Daten codieren

M114 / ARJ / 5-2025

DATEN CODIEREN beinhaltet:

- **Massvorsätze** ***(Repetition)***
- **Bit und Byte** ***(Repetition)***
- **AND / OR / NOT / XOR** ***(Repetition)***
- **Parallele/Serielle Datenübertragung**
- **Data-Clock, Taktsignal bei der Datenübertragung**

- **Zahlensysteme BIN / DEZ / HEX** ***(Repetition)***
- **Binär addieren, DataOverflow**
- **Negative Binärzahlen mit 2-er Komplement**
- **Fliesskommazahlenformat**

- **Online-HEX-Editor hexed.it, Notepad++**

- **Alphanumerische Codes ASCII, ANSI-ASCII, ISO8859-x**
- **Unicode, Unicode V2, UTF8, UTF16 LE-BOM und RE-BOM (Byte-Order)**

- **Reserve: Barcodes EAN8 und EAN13, QR-Codes**

SI-Präfixe (Dezimalpräfixe)

[T] → Tera → 10^{12} → 1'000'000'000'000 → Billion
[G] → Giga → 10^{9} → 1'000'000'000 → Milliarde
[M] → Mega → 10^{6} → 1'000'000 → Million
[k] → kilo → 10^{3} → 1'000 → Tausend

IEC-Präfixe (Binärpräfixe)

[Ti] → Tebi → 2^{40} → 1'099'511'627’776
[Gi] → Gibi → 2^{30} → 1'073'741’824
[Mi] → Mebi → 2^{20} → 1'048’576
[Ki] → Kibi → 2^{10} → 1’024

SI-Präfixe z.B. bei Geschwindigkeitsangaben wie 1Mb/sec
IEC-Präfixe bei Speichergrössen wie z.B. 1MiB

BTW: **B = Byte**
b = bit
1B = 8b
16b = 1 Word
LSB = Least Significant Bit / Kleinstwertigstes Bit
MSB = Most Significant Bit / Höchstwertigste Bit
(*Wichtige Angabe bei z.B. bei Parallelverbindungen damit Stecker nicht falsch angeschlossen wird*)

Der Computer versteht nur 0 und 1

(Wenigstens solange die Quantencomputer mit ihren Q-Bits nicht grossflächig Einzug gehalten haben.)

- **Verarbeitung, Speicherung Binär (BIN)**
- **Darstellung zur besseren Lesbarkeit/Übericht in Hexadezimal (HEX)**

Für sie als Programmierer/in interessant:

- **Wie und wo wird eine Zahl gespeichert?**
- **Welche Basis-Datentypen gibt's es?**
- **Was sind die Wertebereiche dieser Datentypen?**
- **Was passiert bei einer Überschreitung eines Wertebereichs?**

Für sie als Informatiker interessant:

- **Warum ist der Speicherplatz immer ein Vielfaches von 2?**
- **Wie kann ich die Bits und Bytes sichtbar machen?**
  (Einsatz von HEX-Editer. Empfehlung: hexed.it)

Für Informatiker wichtige Zahlensysteme:

BIN **Binärsystem**, Zweiersystem, Dualsystem
Basis: 2
Zeichenvorrat: 0, 1
Zahlenbeispiel: $0110‘1011_B$

DEZ **Dezimalsystem**, Zehnersystem
Basis: 10
Zeichenvorrat: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9
Zahlenbeispiel: 2025_D

HEX **Hexadezimalsystem**, Sechzehnersystem
Basis: 16
Zeichenvorrat: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A (=10), B (=11), C (=12), D (=13), E (=14), F (=15)
Zahlenbeispiel: $FF3C_H$ oder 0xFF3C
Hinweis: Eine Hex-Ziffer entspricht einer vierstelligen Binärzahl bzw. 4 Bit!

Für Informatiker wichtige Zahlensysteme:

BIN **Binärsystem, Zweiersystem, Dualsystem**
Basis: 2
Zeichenvorrat: 0, 1
Zahlenbeispiel: 0110‘1011_B

<u>Beispiel für die Umrechnung von Binär 1011 in Dezimal:</u>

2^3	2^2	2^1	2^0	Wertigkeit der Binärstelle (Basis=2)
8	4	2	1	Wertigkeit Dezimal
1	**0**	**1**	**1**	**Binärwert**
8	0	2	1	Die SUMME ergibt das Dezimaläquivalent: 11

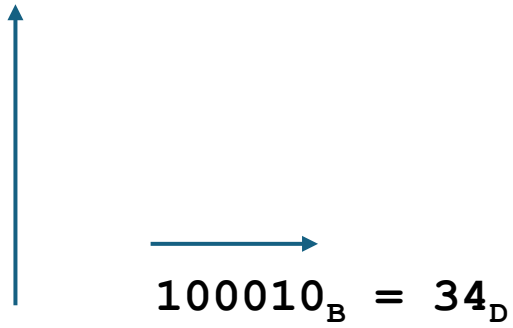
Somit: $1011_B = 11_D$

Für Informatiker wichtige Zahlensysteme:

BIN **Binärsystem**, Zweiersystem, Dualsystem
Basis: 2
Zeichenvorrat: 0, 1
Zahlenbeispiel: $0110‘1011_B$

Beispiel für die Umrechnung Dezimal 34 in Binär: (Fortlaufende Division)

```
34 DIV 2 = 17 REST 0
17 DIV 2 =  8 REST 1
 8 DIV 2 =  4 REST 0
 4 DIV 2 =  2 REST 0
 2 DIV 2 =  1 REST 0
 1 DIV 2 =  0 REST 1
```



$100010_B = 34_D$

Für Informatiker wichtige Zahlensysteme:

HEX **Hexadezimalsystem**, Sechzehnersystem

Basis: 16

Zeichenvorrat: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A (=10), B (=11), C (=12), D (=13), E (=14), F (=15)

Zahlenbeispiel: $FF3C_H$ oder 0xFF3C

Hinweis: Eine Hex-Ziffer entspricht einer vierstelligen Binärzahl bzw. 4 Bit!

