1. System pozycyjny zapisu liczb

Oznaczenia:

R - podstawa pozycyjnego systemu liczenia (liczba naturalna)
L - wartość liczby
a_i - cyfra ; $a_i \in \{0,1,.., R-1\}$

Zapis liczby (łańcuch cyfr):

$$a_k\, a_{k-1} \ldots a_0\, ,\, a_{-1} \ldots a_{-m}$$

Wartość liczby:

$$L = \sum a_i \times R^i$$

- Przykład

  R=10
  Zapis: 235,08
  Wartość: L = 8x0,01 + 0x0.1 + 5x1 + 3x10 + 2x100 = 235,08d

- Przykład

  R=8
  Zapis: 235,04q
  Wartość: L = 4x(1/64) + 0x(1/8) + 5x1 + 3x8 + 2x64 = 157,0625d

Oznaczenia systemu liczenia za pomocą przyrostków:

d - dziesiętny (*decimal*)
b - dwójkowy (*binary*)
q - ósemkowy (*octal*)
h - szesnastkowy (*hexadecimal*)

- Przykład

  35,4q = 29,5d

- Przykład

  11101,1b = 29,5d

- Przykład

  1Dh = 29,5d

W systemie szesnastkowym (R=16) stosowane są cyfry:

0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F.

Cyfry szesnastkowe mają następujące wartości dziesiętne:

Cyfra hex	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
Wartość dzies.	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Zamiast przyrostków literowych stosuje się też indeksy wyrażające podstawę R w postaci dziesiętnej.

- Przykład

  $11101{,}1_2 = 29{,}5_{10}$

- Przykład

  $100,C_{16} = 256,75_{10}$

- Przykład

  $212_3 = 23_{10}$

2. Zamiana systemu liczenia (konwersja podstawy z R1 na R2)

2.1 Liczby całkowite

Dzieli się liczbę przez R2 i odrzuca resztę; dzielenie powtarza się aż do uzyskania wyniku zerowego.
Odrzucone reszty dają zapis liczby w systemie R2.

- Przykład

  Zamiana 29d na postać dwójkową

Kolejna dzielna	Reszta z dzielenia przez 2
29	**1**
14	**0**
7	**1**
3	**1**
1	**1**
0	

$29_{10} = 11101_2$

- Przykład

  Zamiana 29d na postać trójkową

Kolejna dzielna	Reszta z dzielenia przez 3
29	**2**
9	**0**
3	**0**
1	**1**
0	

$29_{10} = 1002_3$

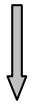
2.2 Liczby ułamkowe

Mnoży się liczbę przez R2 , część całkowita stanowi kolejną cyfrę liczby w systemie R2, część ułamkowa używana jest do dalszego mnożenia. Przekształcenie kończy się po uzyskaniu założonej dokładności lub wyniku zerowego.

♦ Przykład

Zamiana 0,83d na postać dwójkową

Kolejna mnożna	Część całkowita
0,83	**0,**
(1),66	**1**
(1),32	**1**
(0),64	**0**
(1),28	**1**
(0),56	**0**
(1),12	**.**



$0,83_{10} = 0,11010..._2$

2.3 Liczby dwójkowe

Przy konwersji liczb dwójkowych wygodnie jest korzystać z definicji systemu dwójkowego: sumować wagi tych pozycji, na których w zapisie dwójkowym jest 1.

♦ Przykład

Liczba 110101b ma jedynki na pozycjach o wagach równych 1, 4, 16 i 32, więc jej wartość dziesiętna wynosi 1+4+16+32 = 53

Cyfry dwójkowe	1	1	0	1	0	1
Wagi	2^5	2^4	2^3	2^2	2^1	2^0

♦ Przykład

Liczba 0,1101b ma jedynki na pozycjach o wagach równych ½ , ¼ i 1/16 , więc jej wartość dziesiętna wynosi ½ + 1/4 + 1/16 = 13/16

Cyfry dwójkowe	0,	1	1	0	1
Wagi	2^0	2^{-1}	2^{-2}	2^{-3}	2^{-4}

2.4 Konwersja liczb dwójkowych na szesnastkowych

Przy konwersji liczb dwójkowych na szesnastkowe wystarczy korzystać z tablicy cyfr szesnastkowych i każdą czwórkę cyfr dwójkowych zastąpić odpowiednią cyfrą szesnastkową. Analogicznie można przekształcać liczby szesnastkowe na dwójkowe.

