3장. 디지털 코드

01 숫자 코드

02 에러 검출 코드

03 문자 코드

BCD 코드와 3초과 코드

- **BCD**코드(Binary Coded Decimal Code : 2진화 10진 코드, 8421 코드)
- 10진수 0부터 9까지를 2진화한 코드
- 표기는 2진수이지만 의미는 10진수
- 1010부터 1111까지의 6개는 사용되지 않음

10진수	BCD 코드	10진수	BCD 코드	10진수	BCD 코드
0	0000	10	0001 0000	20	0010 0000
1	0001	11	0001 0001	31	0011 0001
2	0010	12	0001 0010	42	
3	0011	13	0001 0011	53	0101 0011
4	0100	14	0001 0100	64	0110 0100
5	0101	15	0001 0101	75	0111 0101
6	0110	16	0001 0110	86	1000 0110
7	0111	17	0001 0111	97	1001 0111
8	1000	18	0001 1000	196	
9	1001	19	0001 1001	237	

❑ BCD 코드의 연산

10진 덧셈 (6+3=9)

```
   0110
+  0011
------
   1001
```

10진 덧셈 (42+37=79)

```
   0100 0010
+  0011 0111
-------------
```

❖ 계산 결과가 BCD코드를 벗어나는 즉, 9를 초과하는 경우에는 계산 결과에 **6(0110)**을 더해준다.

(8+7=15)

```
   1000
+  0111
-------
```

6

예제 3-1 96+58을 BCD로 바꾸어 연산한 결과는?

풀이 96과 58을 BCD로 바꾸어 계산하면 1110 1110이 되어 십의 자리와 1의 자리 모두가 9를 초과하므로 계산 결과에 0110 0110를 더해주면 올바른 결과를 얻는다.

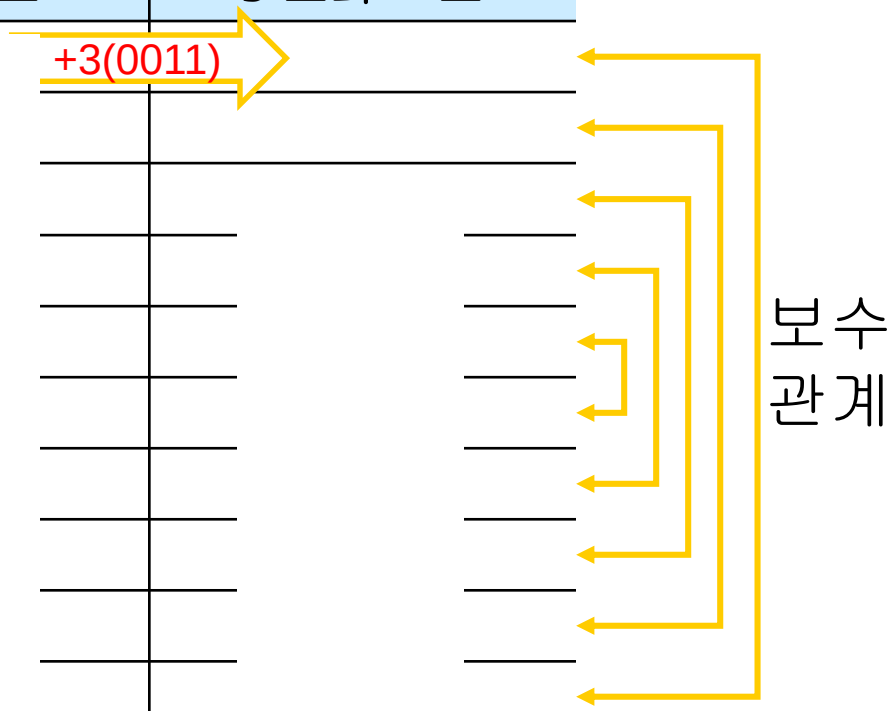
10진 덧셈	BCD 덧셈
96	1001 0110
+ 58	+ 0101 1000
	1110 1110
	+ 0110 0110
154	0001 0101 0100

End of Example

❑ 3초과 코드

- ❖ BCD코드(8421코드)로 표현된 값에 3을 더해 준 값으로 나타내는 코드
- ❖ 자기 보수의 성질이 있음.

