ENGG2020 Digital Logic and Systems

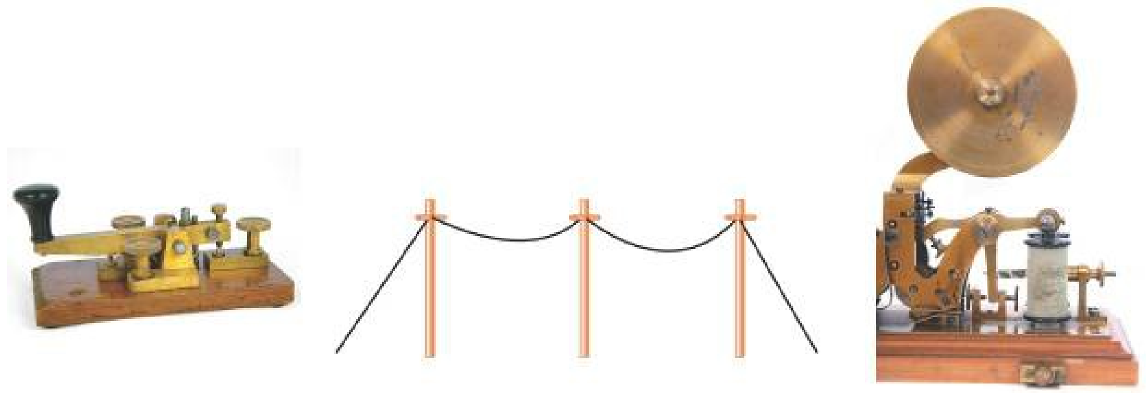
Chapter 1: Digital Number Systems and Representations

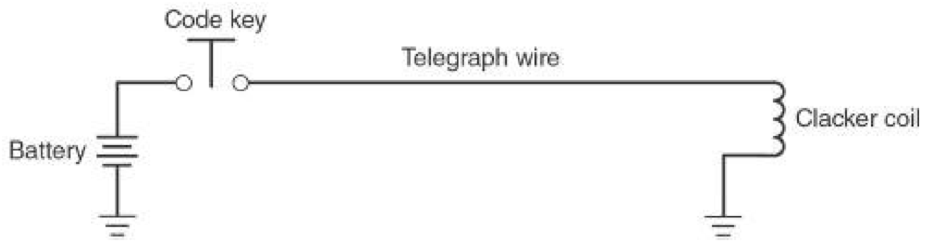
The Chinese University of Hong Kong

Digital and Analog Systems

A large part of the worldwide telecommunications system falls in the category of "digital systems."

It started as a simple digital system that used only two states to represent information.





A telegraph system consisted of a battery, a code key (normally open, momentary contact switch), a telegraph wire, and an electromagnetic "clacker."

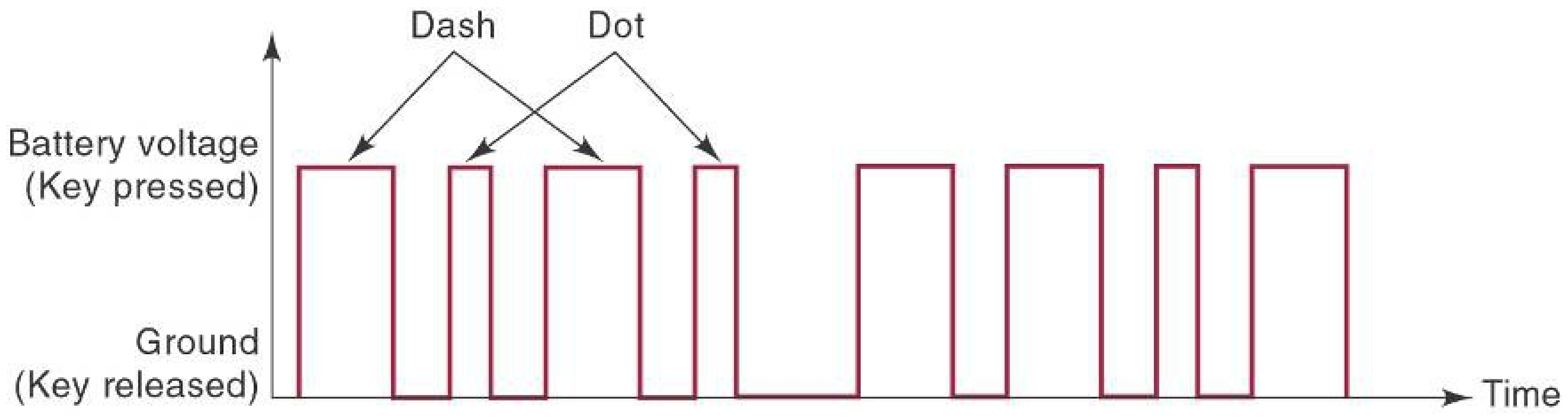
Digital and Analog Systems

The telegraph system used two distinct "symbols" to transmit any word or number.

Short & long electric pulses, the dots & dashes of Morse code—a *digital representation* of information.

The electric signal is either ***on*** or ***off*** at all times.

A timing diagram shows which state (1 or 0) the system is in at any point in time.



Digital and Analog Systems

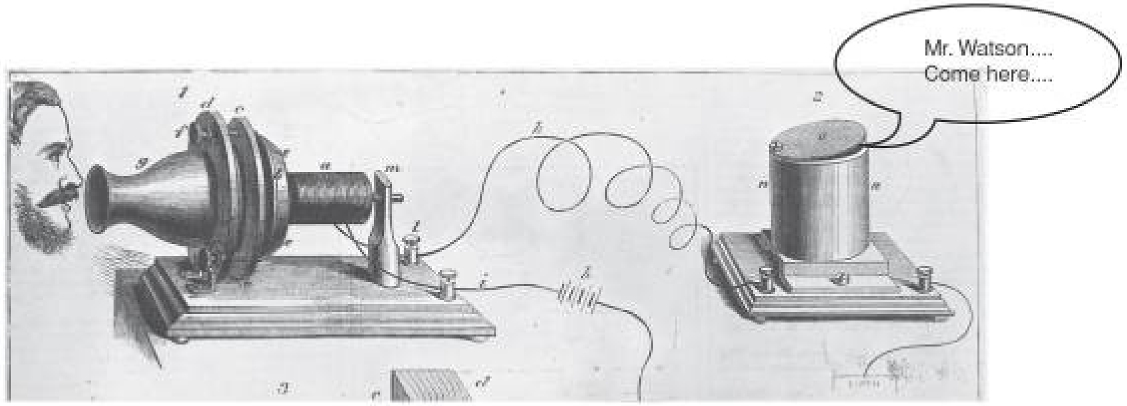
Digital Representation—varies in discrete (separate) steps.

