CÓDIGOS BINARIOS

La codificación puede definirse como el proceso de representación unívoca de letras o números mediante un grupo especial de símbolos (código), de tal manera que a cada una de las primeras se le asigna una combinación de éstos y viceversa. Un ejemplo clásico es el código Morse, que usa puntos y rayas para representar letras y números.

Hasta ahora hemos visto sistemas de representación numérica, en particular el sistema binario con el que cualquier número decimal puede representarse mediante un conjunto de ceros y unos. Obviamente, esto también encuadra en el concepto de codificación, ya que no es otra cosa que un código de representación de cantidades, y se la denomina codificación binaria directa.

Sin embargo, existen situaciones en las que el código binario natural (para distinguirlo de los otros, ya que en sentido estricto todos son binarios) no es el más útil. Así surgen otros códigos de uso específico en determinadas aplicaciones.

Código	*Variante*	*Ponderado*	*Autocomp.*	*Cíclico*	*Detecta/corrige errores*
BINARIO		SI			
BCD	NATURAL	SI			
	EXCESO 3		SI		
	AIKEN	SI	SI		
CONTINUOS	GRAY			SI	
	JONHSON			SI	
DETECTORES	PARIDAD PAR O IMPAR				1 bit
	2 ENTRE 5				1 bit
	BIQUINARIO	SI			1 bit
	QUI BIQUINARIO	SI			1 bit
	7 EXCESO 3				1 bit
CORRECTORES	HAMMING				1 bit

CÓDIGOS BCD

La información procesada por cualquier sistema digital ha de convertirse finalmente al sistema decimal para que pueda ser interpretada con mayor facilidad. Esta es la principal razón de la existencia de los códigos decimales codificados en binario (BCD: **Binary Code Decimal**), que como se verá, se convierten muy fácilmente al sistema decimal.

Los códigos BCD pueden clasificarse en tres clases diferentes, con propiedades particulares que los hacen útiles para determinadas aplicaciones.

BCD Natural

Es uno de los códigos más utilizados en cualquier elemento que requiera como entrada o salida a información decimal. Es un código ponderado (a cada posición se le asigna un peso) con los mismos pesos que el binario natural (8 4 2 1).

Si cada dígito de un número decimal se representa con su equivalente binario se halla el código BCD de ese número decimal. Como un dígito decimal requiere de al menos 4 bits para poder ser representado (0 a 9), el BCD necesita también de 4 bits (con 3 bits se alcanzan a representar 8 combinaciones solamente) para cubrir los diez números decimales. Debe quedar claro que al ser los pesos de cada bit del BCD los mismos que los del binario, su única diferencia está en que aquél "binariza" cada decimal individualmente, mientras que el binario natural toma un decimal completo.

Ejemplo: 214_{10} = 0010 0001 0100 (BCD)

Al usar sólo 10 de las 16 combinaciones posibles con 4 bits, existen 6 combinaciones que no son válidas en BCD y generan error si se presentan:

10=1010 11=1011 12=1100 13=1101 14=1110 15=1111

Obviamente, el pasaje de BCD a decimal se hace en forma exactamente inversa a la explicada: se toma el número BCD y se agrupan sus bits de derecha a izquierda en conjuntos de a 4 (similar a la conversión hexadecimal - decimal) y se obtiene el decimal de cada cuarteto.

Ejemplo: 100001100101 = 1000 0110 0101 = 865

La VENTAJA PRINCIPAL del BCD es la facilidad de conversión en decimal y desde decimal, y como nuestro mundo está regido por este último sistema, esto es importante. En binario, el hardware para conversiones con el decimal crece exponencialmente con el número de bits. En BCD, cada dígito se asocia exactamente con 4 bits; para 2N dígitos se necesita sólo el doble de hardware que para N.

Desde ya, existen contadores integrados que permiten realizar un conteo en BCD e, incluso, permiten al usuario seleccionar si desean realizar el conteo en binario o BCD (CD 4029, por ejemplo). Como consecuencia, también existen decodificadores que realizan el pasaje de BCD a 7 segmentos (displays) para lograr la visualización del número (CD 4511).

