazione NOT su ogni singolo bit si ottiene il valore 1011 0010 1010. In altri termini dal valore \$ 4D5 si passa al valore \$ B2A.

Si noti che sommando fra di loro le cifre omologhe del dato e del complemento ad 1 si ottiene sempre la quantità \$F :

$$\text{\$ } 4 + \text{\$ } B = \text{\$ } F$$

$$\text{\$ } D + \text{\$ } 2 = \text{\$ } F$$

$$\text{\$ } 5 + \text{\$ } A = \text{\$ } F$$

### Rappresentazione in complemento a 2

La rappresentazione in complemento a 2 nasce per poter rappresentare attraverso una codifica binaria sia quantità intere positive che quantità intere negative. Prendiamo ad esempio un numero espresso da un gruppo di 8 bit; con essi posso rappresentare  $2^8=256$  combinazioni differenti; posso quindi fare due tipi di scelte o assegno alle 256 combinazioni quantità numeriche prive di segno ossia UNSIGNED partirò dal valore 0 (0000 0000) e terminerò al valore 255 (1111 1111). L'altra alternativa è quella di suddividere le 256 combinazioni in due gruppi: 128 combinazioni assegnate ai numeri positivi (da +0 a +127) e 128 combinazioni assegnate a numeri negativi (da -1 a -128) in questo ultimo caso si parla di rappresentazione in "COMPLEMENTO A DUE". Può essere interessante osservare come da un numero che esprime una entità positiva si possa ottenere un numero che rappresenti la stessa quantità negativa (o meglio In Complemento a 2) e viceversa.

A titolo di esempio supponiamo di avere il dato binario ad 8 bit di seguito riportato e rappresentato dal valore esadecimale \$15 equivalente alla quantità decimale 21

0	0	0	1	0	1	0	1	\$15	21
---	---	---	---	---	---	---	---	------	----

Supponiamo di voler convertire tale valore nel corrispondente valore negativo -21

Il procedimento è relativamente semplice: prima procedo ad elaborare il “complemento ad 1” del dato di partenza, poi aggiungo 1 (somma aritmetica) ed ottengo così il “complemento a 2”.

0	0	0	1	0	1	0	1	\$15	21	DATO DI PARTENZA
1	1	1	0	1	0	1	0			
								+		
								1		
1	1	1	0	1	0	1	1	\$EB	-21	COMPLEMENTO A 2

E' evidente che se volessi ricavare il “complemento a 2” della quantità \$EB (valore negativo) otterrò ovviamente il valore \$15 (valore positivo).

E' interessante osservare che se sommo aritmeticamente il valore \$15 al valore \$EB il risultato non può che essere \$00 (risultato ad 8 bit).

*Esercizio 1:*

Rappresentare in complemento a 2 il valore esadecimale \$1234

*Esercizio 2:*

Trovare la rappresentazione positiva del dato esadecimale \$E2A5.

**Complemento a 9 di un dato decimale**

Data una quantità decimale espressa attraverso un determinato numero di cifre decimali, la rappresentazione in complemento a 9 si ottiene sostituendo a ciascuna cifra originale il risultato che si ottiene sottraendo a 9 il valore espresso dalla cifra originale.

Esempio 1 : Data la quantità 8345 (decimale a 4 cifre) trovare il complemento a 9 significa sostituire al dato iniziale il risultato della sottrazione 9999 - 8345 ossia 1654; si noti che l'operazione è estremamente semplice e soprattutto può essere fatta cifra x cifra (senza fastidiosi riporti).

La rappresentazione in complemento a 9 è nato per la necessità di trasformare, in campo decimale, una sottrazione in una somma (in complemento a 9).

Esempio 2 : Dovendo effettuare l'operazione 12347 - 3477 posso procedere a calcolare il complemento a 9 a 5 cifre di 3477 esso risulta immediatamente 96522; a questo punto sommo al dato 12347 la quantità 96522 aggiungo alla fine 1 (complemento a 10) e trascuro le cifre eccedenti le 5.

	<b>1</b>	<b>2</b>	<b>3</b>	<b>4</b>	<b>7</b>	DATO A
	9	6	5	2	2	COMPLEMENTO A 9 DI B
<b>1</b>	<b>0</b>	<b>8</b>	<b>8</b>	<b>6</b>	<b>9</b>	SOMMA
					<b>1</b>	1
		<b>8</b>	<b>8</b>	<b>7</b>	<b>0</b>	RISULTATO

## CODICE ASCII (American Standard Code for Information Interchange)

Il codice ASCII è stato introdotto per uniformare i protocolli di conversazione fra unità diverse di un sistema di elaborazione dati ma prodotte e commercializzate da costruttori diversi; è un codice che è nato a 7 bit e che quindi presenta 128 combinazioni possibili è stato poi esteso successivamente a 8 bit in ambito PC (ASCII esteso). Le prime 32 combinazioni (da \$00 a \$1F) e l'ultima (\$7F) costituiscono i cosiddetti caratteri di controllo vale a dire codici che non corrispondono a caratteri stampabili ma piuttosto a comandi diretti verso l'unità ricevente per attivare particolari comportamenti della stessa: ad esempio il codice \$07 attiva il cicalino (bell) sulla unità ricevente, il codice \$0D porta a capo il cursore, il codice \$0A porta il cursore sulla linea successiva, ecc.. I caratteri veri e propri (quelli di una tastiera) occupano i codici da \$20 (Spazio prodotto dalla barra spaziatrice) al codice \$7E (carattere tilde ~). Le cifre numeriche occupano le combinazioni fra \$30 (la cifra 0) e \$39 (la cifra 9). In tabella sono riportati i caratteri del codice ASCII standard a 7 bit; per individuare il codice di un determinato simbolo bisogna individuare la colonna di appartenenza (MSD cifra più significativa a 3 bit) e successivamente la riga di appartenenza (LSD cifra meno significativa a 4 bit. Esempio il simbolo "\$" appartiene alla colonna 2 e riga 4 pertanto la sua codifica ASCII è data dalla combinazione \$24 o %00100100, il simbolo "+" appartiene alla colonna 2 e riga B pertanto la sua codifica ASCII è data dalla combinazione \$2B o %00101011, la lettera "A" appartiene alla colonna 4 e riga 1 pertanto la sua codifica è data dalla combinazione \$41 o %01000001, la lettera "a" appartiene alla colonna 6 e riga 1 pertanto la sua codifica è data dalla combinazione \$61 o %01100001.

N.B.

*Il carattere "\$" o la lettera "H" vengono utilizzati per contrassegnare espressioni di tipo esadecimale, "\$" precede mentre "H" segue un valore esadecimale; es.: \$4F (notazione Motorola) equivale a scrivere 4FH (notazione Intel).*

*Il carattere "%" precede sempre quantità di tipo binario; es.: %01011010.*

**Tabella del codice ASCII**

	0	1	2	3	4	5	6	7
0	NUL	DLE	SP	0	@	P	`	p
1	SOH	DC1	!	1	A	Q	a	q
2	STX	DC2	"	2	B	R	b	r
3	ETX	DC3	#	3	C	S	c	s
4	EOT	DC4	\$	4	D	T	d	t
5	ENQ	NAK	%	5	E	U	e	u
6	ACK	SYN	&	6	F	V	f	v
7	BEL	ETB	`	7	G	W	g	w
8	BS	CAN	(	8	H	X	h	x
9	HT	EM	)	9	I	Y	i	y
A	LF	SUB	*	:	J	Z	j	z
B	VT	ESC	+	;	K	[	k	{
C	FF	FS	,	<	L	\	l	
D	CR	GS	-	=	M	]	m	}
E	SO	RS	.	>	N	^	n	~
F	SI	US	/	?	O	_	o	DEL

\$2E punto (.)    \$60 accento (`)    \$27 apostrofo (')    \$2C virgola (,)

## Caratteri di controllo del codice ASCII

I caratteri di controllo del codice ASCII (prime trentadue combinazioni da \$00 a \$1F) possono essere generati dalla tastiera di un PC attraverso la pressione di una opportuna combinazione di due tasti della tastiera: il tasto CTRL (in basso a sinistra) e un tasto alfanumerico come indicato nella tabella seguente. Da sottolineare che il tasto CTRL va premuto per primo e va mantenuto premuto mentre si preme il secondo tasto della sequenza. Esempio per generare il carattere di controllo BEL (\$07) devo premere e mantenere premuto il tasto CTRL e poi premere il tasto G (maiuscola).

