Binary Codes

Computer Mathematics I

Jiraporn Pooksook
Department of Electrical and Computer Engineering
Naresuan University

BINARY CODES: BCD

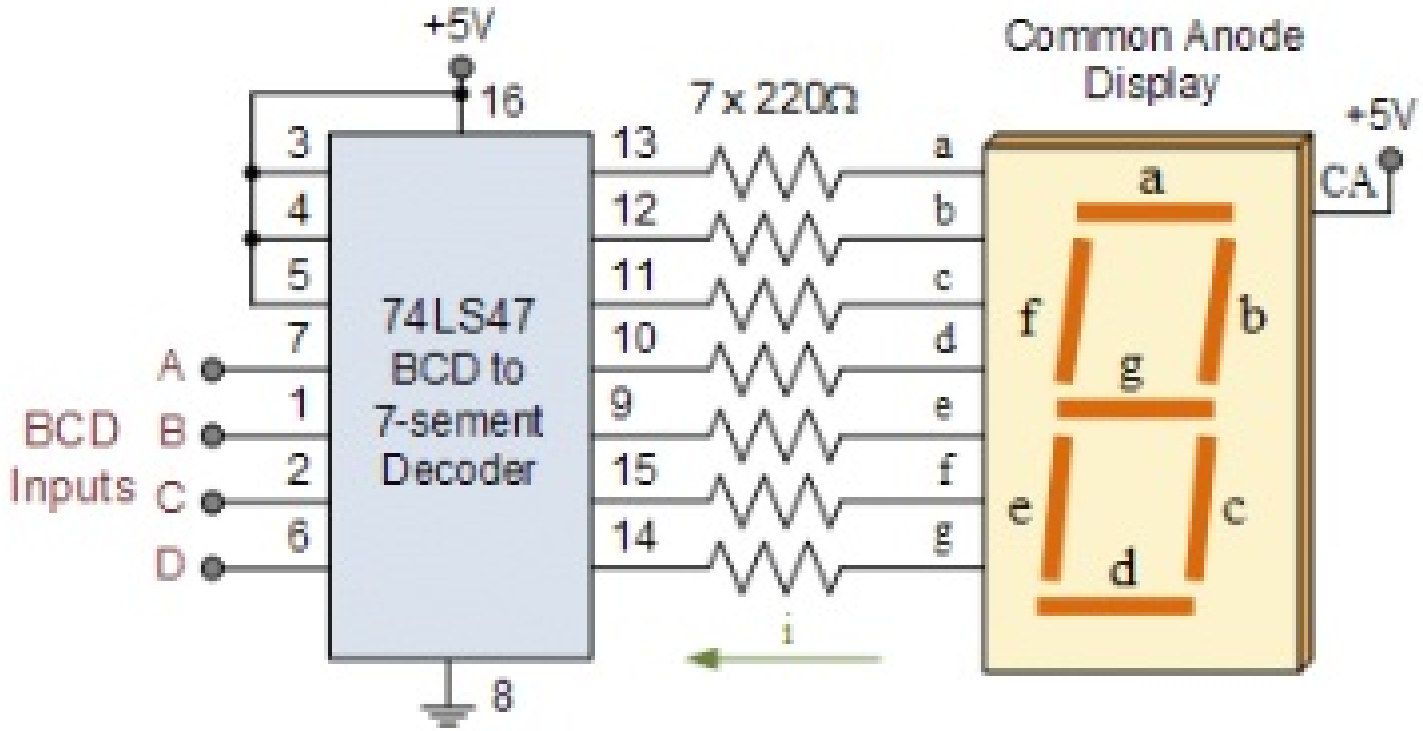

Figure: Retrieved from https://www.electronics-tutorials.ws/binary/binary-coded-decimal.html

BINARY CODES: BCD

Display	D	C	B	A	a	b	c	d	e	f	g
0	0	0	0	0	1	1	1	1	1	1	0
1	0	0	0	1	0	1	1	0	0	0	0
2	0	0	1	0	1	1	0	1	1	0	1
3	0	0	1	1	1	1	1	1	0	0	1
4	0	1	0	0	0	1	1	0	0	1	1
5	0	1	0	1	1	0	1	1	0	1	1
6	0	1	1	0	1	0	1	1	1	1	1
7	0	1	1	1	1	1	1	0	0	0	0
8	1	0	0	0	1	1	1	1	1	1	1
9	1	0	0	1	1	1	1	1	0	1	1

