Steganography & Steganalysis

INSTRUCTOR: JOHN ORTIZ
SENIOR COMPUTER ENGINEER
UTSA
STEGO@SATX.RR.COM

GRAPHICS & AUDIO

Graphics Basics

- The monitor is nothing but a window into video memory
- Today, video memory is often on a separate graphics card, but some architectures allow main memory to serve as video memory
  - When you take a screenshot, all that happens is the contents of video memory are copied and put in an image format
    - Bmp, png, jpg, etc.
- Video memory can be interpreted as text or graphics
  - The DOS console window is a text example
  - Since ultimately even characters are a combination of pixels, in this case memory value indexes a table with the pixel pattern of the character

Graphics

- Red, Green, and Blue can be combined to produce any color
  - often abbreviated to RGB
- For 24 bit color, R has 8 bits, G has 8 bits, and B has 8 bits
  - Each color has 256 possible values
  - 0 is black, 255 is white, everything in between is some shade of red, green, or blue for each respective color
- A screen resolution of 1920 by 1080 is a common value
- This means there are 2,073,600 pixels
- If each pixel is 24 bits (3 bytes), then the amount of memory required to store that screen is 2 * 2,073,600 = 6,220,800
- 16 bit color will cut the size to 2/3 * original
- 8 bit color will cut the size to 1/3 * original

Palettes

- For any number of colors less than 16 million, the .bmp file format uses a palette
  - (24 bit = 2^{24} = 16,777,216)
- Rather than store the actual color of each pixel, the bitmap stores the index of a palette
- Each palette entry has 24 bit color
  - For an 8 bit image, there are 256 palette entries, each of which can store any of the 16 million colors
  - For a 16 bit image, there are 65536 palette entries, each holding one of 16 million colors
  - There is some overhead to this approach, either 3 * 256 or 3 * 65536 for the palette table
  - However, all 16 million colors are available

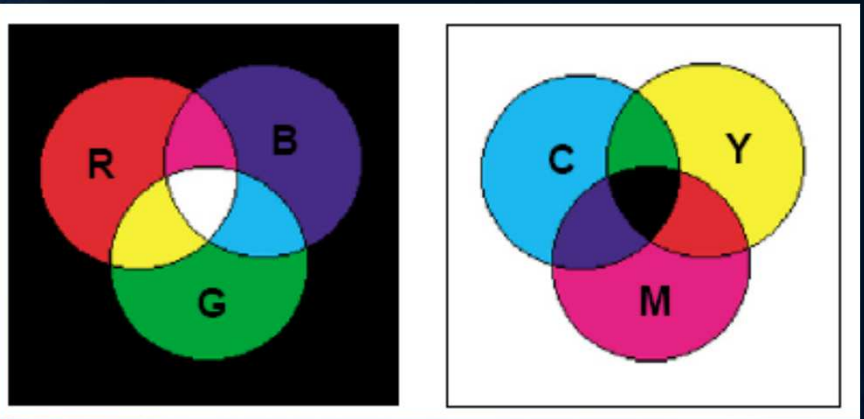
Color Models

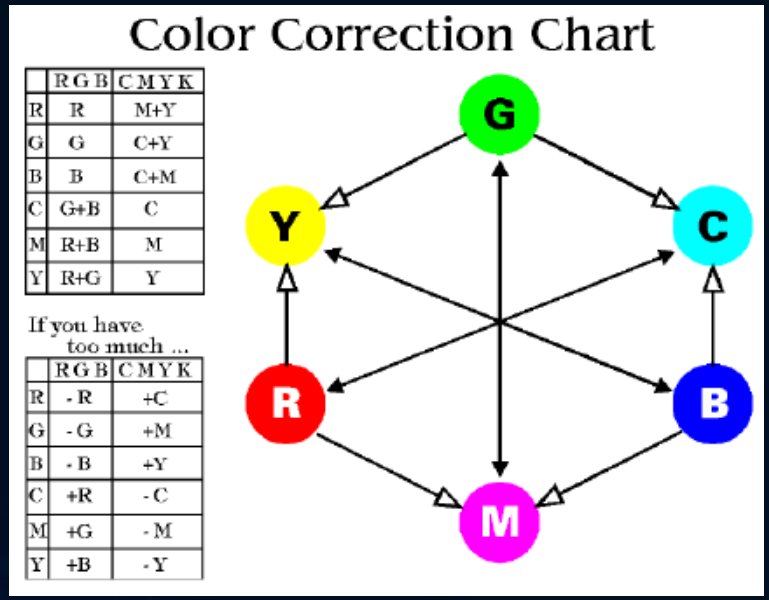
- There are other models besides RGB
- CMY, HIS, YIQ, YUV, YCrCb to name a few
- Cyan, Magenta, Yellow is just the negative of RGB
  - Printers typically use CMY and K for black
- In formulas below, 255 is valid only for 8 bit colors

$$C = 255 - R$$
$$M = 255 - G$$
$$Y = 255 - B$$

Color Models

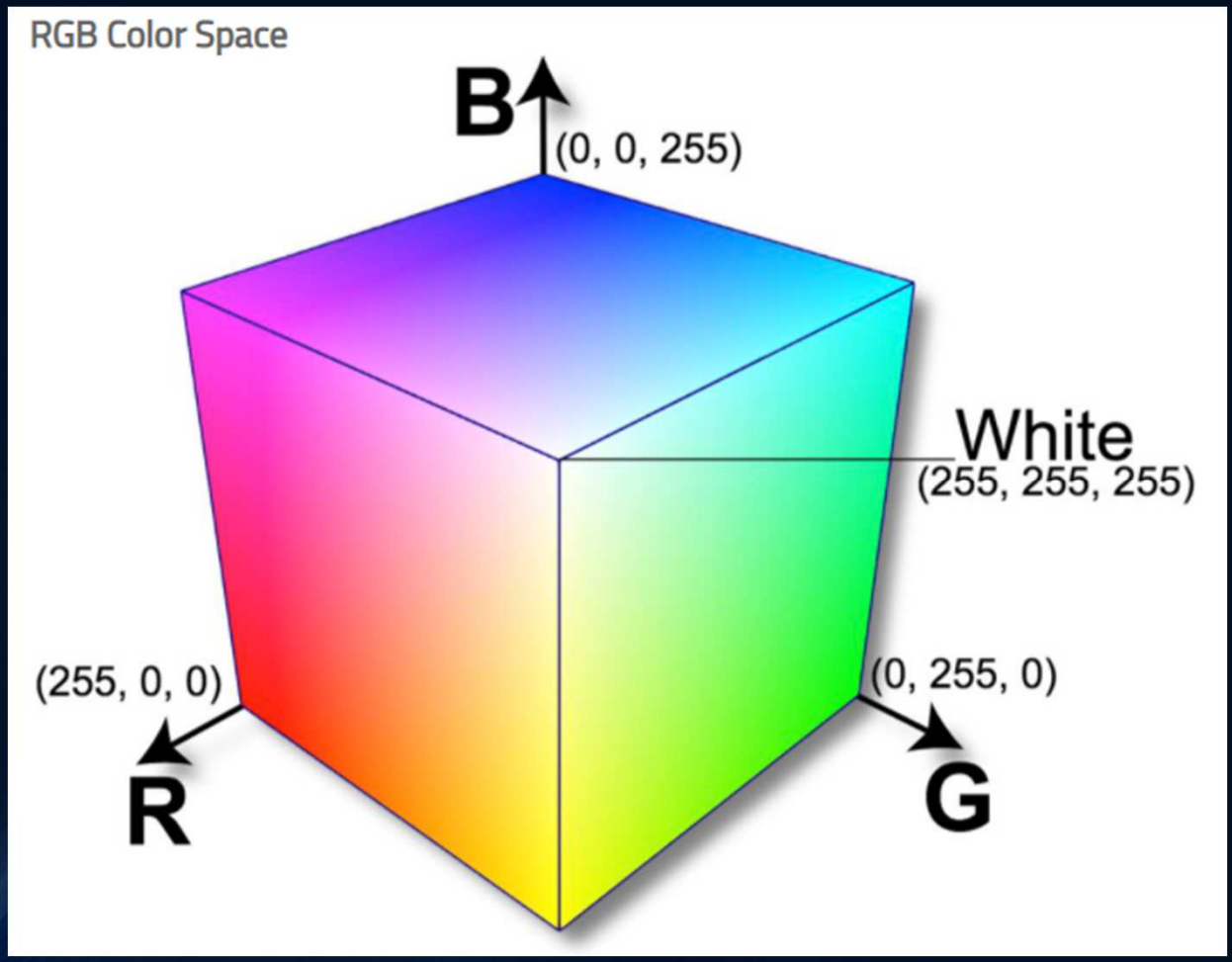
- There are two color models to consider:
  - Additive (as in a monitor)
    - Red, Green, Blue
    - As you add colors, the intensity gets brighter
    - Adding all three results in white light
  - Subtractive (as in a printer)
    - Cyan, Magenta, Yellow
    - As you add colors, the intensity darkens
    - Adding all three results in black