Beispiel für die Umrechnung Hexadezimal 5B7 in Binär:

4 Bit entsprechen einer HEX-Ziffer: 0000=0 bis 1111=F

5	B	7	HEX-Wert
8421	8421	8421	Wertigkeit
0101	1011	0111	Hex-Ziffer in Binär

Somit entspricht der HEX-Wert $5B7_H$ dem Binärwert $0101'1011'0111_B$

HEX

Hexadezimalsystem, Sechzehnersystem

Zeichenvorrat: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A (=10), B (=11), C (=12), D (=13), E (=14), F (=15)

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

- 4 Bit entsprechen einer HEX-Ziffer
- Die "Geschwindigkeit" des Bit-Wechsels 0 zu 1 oder 1 zu 0 halbiert sich bei jeder weiteren linken Kolonne
- Das Bit ganz rechts nennt man LSB (Least Significant Bit)
- Das Bit ganz links nennt man MSB (Most Significant Bit)

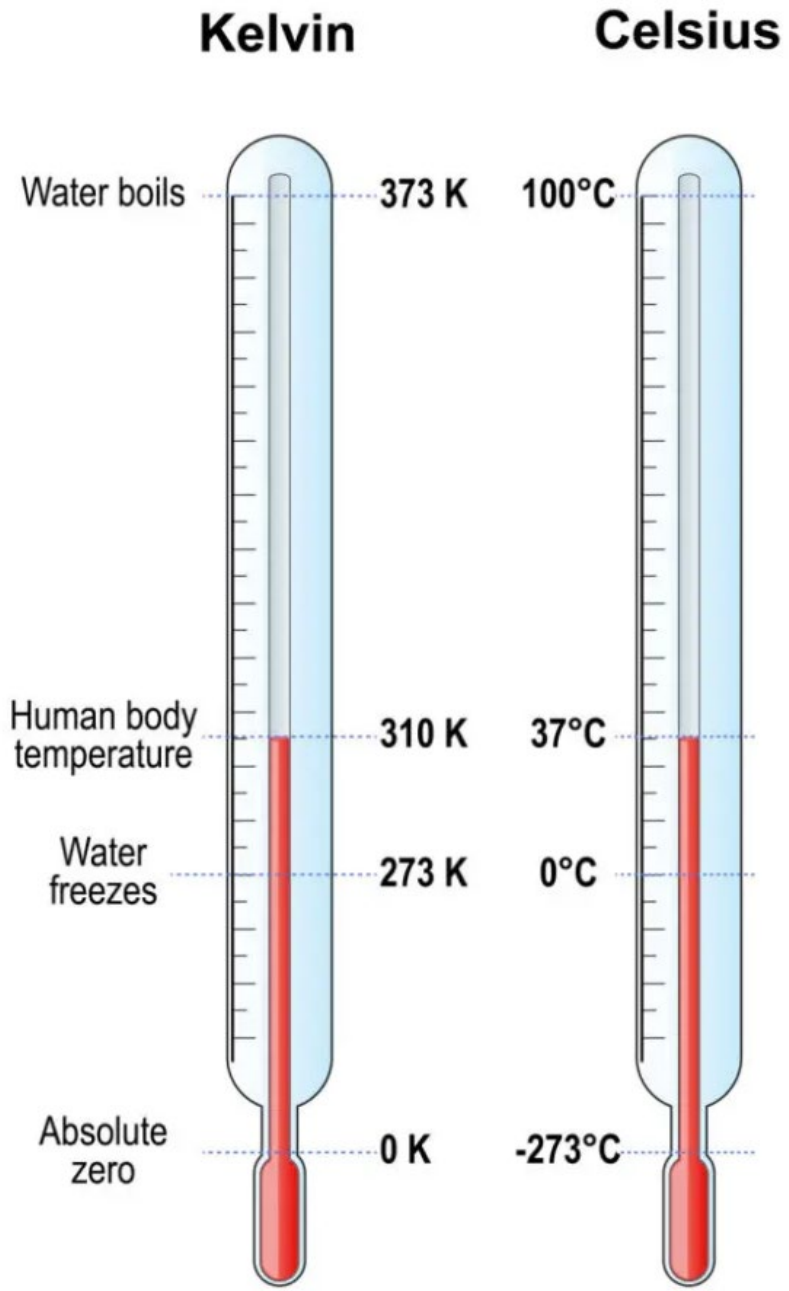
Aufgaben: **Zeit: 8 Minuten Umrechnung ohne Taschenrechner!**

1. Was ist bei der Binärzahl 10111'0111 das MSB (=Most Significant Bit oder Höchstwertigstes Bit) und was ist LSB (= Least Significant Bit oder Kleinstwertigstes Bit)?
2. Umrechnung 11111111_B in Dezimal
3. Umrechnung 01001101_B in Dezimal
4. Wieviele Bit werden benötigt, um 16 Bitkombinationen zu erstellen?
5. Umrechnung 11111111_B in Hexadezimal
6. Umrechnung 10110101_B in Hexadezimal
7. Umrechnung FF_H in Binär
8. Umrechnung $E7_H$ in Binär
9. Umrechnung 37_D in Binär
10. Wieviele Bit werden mindestens benötigt, um 1000 Bitkombinationen darzustellen?

Musterlösungen:

1. MSB und LSB: **1**0111'011**1** **MSB** **LSB**
2. Umrechnung 11111111_B in Dezimal: $128+64+32+16+8+4+2+1=255_D$
3. Umrechnung 01001101_B in Dezimal: $64+8+4+1=77_D$
4. Wieviele Bit werden benötigt, um 16 Bitkombinationen zu erstellen? 4 Bit = 1HEX-Ziffer
5. Umrechnung 11111111_B in Hexadezimal: FF_H
6. Umrechnung 10110101_B in Hexadezimal: $1011=8+2+1=11_D=B_H$, $0101=4+1=5_H$ Somit $B5_H$
7. Umrechnung FF_H in Binär: $1111'1111_B$
8. Umrechnung $E7_H$ in Binär: $E_H=14_D=1110_B$, $7_H=0111_B$ Somit $1110'0111_B$
9. Umrechnung 37_D in Binär:
   37 DIV 2 = 18 R1
   18 DIV 2 = 9 R0
   9 DIV 2 = 4 R1
   4 DIV 2 = 2 R0
   2 DIV 2 = 1 R0
   1 DIV 2 = 0 R1 Somit 0010'0101
10. Wieviele Bit werden mindestens benötigt, um 1000 Bitkombinationen darzustellen?
    Variante 1 durch probieren:
    4 Bit = 16
    5 Bit = 32
    6 Bit = 64
    7 Bit = 128
    8 Bit = 256
    9 Bit = 512
    10 Bit = 1024 Somit 10 Bit
    Variante 2 mit Formel: log 1000 / log 2 = 3/0.30103 = 9.966 Somit 10 Bit

Binärzahlen mit/ohne Vorzeichen



Negative Zahlen = Offset

Binärzahlen mit/ohne Vorzeichen

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

unsigned integer 4Bit
Wertebereich: 0..15

```
0111   +7
0110   +6
0101   +5
0100   +4
0011   +3
0010   +2
0001   +1
0000    0
1001   -1
1010   -2
1011   -3
1100   -4
1101   -5
1110   -6
1111   -7
```

Vorderstes Bit als Vorzeichen?
Funktionieren negative Zahlen so?

