Academic Year 2022/2023 2nd Semester

COMP123 – 121/122

Data Communications

Basic Data Communications Concepts

What we are going to learn…

- Data codes
  – how do we represent data
- Data transmission and modes
- Modes of data flow

Data Representation

- Information comes in different forms such as text, numbers, images, audio and video
- All of the forms can be represented by bit patterns (101011001…)
- Text and numbers can be represented using ASCII or Unicode
- An image consist of a matrix of pixels, which are assigned a bit pattern

Character Codes

- Different sets of bit patterns have been designed to represent text symbols. Each set is called a code.
- Keyboard converts a character into a series of bits using *character codes*, e.g. Morse code, Baudot code, EBCDIC and ASCII.
- Printer, video display convert binary code to the symbols meaningful to people.

Morse Code

- One of the first character codes developed
- Designed with a telegraph operator in mind
- The operator sent combinations of a short beep
- Unsuitable for computer communications
  - extra time required between transmission of characters

Morse Code

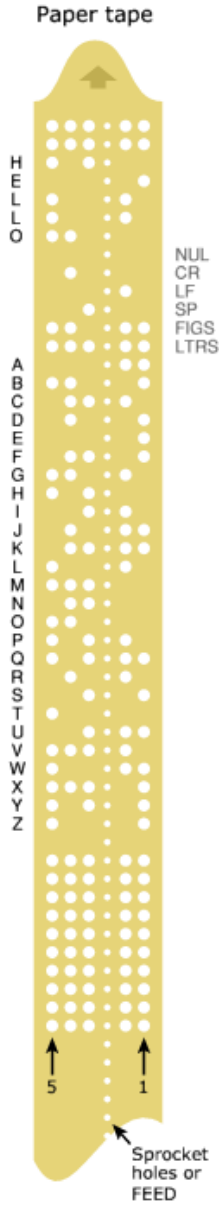
A	•–	N	–•	1	•–––––
B	–•••	O	–––	2	••–––
C	–•–•	P	•––•	3	•••––
D	–••	Q	––•–	4	••••–
E	•	R	•–•	5	•••••
F	••–•	S	•••	6	–••••
G	––•	T	–	7	––•••
H	••••	U	••–	8	–––••
I	••	V	•••–	9	––––•
J	•–––	W	•––	0	–––––
K	–•–	X	–••–	.	•–•–•–
L	•–••	Y	–•––	,	––••––
M	––	Z	––••	?	••––••

Try this Morse Code Translator: http://morsecode.scphillips.com/jtranslator.html

Baudot Code

- One of the first character codes developed with machines
- Five bits for each character
- Special codes are used to shift between upper and lower case
- International Baudot adds a parity bit to check for errors
- Inefficient in switching between upper and lower case

Baudot Code



Character		Data bits				
Lower Case	Upper Case	5	4	3	2	1
A	_	0	0	0	1	1
B	?	1	1	0	0	1
C	:	0	1	1	1	0
D	$	0	1	0	0	1
E	3	0	0	0	0	1
F	!	0	1	1	0	1
G	&	1	1	0	1	0
H	#	1	0	1	0	0
I	8	0	0	1	1	0
J	'	0	1	0	1	1
K	(	0	1	1	1	1
L	)	1	0	0	1	0
M	.	1	1	1	0	0
N	,	0	1	1	0	0
O	9	1	1	0	0	0
P	0	1	0	1	1	0
Q	1	1	0	1	1	1
R	4	0	1	0	1	0
S	BELL	0	0	1	0	1
T	5	1	0	0	0	0
U	7	0	0	1	1	1
V	;	1	1	1	1	0
W	2	1	0	0	1	1
X	/	1	1	1	0	1
Y	6	1	0	1	0	1
Z	"	1	0	0	0	1
Letters (shift to Lower Case column)		1	1	1	1	1
Figures (shift to Upper Case column)		1	1	0	1	1
Space		0	0	1	0	0
Carriage return		0	1	0	0	0
Line feed		0	0	0	1	0
Blank		0	0	0	0	0

EBCDIC

- EBCDIC stands for Extended Binary Coded Decimal Interchange Code
- Eight bit character code developed by IBM, offering 256 different characters
- Includes all uppercase and lowercase letters and the digits 0 – 9
- Also include a large number of special symbols and punctuation marks and a number of control characters

EBCDIC

Bit 4 5 6 7 / 0 1 2 3	0 0 0 0	0 0 0 1	0 0 1 0	0 0 1 1	0 1 0 0	0 1 0 1	0 1 1 0	0 1 1 1	1 0 0 0	1 0 0 1	1 0 1 0	1 0 1 1	1 1 0 0	1 1 0 1	1 1 1 0	1 1 1 1
0 0 0 0	NUL	SOH	STX	ETX	PF	HT	LC	DEL				VT	FF	CR	SO	SI
0 0 0 1	DLE	DC1	DC2	DC3	RES	NL	BS	IL	CAN	EM			IFS	IGS	IRS	IUS
0 0 1 0			FS		BYP	LF	EOB	PRE			SM			ENQ	ACK	BEL
0 0 1 1			SYN		PN	RS	UC	EOT					DC4	NAK		SUB
0 1 0 0	SP										¢	.	<	(	+	|
0 1 0 1	&										!	$	*	)	;	¬
0 1 1 0	-	/									¦	,	%	-	>	?
0 1 1 1										\	:	#	@	'	=	“
1 0 0 0		a	b	c	d	e	f	g	h	i						
1 0 0 1		j	k	l	m	n	o	p	q	r						
1 0 1 0		~	s	t	u	v	w	x	y	z						
1 0 1 1																
1 1 0 0	{	A	B	C	D	E	F	G	H	I						
1 1 0 1	}	J	K	L	M	N	O	P	Q	R						
1 1 1 0			S	T	U	V	W	X	Y	Z						
1 1 1 1	0	1	2	3	4	5	6	7	8	9						□

EBCDIC Coding Example

- Use EBCDIC to encode “Get $12.30”

