

Binary Representation

- The basis of all digital data is binary representation.
- Binary - means 'two'
 - 1, 0
 - True, False
 - Hot, Cold
 - On, Off
- We must be able to handle more than just values for real world problems
 - 1, 0, 56
 - True, False, Maybe
 - Hot, Cold, Luke Warm, Cool
 - On, Off, Leaky

Binary Codes

- One Binary Digit (one bit) can take on values 0, 1. We can represent TWO values:
(0 = hot, 1 = cold), (1 = True, 0 = False), (1 = on, 0 = off)
- Two Binary digits (two bits) can take on values of 00, 01, 10, 11. We can represent FOUR values:
(00 = hot, 01 = warm, 10 = cool, 11 = cold)
- Three Binary digits (three bits) can take on values of 000, 001, 010, 011, 100, 101, 110, 111. We can represent 8 values 000 = Black, 001 = Red, 010 = Pink, 011 = Yellow, 100 = Brown, 101 = Blue, 110 = Green, 111 = White.

Binary Codes (cont.)

- N bits (or N binary Digits) can represent 2^N different values.
- For example, 4 bits can represent 2^4 or 16 different values
- N bits can take on unsigned decimal values from 0 to 2^N-1 .
- Codes usually given in tabular form.

000	black
001	red
010	pink
011	yellow
100	brown
101	blue
110	green
111	white

Binary Coded Decimal (BCD)

- Method to Represent Decimal Digits Using Binary
- Each Digit {0,1,2,3,4,5,6,7,8,9} Represented by a 4-bit String

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

- Illegal Bit Strings:
1010, 1011, 1100, 1101, 1110, 1111

Binary Coded Decimal (BCD)

- Example: $(234.98)_{10}$
0010 0011 0100 . 1001 1000
- Not Binary!
 $(234.98)_{10} = (11101010.11111010\dots)_2$
- Can Represent Exactly Using Bits
- Not exactly a Number System – Can't Always Add Directly

0001 0100	$(14)_{10}$	1001	$(9)_{10}$
<u>0010 0100</u>	<u>$(24)_{10}$</u>	<u>1000</u>	<u>$(8)_{10}$</u>
0011 1000	$(38)_{10}$	10001	$(11)_{10}$

Binary Coded Decimal (BCD)

Addition Correction

- Propagate All Digit Carries
- Add $(6)_{10}$ or $(0110)_2$ to All Illegal Digits
- Propagate all Generated Carries

		1000 0001 0100	$(814)_{10}$
		<u>0101 0111 0110</u>	<u>$(576)_{10}$</u>
1001	$(9)_{10}$	1101 1000 1010	$(1?8?)_{10}$
<u>1000</u>	<u>$(8)_{10}$</u>	<u>0110 0000 0110</u>	<u>$(606)_{10}$</u>
10001	$(11)_{10}$	1 0011 1000 0000	$(1380)_{10}$
<u>0110</u>	<u>$(6)_{10}$</u>	1	<u>$(0010)_{10}$</u>
1 0111	$(17)_{10}$	<u>1 0011 1001 0000</u>	<u>$(1390)_{10}$</u>

Other Decimal Codes

- Weighted Codes: Each Bit Weighted by Value that is Different from 2^n
- Excess- k Codes: Each Digit has a Constant Value k Added to it
- Each of These Variations has 6 illegal Bit Strings (4 bits each)
- Like BCD these are Codes not Number Systems and Require Special Rules for Arithmetic

Other Decimal Codes

Decimal digit	BCD (8421)	2421	Excess-3	(8)(4)(-2)(-1)
0	0000	0000	0011	0000
1	0001	0001	0100	0111
2	0010	0010	0101	0110
3	0011	0011	0110	0101
4	0100	0100	0111	0100
5	0101	1011	1000	1011
6	0110	1100	1001	1010
7	0111	1101	1010	1001
8	1000	1110	1011	1000
9	1001	1111	1100	1111
Unused bit combinations	1010	0101	0000	0001
	1011	0110	0001	0010
	1100	0111	0010	0011
	1101	1000	1101	1100
	1110	1001	1110	1101
	1111	1010	1111	1110

Gray Code for decimal Digits

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

A Gray code changes by only 1 bit for adjacent values. This is also called a 'thumbwheel' code because a thumbwheel for choosing a decimal digit can only change to an adjacent value (4 to 5 to 6, etc) with each click of the thumbwheel. This allows the binary output of the thumbwheel to only change one bit at a time; this can help reduce circuit complexity and also reduce signal noise.