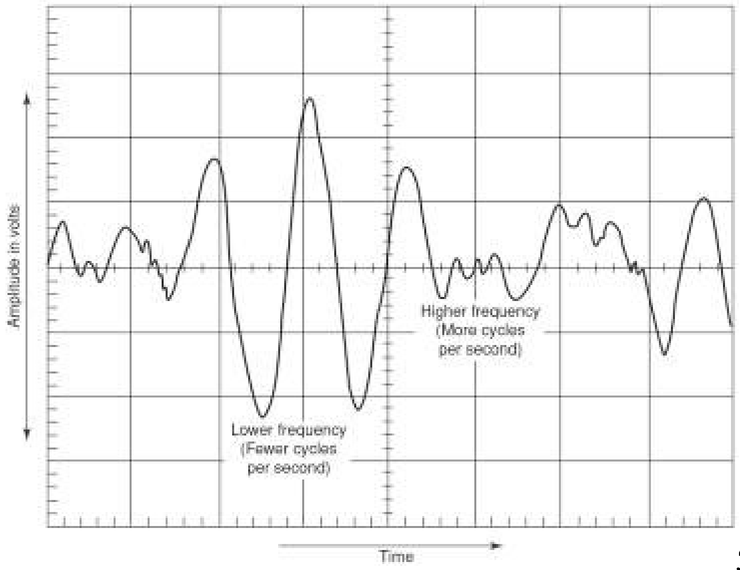
- Passing time is shown as a change in the display on a digital clock at one minute intervals.
- A change in temperature is shown on a digital display only when the temperature changes at least one degree.

Digital and Analog Systems

Analog Representation—a continuously variable, proportional indicator

- Sound through a microphone causes voltage changes.
- Automobile speedometer changes with speed.
- Mercury thermometer varies over a range of values with temperature.





Digital and Analog Systems

A digital system is a combination of devices designed to manipulate logical information or physical quantities represented in digital form.

Quantities can take on only discrete values.

An analog system manipulates physical quantities represented in analog form.

Quantities can vary over a continuous range of values.

Digital and Analog Systems

Advantages of digital

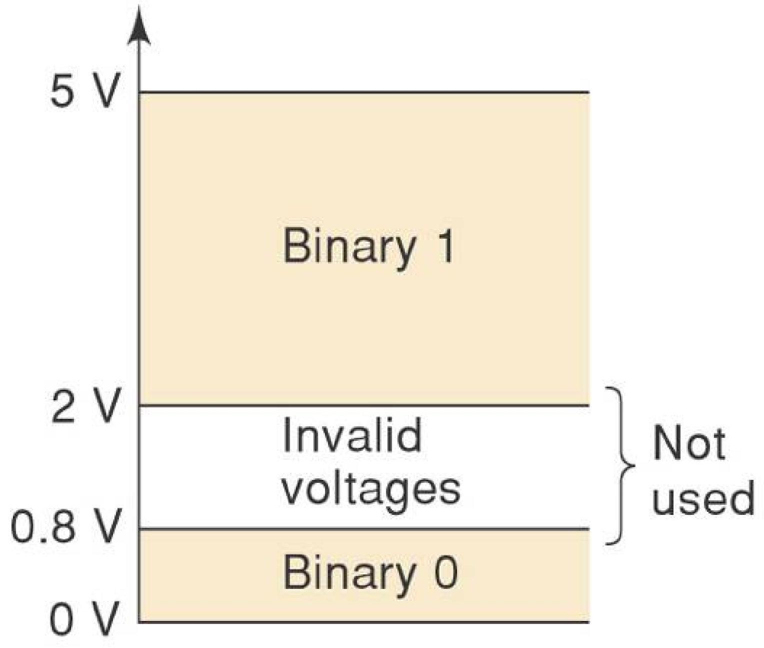
- Ease of design
- Well suited for storing information.
- Accuracy and precision are easier to maintain.
- Programmable operation.
- Less affected by noise.
- Ease of fabrication on IC chips

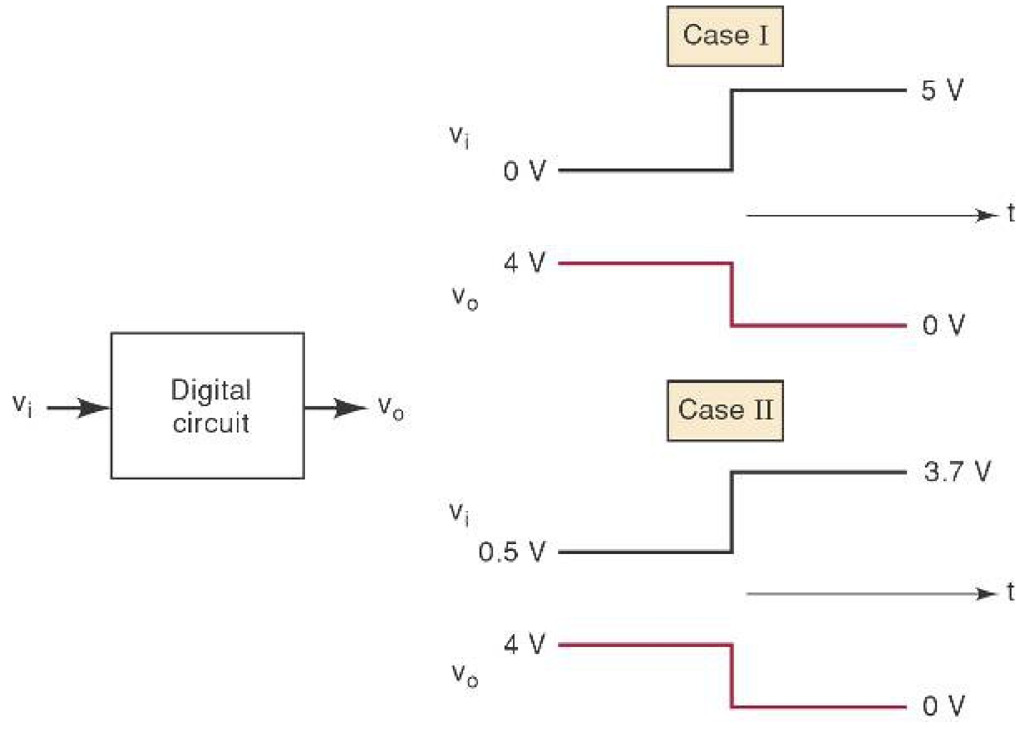
Digital and Analog Systems

Typical representation of the two states of a digital signal.

A *higher* range of voltages represent a valid 1 and a *lower* range of voltages represent a valid 0.

HIGH and LOW are often used to describe the states of a digital system —instead of “1” and “0”





Numerical Representations

Understanding digital systems requires an understanding of the decimal, binary, octal, and hexadecimal numbering systems.