♦ Przykład

Liczba 110110101001000b = 6D48h

(0)110 1101 0100 1000
6 D 4 8

♦ Przykład

Liczba 5AC73h = 1011010110001110011b

5 A C 7 3
0101 1010 1100 0111 0011

♦ Przykład

Liczba 11011010,111001b = DA,E4h

1101 1010 , 1110 01(00)
D A E 4

3. Kodowanie liczb

Przy kodowaniu liczb jest ściśle określony repertuar znaków (np. tylko cyfry 0 i 1) i ściśle określona liczba pozycji w kodzie. Te ograniczenia wymagają sprecyzowania konwencji co do przedstawiania znaku liczby.
Stosowane są 3 kody:

- kod znak-moduł,
- kod uzupełnień do zredukowanej podstawy liczenia,
- kod uzupełnień do podstawy liczenia.

W komputerach dla systemu dwójkowego najbardziej rozpowszechniony jest kod uzupełnieniowy do dwóch (U2, *two's complement*). Przy dwójkowym kodowaniu liczb dziesiętnych dodatkowy problem stanowi dwójkowe kodowanie cyfr systemu dziesiętnego.

Przyjęte oznaczenia:

n - liczba pozycji w zapisie liczby
$N \geq n$

3.1 Kod "znak-moduł"

Definicja:

$$x_{ZM} = x \qquad \text{dla } x \geq 0;$$
$$x_{ZM} = R^{n-1} - x \qquad \text{dla } x \leq 0$$

Dla liczb dwójkowych definicję kodu znak-moduł można sformułować następująco:

$$x_{ZM} = x \qquad \text{dla } x \geq 0;$$
$$x_{ZM} = 2^{n-1} - x \qquad \text{dla } x \leq 0$$

Praktyczny sposób uzyskania zapisu w postaci znak-moduł polega na dopisaniu na pozycji znakowej 0 (gdy liczba dodatnia) lub 1 (gdy liczba ujemna).
W kodzie tym są dwie postacie zera („zero dodatnie" i „zero ujemne"). Przedział przedstawianych liczb jest symetrycznym i wynosi od $-(2^{n-1} - 1)$ do $+(2^{n-1} - 1)$.

- Przykład

  Liczba -13d (czyli - 1101b) zapisana na 6 pozycjach (n=6) ma postać
  100000 - (-1101) = 101101.

- Przykład

  Ta sama liczba -13d zapisana na 8 pozycjach (n=8) ma postać
  10000000 - (-1101) = 10001101.

- Przykład

  Liczba +13d (czyli + 1101b) zapisana na 8 pozycjach (n=8) ma postać
  00001101 (początkowe zera nie mogą być pominięte, gdyż pokazują postać kodu liczby).

- Przykład

  16-bitowe liczby dwójkowe w kodzie ZM:

n =16	Postać dwójkowa	Postać szesnastkowa	Wartość dziesiętna
Zero	0000000000000000 1000000000000000	0000 8000	+0 -0
Liczba największa	0111111111111111	7FFF	$2^{15} - 1$
Liczba najmniejsza	1111111111111111	FFFF	$-(2^{15} - 1)$

3.2 Kod uzupełnień do podstawy

Definicja:

$$x_{UR} = x \qquad \text{dla } x \geq 0;$$
$$x_{UR} = R^N + x \qquad \text{dla } x < 0$$

3.2.1 Uzupełnienie dziesiątkowe

Definicja uzupełnienia dziesiątkowego (U10) liczb dziesietnych:

$$x_{U10} = x \qquad \text{dla } x \geq 0;$$
$$x_{U10} = 10^N + x \qquad \text{dla } x < 0$$