Codes for Characters

Also need to represent Characters as digital data. The **ASCII** code (American Standard Code for Information Interchange) is a 7-bit code for Character data. Typically 8 bits are actually used with the 8th bit being zero or used for error detection (parity checking). 8 bits = 1 **Byte**.

$$'A' = (01000001)_2 = (41)_{16}$$

$$'&' = (00100110)_2 = (26)_{16}$$

7 bits can only represent 2^7 different values (128). This enough to represent the Latin alphabet (A-Z, a-z, 0-9, punctuation marks, some symbols like \$), but what about other symbols or other languages?

ASCII

American Standard Code for Information Interchange

Table 2.5 American Standard Code for Information Interchange (ASCII)*

Least Significant Bit	Most Significant Bit							
	0	1	2	3	4	5	6	7
	0000	0001	0010	0011	0100	0101	0110	0111
0 0000	NUL	DLE	SP	0	@	P	'	p
1 0001	SOH	DC1	!	1	A	Q	a	q
2 0010	STX	DC2	"	2	B	R	b	r
3 0011	ETX	DC3	#	3	C	S	c	s
4 0100	EOT	DC4	\$	4	D	T	d	t
5 0101	ENQ	NAK	%	5	E	U	e	u
6 0110	ACK	SYN	&	6	F	V	f	v
7 0111	BEL	ETB	'	7	G	W	g	w
8 1000	BS	CAN	(8	H	X	h	x
9 1001	HT	EM)	9	I	Y	i	y
A 1010	LF	SUB	*	:	J	Z	j	z
B 1011	VT	ESC	+	;	K	[k	{
C 1100	FF	FS	,	<	L	\	l	
D 1101	CR	GS	-	=	M]	m	}
E 1110	SO	RS	.	>	N	^	n	~
F 1111	SI	US	/	?	O	_	o	DEL

*Bit 7 of the code is assumed to be 0.

Source: J. Uffenbeck, *Microcomputers and Microprocessors: The 8080, 8085, and Z-80*, Prentice Hall, Englewood Cliffs, N.J., 1985.

UNICODE

UNICODE is a 16-bit code for representing alphanumeric data. With 16 bits, can represent 2^{16} or 65536 different symbols.

16 bits = 2 Bytes per character.

0041-005A A-Z

0061-4007A a-z

Some other alphabet/symbol ranges

3400-3d2d Korean Hangul Symbols

3040-318F Hiranga, Katakana, Bopomofo, Hangul

4E00-9FFF Han (Chinese, Japanese, Korean)

UNICODE used by Web browsers, Java, most software these days.

Parity

- Append Extra Bit that Cause Total Number of Ones to be
 - Even: Even Parity
 - Odd: Odd Parity

EXAMPLE:

ASCII 'A' = $(100\ 0001)_2 = (41)_{16}$

Even Parity: $(0100\ 0001)_2 = (41)_{16}$

Odd Parity: $(1100\ 0001)_2 = (C1)_{16}$

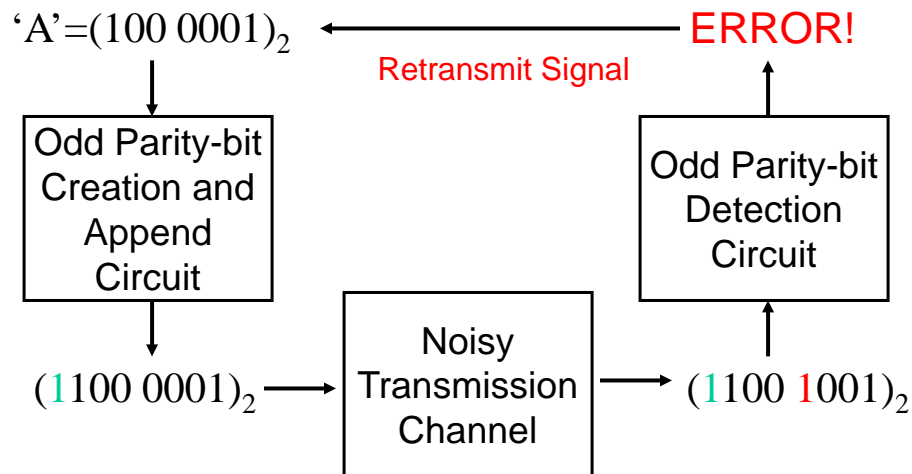
ASCII 'T' = $(101\ 0100)_2 = (54)_{16}$

Even Parity: $(1101\ 0100)_2 = (D4)_{16}$

Odd Parity: $(0101\ 0100)_2 = (54)_{16}$

Parity

- Allows For Single-bit Error Detection



Parity

- Allows For **Single-bit** Error Detection

'A' = $(100\ 0001)_2$

Even Parity-bit
Creation and
Append
Circuit

$(0100\ 0001)_2$

Noisy
Transmission
Channel

NO ERROR!

Received OK Signal

Odd Parity-bit
Detection
Circuit

$(0100\ 0001)_2$