G	e	t	space
11000111	10000101	10100011	00000000

$	1	2	.
01011011	11110001	11110010	01011100

3	0
11110011	11110000

ASCII

- ASCII stands for American Standard Code for Information Interchange
- Seven-bit code with a single additional parity bit
- Offer 128 different characters
- Extended ASCII is a modified version of ASCII, which was never internationally standardized

ASCII

Bits 7654321	Character	Bits 7654321	Character	Bits 7654321	Character	Bits 7654321	Character
0000000	NUL	0100000	SP	1000000	@	1100000	`
0000001	SOH	0100001	!	1000001	A	1100001	a
0000010	STX	0100010	"	1000010	B	1100010	b
0000011	ETX	0100011	#	1000011	C	1100011	c
0000100	EOT	0100100	$	1000100	D	1100100	d
0000101	ENQ	0100101	%	1000101	E	1100101	e
0000110	ACK	0100110	&	1000110	F	1100110	f
0000111	BEL	0100111	'	1000111	G	1100111	g
0001000	BS	0101000	(	1001000	H	1101000	h
0001001	HT	0101001	)	1001001	I	1101001	i
0001010	LF	0101010	*	1001010	J	1101010	j
0001011	VT	0101011	+	1001011	K	1101011	k
0001100	FF	0101100	,	1001100	L	1101100	1
0001101	CR	0101101	–	1001101	M	1101101	m
0001110	SO	0101110	.	1001110	N	1101110	n
0001111	SI	0101111	/	1001111	O	1101111	o
0010000	DLE	0110000	0	1010000	P	1110000	p
0010001	DC1	0110001	1	1010001	Q	1110001	q
0010010	DC2	0110010	2	1010010	R	1110010	r
0010011	DC3	0110011	3	1010011	S	1110011	s
0010100	DC4	0110100	4	1010100	T	1110100	t
0010101	NAK	0110101	5	1010101	U	1110101	u
0010110	SYN	0110110	6	1010110	V	1110110	v
0010111	ETB	0110111	7	1010111	W	1110111	w
0011000	CAN	0111000	8	1011000	X	1111000	x
0011001	EM	0111001	9	1011001	Y	1111001	y
0011010	SUB	0111010	:	1011010	Z	1111010	z
0011011	ESC	0111011	;	1011011	[	1111011	{
0011100	FS	0111100	<	1011100	\	1111100	|
0011101	GS	0111101	=	1011101	]	1111101	}
0011110	RS	0111110	>	1011110	^	1111110	~
0011111	US	0111111	?	1011111	—	1111111	DEL

ASCII Coding Example

- Use ASCII (without parity) to encode “Get $12.30”

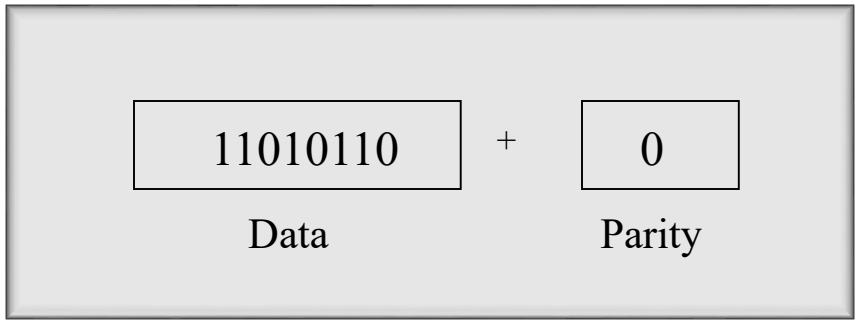
G	e	t	space
1000111	1100101	1110100	0000000

$	1	2	.
0100100	0110001	0110010	0101110

3	0
0110011	0110000

Parity and Bytes

- Parity - an extra bit added to the byte to check for errors (Even/Odd parity)
- Byte - in data communications, a group of bits, usually 8, though sometimes more, or less, depending on the character code.



Odd parity

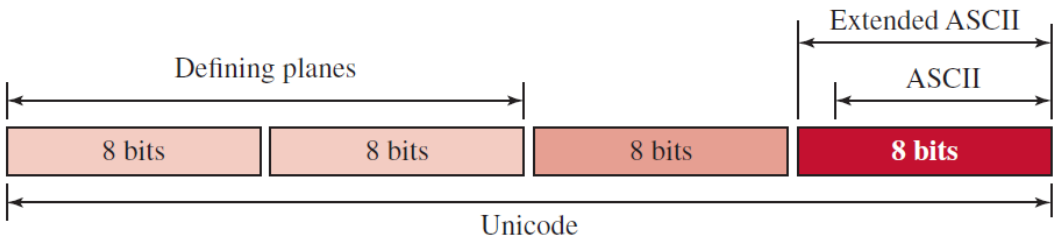
Unicode

- EBCDIC and ASCII cannot represent all the different types of symbols in the English language, e.g. many of the technical symbols used in engineering and mathematics
- EBCDIC and ASCII cannot represent symbols from language other than English
- Unicode is an encoding technique that provides a unique coding value for every character in every language, no matter what the platform.
- Unicode supports more than 110 different code charts (language and symbol sets), e.g. the Greek symbol β has the Unicode value of 0000001110110010 (or 03B2 in Hex)
- Originally a 2-byte character set

Unicode

- Now a 4-byte code fully compatible with ASCII and Extended ASCII
- The ASCII set, called Basic Latin, is Unicode with the most significant 25 bits set to zero.
- Extended ASCII, called Latin-1, is Unicode with the most significant 24 bits set to zero

Figure A.1 *Unicode bytes*



Unicode

- Each character or symbol is defined by a 32-bit number
- The code can define up to 2^{32} (4,294,967,296) characters or symbols
- The notation uses hexadecimal digits in the following format

$$U - XXXXXXXX$$

- The numbering goes from U-00000000 to U-FFFFFFFF
- More details can be found at www.unicode.org