- **Decimal** – 10 symbols (base 10)
- **Hexadecimal** – 16 symbols (base 16)
- **Octal** – 8 symbols (base 8)
- **Binary** – 2 symbols (base 2)

Numerical Representations

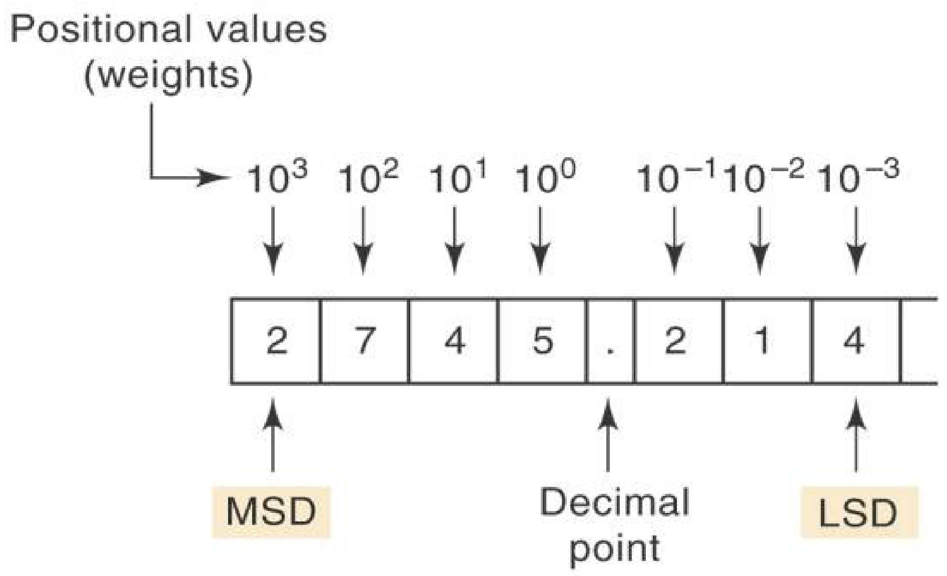
The Decimal (base 10) System

10 symbols: 0, 1, 2, 3, 4, 5, 6 , 7, 8, 9.

Each number is a *digit*.

Most significant digit (MSD) & least significant digit (LSD).

Positional value may be stated as a digit multiplied by a power of 10.



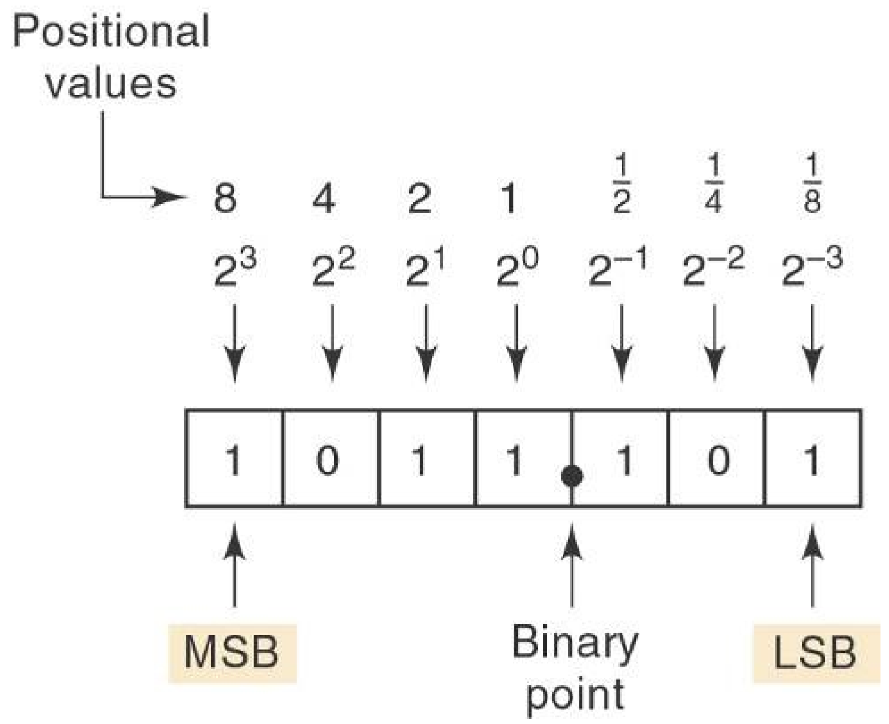
Numerical Representations

The Binary (base 2) System

2 symbols: 0,1

Lends itself to electronic circuit design since only two different voltage levels are required.

Positional value may be stated as a digit multiplied by a power of 2.



Base Conversion: Binary to Decimal

Convert binary to decimal by summing the positions that contain a 1:

$$\begin{aligned} &1 \quad 1 \quad 0 \quad 1 \quad 1_2 \\ &2^4 + 2^3 + 0 + 2^1 + 2^0 = 16 + 8 + 2 + 1 \\ &= 27_{10} \end{aligned}$$

$$\begin{aligned} &1 \quad 0 \quad 1 \quad 1 \quad 0 \quad 1 \quad 0 \quad 1_2 = \\ &2^7 + 0 + 2^5 + 2^4 + 0 + 2^2 + 0 + 2^0 = 181_{10} \end{aligned}$$

Base Conversion: Decimal to Binary

Let us reverse the process

Note that all positions must be accounted for

$$\begin{aligned} 45_{10} &= 32 + 8 + 4 + 1 = 2^5 + 0 + 2^3 + 2^2 + 0 + 2^0 \\ &= 1 \quad 0 \quad 1 \quad 1 \quad 0 \quad 1_2 \end{aligned}$$

$$\begin{aligned} 76_{10} &= 64 + 8 + 4 = 2^6 + 0 + 0 + 2^3 + 2^2 + 0 + 0 \\ &= 1 \quad 0 \quad 0 \quad 1 \quad 1 \quad 0 \quad 0_2 \end{aligned}$$

Base Conversion: Decimal to Binary

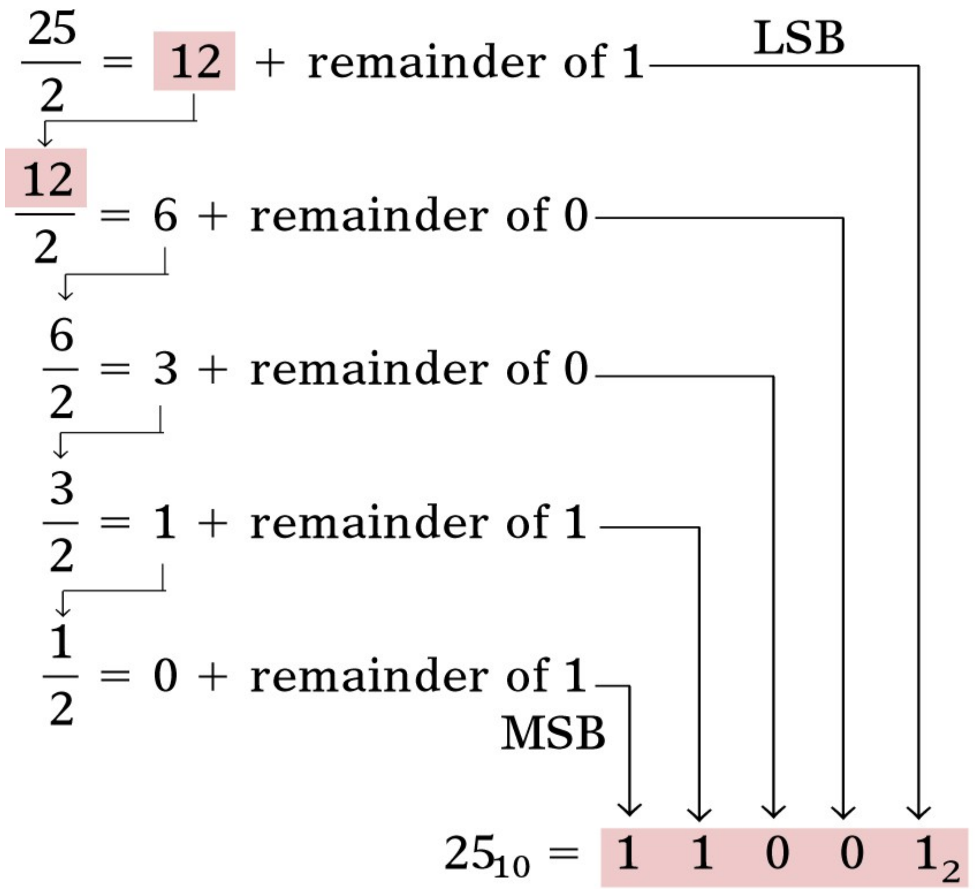
Repeated Division

Divide the decimal number by 2.