- Przykład

  Dla n=7
  $+ 672_{10} = 0000672_{U10}$

- Przykład

  Dla n=7
  $- 672_{10} = (10000000 - 672)_{U10} = 9999328_{U10}$

- Przykład

  5-cyfrowe liczby dziesiętne w kodzie U10:

n = 5 (pozycji dziesiętnych)	Postać zakodowana	Wartość dziesiętna
Zero	00000	+0
Liczba największa	49999	+ 49999
Liczba najmniejsza	50000	- 50000

3.2.2 Uzupełnienie dwójkowe

Definicja uzupełnienia dwójkowego (U2) liczb dwójkowych:

$$x_{U2} = x \qquad \text{dla } x \geq 0;$$
$$x_{U2} = 2^N + x \qquad \text{dla } x < 0$$

- Przykład

  Dla n=7
  $+ 11010_2 = 0011010_{U2}$

- Przykład

  Dla n=7
  $- 11010_2 = (10000000 - 11010)_{U2} = 1100110_{U2}$

- Przykład

  Dla n=7
  $- 1000000_2 = (10000000 - 1000000)_{U2} = 1000000_{U2}$

♦ Przykład

16-bitowe liczby dwójkowe w kodzie U2:

n =16	Postać dwójkowa	Postać szesnastkowa	Wartość dziesiętna
Zero	0000000000000000	0000	+0
Liczba największa	0111111111111111	7FFF	$2^{15} - 1$
Liczba najmniejsza	1000000000000000	8000	-2^{15}

3.3 Kod uzupełnień do podstawy zredukowanej

Definicja:

$$x_{U(R-1)} = x \qquad \text{dla } x \geq 0;$$
$$x_{U(R-1)} = R^N + x - 1 \qquad \text{dla } x \leq 0$$

3.3.1 Uzupełnienie dziewiątkowe

Definicja uzupełnienia dziewiątkowego (U9) liczb dziesiętnych:

$$x_{U9} = x \qquad \text{dla } x \geq 0;$$
$$x_{U9} = 10^N + x - 1 \qquad \text{dla } x < 0$$

♦ Przykład

Dla n=7

$+ 672_{10} = 0000672_{U9}$

♦ Przykład

Dla n=7

$- 672_{10} = (10000000 - 672 - 1)_{U9} = (9999999 - 672)_{U9} = 9999327_{U9}$

♦ Przykład

5-cyfrowe liczby dziesiętne w kodzie U9:

n = 5 (pozycji dziesiętnych)	Postać zakodowana	Wartość dziesiętna
Zero	00000 99999	+ 0 - 0
Liczba największa	49999	+ 49999
Liczba najmniejsza	50000	- 49999

3.3.2 Uzupełnienie jedynkowe

Definicja uzupełnienia jedynkowego (U1) liczb dwójkowych:

$$x_{U1} = x \qquad \text{dla } x \geq 0;$$
$$x_{U1} = 2^N + x - 1 \qquad \text{dla } x \leq 0$$

Kod ten nazywa się też „kodem odwrotnym”, gdyż postać liczby ujemnej uzyskuje się z zapisu dwójkowego przez zamianę wszystkich zer na jedynki i wszystkich jedynek na zera.

- Przykład

  Dla n=7
  $+ 11010_2 = 0011010_{U1}$

- Przykład

  Dla n=7
  $- 11010_2 = (10000000 - 11010 - 1)_{U1} = (1111111 - 11010)_{U1} = 1100101_{U1}$

- Przykład

  16-bitowe liczby dwójkowe w kodzie U1:

n =16	Postać dwójkowa	Postać szesnastkowa	Wartość dziesiętna
Zero	1111111111111111 0000000000000000	FFFF 0000	-0 +0
Liczba największa	0111111111111111	7FFF	$2^{15} - 1$
Liczba najmniejsza	1000000000000000	8000	$-(2^{15} - 1)$

3.4 Dwójkowe kodowanie liczb dziesiętnych

Do zakodowania wszystkich ***cyfr*** dziesiętnych wystarczają 4 pozycje dwójkowe.
Spośród wielkiej liczby 4-pozycyjnych kodów dwójkowych niemal wyłącznie jest stosowany kod BCD (*Binary Coded Decimal*), choć były również używane inne kody mające interesujące właściwości - np. tzw. samouzupełnieniowe kody „plus 3”, czy kod Aikena. Kod BCD i kod Aikena są kodami wagowymi (każdej pozycji można przypisać wagę analogicznie jak w systemie pozycyjnym). Kod BCD nazywany jest kodem 8-4-2-1.