Examples of Unicode Code Charts

C0 Controls and Basic Latin

	000	001	002	003	004	005	006	007
0	NUL 0000	DLE 0010	SP 0020	0 0030	@ 0040	P 0050	` 0060	p 0070
1	SOH 0001	DC1 0011	! 0021	1 0031	A 0041	Q 0051	a 0061	q 0071
2	STX 0002	DC2 0012	" 0022	2 0032	B 0042	R 0052	b 0062	r 0072
3	ETX 0003	DC3 0013	# 0023	3 0033	C 0043	S 0053	c 0063	s 0073
4	EOT 0004	DC4 0014	$ 0024	4 0034	D 0044	T 0054	d 0064	t 0074
5	ENQ 0005	NAK 0015	% 0025	5 0035	E 0045	U 0055	e 0065	u 0075
6	ACK 0006	SYN 0016	& 0026	6 0036	F 0046	V 0056	f 0066	v 0076
7	BEL 0007	ETB 0017	' 0027	7 0037	G 0047	W 0057	g 0067	w 0077
8	BS 0008	CAN 0018	(0028	8 0038	H 0048	X 0058	h 0068	x 0078
9	HT 0009	EM 0019	) 0029	9 0039	I 0049	Y 0059	i 0069	y 0079
A	LF 000A	SUB 001A	* 002A	: 003A	J 004A	Z 005A	j 006A	z 007A
B	VT 000B	ESC 001B	+ 002B	; 003B	K 004B	[005B	k 006B	{ 007B
C	FF 000C	FS 001C	, 002C	< 003C	L 004C	\ 005C	l 006C	| 007C
D	CR 000D	GS 001D	- 002D	= 003D	M 004D	] 005D	m 006D	} 007D
E	SO 000E	RS 001E	. 002E	> 003E	N 004E	^ 005E	n 006E	~ 007E
F	SI 000F	US 001F	/ 002F	? 003F	O 004F	_ 005F	o 006F	DEL 007F

