

Codes

❖ BCD (Binary Coded Decimal)

Addition on BCD

❖ Gray Code

❖ Parity Code

❖ Two-out-of-five Code

❖ Aiken Code

❖ Alphanumeric Codes

Codes

Some examples of data we use in our daily life are numbers, texts, pictures, videos, etc. But digital systems are based on only 0 and 1. The process of converting data to digital systems is called encoding.

There are two types of codes: Numeric and Alphanumeric codes.

There are two types of numeric codes: The ones which are based on weights of digits, and the ones which aren't.

Most widely used alphanumeric codes are ASCII, Extended ASCII, EBCDIC and UNICODE.

BCD (Binary Coded Decimal)

In BCD code, each decimal digit is converted to a 4-bit binary number. We convert each decimal digit separately. The weights of this 4-bit binary number are 2^3 - 2^2 - 2^1 - 2^0 (8-4-2-1) respectively. This code is also known as 8421 code.

Example: Let's convert 25_{10} to BCD.

2	5	
↓	↓	
0010	0101	So, $25_{10} = 00100101_{\text{BCD}}$

BCD Addition

Since each bit have a weight, we can do arithmetic operations on BCD. We do BCD addition regarding three rules below.

1. We do BCD addition the same way as binary addition.
2. If the sum of two 4-bit groups is less than or equal to 9, then the result is valid.
3. If the sum is greater than 9 or we have a carry bit, then the result is invalid. In this case, we add 6 $(0110)_2$ to the sum. The new sum is valid even if we have a carry bit *after* adding 6.

BCD Addition

Example: Let's do $8_{10} + 4_{10}$ in BCD.

$$8_{10} = 1000_{\text{BCD}}$$

$$4_{10} = 0100_{\text{BCD}}$$

$$\begin{array}{r} 1000_{\text{BCD}} \\ + 0100_{\text{BCD}} \\ \hline \end{array}$$

$$\begin{array}{r} 1100_{\text{BCD}} \\ + 0110_{\text{BCD}} \\ \hline \end{array}$$

$$\begin{array}{r} 1100_{\text{BCD}} \\ + 0110_{\text{BCD}} \\ \hline \end{array}$$

$$\begin{array}{r} 1100_{\text{BCD}} \\ + 0110_{\text{BCD}} \\ \hline \end{array}$$

$0001\ 0010_{\text{BCD}}$ → We have a carry bit, but it's OK.

Example: Let's do $39_{10} + 97_{10}$ in BCD.

$$39_{10} = 0011\ 1001_{\text{BCD}}$$

$$97_{10} = 1001\ 0111_{\text{BCD}}$$

1

$$\begin{array}{r} 0011\ 1001_{\text{BCD}} \\ + 1001\ 0111_{\text{BCD}} \\ \hline \end{array}$$

$$\begin{array}{r} 0011\ 1001_{\text{BCD}} \\ + 1001\ 0111_{\text{BCD}} \\ \hline \end{array}$$

We need to add 6
Because it's greater than 9.

← 1101 0000 → We need to add 6
Because there's a carry bit.

$$\begin{array}{r} 1101\ 0000 \\ + 0110\ 0110 \\ \hline \end{array}$$

$$0001\ 0011\ 0110_{\text{BCD}}$$

Gray Code

In Gray code, bits don't have weights. Therefore, we can't do arithmetic operations in Gray code. What makes gray code special is that only one bit changes when we increase a number by one. This code is mostly used for error-checking purposes.

Decimal	Binary	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

Gray Code

To convert from binary to Gray code, we take the MSB as it is. We calculate next Gray code bits by adding each two consequent bits. We ignore the carry bits if there are any.

Example: Let's convert 101101_2 to Gray code.

$$\begin{array}{cccccc} 1 & + & 0 & + & 1 & + & 1 & + & 0 & + & 1 \\ \downarrow & & \downarrow & & \downarrow & & \downarrow & & \downarrow & & \downarrow \\ 1 & & 1 & & 1 & & 0 & & 1 & & 1 \end{array} \quad \text{So, } 101101_2 = 111011_{\text{Gray}}$$

To convert from Gray code to binary, we take the MSB as it is. We calculate next bit of binary number by adding the binary MSB to the next Gray code bit. We calculate the next binary bit by adding the second-left bit of the binary number to the third-left bit of Gray code number. And it goes on by repeating the same process for next bits. We ignore the carry bits if there are any.