Color Models

- RGB



Color Models

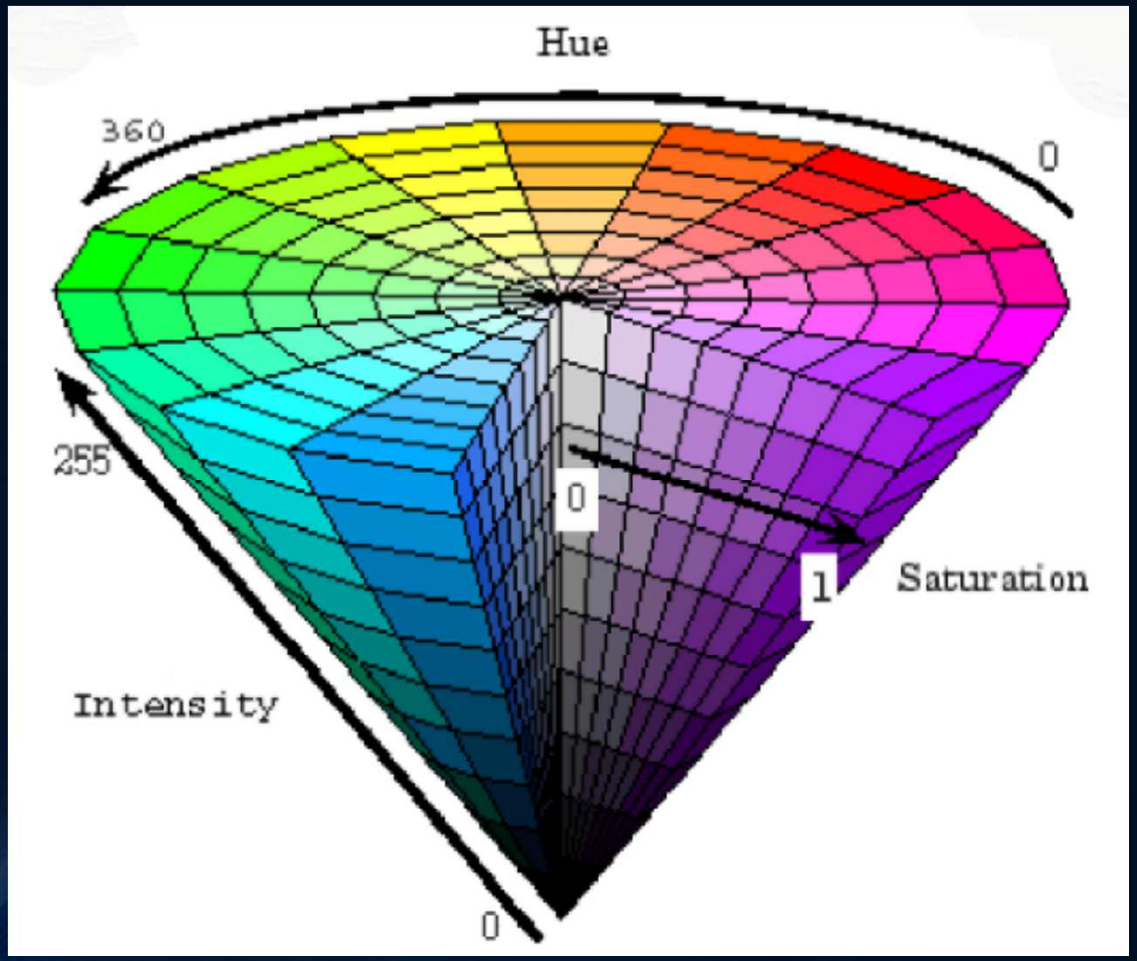
- Hue-Saturation-Intensity (HSI) Model
  - Intensity: brightness of the color
  - Hue: shade of color (green, blue, orange, etc.)
  - Saturation: measure of how much "white" is in the color (pink is red with more white)
- HSI based on heuristics relating to human perception
  - Also separates out intensity from chromaticity
- Humans can view chromaticity as the hue and saturation parts of HSI

Color Models

- H- Hue is the color (yellow, red, purple, etc)
- S - Saturation how much a pure color is diluted by white light
- I - Intensity ranges from low (black) to high (white)
- HSI is more convenient to some graphics designers because it provides direct control of brightness and hue
- Chromaticity values are more closely associated with the intrinsic character of a surface rather than the source that is lighting it
- Because of this latter property, and the fact that HSI also normalizes the brightness, HSI can provide a better environment for automated computer vision algorithms including object recognition

Color Models

- HSI



YIQ / YUV

- In practice, luminance is encoded using more bits as human vision system is more sensitive to luminance or brightness variations than to the chromaticity values
- There is also a similar model called YUV that is used in some digital video and JPEG and MPEG compression algorithms
- Conversion formulas

$$Y = 0.30R + 0.59G + 0.11B$$
$$I = 0.60R - 0.28G - 0.32B$$
$$Q = 0.21R - 0.52G + 0.31B$$

$$Y = 0.30R + 0.59G + 0.11B$$
$$U = 0.493 * (B - Y)$$
$$V = 0.877 * (R - Y)$$

YC_RC_B

- YC_rC_b is the method of encoding color video while maintaining compatibility with black-and-white video
  - Y is the luminance (brightness), C_r & C_b are chrominance (hue, saturation)
  - when color televisions were invented, the government insisted that the color signal be compatible with the (millions) of black and white tvs
  - by separating luminance from chrominance, that requirement was met
- Used for MPEG1 and MPEG2
- DVD players often have these 3 outputs
- Uses less bandwidth than three separate video signals in RGB
- Compression algorithms can achieve a higher degree of compression with YC_rC_b encoding