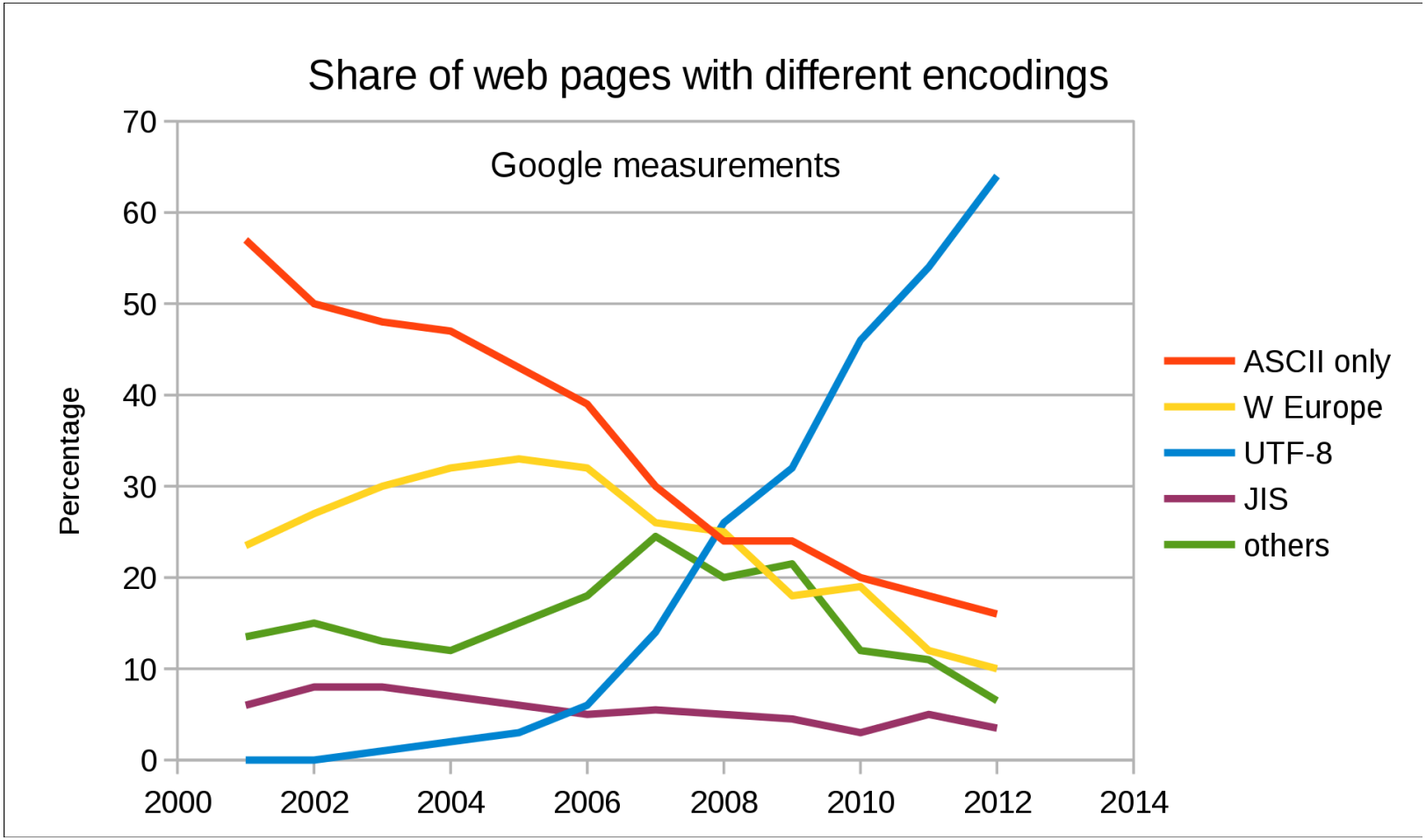
Greek and Coptic

	037	038	039	03A	03B	03C	03D	03E	03F
0	Ϸ 0370		ΐ 0390	Π 03A0	ΰ 03B0	π 03C0	ϐ 03D0	Ϣ 03E0	ϰ 03F0
1	ϸ 0371		A 0391	P 03A1	α 03B1	ρ 03C1	ϑ 03D1	ϣ 03E1	ϱ 03F1
2	Ϲ 0372		B 0392		β 03B2	ς 03C2	ϒ 03D2	Ϥ 03E2	ϲ 03F2
3	Ϳ 0373		Γ 0393	Σ 03A3	γ 03B3	σ 03C3	ϓ 03D3	ϥ 03E3	ϳ 03F3
4	ʹ 0374	΄ 0384	Δ 0394	T 03A4	δ 03B4	τ 03C4	ϔ 03D4	Ϧ 03E4	ϴ 03F4
5	͵ 0375	΅ 0385	E 0395	Y 03A5	ε 03B5	υ 03C5	ϕ 03D5	ϧ 03E5	ϵ 03F5
6	Ͷ 0376	Ά 0386	Z 0396	Φ 03A6	ζ 03B6	φ 03C6	ϖ 03D6	Ϩ 03E6	϶ 03F6
7	ͷ 0377	· 0387	H 0397	X 03A7	η 03B7	χ 03C7	ϗ 03D7	ϩ 03E7	Ϸ 03F7
8		Έ 0388	Θ 0398	Ψ 03A8	θ 03B8	ψ 03C8	Ϙ 03D8	Ϫ 03E8	ϸ 03F8
9		Ή 0389	I 0399	Ω 03A9	ι 03B9	ω 03C9	ϙ 03D9	ϫ 03E9	Ϲ 03F9
A	ͺ 037A	Ί 038A	K 039A	Ϊ 03AA	κ 03BA	ϊ 03CA	Ϛ 03DA	Ϭ 03EA	Ϻ 03FA
B	ͻ 037B		Λ 039B	Ϋ 03AB	λ 03BB	ϋ 03CB	ϛ 03DB	ϭ 03EB	ϻ 03FB
C	ͼ 037C	Ό 038C	M 039C	ά 03AC	μ 03BC	ό 03CC	F 03DC	Ϯ 03EC	ϼ 03FC
D	ͽ 037D		N 039D	έ 03AD	ν 03BD	ύ 03CD	ϝ 03DD	ϭ 03ED	Ͻ 03FD
E	; 037E	Ύ 038E	Ξ 039E	ή 03AE	ξ 03BE	ώ 03CE	Ϟ 03DE	Ϯ 03EE	Ͼ 03FE
F		Ώ 038F	O 039F	ί 03AF	ο 03BF	Ϗ 03CF	ϟ 03DF	ϯ 03EF	Ͽ 03FF

Encoding of Unicode - UTF-8

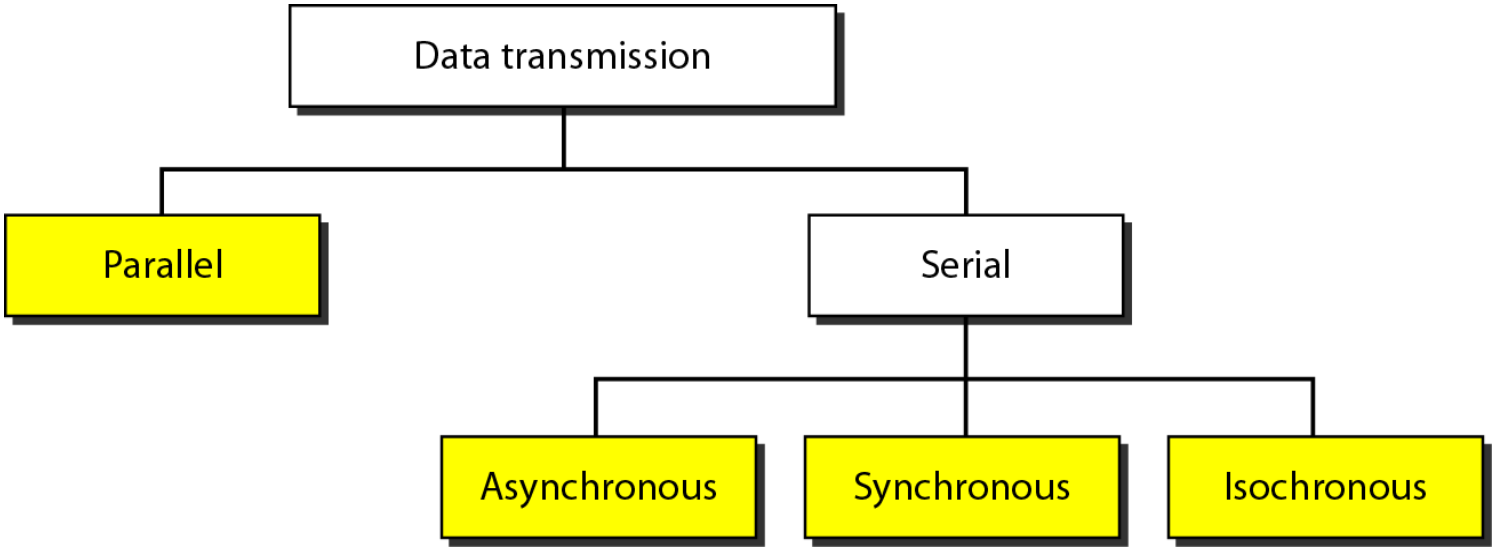
- UTF-8 is a variable-width encoding that can represent every character in the Unicode character set
- UTF-8 has become the dominant character encoding for the World Wide Web

Bits of code point	First code point	Last code point	Bytes in sequence	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6
7	U+0000	U+007F	1	0xxxxxxx					
11	U+0080	U+07FF	2	110xxxxx	10xxxxxx				
16	U+0800	U+FFFF	3	1110xxxx	10xxxxxx	10xxxxxx			
21	U+10000	U+1FFFFF	4	11110xxx	10xxxxxx	10xxxxxx	10xxxxxx		
26	U+200000	U+3FFFFFF	5	111110xx	10xxxxxx	10xxxxxx	10xxxxxx	10xxxxxx	
31	U+4000000	U+7FFFFFFF	6	1111110x	10xxxxxx	10xxxxxx	10xxxxxx	10xxxxxx	10xxxxxx