Write the remainder after each division until a quotient of zero is obtained.

The first remainder is the LSB.

The last is the MSB.



$\frac{25}{2} = 12 + \text{remainder of } 1$
LSB
$\frac{12}{2} = 6 + \text{remainder of } 0$
$\frac{6}{2} = 3 + \text{remainder of } 0$
$\frac{3}{2} = 1 + \text{remainder of } 1$
$\frac{1}{2} = 0 + \text{remainder of } 1$
MSB
$25_{10} = 1\ 1\ 0\ 0\ 1_2$

Base Conversion: Decimal to Binary

Convert 37_{10} to binary

$$\frac{37}{2} = 18.5 \longrightarrow \text{remainder of 1 (LSB)}$$

$$\frac{18}{2} = 9.0 \longrightarrow 0$$

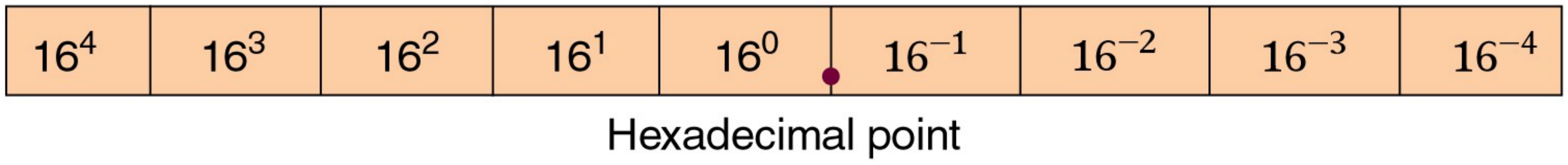
$$\frac{9}{2} = 4.5 \longrightarrow 1$$

$$\frac{4}{2} = 2.0 \longrightarrow 0$$

$$\frac{2}{2} = 1.0 \longrightarrow 0$$

$$\frac{1}{2} = 0.5 \longrightarrow 1 \text{ (MSB)}$$

Base Conversion: Hexadecimal to Decimal



Hexadecimal allows convenient handling of long binary strings, using groups of 4 bits—Base 16

16 possible symbols: 0-9 and A-F

Hexadecimal	Decimal	Binary
0	0	0000
1	1	0001
2	2	0010
3	3	0011
4	4	0100
5	5	0101
6	6	0110
7	7	0111
8	8	1000
9	9	1001
A	10	1010
B	11	1011
C	12	1100
D	13	1101
E	14	1110
F	15	1111

Base Conversion: Hexadecimal to Decimal

Convert from hex to decimal by multiplying each hex digit by its positional weight.

$$\begin{aligned} 356_{16} &= 3 \times 16^2 + 5 \times 16^1 + 6 \times 16^0 \\ &= 768 + 80 + 6 \\ &= 854_{10} \end{aligned}$$

In a 2nd example, the value 10 was substituted for A and 15 substituted for F.

$$\begin{aligned} 2AF_{16} &= 2 \times 16^2 + 10 \times 16^1 + 15 \times 16^0 \\ &= 512 + 160 + 15 \\ &= 687_{10} \end{aligned}$$

For practice, verify that $1BC2_{16}$ is equal to 7106_{10}

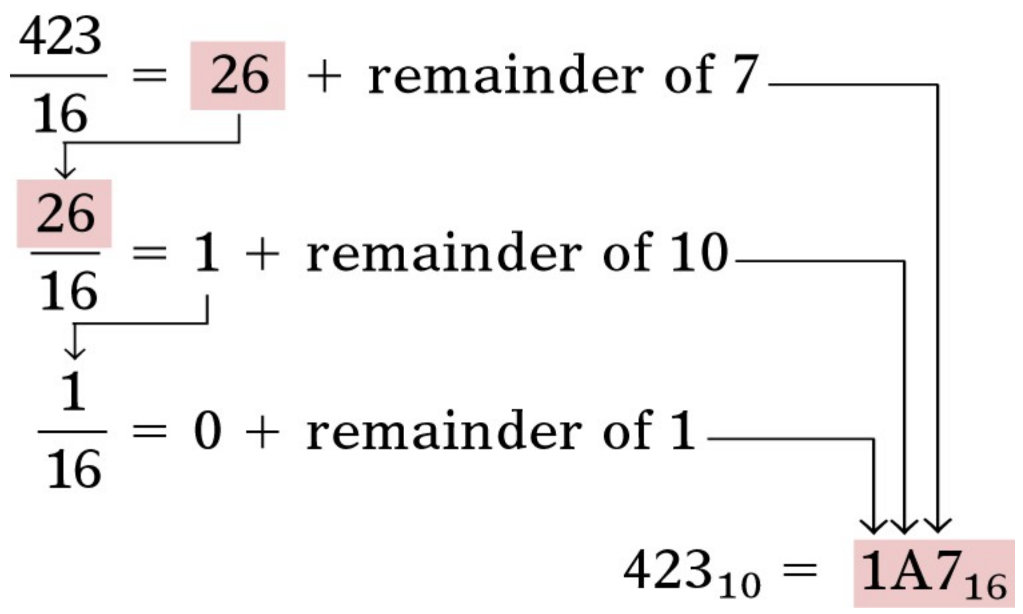
Base Conversion: Decimal to Hexadecimal

Convert from decimal to hex by using the repeated division method used for decimal to binary conversion.

Divide the decimal number by 16

The first remainder is the LSB—the last is the MSB.

Convert 423_{10} to hex:



$$\frac{423}{16} = \boxed{26} + \text{remainder of } 7$$

$$\frac{\boxed{26}}{16} = 1 + \text{remainder of } 10$$

$$\frac{1}{16} = 0 + \text{remainder of } 1$$

$$423_{10} = \boxed{1A7_{16}}$$

Base Conversion: Decimal to Hexadecimal

Convert 214_{10} to hex:

$$\frac{214}{16} = 13 + \text{remainder of } 6$$

$$\frac{13}{16} = 0 + \text{remainder of } 13$$

$$214_{10} = D6_{16}$$

Base Conversion: Hexadecimal to Binary

Convert hex 9F2 to binary

Leading zeros can be added to the left of the MSB to fill out the last group

$$
\begin{array}{rcccc}
9F2_{16} = & 9 & F & 2 \\
 & \downarrow & \downarrow & \downarrow \\
 = & 1\ 0\ 0\ 1 & 1\ 1\ 1\ 1 & 0\ 0\ 1\ 0 \\
 = & \multicolumn{3}{l}{100111110010_2}
\end{array}
$$

Hex	Binary
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001
A	1010
B	1011
C	1100
D	1101
E	1110
F	1111

For practice, verify that $BA6_{16} = 101110100110_2$

Base Conversion: Binary to Hexadecimal

Convert from binary to hex by grouping bits in four starting with the LSB.