La DESVENTAJA más importante del BCD radica en su relativa ineficiencia ya que, como se dijo, sólo utiliza 10 de las 16 combinaciones posibles (62,5%). Esto hace que el número de bits necesarios para representar un número decimal en BCD sea mayor que el necesario en binario natural.

Ejemplo: 114 = 000100010100 (BCD, 12 bits)

= 1110010 (binario natural, 7 bits)

10 = 1010 (binario, 4 bits)

= 00010000 (BCD, 8 bits)

Las APLICACIONES de este código se encuadran en voltímetros digitales, frecuencímetros, cronómetros, etc. es decir en todo sistema que implique la lectura/escritura de información por parte de un operador. Su uso en cálculos (aritmética lógica) no es común porque al no ser eficiente requiere de mayor espacio de memoria y dificulta las operaciones aritméticas reduciendo la velocidad del sistema.

BCD Aiken

Este es otro código BCD PONDERADO pero en el que se alteran los pesos de las posiciones de los bits respecto del binario natural (2 4 2 1). Como consecuencia de esto, las cinco primeras combinaciones de este código son las mismas que en binario natural y las cinco restantes son el "espejo" de las primeras.

Este código tiene, además, la propiedad de ser AUTOCOMPLEMENTARIO. Un código es autocomplementario cuando la combinación correspondiente al complemento a 9 de la combinación N, es decir 9 - N, se obtiene invirtiendo la combinación correspondiente a N. En otras palabras, el complemento a 9 de la combinación enésima se obtiene invirtiendo dicha combinación. (El complemento a 9 se usa en resta de decimales por lo que el Aiken facilita esta operación). Por ejemplo, N = 3 --------> 0011

9 - N = 6 ---> 1100

¿Porqué son útiles los autocomplementarios?
Porque la resta de decimales se hace precisamente haciendo una suma del minuendo más el complemento a 9 del sustraendo.
Por ejemplo:

```
342 - 128 =    342
             + 871
               -----
             1  213
                +1
               -----
```

Las reglas de esta operación son:

- Si existe acarreo (como en el ejemplo anterior) debe sumarse al dígito menos significativo.
- Si el resultados es negativo se le debe aplicar el complemento a 9.
- Minuendo y sustraendo deben tener tantos dígitos como dígitos vaya a tener el resultado (ver último ejemplo).

Ejemplos:
a) 8-6 = 8+3 = 1 + 1(acarreo) = 2
b) 7-2 = 7+7 = 4+ 1(acarreo) = 5
c) 24-12 = 11 + 1(acarreo) = 12
d)99-11 = 99+88 = 87+ 1(acarreo) = 88
e) 5-7 = 5+2 = 7 = 2 (comp. 9)
f) 1-1 = 1+8 = 9 = 0 (comp. 9)
g) 18-27 = 18+72 = 90 = 09 (comp. 9)
h) -4-2 = 5+7 = 2+1(acarreo) = 3 = 6 (comp. 9)
i)-3-7 = -03-07 (agrego un dígito pues la suma será de dos dígitos)
= 96 + 92 = 88 +1(acarreo) = 89 = 10 (comp. 9)

BCD Exceso 3

Es un BCD natural desplazado en 3 ya que la conversión desde decimal se realiza igual pero sumando 3 a cada dígito decimal antes de codificarlo en binario. Este código no es ponderado, pero tiene también la propiedad de ser autocomplementario.
Ejemplo:

23_{10} ----> 2 3 99_{10} --- 9 9
+ 3 + 3 + 3 + 3
---- ---- --- ---
5 6 = 0101 0110 12 12 = 1100 1100

Debe tenerse claro que, una vez sumado 3, se codifica en binario natural y no en BCD natural

Las combinaciones no válidas en BCD exceso 3 no son las mismas que en BCD natural, ya que se ha desplazado tres posiciones:

0=0000 1=0001 2=0010 13=1101 14=1110 15=1111

Otros códigos BCD

Cada uno podría inventar un código si es que necesita usarlo en una aplicación particular. Existen varios BCD además de los mencionados; son todos ponderados pero algunos con pesos negativos. Por ejemplo, con pesos sólo positivos se tiene:

5 2 1 1 6 3 1 1 7 3 2 1 4 3 1 1 5 4 2 1

y con pesos positivos y negativos:

5 3 1 -1 6 3 -1 -1 7 5 3 -6 8 4 -3 -2 5 4 2 -3

Al Exceso 3 y al Aiken pueden agregarse, además, los siguientes códigos BCD ponderados autocomplementarios:

4 2 2 1 3 3 2 1 6 3 1 -1 (ver en la Tabla al final el desarrollo de este código)

CÓDIGOS CONTINUOS

Son aquellos en los que cada combinación difiere de la anterior y posterior en un sólo bit. Esto los hace particularmente útiles en codificación de información para la medición de desplazamientos lineales y angulares (rotaciones) de tipo absoluto mediante encoders ópticos. Son NO PONDERADOS.

Código Gray

Recibe también el nombre de REFLEJADO ya que la obtención de un código Gray de n bits se obtiene a partir del de n-1 bits, repitiendo simétricamente las combinaciones de éste y añadiendo por la izquierda un bit 0 para las 2^{n-1} primeras combinaciones y un 1 para las 2^{n-1}combinaciones siguientes.

Luego, para obtener el código Gray se procede de la siguiente manera:

- se escriben un 0 y un 1 y se obtiene un Gray de 2 bits (a)
- se establece un eje de simetría y se reflejan ambos (b)
- se agregan 0 a las primeras 2 combinaciones (encima del espejo) y 1 a las 2 combinaciones siguientes (debajo del espejo) (c)
- si es necesario mayor cantidad de bits, se refleja (d) y continúa el proceso (e)

0	0	00	00	000
1	1	01	01	001

	1	11	11	011
	0	10	10	010

			10	110
			11	111
			01	110
			00	110
(a)	(b)	(c)	(d)	(e)

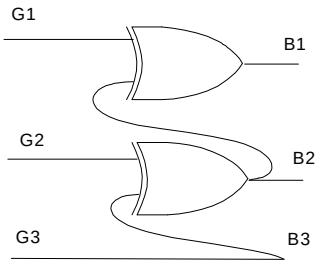
El código Gray es, además, un código CÍCLICO debido a que la última combinación difiere de la primera en un solo bit.

La UTILIDAD PRIMORDIAL de este código está en la transducción de desplazamientos lineales o angulares. El hecho de ser continuo lo hace aventajar notablemente al binario natural en la facilidad de lecturas de regletas codificadas mediante sistemas ópticos. Por ejemplo, el cambio de 7 a 8 en binario natural implica la modificación de cuatro bits simultáneamente mientras que en Gray sólo varía uno. Esto disminuye notablemente la posibilidad de lecturas erróneas. Por ejemplo,

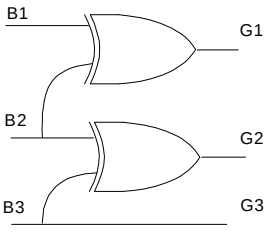
decimal	binario natural	Gray
7	0111	0100
	1111 (15)	
8	1000	1100

Una GRAN VENTAJA de este código es su facilidad de conversión al binario natural mediante las funciones O-exclusiva:

Bi = Gi xor Bi+1 para i entre 1 y n-1
Bn = Gn



Gi = Bi xor Bi+1 para i entre 1 y n-1
Gn = Bn



Código Johnson

Para n bits permite representar 2n combinaciones diferentes, de modo que no es muy eficiente (50%). Su principal VENTAJA reside en la gran sencillez de diseño de sistemas de conteo en este código y su descodificación.

CODIGOS DETECTORES DE ERROR

En el manejo, y especialmente en la transmisión de información, es posible que se produzcan errores debido a la presencia de ruido (se entiende por tal a cualquier deformación de la señal original) en el proceso o por falla de los componentes. Se entiende por error a la aparición en la señal de una combinación que no pertenece al código que está siendo utilizado. Dadas las características tecnológicas, es muy raro que existan errores de más de un bit. Por otro lado, la detección y corrección de errores de más de un bit es muy cara y sólo se justifica en casos de extrema necesidad como por ejemplo los desarrollos militares y espaciales..