YC_RC_B

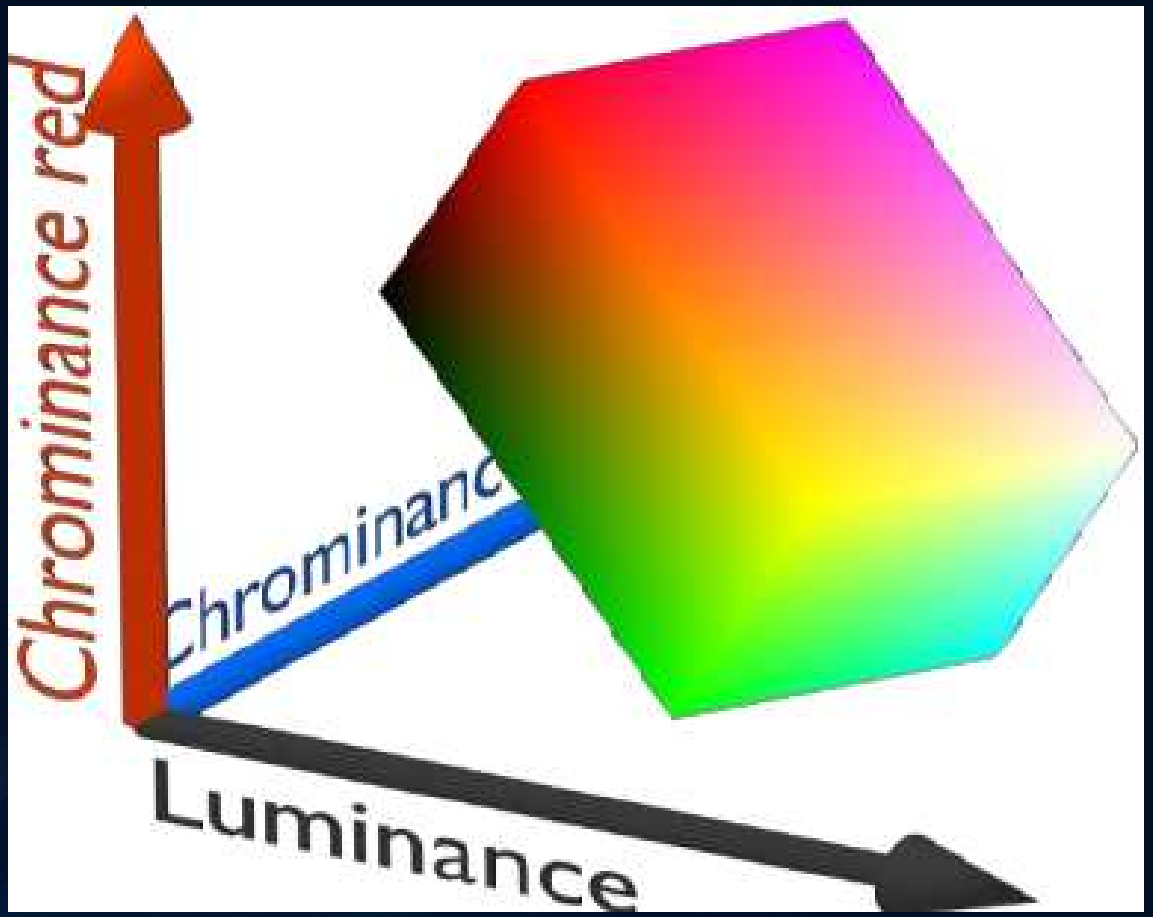
- YC_RC_B

$$Y = 0.2989R + 0.5866G + 0.1145B$$
$$C_b = -0.1687R - 0.3313G + 0.5B + 2^4$$
$$C_r = 0.5R - 0.4187G - 0.0813B + 2^4$$

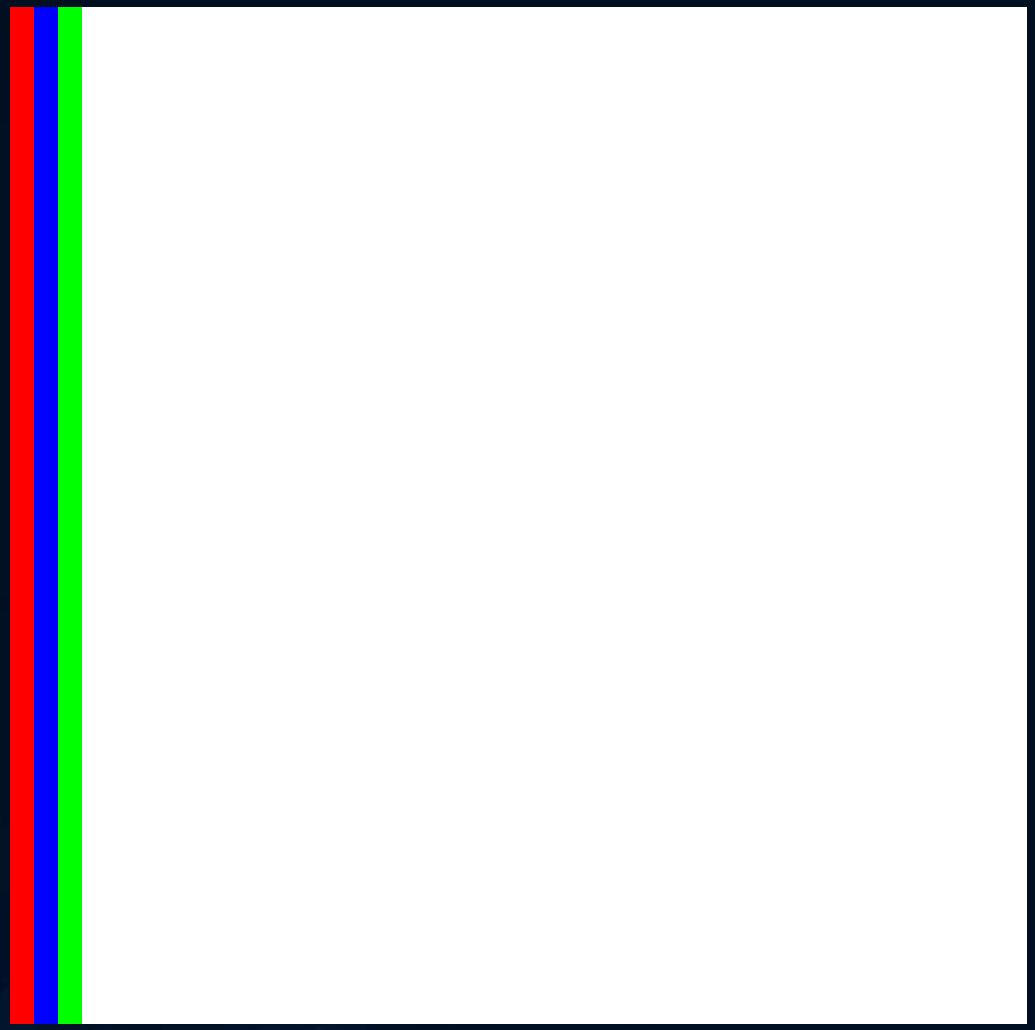
$$R = Y + 1.402C_r$$
$$G = Y - 0.34414(C_b - 2^4) - 0.71414(C_r - 2^4)$$
$$B = Y + 1.722(C_b - 2^4)$$