Each group is then converted to the hex equivalent

The binary number is grouped into groups of four bits & each is converted to its equivalent hex digit.

$$1\,1\,1\,0\,1\,0\,0\,1\,1\,0_2 = \underbrace{0\;0\;1\;1}_{3}\;\underbrace{1\;0\;1\;0}_{A}\;\underbrace{0\;1\;1\;0}_{6}$$

$$= 3A6_{16}$$

For practice, verify that 101011111 2 = $15F_{16}$

Hex	Binary
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001
A	1010
B	1011
C	1100
D	1101
E	1110
F	1111

Binary Coded Decimal

Binary Coded Decimal (BCD) is a widely used way to present decimal numbers in binary form.

Combines features of both decimal and binary systems.

Each digit is converted to a binary equivalent.

BCD is *not* a number system.

A BCD number is *not* the same as a straight binary number.

The primary advantage of BCD is the relative ease of converting to and from decimal.

Binary Coded Decimal

Convert the number 874_{10} to BCD:

Each decimal digit is represented using 4 bits.

Each 4-bit group can never be greater than 9.

$$\begin{tikzcd} 8 \arrow[d] & 7 \arrow[d] & 4 \arrow[d] & \text{(decimal)} \\ 1000 & 0111 & 0100 & \text{(BCD)} \end{tikzcd}$$

Reverse the process to convert BCD to decimal.

$$\begin{tikzcd} 9 \arrow[d] & 4 \arrow[d] & 3 \arrow[d] & \text{(decimal)} \\ 1001 & 0100 & 0011 & \text{(BCD)} \end{tikzcd}$$

Binary Coded Decimal

Convert 0110100000111001 (BCD) to its decimal equivalent.

$$\underbrace{0110}_{6}\ \underbrace{1000}_{8}\ \underbrace{0011}_{3}\ \underbrace{1001}_{9}$$

Convert BCD 011111000001 to its decimal equivalent.

$$\underbrace{0111}_{7}\ \underset{\downarrow}{1100}\ \underbrace{0001}_{1}$$

The forbidden group represents an error in the BCD number.

Binary Addition and Subtraction

Binary numbers are added like decimal numbers.

In decimal, when numbers sum more than 9, a carry results.

In binary when numbers sum more than 1, a carry takes place.

Binary subtraction is performed just like the subtraction of decimal numbers.

$$0 - 0 = 0$$
$$1 - 1 = 0$$
$$1 - 0 = 1$$
$$0 - 1 \Rightarrow \text{needs to borrow} \Rightarrow 10 - 1 = 1$$

Binary Addition and Subtraction

Addition: Bitwise (digit wise) right to left addition

	0 1 1 0 0	1 0 1 1 0
	+ 1 0 0 0 1	+ 1 0 1 1 1
Carries	0 0 0 0 0	1 0 1 1 0 0
Sum	1 1 1 0 1	1 0 1 1 0 1

Binary Addition and Subtraction

Example: 100 1101 – 1 0111

	6	5	4	3	2	1	0	Column
	1	0	0	1	1	0	1	**Minuend**
-			1	0	1	1	1	**Subtrahend**
		-1		0	0			**M' = Minuend – Borrow from rhs**
	0	1	10	0	10	10		**M * = M' + Borrow from lhs**
	0	1	1	0	1	1	0	**M* - Subtrahend**

Binary Multiplication and Division

Example: $(10111)_2 * (1010)_2$

			1	**0**	**0**	**1**	**1**	**Multiplicand**
*				**1**	**0**	**1**	**0**	**Multiplier**
			0	**0**	**0**	**0**	**0**	
		1	**0**	**1**	**1**	**1**		
	0	**0**	**0**	**0**	**0**			
1	**0**	**1**	**1**	**1**				
1	**1**	**1**	**0**	**0**	**1**	**1**	**0**	**Product**

Binary Multiplication and Division

Example: $(1110111)_2 / (1001)_2$

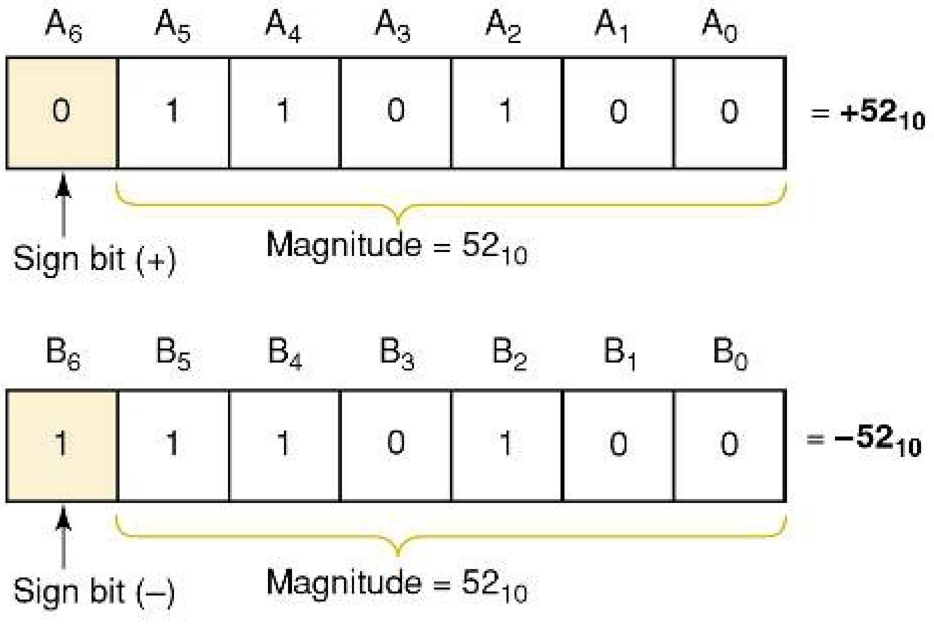
```
                              1  1  0  1   Quotient
                     ---------------------
Divisor   1  0  0  1 | 1  1  1  0  1  1  1   Dividend
                       1  0  0  1
                     ---------------------
                          1  0  1  1
                          1  0  0  1
                     ---------------------
                                1  0  1  1
                                1  0  0  1
                     ---------------------
                                      1  0   Remainder
```

Representing Signed Numbers

Sign and magnitude representation of binary number

A sign bit of 0 indicates a positive number.

A sign bit of 1 indicates a negative number.



Representing Signed Numbers

The **2’s complement system** is the most commonly used way to represent signed numbers.

To change a binary number to 2’s complement it must first be changed to **1’s complement**.