Figure: Retrieved from http://electronics-course.com/bcd-7-segment

BINARY CODES: BCD

Binary Coded Decimal system is represented by a group of 4 bits using binary notation from 0000 to 1001.
Binary coded decimal is different from hexadecimal because BCD numbers stop at 9 binary 1001_2.
The advantage of BCD is that it allows easy conversion between decimal (base-10) and binary (base-2) form. However, the disadvantage is that BCD code is wasteful.

Example: 12 to BCD=?
$0001_2 = 1$
$0010_2 = 2$
Then, = 0001 0010

Example: 92 to BCD=?
$1001_2 = 9$
$0010_2 = 2$
Then, = 1001 0010

BINARY CODES: EXCESS-3

This is an unweighted code. It is obtained from the value of BCD adding by 3.

Example: 12 to Excess-3=?
$0100_2 = 1$
$0101_2 = 2$
Then, = 0100 0101

Example: 92 to Excess-3=?
$1100_2 = 9$
$0101_2 = 2$
Then, = 1100 0101

BINARY CODES: 84-2-1

This is also a weighted code similarly to BCD.

Example: 12 to 84-2-1=?
0111 = 1
0110 = 2
Then, = 0111 0110

Example: 92 to 84-2-1=?
1111 = 9
0110 = 2
Then, = 1111 0110

BINARY CODES: 2421

This is also a weighted code similarly to BCD.

Example: 12 to 2421=?
0001 = 1
0010 = 2
Then, = 0001 0010

Example: 92 to 2421=?
1111 = 9
0010 = 2
Then, = 1111 0010

SELF-COMPLEMENTING

The excess-3, 84-2-1, and 2421 codes are examples of self-complementing codes.

Example : Excess-3

395 = 0110 1100 1000

604 = 1001 0011 0111 (The 9's complement of each number)

TABLE 1-2
Binary codes for the decimal digits

Decimal digit	(BCD) 8421	Excess-3	84-2-1	2421	(Biquinary) 5043210
0	0000	0011	0000	0000	0100001
1	0001	0100	0111	0001	0100010
2	0010	0101	0110	0010	0100100
3	0011	0110	0101	0011	0101000
4	0100	0111	0100	0100	0110000
5	0101	1000	1011	1011	1000001
6	0110	1001	1010	1100	1000010
7	0111	1010	1001	1101	1000100
8	1000	1011	1000	1110	1001000
9	1001	1100	1111	1111	1010000

Figure: Retrieved from Digital Design / M. Morris R. Mano, Micheal D. Ciletti.-3nd ed.

GRAY CODE

If the device uses natural binary codes, positions 3 and 4 are next to each other but all three bits of the binary representation differ.In the transition between states 3 and 4, all three switches change state.

The transition might look like 011-001-101-100. When the switches appear to be in position 001, the observer cannot tell if that is the "real" position 001, or a transitional state between two other positions.

Gray code is **cyclic** which means it is arranged so that every transition from one value to the next value involves only one bit change.
The example of 2 different cyclic gray codes:

000,001,011,010,110,111,101,100

000,001,011,111,101,100,110,010

GRAY CODE

Converting Binary to Gray:

1. The most significant bit of the gray number is the most significant bit of the binary code
2. Add the next significant bit of the binary number to the next significant bit of the binary number to obtain the next gray coded bit
3. Repeat until all bits are changed.

Example:
Binary = 0000 Gray = 0000
Binary = 0001 Gray = 0001
Binary = 0010 Gray = 0011
Binary = 0011 Gray = 0010
Binary = 0100 Gray = 0110

CONSTRUCT N-BIT GRAY CODE

n-bit code can be generated by n-1 bit code.
Example(from 2 bits to 3 bits):
Adding prefix 0 to the gray code in 2 bits
Gray = 00 $\Rightarrow$ 000
Gray = 01 $\Rightarrow$ 001
Gray = 11 $\Rightarrow$ 011
Gray = 10 $\Rightarrow$ 010
Then adding prefix 1 to the reflects
Gray = 10 $\Rightarrow$ 110
Gray = 11 $\Rightarrow$ 111
Gray = 01 $\Rightarrow$ 101
Gray = 00 $\Rightarrow$ 100
This method is the cyclic code.