YC_RC_B

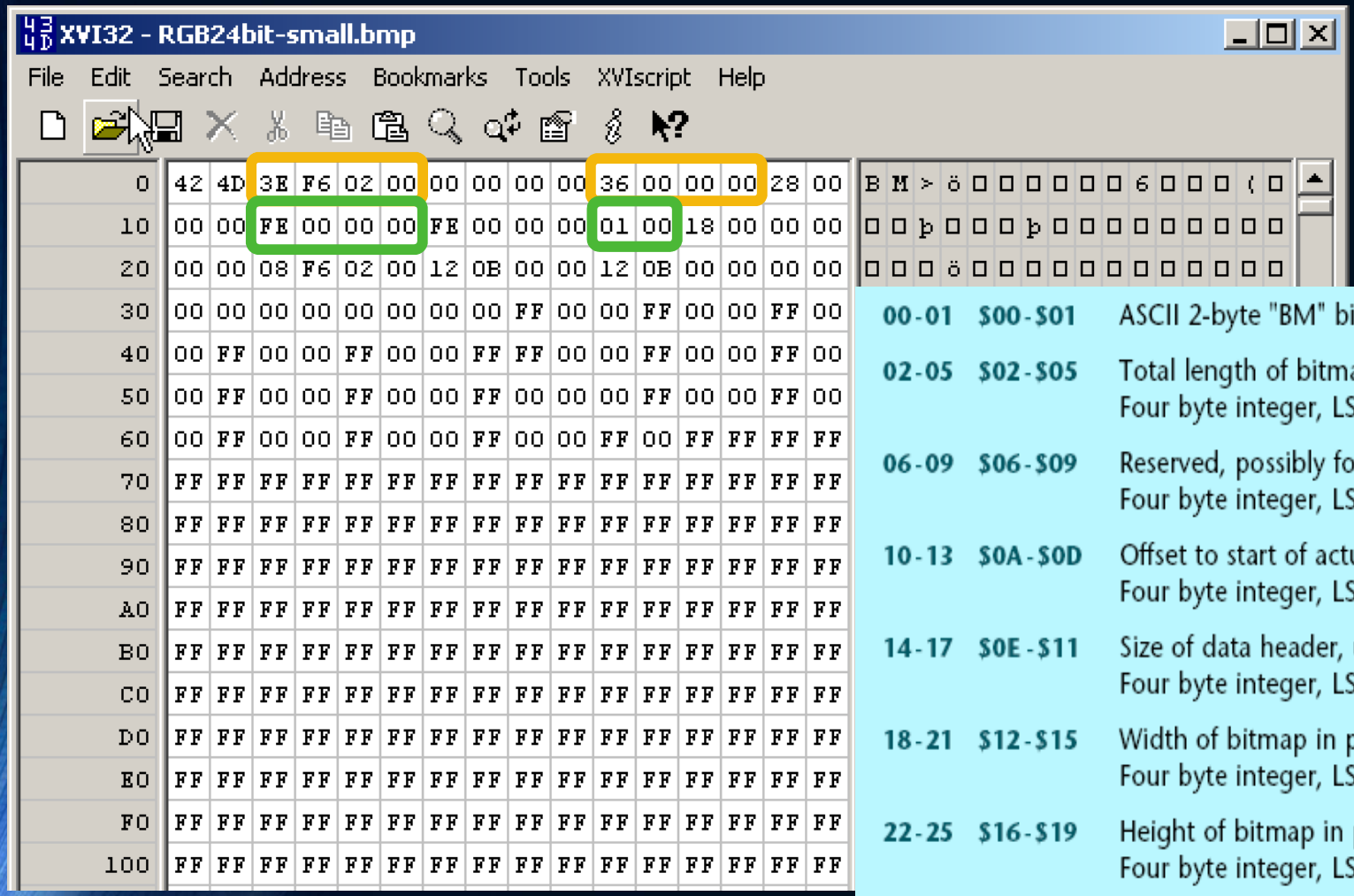
- YC_RC_B



Bitmap Example



24-Bit Bitmap



00-01	$00-$01	ASCII 2-byte "BM" bitmap identifier.
02-05	$02-$05	Total length of bitmap file in bytes. Four byte integer, LSB first.
06-09	$06-$09	Reserved, possibly for image id or revision. Four byte integer, LSB first.
10-13	$0A-$0D	Offset to start of actual pixel data. Four byte integer, LSB first.
14-17	$0E-$11	Size of data header, usually 40 bytes. Four byte integer, LSB first.
18-21	$12-$15	Width of bitmap in pixels. Four byte integer, LSB first.
22-25	$16-$19	Height of bitmap in pixels. Four byte integer, LSB first.
26-27	$1A-$1B	Number of color planes. Usually 01 Two byte integer, LSB first.

24-Bit Bitmap

XVI32 - RGB24bit-small.bmp

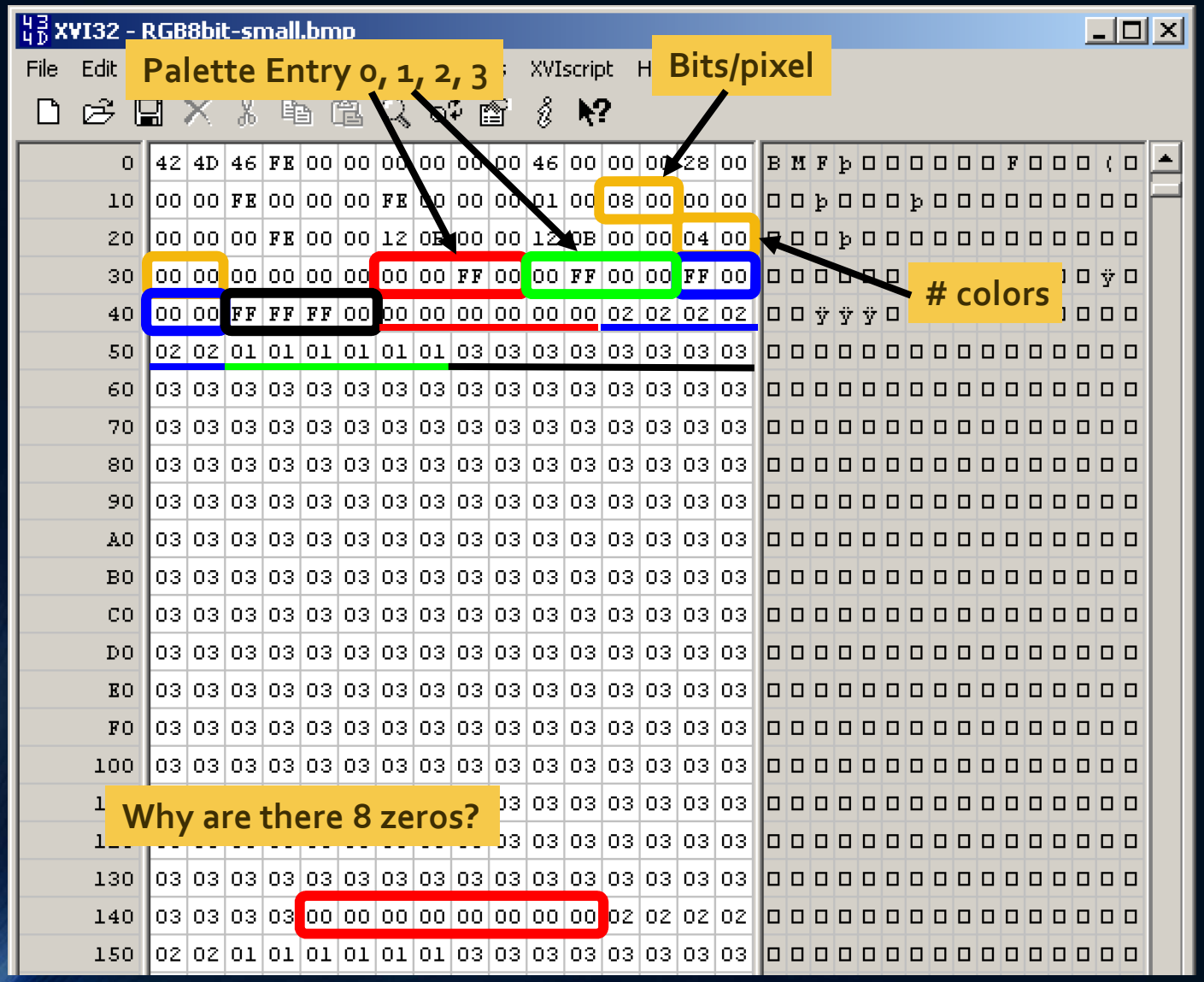
File Edit Search Address Bookmarks Tools XVIscript Help