10진수	BCD 코드	3-초과 코드
0		
1		
2		
3		
4		
5		
6		
7		
8		
9		



예제 3-2 10진수 38을 3초과 코드로 변환하여라.

풀이 $38_{(10)}$은

- 10의 자리 $0011_{(2)}$ ⇨ 3초과 코드 0110
- 1의 자리 $1000_{(2)}$ ⇨ 3초과 코드 1011
- 따라서 $38_{(10)} = 0011\ 1000_{(\text{BCD code})} = 0110\ 1011_{(\text{excess 3 code})}$

End of Example

다양한 2진 코드들

❑ 가중치코드(weighted code)

❖ 그 위치에 따라 정해진 값을 갖는 코드

10진수	8421코드 (BCD)	2421 코드	84-2-1 코드	링 카운터 (ring counter) 9876543210
0	0000	0000	0000	0000000001
1	0001	0001	0111	0000000010
2	0010	0010	0110	0000000100
3	0011	0011	0101	0000001000
4	0100	0100	0100	0000010000
5	0101	1011	1011	0000100000
6	0110	1100	1010	0001000000
7	0111	1101	1001	0010000000
8	1000	1110	1000	0100000000
9	1001	1111	1111	1000000000

❖ 8421 코드

10진수	8421코드(BCD)	
0	0000	8×0 + 4×0 + 2×0 + 1×0 = 0
1	0001	8×0 + 4×0 + 2×0 + 1×1 = 1
2	0010	8×0 + 4×0 + 2×1 + 1×0 = 2
3	0011	8×0 + 4×0 + 2×1 + 1×1 = 3
4	0100	8×0 + 4×1 + 2×0 + 1×0 = 4
5	0101	**8×0 + 4×1 + 2×0 + 1×1 = 5**
6	0110	8×0 + 4×1 + 2×1 + 1×0 = 6
7	0111	8×0 + 4×1 + 2×1 + 1×1 = 7
8	1000	8×1 + 4×0 + 2×0 + 1×0 = 8
9	1001	8×1 + 4×0 + 2×0 + 1×1 = 9

☞ **자기보수 성질 없음**

01 숫자 코드

❖ 2421 코드

10진수	2421 코드	
0	0000	2×0+4×0+2×0+1×0=0
1	0001	2×0+4×0+2×0+1×1=1
2	0010	2×0+4×0+2×1+1×0=2
3	0011	2×0+4×0+2×1+1×1=3
4	0100	2×0+4×1+2×0+1×0=4
5	1011	2×1+4×0+2×1+1×1=5
6		
7	1101	2×1+4×1+2×0+1×1=7
8	1110	2×1+4×1+2×1+1×0=8
9	1111	2×1+4×1+2×1+1×1=9

☞ 자기보수 성질을 가짐

❖ 84-2-1 코드

84-2-1 코드	
0000	8×0+4×0−2×0−1×0=0
0111	8×0+4×1−2×1−1×1=1
0110	8×0+4×1−2×1−1×0=2
0101	8×0+4×1−2×0−1×1=3
0100	8×0+4×1−2×0−1×0=4
1011	8×1+4×0−2×1−1×1=5
1001	8×1+4×0−2×0−1×1=7
1000	8×1+4×0−2×0−1×0=8
1111	8×1+4×1−2×1−1×1=9

☞ 자기보수 성질을 가짐

예제 3-3 10진수 3468을 2421코드로 변환하여라.

풀이

각 자리 별로 변환하면 다음과 같다. 여기서 3, 4, 6은 각각 2가지 경우가 존재한다.

3	4	6	8
0011 or 1001	0100 or 1010	1100 or 0110	1110

End of Example

❑ 비가중치코드(non-weighted code)

- ❖ 각각의 위치에 해당하는 값이 없는 코드
- ❖ 데이터 변환과 같은 특수한 용도로 사용됨

10진수	3-초과 코드	그레이코드
0		
1		
2		
3		
4		
5		
6		
7		
8		
9		

그레이 코드(Gray Code)

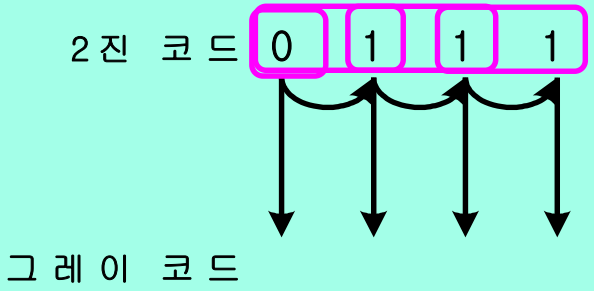
- 가중치가 없는 코드이기 때문에 연산에는 부적당하지만, 아날로그-디지털 변환기나 입출력 장치 코드로 주로 쓰인다.
- 연속되는 코드들 간에 하나의 비트만 변화하여 새로운 코드가 된다.