Liczba 4-elementowych kodów dziesiętnych:

$$\frac{16!}{10!\,6!}\ 10! = 29\ 059\ 430\ 400$$

Cyfra dziesiętna	Kod BCD	Kod +3	Kod Aikena
0	0 0 0 0	0 0 1 1	0 0 0 0
1	0 0 0 1	0 1 0 0	0 0 0 1
2	0 0 1 0	0 1 0 1	0 0 1 0
3	0 0 1 1	0 1 1 0	0 0 1 1
4	0 1 0 0	0 1 1 1	0 1 0 0
5	0 1 0 1	1 0 0 0	1 0 1 1
6	0 1 1 0	1 0 0 1	1 1 0 0
7	0 1 1 1	1 0 1 0	1 1 0 1
8	1 0 0 0	1 0 1 1	1 1 1 0
9	1 0 0 1	1 1 0 0	1 1 1 1
Wagi	8 4 2 1		2 4 2 1

4. Liczby zmiennopozycyjne

4.1 Prosty przykład formatu zmp

W maszynach VAX w formacie zmiennopozycyjnym krótkim (*literal*) liczba zajmuje 6 bitów: 3 bity określają wykładnik (e) i 3 bity część ułamkową (f).

e	f
3	3

Wartość liczby wynosi

$$L = 2^{e}\ (0,1f)$$

♦ Przykład

Postać dwójkowa liczb 0,75; 2,5; 104.

Dziesiętnie	Dwójkowo	Postać znormalizowana	Mantysa (m)	e	f
0,75	0,11	2^0 x 0,11	0,1100	000	100
2,5	10,1	2^2 x 0,101	0,1010	010	010
104	1101000	2^7 x 0,1101	0,1101	111	101

♦ Przykład

Wartości liczb o postaciach szesnastkowych 35; 1B.

35h = 110101b czyli e = 110b = 6d , f = 101b, m = **0,1**101b,
$L=2^6$ x 0,1101 = 110100b = 52d

1Bh = 011011b czyli e= 011b = 3d, f = 011b, m = **0,1**011b,
$L=2^3$ x 0,1011 = 101,1b = 5,5d

♦ Przykład

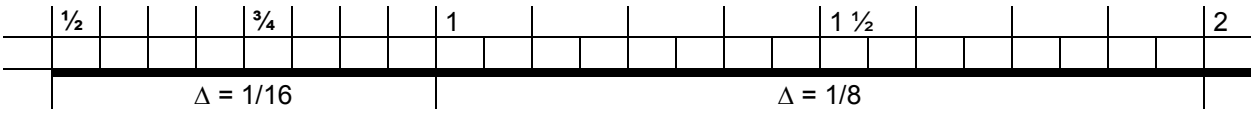
Zakres przedstawianych liczb.

Postać L_m: 000 000 czyli e = 000b = 0, m = **0,1**000,
$L_m = 2^0$ x 0,1= 0,5d
Postać L_M: 111 111 czyli e = 111b = 7d, m = **0,1**111,
$L_M = 2^7$ x 0,1111= 1111000b = 120d

♦ Przykład

Położenie liczb zmp. na osi liczbowej.

- Można przedstawić 64 liczby (od ½ do 120),
- nie ma liczb ujemnych,
- nie ma zera.



