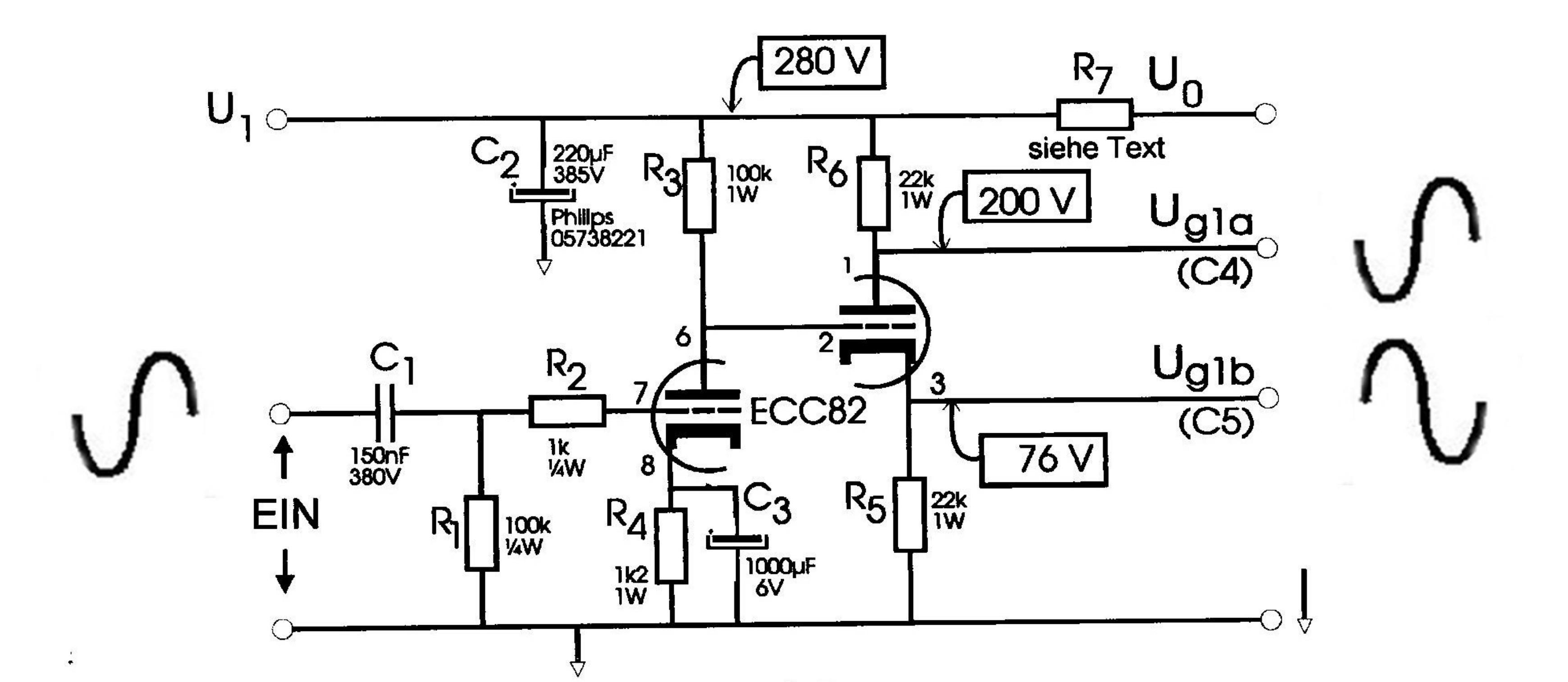
Analysis of the WAVE file format in order to generate test signals

Motivation: tube amplifier

-Creation of a asymmetric test signal to trace electronic circuits

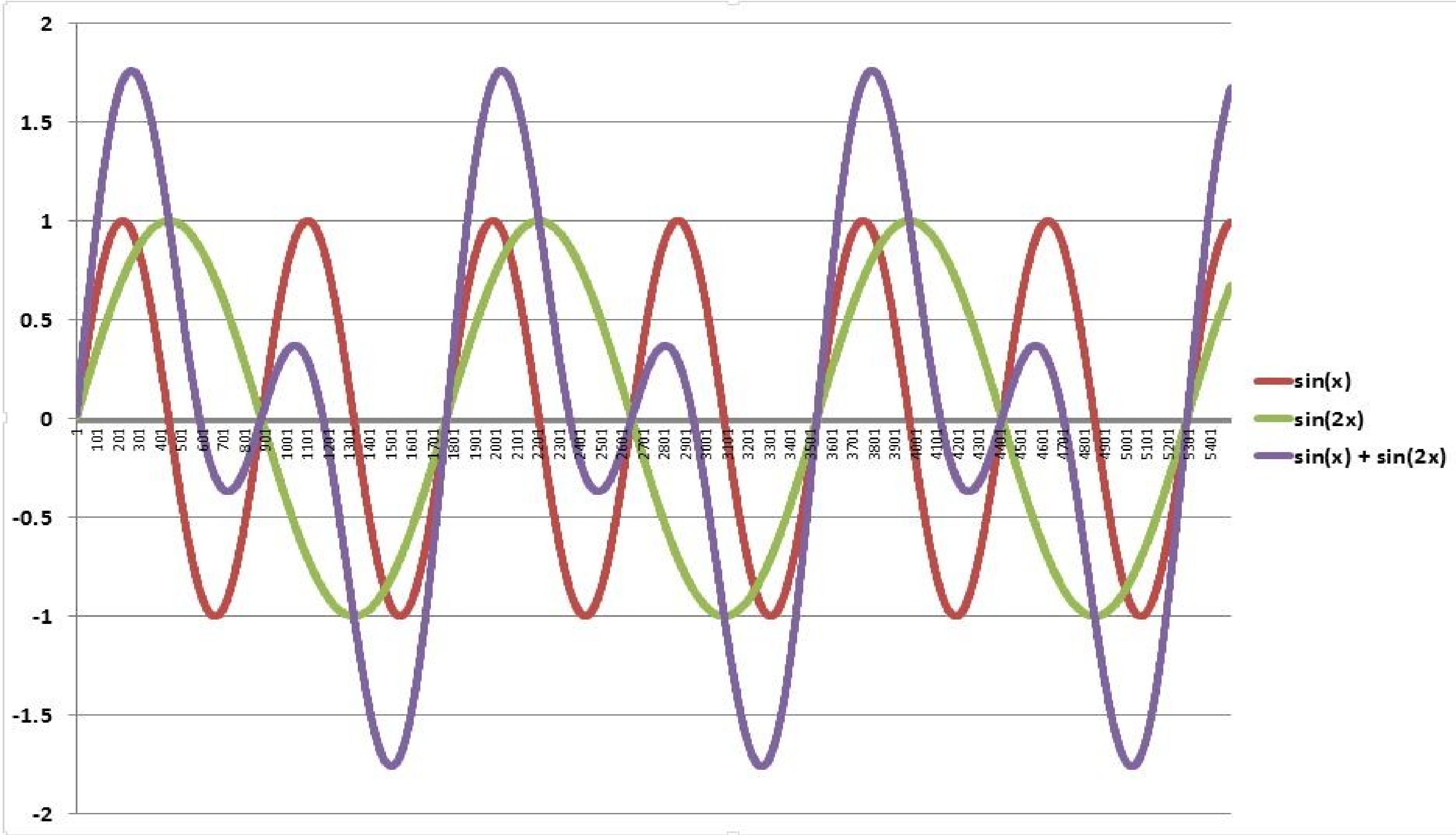
Requirement: asymmetric signal to demonstrate function of phase splitter



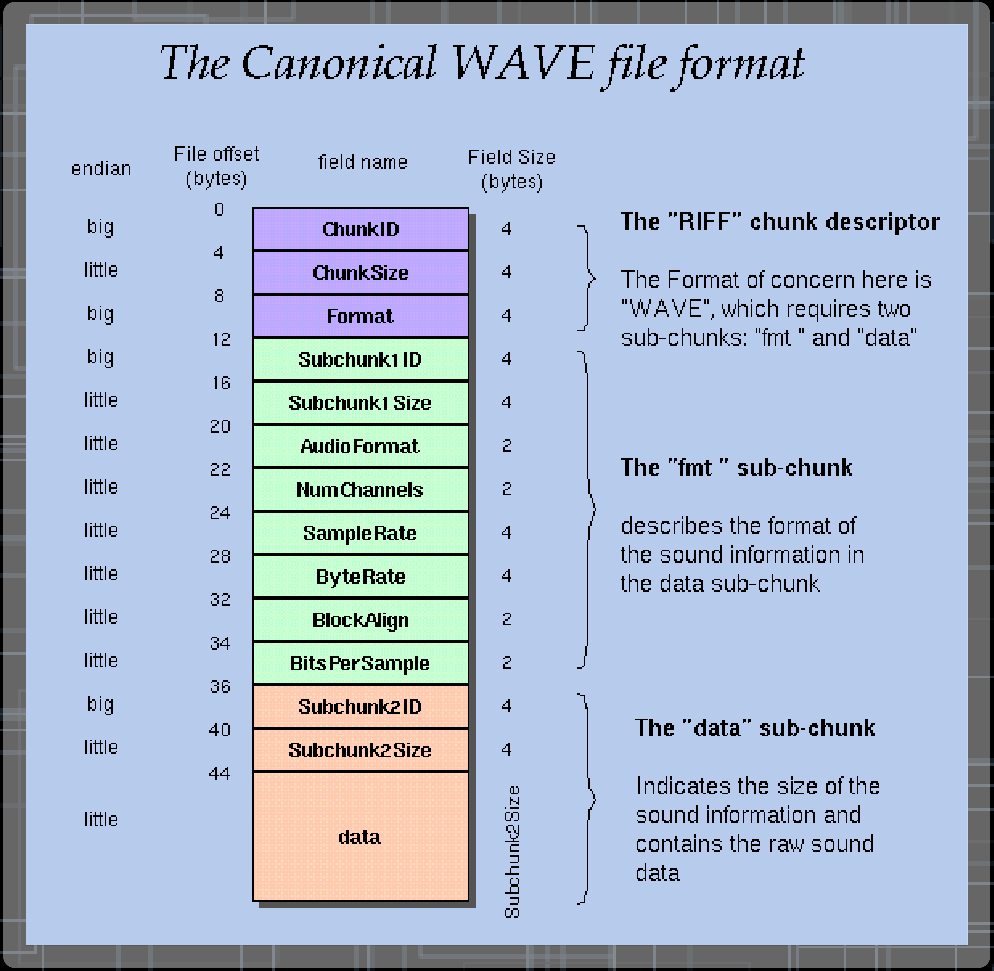
Plan

- Analyze WAVE file format
- Calculate signal by adding multiple sine curves
- Write script to generate WAVE file
- Use pc sound card to play file to provide test signal

Adding multiple sine curves



Websearch



The Canonical WAVE file format

endian	File offset (bytes)	field name	Field Size (bytes)
big	0	ChunkID	4
little	4	ChunkSize	4
big	8	Format	4
big	12	Subchunk1 ID	4
little	16	Subchunk1 Size	4
little	20	AudioFormat	2
little	22	NumChannels	2
little	24	SampleRate	4
little	28	ByteRate	4
little	32	BlockAlign	2
little	34	BitsPerSample	2
big	36	Subchunk2ID	4
little	40	Subchunk2Size	4
little	44	data	Subchunk2Size

The "RIFF" chunk descriptor

The Format of concern here is "WAVE", which requires two sub-chunks: "fmt " and "data"

The "fmt " sub-chunk

describes the format of the sound information in the data sub-chunk

The "data" sub-chunk

Indicates the size of the sound information and contains the raw sound data

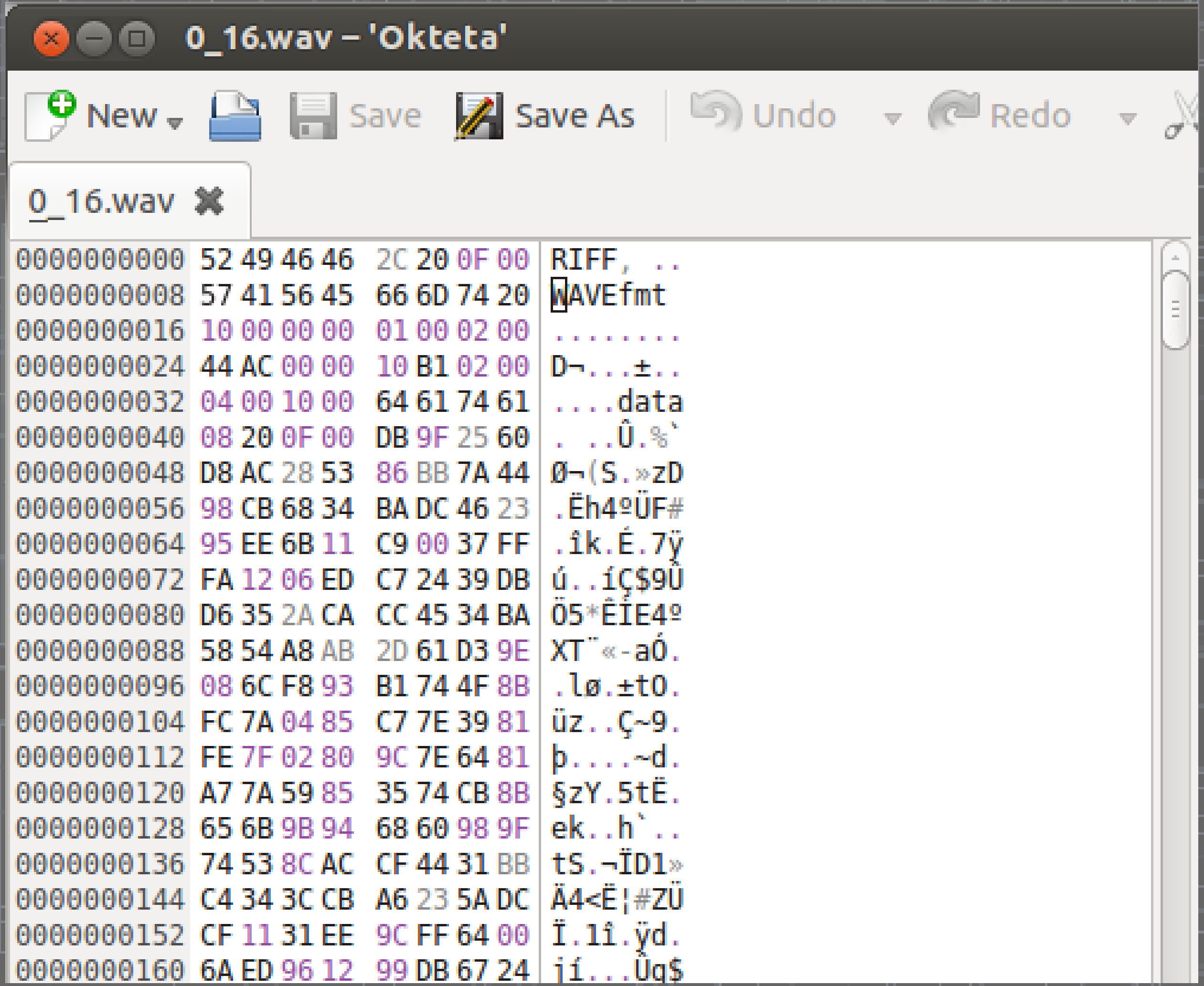
Analysis of an existing .wav file

- 5.3 second long 1khz 16 bit stereo sine wave, 0 dBFS („0 decibel full scale“ meaning: using maximal possible amplitude at sine wave peak), 44.1 kHz sampling rate