### Generazione dei Caratteri di controllo

hex	car	sequenza	hex	car	sequenza
00	NUL	CTRL @	10	DLE	CTRL P
01	SOH	CTRL A	11	DC1	CTRL Q
02	STX	CTRL B	12	DC2	CTRL R
03	ETX	CTRL C	13	DC3	CTRL S
04	EOT	CTRL D	14	DC4	CTRL T
05	ENQ	CTRL E	15	NAK	CTRL U
06	ACK	CTRL F	16	SYN	CTRL V
07	BEL	CTRL G	17	ETB	CTRL W
08	BS	CTRL H	18	CAN	CTRL X
09	HT	CTRL I	19	EM	CTRL Y
0A	LF	CTRL J	1A	SUB	CTRL Z
0B	VT	CTRL K	1B	ESC	CTRL [
0C	FF	CTRL L	1C	FS	CTRL \
0D	CR	CTRL M	1D	GS	CTRL ]
0E	SO	CTRL N	1E	RS	CTRL ^
0F	SI	CTRL O	1F	US	CTRL _

## Significato dei caratteri di controllo del codice ASCII

NUL : Null character.

SOH : Start of Heading/Inizio intestazione di un messaggio (informazione di istradamento del messaggio).

STX : Start of Text/Inizio del messaggio vero e proprio e quindi anche fine dell'intestazione del messaggio iniziata con un SOH.

ETX : End of Text/Fine del messaggio iniziato con un STX.

EOT : End of Transmission/Fine della trasmissione: conclude la trasmissione di uno o più testi ciascuno dei quali deve essere concluso da ETX.

ENQ : Enquiry/Richiesta di identificazione inviata al dispositivo ricevente.

BEL : Bell/Attivazione del segnale acustico sul dispositivo ricevente.

BS : Backspace/Comando di retrocedere di un carattere sulla unità di stampa del messaggio (video o stampante).

HT : Horizontal Tabulation/Tabulazione orizzontale.

LF : Line Feed/Avanzamento sulla successiva riga di stampa.

VT : Vertical Tabulation/Tabulazione verticale.

FF : Formed Feed/Passaggio sulla prima linea di stampa del foglio successivo .

CR : Carriage Return/Ritorno a capo (sulla medesima linea di stampa).

SO : Shift Out/Disinserione della unità ricevente: le combinazioni di codice seguente saranno interpretate in modo non conforme alla tabella del codice ASCII finchè alla unità ricevente non giungerà un codice di inserzione SI.

SI : Shift In/Inserzione della unità ricevente: le combinazioni di codice seguente saranno interpretate in modo conforme alla tabella del codice ASCII.

DLE : Data Link Escape/Esclusione collegamenti dati.

DC1 - DC2 : Device Control characters/Caratteri di controllo particolari.

ACK : Acknowledge/Messaggio di dati ricevuti correttamente inoltrato verso il dispositivo trasmittente dalla unità ricevente.

NACK: Negative Acknowledge/Messaggio di dati ricevuti scorrettamente inoltrato verso il dispositivo trasmittente dalla unità ricevente.

SYN : Synchronize/Segnale di sincronismo emesso da una unità trasmittente sincrona (in condizioni di riposo) per rendere possibile il sincronismo della unità ricevente in caso di inizio-trasmissione di un messaggio.

ETB : End of Transmission Block/Fine di un blocco di trasmissione o eventualmente separazione tra più blocchi trasmessi.

CAN : Cancel/Il dato è in errore o deve essere trascurato.

EM : End of Medium/Fine del Supporto di Informazione.

SUB : Separatore.

ESC : Escape.

FS : Field Separator/Separatore di Campo.

GS : Group Separator/Separatore di Gruppo.

RS : Record Separator/Separatore di Record.

US : Unit Separator/Separatore di Unità.

DELETE : Delete/Cancellazione di caratteri errati.

		COLONNE								
		0	16	32	48	64	80	96	112	
		0	1	2	3	4	5	6	7	
RIGHE	0	0	NUL	DLE	SP	0	@	P	`	p
	1	1	SOH	DC1	!	1	A	Q	a	q
	2	2	STX	DC2	"	2	B	R	b	r
	3	3	ETX	DC3	#	3	C	S	c	s
	4	4	EOT	DC4	\$	4	D	T	d	t
	5	5	ENQ	NAK	%	5	E	U	e	u
	6	6	ACK	SYN	&	6	F	V	f	v
	7	7	BEL	ETB	'	7	G	W	g	w
	8	8	BS	CAN	(	8	H	X	h	x
	9	9	HT	EM	)	9	I	Y	i	y
	10	A	LF	SUB	*	:	J	Z	j	z
	11	B	VT	ESC	+	;	K	[	k	{
	12	C	FF	FS	,	<	L	\	l	
	13	D	CR	GS	-	=	M	]	m	}
	14	E	SO	RS	.	>	N	^	n	~
	15	F	SI	US	/	?	O	_	o	DEL

In giallo i caratteri di controllo, in azzurro le cifre decimali, in verde le lettere maiuscole dell'alfabeto, in fucsia le lettere minuscole, in grigio tutte le rimanenti lettere.

## Esadecimale codificato ASCII

La necessità di trasmettere dati binari ad 8 bit attraverso un canale di comunicazione seriale operante in codice ASCII a 7 bit, ha indotto Motorola e Intel (aziende leader nel campo dei microprocessori negli anni 70) ad introdurre e ad implementare questo standard.

Il meccanismo è molto semplice: un byte da 8 bit viene scomposto in due cifre a 4 bit MSD (4 bit più significativi) LSD (4 bit meno significativi); le due cifre MSD e LSD vengono codificate con due caratteri ASCII a 7 bit.

Esempio:

```
devo trasmettere a distanza l'informazione a 8 bit  %11000101;
scompongo in due blocchi a 4 bit MSD  %1100  $C      LSD  %0101  $5
codifico in ASCII  MSD:   C  →  $43  %100 0011
codifico in ASCII  LSD:   5  →  $35  %011 0101
```

Analogamente si procede per un dato a 16, 24 o 32 bit.

Esempio:

	Valore Hex	Codifica ASCII								
8	bit \$45	\$34	\$35							
16	bit \$2796	\$32	\$37	\$39	\$36					
24	bit \$F34964	\$46	\$33	\$34	\$39	\$36	\$34			
32	bit \$FFA0A55A	\$46	\$46	\$41	\$30	\$41	\$35	\$35	\$41	

E' opportuno osservare che questo standard rallenta notevolmente la trasmissione in quanto la trasmissione di una informazione rappresentata da un byte (8 bit) comporta l'invio sul canale seriale di un numero di bit quasi doppio (14 bit).

Su questo standard si basa il protocollo S0-S9 ampiamente utilizzato nei programmatori di memoria "stand alone". Di seguito si riporta il protocollo S0-S9 introdotto da Motorola Semiconductors Inc.

M68MM19SB SUPERbug FIRMWARE USER'S GUIDE Copyright 1980 by Motorola Inc.

## S- Records

Un S record è un formato standard usato nella trasmissione e ricezione di programmi e dati. Ci sono dieci possibili record standard S, sei dei quali tuttora in uso, due sono stati definiti ma non sono in uso, e due sono riservati. Essi sono rappresentati nella seguente tabella:

S0	Header record	active
S1	16 bit address Data record	active
S2	24 bit address Data record	active
S3	32 bit address Data record	
S4	reserved	
S5	Transmitted Data Record Count record	active
S6	reserved	
S7	32 bit address End of File/Execution Address record	
S8	24 bit address End of File/Execution Address record	active
S9	16 bit address End of File/Execution Address record	active

Lo standard strutturale dei record S è definito come segue:

FRAME	VALUE		DESCRIPTION	BYTE	CHECKSUM
	\$0D		CR		
	\$0A		LF		
	\$00		NULL		
1	\$53	S	Start of record		
2	\$30-\$39	(0-9)	Record Type		
3,4			Byte Count		*
5-8			Address (for 16 bit)	*	*
[5-10			Address (for 24 bit)	*	*]
[5-12			Address (for 32 bit)	*	*]
:				*	*
:			Data	*	*
:				*	*
N-1,N			Checksum	*	

“S” e il tipo di record sono rappresentati direttamente in ASCII.

Il numero Byte Count (posizione 3,4), l'indirizzo, i dati, il checksum sono rappresentati in esadecimale codificato ASCII; in altri termini un byte di dati occupa due frames (due caratteri) con la cifra più significativa MSD collocata nel primo carattere.