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

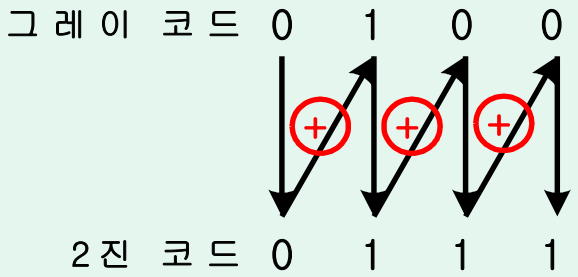
이웃하는 코드간에 한 비트만 다르다.

01 숫자 코드

2진 코드를 그레이 코드로 변환하는 방법



그레이 코드를 2진 코드로 변환하는 방법



XOR 진리표

입력		출력
A	B	F
0	0	
0	1	
1	0	
1	1	

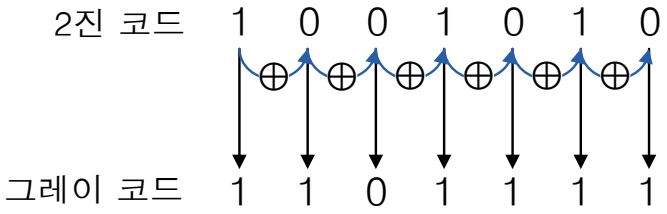
$F=A \oplus B$

예제 3-5 다음 2진 코드는 그레이 코드로, 그레이 코드는 2진 코드로 변환하여라.

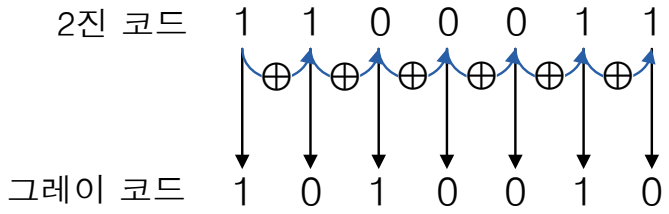
(a) $1001010_{(2)}$ (b) $1100011_{(2)}$ (c) $1001010_{(G)}$ (d) $1011101_{(G)}$