- http://www.rme-audio.de/old/download/audtest.htm

- Hex file editor

Display of test file in a hex editor



```
0000000000 52 49 46 46  2C 20 0F 00  RIFF, ..
0000000008 57 41 56 45  66 6D 74 20  WAVEfmt 
0000000016 10 00 00 00  01 00 02 00  ........
0000000024 44 AC 00 00  10 B1 02 00  D¬...±..
0000000032 04 00 10 00  64 61 74 61  ....data
0000000040 08 20 0F 00  DB 9F 25 60  . ..Û.%`
0000000048 D8 AC 28 53  86 BB 7A 44  Ø¬(S.»zD
0000000056 98 CB 68 34  BA DC 46 23  .Ëh4ºÜF#
0000000064 95 EE 6B 11  C9 00 37 FF  .îk.É.7ÿ
0000000072 FA 12 06 ED  C7 24 39 DB  ú..íÇ$9Û
0000000080 D6 35 2A CA  CC 45 34 BA  Ö5*ÊÌE4º
0000000088 58 54 A8 AB  2D 61 D3 9E  XT¨«-aÓ.
0000000096 08 6C F8 93  B1 74 4F 8B  .lø.±tO.
0000000104 FC 7A 04 85  C7 7E 39 81  üz..Ç~9.
0000000112 FE 7F 02 80  9C 7E 64 81  þ....~d.
0000000120 A7 7A 59 85  35 74 CB 8B  §zY.5tË.
0000000128 65 6B 9B 94  68 60 98 9F  ek..h`..
0000000136 74 53 8C AC  CF 44 31 BB  tS.¬ÏD1»
0000000144 C4 34 3C CB  A6 23 5A DC  Ä4<Ë¦#ZÜ
0000000152 CF 11 31 EE  9C FF 64 00  Ï.1î.ÿd.
0000000160 6A ED 96 12  99 DB 67 24  jí...Ûg$
```

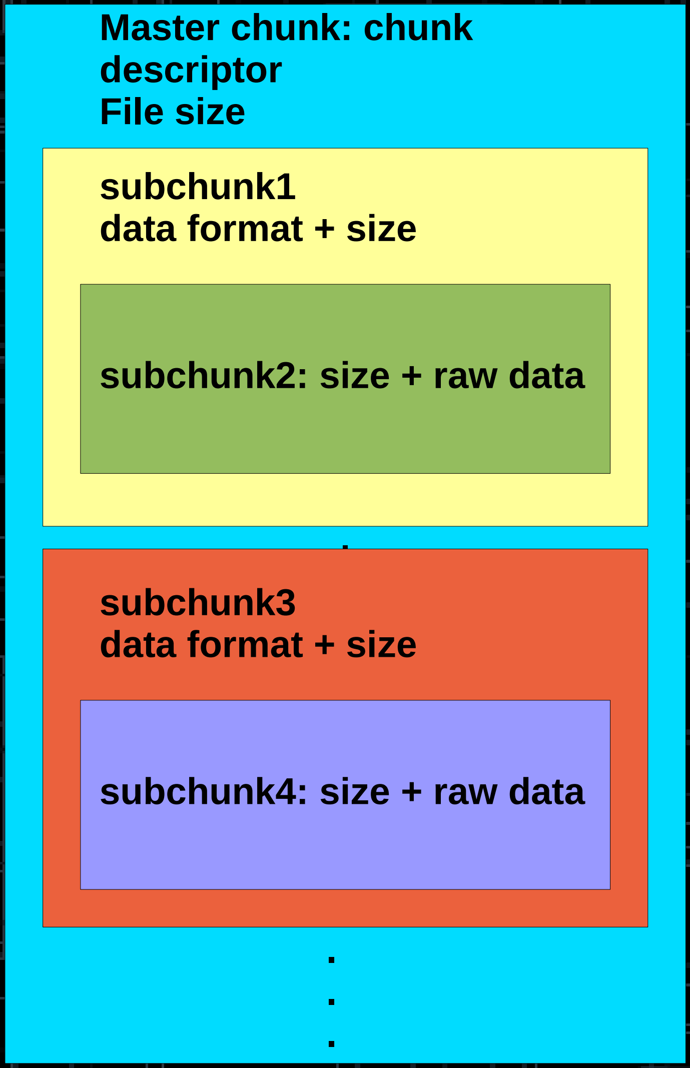
RIFF: Resource Interchange File Format

- Developed by Microsoft + IBM 1991

  generic container format for storage of multimedia data
- A RIFF file is composed of multiple discrete sections of data called chunks
- Chunk = Brocken / Klumpen

RIFF: Container format for multimedia data

logical structure



Master chunk: chunk descriptor
File size

subchunk1
data format + size

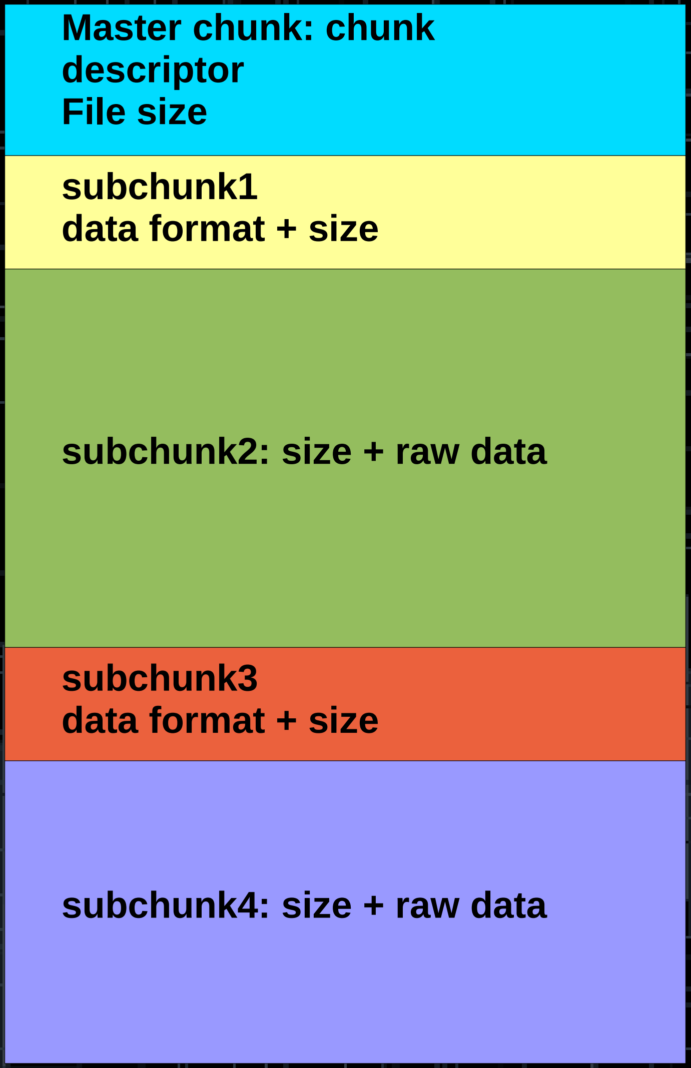
subchunk2: size + raw data

subchunk3
data format + size

subchunk4: size + raw data

.
.
.

implementation



Master chunk: chunk descriptor
File size

subchunk1
data format + size

subchunk2: size + raw data

subchunk3
data format + size

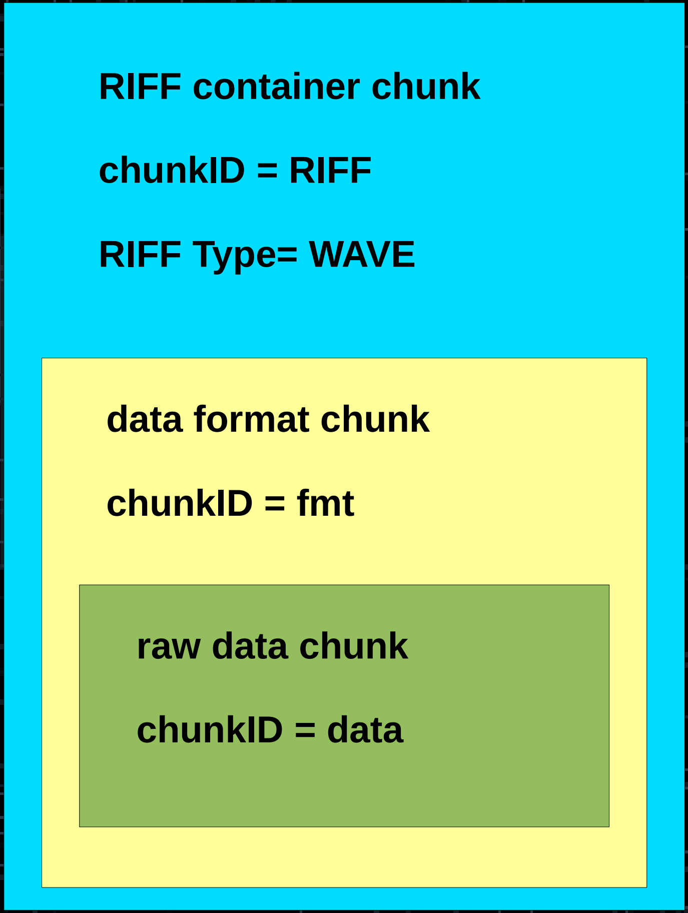
subchunk4: size + raw data

WAVE: implementation of RIFF

- WAVE: Waveform Audio File Format
- Implementation of RIFF for audio data
- Consists of three chunks
- RIFF container chunk
- fmt data format chunk
- data chunk with raw data
- Pulse code modulated data (PCM)

WAVE file structure

logical structure



RIFF container chunk

chunkID = RIFF

RIFF Type= WAVE

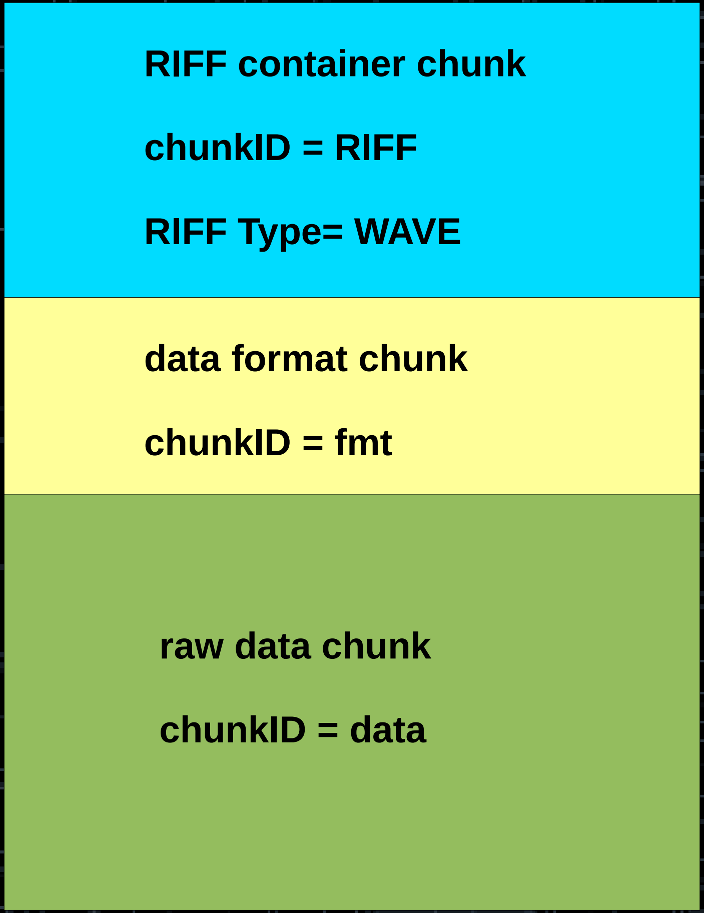
data format chunk

chunkID = fmt

raw data chunk

chunkID = data

implementation



RIFF container chunk

chunkID = RIFF

RIFF Type= WAVE

data format chunk

chunkID = fmt

raw data chunk

chunkID = data

What is Endianess or Byte Order ?

- Numbers exeeding memory register width (8bit) need to be distributed over several registers
- Byte order: order by which several one byte segments are read into 8bit memory

About byte order

32 bit decimal integer 439'041'101

in binary is: 11010001010110011110001001101

binary	00011010	00101011	00111100	01001101
hexadecimal	1A	2B	3C	4D

Big endian

→

1A 2B 3C 4D

Little endian

←

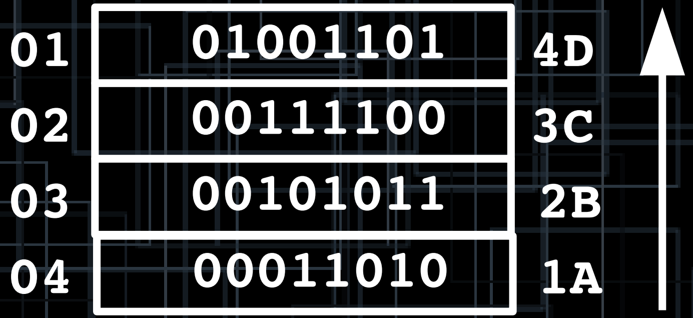
4D 3C 2B 1A

About byte order

8 Bit Memory Register

Little Endian

4D 3C 2B 1A



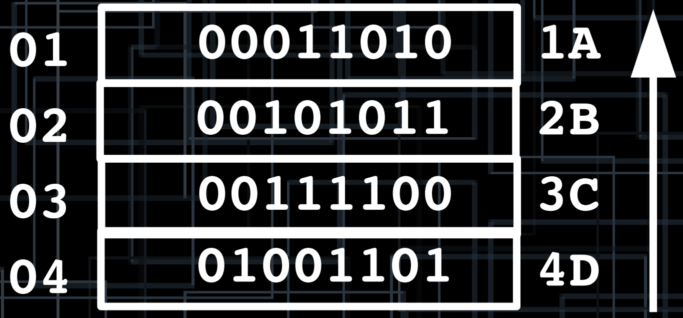
01	01001101	4D
02	00111100	3C
03	00101011	2B
04	00011010	1A

least significant byte

at samllest address

Big Endian

1A 2B 3C 4D

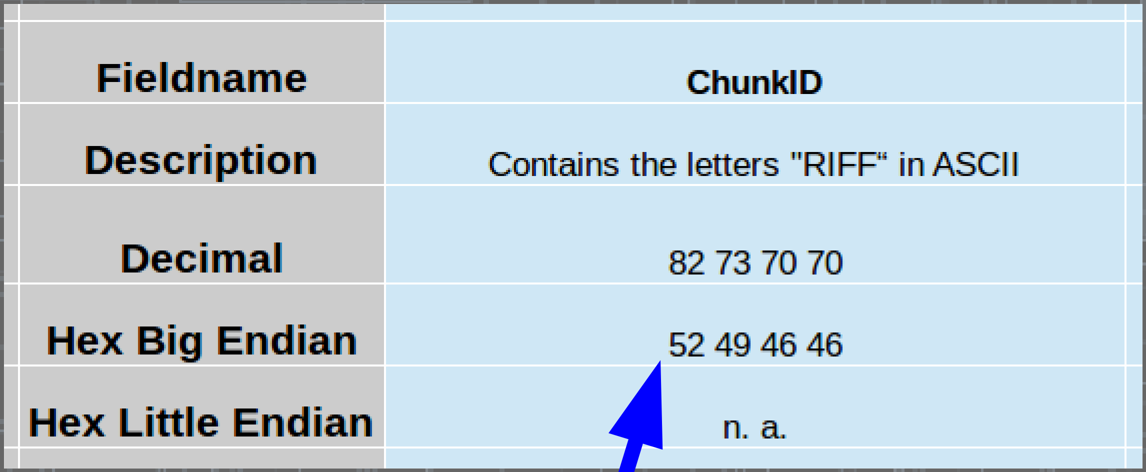


01	00011010	1A
02	00101011	2B
03	00111100	3C
04	01001101	4D

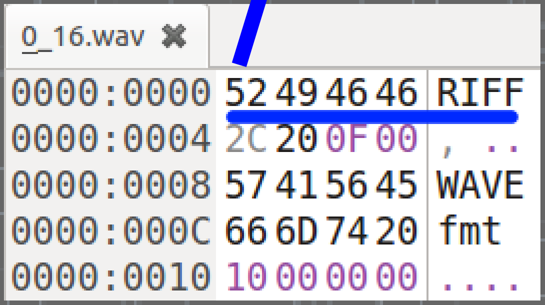
most significant byte

at samllest address

Master chunk: ChunkID



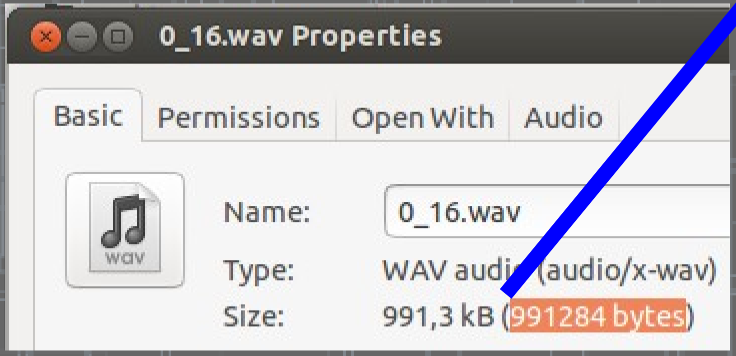
Fieldname	**ChunkID**
Description	Contains the letters "RIFF" in ASCII
Decimal	82 73 70 70
Hex Big Endian	52 49 46 46
Hex Little Endian	n. a.



```
0000:0000 52 49 46 46 RIFF
0000:0004 2C 20 0F 00 , ..
0000:0008 57 41 56 45 WAVE
0000:000C 66 6D 74 20 fmt
0000:0010 10 00 00 00 ....
```