0	42	4D	3E	F6	02	00	00	00	00	00	36	00	00	00	28	00
10	00	00	FE	00	00	00	FE	00	00	00	01	00	18	00	00	00
20	00	00	08	F6	02	00	12	0B	00	00	12	0B	00	00	00	00
30	00	00	00	00	00	00	00	00	FF	00	00	FF	00	00	FF	00
40	00	FF	00	00	FF	00	00	FF	FF	00	00	FF	00	00	FF	00
50	00	FF	00	00	FF	00	00	FF	00	00	00	FF	00	00	FF	00
60	00	FF	00	00	FF	00	00	FF	00	00	FF	00	FF	FF	FF	FF
70	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
80	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
90	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
A0	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
B0	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
C0	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
D0	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
E0	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
F0	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
100	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF

28-29	$1C-$1D	Number of bits per pixel. Sets color mode. Two byte integer, LSB first. 1 - Monochrome 4 - 16 lookup colors 8 - 256 lookup colors 16 - 65,536 lookup colors 24 - 16,777,216 RGB colors 32 - 16,777,216 RGB colors + alpha
30-33	$1E-$21	Non-lossy compression mode in use Four byte integer, LSB first. 0 - None 1 - 8-bit run length encoded 1 - 4-bit run length encoded
34-37	$22-$25	Size of stored pixel data Four byte integer, LSB first.
38-41	$26-$29	Width resolution in pixels per meter Four byte integer, LSB first.
42-45	$2A-$2D	Height resolution in pixels per meter Four byte integer, LSB first.
46-49	$2E-$31	Number of colors actually used. Four byte integer, LSB first.
50-53	$32-$35	Number of important colors Four byte integer, LSB first.

0x2F608 = 194056 bytes = 254 * 254 * 3 + 508

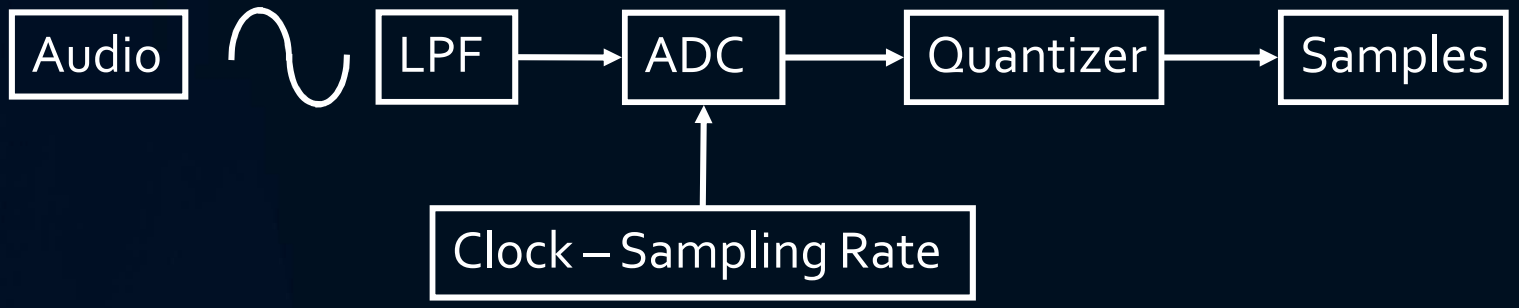
8-Bit Paletted Bitmap



8-Bit Paletted Bitmap - Grayscale

0	42	4D	36	02	01	00	00	00	00	00	36	04	00	00	28	00
10	00	00	FE	00	00	00	FE	00	00	00	01	00	08	00	00	00
20	00	00	00	FE	00	00	12	0B	00	00	12	0B	00	00	00	01
30	00	00	00	00	00	00	00	00	00	00	01	01	01	00	02	02
40	02	00	03	03	03	00	04	04	04	00	05	05	05	00	06	06
50	06	00	07	07	07	00	08	08	08	00	09	09	09	00	0A	0A
60	0A	00	0B	0B	0B	00	0C	0C	0C	00	0D	0D	0D	00	0E	0E
70	0E	00	0F	0F	0F	00	10	10	10	00	11	11	11	00	12	12
80	12	00	13	13	13	00	14	14	14	00	15	15	15	00	16	16
90	16	00	17	17	17	00	18	18	18	00	19	19	19	00	1A	1A
A0	1A	00	1B	1B	1B	00	1C	1C	1C	00	1D	1D	1D	00	1E	1E
B0	1E	00	1F	1F	1F	00	20	20	20	00	21	21	21	00	22	22
C0	22	00	23	23	23	00	24	24	24	00	25	25	25	00	26	26
D0	26	00	27	27	27	00	28	28	28	00	29	29	29	00	2A	2A

Audio

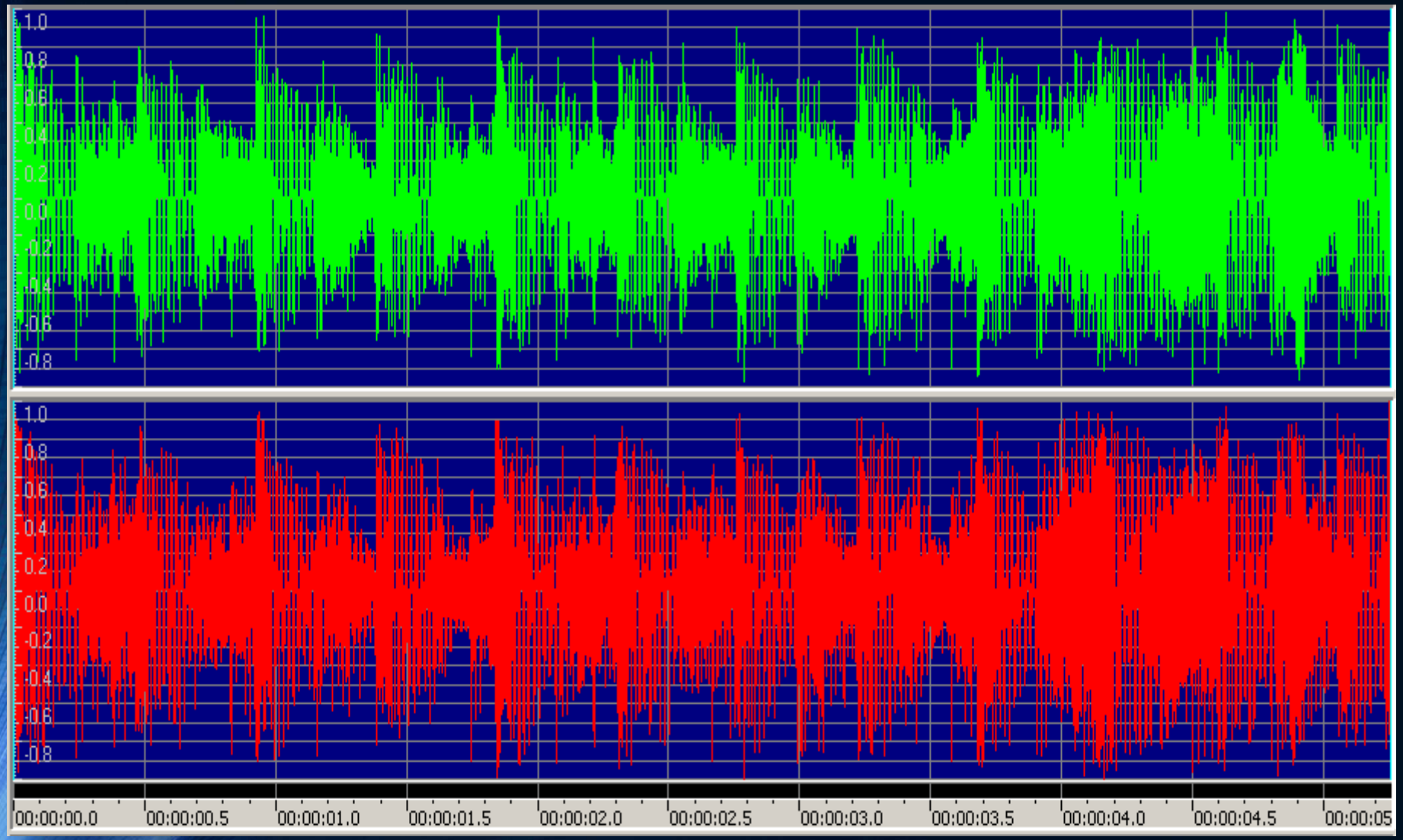


- Audio is digitized by converting the analog voltage level into discrete values
- LPF – Low Pass Filter –removes frequencies higher than the Nyquist rate
  - It's like turning down the treble on your stereo
- Clock is the sampling rate
- If clock is 44.1 KHz, LPF removes frequencies above 22.05 KHz
  - In practice, you need a little extra removed, so 20 KHz is the cutoff
- Sampling rate determines the frequency response
  - Too low and it will sound like an AM radio
  - Tradeoff is in data storage space