4.2 Liczby zmp. 32-bitowe wg standardu IEEE 754

Liczby zmp. 32-bitowe wg standardu IEEE 754 mają format

s	e	f
1	8	23

i są kodowane następująco:

	f = 0	f ≠ 0
$e_m < e < e_M$	$(-1)^s\ 2^{e-127}\ (1.f)$	
$e = e_m = 0...0$	± 0	$(-1)^s\ 2^{-126}\ (0.f)$
$e = e_M = 1...1$	$(-1)^s$ INF	NaN

- liczby znormalizowane
- liczby nieznormalizowane

INF - nieskończoność, ∞ (*infinity*)
NaN - tzw. "nieliczba" (*not-a-number*), np. 35/0, sqrt(-5)

♦ Przykład

Postać dwójkowa i szesnastkowa liczb dziesiętnych 10, 100, -10, -100.

+10d = 1010b = 2^3 x 1,01 czyli e = 127+3 = 130d = 10000010b,
f = 010...0
postać bin. liczby zmp.: 0 10000010 010.....0
postać hex.: 41 20 00 00

+100d = 1100100d = 2^6 x 1,1001 czyli e = 127+6 = 133d = 10000101b,
f = 10010...0
postać bin. liczby zmp.: 0 10000101 100100...0
postać hex.: 42 C8 00 00

-10d zmienia się tylko bit znaku w stosunku do +10d
postać bin. liczby zmp.: 1 10000010 010.....0
postać hex.: C1 20 00 00

♦ Przykład

Postać hex i wartość dziesiętna
(a) największej liczby znormalizowanej ($+L_M$),
(b) najmniejszej liczby dodatniej znormalizowanej ($+L_m$),
(c) największej liczby nieznormalizowanej.

(a)
postać bin.: 0 11111110 11111111111111111111111
postać hex.: 7F 7F FF FF
wartość: $2^{(254-127)} \times (2 - 2^{-23}) \approx 2^{128} \approx 3{,}37 \times 10^{38}$

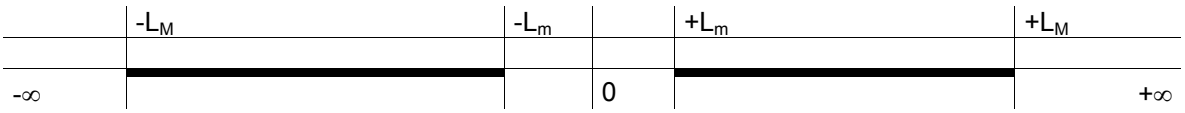
(b)
postać bin.: 0 00000001 00000000000000000000000
postać hex.: 00 80 00 00
wartość: $2^{(1-127)} \times 1 = 2^{-126} \approx 1{,}2 \times 10^{-38}$

(c) *Uwaga: inna definicja (e=0) !*

postać bin.: 0 00000000 11111111111111111111111
postać hex.: 00 7F FF FF
wartość: $2^{-126} \times (1 - 2^{-23}) = L_m - 2^{-149}$

- Przykład

Zakres przedstawianych liczb.



4.3 Liczby zmp. 32-bitowe wg standardu IBM 360/370

Liczby zmp. 32-bitowe wg standardu IBM 360/370 mają format

s	e	f
1	7	24

i interpretację

$$L = (-1)^s \, 16^{e-64} \, (0.f)$$

Część ułamkowa (f) liczby znormalizowanej ma pierwszą cyfrę szesnastkową ułamka (f) różną od zera (czyli na bitach <8:11> jest co najmniej jedna 1).
Uwaga: bity w słowie IBM są numerowane od lewej strony <0:31>.

5. Kod ASCII

Kod ASCII (*American Standard Code for Information Interchange*) oryginalnie jest kodem 7-bitowym, tzn. każdy znak kodowany jest jako zestaw siedmiu zer i jedynek. Niemal identyczny z ASCII jest kod ISO-7.

5.1 Tablica kodu ASCII

Jak w każdym kodzie znakowym, w ASCII zawarte są:
- znaki cyfr dziesiętnych,
- znaki liter alfabetu angielskiego,
- znaki graficzne stosowane w piśmie (np. kropka, cudzysłów, nawiasy),
- znaki sterujące formatem (np. spacja, tabulacje, nowa linia),
- znaki sterujące transmisją - niewidoczne w tekście, używane w protokołach transmisji znakowej (np. początek nagłówka, koniec tekstu).