Master chunk: ChunkSize

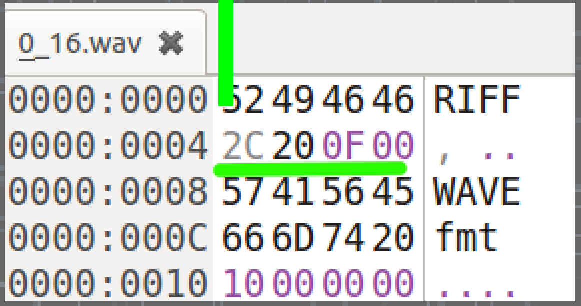
Fieldname	ChunkSize
Description	Size of the entire file in bytes minus 8 bytes of the leading two fields
Decimal	991'284 – 8 = 991'276
Hex Big Endian	00 0F 20 2C
Hex Little Endian	2C 20 0F 00



Name: 0_16.wav

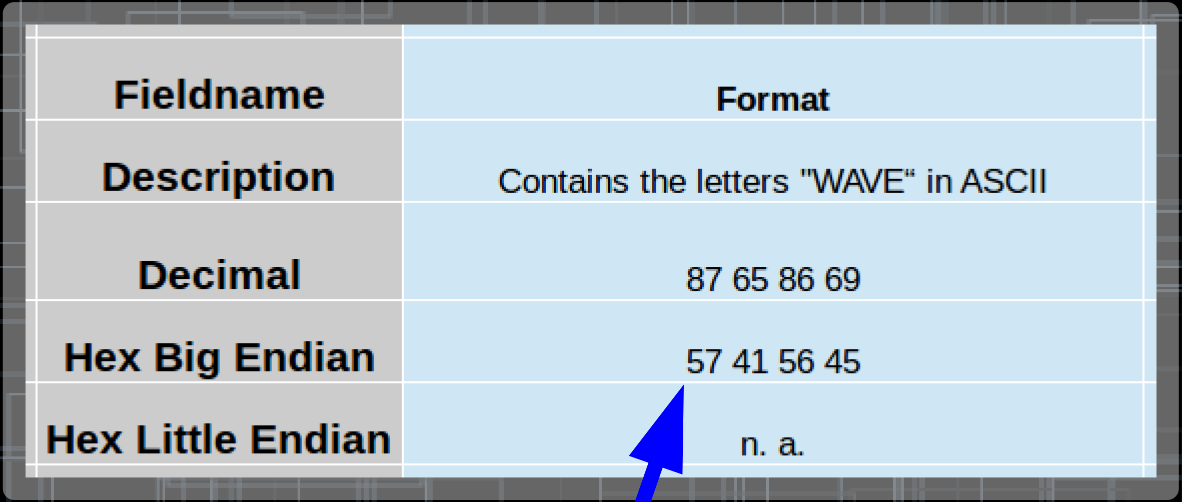
Type: WAV audi (audio/x-wav)

Size: 991,3 kB (991284 bytes)

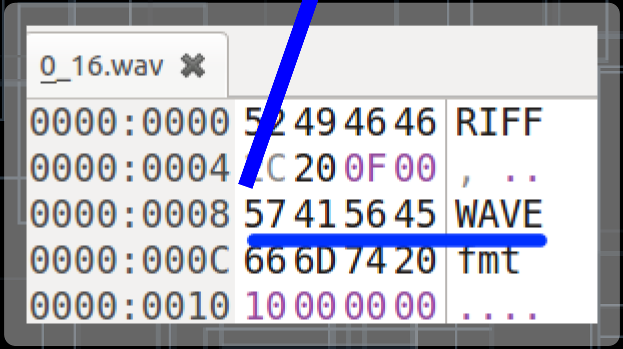


```
0000:0000 52 49 46 46 RIFF
0000:0004 2C 20 0F 00 , ..
0000:0008 57 41 56 45 WAVE
0000:000C 66 6D 74 20 fmt
0000:0010 10 00 00 00 ....
```

Master chunk: Format



Fieldname	Format
Description	Contains the letters "WAVE" in ASCII
Decimal	87 65 86 69
Hex Big Endian	57 41 56 45
Hex Little Endian	n. a.



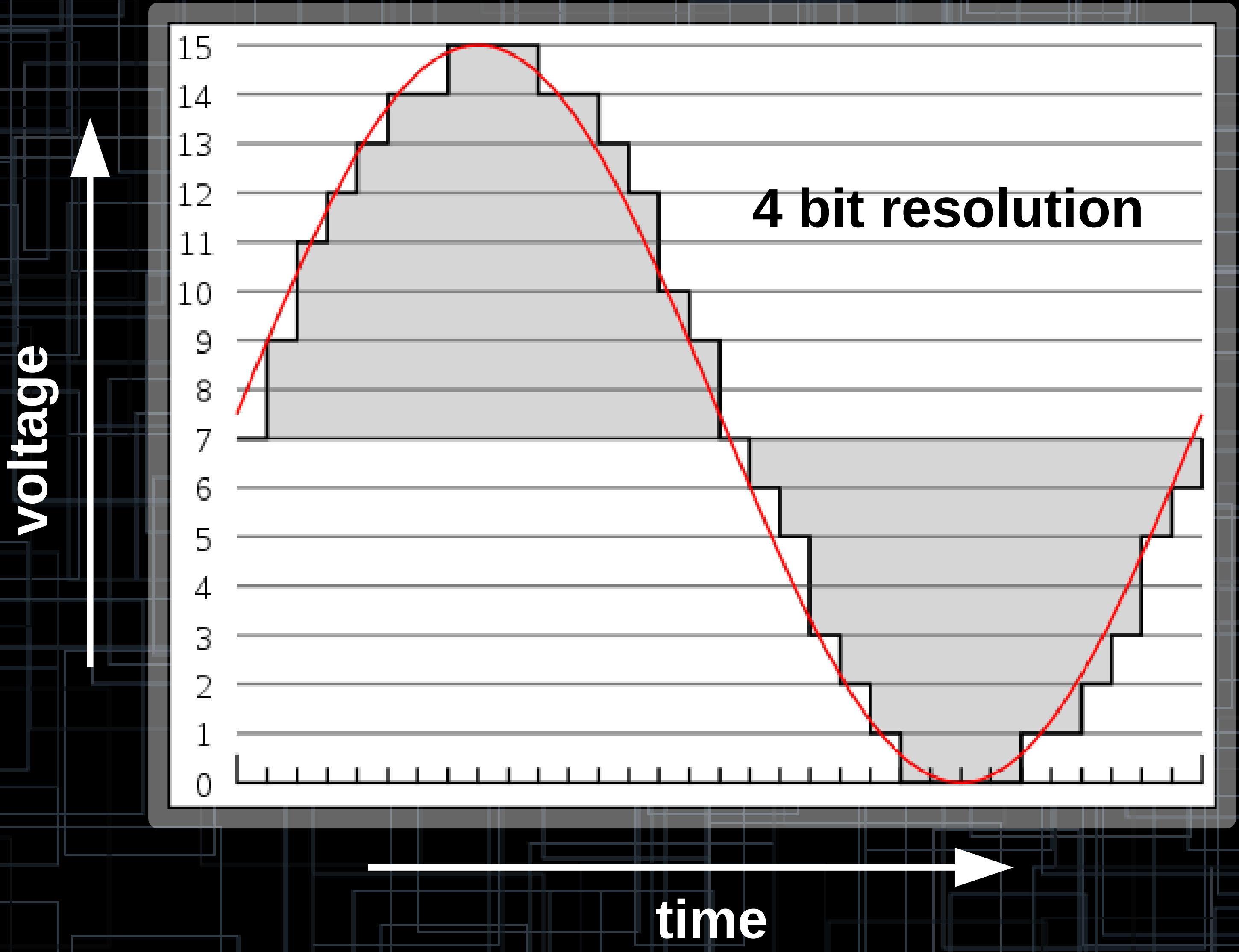
0_16.wav

```
0000:0000 52 49 46 46 RIFF
0000:0004 2C 20 0F 00 , ..
0000:0008 57 41 56 45 WAVE
0000:000C 66 6D 74 20 fmt
0000:0010 10 00 00 00 ....
```

RIFF chunk descriptor: done ✓

- ChunkID = 52 49 46 46 = RIFF ✓
- ChunkSize = 2C 20 0F 00 = 991276 ✓
- Format = 57 41 56 45 = WAVE ✓

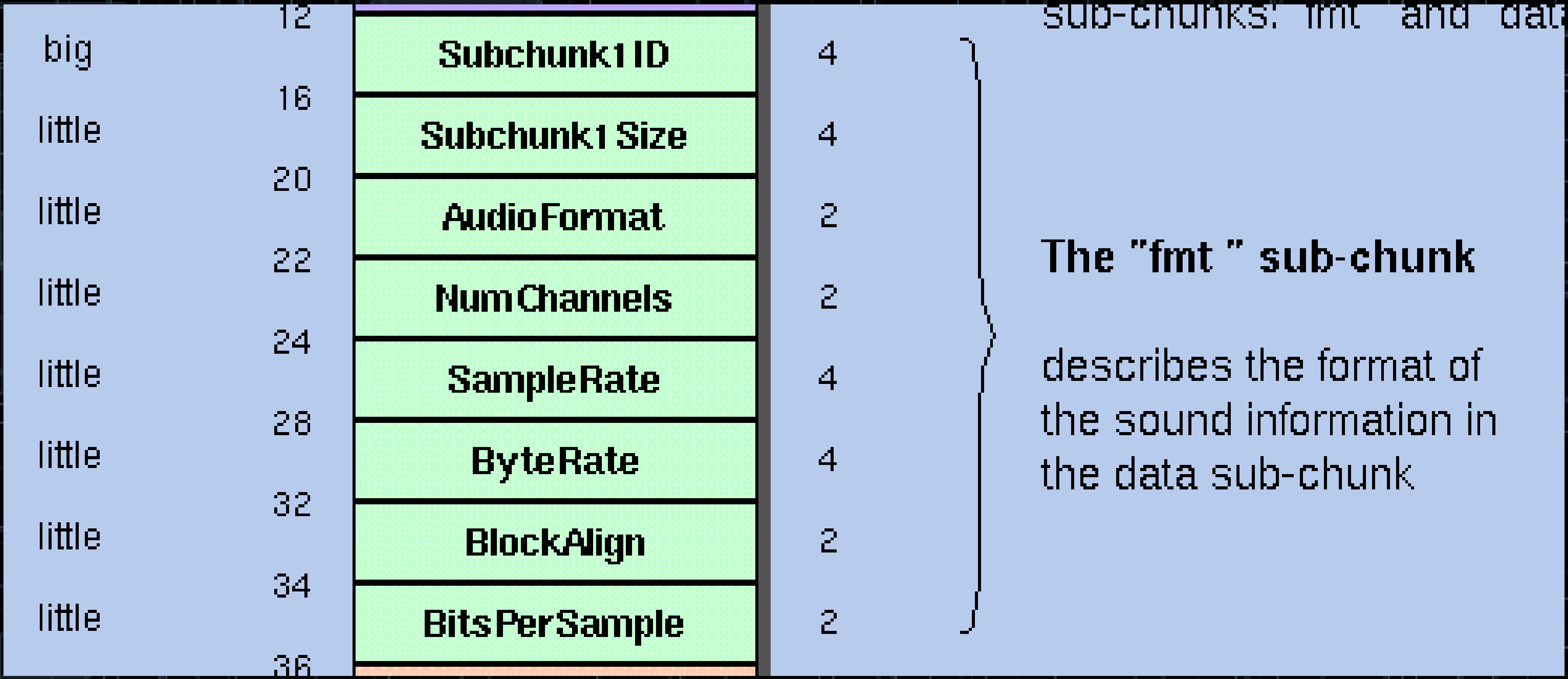
PCM: Pulse Code Modulation



Common PCM format

- CD (compact disc) format
- 2 Channels (stereo)
- 44'100 Hz sampling rate
- 16 bit resolution

The format, “fmt” sub-chunk

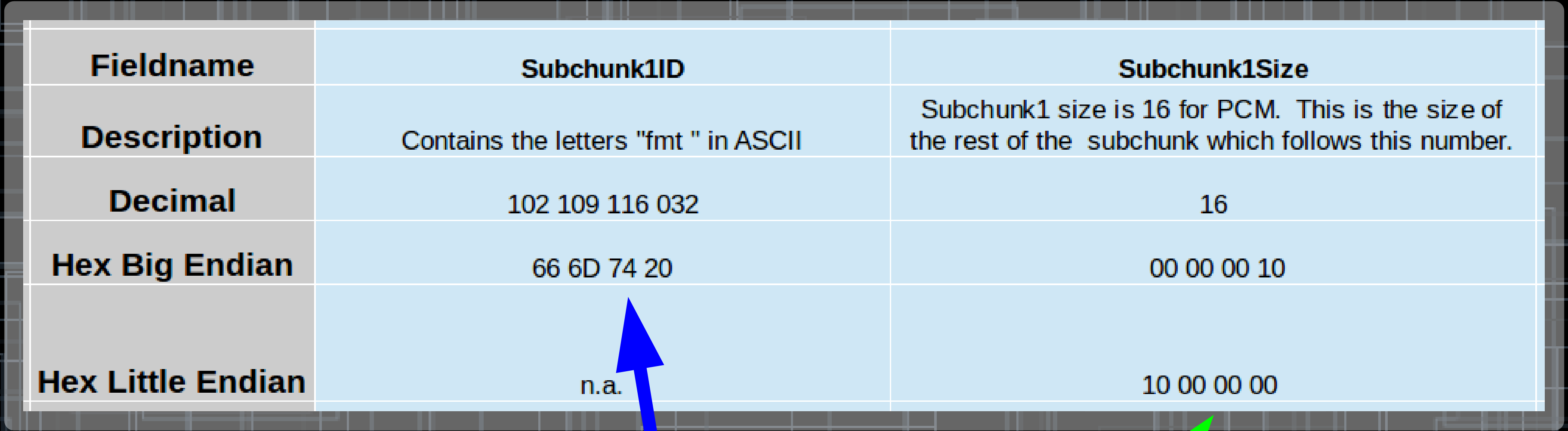


sub-chunks: fmt and dat

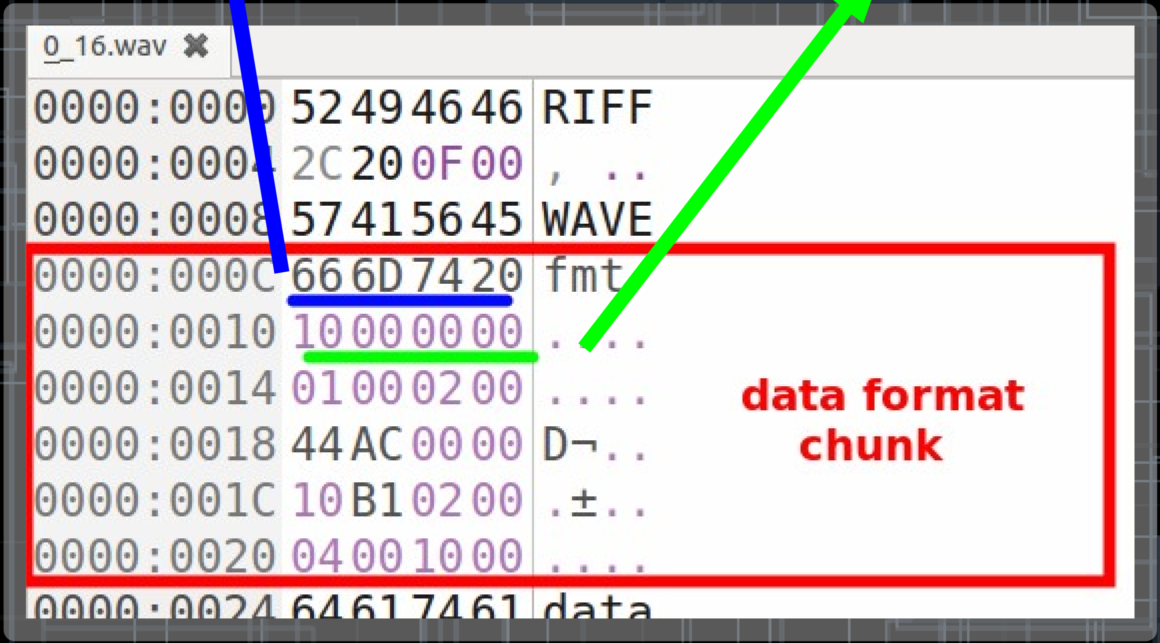
	12		
big		Subchunk1 ID	4
	16		
little		Subchunk1 Size	4
	20		
little		AudioFormat	2
	22		
little		NumChannels	2
	24		
little		SampleRate	4
	28		
little		ByteRate	4
	32		
little		BlockAlign	2
	34		
little		BitsPerSample	2
	36		