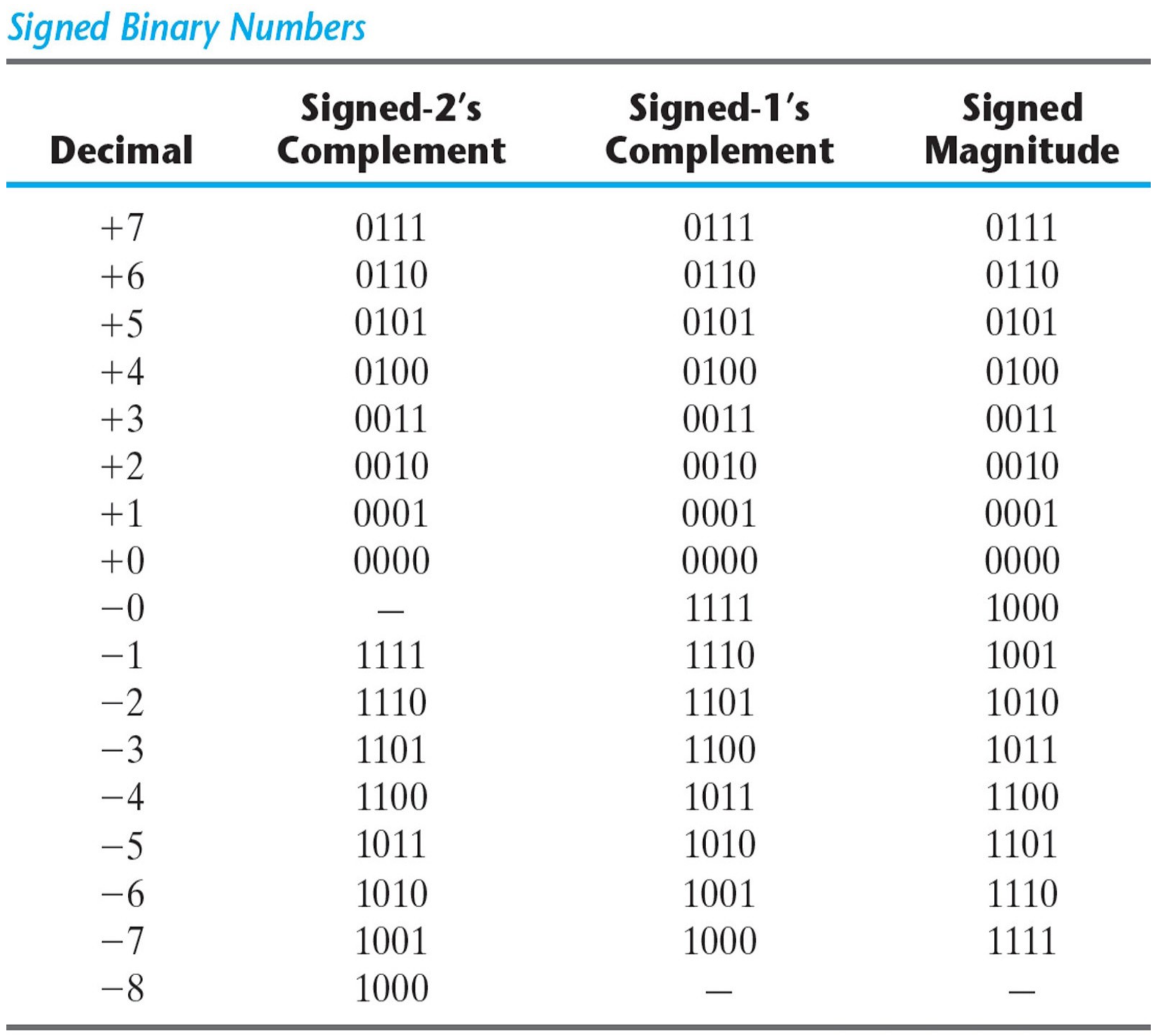
- Change each bit to its complement (opposite).
- To convert 1’s complement to 2’s complement, add 1 to the 1’s complement.

A number is negated when converted to the opposite sign.

- A binary number can be negated by taking the 2’s complement of it.

Representing Signed Numbers

Signed binary numbers



Signed Binary Numbers

Decimal	Signed-2's Complement	Signed-1's Complement	Signed Magnitude
+7	0111	0111	0111
+6	0110	0110	0110
+5	0101	0101	0101
+4	0100	0100	0100
+3	0011	0011	0011
+2	0010	0010	0010
+1	0001	0001	0001
+0	0000	0000	0000
−0	—	1111	1000
−1	1111	1110	1001
−2	1110	1101	1010
−3	1101	1100	1011
−4	1100	1011	1100
−5	1011	1010	1101
−6	1010	1001	1110
−7	1001	1000	1111
−8	1000	—	—

Addition and Subtraction in the 2’s Complement System

The addition operation

- The sign bits are added with the magnitude bits.
- If addition results in a carry of the sign bit, the carry bit is ignored.
- If the result is *positive*, it is in pure binary form.
- If the result is *negative*, it is in 2’s complement form.

Subtraction using the 2’s-complement system actually involves the operation of addition.

- The number subtracted (subtrahend) is negated.
- The result is added to the minuend.
- The answer represents the difference.

Addition and Subtraction in the 2's Complement System

Examples

$$
\begin{array}{rr}
+\ 6 & 00000110 \\
+13 & 00001101 \\
\hline
+19 & 00010011
\end{array}
\qquad\qquad
\begin{array}{rr}
-\ 6 & 11111010 \\
+13 & 00001101 \\
\hline
+\ 7 & 00000111
\end{array}
$$

$$
\begin{array}{rr}
+\ 6 & 00000110 \\
-13 & 11110011 \\
\hline
-\ 7 & 11111001
\end{array}
\qquad\qquad
\begin{array}{rr}
-\ 6 & 11111010 \\
-13 & 11110011 \\
\hline
-19 & 11101101
\end{array}
$$

Addition and Subtraction in the 2’s Complement System

Overflow can occur only when two positive or two negative numbers are being added—which always produces an incorrect result.

- Sum of two positive numbers yields a negative result
- Sum of two negative numbers yields a positive result

Binary-Coded Decimal Addition

Binary-Coded Decimal uses four bits to represent a decimal number, 0-9

Six values (states) are not legitimate BCD in the four bits

Should be care during BCD addition

Six must be added to the result to flip through the unwanted states

Binary-Coded Decimal Addition

Examples

BCD	1	1		
	0001	1000	0100	184
	+0101	0111	0110	+576
Binary sum	0111	10000	1010	
Add 6		0110	0110	
BCD sum	0111	0110	0000	760

Gray Code

The **Gray code** is used in applications where numbers change rapidly.