Share of web pages with different encodings
Google measurements
Percentage
70
60
50
40
30
20
10
0
2000
2002
2004
2006
2008
2010
2012
2014
ASCII only
W Europe
UTF-8
JIS
others

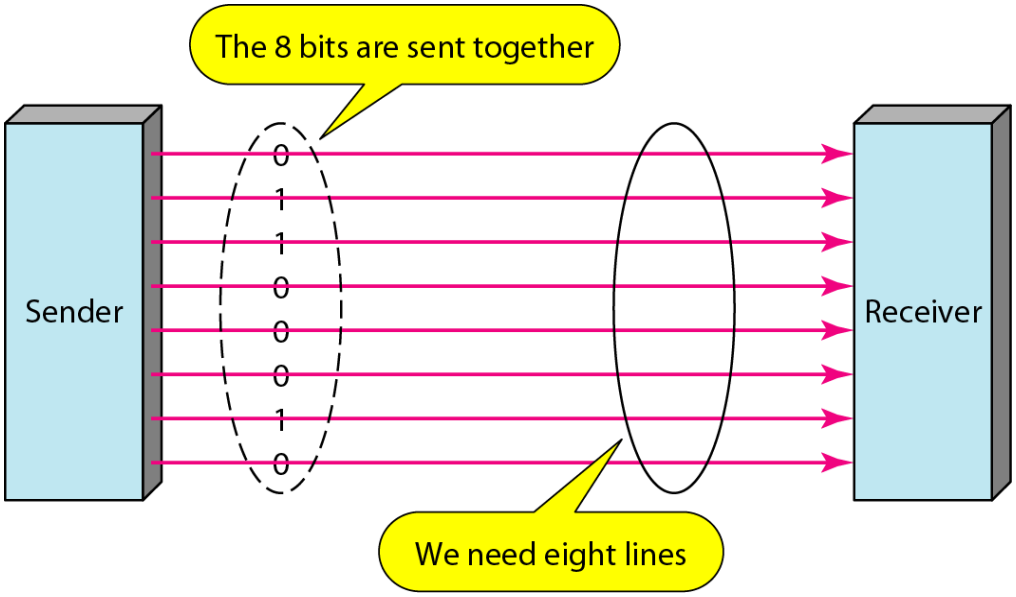
Data Transmission and Modes

- There are two modes of data transmission – Parallel and Serial Mode
- Serial transmission occurs in three ways: asynchronous, synchronous and isochronous



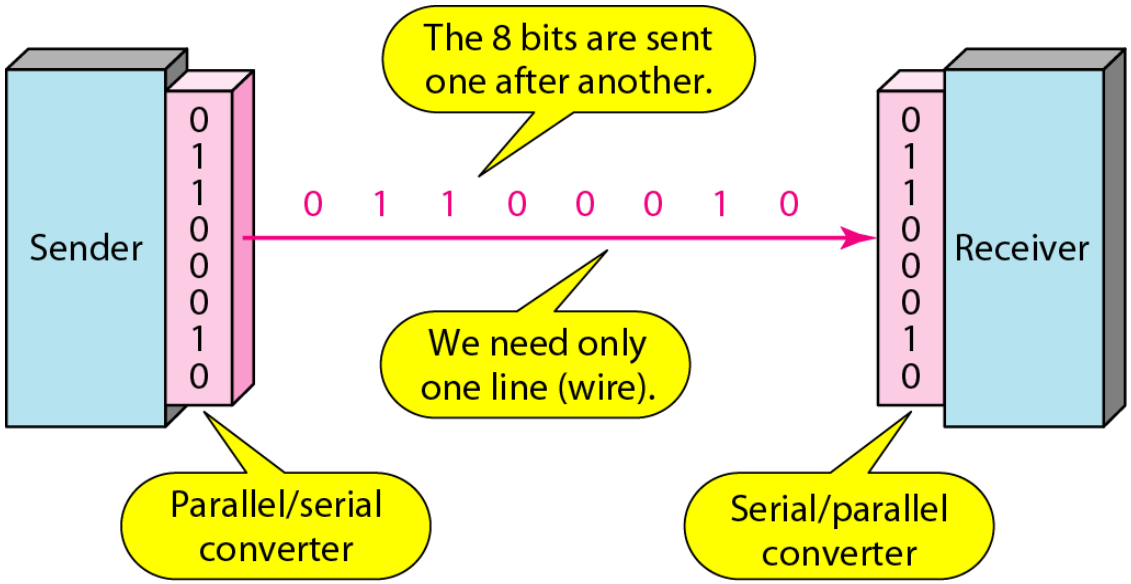
Parallel Transmission

- Binary data (0 or 1) may be organized into groups of n bits.
- n bits can be sent at one time.
- Advantage – Speed
- Disadvantage – Cost
- Good for short distance transmission



Serial Transmission

- One bit follows another and 1 bit is sent at one time
- Advantage – Reduce cost by a factor of n
- Disadvantage – Speed
- Need P/S and S/P converter
- Good for long distance transmission