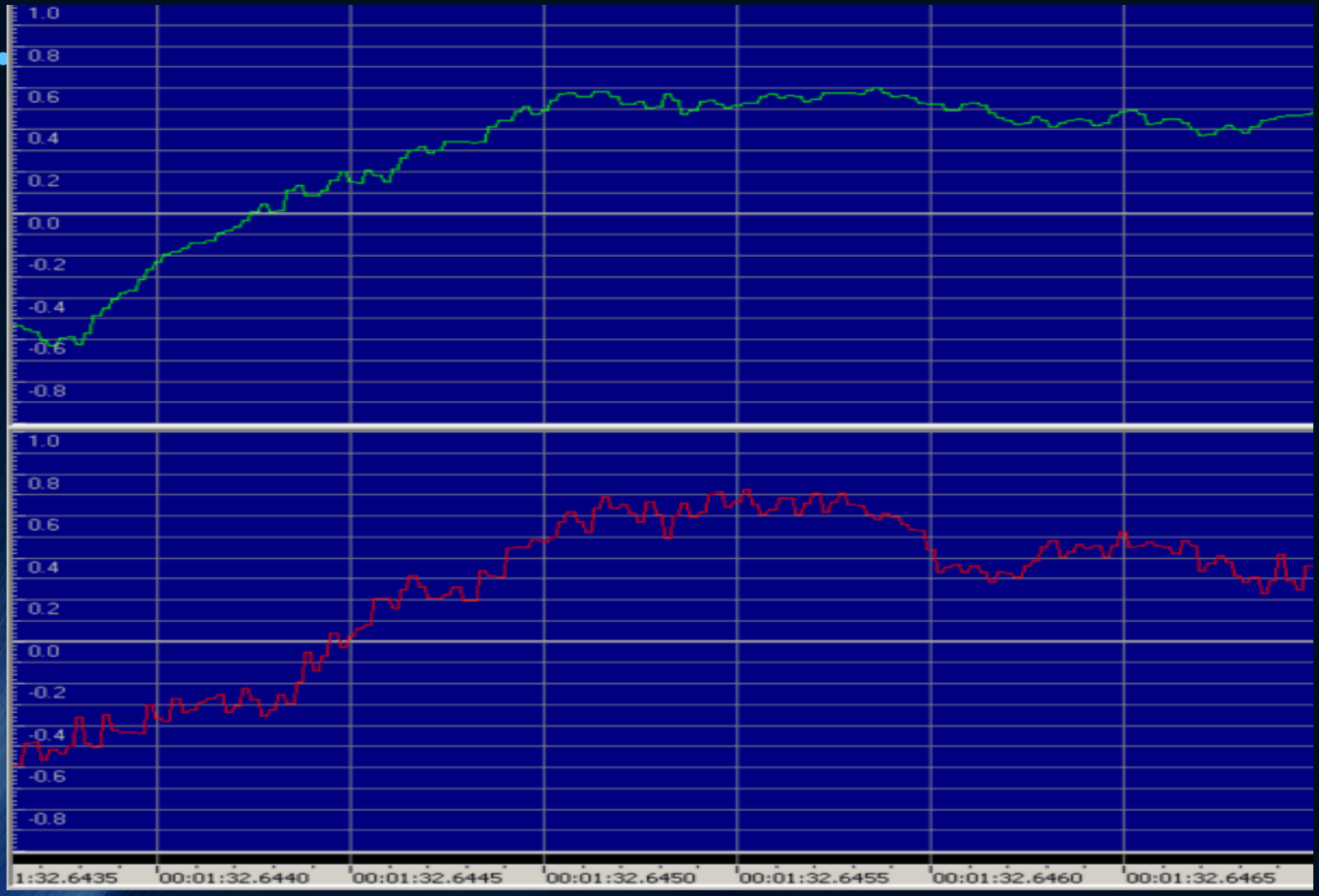
Audio

- The range of values depends upon how many bits per sample
  - For CD quality, 16 bits are used (-32768 to +32767)
  - For voice quality, 8 bits are used (-128 to +127)
- The tradeoff is in the amount of data to store
  - CD Quality: 2 channels * 16 bits/sample * 44100 samples/sec
    - = 176400 bytes/sec
  - Voice Quality: 1 channel * 8 bits/sample * 8000 samples/sec
    - = 8000 bytes/sec
- The sampling rate and the number of bits/sample together determine the overall fidelity