풀이

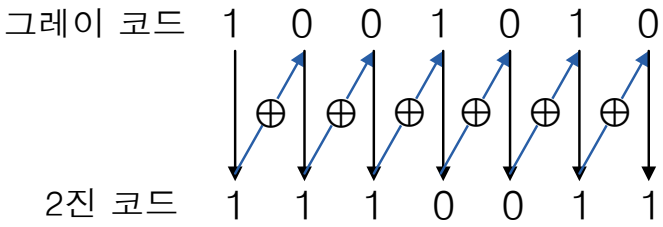
(a) $1001010_{(2)}$



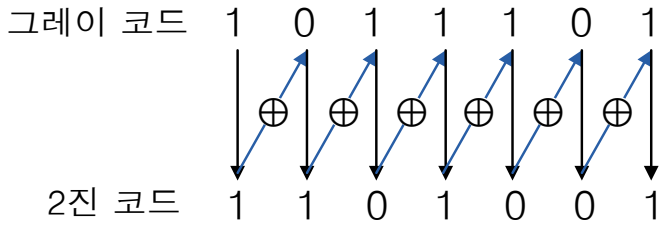
(b) $1100011_{(2)}$



(c) $1001010_{(G)}$



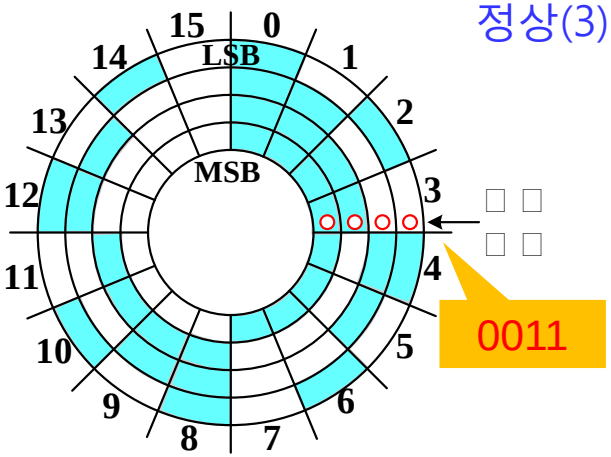
(d) $1011101_{(G)}$

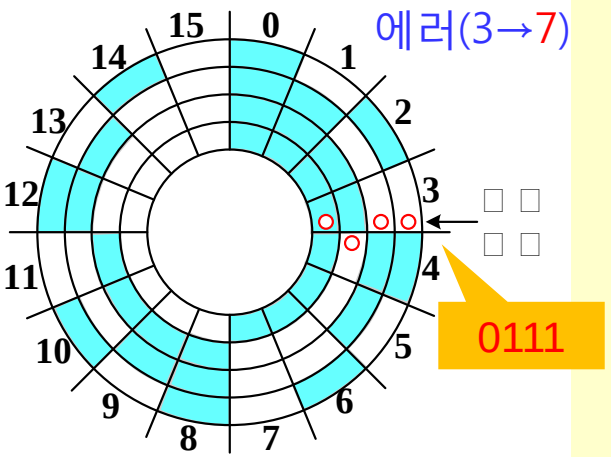


End of Example

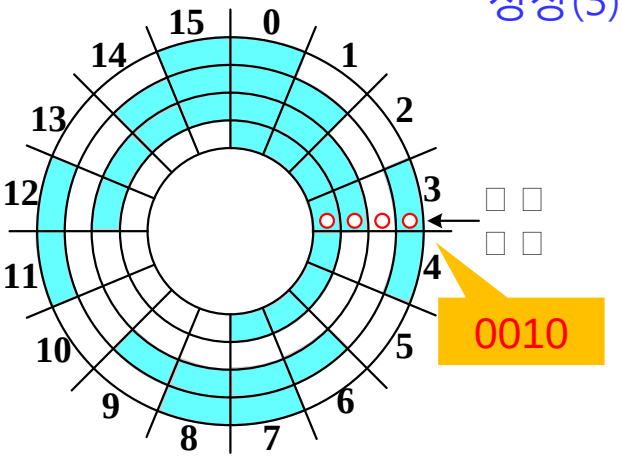
❖ 그레이 코드 입력장치 적용 예

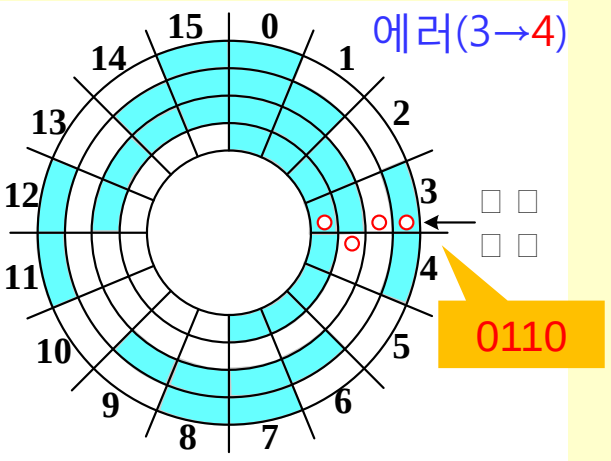
2진코드





그레이코드





0

1

그레이 코드는 오차가 적음

패리티 비트

- 짝수패리티(even parity) : 데이터에서 1의 개수를 짝수 개로 맞춤
- 홀수패리티(odd parity) : 1의 개수를 홀수 개로 맞춤
- 패리티 비트는 데이터 전송과정에서 에러 검사를 위한 추가비트. 패리티는 단지 에러 검출만 가능하며, 여러 비트에 에러가 발생할 경우에는 검출이 안될 수도 있음.