The "fmt " sub-chunk

describes the format of the sound information in the data sub-chunk



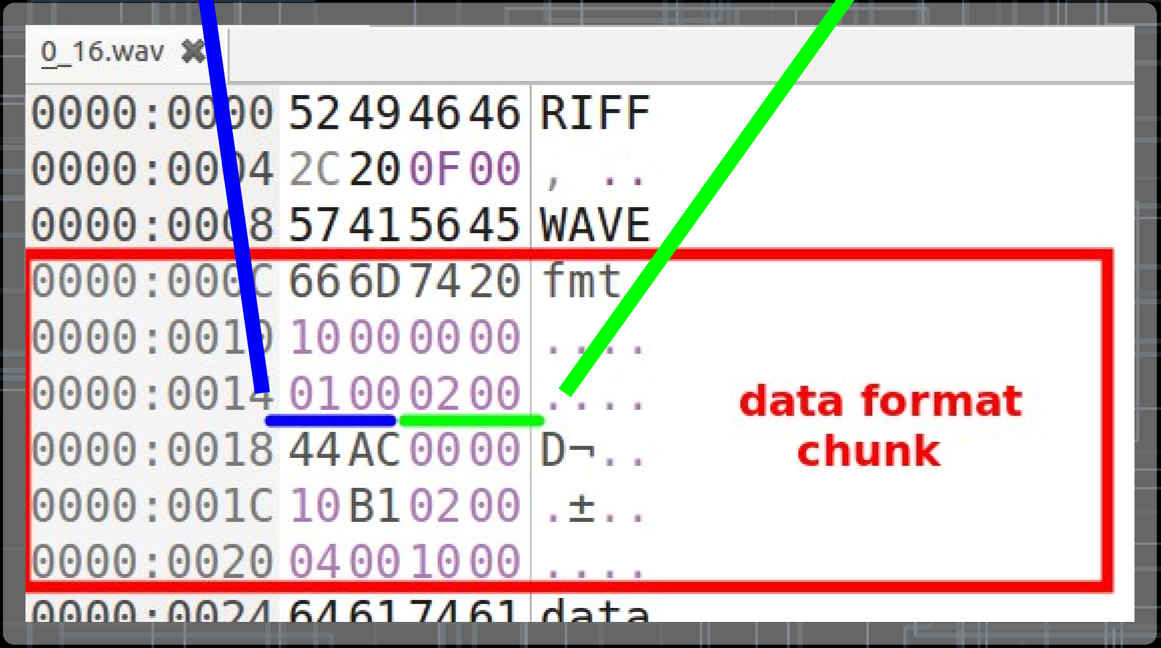
Fieldname	Subchunk1ID	Subchunk1Size
Description	Contains the letters "fmt " in ASCII	Subchunk1 size is 16 for PCM. This is the size of the rest of the subchunk which follows this number.
Decimal	102 109 116 032	16
Hex Big Endian	66 6D 74 20	00 00 00 10
Hex Little Endian	n.a.	10 00 00 00



0_16.wav

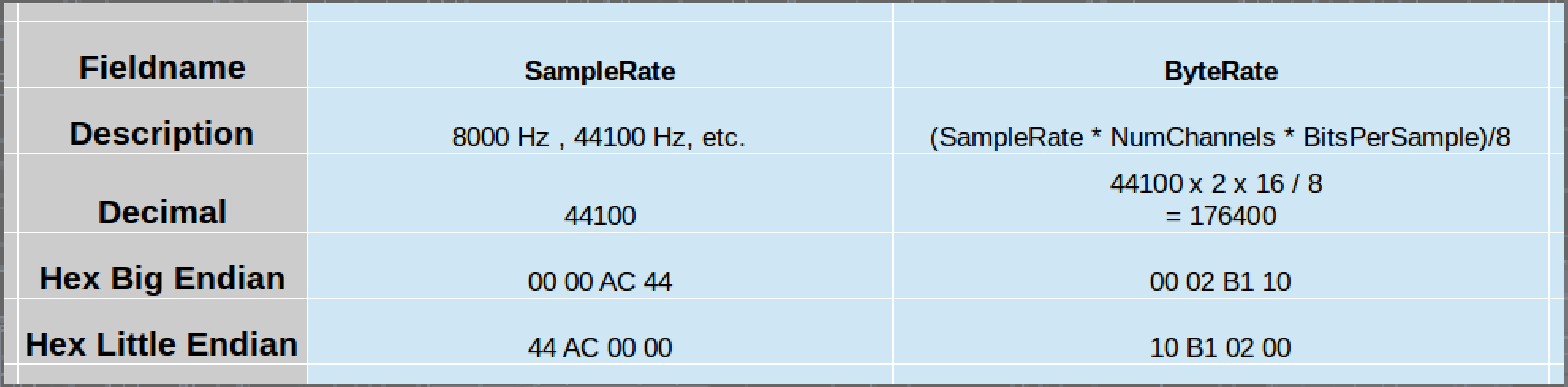
```
0000:0000 52494646 RIFF
0000:0004 2C200F00 , ..
0000:0008 57415645 WAVE
0000:000C 666D7420 fmt
0000:0010 10000000 ....
0000:0014 01000200 ....
0000:0018 44AC0000 D¬..
0000:001C 10B10200 .±..
0000:0020 04001000 ....
0000:0024 64617461 data
```

Fieldname	AudioFormat	NumChannels
Description	PCM = 1 (i.e. Linear quantization) Values other than 1 indicate some form of compression.	Mono = 1, Stereo = 2, etc.
Decimal	1	2
Hex Big Endian	00 01	00 02
Hex Little Endian	01 00	02 00

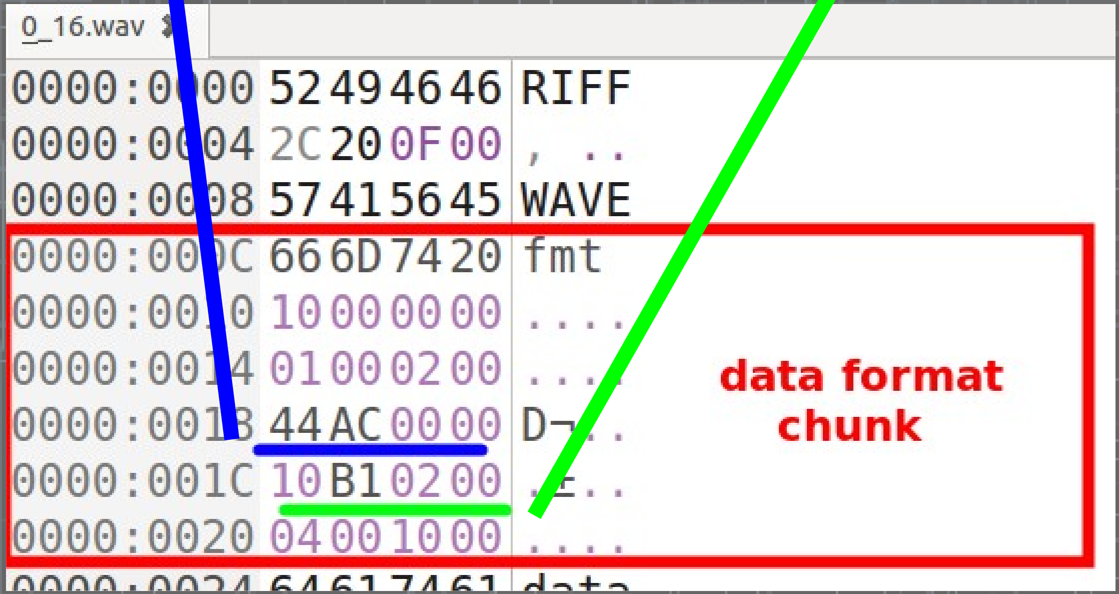


```
0000:0000 52 49 46 46 RIFF
0000:0004 2C 20 0F 00 , ..
0000:0008 57 41 56 45 WAVE
0000:000C 66 6D 74 20 fmt
0000:0010 10 00 00 00 ....
0000:0014 01 00 02 00 ....
0000:0018 44 AC 00 00 D¬..
0000:001C 10 B1 02 00 .±..
0000:0020 04 00 10 00 ....
0000:0024 64 61 74 61 data
```

data format chunk



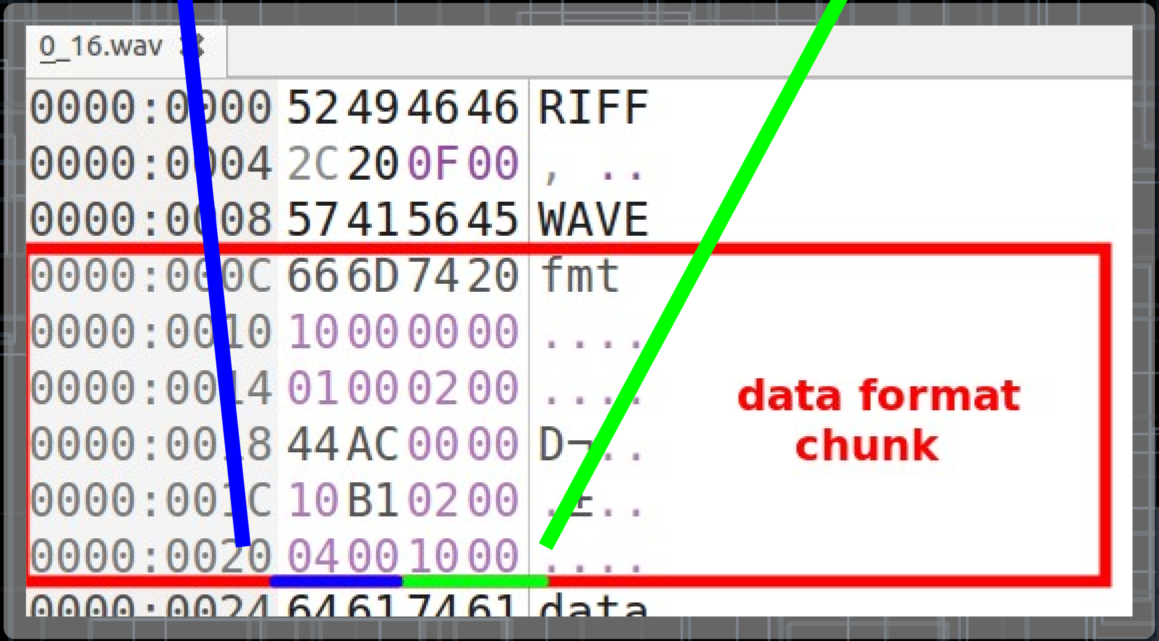
Fieldname	SampleRate	ByteRate
Description	8000 Hz , 44100 Hz, etc.	(SampleRate * NumChannels * BitsPerSample)/8
Decimal	44100	44100 x 2 x 16 / 8 = 176400
Hex Big Endian	00 00 AC 44	00 02 B1 10
Hex Little Endian	44 AC 00 00	10 B1 02 00



```
0000:0000 52494646 RIFF
0000:0004 2C200F00 , ..
0000:0008 57415645 WAVE
0000:000C 666D7420 fmt
0000:0010 10000000 ....
0000:0014 01000200 ....
0000:0018 44AC0000 D¬..
0000:001C 10B10200 .±..
0000:0020 04001000 ....
0000:0024 64617461 data
```

data format chunk

Fieldname	BlockAlign	BitsPerSample
Description	The number of bytes for one sample including all channels	8 bits = 8, 16 bits = 16, etc.
Decimal	2 x 16 / 8 = 4	16
Hex Big Endian	00 04	00 10
Hex Little Endian	04 00	10 00