Czasem stosowany jest kod ASCII rozszerzony do 8 (każdy znak zajmuje 8 bitów) - jest to związane z bajtową organizacją pamięci, w której najmniejszą adresowalną porcją danych jest 8 bitowy bajt. W kodzie 8-bitowym skrajny lewy bit ma postać 0. Ustawienie tego bitu na 1 umożliwia kodowanie dodatkowych znaków; np. znaków narodowych występujących w różnych alfabetach łacińskich.

W poniższych tablicach podano wartość *szesnastkową* kombinacji kodowych, np. znak „q” jest kodowany na 7 bitach jako 71h, czyli dwójkowo 1110001. W przypadku kodowania na 8 bitach ten sam znak ma postać dwójkową 01110001.

Znaki graficzne ASCII:

Wartość hex	Znak	Wartość hex	Znak	Wartość hex	Znak
20	SP (spacja)	40	@	60	
21	!	41	A	61	a
22	„	42	B	62	b
23	#	43	C	63	c
24	$	44	D	64	d
25	%	45	E	65	e
26	&	46	F	66	f
27	‘	47	G	67	g
28	(	48	H	68	h
29	)	49	I	69	i
2A	*	4A	J	6A	j
2B	+	4B	K	6B	k
2C	,	4C	L	6C	l
2D	-	4D	M	6D	m
2E	.	4E	N	6E	n
2F	/	4F	O	6F	o
30	0	50	P	70	p
31	1	51	Q	71	q
32	2	52	R	72	r
33	3	53	S	73	s
34	4	54	T	74	t
35	5	55	U	75	u
36	6	56	V	76	v
37	7	57	W	77	w
38	8	58	X	78	x
39	9	59	Y	79	y
3A	:	5A	Z	7A	z
3B	;	5B	[	7B	{
3C	<	5C	\	7C	|
3D	=	5D	]	7D	}
3E	>	5E	^	7E	~
3F	?	5F	_	7F	DEL

Znaki funkcyjne ASCII:

Wartość hex	Znak	Nazwa angielska	Znaczenie
0	NUL	Null	Znak pusty
1	SOH	Start of Heading	Początek nagłówka
2	STX	Start of Text	Początek tekstu
3	ETX	End of Text	Koniec tekstu
4	EOT	End of Transmission	Koniec transmisji
5	ENQ	Enquiry	Pytanie
6	ACK	Acknowledge	Odpowiedź pozytywna
7	BEL	Bell	Dzwonek
8	BS	Backspace	Cofacz
9	HT	Horizontal tab	Tabulacja pozioma
A	LF	Line Feed	Zmiana wiersza
B	VT	Vertical tab	Tabulacja pionowa
C	FF	Form Feed	Zmiana arkusza
D	CR	Carriage Return	Powrót karetki
E	SO	Shift out	Wyjście z kodu
F	SI	Shift in	Powrót do kodu
10	DLE	Data Link Escape	Zmiana znaczenia
11	DC1	Device Control 1	Sterowanie urządzenia
12	DC2	Device Control 2	Sterowanie urządzenia
13	DC3	Device Control 3	Sterowanie urządzenia
14	DC4	Device Control 4	Sterowanie urządzenia
15	NAK	Negative Acknowledge	Odpowiedź negatywna
16	SYN	Synchronous Idle	Znak synchronizujący
17	ETB	End of Transmit. Block	Koniec bloku
18	CAN	Cancel	Kasowanie
19	EM	End of Medium	Koniec nośnika
1A	SUB	Substitute	Podstawienie
1B	ESC	Escape	Zmiana kodu
1C	FS	File Separator	Seperator plików
1D	GS	Group Separator	Separator grup
1E	RS	Record Separator	Separator rekordów
1F	US	Unit Separator	Separator jednostek

Przy transmisji i przechowywaniu tekstów w ASCII dodawany bywa dodatkowy tzw. „bit parzystości", dzięki czemu każdy znak, po nacechowaniu parzystością ma zawsze parzystą (*ewen*) lub nieparzystą (*odd*) - zależnie od przyjętej reguły - liczbę jedynek. Bit parzystości może być dodany jako ósmy bit do kodu 7-bitowego lub jako dziewiąty bit do kodu 8-bitowego.