Esempio:

Valore Hex	Codifica ASCII			
\$45	\$34	\$35		
\$2796	\$32	\$37	\$39	\$36
\$A8	\$41	\$38		
\$AB12	\$41	\$42	\$31	\$32

Il Byte Count (posizione 3,4) rappresenta il numero di bytes che seguono nel record 'Sx' compreso il byte di Checksum. Il valore numerico di Byte Count risulta sempre maggiore o uguale a 3 in quanto nel record 'Sx' è sempre presente un campo Address (2 bytes) ed un campo Checksum (1 byte).

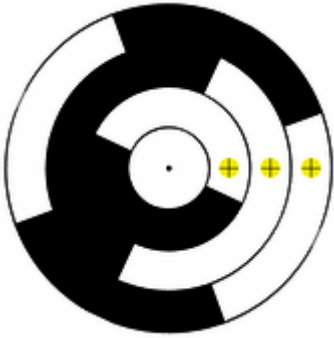
Il checksum è il complemento a 1 della somma a 8 bit di tutti i byte partendo dal Byte Count, comprendendo i byte di address e fino all'ultimo byte di dati.

## Codice Gray

Da Wikipedia, l'enciclopedia libera.

Un **codice Gray**, o **codice di Gray**, è un codice binario a lunghezza fissa. Si possono usare codici di Gray di tutte le lunghezze: il codice di lunghezza  $s$  è costituito da tutte le  $2^s$  sequenze di  $s$  bit e consente di rappresentare tutti gli interi da 0 a  $2^s - 1$ .

Esso differisce dalla notazione posizionale binaria degli interi in quanto prevede che si passi da un intero al successivo modificando un solo bit; questa caratteristica semplifica e rende meno soggette ad errori le operazioni di dispositivi elettronici che devono scorrere informazioni organizzate in sequenze. Evidentemente la codifica di Gray risulta poco sensata per interi da sottoporre ad operazioni come somme o prodotti.



Matrice di commutazione in un encoder rotativo Gray a tre bit

Diversi dispositivi elettronici di acquisizione di posizione, tra cui gli encoder (lineari o rotativi, come - per esempio - i regolatori di volume digitali negli impianti Hi-Fi), codificano il valore digitale della posizione chiudendo o aprendo una serie di contatti elettrici o barriere ottiche. Il problema è che a causa delle tolleranze meccaniche è improbabile che due o più bit di una cifra possano commutare esattamente nello stesso istante. Viene a crearsi una configurazione fisica intermedia in cui è codificato un valore indesiderato, che può generare errore nella successiva elaborazione.

Per evitare queste difficoltà fu progettato e brevettato nel 1953 dal ricercatore Frank Gray dei laboratori Bell il codice che ora porta il suo nome.

Negli encoder che utilizzano questo codice, il passaggio da un valore al successivo o precedente comporta la commutazione di un unico circuito, eliminando ogni possibile errore dovuto a codifiche binarie intermedie.

Va notato che anche nel passaggio dall'ultima alla prima parola del codice cambia solamente un bit.

BINARIO 4 bit	GRAY 4 bit
0000	0000
0001	0001
0010	0011
0011	0010
0100	0110
0101	0111
0110	0101
0111	0100
1000	1100
1001	1101
1010	1111
1011	1110
1100	1010
1101	1011
1110	1001
1111	1000

## Codice Baudot

Da Wikipedia, l'enciclopedia libera.

Il **codice Baudot**, così chiamato in onore del suo ideatore Émile Baudot, è un sistema di codifica per un set di caratteri utilizzato nelle telescriventi prima dei sistemi EBCDIC e ASCII.

### Cenni storici e caratteristiche

Il codice originario fu sviluppato nel 1874 con il nome di International Telegraph Alphabet n. 1 (**ITA1**) e non è più in uso. Veniva codificato utilizzando una tastiera a cinque tasti dove ogni tasto corrispondeva ad un bit di un sistema di codifica a cinque livelli. Un sistema meccanico scandiva la tastiera, dopo di che liberava i tasti per permettere l'inserimento del carattere successivo.

Intorno al 1901 il codice originario fu rivisto da Donald Murray, il quale riorganizzò i caratteri, ne aggiunse di nuovi e introdusse i codici *shift*, che consentivano di cambiare il set di caratteri in uso. Con l'introduzione di questa codifica si cominciarono ad usare tastiere più simili alle attuali. La codifica dei caratteri fu riorganizzata in modo che i caratteri più utilizzati corrispondessero ad un minore numero di commutazioni e quindi minore usura dei meccanismi. Una ulteriore modifica dovuta essenzialmente alla Western union fu la rimozione di alcuni caratteri. Questa ultima versione è nota come **codice Baudot** oppure International Telegraph Alphabet No 2 (**ITA2**). Il codice ITA2 è ancora usato in alcune applicazioni e in particolare tra i radioamatori (RTTY).

Codifica Baudot ITA2			
Codice hex	Codice binario	Serie dei caratteri (LTRS)	Serie dei simboli (FIGS)
0	0	<b>NULL</b> : nessun carattere, nastro senza perforazione	
1	1	E	3
2	10	<b>LF</b> (Line feed): avanzamento di una linea	
3	11	A	-
4	100	<b>SP</b> - Spazio	
5	101	S	'
6	110	I	8
7	111	U	7
8	1000	<b>CR</b> (Carriage Return): ritorno del carrello a inizio riga	
9	1001	D	<b>ENQ</b> iry: richiesta di identificazione
0A	1010	R	4
0B	1011	J	<b>BELL</b> : suona un campanello
0C	1100	N	,
0D	1101	F	!
0E	1110	C	:
0F	1111	K	(
10	10000	T	5
11	10001	Z	+
12	10010	L	)
13	10011	W	2
14	10100	H	£
15	10101	Y	6
16	10110	P	0
17	10111	Q	1
18	11000	O	9
19	11001	B	?
1A	11010	G	&
1B	11011	<b>FIGS</b> : passa alla modalità simboli	
1C	11100	M	.
1D	11101	X	/
1E	11110	V	;
1F	11111	<b>LTRS</b> : passa alla modalità lettere	

NOTA: i codici utilizzati nella tabella si intendono con il bit meno significativo a destra. l'ordine di trasmissione dei bit può però differire tra i diversi produttori.

Nel codice ITA2 i caratteri sono rappresentati da cinque bit, e sono impiegate due serie di simboli, le lettere (LTRS) ed i simboli (FIGS). Il segnale FIGS (11011) indica che i caratteri successivi devono essere interpretati come simboli, fino alla ricezione del segnale LTRS (11111). Il codice ENQuery richiede all'altro terminale di identificarsi (equivale a chiedere "chi sei?"). Nel sistema telex i simboli corrispondenti ai codici 0D, 14 e 1A non sono utilizzati.

Si noti che la codifica binaria dei codici di controllo sono definiti in modo da essere simmetrici, in modo che inserendo un nastro perforato al contrario non causi problemi agli apparati, ed il testo trasmesso possa comunque essere interpretato. I codici FIGS (11011), LTRS (11111) e spazio (00100) sono simmetrici, mentre CR (01000) e LF (00010), poiché sono normalmente inviati in coppia, producono lo stesso risultato (testina a capo sulla nuova linea). Il codice LTRS (11111) è usato anche per cancellare caratteri dal nastro riproforandolo, come l'equivalente DEL nel codice ASCII. La sequenza RYRYRY... è usata nei messaggi di prova poiché rappresentando la sequenza 01010 10101 ... impegna al massimo i meccanismi delle telescriventi. Alcune implementazioni del codice ITA2 negli Stati Uniti, differiscono nell'uso dei codici ENQ, + e dei simboli associati ai caratteri f, g, h rispetto al codice standard rappresentato nella tabella.

## Il Codice Fiscale

<http://webservices.dotnethell.it>



Mostreremo in questo articolo un algoritmo molto comune; è l'algoritmo che ci permette di calcolare il Codice Fiscale.

