

JE4XX

Multiple Channel JPEG Baseline
Encoder IP-Core

Users Manual

Rev 1.0

Table of Contents

1.	The JPEG Encoding Standard	4
1.1	Level Shift	4
1.2	Blocks.....	4
1.3	Components.....	5
1.4	DCT.....	6
1.5	Zigzag Order	7
1.6	Quantization.....	7
1.7	Differential DC Encoding.....	7
1.8	Run / Size Symbols Encoding.....	8
1.9	Huffman Coding	9
2.	The JPEG File Format.....	10
2.1	SOI Start of Image	10
2.2	DQT Define Quantization Table	10
2.3	SOF Start of Frame.....	11
2.4	DHT Define Huffman Table	11
2.5	SOS Start of Scan.....	12
2.6	DNL Define Number of Lines	12
2.7	EOI End Of Image.....	12
3.	Encoding Example	13
4.	The JE4XX JPEG Encoder	15
4.1	Technical Features.....	15
4.2	Differences Between the JE430, JE440 and JE490 Cores	15
4.3	Needed Resources	16
4.4	Functional Description.....	17
4.5	Typical Application	18
4.6	JE4XX Core Entity	19
4.7	Core Interface	21
4.7.1	Control Interface.....	22
4.7.2	Sample Input Interface.....	22
4.7.3	Compressed Data Output Interface.....	24
4.7.4	Tables Programming Interface.....	25
5.	Quantization Tables	27
5.1	Default Quantization Table for Luminance	28
5.2	Default Quantization Table for Chrominance	28
5.3	Changing the Compression Rate	29
6.	Huffman Tables.....	30
6.1	Default Huffman Table for Luminance.....	30
6.2	Default Huffman Table for Chrominance.....	31
7.	Hints and Restrictions	32
7.1	Initialization	32
7.2	Image Size	32
8.	Literature and Links.....	34
8.1	Documents from the Internet.....	34
8.2	Hard Book	34
8.3	Internet Links.....	34
9.	Revision Info	35

List of Figures:

Figure 1: JPEG Encoding Structure.....	4
Figure 2: Dividing Image in Blocks	5
Figure 3: Color Image as Blocks and Components	5
Figure 4: Color Components	6
Figure 5: Subsampled Color Components.....	6
Figure 6: Zigzag Order from 8x8 Block.....	7
Figure 7: DC Coefficient Size, VLI Symbols	8
Figure 8: AC Coefficients Run/Size, VLI Symbols	9
Figure 9: Block with Original Samples	13
Figure 10: Block with Level Shifted Samples.....	13
Figure 11: Coefficients After DCT.....	13
Figure 12: Quantization Table	13
Figure 13: Quantized Coefficients	13
Figure 14: Values in Zigzag Order.....	13
Figure 15: Huffman Encoding.....	14
Figure 16: Encoder Structure	17
Figure 17: Block Diagram of a Typical Application	18

List of Tables:

Table 1: Size Symbols.....	8
Table 2: AC Run / Size Symbols	9
Table 3: Needed Resources for JE430.....	16
Table 4: Needed Resources for JE440.....	16
Table 5: Core Entity Signals	21
Table 6: Zigzag Order.....	27
Table 7: Default Quantization Table for Luminance.....	28
Table 8: Default Quantization Table for Chrominance.....	28
Table 9: Luminance Number of DC Codes.....	30
Table 10: Luminance DC Symbol to Code Assignment.....	30
Table 11: Luminance Number of AC Codes	30
Table 12: Luminance AC Symbol to Code Assignment	30
Table 13: Chrominance Number of DC Codes	31
Table 14: Chrominance DC Symbol to Code Assignment	31
Table 15: Chrominance Number of AC Codes	31
Table 16: Chrominance AC Symbol to Code Assignment	31

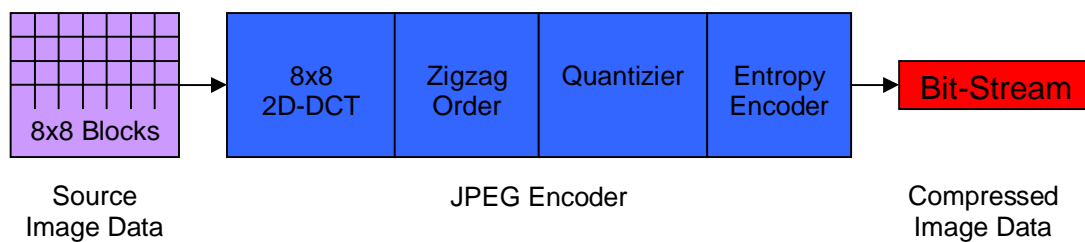
1. The JPEG Encoding Standard

The JPEG standard defines four modes of operation:

- Sequential DCT-based
- Progressive DCT-based
- Hierarchical
- Sequential lossless

The most used mode is the Sequential DCT-based with Huffman coding, which will be described by the following overview.