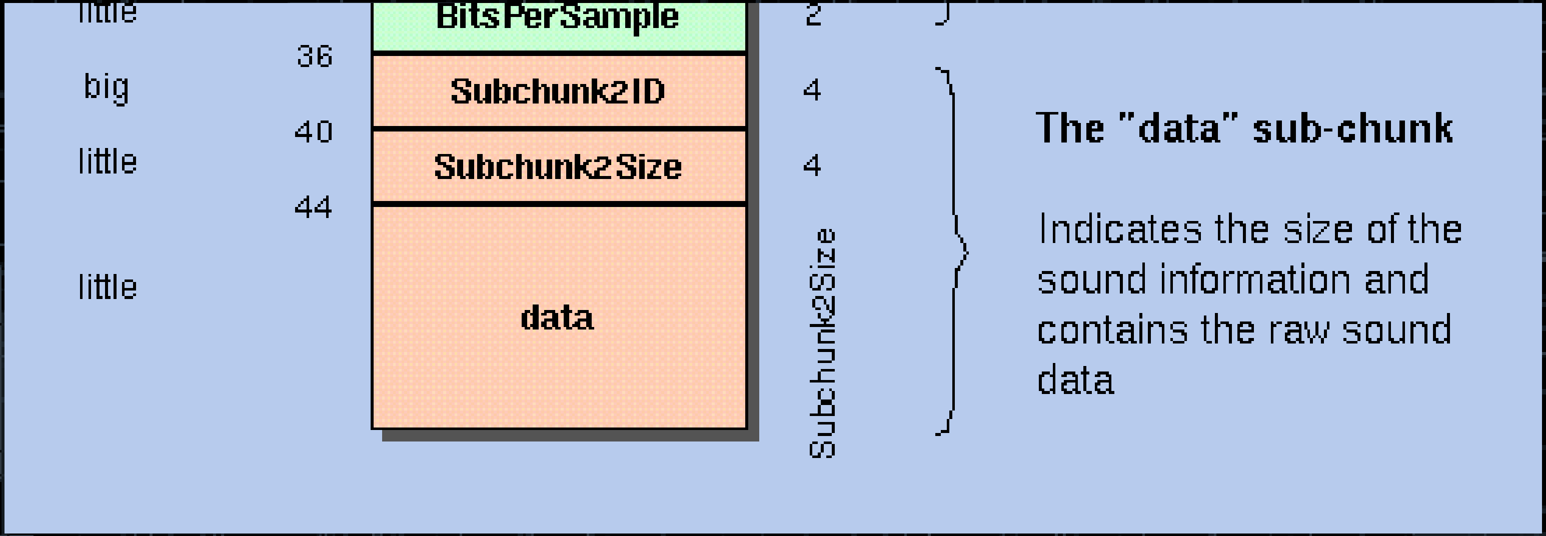
```
0000:0000 52 49 46 46 RIFF
0000:0004 2C 20 0F 00 , ..
0000:0008 57 41 56 45 WAVE
0000:000C 66 6D 74 20 fmt
0000:0010 10 00 00 00 ....
0000:0014 01 00 02 00 ....
0000:0018 44 AC 00 00 D¬..
0000:001C 10 B1 02 00 .±..
0000:0020 04 00 10 00 ....
0000:0024 64 61 74 61 data
```

data format chunk

“fmt” sub-chunk: done ✓

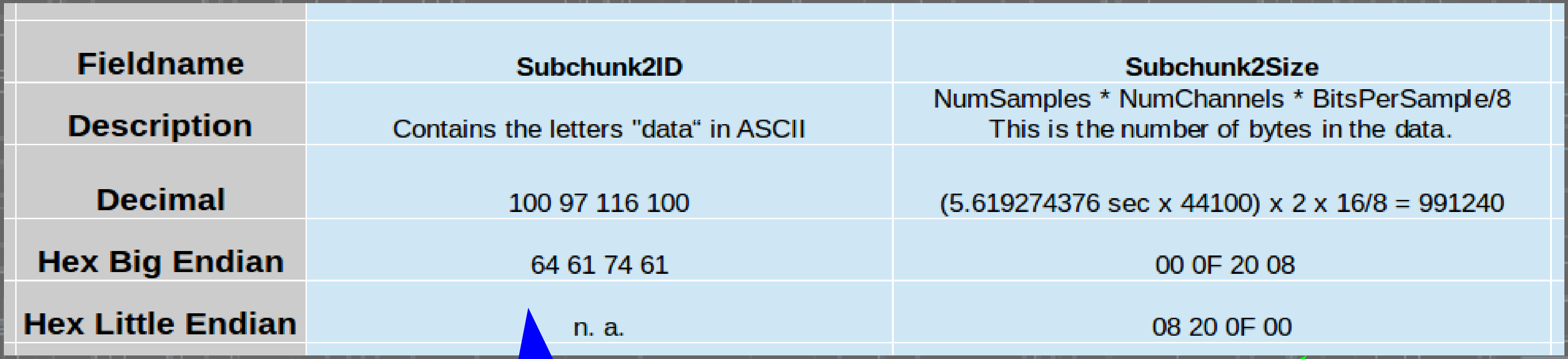
- Subchunk1ID = 66 6D 74 20 = fmt ✓
- Subchunk1Size = 10 00 00 00 = 16 ✓
- AudioFormat = 01 00 = 1 ✓
- NumChannels = 02 00 = 2 ✓
- SampleRate = 44 AC 00 00 = 44100 ✓
- ByteRate = 10 B1 02 00 = 176400 ✓
- BlockAlign = 04 00 = 4 ✓
- BitsPerSample = 10 00 = 16 ✓

The data sub-chunk

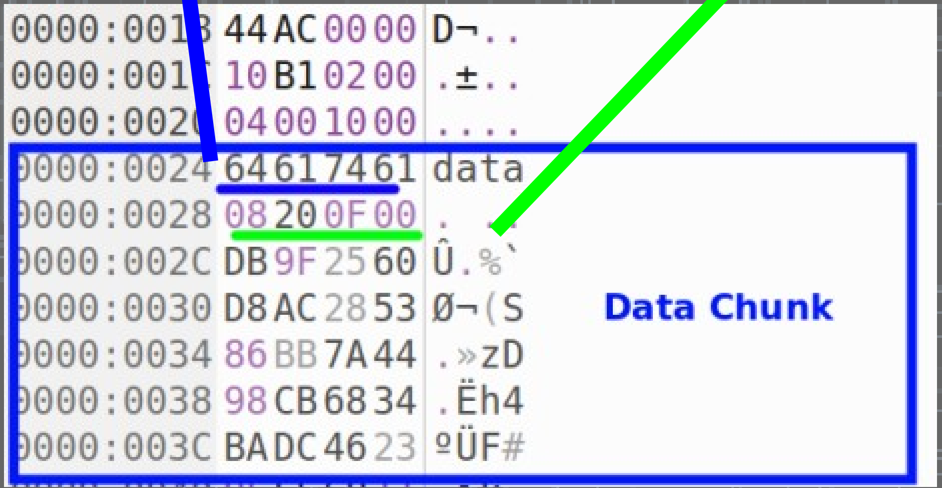


The "data" sub-chunk

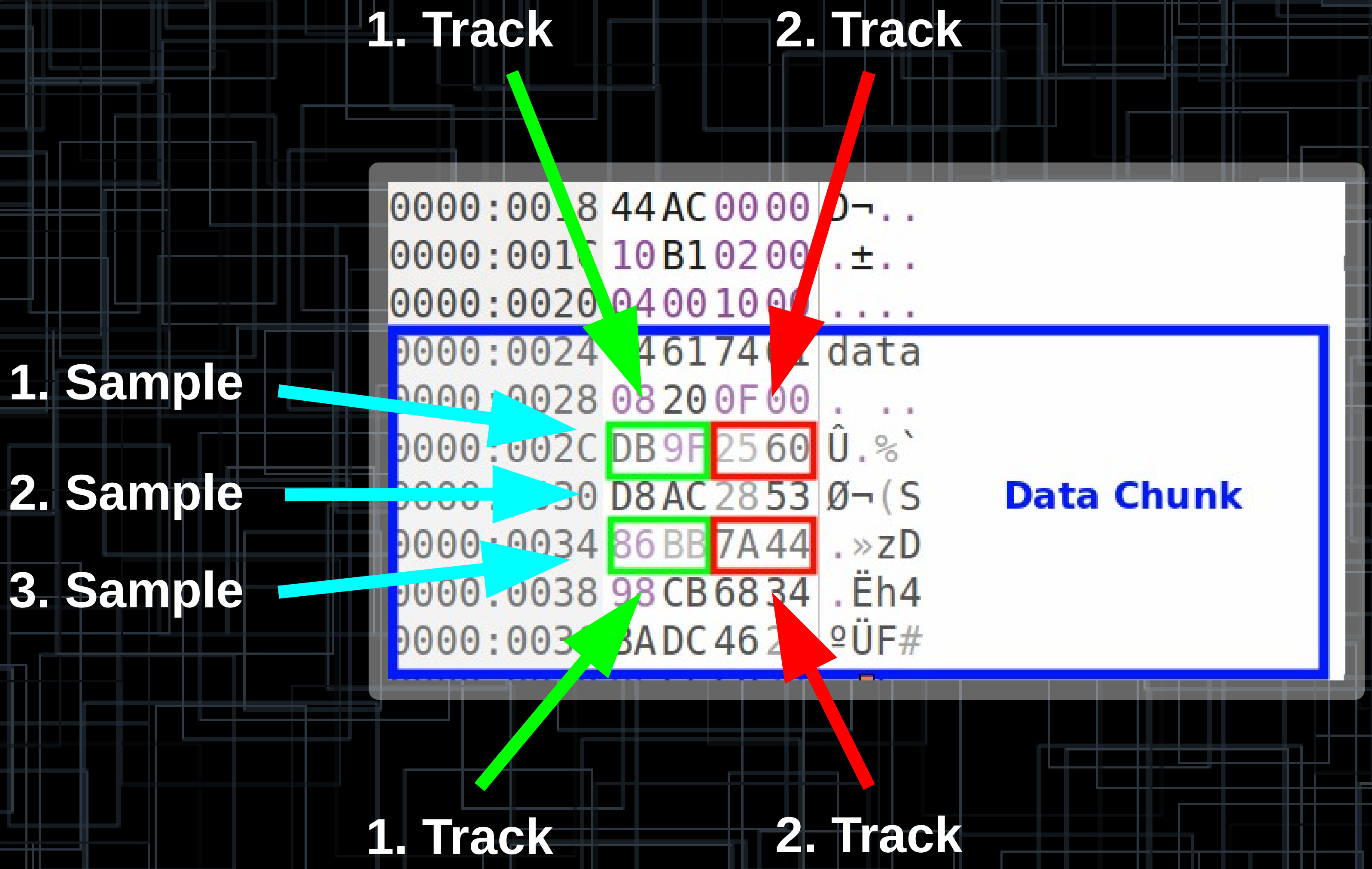
Indicates the size of the sound information and contains the raw sound data



Fieldname	Subchunk2ID	Subchunk2Size
Description	Contains the letters "data" in ASCII	NumSamples * NumChannels * BitsPerSample/8 This is the number of bytes in the data.
Decimal	100 97 116 100	(5.619274376 sec x 44100) x 2 x 16/8 = 991240
Hex Big Endian	64 61 74 61	00 0F 20 08
Hex Little Endian	n. a.	08 20 0F 00

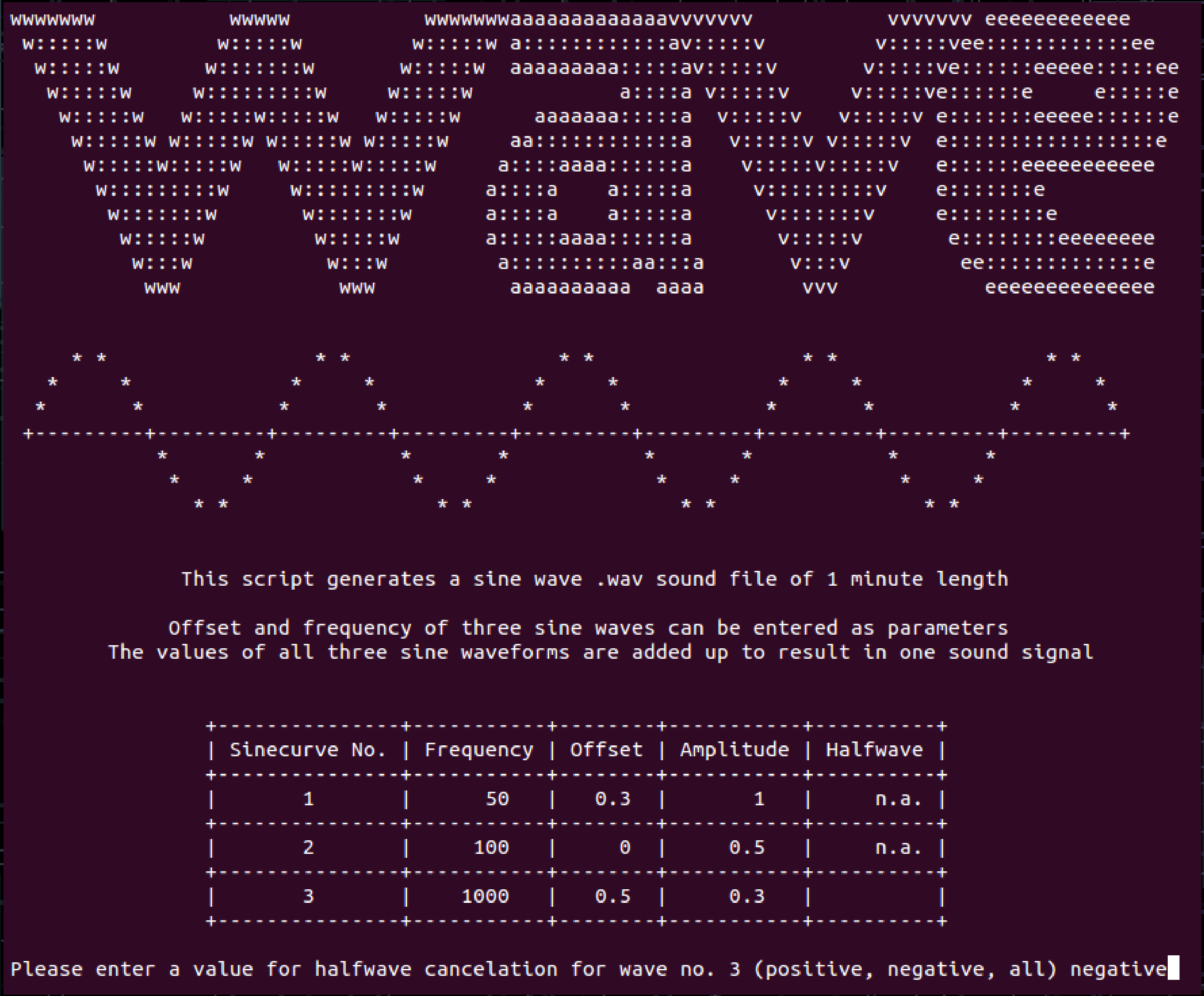


```
0000:0018 44AC0000 D¬..
0000:001C 10B10200 .±..
0000:0020 04001000 ....
0000:0024 64617461 data
0000:0028 08200F00 . ..
0000:002C DB9F2560 Û.%`
0000:0030 D8AC2853 Ø¬(S
0000:0034 86BB7A44 .»zD
0000:0038 98CB6834 .Ëh4
0000:003C BADC4623 ºÜF#
```



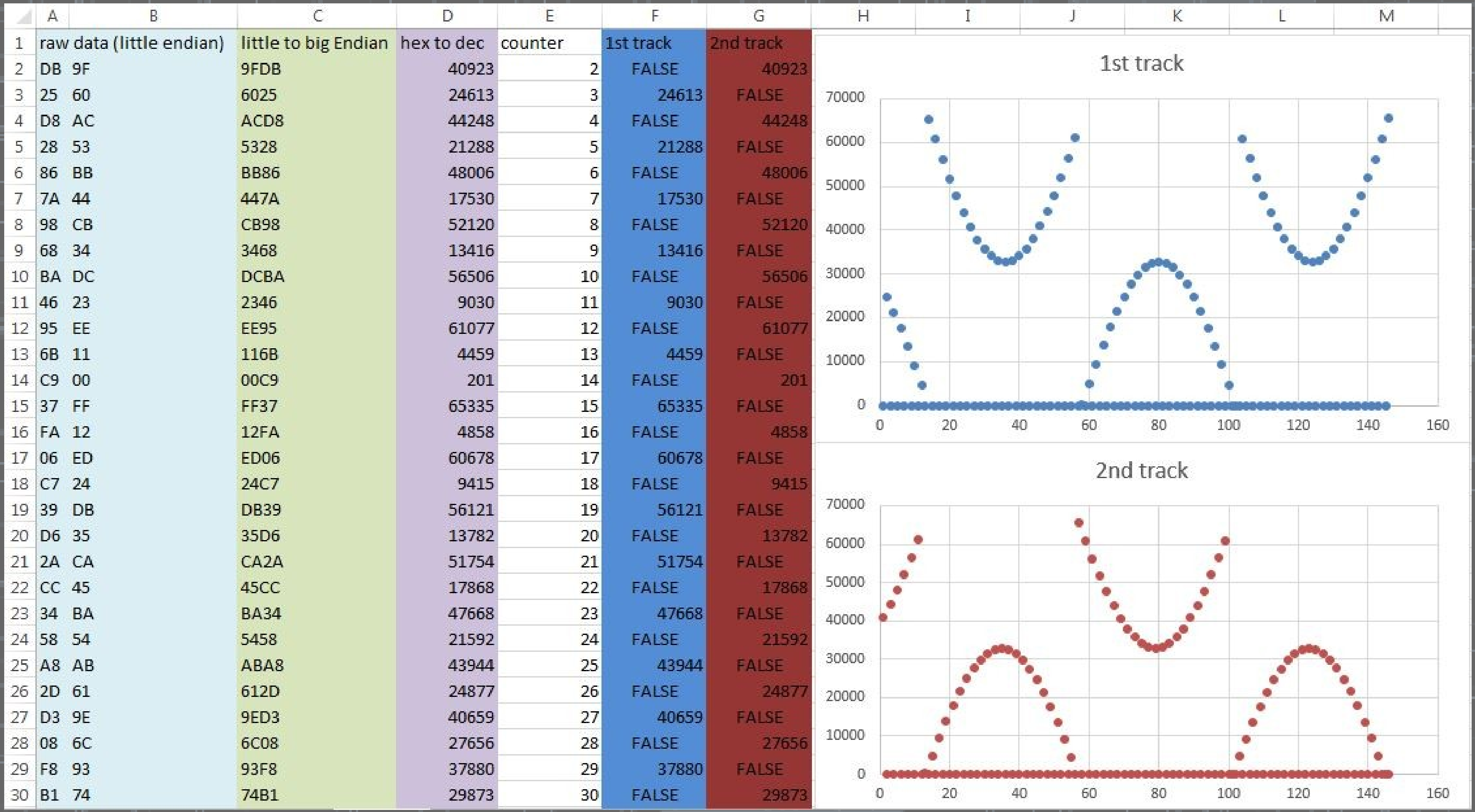
```
0000:0018 44 AC 00 00 D¬..
0000:001C 10 B1 02 00 .±..
0000:0020 04 00 10 00 ....
0000:0024 64 61 74 61 data
0000:0028 08 20 0F 00 . ..
0000:002C DB 9F 25 60 Û.%`
0000:0030 D8 AC 28 53 Ø¬(S
0000:0034 86 BB 7A 44 .»zD
0000:0038 98 CB 68 34 .Ëh4
0000:003C BA DC 46 23 ºÜF#
```