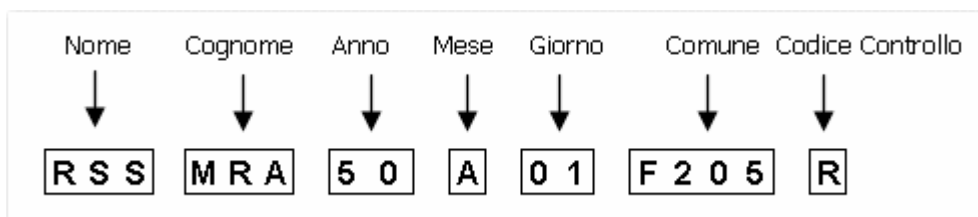
### Cos'è il Codice Fiscale

Il Codice Fiscale è un "codice" (lo dice la parola stessa), che identifica in modo univoco le persone che sono iscritte nei registri dell'anagrafe tributaria cioè i dati che servono poi per fare funzionare tutto il sistema tributario (Fisco, da qui "fiscale"), quindi pagamento tasse, imposte, e così via.

Il codice fiscale per le persone fisiche, viene rilasciato dal Ministero delle Finanze in base ai criteri stabiliti dal D.M. del 23/12/1976. Il procedimento per il calcolo fornisce sempre un codice fiscale PRESUNTO. I Codici Fiscali, infatti, vengono **generati esclusivamente dal Ministero delle Finanze**, che ne assicura anche l'univocità.

### Struttura del Codice Fiscale

E' un codice Alfanumerico (composto da lettere e numeri) di 16 caratteri. I primi 15 sono relativi ai dati personali (nome, cognome, sesso, data di nascita e luogo di nascita) mentre l'ultimo è un carattere di controllo che viene calcolato con delle formule applicate ai precedenti 15 caratteri. Vediamo un codice fiscale:



### Un codice fiscale di esempio

Come vedete è composto dai seguenti blocchi:

- 3 lettere per il cognome
- 3 lettere per il nome
- l'anno di nascita (numero)
- il mese della data di nascita (lettera)
- il giorno della data di nascita (numero)
- il codice del comune di nascita
- il carattere di controllo

### L'algoritmo di calcolo

#### Cognome

Sono necessari, come detto prima, 3 caratteri per rappresentare il cognome. E' possibile che le consonanti siano meno di tre, in questo caso è possibile aggiungere le vocali nell'ordine in cui compaiono nel cognome.



<b>A</b>	GENNAIO	<b>L</b>	LUGLIO
<b>B</b>	FEBBRAIO	<b>M</b>	AGOSTO
<b>C</b>	MARZO	<b>P</b>	SETTEMBRE
<b>D</b>	APRILE	<b>R</b>	OTTOBRE
<b>E</b>	MAGGIO	<b>S</b>	NOVEMBRE
<b>H</b>	GIUGNO	<b>T</b>	DICEMBRE

### Giorno

In questo caso è sufficiente riportare il numero del giorno, con il particolare che per le **donne** questo numero deve essere aumentato di **40!**

### Esempi:

Uomo nato il : **22/10/1980** - Codice Giorno: **22**  
 Donna nata il : **22/10/1980** - Codice Giorno: **62**

### Comune di nascita

E' composto da quattro caratteri alfanumerici. E' il codice meccanografico del comune rilevato dagli archivi catastali.

Esempio :

**Brescia** - Codice Meccanografico del Comune : **B157**

### CODICI MECCANOGRAFICI di alcuni COMUNI della prov. di Brescia

CODICE	COMUNE	CAP
A569	BAGNOLO MELLA	25021
A578	BAGOLINO	25072
A729	BEDIZZOLE	25081
B040	BORGOSATOLLO	25010
B092	BOTTICINO MATTINA	25080
B093	BOTTICINO SERA	25082
B102	BOVEZZO	25073
B149	BRENO	25043
B157	BRESCIA	25100
B394	CALCINATO	25011
B450	CALVISANO	25012
B698	CAPRIANO DEL COLLE	25020
B817	CARPENEDOLO	25013
C055	CASTEGNATO	25045
C293	CASTENEDOLO	25014
C417	CEDEGOLO	25051
C439	CELLATICA	25060
C850	COLLEBEATO	25060
C948	CONCESIO	25062
D251	DARFO BOARIO TERME	25047
D284	DESENZANO DEL GARDA	25015
D391	EDOLO	25048
D634	FLERO	25020
D891	GAMBARA	25020
D917	GARDONE RIVIERA	25083
D918	GARDONE VAL TROMPIA	25063
D924	GARGNANO	25084
D940	GAVARDO	25085
E271	GUSSAGO	25064

E280	IDRO	25074
E333	ISEO	25049
E667	LONATO	25017
E738	LUMEZZANE	25065
E739	LUMEZZANE PIEVE	25066
E741	LUMEZZANE SANT'APOLLONIO	25067
E884	MANERBIO	25025
E928	MARCHENO	25060
F063	MAZZANO	25080
F471	MONTICHIARI	25018
F680	MONTIRONE	25010
F851	NAVE	25075
G149	ORZINUOVI	25034
G150	ORZIVECCHI	25030
G170	OSPITALETTO	25035
G264	PALAZZOLO SULL'OGGIO	25036
G844	PONTE DI LEGNO	25056
H230	REMEDELLO	25010
H232	REMEDELLO SOTTO	25010
H256	REZZATO	25086
H525	RONCADELLE	25030
H598	ROVATO	25038
H717	SALO'	25087
I433	SAREZZO	25068
L495	URAGO MELLA	25128
L777	VEROLANUOVA	25028
L778	VEROLAVECCHIA	25029
L812	VESTONE	25078
M104	VOBARNO	25079

### **CODICI MECCANOGRAFICI di alcuni STATI ESTERI**

CODICE	NAZIONE
Z100	ALBANIA
Z102	AUSTRIA
Z103	BELGIO
Z104	BULGARIA
Z105	CECOSLOVACCHIA
Z107	DANIMARCA
Z109	FINLANDIA
Z110	FRANCIA
Z112	GERMANIA
Z114	GRAN BRETAGNA
Z115	GRECIA
Z116	IRLANDA
Z117	ISLANDA
Z118	JUGOSLAVIA (SERBIA-MONTENEGRO)
Z120	LUSSEMBURGO
Z125	NORVEGIA
Z126	OLANDA
Z127	POLONIA
Z128	PORTOGALLO
Z129	ROMANIA

Z131	SPAGNA
Z132	SVEZIA
Z133	SVIZZERA
Z134	UNGHERIA
Z135	UNIONE SOVIETICA
Z138	UCRAINA
Z139	BIELORUSSIA
Z140	MOLDAVIA
Z149	CROAZIA
Z150	SLOVENIA
Z153	BOSNIA-ERZEGOVINA
Z156	REPUBBLICA CECA
Z200	AFGHANISTAN
Z203	ARABIA SAUDITA
Z226	ISRAELE
Z236	PAKISTAN
Z301	ALGERIA
Z311	CONGO
Z312	ZAIRE
Z313	COSTA D'AVORIO
Z315	ETIOPIA
Z318	GHANA
Z322	KENIA
Z326	LIBIA
Z330	MAROCCO
Z335	NIGERIA
Z336	EGITTO
Z345	SOMALIA
Z347	REPUBBLICA SUDAFRICANA
Z348	SUDAN
Z352	TUNISIA
Z404	STATI UNITI D'AMERICA

### Il codice di controllo

Rimane l'ultimo carattere, è il codice di controllo, forse la procedura più noiosa è proprio il calcolo di quest'ultimo carattere e quindi gli dedichiamo un intero paragrafo. Partiamo subito mostrando questa tabella:

### Caratteri posizione pari

0=0	1=1	2=2	3=3	4=4	5=5	6=6	7=7	8=8	9=9	A=0	B=1	C=2
D=3	E=4	F=5	G=6	H=7	I=8	J=9	K=10	L=11	M=12	N=13	O=14	P=15
Q=16	R=17	S=18	T=19	U=20	V=21	W=22	X=23	Y=24	Z=25			

### Caratteri posizione dispari

0=1	1=0	2=5	3=7	4=9	5=13	6=15	7=17	8=19	9=21	A=1	B=0	C=5
D=7	E=9	F=13	G=15	H=17	I=19	J=21	K=2	L=4	M=18	N=20	O=11	P=3
Q=6	R=8	S=12	T=14	U=16	V=10	W=22	X=25	Y=24	Z=23			

### Carattere di controllo

0=A	1=B	2=C	3=D	4=E	5=F	6=G	7=H	8=I	9=J	10=K	11=L	12=M
13=N	14=O	15=P	16=Q	17=R	18=S	19=T	20=U	21=V	22=W	23=X	24=Y	25=Z

### Tabelle per le conversioni dei caratteri

Si comincia con il prendere i caratteri del codice fiscale fin qui calcolato che sono 15, si prendono quelli in posizione pari e si convertono con i numeri corrispondenti della prima tabella. Tutti questi numeri vengono sommati.