Was ist mit **1000** ? Gibt's es nicht?

Wie wärs mit einer Proberechnung?

```
0001 +   (Entspricht +1)
1010     (Entspricht -2)
----
1011     (Entspricht -3)     Na, also!
```

0111	+7
0110	+6
0101	+5
0100	+4
0011	+3
0010	+2
0001	+1
0000	0
1111	-1
1110	-2
1101	-3
1100	-4
1011	-5
1010	-6
1001	-7
1000	-8

Gegeben: signed integer 4Bit
Mit Wertebereich: -8..+7

2-er Komplement-Rezept:

1. Ich möchte z.B. eine -5
2. Ich nehme eine +5 (=0101)
3. Ich negiere jede Stelle einzeln
   Aus 0101 wir 1010
4. Ich addiere 1 dazu
   1010 + 1 = 1011
   1011 ist -5. Wers nicht glaubt, siehe Tabelle

0111	+7
0110	+6
0101	+5
0100	+4
0011	+3
0010	+2
0001	+1
0000	0
1111	-1
1110	-2
1101	-3
1100	-4
1011	-5
1010	-6
1001	-7
1000	-8

Gegeben: signed integer 4Bit
Mit Wertebereich: -8..+7

2-er Komplement-Rezept:
Alternative Variante mit Wertigkeit:

−8	+4	+2	+1
1	0	1	1

Man beachte die negative Wertigkeit der höchsten Stelle (MSB=Most significantest bit)

Aufgaben: **Zeit: 6 Minuten Ohne Taschenrechner!**

1. Addiere die beiden 4Bit Integer $0101_B + 0011_B$
2. Addiere die beiden 4Bit Integer $1101_B + 0110_B$
3. Umrechnung -4_D in Unsigned Integer mit Bitbreite 4
4. Umrechnung -18_D in Unsigned Integer mit Bitbreite 8

Musterlösungen:

1. Addiere die beiden 4Bit Integer 0101_B + 0011_B

```
0101+
0011
1000B
```

2. Addiere die beiden 4Bit Integer 1101_B + 0110_B

```
1101+
0110
0011B   (Data Overflow!)
```

3. Umrechnung -4_D in Unsigned Integer mit Bitbreite 4

   Mit 2-er Komplement
   Positiver Wert $4_D=0100_B$
   Invertiert 1011_B
   +1 1011+1=**1100_B**

   Mit Wertigkeit
   -8 4 2 1
   1 1 0 0 Somit **1100_B**

4. Umrechnung -18_D in Unsigned Integer mit Bitbreite 8

   Mit 2-er Komplement
   Positiver Wert $18_D=0001'0010_B$
   Invertiert $1110'1101_B$
   +1 $1110'1101_B$ +1=**$1110'1110_B$**

   Mit Wertigkeit
   -128 64 32 16 8 4 2 1
   1 1 1 0 1 1 1 0 Somit **$1110'1110_B$**

Fliesskommazahlen (Gleitkomma)

INTEGER HAT ANDEREN ZWECK ALS FLOAT

Dies wäre falsch: (Pseudocode)

```
float f = 300’000’000.0;
f = f + 1.0;   //liegt nicht mehr in der Genauigkeit
printf("%f",f); //Ergibt immer noch 300’000’000
```

Dies ist ok: (Pseudocode)

```
float f = 3.0;
f = 1.0 / f;
```

FLOATing point number = Fliesskommazahl

Beispiel zu "Was ist eine Fliesskommazahl":
Die Zahl Pi kann man so schreiben…

+3.14156
+3.14156 x 10^{+0}
+314.156 x 10^{-2}
+0.0031456 x 10^{+3}

Mantisse=Zahlen und Vorzeichen vor dem Exponenten
Basis=10
Exponent=Zahlen und Vorzeichen im Exponenten