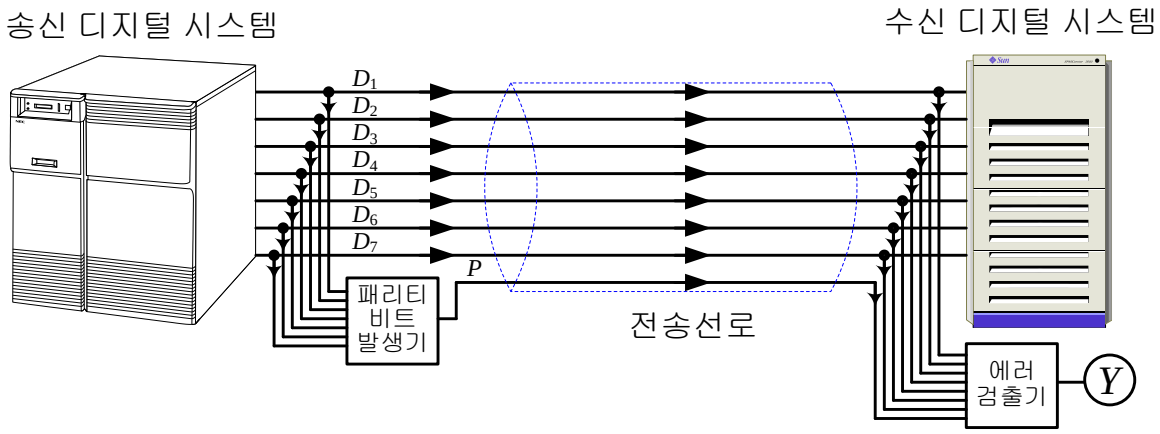
7비트 ASCII 코드에 패리티 비트를 추가한 코드

데이터	짝수패리티	홀수패리티
…	…	…
A	1000001	1000001
B	1000010	1000010
C	1000011	1000011
D	1000100	1000100
…	…	…

❑ 데이터 전송 시스템에서 패리티 비트를 사용한 에러 검출

❖ 에러를 검출하기 위하여 송신측에 패리티 발생기를 구성하고 수신측에는 패리티 검출기를 구성하여 그 출력을 보고 에러 발생 여부를 판단

짝수 패리티	Y=0(에러 없음), Y=1(에러 발생)
홀수 패리티	Y=1(에러 없음), Y=0(에러 발생)



7비트 데이터를 전송할 때 패리티 비트를 사용하여 에러를 검출하는 시스템

예제 3-6 홀수 패리티 시스템에서 코드 그룹 10110, 11010, 110011, 10101110100, 1100010101011을 수신하였다. 에러가 발생한 그룹을 찾아보아라.

풀이

- 홀수 패리티가 필요하므로 1이 짝수 개인 그룹은 에러 발생
- 110011, 10101110100에 비트 에러가 발생하였음

End of Example

병렬 패리티(parallel parity)

❖ 패리티를 블록 데이터에 적용해서 가로와 세로 데이터들에 대해서 패리티를 적용하면 에러를 검출하여 그 위치를 찾아 정정할 수 있다.

1	0	1	0	1	1	1	1	*0*
1	0	0	0	0	0	1	1	*1*
0	1	0	0	0	0	0	0	*1*
1	1	1	1	0	0	0	0	*0*
1	0	1	1	1	0	0	1	*1*
0	0	0	0	0	1	1	1	*1*
1	1	1	1	1	1	1	1	*0*
0	1	1	1	1	0	0	0	*0*
1	*0*	*1*	*0*	*0*	*1*	*0*	*1*	*0*

원래 데이터 블록

가로세로 모두 1의 개수가 짝수임

1	0	1	0	1	1	1	1	*0*
1	0	0	0	0	0	1	1	*1*
0	1	0	0	0	0	0	0	*1*
1	1	1	1	0	0	0	0	*0*
1	0	1	1	0	0	0	1	*1*
0	0	0	0	0	1	1	1	*1*
1	1	1	1	1	1	1	1	*0*
0	1	1	1	1	0	0	0	*0*
1	*0*	*1*	*0*	*0*	*1*	*0*	*1*	*0*