ERROR DETECTING CODE: BIT PARITY

Binary information can transmit from one location to another. The purpose of an error-detection code is to detect bit-reversal errors. A parity bit is an extra bit included with a message to make the total number of 1's transmitted either odd or even.

- If the odd parity is adopted, the P bit is chosen such that the total number of 1's is odd in the five bits that constitute the message and P.
- If the even parity is adopted, the P bit is chosen such that the total number of 1's is even in the five bits that constitute the message and P.

Even parity is more common.

ERROR DETECTING CODE: BIT PARITY

Odd parity		Even parity	
Message	P	Message	P
0000	1	0000	0
0001	0	0001	1
0010	0	0010	1
0011	1	0011	0
0100	0	0100	1
0101	1	0101	0
0110	1	0110	0
0111	0	0111	1
1000	0	1000	1
1001	1	1001	0
1010	1	1010	0
1011	0	1011	1
1100	1	1100	0
1101	0	1101	1
1110	0	1110	1
1111	1	1111	0

Figure: Retrieved from Digital Design / M. Morris R. Mano, Michael

ERROR DETECTING CODE: BIT PARITY

The parity bit's task:

- If an even parity bit is generated in the sending end for each message transmission, the message together with the parity bit is transmitted to its destination.
- The parity of the received data is checked. If the parity of the received information is not even, it means that at least one bit has changed value during the transmission.
- After an error is detected, there are many solution depending on the situation. One possibility is to request retransmission.

EVEN-PARITY BCD

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

EVEN-PARITY BCD

- If the number of 1's is odd, then P = 1.
- If the number of 1's is even, then P = 0.

Example: 0100 1 , 0110 0
What if the data 01010 is changed to
01011 Detectable?
11010 Detectable?
11111 Detectable?
00110 Detectable?
Problem: Can't detect which bit is in error. Can't detect even numbers of errors.

ERROR-DETECTING CODE: BIT PARITY

Minimum distance of a code = smallest no. of bits in which any two codewords differ

Example:

Message 1 = 01101

Message 2 = 10011

Message 3 = 00110

Message 4 = 11000

Distance between codes:

Message 1 and Message 2 = 4

Message 1 and Message 3 = 3

Hence, the minimum distance of a code = 3.

ERROR-DETECTING CODE

The capability of code to be error detecting and/or error-correcting can be determined from its minimum distance. If a code's minimum distance is 2c+d+1 , it can correct errors in upto c bits and detect errors in upto d additional bits.

What is the minimum distance of Even-Parity BCD ?

ERROR-DETECTING CODE: 2 OUT OF 5 CODE

A two-out-of-five code is an m of n code that provides exactly ten possible combinations. It uses five bits representation. There are ways to assign weights to each bit such that the set bits sum to the desired value, with an exception for zero. A two-out-of-five code can have the number of 1's = 2.

Decimal	01247
0	00011
1	11000
2	10100
3	01100
4	10010
5	01010
6	00110
7	10001
8	01001
9	00101

ERROR-DETECTING CODE: 2 OUT OF 5 CODE

If the number of 1's $\neq 2$ and the number of 0's $\neq 3$, detects as errors.

What is the minimum distance of 2-out-of-5 code?
What if the data 01010 is changed to
01011 Detectable?
11010 Detectable?
01001 Detectable?
00110 Detectable?

ERROR-CORRECTING

- After an error is detected, request for retransmission.
- The sender will send multiple times of data, usually three times.
- The receiver justifies the error using the repeated messages.

Example: Even-Parity BCD
Message = 01010

- If we receive 01010 (no error)
- If we receive 01011, we request for a retransmission. Then, we receive 01010 01010 01010 (no error).
- If we receive, 01011, we request for a retransmission. Then, we receive 01010 01011 01110. (error)

ERROR-CORRECTING: HAMMING CODE

- It is invented by R. W. Hamming.
- k parity bits are added to an n-bit data word, forming a new word of n + k bits.
- The bit positions are numbered in sequence from 1 to n + k.
- Those positions numbered with powers of two are reserved for the parity bits.
- The remaining bits are the data bits.
- The code can be used with words of any length.