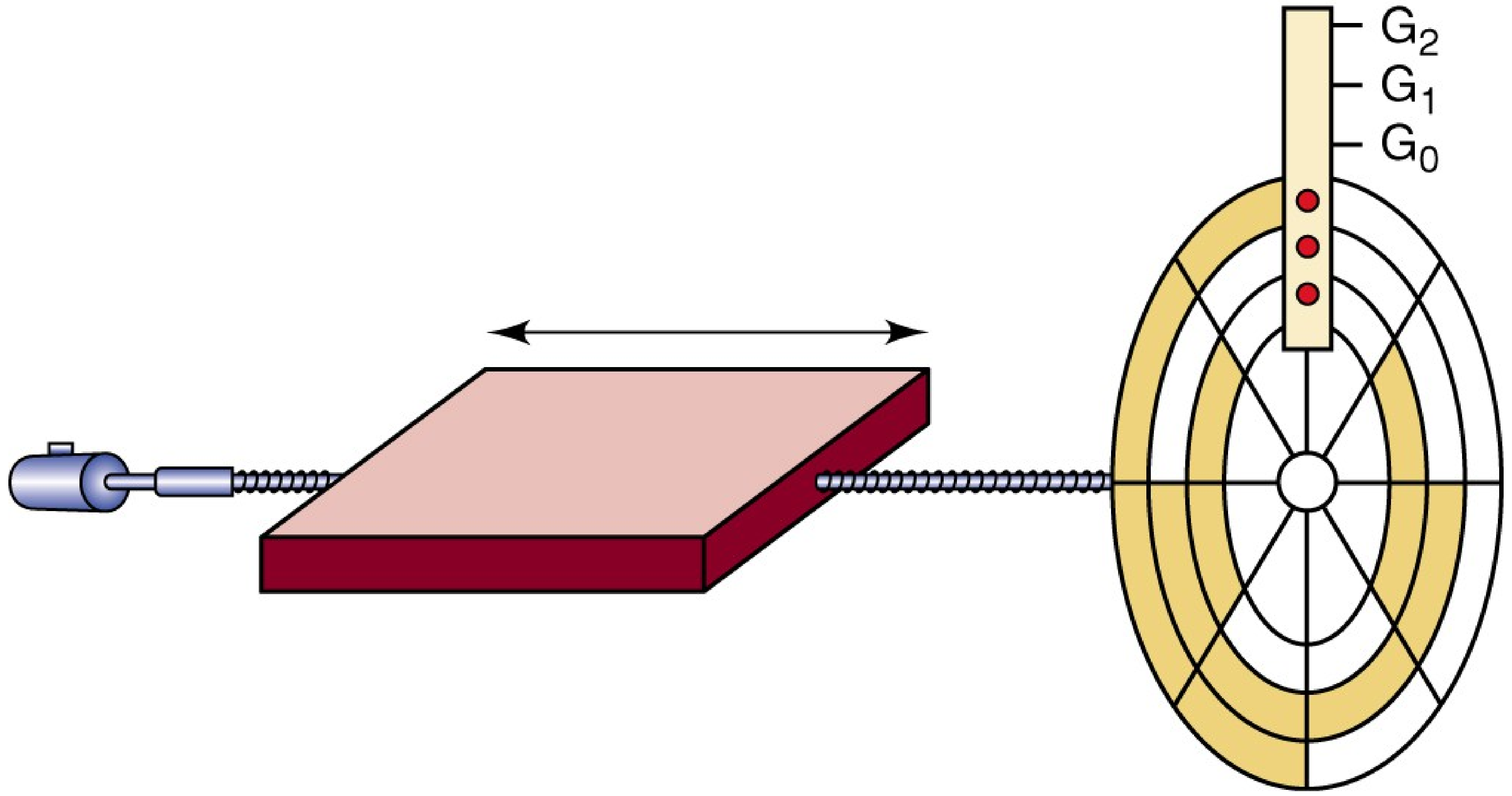
Only one bit changes from each value to the next.

Three bit binary and Gray code equivalents.

B_2	B_1	B_0	G_2	G_1	G_0
0	0	0	0	0	0
0	0	1	0	0	1
0	1	0	0	1	1
0	1	1	0	1	0
1	0	0	1	1	0
1	0	1	1	1	1
1	1	0	1	0	1
1	1	1	1	0	0

Gray Code

An eight-position, three-bit shaft encoder



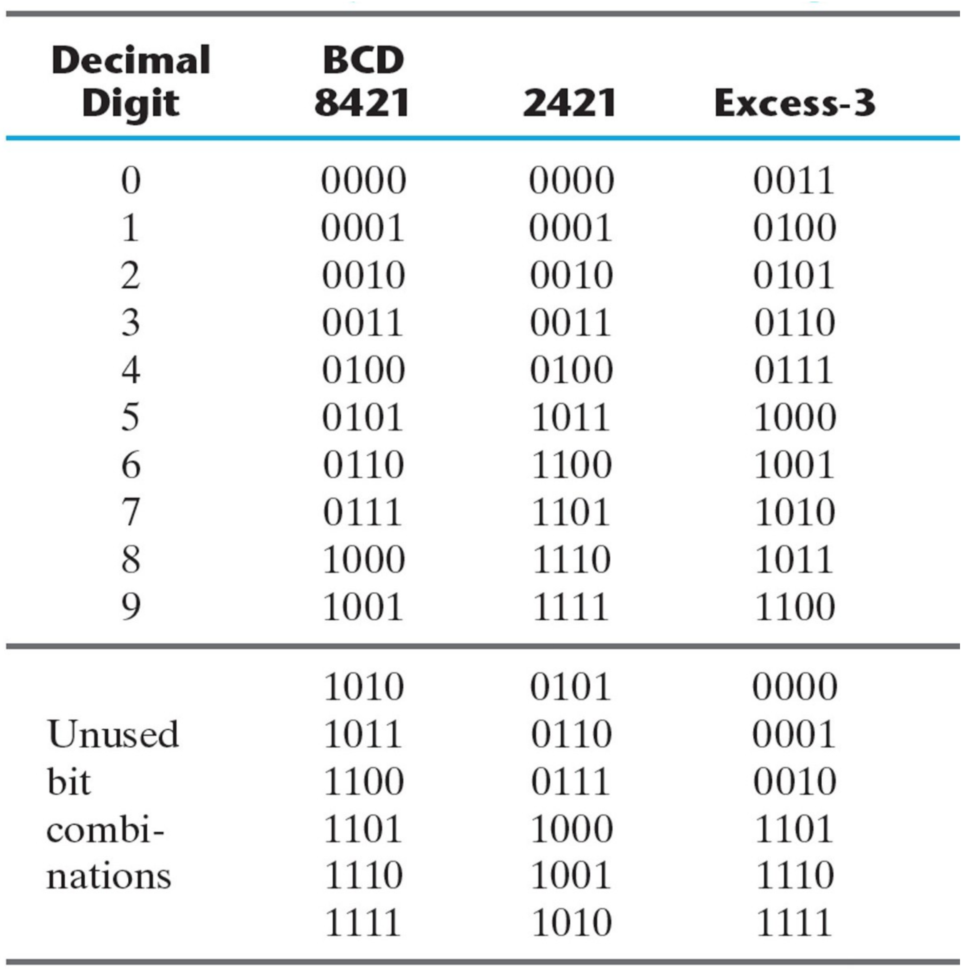
Gray Code

Decimal numbers 1 – 15 in binary, hex, BCD, Gray

Decimal	Binary	Hexadecimal	BCD	GRAY
0	0	0	0000	0000
1	1	1	0001	0001
2	10	2	0010	0011
3	11	3	0011	0010
4	100	4	0100	0110
5	101	5	0101	0111
6	110	6	0110	0101
7	111	7	0111	0100
8	1000	8	1000	1100
9	1001	9	1001	1101
10	1010	A	0001 0000	1111
11	1011	B	0001 0001	1110
12	1100	C	0001 0010	1010
13	1101	D	0001 0011	1011
14	1110	E	0001 0100	1001
15	1111	F	0001 0101	1000

Other Decimal Codes

Excess-3 and weighted codes



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

The Byte, Nibble, and Word

Most microcomputers handle and store binary data and information in groups of eight bits.

- **8 bits = 1 byte**.
- A byte can represent numerous types of data/information.