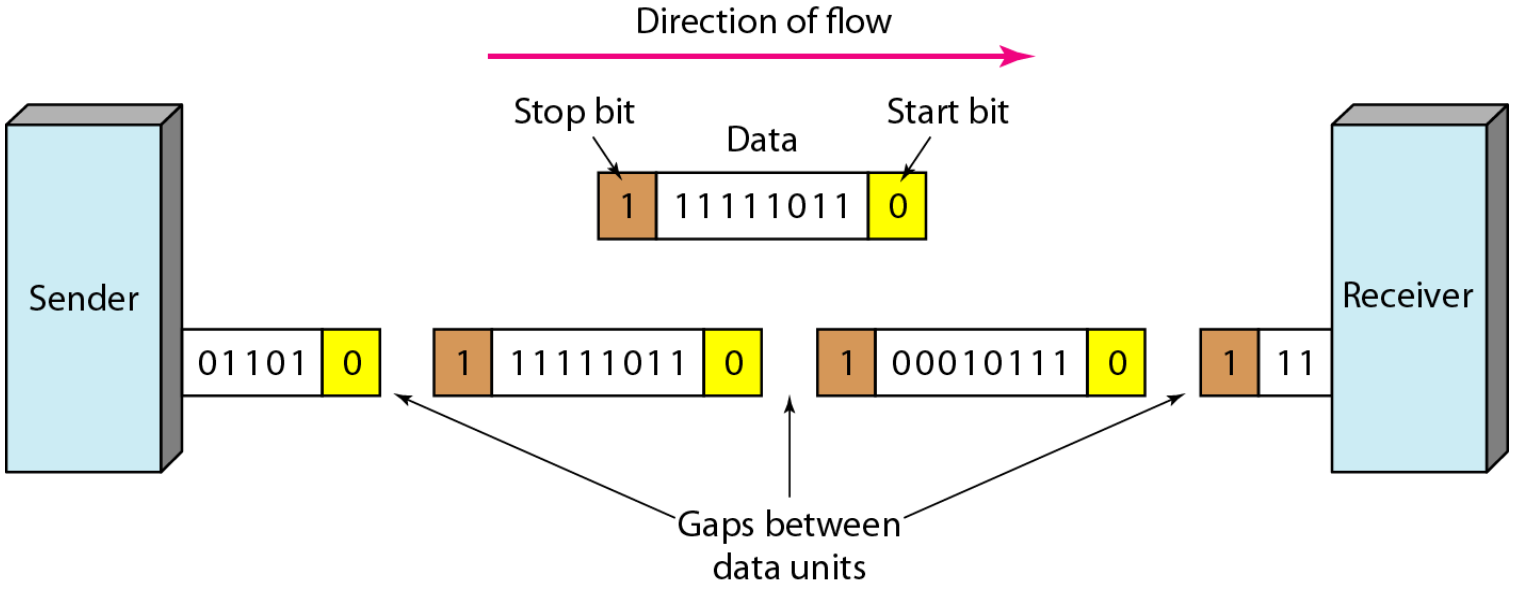


Synchronization

- Transmission of stream of bits needs cooperation and agreement between the sender and receiver
- The receiver needs to know the rate of transmission
- The receiver needs to sample the line to determine the bits
- There are two techniques: Asynchronous and Synchronous

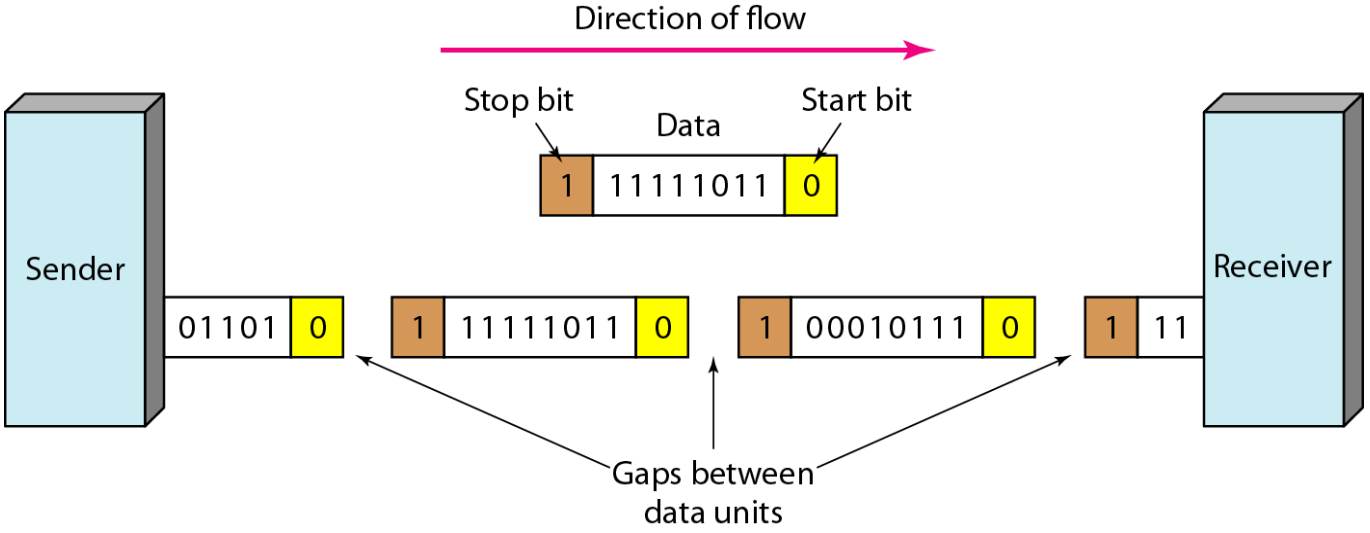
Asynchronous Transmission

- Each character of data (8 bits) is treated independently
- 1 start bit (0) at the beginning and 1 or more stop bits (1s) at the end of each byte.
- Asynchronous here means “asynchronous at the byte level,” but the bits are still synchronized; their durations are the same.