에러가 발생한 블록

가로세로 회색 부분에 1의 개수가 홀수임 : 겹치는 부분 에러

해밍(hamming code)코드

- 에러를 검출하고 정정할 수 있는 코드
- 추가적으로 많은 비트가 필요하므로 많은 양의 데이터 전달이 필요
- 데이터 비트와 패리티 비트와의 관계

$$2^{p-1} - p + 1 \le d \le 2^{p} - p - 1$$

p는 패리티 비트의 수, d는 데이터 비트의 수,

- p=4일 때, $2^{4-1} - 4 + 1 \le d \le 2^{4} - 4 - 1$이므로 $5 \le d \le 11$이다.
- 따라서 데이터 비트수가 5개 이상 11개 이하일 때 패리티는 4개가 필요하다.
- 따라서 데이터 비트수가 5개 이상 11개 이하일 때 패리티는 4개가 필요하다.
- 패리티 비트의 위치는 앞에서 부터 $2^0, 2^1, 2^2, 2^3, 2^4, \ldots$번째, 즉 1, 2, 4, 8, 16, … 번째이다.
- 패리티의 위치에 따라 기호 $P_1, P_2, P_4, P_8, \ldots$로 표시
- 데이터 비트는 나머지 위치에 순서대로 들어간다.

❑ 에러 정정 코드 : 해밍(hamming code)코드

- 해밍코드에서는 짝수 패리티를 사용

비트 위치	1	2	3	4	5	6	7	8	9	10	11	12
기호	P_1	P_2	D_3	P_4	D_5	D_6	D_7	P_8	D_9	D_{10}	D_{11}	D_{12}
P_1 영역	✓		✓		✓		✓		✓		✓	
P_2 영역		✓	✓			✓	✓			✓	✓	
P_4 영역				✓	✓	✓	✓					✓
P_8 영역								✓	✓	✓	✓	✓

❖ 8비트 데이터의 에러 정정 코드

$$P_1 = D_3 \oplus D_5 \oplus D_7 \oplus D_9 \oplus D_{11}$$

$$P_2 = D_3 \oplus D_6 \oplus D_7 \oplus D_{10} \oplus D_{11}$$

$$P_4 = D_5 \oplus D_6 \oplus D_7 \oplus D_{12}$$

$$P_8 = D_9 \oplus D_{10} \oplus D_{11} \oplus D_{12}$$

for example

P_1	P_2	D_3	P_4	D_5	D_6	D_7	P_8	D_9	D_{10}	D_{11}	D_{12}
		0		0	1	0		1	1	1	0

$$P_1 = D_3 \oplus D_5 \oplus D_7 \oplus D_9 \oplus D_{11} = 0 \oplus 0 \oplus 0 \oplus 1 \oplus 1 = 0$$

$$P_2 = D_3 \oplus D_6 \oplus D_7 \oplus D_{10} \oplus D_{11} = 0 \oplus 1 \oplus 0 \oplus 1 \oplus 1 = 1$$

$$P_4 = D_5 \oplus D_6 \oplus D_7 \oplus D_{12} = 0 \oplus 1 \oplus 0 \oplus 0 = 1$$

$$P_8 = D_9 \oplus D_{10} \oplus D_{11} \oplus D_{12} = 1 \oplus 1 \oplus 1 \oplus 0 = 1$$

❖ 해밍코드에서 패리티 비트 생성 과정

비트위치	1	2	3	4	5	6	7	8	9	10	11	12
기호	P_1	P_2	D_3	P_4	D_5	D_6	D_7	P_8	D_9	D_{10}	D_{11}	D_{12}
원본 데이터			0		0	1	0		1	1	1	0
P_1 영역	0		0		0		0		1		1	
P_2 영역		1	0			1	0			1	1	
P_4 영역				1	0	1	0					0
P_8 영역								1	1	1	1	0
생성된 코드	0	1	0	1	0	1	0	1	1	1	1	0

생성된 패리티

❖ 해밍코드에서 패리티 비트 검사 과정

전송된 데이터 : 010111011110

P_1	P_2	D_3	P_4	D_5	D_6	D_7	P_8	D_9	D_{10}	D_{11}	D_{12}
0	1	0	1	1	1	0	1	1	1	1	0

☞ 패리티들을 포함하여 검사

$$P_1 = P_1 \oplus D_3 \oplus D_5 \oplus D_7 \oplus D_9 \oplus D_{11} = 0 \oplus 0 \oplus 1 \oplus 0 \oplus 1 \oplus 1 = 1$$

$$P_2 = P_2 \oplus D_3 \oplus D_6 \oplus D_7 \oplus D_{10} \oplus D_{11} = 1 \oplus 0 \oplus 1 \oplus 0 \oplus 1 \oplus 1 = 0$$

$$P_4 = P_4 \oplus D_5 \oplus D_6 \oplus D_7 \oplus D_{12} = 1 \oplus 1 \oplus 1 \oplus 0 \oplus 0 = 1$$

$$P_8 = P_8 \oplus D_9 \oplus D_{10} \oplus D_{11} \oplus D_{12} = 1 \oplus 1 \oplus 1 \oplus 1 \oplus 0 = 0$$