Norm IEEE 754

32 Bit → Single Precision

64 Bit → Double Precision

Gleitkommazahl: $x = \mathbf{v * m * b^{e}}$

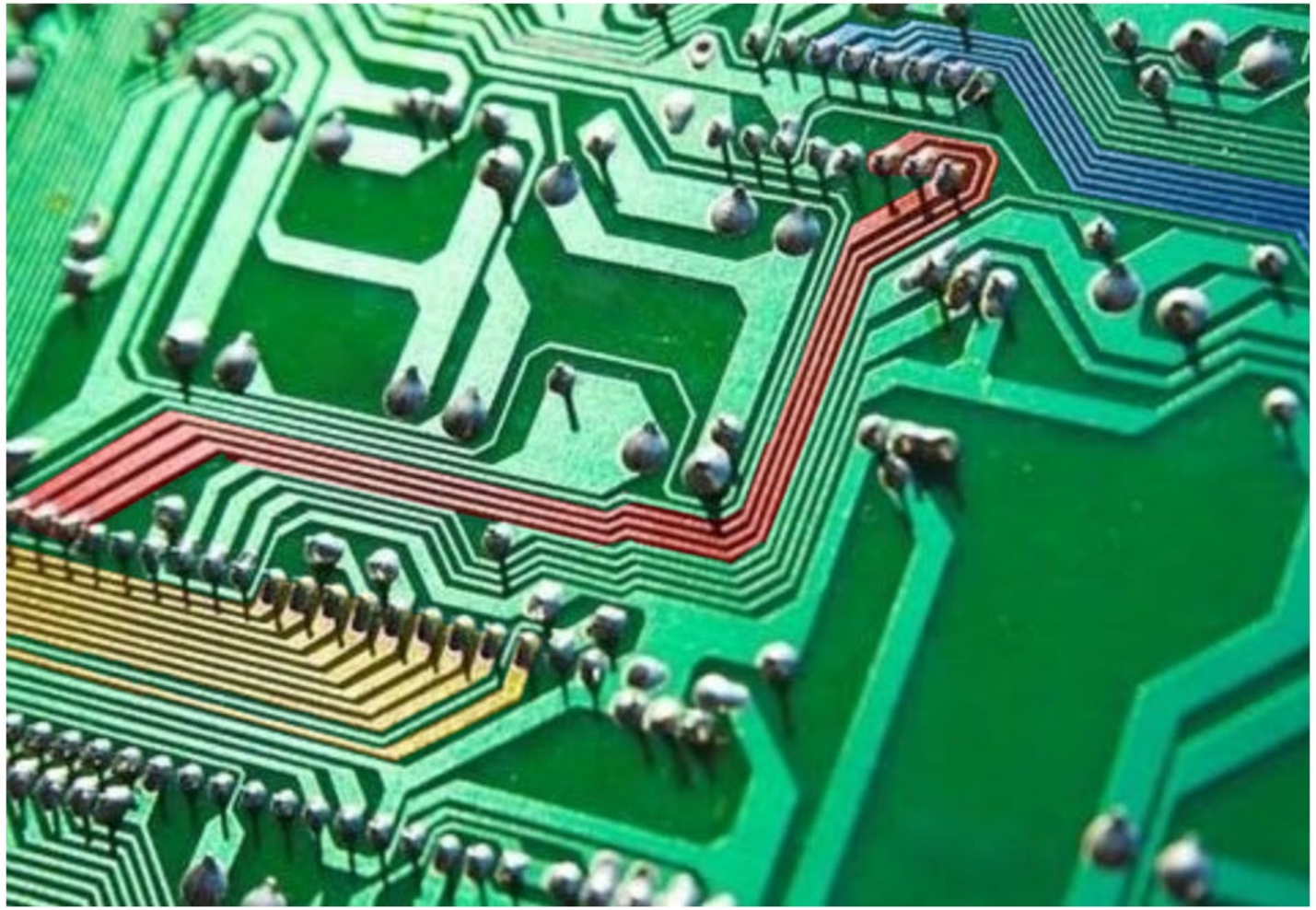
v: Vorzeichen → 1 Bit

m: Mantisse → Single:23Bit, Double:52Bit

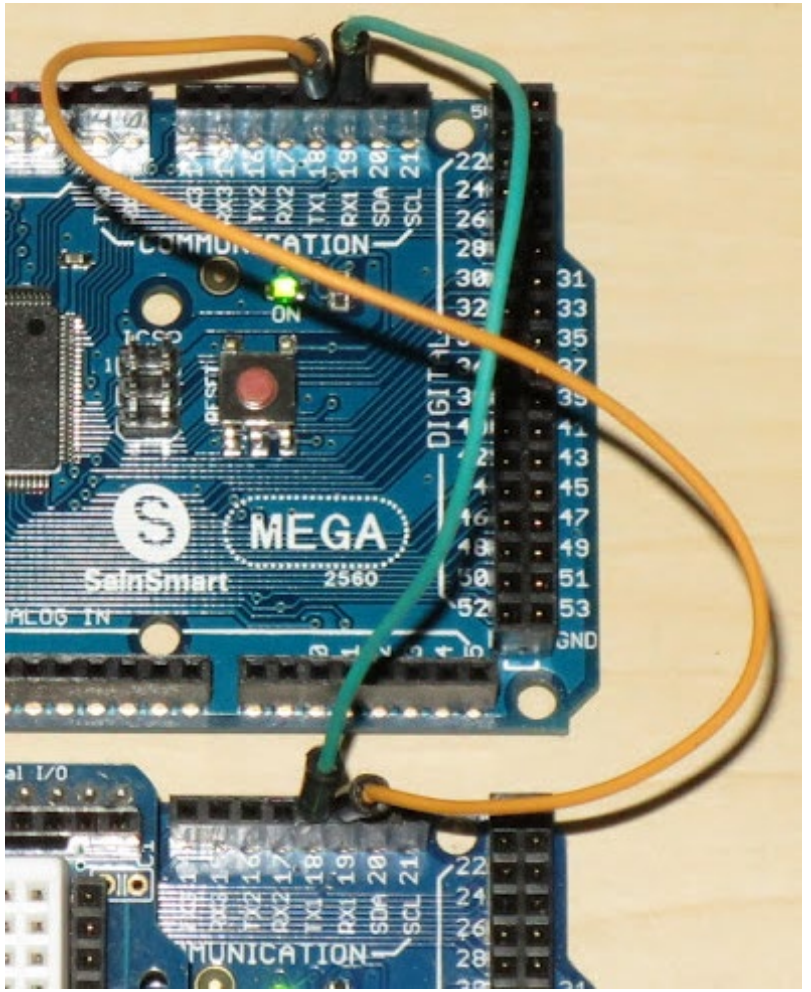
b: Basis → Bei normalisierten Gleitkommazahlen ist b=2