Korn shell script “wave.sh”



```
wwwwwww           wwwww           wwwwwwwaaaaaaaaaaaaavvvvvvv           vvvvvvv eeeeeeeeeeee
 w:::::w         w:::::w         w:::::w a::::::::::::av:::::v         v:::::vee::::::::::::ee
  w:::::w       w:::::::w       w:::::w  aaaaaaaaa:::::av:::::v       v:::::ve::::::eeeee:::::ee
   w:::::w     w:::::::::w     w:::::w            a::::a v:::::v     v:::::ve::::::e     e:::::e
    w:::::w   w:::::w:::::w   w:::::w      aaaaaaa:::::a  v:::::v   v:::::v e:::::::eeeee::::::e
     w:::::w w:::::w w:::::w w:::::w     aa::::::::::::a   v:::::v v:::::v  e:::::::::::::::::e
      w:::::w:::::w   w:::::w:::::w     a::::aaaa::::::a    v:::::v:::::v   e::::::eeeeeeeeeee
       w:::::::::w     w:::::::::w     a::::a    a:::::a     v:::::::::v    e:::::::e
        w:::::::w       w:::::::w      a::::a    a:::::a      v:::::::v     e::::::::e
         w:::::w         w:::::w       a:::::aaaa::::::a       v:::::v       e::::::::eeeeeeee
          w:::w           w:::w         a::::::::::aa:::a       v:::v         ee:::::::::::::e
           www             www           aaaaaaaaaa  aaaa        vvv            eeeeeeeeeeeeee


     * *                 * *                 * *                 * *                 * *
   *     *             *     *             *     *             *     *             *     *
  *       *           *       *           *       *           *       *           *       *
 +---------+---------+---------+---------+---------+---------+---------+---------+---------+
            *       *           *       *           *       *           *       *
             *     *             *     *             *     *             *     *
               * *                 * *                 * *                 * *


              This script generates a sine wave .wav sound file of 1 minute length

             Offset and frequency of three sine waves can be entered as parameters
        The values of all three sine waveforms are added up to result in one sound signal


                +---------------+-----------+--------+-----------+----------+
                | Sinecurve No. | Frequency | Offset | Amplitude | Halfwave |
                +---------------+-----------+--------+-----------+----------+
                |       1       |      50   |   0.3  |       1   |     n.a. |
                +---------------+-----------+--------+-----------+----------+
                |       2       |     100   |     0  |     0.5   |     n.a. |
                +---------------+-----------+--------+-----------+----------+
                |       3       |    1000   |   0.3  |     0.3   |          |
                +---------------+-----------+--------+-----------+----------+

Please enter a value for halfwave cancelation for wave no. 3 (positive, negative, all) negative
```

raw data “two's complement”



	A	B	C	D	E	F	G
1	raw data (little endian)		little to big Endian	hex to dec	counter	1st track	2nd track
2	DB	9F	9FDB	40923	2	FALSE	40923
3	25	60	6025	24613	3	24613	FALSE
4	D8	AC	ACD8	44248	4	FALSE	44248
5	28	53	5328	21288	5	21288	FALSE
6	86	BB	BB86	48006	6	FALSE	48006
7	7A	44	447A	17530	7	17530	FALSE
8	98	CB	CB98	52120	8	FALSE	52120
9	68	34	3468	13416	9	13416	FALSE
10	BA	DC	DCBA	56506	10	FALSE	56506
11	46	23	2346	9030	11	9030	FALSE
12	95	EE	EE95	61077	12	FALSE	61077
13	6B	11	116B	4459	13	4459	FALSE
14	C9	00	00C9	201	14	FALSE	201
15	37	FF	FF37	65335	15	65335	FALSE
16	FA	12	12FA	4858	16	FALSE	4858
17	06	ED	ED06	60678	17	60678	FALSE
18	C7	24	24C7	9415	18	FALSE	9415
19	39	DB	DB39	56121	19	56121	FALSE
20	D6	35	35D6	13782	20	FALSE	13782
21	2A	CA	CA2A	51754	21	51754	FALSE
22	CC	45	45CC	17868	22	FALSE	17868
23	34	BA	BA34	47668	23	47668	FALSE
24	58	54	5458	21592	24	FALSE	21592
25	A8	AB	ABA8	43944	25	43944	FALSE
26	2D	61	612D	24877	26	FALSE	24877
27	D3	9E	9ED3	40659	27	40659	FALSE
28	08	6C	6C08	27656	28	FALSE	27656
29	F8	93	93F8	37880	29	37880	FALSE
30	B1	74	74B1	29873	30	FALSE	29873

Resolution: 16 bit

2^{16} = 65'536

1, 2, 3 … 65535,65536

0, 1, 2 … 65534,65535

-32768,-32767,...,-1,0,1,...,32767

~~-32768~~ -32767,...,-1,0,1,...,32767

two's complement

0,1,...,32767,32768,...,65535

-32768,........,-1,0,1,...,32767

+ 65536

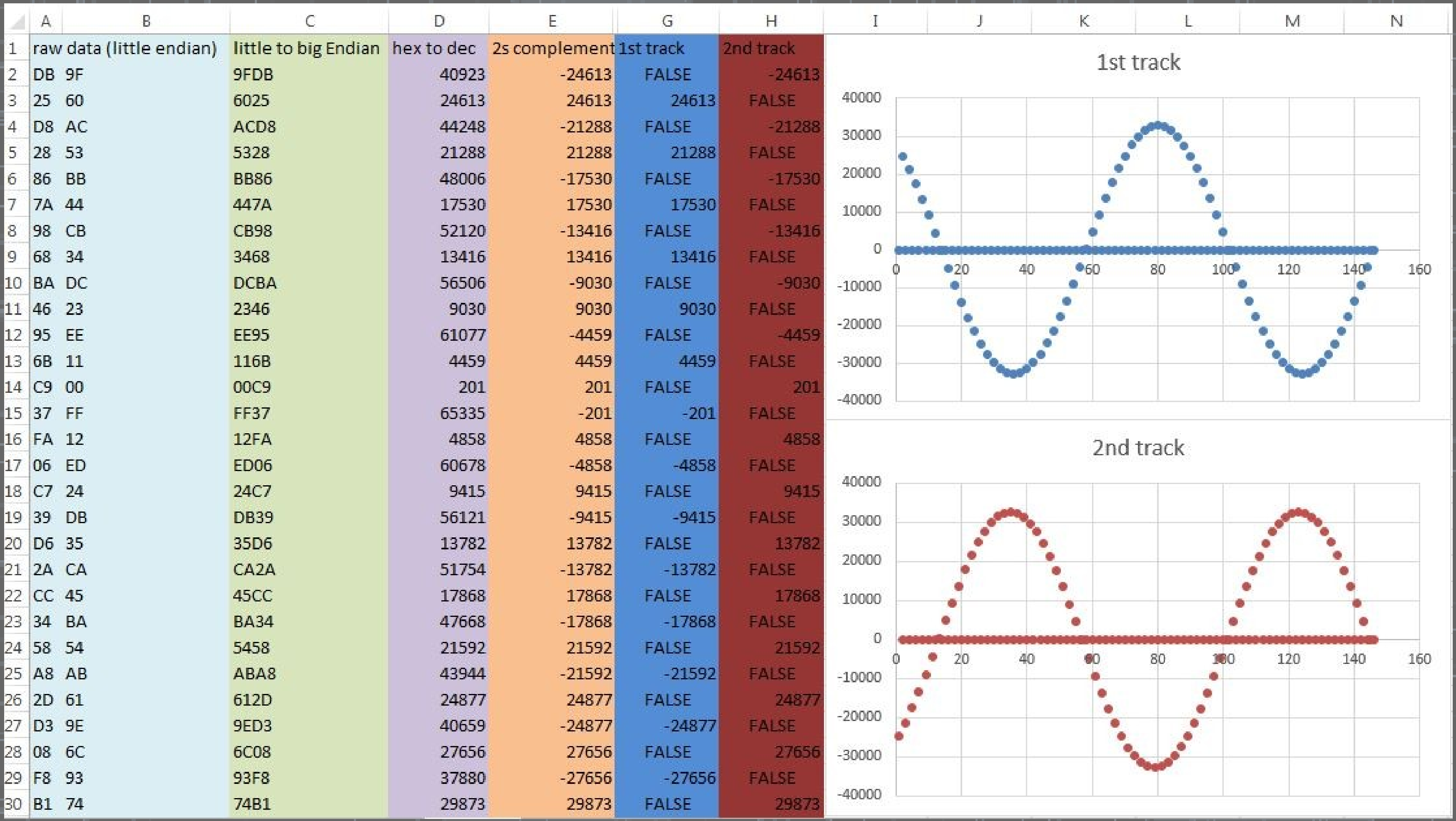
32768,...65535,0,1,...,32767

why two's complement ?

- Allows to do binary substraction by addition
- Allows to drop cary bit

➔ simplifies design of computing units

raw data transformed



	A	B	C	D	E	G	H
1	raw data (little endian)		little to big Endian	hex to dec	2s complement	1st track	2nd track
2	DB	9F	9FDB	40923	-24613	FALSE	-24613
3	25	60	6025	24613	24613	24613	FALSE
4	D8	AC	ACD8	44248	-21288	FALSE	-21288
5	28	53	5328	21288	21288	21288	FALSE
6	86	BB	BB86	48006	-17530	FALSE	-17530
7	7A	44	447A	17530	17530	17530	FALSE
8	98	CB	CB98	52120	-13416	FALSE	-13416
9	68	34	3468	13416	13416	13416	FALSE
10	BA	DC	DCBA	56506	-9030	FALSE	-9030
11	46	23	2346	9030	9030	9030	FALSE
12	95	EE	EE95	61077	-4459	FALSE	-4459
13	6B	11	116B	4459	4459	4459	FALSE
14	C9	00	00C9	201	201	FALSE	201
15	37	FF	FF37	65335	-201	-201	FALSE
16	FA	12	12FA	4858	4858	FALSE	4858
17	06	ED	ED06	60678	-4858	-4858	FALSE
18	C7	24	24C7	9415	9415	FALSE	9415
19	39	DB	DB39	56121	-9415	-9415	FALSE
20	D6	35	35D6	13782	13782	FALSE	13782
21	2A	CA	CA2A	51754	-13782	-13782	FALSE
22	CC	45	45CC	17868	17868	FALSE	17868
23	34	BA	BA34	47668	-17868	-17868	FALSE
24	58	54	5458	21592	21592	FALSE	21592
25	A8	AB	ABA8	43944	-21592	-21592	FALSE
26	2D	61	612D	24877	24877	FALSE	24877
27	D3	9E	9ED3	40659	-24877	-24877	FALSE
28	08	6C	6C08	27656	27656	FALSE	27656
29	F8	93	93F8	37880	-27656	-27656	FALSE
30	B1	74	74B1	29873	29873	FALSE	29873

data sub-chunk: done ✓

- ChunkID = 64 61 74 61 = data ✓
- Subchunk2Size = 08 20 0F 00 = 991240 ✓
- data = two's complement ✓

program

- 1. part : master chunk, subchunk1, subchunk2 (headers)

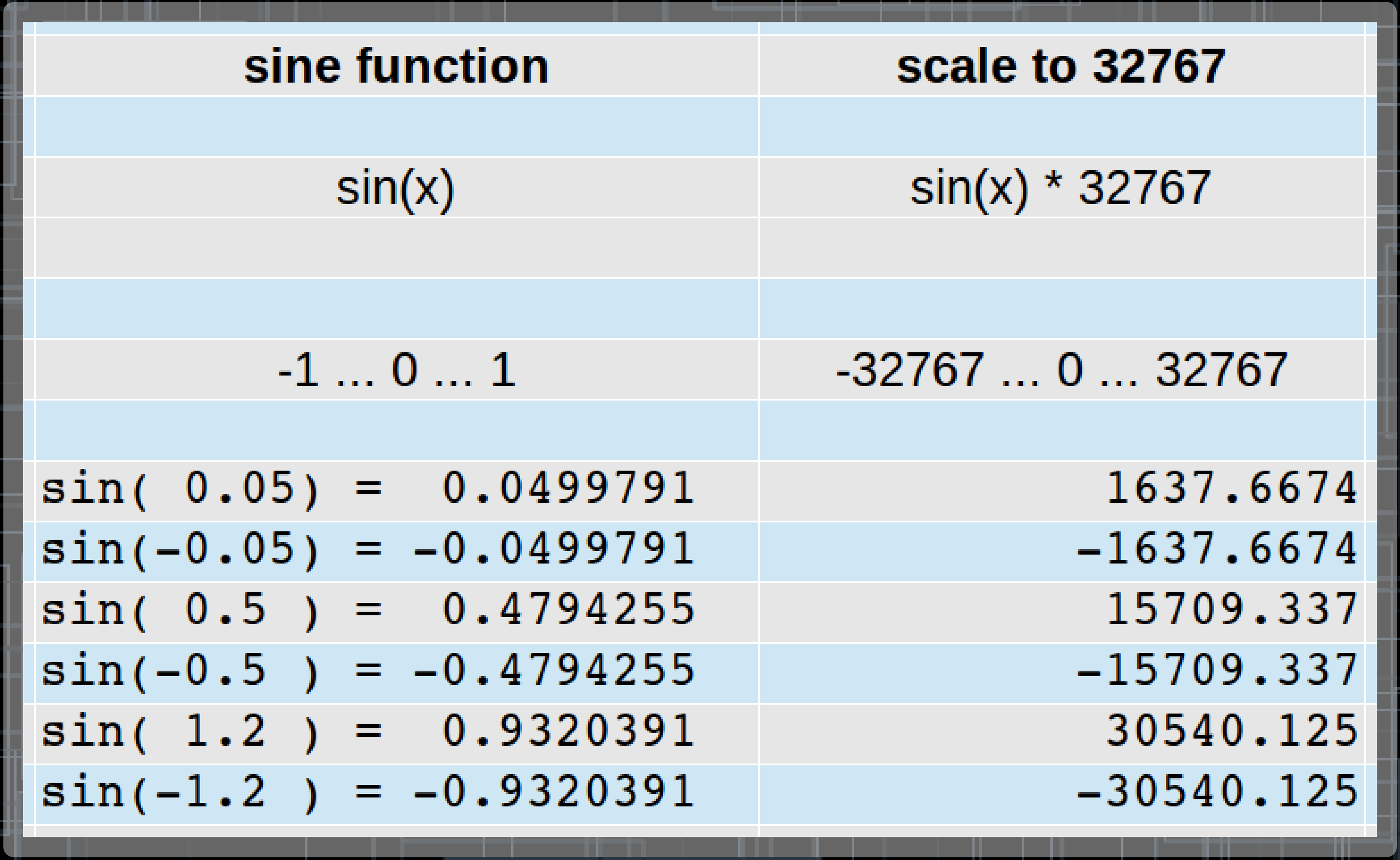
➔ Easy: constant for identical size (1 minute)