Allo stesso modo con i caratteri dispari che devono essere convertiti però utilizzando la seconda tabella e vengono tutti sommati.

I valori ottenuti vengono a loro volta sommati e il totale viene diviso per 26. Il resto della divisione deve essere convertito usando l'ultima tabella. Il carattere corrispondente è il codice di controllo!

### IL CODICE MORSE (Alfabeto MORSE)

Il codice Morse fu inventato da Samuel Finley Breese Morse nel 1836. È formato da combinazioni di segnali lunghi e brevi ("linea" e "punto") con cui si rappresentano tutti numeri e le lettere dell'alfabeto. Punti e linee si possono riprodurre vocalmente usando la sillaba TI per i punti e la sillaba TA per le linee.

Usato inizialmente per la telegrafia a filo, il codice Morse è stato successivamente adottato per la radiotelegrafia. Utilizzato normalmente fino a pochi anni fa per le comunicazioni, oggi "sopravvive" solo in parte del campo amatoriale.

L'abbreviazione in codice Morse più conosciuta è senz'altro l'SOS ( TI-TI-TI TA-TA-TA TI-TI-TI ).

**NOTA:** L'uso di punti e linee è la rappresentazione classica. Quando si leggono è forse più utile pensarli in forma sonora e non leggere la lettera A come "punto linea" ma come TI-TA.

alfabeto	codice morse
A	.-
B	-...
C	-.-.
D	-..
E	.
F	..-.
G	--.
H	....
I	..
J	.-.-
K	-.-
L	.-..
M	--
N	-.
O	---
P	.-.-
Q	--.-
R	.-.
S	...
T	-
U	..-
V	...-
W	.-.-
X	-..-
Y	-.--
Z	--..

numeri	codice morse
1	.-....
2	..-.-
3	...--
4	....-
5	.....
6	-....
7	--...
8	---..
9	----.
0	-----

segni	codice morse
punto (.)	.-.-.-
virgola (,)	-.-.-
punto interr. (?)	..-.-.
linetta (-)	-...-
barra (/)	-.-.
invito a trasmettere	-.-
errore	.....
attendere	.-...
fine messaggio	.-.-.
fine trasmissione	...-.-

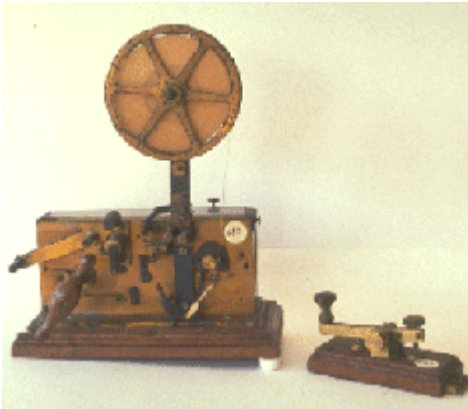
La durata di una linea è pari a 3 punti

Lo spazio tra parti della stessa lettera è pari ad 1 punto

Lo spazio tra due lettere della stessa parola è pari a 3 punti

Lo spazio fra 2 parole è pari a 5 punti

Il codice o alfabeto Morse è stato ( e lo è ancora a livello radioamatoriale) il sistema di comunicazione che ha segnato la storia delle comunicazioni a distanza, il telegrafo è stato utilizzato fino a pochi anni fa dalle maggiori Istituzioni dello Stato, enti pubblici e privati come : Forze Armate, Polizia di Stato, Prefetture, Ministeri, le navi mercantili e il traffico commerciale, quelle passeggeri, agenzie di stampa, stazioni costiere, stazioni meteorologiche , ambasciate, ferrovie, uffici postali....., salvando molte vite umane nei casi di naufragio, per citarne alcuni:



- 1909 - Il naufragio del transatlantico inglese Republic che entrò in collisione con il piroscafo italiano Florida. Il radiotelegrafista Binns a bordo del Republic fece arrivare quattro navi in soccorso che trasbordarono i passeggeri a bordo del Florida. In quell'occasione il radiotelegrafista Binns rimase al suo posto chiedendo aiuto per 14 ore, salvandosi per ultimo su ordine del proprio comandante. Grazie alla radiotelegrafia si salvarono circa 2000 naufraghi. L'avvenimento ebbe enorme risonanza e vennero promulgate speciali leggi per rendere obbligatorio l'impianto di stazioni radio a bordo delle navi.
- 1912 - Il tragico affondamento del transatlantico Titanic che, nella notte del 14 Aprile 1912 con a bordo 2358 persone, urtò contro un iceberg a 270 miglia da Capo Race (Terranova). Circa 705 persone furono salvate dalle navi Carpathia e Olympic chiamate in aiuto col telegrafo Marconi. Il secondo telegrafista del Titanic Mr. Bride, quando fu salvato, prese posto nella cabina radio del Carpathia e rimase a radiotelegrafare per cinque giorni i dispacci dei sopravvissuti.
- 1928 - Il dirigibile Italia e la vicenda della tenda rossa, dove il suo grande

protagonista il radiotelegrafista Biagi, dopo aver riparato la stazione radio ad onde corte rinvenuta nei pressi della navicella, riuscì dopo circa 10 giorni di disperati tentativi a farsi sentire e successivamente permise l'individuazione della tenda rossa.

Grazie alle comunicazioni radiotelegrafiche, inoltre, sono state salvate e curate moltissime persone, da citare i bollettini medici effettuati dai radiotelegrafisti a bordo delle navi, che aggiornavano lo stato di salute dei malati con la conseguente richiesta dei farmaci e interventi chirurgici nei casi più gravi.

Fonte : <http://web.tiscali.it/dac>.

## Alfabeto fonetico NATO

Da Wikipedia, l'enciclopedia libera.

L'**alfabeto fonetico radiotelegrafico**, chiamato spesso anche **alfabeto fonetico NATO**, venne sviluppato negli anni cinquanta dall'Organizzazione Internazionale dell'Aviazione Civile (ICAO) per essere comprensibile (e pronunciabile) per tutti i piloti e gli operatori dell'aviazione civile. Rimpiazzò altri alfabeti fonetici, ad esempio quello dell'esercito statunitense ("able baker") e diverse versioni dell'alfabeto fonetico della RAF.

<b>A</b>	ALPHA	<b>N</b>	NOVEMBER
<b>B</b>	BRAVO	<b>O</b>	OSCAR
<b>C</b>	CHARLIE	<b>P</b>	PAPA
<b>D</b>	DELTA	<b>Q</b>	QUEBEC
<b>E</b>	ECO	<b>R</b>	ROMEO
<b>F</b>	FOX TROT	<b>S</b>	SIERRA
<b>G</b>	GOLF	<b>T</b>	TANGO
<b>H</b>	HOTEL	<b>U</b>	UNIFORM
<b>I</b>	INDIA	<b>V</b>	VICTOR
<b>J</b>	JULIETTE	<b>W</b>	WHISKY
<b>K</b>	KILO	<b>X</b>	X-RAY
<b>L</b>	LIMA	<b>Y</b>	YANKEE
<b>M</b>	MIKE	<b>Z</b>	ZULU

Viene a volte erroneamente indicato come *Alfabeto fonetico internazionale*, che è in realtà il nome ufficiale di un alfabeto usato in linguistica, creato alla fine del XIX secolo e costituito da segni, anche designati appositamente.

Venne adottato, con piccole modifiche dalla NATO. L'alfabeto fonetico NATO è ampiamente utilizzato negli affari e nelle telecomunicazioni, in Europa e Nord America. È stato adottato dall'Unione Internazionale telecomunicazioni (ITU). Anche se è composto da parole inglesi, le lettere codificate possono essere riconosciute facilmente da persone che parlano altre lingue.

L'alfabeto viene usato per scandire parti di un messaggio o di una segnalazione che sono critiche o difficili da riconoscere durante una comunicazione vocale. Ad esempio il messaggio "procedere alle coordinate DH98" può essere trasmesso come "procedere alle coordinate Delta-Hotel-Niner-Eight" e un C-130 che vola dritto di fronte a voi può essere descritto come "Charlie One Tree Zero a ore dodici".