e: Exponent → Single:8Bit, Double:11Bit

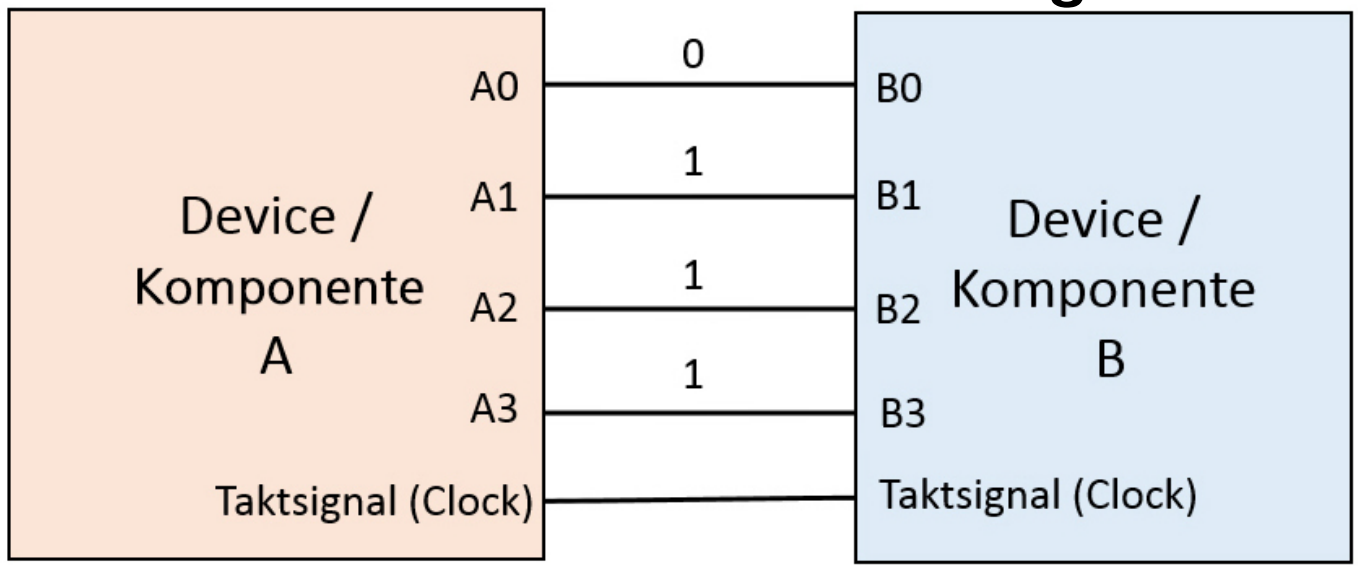
Parallele Datenverbindung



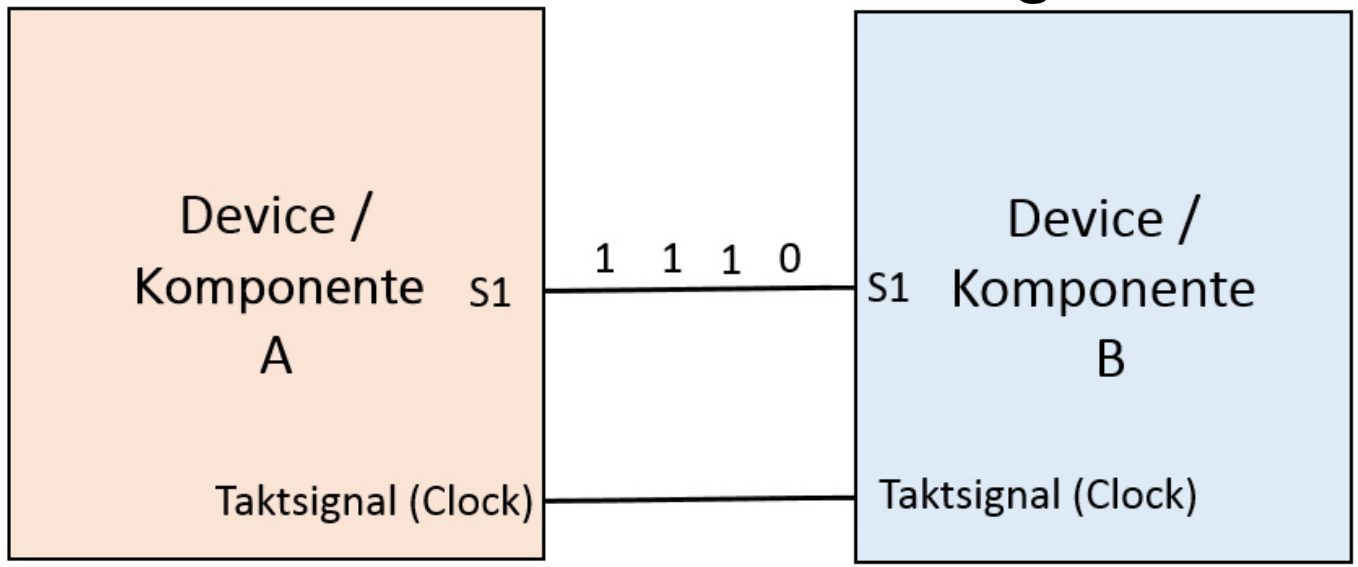
Serielle Datenverbindung



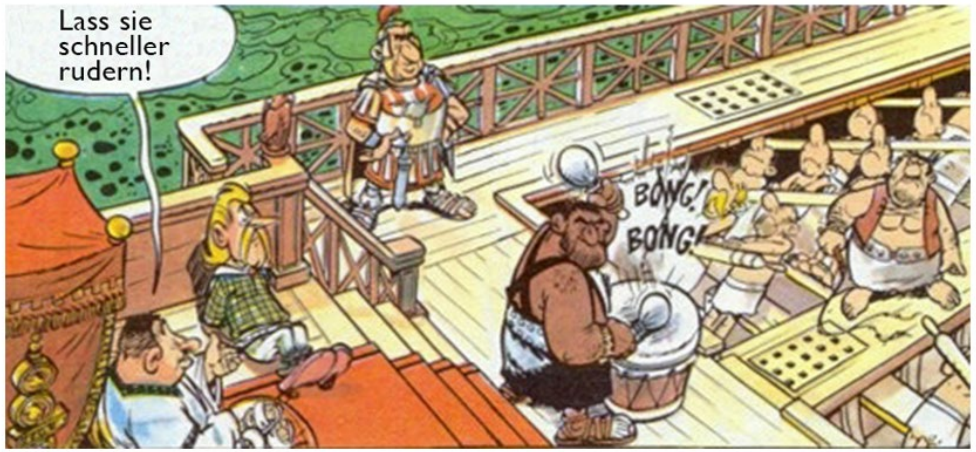
Parallele Datenverbindung

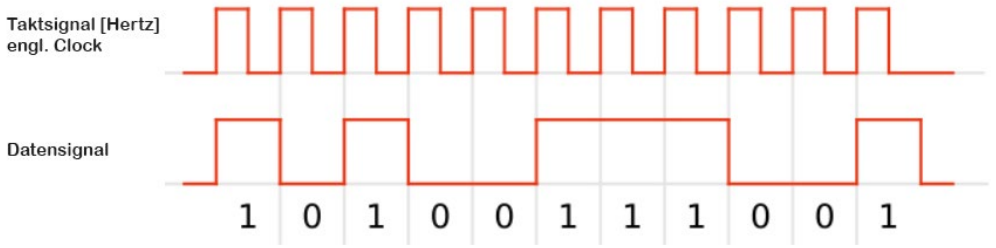


Serielle Datenverbindung

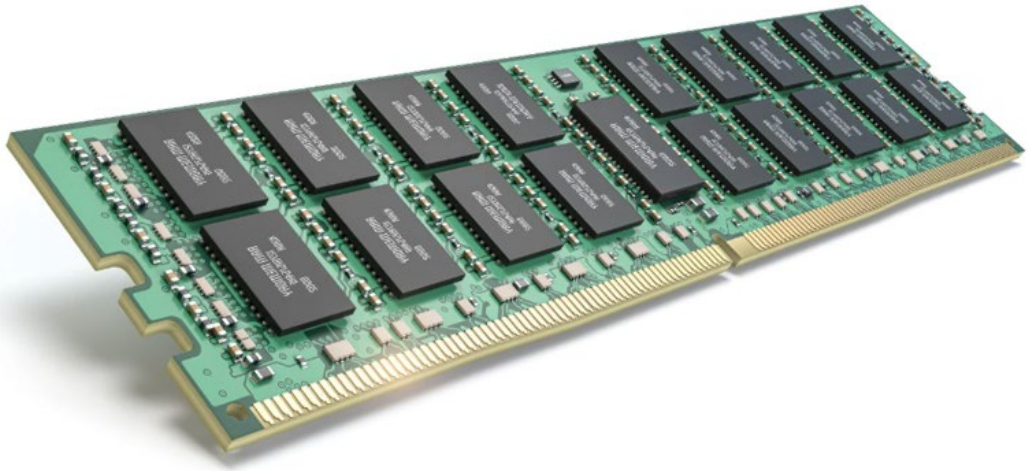


Taktsignal (Rechtecksignal)





Primärspeicher (Flüchtig / RAM)



Adressbus 4 Bit
0..15 (16 Speicherstellen)

Adresse								
1111	0	1	0	1	1	1	1	0
0010	1	0	0	1	1	0	1	0
0001	0	1	0	1	1	0	0	0
0000	1	0	0	1	1	0	1	0

Kapazität des Speichers:
16 x 8 bit = 128 bit
oder
16 x 1 Byte = 16 Byte

Datenbus 8 Bit
1 Byte pro
Speicherstelle

Dies ist der Grund dafür, Warum Speichergrössen mit Binärpräfixen (kiB, MiB etc.) angegeben werden.

Aufgaben: **Zeit: 6 Minuten**

1. Welche Speicherkapazität in kiB (Hinweis: 1kiB=1024B) besitzt ein Arbeitsspeicher mit 12-Bit-Adressbus und 16-Bit-Datenbus?
2. Zwei Geräte sind mit einer seriellen Leitung und einer Leitung für das Taktsignal von 1MHz verbunden.
   a. Wie viele Bytes können pro Sekunde übertragen werden?
   b. Wie viele Bytes pro Sekunde können übertragen werden, wenn die Verbindung der beiden Geräte nicht seriell, sondern 8 Bit-parallel wäre?

Musterlösungen:

1. Welche Speicherkapazität in kiB (Hinweis: 1kiB=1024B) besitzt ein Arbeitsspeicher mit 12-Bit-Adressbus und 16-Bit-Datenbus?
   12Bit Adresse ergibt 2^{12} oder 4096 Speicherplätze. Pro Speicherplatz 2B ergibt 8192Byte. In KiBi ausgedrückt: 8192/1024= 8kiBi
2. Zwei Geräte sind mit einer seriellen Leitung und einer Leitung für das Taktsignal von 1MHz verbunden.
   a. Wie viele Bytes können pro Sekunde übertragen werden?
      1MHz = 1'000'000 Hz. Somit 1'000’000Bit pro Sekunde oder 125kB/s
   b. Wie viele Bytes pro Sekunde können übertragen werden, wenn die Verbindung der beiden Geräte nicht seriell, sondern 8 Bit-parallel wäre?
      8x mehr. Somit 1MB/s

Codierung von Text (Alphanumerische Codes)

Serielle Leitung

Telexgerät
(ca. 1963)

ASCII-Code (1963)
American
Standard
Code for