Si en un código binario se usan todas las combinaciones posibles (2^n) nunca se podrá detectar un error ya que cualquier variación transformará a la combinación en otra que también pertenece al código. Por lo tanto, la detección de errores en un código binario se logra sólo si no se utilizan todas las combinaciones posibles. Esta es una condición necesaria pero no suficiente ya que continuamos con la posibilidad de que el error cree una combinación válida. Por ejemplo, el BCD exceso tres no usa todas las combinaciones, pero si por error de un bit, la combinación 0011 se convierte en 0111, no es posible detectarlo.

Para establecer la condición necesaria y suficiente definimos la DISTANCIA entre dos combinaciones binarias cualesquiera como el número de bits que deben cambiarse en una de ellas para obtener la otra. Como puede verse, esta distancia es variable; por ejemplo en el binario natural varía entre 1 y 4. La DISTANCIA MÍNIMA en un código se define como la menor distancia que pueda hallarse entre dos combinaciones cualesquiera. Por ejemplo, en el binario natural y los BCD vistos antes la distancia mínima es 1 y por lo tanto un error en un bit de un número cualquiera puede convertirlo en otro número perteneciente al mismo código y hacer que el error no sea detectable.

De esto se deduce que, para que un código pueda detectar errores, su distancia mínima debe ser mayor que 1. En general el número de bits erróneos que se pueden detectar es igual al número en que la distancia mínima supera a la unidad. Así, si la dm = 2 se detectan errores de 1 bit

CÓDIGOS DE PARIDAD

Se basan en el agregado de un bit extra a la palabra binaria que constituye el dato y que se denomina BIT DE PARIDAD.

En el método de PARIDAD PAR se busca que siempre el dato tenga siempre un número par de 1 lógicos. Luego, si el dato ya lo contiene, el bit de paridad será un 0, de lo contrario será un 1. El método de PARIDAD DE IMPAR es análogo pero para número impar de unos.

A cualquier código puede agregársele un bit de paridad con lo que su distancia mínima pasa a ser igual a 2 y, con ello, permite la detección de errores de 1 sólo bit. A pesar de esta limitación, debido a la baja probabilidad de producción de errores, los códigos de paridad son los más utilizados.

Ejemplo de código de paridad impar es el BCD exceso tres al que se le agregó un bit de paridad; se puede ver que la distancia mínima ha aumentado a 2.

OTROS CÓDIGOS DETECTORES

Otros códigos detectores son el 2 entre 5 y biquinario, ambos de distancia mínima igual a 2.

CODIGOS CORRECTORES DE ERROR

Los códigos detectores sólo determinan si existe un error pero no son capaces de corregirlos. En el caso de sistemas digitales que trabajan en tiempo real, como por ejemplo los que actúan sobre un proceso industrial a partir de los valores de ciertas variables del proceso, se hace necesario poder corregir los errores en las transmisiones ya que no es posible volver a "pedir" un dato.

Para esto existen códigos correctores que analizan la información recibida y, si hay error, detectan en qué bit se produjo de modo que pueden corregirlo. Los CÓDIGOS DE HAMMING son los de mayor difusión dentro de los correctores de errores de un sólo bit.

PRINCIPALES CÓDIGOS BINARIOS

	Binario natural 8421	*BCD natural* 8421	*BCD exceso 3*	*BCD Aiken* 2421	*BCD autocomp.* 631-1	*Gray*	*Johnson*	*BCD exceso 3 con paridad impar*
0	0000	0000	0011	0000	0000	0000	00000	0011 1
1	0001	0001	0100	0001	0010	0001	00001	0100 0
2	0010	0010	0101	0010	0101	0011	00011	0101 1
3	0011	0011	0110	0011	0100	0010	00111	0110 1
4	0100	0100	0111	0100	0110	0110	01111	0111 0
5	0101	0101	1000	1011	1001	0111	11111	1000 0
6	0110	0110	1001	1100	1011	0101	11110	1001 1
7	0111	0111	1010	1101	1010	0100	11100	1010 1
8	1000	1000	1011	1110	1101	1100	11000	1011 0
9	1001	1001	1100	1111	1111	1101	10000	1100 1
10	1010					1111		
11	1011					1110		
12	1100					1010		
13	1101					1011		
14	1110					1001		
15	1111					1000		