- Przykład

Tekst <**Zima 98**> jest łańcuchem 7 znaków ASCII o wartościach szesnastkowych:

5A 69 6D 61 20 39 38

<Z> <i> <m> <a> SP <9> <8>

Jeżeli znaki są kodowane na 7 bitach postać dwójkowa tego komunikatu jest:

1011010 1101001 1101101 1100001 0100000 0111001 0111000

Jeżeli znaki są kodowane na 8 bitach postać dwójkowa tego komunikatu jest:

01011010 01101001 01101101 01100001 00100000 00111001 00111000

czyli szesnastkowo:

5A 69 6D 61 20 39 38

Jeżeli znaki są kodowane na 8 bitach z bitem parzystości „parzystej” postać dwójkowa tego komunikatu jest:

01011010 01101001 11101101 11100001 10100000 00111001 10111000

czyli szesnastkowo:

5A 69 ED E1 A0 39 B8

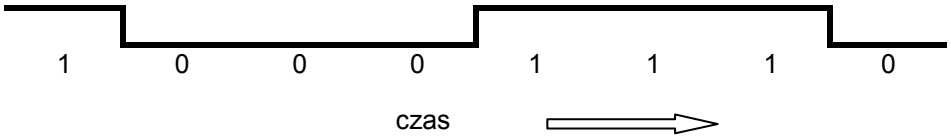
Jeżeli znaki są kodowane na 8 bitach z bitem parzystości „nieparzystej” postać dwójkowa tego komunikatu jest:

11011010 11101001 01101101 01100001 00100000 10111001 00111000

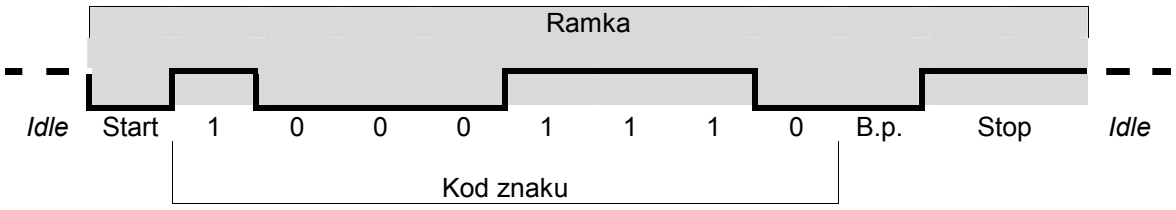
czyli szesnastkowo:

DA E9 6D 61 20 B9 38

Przy transmisji szeregowej (bit po bicie) bity znaku są przesyłane zaczynając od bitu najmniej znaczącego (w powyższym zapisie od prawej strony) stąd wykres czasowy transmisji 8-bitowego znaku „q” będzie wyglądał:



Transmisja asynchroniczna umożliwia przesyłanie kolejnych znaków w dowolnych odstępach. W tym celu każdy znak jest umieszczany w tzw. ramce (*frame*) składającej się z jednego „bitu startu” - na początku i jednego lub dwóch „bitów stopu” - na końcu znaku. W ramce może być umieszczony kod 7- lub 8-bitowy, z bitem parzystości lub bez. Bit startu ma wartość 0, bity stopu - wartość 1. W czasie jałowym (*idle*), pomiędzy ramkami, stan linii jest równy 1, więc odbiornik może wykryć początek nowej ramki jako przejście $1 \rightarrow 0$ („start”).
Wykres czasowy transmisji ramki z 8-bitowym znakiem „q” opatrzonym bitem parzystości parzystej będzie wyglądał:



5.2 Zastosowanie znaków sterujących transmisją

Przy transmisji szeregowej znakowej stosowane są różne konwencje dotyczące organizacji przesyłania i struktury przesyłanego komunikatu zwane protokołami. Na przykład, protokół BISYNC (*Binary Synchronous Transmission*) przewiduje przesyłanie komunikatów w postaci tzw. „ramek” zawierających oprócz tekstu komunikatu część sterującą czyli tzw. „nagłówek” (*header*). W nagłówku może być podany np. adres odbiornika i liczba znaków tekstu.