Audio

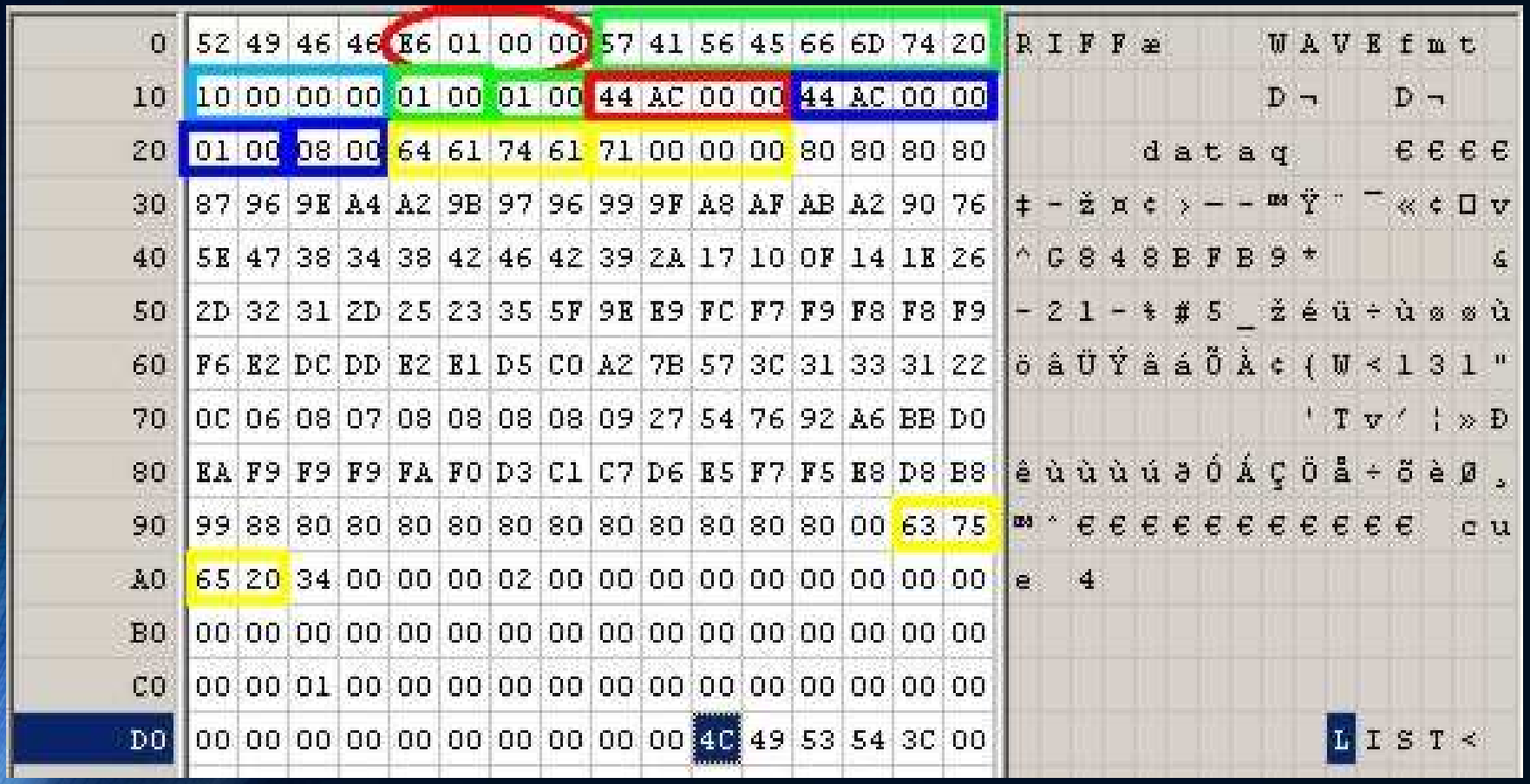

1.0
0.8
0.6
0.4
0.2
0.0
-0.2
-0.4
-0.6
-0.8
00:00:00.0
00:00:00.5
00:00:01.0
00:00:01.5
00:00:02.0
00:00:02.5
00:00:03.0
00:00:03.5
00:00:04.0
00:00:04.5
00:00:05

Audio


1.0
0.8
0.6
0.4
0.2
0.0
-0.2
-0.4
-0.6
-0.8
1.0
0.8
0.6
0.4
0.2
0.0
-0.2
-0.4
-0.6
-0.8
1:32.6435
00:01:32.6440
00:01:32.6445
00:01:32.6450
00:01:32.6455
00:01:32.6460
00:01:32.6465

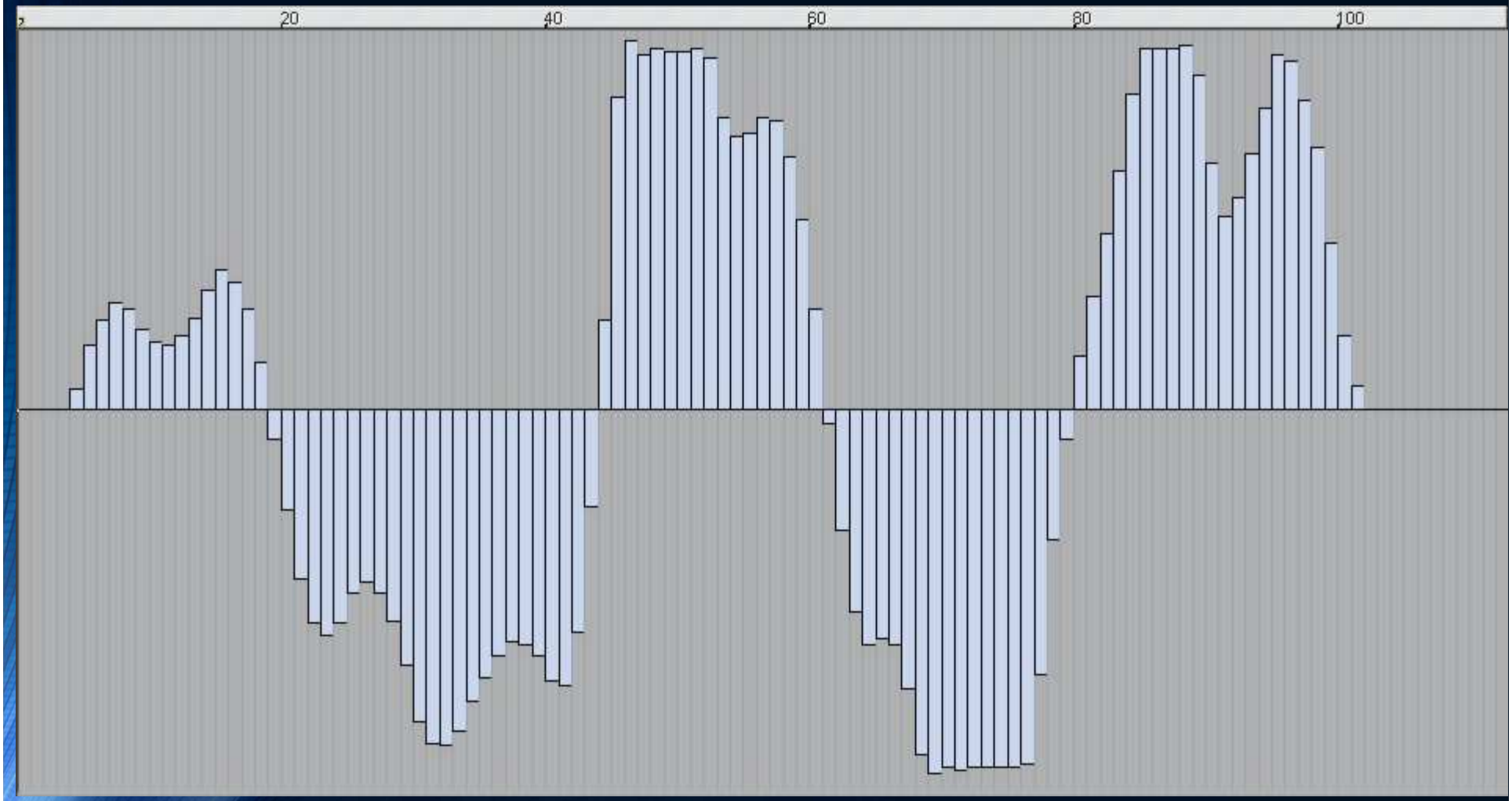
Audio

8-Bit Wave



0	52	49	46	46	E6	01	00	00	57	41	56	45	66	6D	74	20
10	10	00	00	00	01	00	01	00	44	AC	00	00	44	AC	00	00
20	01	00	08	00	64	61	74	61	71	00	00	00	80	80	80	80
30	87	96	9E	A4	A2	9B	97	96	99	9F	A8	AF	AB	A2	90	76
40	5E	47	38	34	38	42	46	42	39	2A	17	10	0F	14	1E	26
50	2D	32	31	2D	25	23	35	5F	9E	E9	FC	F7	F9	F8	F8	F9
60	F6	E2	DC	DD	E2	E1	D5	C0	A2	7B	57	3C	31	33	31	22
70	0C	06	08	07	08	08	08	08	09	27	54	76	92	A6	BB	D0
80	EA	F9	F9	F9	FA	F0	D3	C1	C7	D6	E5	F7	F5	E8	D8	B8
90	99	88	80	80	80	80	80	80	80	80	80	80	80	00	63	75
A0	65	20	34	00	00	00	02	00	00	00	00	00	00	00	00	00
B0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
C0	00	00	01	00	00	00	00	00	00	00	00	00	00	00	00	00
D0	00	00	00	00	00	00	00	00	00	00	4C	49	53	54	3C	00