Figure 1: JPEG Encoding Structure



Typical compress rates are values from 10 to 15.

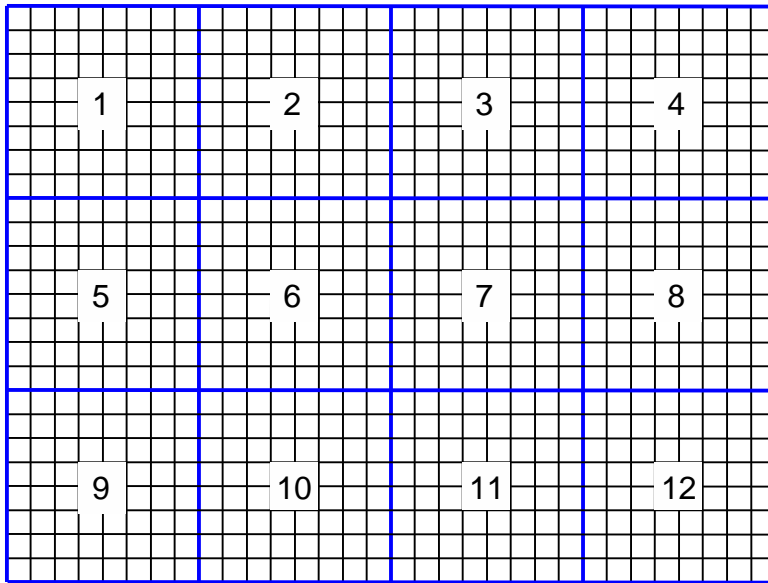
1.1 Level Shift

The image samples must be in the two's complement format with a size of 8 bit. If the samples are in integer format, a value of 80h (800h with 12 bit precision) must be subtracted.

1.2 Blocks

The complete image is divided into 8x8 blocks, starting in the top left row. The blocks are read out in the same order, beginning in the top left corner to the top right corner, and then continuing the next eight rows until all done. If the number of pixels / lines are not a multiple of 8, the last pixel / line is repeated until the block is completed. Any extra pixels / lines are discarded by the decoding process.

Figure 2: Dividing Image in Blocks

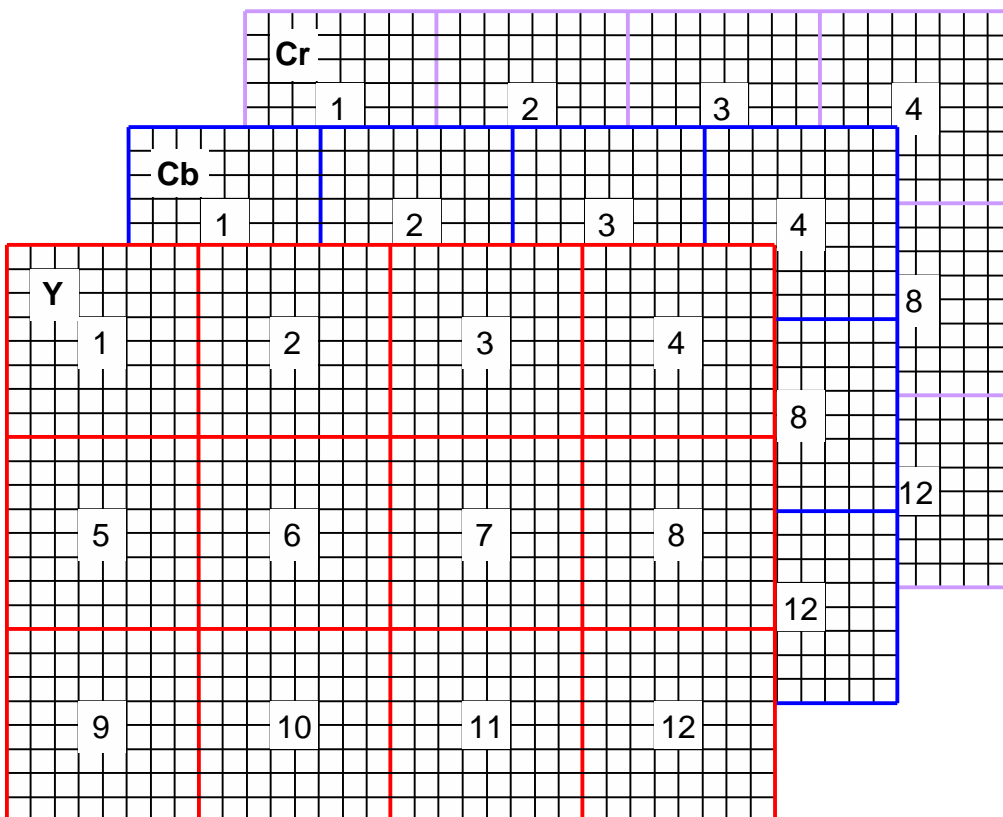


1.3 Components

The parts of a pixel are named components. A mono chrome image has only one and a color image three components. One ore more components may be subsampled but usually this is done, only with the chrominance components.