SYN	SYN	SOH	Nagłówek	STX	Tekst	ETX	BCC

Pole nagłówka zaczyna się od znaku sterującego SOH, pole tekstu od znaku STX a kończy się znakiem ETX.
Przesłanie ramki poprzedzone jest znakami SYN służącymi do synchronizacji odbiornika (dopiero po wykryciu znaku SYN odbiornik zaczyna kompletować nadchodzące bity w znaki).
Ramka kończy się znakiem kontrolnym BCC (*block check character*), który jest tworzony w nadajniku podczas wysyłania znaku i służy do sprawdzenia poprawności transmisji. BCC może być np. tzw. sumą kontrolną (suma mod 2), w której każdy bit jest bitem parzystości wzdłużnej wyznaczonym dla wszystkich poprzednio wysłanych znaków.

♦ Przykład

Transmisja tekstu <Zima 98> wg protokołu BISYNC.
Kod ASCII 8-bitowy z bitem parzystości parzystej (*even*). Adres odbiornika = 5.
Ramka zawiera następującą sekwencję 15 znaków:

SYN SYN SOH <5> <7> STX <Z> <i> <m> <a> SP <9> <8> ETX BCC

Postać dwójkowa ramki:

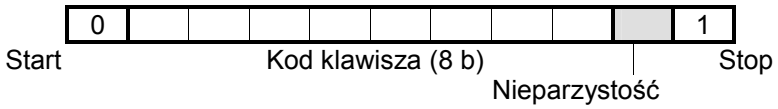
Znak	Kod (7 bitów + bit parz.)	Komentarz
SYN	1001 0110	Synchronizacja ramki
SYN	1001 0110	
SOH	1000 0001	Początek nagłówka
<5>	0011 0101	Adres odbiornika (=5)
<7>	1011 0111	Długość tekstu (7 znaków)
STX	1000 0010	Początek tekstu
<Z>	0101 1010	Tekst
<i>	0110 1001	
<m>	1110 1101	
<a>	1110 0001	
SP	1010 0000	
<9>	0011 1001	
<8>	1011 1000	
ETX	0000 0011	Koniec tekstu
BCC	1001 1100	Parzystość wzdłużna

5.3 Kody klawiatury

Każdy klawisz ma swój 8-bitowy kod (*scan code*) zależny tylko od położenia na klawiaturze (nie od opisu tego klawisza). Naciśnięcie klawisza generuje tzw. *make code*, a zwolnienie - tzw. *break code*, przy czym

break code = 128 + make code.

Kod klawisza jest przesyłany do procesora szeregowo w postaci 11-bitowej ramki synchronizowanej przebiegiem zegarowym dostarczanym osobnym przewodem (kabel 5-przewodowy: dane, zegar, zero, +5v, reset).



Odebranie kodu powoduje przerwanie IRQ1 obsługiwane przez program wskazany wektorem 09H.
Interpretacja działania klawisza zależy od procedury INT 09H będącej sterownikiem klawiatury (*keyboard handler*).

♦ Przykład

Niektóre kody wciśnięcia (*make code*); klawiatura AT.

Klawisz	Kod	Klawisz	Kod	Klawisz	Kod	Klawisz	Kod
Enter	1C	1	02	Q	0F	A	1E
Lewy Shift	2A	2	03	W	10	S	1F
Prawy Shift	36	3	04	E	11	D	20
Ctrl	1D	4	05	R	12	F	21

- Przykład

  Sekwencja kodów (hex): 02, 82, 2A, 1E, 9E, AA
  oznacza sekwencję następujących czynności:

  1. wciśnięcie <1>
  2. zwolnienie <1>
  3. wciśnięcie <Shift>
  4. wciśnięcie <A>
  5. zwolnienie <A>
  6. zwolnienie <Shift>