La scrittura di alcune lettere può variare in alcune versioni pubblicate dell'alfabeto. In particolare *Alpha* può essere scritto come *Alfa* e *Juliet* come *Juliett*. Poiché questo alfabeto è pensato per essere parlato, questo non è un problema, in quanto la pronuncia rimane invariata.

### Codice EBCDIC

Il Codice EBCDIC (**E**xtended **B**CD **I**nterchange **C**ode) è stato introdotto dalla IBM. Lo schema di codifica EBCDIC (Extended Binary Coded Decimal Interchange Code), messo a punto dall'IBM per i propri sistemi, è impostato sull'uso di codici a 8 bit per la rappresentazione di 256 caratteri, invece dei 7 bit per 128 caratteri del codice ASCII standard. Benché non molto usato sui microcomputer, l'EBCDIC è ben noto e riconosciuto a livello internazionale, soprattutto come codice IBM per mainframe e minicomputer.

L'incompatibilità tra l'ASCII e l'EBCDIC comporta considerevoli problemi nello sviluppo di protocolli di collegamento tra sistemi che usano i due diversi codici.

*Tratto da "Codifica dei caratteri," Microsoft® Encarta® Enciclopedia Online 2007*

### Codice dei Colori RGB (R red – G green – B blue)

<http://www.webalice.it/lagreca2003/htmlpoint/rgb.htm>

**RGB** sta per "Red (Rosso), Green (Verde), Blue (Blu)". Si tratta del modello di definizione e visualizzazione dei colori standard per il mondo informatico. Dato che il computer forma i colori di visualizzazione utilizzando **pixels** dei tre colori RGB, possiamo quindi ottenere ciascun colore come sommatoria di parti di Rosso, Verde e Blu (colori primari). Il sistema funziona quindi in modalità additiva: aggiungendo i colori l'uno all'altro si ottengono milioni di altri colori diversi. Il bianco è la somma di parti identiche di rosso, verde e blu.



L'assenza di tutti i tre colori primari genera il nero. Quando due colori primari vengono sommati, un colore secondario, più chiaro dei suoi componenti viene creato. Ad esempio:



Blue + Verde = Cyan



Dato che il computer forma i colori di visualizzazione utilizzando pixels dei tre colori RGB, possiamo quindi definire ciascun colore come sommatoria di parti di Rosso, Verde e Blu. La scala che utilizziamo per percepire i colori riprodotti da un computer va da 0 a 255. Quindi ci sono ben 256 gradazioni per ciascun colore primario.

I valori RGB, vengono rappresentati da tre numeri esadecimali. La rappresentazione RGB utilizza 24 bit con possibilità di rappresentare  $2^{24} = 2^4 \cdot 2^{10} \cdot 2^{10} = 16$  milioni di colori.

**Tabella dei principali colori RGB**

COLORE	R	G	B	
	red	green	blue	
NERO	0	0	0	
BLU	0	0	FF	
CIANO	0	FF	FF	
BLU SCURO	0	0	8B	
GRIGIO SCURO	A9	A9	A9	
ARANCIO SCURO	FF	8C	0	
VIOLA SCURO	94	0	D3	
ORO	FF	D7	0	
GRIGIO	80	80	80	
VERDE SCURO	0	80	0	
VERDE	0	FF	0	
MAGENTA	FF	0	FF	
MARRONE	80	0	0	
BLU MARE	0	0	80	
ARANCIO	FF	A5	0	
ROSSO	FF	0	0	
ARGENTO	C0	C0	C0	
AZZURRO	87	CE	EB	
BIANCO NEVE	FF	FA	FA	
VIOLETTO	EE	82	EE	
BIANCO	FF	FF	FF	
GIALLO	FF	FF	0	

### **Codice ISBN (International Standard Book Number)**

Nel 1965 W.H. Smith, un grande rivenditore inglese di libri, annunciò di voler passare alla gestione computerizzata entro due anni; necessitando, però, di un adeguato sistema di catalogazione, commissionò lo studio a dei consulenti, e di accordo, assieme ad altri venditori di libri, diede vita allo standard ISBN. L'ISO convocò una commissione per discutere della possibilità di ampliare questo metodo all'uso internazionale. Il primo incontro si tenne a Londra nel 1968 con rappresentanti di Danimarca, Francia, Germania, Irlanda, Paesi Bassi, Norvegia, Regno Unito, USA e osservatori dell'UNESCO. Altri paesi contribuirono con suggerimenti scritti e dichiarazioni d'interesse. La proposta finale venne fatta nel 1969, e l'ISBN divenne standard ISO nel

1970, come ISO 2108. Dal 1 gennaio 2007 si sono aggiunte 3 cifre davanti al ISBN "vecchio", portando così le cifre che compongono l'ISBN da 10 a 13.

## Formato

Il codice ISBN di un libro è formato da una stringa di 10 cifre, divise in 4 settori

1. **Gruppo** - identificativo del paese, dell'area o del linguaggio, composto da 1 a 5 cifre
2. **Editore** - identificativo della casa editrice, composto da 1 a 7 cifre
3. **Titolo** - identificativo del libro, composto da 1 a 6 cifre
4. **Controllo** - rappresenta un codice di controllo necessario a verificare l'esattezza dell'algoritmo, è composta da un solo numero variabile da 0 a 10 (quest'ultimo indicato però dalla cifra romana "X") .



La lunghezza delle prime due stringhe è inversamente proporzionale al numero di pubblicazioni, rispettivamente nel paese e dell'editore, mentre la terza è più grande tanti più libri si pubblicano. Ovviamente, la somma delle prime tre parti deve essere di 9 cifre. Ad esempio, il primo campo è 0 per la lingua inglese, 3 per gli editori tedeschi, 982 per l'area Sud Pacifico, 88 è l'italiano. La cifra di controllo è basata sulle proprietà dei numeri primi. *Esempio:* dato un codice 88-515-2159 si moltiplica ogni cifra per un peso in base alla posizione della cifra stessa

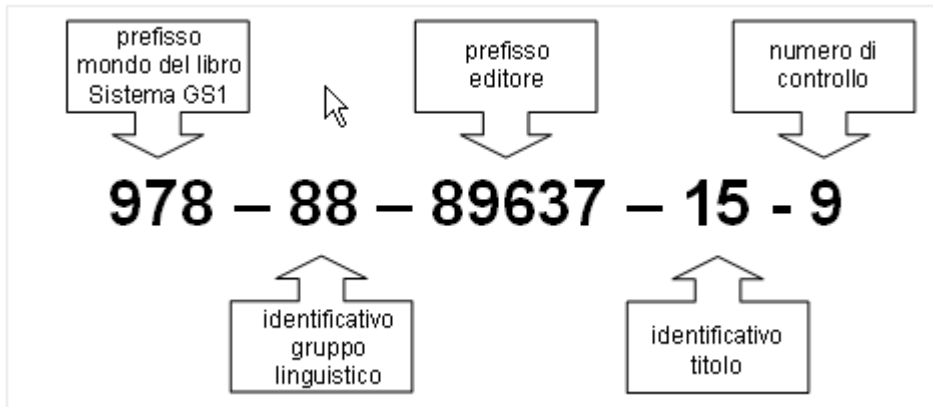
## ISBN-10 e ISBN-13

Da gennaio 2007, visto che in alcuni settori ci si avvia verso una carenza di codici disponibili, la codifica ISBN è stata modificata per presentare 13 cifre. I codici esistenti sono stati conservati anteposando al codice esistente il prefisso "978", mentre sono stati aggiunti nuovi codici con il prefisso "979", in modo da raddoppiare il numero totale di codici disponibili. Ora il codice di controllo è calcolato in modo differente. Questo nuovo codice, chiamato ISBN-13, corrisponde al codice *Bookland* EAN (European Article Number), già in uso presso alcuni editori fin dagli anni ottanta. L'EAN, nato come codice europeo, è stato ora adottato internazionalmente e si applica a tutti i media, non solo ai libri. Il codice EAN per altri prodotti non librari identifica lo stato di provenienza del prodotto con le prime tre cifre; per aver spazio sufficiente per tutti i libri e per poter convertire facilmente gli ISBN in EAN si è preferito per i libri adottare un altro metodo, creando uno stato fittizio, il *Bookland*, a cui è stato assegnato il prefisso 978, indipendentemente dallo stato di origine.

## Che cosa è il codice ISBN

L'ISBN - International Standard Book Number - è un numero che identifica a livello internazionale in modo univoco e duraturo un titolo o una edizione di un titolo di un determinato editore. Oltre a identificare il libro, si attribuisce a tutti quei prodotti creati per essere utilizzati come libro. Possono richiederlo: le case editrici e tutti quegli enti/fondazioni pubblici o privati che hanno una produzione editoriale.