- 검사된 패리티를 $P_8\,P_4\,P_2\,P_1$ 순서대로 정렬
- 모든 패리티가 0이면 에러 없음
- 하나라도 1이 있으면 에러 발생: 결과가 0101이므로 에러 있음
- 0101을 10진수로 바꾸면 5이며, 수신된 데이터에서 앞에서 5번째 비트 010111011110에 에러가 발생한 것이므로 010101011110으로 바꾸어 주면 에러가 정정된다.

❖ 해밍코드에서 에러가 발생한 경우 교정

비트위치		1	2	3	4	5	6	7	8	9	10	11	12
기호		P_1	P_2	D_3	P_4	D_5	D_6	D_7	P_8	D_9	D_{10}	D_{11}	D_{12}
Error 해밍코드		0	1	0	1	1	1	0	1	1	1	1	0
P_1 계산	1	0		0		1		0		1		1	
P_2 계산	0		1	0			1	0			1	1	
P_4 계산	1				1	1	1	0					0
P_8 계산	0								1	1	1	1	0

$P_8P_4P_2P_1$ =0101= ***5*** : 5**번째 비트에 에러 발생**, 1 → 0**으로 교정**

예제 3-8 다음 해밍코드 중 에러가 있는지 검사하여라.

1 1 1 1 0 1 0 0 1 0 1 0

풀이

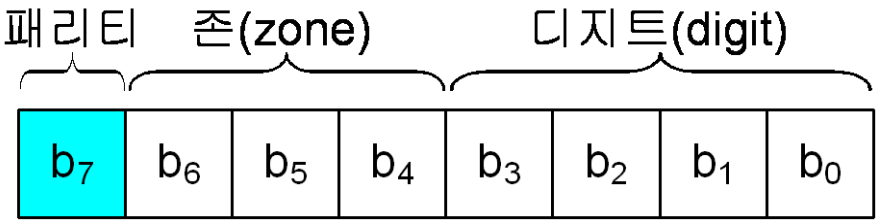
비트 위치		1	2	3	4	5	6	7	8	9	10	11	12
기호		P_1	P_2	D_3	P_4	D_5	D_6	D_7	P_8	D_9	D_{10}	D_{11}	D_{12}
해밍코드		1	1	1	1	0	1	0	0	1	0	1	0
P_1 계산	0	1		1		0		0		1		1	
P_2 계산	0		1	1			1	0			0	1	
P_4 계산	0				1	0	1	0					0
P_8 계산	0								0	1	0	1	0

$P_8 P_4 P_2 P_1 = 0000$ 이므로 에러 없음

End of Example

ASCII (American Standard Code for Information Interchange) 코드

- 미국 국립 표준 연구소(ANSI)가 제정한 정보 교환용 미국 표준 코드
- 128가지의 문자를 표현 가능



❖ ASCII 코드의 구성

parity	zone bit			digit bit			
7	6	5	4	3	2	1	0
C	1	0	0	영문자 A~O(0001~1111)			
	1	0	1	영문자 P~Z(0000~1010)			
	0	1	1	숫자 0~9(0000~1001)			

❖ 표준 ASCII 코드표

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

❖ 확장 ASCII 코드표

표준 BCD 코드

EBCDIC(Extended Binary Coded Decimal Interchange Code) 코드

- 대형 컴퓨터와 IBM 계열 컴퓨터에서 많이 사용되고 있는 8비트 코드

유니코드(Unicode)

- 플랫폼, 프로그램, 언어에 상관없이 모든 문자에 대해 고유 번호를 제공
- ASCII 코드의 한계성을 극복하기 위하여 개발된 인터넷 시대의 표준

한글코드

- 조합형

조합형

상위 바이트								하위 바이트							
1															
	초성					중성					종성				

- 완성형

1987년 정부가 한국표준으로 정한 것으로 가장 많이 사용되는 한글 음절을 2 바이트의 2진수와 1 대 1로 대응하여 표현하는 방법