- 2. part : payload

➔ For 1 minute : 44100 x 60 iterations of sin(x)

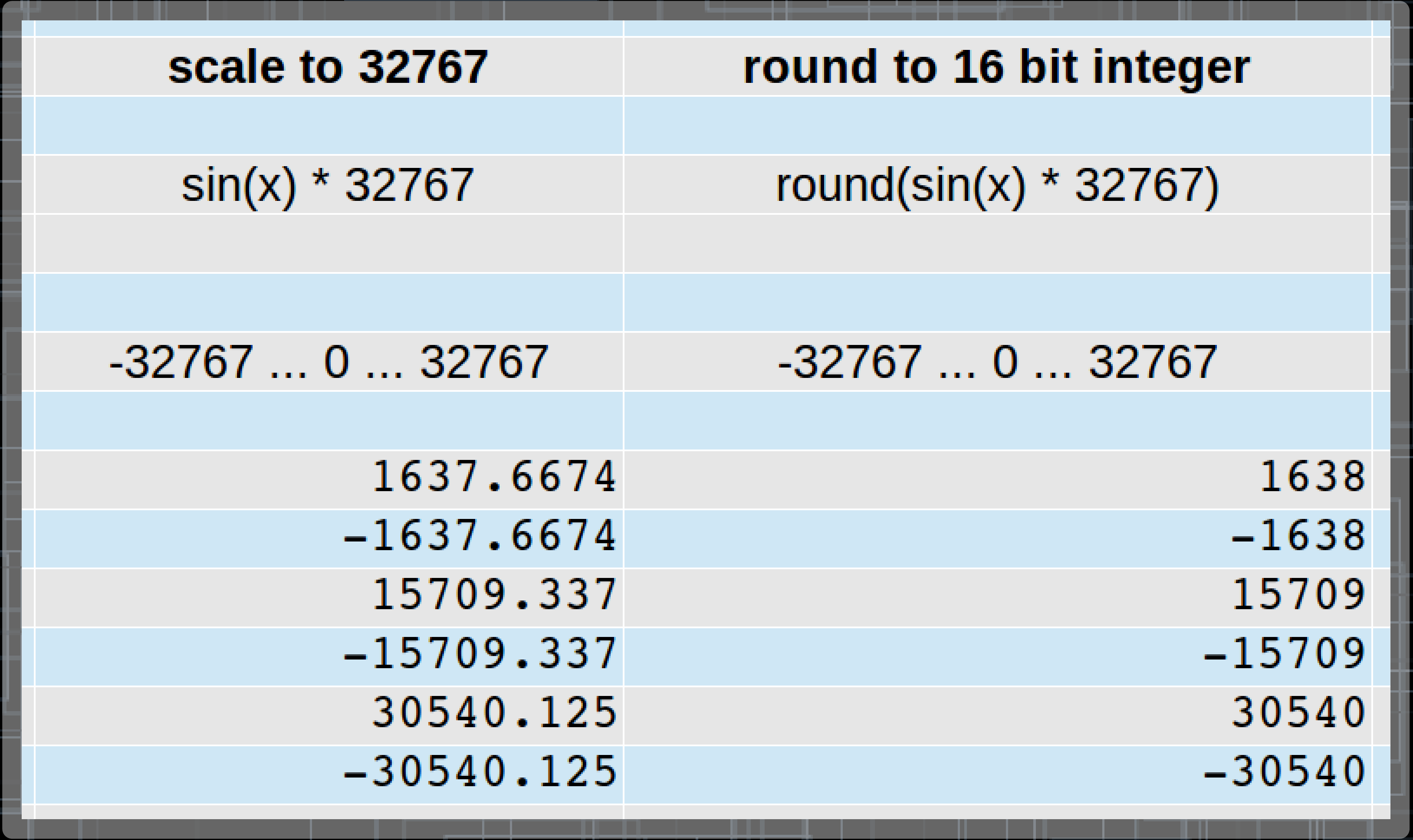
and its transformation to little endian 2's complement

from sin(x) to raw data



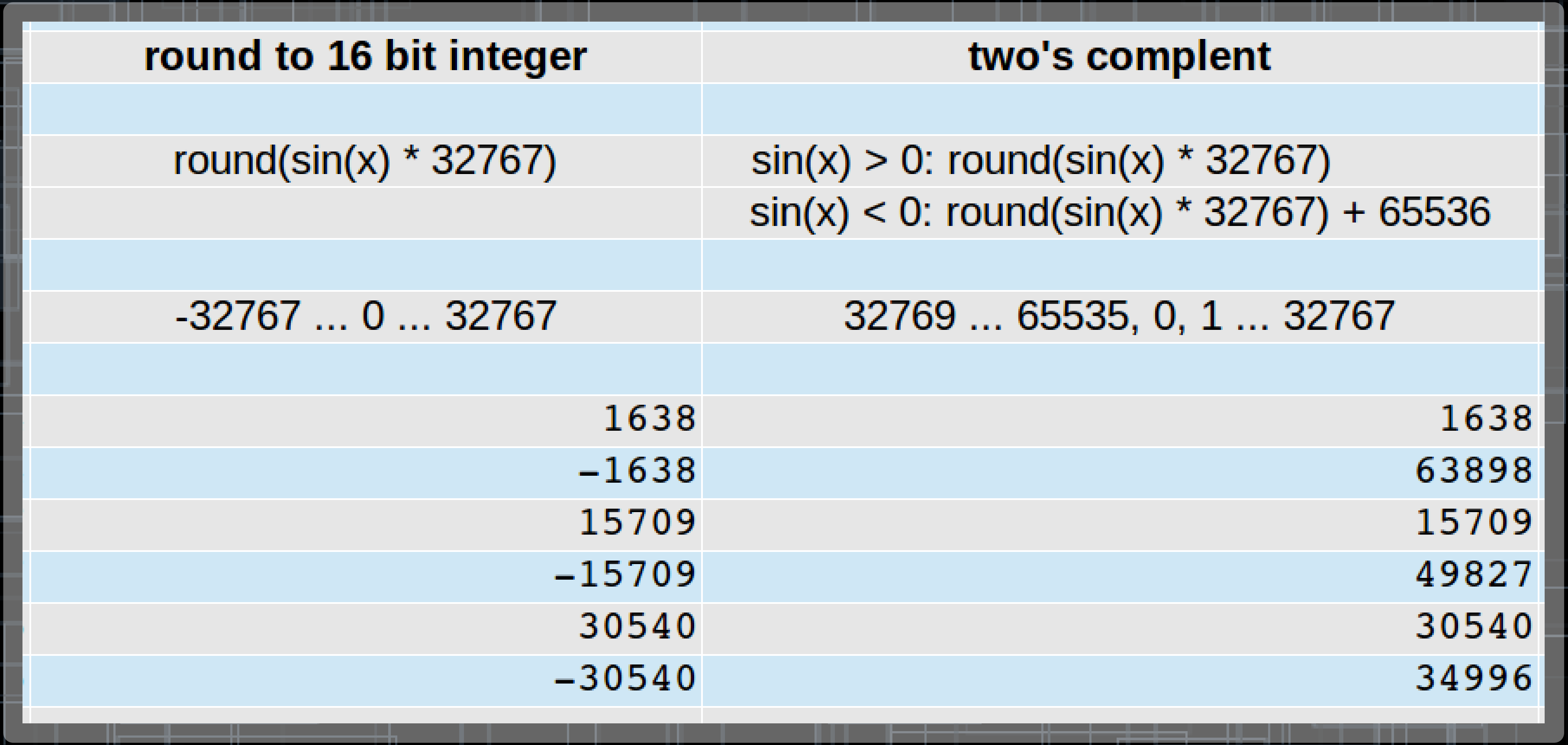
sine function	scale to 32767
sin(x)	sin(x) * 32767
-1 ... 0 ... 1	-32767 ... 0 ... 32767
sin(0.05) = 0.0499791	1637.6674
sin(-0.05) = -0.0499791	-1637.6674
sin(0.5) = 0.4794255	15709.337
sin(-0.5) = -0.4794255	-15709.337
sin(1.2) = 0.9320391	30540.125
sin(-1.2) = -0.9320391	-30540.125

from sin(x) to raw data



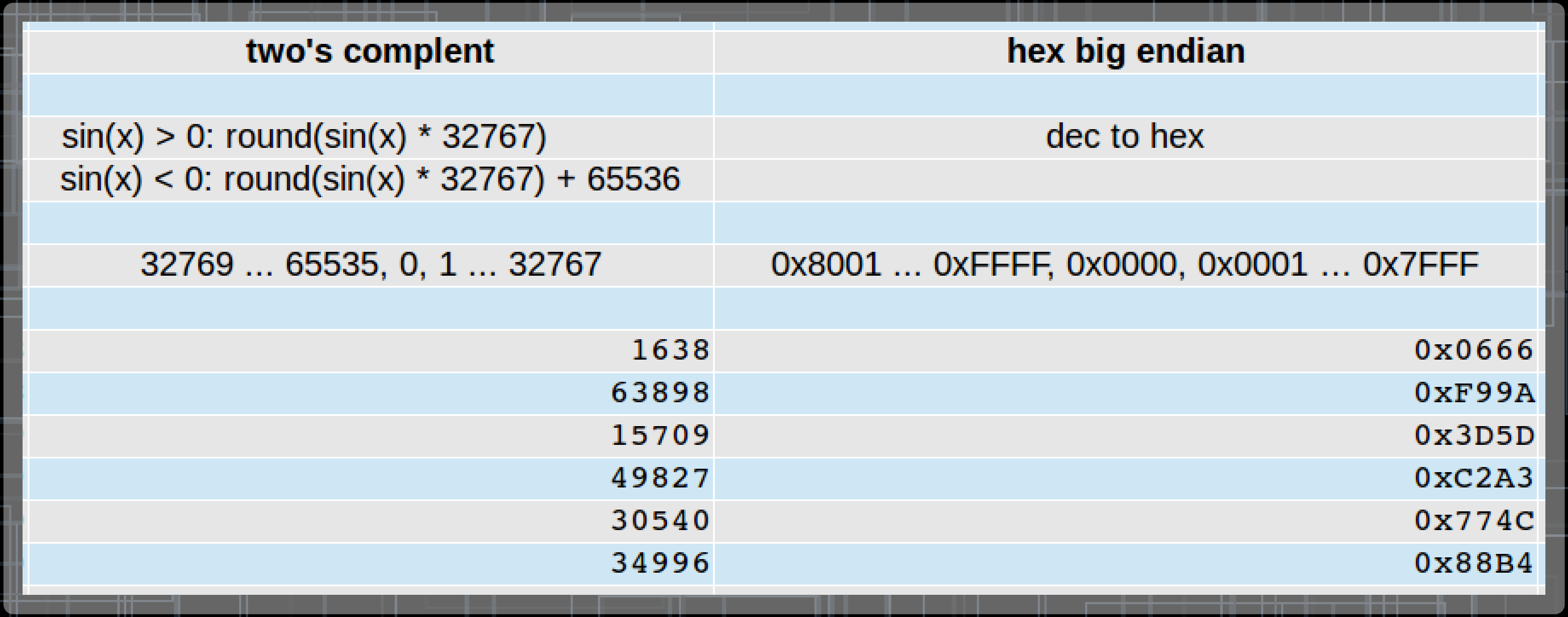
scale to 32767	round to 16 bit integer
sin(x) * 32767	round(sin(x) * 32767)
-32767 ... 0 ... 32767	-32767 ... 0 ... 32767
1637.6674	1638
-1637.6674	-1638
15709.337	15709
-15709.337	-15709
30540.125	30540
-30540.125	-30540

from sin(x) to raw data



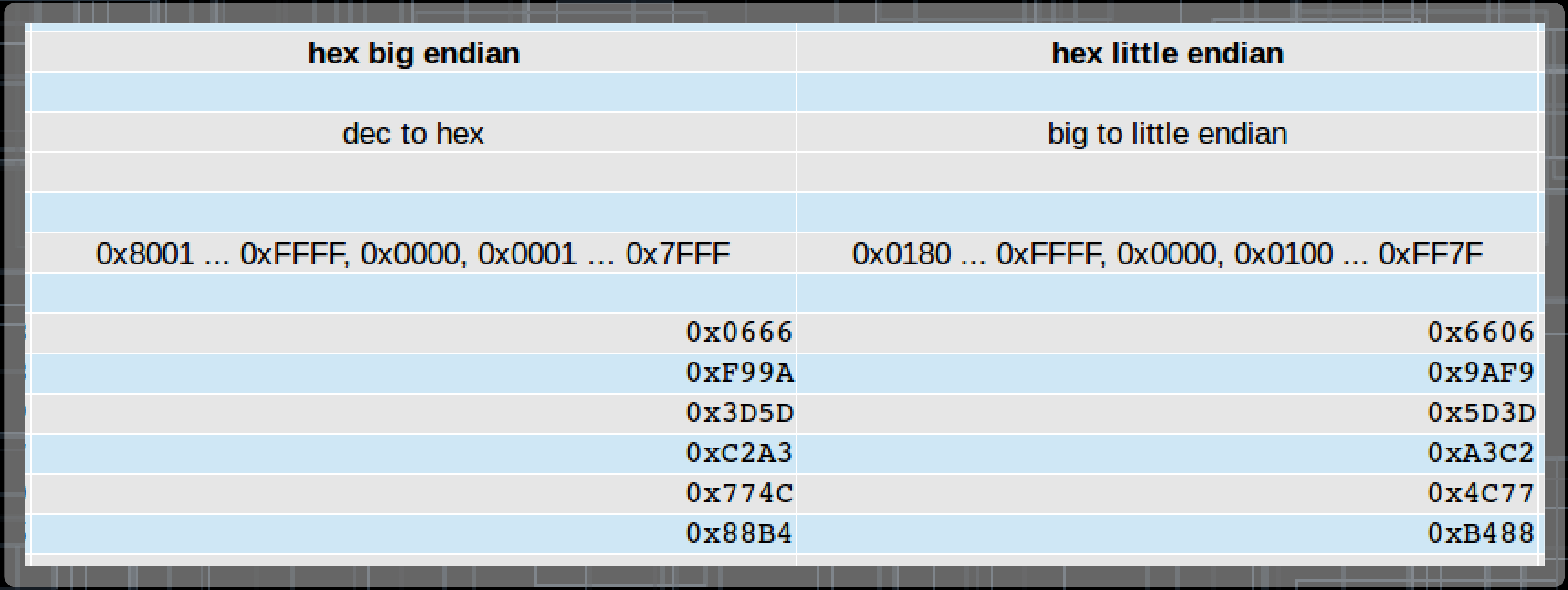
round to 16 bit integer	two's complent
round(sin(x) * 32767)	sin(x) > 0: round(sin(x) * 32767)
	sin(x) < 0: round(sin(x) * 32767) + 65536
-32767 ... 0 ... 32767	32769 ... 65535, 0, 1 ... 32767
1638	1638
-1638	63898
15709	15709
-15709	49827
30540	30540
-30540	34996

from sin(x) to raw data



two's complent	hex big endian
sin(x) > 0: round(sin(x) * 32767)	dec to hex
sin(x) < 0: round(sin(x) * 32767) + 65536	
32769 ... 65535, 0, 1 ... 32767	0x8001 ... 0xFFFF, 0x0000, 0x0001 ... 0x7FFF
1638	0x0666
63898	0xF99A
15709	0x3D5D
49827	0xC2A3
30540	0x774C
34996	0x88B4