L'ISBN - a partire dal 1° gennaio 2007 - è formato da un codice di 13 cifre, suddivise in 5 parti dai trattini di divisione:



- la prima parte è il prefisso di tre cifre che nella rappresentazione a barre del codice ISBN nel sistema GS1 identifica il mondo del libro (978 e in futuro anche 979)
- la seconda parte è l'identificativo dell'area linguistica. Identifica il Paese o l'area linguistica dell'editore
- la terza parte è il prefisso editore: identifica l'editore/marchio editoriale
- la quarta parte è il numero identificativo del titolo
- la quinta parte è il numero di controllo, una garanzia contro possibili errori

A partire da un codice ISBN si può generare il [codice a barre](#) a esso collegato. Viene attribuito come il codice ISBN e utilizzato per la lettura ottica.

Fonte : <http://www.isbn.it/>

## Codice a barre

Il **codice a barre** può essere definito come una simbologia o un alfabeto per la codifica di informazioni in un formato tale da poter essere acquisito automaticamente da opportuni lettori. Il computer è stato protagonista dello sviluppo tecnologico degli ultimi decenni, durante i quali si è venuto ad affermare in ogni ramo di attività grazie alla sua capacità di elaborare enormi quantità di dati in tempi brevissimi, di raccogliere ed archiviare l'informazione in spazi ridotti, di controllare processi industriali, ecc..



Tuttavia presenta un lato debole che da sempre ne ha limitato l'efficienza: l'interfaccia con il mondo esterno, e particolarmente l'acquisizione delle informazioni. Superato completamente il supporto fisico della scheda perforata, l'acquisizione dei dati avviene solitamente tramite operatore umano, mediante l'uso di terminali. Da ciò derivano alcuni problemi, quali lentezza e alta probabilità di errore. Esistono poi molti processi operativi (ad esempio linee di trasporto e sistemi di smistamento di linee automatiche di produzione) nei quali la velocità e la frequenza degli oggetti da identificare sono tali da rendere impossibile ad un eventuale operatore captare i dati e trasmetterli al calcolatore. Per questo si è reso necessario un sistema di codifica dei dati adatto a sistemi automatici di rilevamento, per sfruttare la piena potenzialità del computer. Fin dall'inizio degli anni '70 si sono venute sviluppando diverse tecnologie, tre delle quali sono risultate maggiormente significative: la tecnologia OCR, la tecnologia magnetica e la tecnologia del **codice a barre**.

La tecnologia OCR (Optical Character Recognition) consiste nel codificare i caratteri in modo tale che possano essere letti anche direttamente dall'operatore. La stampa è molto complessa e con tolleranze molto rigide risolta con stampanti dedicate, ma soprattutto i metodi di riconoscimento sono complessi e piuttosto lenti. E' una tecnologia complessivamente costosa e non altamente efficiente, per cui non si è mai affermata in modo definitivo.

La tecnologia magnetica presenta il vantaggio che le informazioni contenute nel supporto fisico possono essere agevolmente modificate, ma allo stesso tempo è costosa ed il rischio di perdere o alterare le informazioni è tuttora elevato.

Il **codice a barre (bar code)** si è imposto nel tempo come la tecnologia vincente. La codifica si basa su un concetto binario, quindi è già di per se vicina al linguaggio dei calcolatori. La diversa logica di codifica, dettata da diverse esigenze applicative, ha portato a diversi tipi di codici a barre. Alcuni di essi sono ampiamente diffusi, altri vengono usati solo in speciali settori, altri ancora solo in determinati paesi.

### Tipi di codici a barre

Tra i tipi più diffusi in Italia, senz'altro troviamo il codice EAN (European Article Number) che viene utilizzato nella grande distribuzione, seguito dal Farmacode o codice 32 (una rielaborazione matematica del Codice 39), adottato per l'identificazione dei farmaci e delle specialità vendibili al banco nelle farmacie, nell'ambito industriale hanno trovato grande diffusione il codice 128, il codice 39 e il 2/5 interleaved. La maggior parte dei codici ha un codice di controllo, (*checksum digit*), che l'unità di lettura è in grado di ricalcolare e verificare per assicurare la corretta lettura e l'integrità dei dati.

Le specifiche di come viene costruito un codice a barre sono disponibili sul Manuale del Codice a Barre. <http://www.codiceabarre.it/>

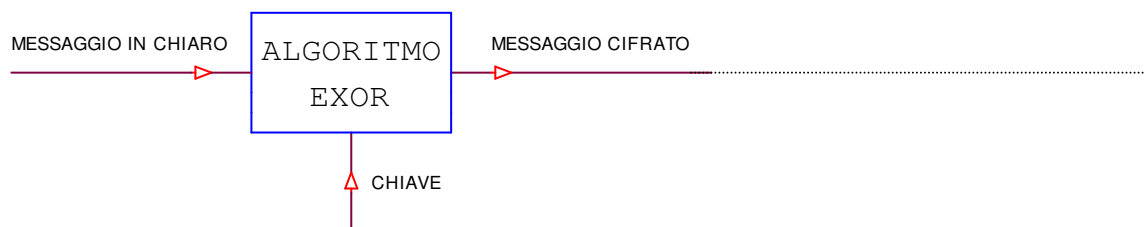
### Elementi di cifratura di un messaggio ASCII

*Crittografia vuol dire scrittura segreta. L'arte di scrivere messaggi segreti che possano essere letti e compresi solo dal destinatario risale alla più remota antichità, se già la Bibbia parla di un codice segreto per scrivere il nome di Babele (il codice Atbash).*

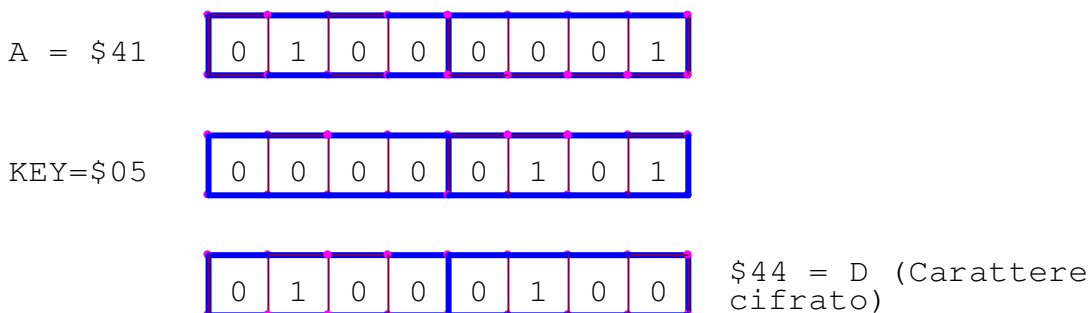
*Per secoli quest'arte è stata appannaggio quasi esclusivo dei militari e dei diplomatici, e i metodi crittografici erano specifici per l'invio di messaggi materiali affidati a corrieri.*

*Nel XX secolo però prima l'invenzione della radio, poi quella del computer hanno cambiato in modo radicale lo scenario. La necessità di comunicare informazioni in modo riservato si è ampliata notevolmente. Oggi l'utente del Bancomat, di una pay-tv e chi effettua acquisti su Internet con la carta di credito fa uso, spesso senza rendersene conto, di tecniche crittografiche.*

Analizziamo come funziona un semplice metodo di cifratura di un carattere ASCII basato sulla operazione EXOR. Questo tipo di cifratura si dice a chiave segreta e simmetrica in quanto la chiave di criptazione del messaggio è identica a quella di decriptazione.



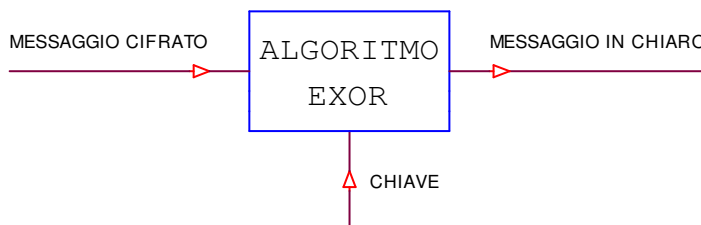
Lo schema sopra riportato propone lo scenario in cui opera chi effettua la cifratura del messaggio. Un determinato messaggio deve essere inviato ad un destinatario ma nello stesso tempo si vuole renderlo incomprensibile a terze persone che non sono i destinatari della comunicazione.