Information
Interchange

Pro Zeichen ursprünglich: 7 Bit

DEC	HEX	BIN	CHAR	BEZEICHNUNG
000	00	00000000	NUL	Null Character
001	01	00000001	SOH	Start of Header
002	02	00000010	STX	Start of Text
003	03	00000011	ETX	End of Text
004	04	00000100	EOT	End of Transmission
005	05	00000101	ENQ	Enquiry
006	06	00000110	ACK	Acknowledgment
007	07	00000111	BEL	Bell
008	08	00001000	BS	Backspace
009	09	00001001	HT	Horizontal Tab
010	0A	00001010	LF	Line Feed
011	0B	00001011	VT	Vertical Tab
012	0C	00001100	FF	Form Feed
013	0D	00001101	CR	Carriage Return
014	0E	00001110	SO	Shift Out
015	0F	00001111	SI	Shift In
016	10	00010000	DLE	Data Link Escape
017	11	00010001	DC1	XON Device Control 1
018	12	00010010	DC2	Device Control 2
019	13	00010011	DC3	XOFF Device Control 3
020	14	00010100	DC4	Device Control 4
021	15	00010101	NAK	Negative Acknowledgement
022	16	00010110	SYN	Synchronous Idle
023	17	00010111	ETB	End of Transmission Bloc
024	18	00011000	CAN	Cancel
025	19	00011001	EM	End of Medium
026	1A	00011010	SUB	Substitute
027	1B	00011011	ESC	Escape
028	1C	00011100	FS	File Separator
029	1D	00011101	GS	Group Separator
030	1E	00011110	RS	Request to Send Record Separator
031	1F	00011111	US	Unit Separator

DEC	HEX	BIN	CHAR	BEZEICHNUNG
032	20	00100000	SP	Space
033	21	00100001	!	Exclamation mark
034	22	00100010	"	Double quote
035	23	00100011	#	Number sign
036	24	00100100	$	Dollar sign
037	25	00100101	%	Percent
038	26	00100110	&	Ampersand
039	27	00100111	'	Single quote
040	28	00101000	(	Left opening parenthesis
041	29	00101001	)	Right closing parenthesis
042	2A	00101010	*	Asterisk
043	2B	00101011	+	Plus
044	2C	00101100	,	Comma
045	2D	00101101	-	Minus or dash
046	2E	00101110	.	Dot
047	2F	00101111	/	Forward slash
048	30	00110000	0	
049	31	00110001	1	
050	32	00110010	2	
051	33	00110011	3	
052	34	00110100	4	
053	35	00110101	5	
054	36	00110110	6	
055	37	00110111	7	
056	38	00111000	8	
057	39	00111001	9	
058	3A	00111010	:	Colon
059	3B	00111011	;	Semi-colon
060	3C	00111100	<	Less than sign
061	3D	00111101	=	Equal sign
062	3E	00111110	>	Greater than sign
063	3F	00111111	?	Question mark