Binary numbers are often broken into groups of four bits.

- Because a group of **four bits is half as big as a byte**, it was named a **nibble**.

A **word** is a group of bits that represents a certain unit of information.

- **Word size** can be defined as the number of bits in the binary word a digital system operates on.
- PC word size is eight bytes (64 bits).

Alphanumeric Codes

Represents characters and functions found on a computer keyboard.

26 lowercase & 26 uppercase letters, 10 digits, 7 punctuation marks, 20 to 40 other characters.

ASCII – American Standard Code for Information Interchange.

Seven bit code: $2^7 = 128$ possible code groups

Examples of use: transfer information between computers; computers & printers; internal storage.

Alphanumeric Codes

ASCII – American Standard Code for Information Interchange

Character	HEX	Decimal	Character	HEX	Decimal	Character	HEX	Decimal	Character	HEX	Decimal
NUL (null)	0	0	Space	20	32	@	40	64	.	60	96
Start Heading	1	1	!	21	33	A	41	65	a	61	97
Start Text	2	2	“	22	34	B	42	66	b	62	98
End Text	3	3	#	23	35	C	43	67	c	63	99
End Transmit.	4	4	$	24	36	D	44	68	d	64	100
Enquiry	5	5	%	25	37	E	45	69	e	65	101
Acknowlege	6	6	&	26	38	F	46	70	f	66	102
Bell	7	7	`	27	39	G	47	71	g	67	103
Backspace	8	8	(	28	40	H	48	72	h	68	104
Horiz. Tab	9	9	)	29	41	I	49	73	i	69	105
Line Feed	A	10	*	2A	42	J	4A	74	j	6A	106
Vert. Tab	B	11	+	2B	43	K	4B	75	k	6B	107
Form Feed	C	12	,	2C	44	L	4C	76	l	6C	108
Carriage Return	D	13	-	2D	45	M	4D	77	m	6D	109
Shift Out	E	14	.	2E	46	N	4E	78	n	6E	110

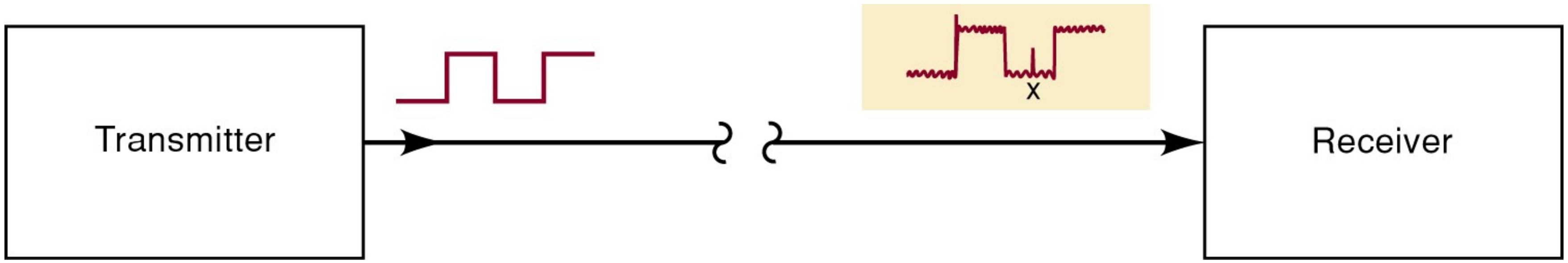
Error Detection

Electrical noise can cause errors during data transmission.

Spurious fluctuations in voltage or current present in all electronic systems.

Many digital systems **employ methods for error detection**—and sometimes correction.

One of the simplest and most widely used schemes for error detection is the **parity method**.



Error Detection

The parity method of error detection requires the addition of an extra bit to a code group.

Called the **parity bit, it can be either a 0 or 1**, depending on the number of 1s in the code group.

There are two parity methods, even and odd.

The transmitter and receiver must “agree” on the type of parity checking used.

Even seems to be used more often.

Error Detection

Even parity method—the total number of bits in a group *including* the parity bit must add up to an *even* number.

The binary group **1 0 1 1** would require the addition of a parity bit **1**, making the group **1 1 0 1 1**.

The parity bit may be added at either end of a group.

Odd parity method—the total number of bits in a group *including* the parity bit must add up to an *odd* number.

The binary group **1 1 1 1** would require the addition of a parity bit **1**, making the group **1 1 1 1 1**.

Error Detection

When ASCII characters are transmitted there must be a way to tell the receiver a new character is coming.

There is often a need to detect errors in the transmission as well.

The method of transfer is called asynchronous data communication.

The parity bit becomes a part of the code word. Adding a parity bit to the seven-bit ASCII code produces an eight-bit code.

Error Detection

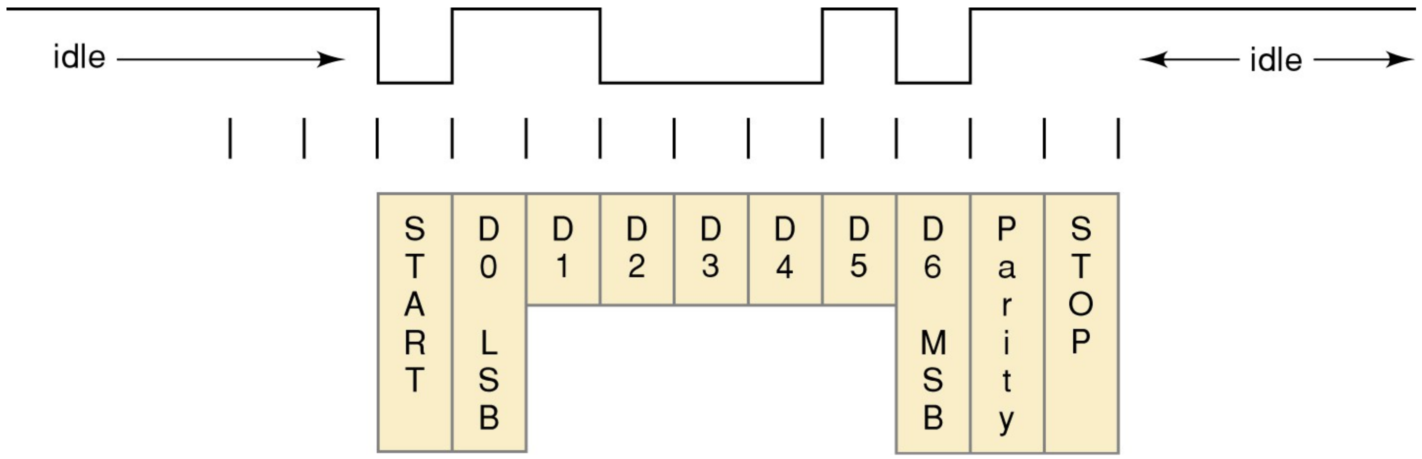
An ASCII character must be “framed” so the receiver knows where the data begins and ends.

The first bit must always be a start bit (logic 0).

ASCII code is sent LSB first and MSB last.

After the MSB, a parity bit is appended to check for transmission errors.

Transmission is ended by sending a stop bit (logic 1).



idle
idle
START
D0 LSB
D1
D2
D3
D4
D5
D6 MSB
Parity
STOP