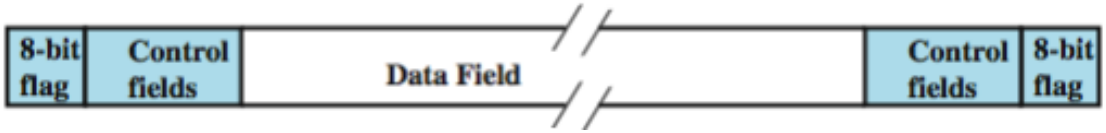
Asynchronous Transmission

- The receiver samples each bit in the character and then look for the beginning of the next character
- Advantage – Cheap and Effective
- Disadvantage – Slow due to the addition of start and stop bits
- Good for low-speed communication, e.g. connection of a keyboard

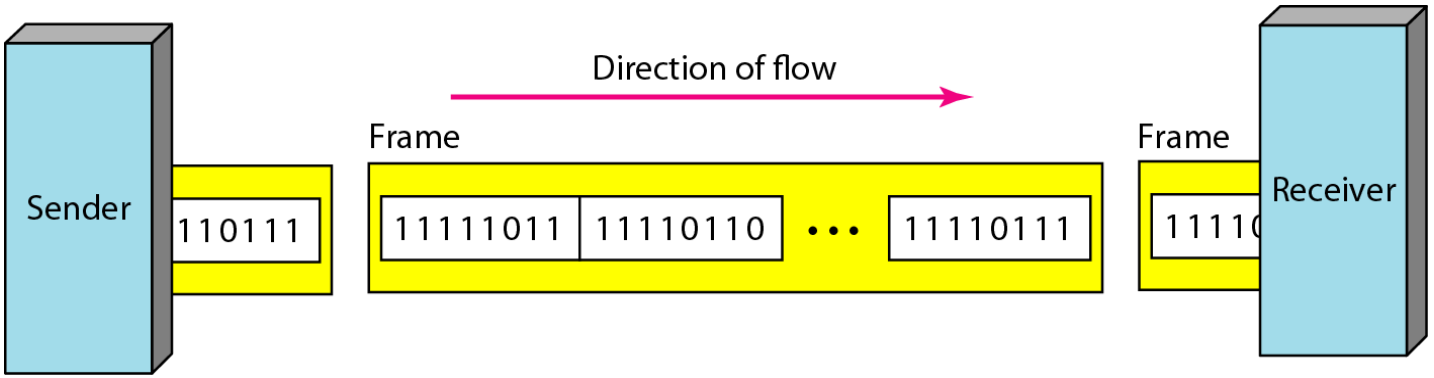


Synchronous Transmission

- Each block of data is formatted as a frame that includes a starting and an ending flag



- Bits are sent one after another without start or stop bits or gaps; receiver needs to group the bits
- Timing is important due to the lack of start and stop bits
- More efficient than asynchronous for large blocks of data (Speed)



Example: Asynchronous Transmission Efficiency

ABCDEFGHIJKLMNOPQRSTUVWXYZ, etc., ABCDEFGHIJKLMNOPQRSTUVWXYZ
Each character = 8 data bits, 1 start bit, 1 stop bit = 10 total bits

If we need to send 1000 characters,
then 10,000 total bits are sent,
of which 8000 are data bits → 80% efficiency

$$\text{Efficiency} = \frac{\text{Data}}{\text{Data} + \text{Overhead}}$$

If we need to send 40 characters,
then 400 total bits are sent,
of which 320 are data bits → 80% efficiency

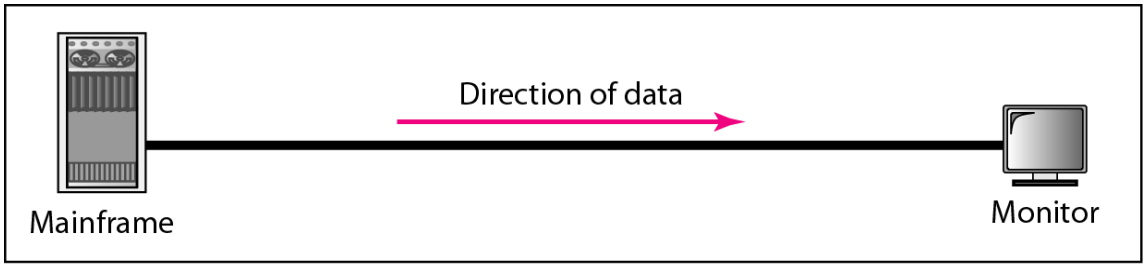
If we need to send 20 characters,
then 200 total bits are sent,
of which 160 are data bits → 80% efficiency

Example: Synchronous Transmission Efficiency

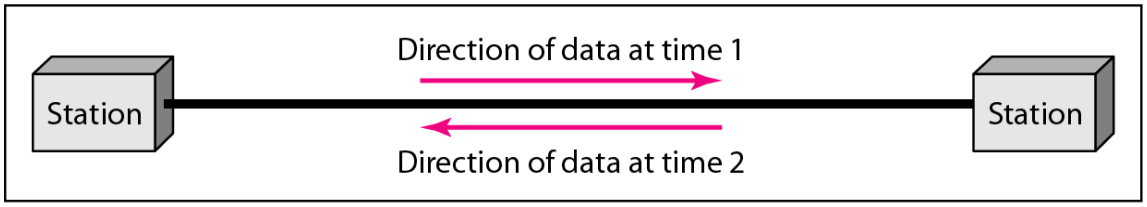
*****ABCDEFGHIJKLMNOPQRSTUVWXYZ, etc., ABCDEFGHIJKLMNOPQRSTUVWXYZ*****
Assume that each block of data needs 10 special characters (shown as *)
Each character = 8 data bits
If we need to send 1000 characters,
we add the 10 special characters,
for a total of 1010 characters,
so we send 8080 total bits,
of which 8000 are data bits → 99.1% efficiency
If we need to send 40 characters,
we add the 10 special characters,
for a total of 50 characters,
so we send 400 total bits,
of which 320 are data bits → 80% efficiency
If we need to send 20 characters,
we add the 10 special characters,
for a total of 30 characters,
so we send 240 total bits,
of which 160 are data bits → 66.7% efficiency