DEC	HEX	BIN	CHAR	BEZEICHNUNG
064	40	01000000	@	AT symbol
065	41	01000001	A	
066	42	01000010	B	
067	43	01000011	C	
068	44	01000100	D	
069	45	01000101	E	
070	46	01000110	F	
071	47	01000111	G	
072	48	01001000	H	
073	49	01001001	I	
074	4A	01001010	J	
075	4B	01001011	K	
076	4C	01001100	L	
077	4D	01001101	M	
078	4E	01001110	N	
079	4F	01001111	O	
080	50	01010000	P	
081	51	01010001	Q	
082	52	01010010	R	
083	53	01010011	S	
084	54	01010100	T	
085	55	01010101	U	
086	56	01010110	V	
087	57	01010111	W	
088	58	01011000	X	
089	59	01011001	Y	
090	5A	01011010	Z	
091	5B	01011011	[	Left opening bracket
092	5C	01011100	\	Back slash
093	5D	01011101	]	Right closing bracket
094	5E	01011110	^	Caret cirumflex
095	5F	01011111	_	Underscore

DEC	HEX	BIN	CHAR	BEZEICHNUNG
096	60	01100000	`	
097	61	01100001	a	
098	62	01100010	b	
099	63	01100011	c	
100	64	01100100	d	
101	65	01100101	e	
102	66	01100110	f	
103	67	01100111	g	
104	68	01101000	h	
105	69	01101001	i	
106	6A	01101010	j	
107	6B	01101011	k	
108	6C	01101100	l	
109	6D	01101101	m	
110	6E	01101110	n	
111	6F	01101111	o	
112	70	01110000	p	
113	71	01110001	q	
114	72	01110010	r	
115	73	01110011	s	
116	74	01110100	t	
117	75	01110101	u	
118	76	01110110	v	
119	77	01110111	w	
120	78	01111000	x	
121	79	01111001	y	
122	7A	01111010	z	
123	7B	01111011	{	Left opening brace
124	7C	01111100	|	Vertical bar
125	7D	01111101	}	Right closing brace
126	7E	01111110	~	Tilde
127	7F	01111111	DEL	Delete

Ausgangslage ist ASCII-7Bit erweitert um 1 Bit auf 8 Bit.

Zeichen 128 bis 255 gemäss dem ISO-8859-Standard (Zeichentabelle)

ISO-Standard 8859-1	Latin-1, Westeuropäisch oder ANSI-ASCI, EBCDIC ANSI-ASCII (ANSI: American National Standards Institute)
ISO-Standard 8859-2	Latin-2, Mitteleuropäisch
ISO-Standard 8859-3	Latin-3, Südeuropäisch
ISO-Standard 8859-4	Latin-4, Nordeuropäisch
ISO-Standard 8859-5	Kyrillisch
ISO-Standard 8859-6	Arabisch
ISO-Standard 8859-7	Griechisch
ISO-Standard 8859-8	Hebräisch
ISO-Standard 8859-9	Latin-5, Türkisch
ISO-Standard 8859-10	Latin-6, Nordisch
ISO-Standard 8859-11	Thai
ISO-Standard 8859-12	existiert nicht
ISO-Standard 8859-13	Latin-7, Baltisch
ISO-Standard 8859-14	Latin-8, Keltisch
ISO-Standard 8859-15	Latin-9, Westeuropäisch
ISO-Standard 8859-16	Latin-10, Südosteuropäisch

Unicode, der ANSI-ASCII-Nachfolger

- Maximal **8 Byte** (=64 Bit)
  Theoretisch `U+xxxx'xxxx'xxxx'xxxx`

- **Unicode V2.0** nutzt **1‘114‘112** verschiedene Codepunkte
  `U+0000'0000 bis U+0010'FFFF`
  `V16.0 (Sept. 2024) 154'998 verschiedene Zeichen, 168 Schriftsysteme`

Unicode Transformations Formate

- **UTF-8** belegt pro Zeichen 1, 2, 3 oder 4 Byte

- **UTF-16** belegt pro Zeichen 2 oder 4 Byte

- UTF = UCS Transformation Format, wobei UCS = Universal-Coded-Character-Set

UTF-8

Unicode-Bereich (Hexadez.)	UTF-8 Kodierung (Binär)	Bemerkungen
0000 0000 – 0000 007F	0xxx xxxx	In diesem Bereich (128 Zeichen) entspricht UTF-8 genau dem ASCII-Code: Das höchste Bit ist *0*, die restliche 7-Bit-Kombination ist das ASCII-Zeichen
0000 0080 – 0000 07FF	110xxxxx 10xxxxxx	Das erste Byte enthält binär 11xxxxxx, die folgenden Bytes 10xxxxxx; die *x* stehen für die fortlaufende Bitkombination des Unicode-Zeichens. Die Anzahl der Einsen vor der höchsten 0 im ersten Byte ist die Anzahl der Bytes für das Zeichen. (In Klammern jeweils die theoretisch maximal möglichen.)
0000 0800 – 0000 FFFF	1110xxxx 10xxxxxx 10xxxxxx	
0001 0000 – 0010 FFFF [0001 0000 – 001F FFFF]	11110xxx 10xxxxxx 10xxxxxx 10xxxxxx	

UTF-8

Zeichen	Unicode	Unicode (Binär)	UTF-8 (Binär)	UTF-8 (Hexadez.)
Buchstabe *y*	U+0079	00000000 01111001	01111001	0x79
Buchstabe *ä*	U+00E4	00000000 1110 0100	11000011 1010 0100	0xC3 0xA4
Zeichen für eingetragene Marke ®	U+00AE	0000 0000 1010 1110	11000010 1010 1110	0xC2 0xAE
Eurozeichen €	U+20AC	0010 0000 1010 1100	11100010 10000010 10101100	0xE2 0x82 0xAC

UTF-16

- Ein UTF-16 Zeichen belegt in unserem Sprachraum 16 Bit oder 2 Byte
  Exotische Zeichen können auch 4 Byte belegen.