8-Bit Wave



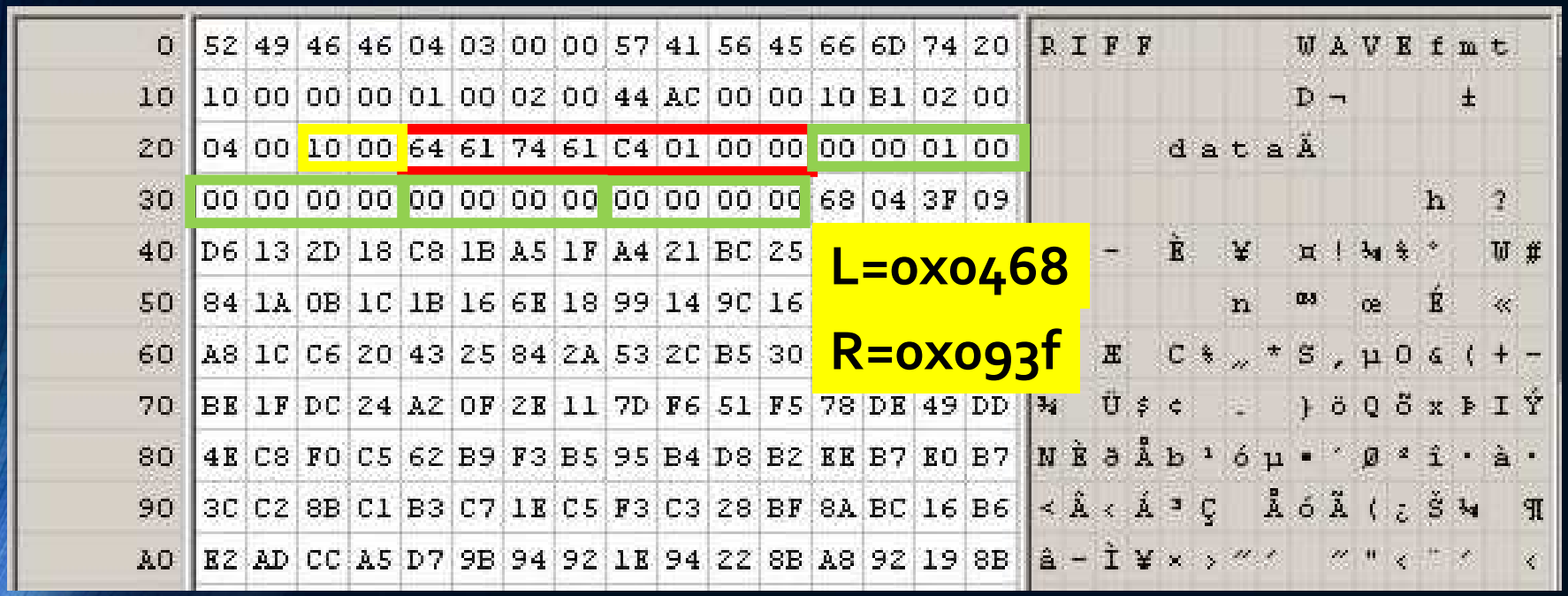
16-Bit Wave

- Samples for each channel are interleaved

Bits/sample

"data" & size

4 samples at midpoint

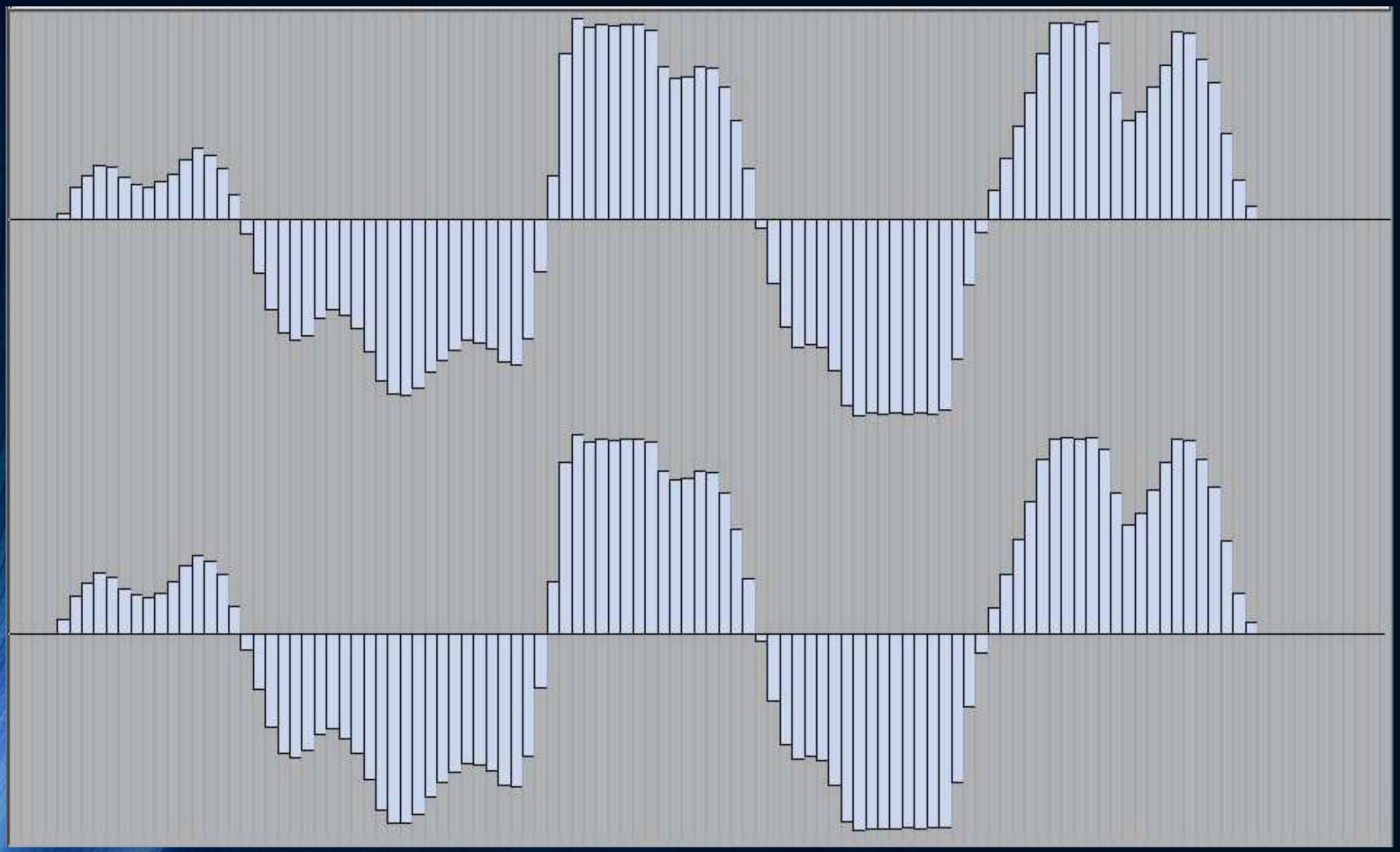


0	52	49	46	46	04	03	00	00	57	41	56	45	66	6D	74	20
10	10	00	00	00	01	00	02	00	44	AC	00	00	10	B1	02	00
20	04	00	10	00	64	61	74	61	C4	01	00	00	00	00	01	00
30	00	00	00	00	00	00	00	00	00	00	00	00	68	04	3F	09
40	D6	13	2D	18	C8	1B	A5	1F	A4	21	BC	25				
50	84	1A	0B	1C	1B	16	6E	18	99	14	9C	16				
60	A8	1C	C6	20	43	25	84	2A	53	2C	B5	30				
70	BE	1F	DC	24	A2	0F	2E	11	7D	F6	51	F5	78	DE	49	DD
80	4E	C8	F0	C5	62	B9	F3	B5	95	B4	D8	B2	EE	B7	E0	B7
90	3C	C2	8B	C1	B3	C7	1E	C5	F3	C3	28	BF	8A	BC	16	B6
A0	E2	AD	CC	A5	D7	9B	94	92	1E	94	22	8B	A8	92	19	8B

L=0x0468

R=0x093f

16-Bit Wave



HANDS - ON

- Hex Editor Basics
- MD5 and SHA256 Hashing
- WBH for Histogram and Entropy Analysis

Questions &Comments