Example: Let's convert 101101_{Gray} to binary.

$$\begin{array}{cccccc} 1 & 0 & 1 & 1 & 0 & 1 \\ \downarrow + & \downarrow + & \downarrow + & \downarrow + & \downarrow + & \downarrow \\ 1 & 1 & 0 & 1 & 1 & 0 \end{array} \quad \text{So, } 101101_{\text{Gray}} = 110110_2$$

Parity Code

Parity code is mostly used for checking the validity of data transferred. There are two types of parity code: Even parity, and odd parity. When we need to transfer 7-bit data, we can add an extra bit for error checking.

Assuming that we are using odd parity bit, we check if the number of 1's are odd or even. If the number of 1's is odd, then we add 0 as parity bit. If it is even, we add 1. Likewise, if we are using even parity bit, we add 0 if the number of 1's is even, and 1 if it's odd.

Example: Let's code $(1010001)_2$ with *odd* parity.

The number of 1's is 3, which is an odd number. Therefore, we need to add 0 as parity bit. So, encoded data would be **0**1010001₂.

Example: Let's code $(1010001)_2$ with *even* parity.

The number of 1's is 3, which is an odd number. Therefore, we need to add 1 as parity bit. So, encoded data would be **1**1010001₂.

Two-out-of-five Code

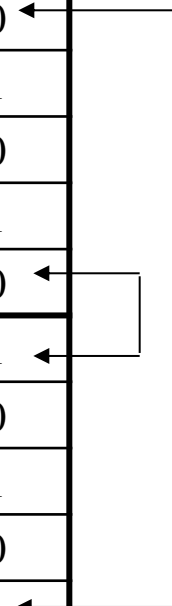
In 2-out-of-5 code, we represent each decimal digit with a 5-bit binary number which have two 1's in it. This makes error checking easier. The weights of bits are '7-4-2-1-0'.

Digits	2-out-of-5 code
	7 4 2 1 0
0	1 1 0 0 0
1	0 0 0 1 1
2	0 0 1 0 1
3	0 0 1 1 0
4	0 1 0 0 1
5	0 1 0 1 0
6	0 1 1 0 0
7	1 0 0 0 1
8	1 0 0 1 0
9	1 0 1 0 0

Aiken Code

In Aiken code, we represent each decimal digit with a 4-bit binary number. The weights of bits are '2-4-2-1'. Digits 0-4 start with 0, and 5-9 start with 1. Aiken code is example of symmetric codes.

Digit	Aiken Code
	2 4 2 1
0	0 0 0 0
1	0 0 0 1
2	0 0 1 0
3	0 0 1 1
4	0 1 0 0
5	1 0 1 1
6	1 1 0 0
7	1 1 0 1
8	1 1 1 0
9	1 1 1 1



Alphanumeric Codes

- In transmission, we need not only numbers, but also the letters, punctuation marks and other symbols. The most popular alphanumeric code is ASCII (American Standard Code for Information Interchange).
- ASCII, is an alphanumeric code which is widely used in computers and other systems. For example, when we press a key on the keyboard, a numeric code that corresponds to that key is sent to the computer. This numeric code consists of 7-bits. 8th bit can be used for error checking. With 7 bits, we can represent different characters. First 32 characters are not displayed on the screen, but the rest can be displayed. We can also use ASCII code in transmission between computers and printers.

ASCII

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040	 	Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH (start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX (start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX (end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT (end of transmission)	36	24	044	$	\$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ (enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK (acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL (bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS (backspace)	40	28	050	((72	48	110	H	H	104	68	150	h	h
9	9	011	TAB (horizontal tab)	41	29	051))	73	49	111	I	I	105	69	151	i	i
10	A	012	LF (NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT (vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF (NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR (carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO (shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI (shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE (data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1 (device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2 (device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3 (device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4 (device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK (negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN (synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB (end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN (cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM (end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB (substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC (escape)	59	3B	073	;	;	91	5B	133	[[123	7B	173	{	{
28	1C	034	FS (file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174	|	
29	1D	035	GS (group separator)	61	3D	075	=	=	93	5D	135]]	125	7D	175	}	}
30	1E	036	RS (record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US (unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177		DEL

Source: www.LookupTables.com