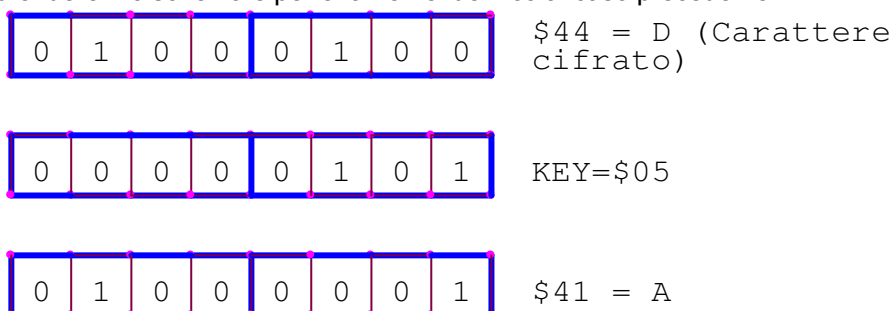
Un messaggio è un insieme di caratteri per cui potremmo pensare di cifrare uno dopo l'altro i caratteri. A titolo di esempio supponiamo, di avere il carattere "A" maiuscolo da codificare e poniamo di utilizzare una chiave molto semplice (numero 5: semplice perché ha solamente due bit a 1). Eseguendo la operazione

EXOR fra il carattere "A" e la chiave 5 si ottiene il carattere D. Procedendo in modo analogo si può facilmente verificare che il carattere "B" diventa G; C diventa F; 0 diventa 5 ecc.

Si osservi che l'operazione di cifratura lascia inalterati i bit presenti nel carattere iniziale in corrispondenza agli 0 presenti nella chiave mentre modifica (esegue una operazione NOT) sui bit che si trovano in corrispondenza degli 1 presenti nella chiave.



Lo schema sopra riportato propone lo scenario in cui opera chi riceve il messaggio cifrato e deve "decifrarlo" per poterlo comprendere. Lo schema è perfettamente identico al caso precedente.



Eseguendo l'operazione EXOR fra il carattere cifrato "D" e la chiave 5 si riottiene il carattere "A"; il messaggio cifrato ritorna "in chiaro".

Si osservi che l'operazione di decifrazione lascia inalterati i bit presenti nel carattere iniziale in corrispondenza agli 0 presenti nella chiave (bit che non erano stati modificati nel processo di cifratura) mentre modifica (esegue una operazione NOT) sui bit che si trovano in corrispondenza degli 1 presenti nella chiave (intervenendo così solo sui bit modificati dal processo di cifratura).

Esempio:

Il messaggio "IPSIA MORETTO BRESCIA" cifrato con chiave segreta 5 diviene "LUVLD%HJW@QQJ%GW@VFLD"

Approfondimenti:

<http://www.liceofoscarini.it/studenti/crittografia/index.html>

### Controllo e recupero errori

Quando un messaggio digitale codificato (testo, immagine, o altro) viene trasmesso a distanza, esso subisce delle inevitabili trasformazioni o modifiche dovute a rumori o disturbi che si sovrappongono al segnale nel tragitto compreso fra trasmettitore e ricevitore. Essi tendono a modificarne il contenuto informativo. In ricezione perciò il messaggio ricevuto può risultare diverso rispetto a quello trasmesso. Il cambiamento in un messaggio di uno o più bit è causa di errori. Si parla di rumore di tipo *burst* quando gli effetti si manifestano su bit fra di loro vicini.

Teoricamente gli errori si possono:

- prevenire (*forward error control*) inserendo bit ridondanti che facilitino il recupero del messaggio anche in presenza di errori (p.es. dal contesto)

- curare (*feedback error control*) prevedendo un protocollo che causi la ri-trasmissione delle parti di messaggio errate

Al momento la norma e' di curare gli errori, vi sono però vantaggi e svantaggi ad entrambi i metodi.

Gli errori devono venire:

- scoperti
- corretti

e ciò deve avvenire al livello più basso possibile del modello OSI (livello fisico). Quindi errori dovuti a fenomeni fisici devono essere recuperati a livelli fisici.

## Controllo di Parità

Questo metodo e' usato soprattutto con la trasmissione seriale asincrona. Si usi il codice ASCII puro a 7 bit per carattere. In tal caso l'ottavo bit non serve alla codifica di un carattere e può venire usato per il controllo di parità.

Si concorda a priori tra dispositivo trasmittente e ricevente se la parità debba essere **pari** o **dispari**, ovvero se la somma dei bit settati a 1 in un byte trasmesso debba sempre essere un numero pari o dispari.

Il trasmittente setta a 1 o 0 l'ottavo bit in modo da forzare la parità concordata.

Il ricevente controlla che la parità sia osservata in ogni byte. Se un byte e' di parità sbagliata allora contiene almeno un bit errato.

Il problema e' che il controllo di parità scopre 1 o 3 bit errati ma non 2 o 4 o un numero pari qualsiasi. Il controllo di parità e' comunque estremamente semplice da implementare in hardware. Al momento anche su linee seriali asincrone non viene quasi più usato.

## Check di ridondanza longitudinale LRC (Longitudinal Redundancy Check)

E' anche detto controllo di parità longitudinale LRC, mentre il bit di parità è anche detto controllo di parità trasversale. Consiste nell'inserimento di un byte in coda ad un blocco-dati di  $n$  byte.

Il Byte LRC viene calcolato eseguendo l'operazione di OR ESCLUSIVO su tutti i byte costituenti il blocco-dati da trasmettere.

```
Function LRC(str : string) : byte;
  Var LRC : byte;
      C : char;
  Begin
    LRC := 0;
    For k := 1 to Length(str) do
      Begin
        C := str[k]; // recupero uno dopo l'altro i caratteri contenuti in str
        LRC := LRC XOR Ord(c);
      End;
    Result := LRC;
  End;
```

Per quanto più efficace del solo controllo di parità trasversale, questo metodo non si accorge se quattro bit "agli angoli di un quadrato" sono errati.

## Checksum

Da Wikipedia, l'enciclopedia libera.

La parola **checksum**, tradotta letteralmente significa *somma di controllo*. È una sequenza di bit che viene utilizzata per verificare l'integrità di un blocco di dati o di un messaggio che può subire alterazioni durante la trasmissione. Il tipo più semplice di *checksum* è un byte ricavato sommando (somma modulo 256) ogni byte

contenuto nel blocco-dati. Per controllare l'integrità del messaggio sarà sufficiente, in ricezione, effettuare la stessa operazione di somma e confrontarla con il *checksum* memorizzato. Se i due valori coincidono, i dati possono essere considerati integri.

Questa semplice forma di *checksum* non è molto accurata in quanto non permette di rilevare certe tipologie di errore come:

- il riordinamento dei byte del messaggio
- l'inserimento di byte con valore 0
- la presenza di diversi errori che sommati tra loro danno 0

In questi casi, quando cioè ci sono due o più serie di bit che hanno lo stesso *checksum*, si parla di *collisioni*. Ovviamente, minore è la probabilità di collisioni, migliore è la qualità dell'algoritmo di controllo e quindi la sicurezza nella verifica dell'integrità.

Con il tempo sono nati diversi metodi di controllo più sofisticati, come il checksum di Fletcher, l'Adler-32 il Cyclic redundancy check (CRC), che valutano non solo la somma dei bit, ma anche la loro posizione. Il prezzo della maggiore affidabilità viene pagato nelle risorse necessarie al calcolo del *checksum*.

Questi metodi sono utili per la verifica di corruzioni accidentali (errori di trasferimento, o di memorizzazione, perdita di dati), ma non sono sicuri contro gli attacchi di malintenzionati, in quanto le loro strutture matematiche non sono particolarmente complesse da aggirare. Per questo tipo di attacchi vengono utilizzati algoritmi di hash crittografati, come l'MD5, lo [SHA-1](#) (in cui però sono state trovate o sembra possibile trovare *collisioni*), o lo [SHA-256](#), per il momento incorruttibile.

I checksum vengono usati spesso su internet per poter garantire che i dati scaricati siano corretti e per garantirne l'autenticità. Per esempio nel download di software, il distributore del programma pubblica il *checksum* (in genere MD5 o SHA-1), che nello specifico viene chiamato [digest](#), che viene controllato dall'utente per verificare l'integrità dei dati.

-----

<DA COMPLETARE>