Modes of Data Flow

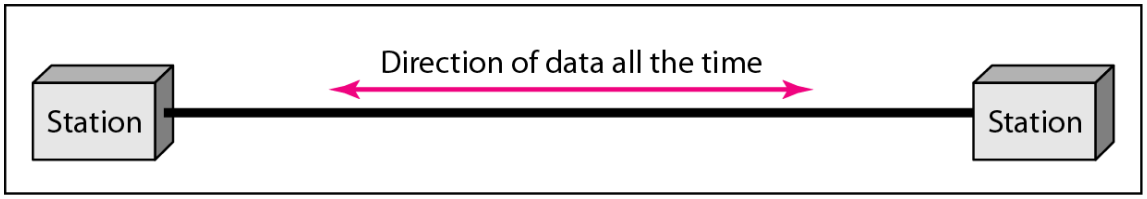
- Communication between two devices can be Simplex, Half-Duplex (HD) or Full-Duplex (FD)



a. Simplex

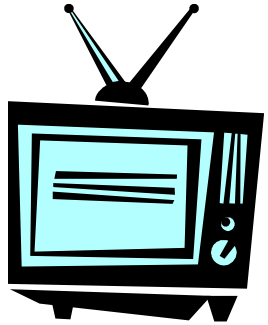


b. Half-duplex

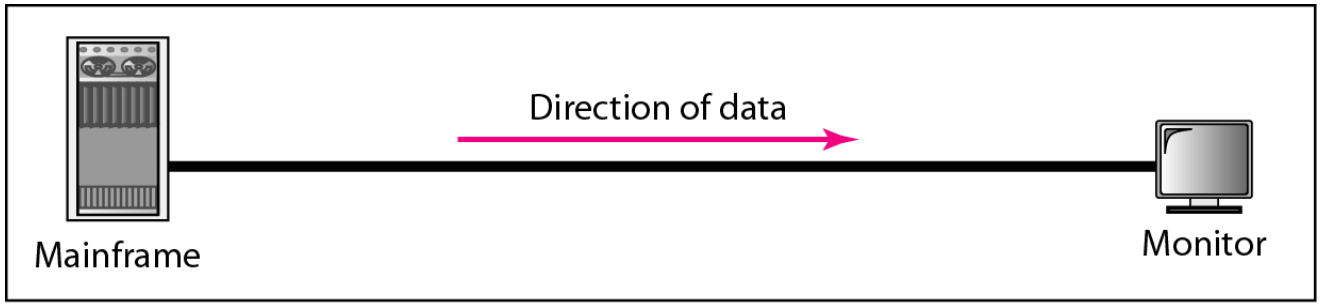


c. Full-duplex

Simplex

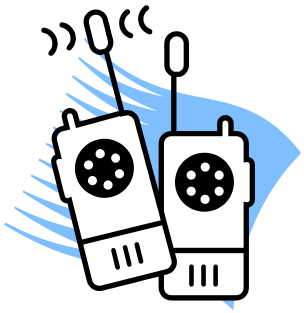


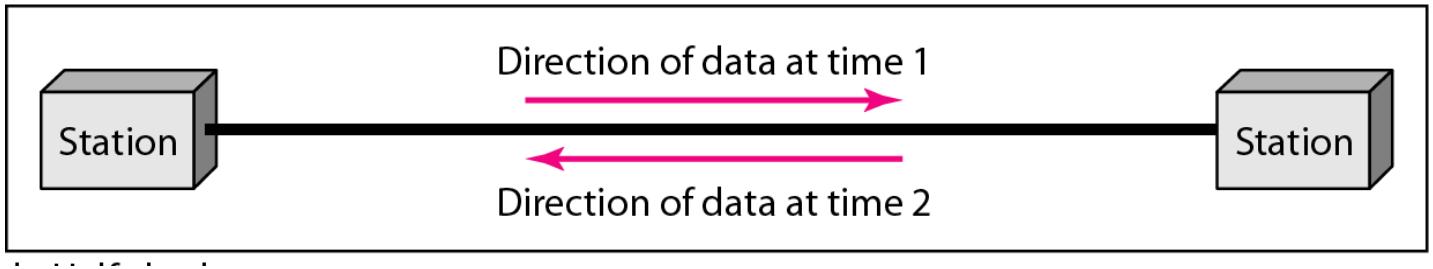
- Signals transmitted in one direction, e.g. TV
- Used for video monitors at the airport for displaying the arrival and departure information, or receive-only printers
- The entire capacity of the channel can be used to send data in one direction



Half Duplex (HD)

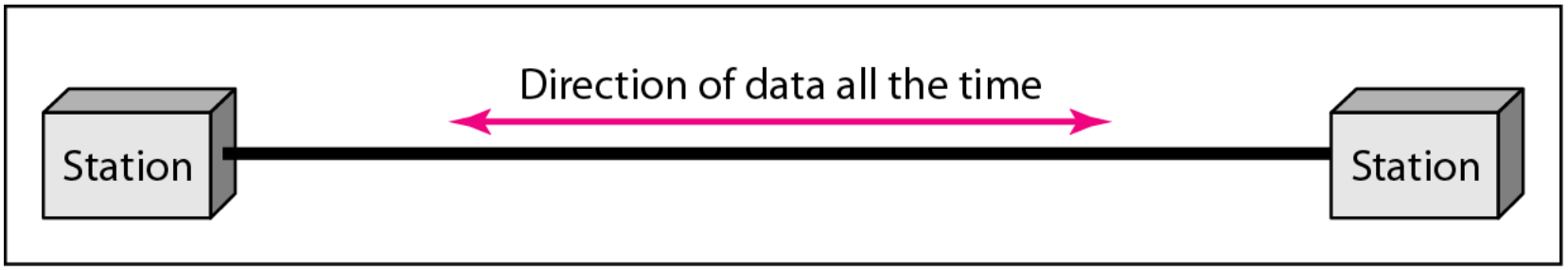
- Both stations transmit, but only one at a time, e.g. police radio, walkie-talkies
- Human conversations are conducted in half duplex fashion.
- The entire capacity of the channel can be used for each direction





Full Duplex (FD)

- Simultaneous transmissions in both transmission paths, e.g. Telephone
- Like a two-way street with traffic flowing in both directions at the same time
- The capacity of the channel must be divided between the two directions



Summary

- Data codes
  - Morse code, Baudot code, EBCDIC, ASCII, Unicode
- Data transmission and modes
  - serial and parallel transmission
  - synchronous and asynchronous transmission
- Modes of data flow
  - simplex
  - half-dulplex (HD)
  - full-duplex (FD)