Example: data 8 bits = 11000100 and four parity bits

Bit position	1	2	3	4	5	6	7	8	9	10	11	12
	P_1	P_2	1	P_4	1	0	0	P_8	0	1	0	0

ERROR-CORRECTING: HAMMING CODE

	D_1	D_2	D_3
P_1	x		x
P_2		x	x

- Add at least 2 parity bits to the data bits.
- Calculate 2 parity bits from its group.
- When errors are detected, we can justify with the group of parity bits.

ERROR-CORRECTING: HAMMING CODE

Example: 2 parity bits + 3 data bits
Data = 101 with even parity hamming code

	1	0	1
P_1	x		x
P_2		x	x

- $P_1 = 0$ (the number of 1's is even)
- $P_2 = 1$ (the number of 1's is odd)

Thus, message = 01 101

ERROR-CORRECTING: HAMMING CODE

Example: 2 parity bits + 3 data bits
Data = 101 with even parity hamming code

	1	0	1
P_1	x		x
P_2		x	x

Message = 01 101
If we receive 01 001

- (1st bit) parity bit = 0 then the number of 1's of data bit position 1,3 must be even. (error)
  data bit position 1 = 0 and data bit position 3 = 1, hence bit position 1 must be 1.
- (2nd bit) parity bit = 1 then the number of 1's of data bit position 2,3 must be odd. (correct)

CREATE HAMMING CODE

Example: Create hamming code with 3 data bits and 3 parity bits.
Data = 101

Bit position	1	2	3	4	5	6
	P_1	P_2	D_1	P_4	D_2	D_3

$P_1 = 1$, $P_2 = 0$, $P_3 = 1$
$D_1 = ?$, $D_2 = ?$

CREATE HAMMING CODE

Any numbers can be written in base 2 as sum of 2^i

number	1	2	3	4	5	6	7
$2^0=1$	x		x		x		x
$2^1=2$		x	x			x	x
$2^2=4$				x	x	x	x

CREATE HAMMING CODE

Each parity bit depends on its group which is defined by its summation position.

	P_1	P_2	D_1	P_4	D_2	D_3
2^0=1	x		x		x	
2^1=2		x	x			x
2^2=4				x	x	x
position	1	2	3	4	5	6

P_1 = XOR of bits D_1,D_2
P_2 = XOR of bits D_1,D_3
P_3 = XOR of bits D_2,D_3

CREATE HAMMING CODE

Example: Data = 101

	P_1	P_2	1	P_4	0	1
2^0=1	x		x		x	
2^1=2		x	x			x
2^2=4				x	x	x
position	1	2	3	4	5	6

P_1 = XOR of 1, 0 = 1
P_2 = XOR of 1, 1 = 0
P_3 = XOR of 0, 1 = 1
Hence, message = 101101

1 BIT CORRECTION: HAMMING CODE

Example: Message = 101101

	P_1	P_2	1	P_4	0	1
2^0=1	x		x		x	
2^1=2		x	x			x
2^2=4				x	x	x
position	1	2	3	4	5	6

If we receive 101111

- check bit 1: = 1 $\Rightarrow$ check bits 3,5 = 1,1 (wrong)
- check bit 2: = 0 $\Rightarrow$ check bits 3,6 = 1,1 (ok)
- check bit 4: = 1 $\Rightarrow$ check bits 5,6 = 1,1 (wrong)
- the bad bit is bit 1+4 = 5.
- Data is error. It should change bit 5 from 1 to 0.

1 BIT CORRECTION: HAMMING CODE

Example: Message = 101101

	P_1	P_2	1	P_4	0	1
2^0=1	x		x		x	
2^1=2		x	x			x
2^2=4				x	x	x
position	1	2	3	4	5	6

If we receive 111101

- check bit 1: = 1 $\Rightarrow$ check bits 3,5 = 1,0 (ok)
- check bit 2: = 1 $\Rightarrow$ check bits 3,6 = 1,1 (wrong)
- check bit 4: = 1 $\Rightarrow$ check bits 5,6 = 0,1 (ok)
- the bad bit is bit 2.
- Parity bit is error. Data is correct.