- Je nach Byteorder führt das höherwertige Byte oder das niederwertige Byte den UTF-16-Code an.
  (Byteorder: Etwas Ähnlichem begegnen wir bei der Datumsangabe:
  Feb-23 (höherwertiges zuerst)
  23-Feb (niederwertiges zuerst)

- Um sich bei der Byteorder sicher zu sein, wird ein UTF-16 Code mit einer Codesequenz eingeleitet:
  FE FF: BE Big Endian
  FF FE: LE Little Endian

Sie haben nun erfahren, wie UTF-8 funktioniert.
Das zweite, wichtige Unicode Transformationsformat ist UTF-16.

Erfahren sie nun hier im Selbststudium, wie UTF-16 funktioniert:

https://de.wikipedia.org/wiki/UTF-16

Beantworten sie nun die folgenden Kontrollfragen: **Zeit: 10 Minuten**

1. Wo wird UTF8 eingesetzt, wo UTF-16?
2. Was bedeutet die Byteorder bei UTF-16?
3. Was bedeutet «abwärtskompatibel» im Zusammenhang mit UTF-8? Existiert diese «Abwärtskompatibilität» auch bei UTF-16?
4. Wie beurteilen sie den Speicherbedarf von deutsch- oder englischsprachigem Text bei ISO-8859-Codierung, UTF-8-Codierung und UTF-16-Codierung?
5. Wann wird ein UTF-16-Code vier Byte lang?
6. Wie müssen sie vorgehen, wenn sie z.B. in Word ein Zeichen eingeben wollen, das sie auf der Tastatur zwar nicht finden, dessen Unicode sie aber kennen? (z.B. das Eurozeichen)
7. Was wird das Problem sein, wenn sie auf dieselbe Weise wie in der vorangegangenen Aufgabe das Unicode-Zeichen des Violinschlüssels (Musiknoten) in eine Wordtext einfügen wollen?

Lösungen zu den Kontrollfragen:

1. Wo wird UTF8 eingesetzt, wo UTF-16?
   UTF8: Internetprotokolle, UTF16: .Net-Framework, Java, Tcl
2. Was bedeutet die Byteorder bei UTF-16?
   Big Endian (UTF-16BE), Little Endian (UTF-16LE)
   *UTF16-BE-BOM: **FE FF**… UTF16-LE-BOM: **FF FE**… Siehe auch Beispiel im Anschluss!*
3. Was bedeutet «abwärtskompatibel» im Zusammenhang mit UTF-8? Existiert diese «Abwärtskompatibilität» auch bei UTF-16?
   Die ersten 128 Unicodezeichen (U+0000 bis U+007F entsprechen den Positionen 0–127 in allen ISO-8859-Varianten (7-Bit-ASCII). In UTF-8 deckungsgleich als ein Byte dargestellt.
   Bei UTF16 existiert dies nicht.
4. Wie beurteilen sie den Speicherbedarf von deutsch- oder englischsprachigem Text bei ISO-8859-Codierung, UTF-8-Codierung und UTF-16-Codierung?
   ISO-8859-Codierung und UTF-8-Codierung ist effizienter, weil UTF16 immer 2Byte belegt.
5. Wann wird ein UTF-16-Code vier Byte lang?
   Unicode-Zeichen außerhalb der BMP (U+10000 bis U+10FFFF). BMP=Basic Multilingual Plane
6. Wie müssen sie vorgehen, wenn sie z.B. in Word ein Zeichen eingeben wollen, das sie auf der Tastatur zwar nicht finden, dessen Unicode sie aber kennen? (z.B. das Eurozeichen)
   U+20AC eingeben, danach ALT-C drücken.
7. Was wird das Problem sein, wenn sie auf dieselbe Weise wie in der vorangegangenen Aufgabe das Unicode-Zeichen des Violinschlüssels (`U+1D11E`) in eine Wordtext einfügen wollen?
   Gewählter Font-Satz wie z.B. Arial enthält Violinzeichen nicht.

Aufgaben: **Zeit: 9 Minuten** **Hilfsmittel: Notepad++ und hexed.it**

Untersuchen sie den Textstring **€URO**. Beachten sie, dass es sich beim ersten Zeichen nicht um ein **E** sondern um das Eurozeichen **€** handelt.

1. Wie lautet der **ANSI-ASCII**-Code für das Eurozeichen €?
2. Wie lautet der **Unicode** für das Eurozeichen €?
3. Schreiben sie den **ANSI-ASCII** Text €URO in HEX und in Binär.
4. Schreiben sie den **UTF-8** Text €URO in HEX und in Binär.
5. Schreiben sie den **UTF-16-BE** Text €URO in HEX und in Binär.
6. Schreiben sie den **UTF-16-LE** Text €URO in HEX und in Binär.
7. Vergleichen sie die Resultate.

Musterlösungen:

Untersuchen sie den Textstring **€URO**. Beachten sie, dass es sich beim ersten Zeichen nicht um ein **E** sondern um das Eurozeichen **€** handelt.

1. Wie lautet der **ANSI-ASCII**-Code für das Eurozeichen €? 0x80 oder 1000’0000
2. Wie lautet der **Unicode** für das Eurozeichen €? U+20AC oder 0010’0000 1010’1100
3. Schreiben sie den **ANSI-ASCII** Text €URO in HEX und in Binär.

€		U		R		O	
8	0	5	5	5	2	4	F
1000	0000	0101	0101	0101	0010	0100	1111

4. Schreiben sie den **UTF-8** Text €URO in HEX und in Binär.

€				U		R		O	
E	2	8	2	A	C	5	5	5	2	4	F
1110	0010	1000	0010	1010	1100	0101	0101	0101	0010	0100	1111

5. Schreiben sie den **UTF-16-BE** Text €URO in HEX und in Binär.

BE-16	€			U				R				O			
FE	FF	2	0	A	C	0	0	5	5	0	0	5	2	0	0	4	F
		0010	0000	1010	1100	0000	0000	0101	0101	0000	0000	0101	0010	0000	0000	0100	1111

6. Schreiben sie den **UTF-16-LE** Text €URO in HEX und in Binär.

LE-16	€			U				R				O			
FF	FE	A	C	2	0	5	5	0	0	5	2	0	0	4	F	0	0
		1010	1100	0010	0000	0101	0101	0000	0000	0101	0010	0000	0000	0100	1111	0000	0000

7. Vergleichen sie die Resultate.
   Bis auf das Eurozeichen sind ANSI-ASCII und UTF-8 identisch.
   Bei BE-16 und LE-16 unterscheiden sich bei FE-FF und pro 16-Bit-Charakter die Byte-Order.