from sin(x) to raw data

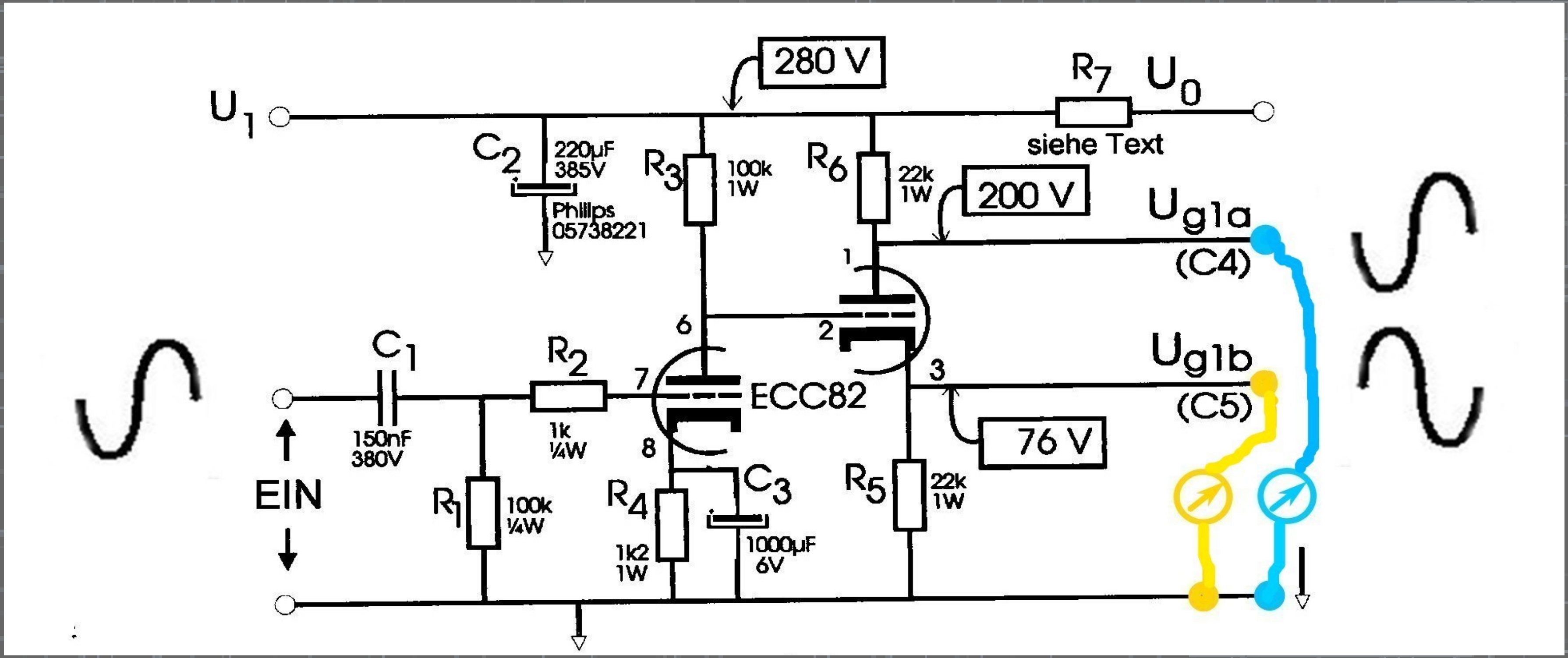


hex big endian	hex little endian
dec to hex	big to little endian
0x8001 ... 0xFFFF, 0x0000, 0x0001 … 0x7FFF	0x0180 ... 0xFFFF, 0x0000, 0x0100 ... 0xFF7F
0x0666	0x6606
0xF99A	0x9AF9
0x3D5D	0x5D3D
0xC2A3	0xA3C2
0x774C	0x4C77
0x88B4	0xB488

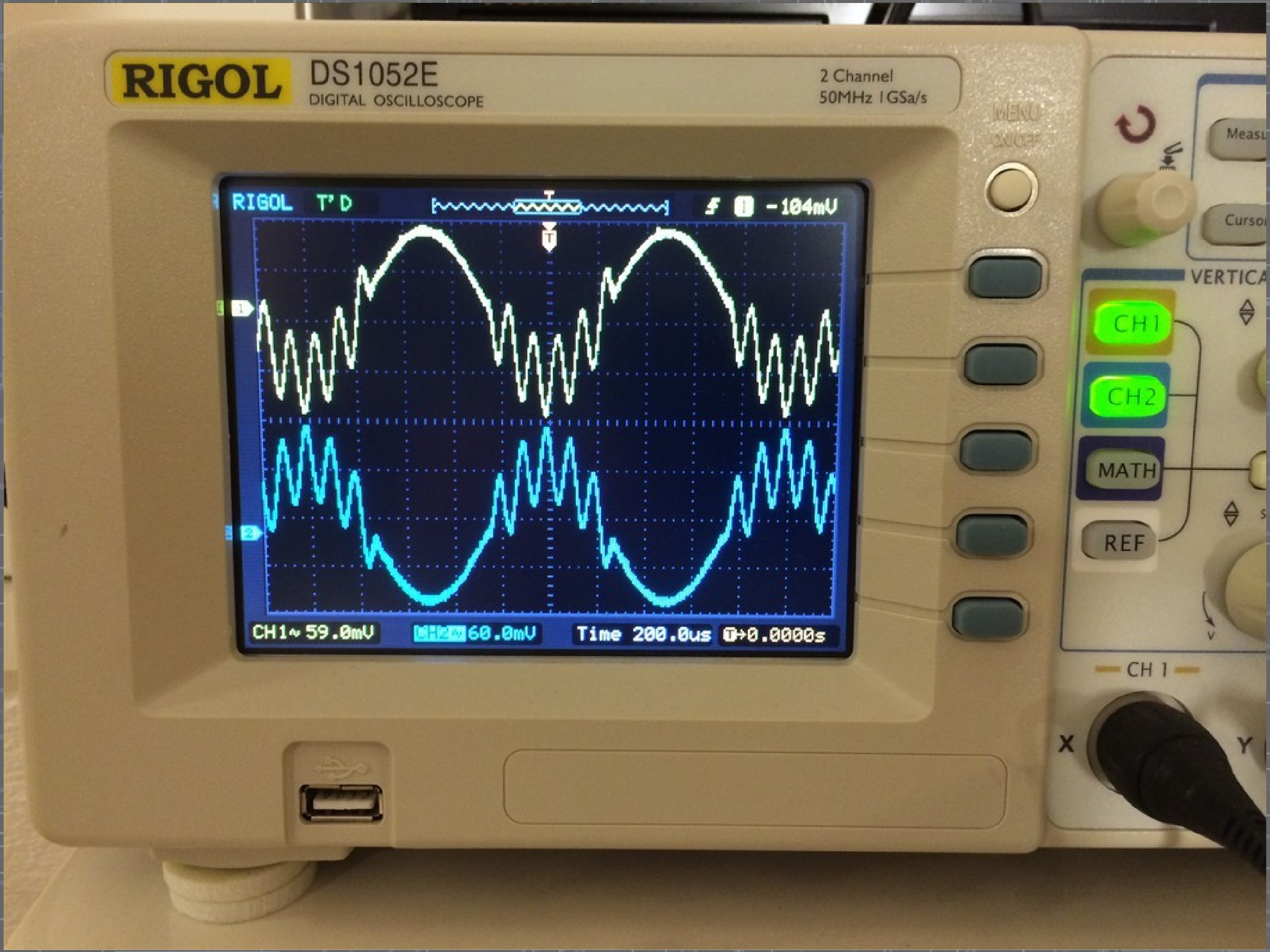
Korn shell script “wave.sh”

- Why shell script ?
- Training for LPIC
- Train vi editor
- Getting familiar with shell scripting
- Why Korn shell ?
- Supports sine function
- Can deal with floating point numbers

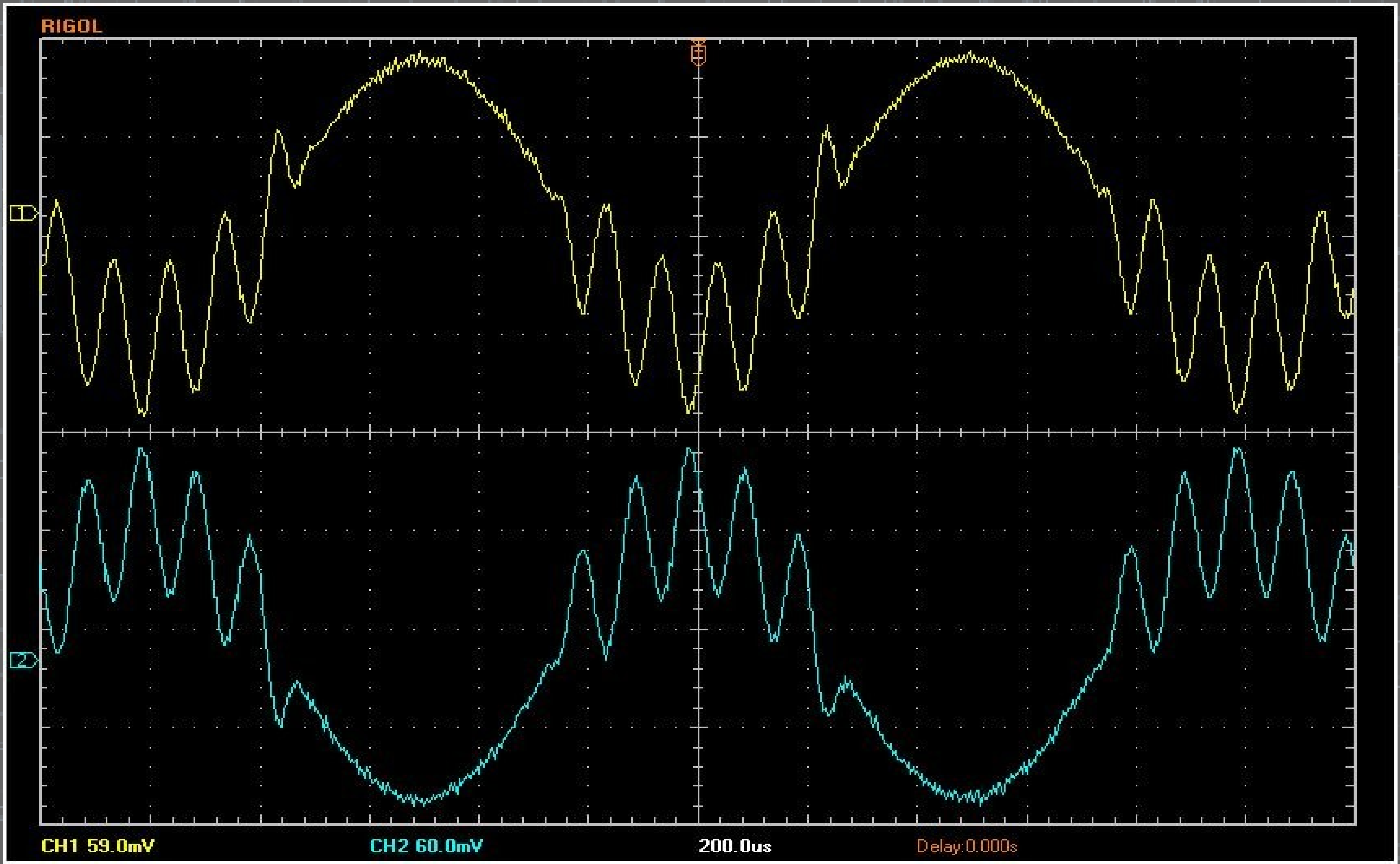
Action of phase splitter circuit



Action of phase splitter circuit



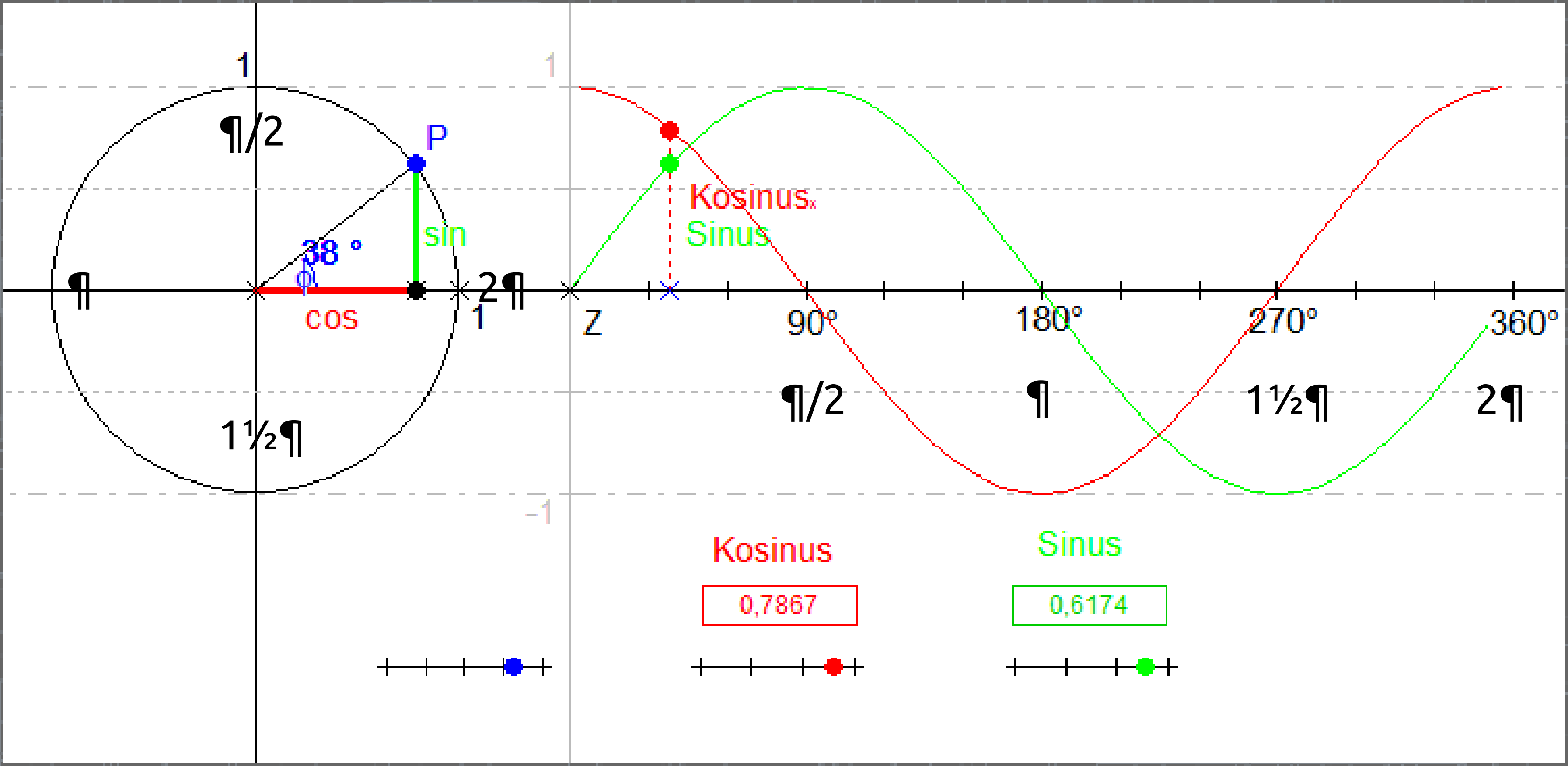
Action of phase splitter circuit



Outlook: further steps

- Measure rest of tube amp circuit
- Linux Alsa Sound System
- Porting Korn shell script do a different programming language
- Other PCM formats: CD, .au, .aiff
- Compressed audio formats: mp3
- Other RIFF file containers: .avi
- Torture sound card: throw nasty signals at it

Geometrical definition of the sine function



References

- RIFF: http://de.wikipedia.org/wiki/Resource_Interchange_File_Format
- WAVE: http://de.wikipedia.org/wiki/RIFF_WAVE
- WAVE: https://ccrma.stanford.edu/courses/422/projects/WaveFormat/

References

- Endianess: http://de.wikipedia.org/wiki/Byte-Reihenfolge
- Endianess: http://www.ietf.org/rfc/ien/ien137.txt
- PCM: http://en.wikipedia.org/wiki/Pulse-code_modulation