Here a color image with the components Y, Cb and Cr, no subsampling.

Figure 3: Color Image as Blocks and Components



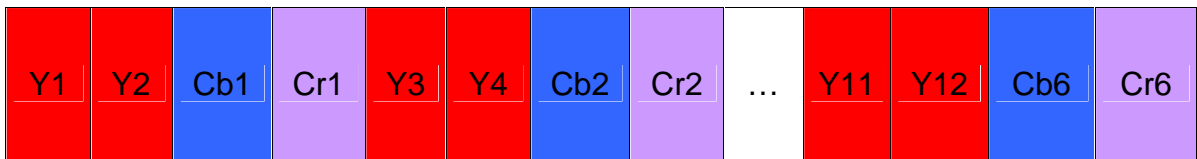
The blocks are processed in the order components and then the indexes.

Figure 4: Color Components



When the chrominance components (Cb and Cr) are subsampled with a value of 2 then one chrominance sample pair is used for two luminance samples.

Figure 5: Subsampled Color Components



1.4 DCT

The Discrete Cosine Transformation (DCT) transforms the 64 samples array of an 8x8 block into an 8x8 array of coefficients. Doing this by using the following equation:

$$S(v,u) = \frac{C(v)}{2} * \frac{C(u)}{2} * \sum_{y=0}^7 \sum_{x=0}^7 S(y,x) * \cos\left(\frac{(2x+1)u\pi}{16}\right) * \cos\left(\frac{(2y+1)v\pi}{16}\right)$$

$$C(u) = \frac{1}{\sqrt{2}} \text{when } u = 0 \text{ else } 1$$

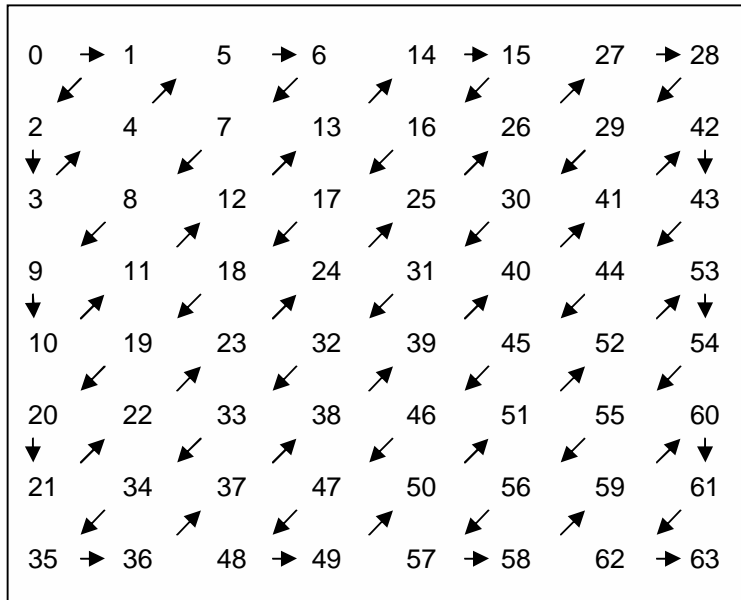
$$C(v) = \frac{1}{\sqrt{2}} \text{when } v = 0 \text{ else } 1$$

The two indices x and y represent the sample placement, the indices u and v represent the coefficients frequencies. The top left element (S(0,0)) is the DC coefficient, the bottom right element (S(7,7)) is the coefficient with the highest horizontal and vertical frequencies.

1.5 Zigzag Order

The coefficients of a transformed block are rearranged in that way that there are sorted from DC to the highest frequencies. The goal is to have a large number of subsequent zeros for the later Run / Size encoding.

Figure 6: Zigzag Order from 8x8 Block



1.6 Quantization

The Quantization reduces the accuracy of the coefficients. This is done by dividing each coefficient by the value in the Quantization table with the same indices. When using larger values in the table consequence a higher compression rate, but also more artifacts. On the other hand, the using of lower values results in less compression and lossy. The higher frequently coefficients are lower and the table values for these coefficients are larger, so much of quantized coefficients become zero. This is important for a high compression rate. The Quantization stage is the mean reason for lossy, all other stages are lossless (the DCT is a little lossy cause the rounding errors).

1.7 Differential DC Encoding

The DC coefficient represents the average value of all 64 samples. Because the average differs only slightly from one block to the next, the DC coefficient is differential encoded. From the DC coefficient, is subtracted the DC coefficient from the previous block of the same component. For the first block in the image, is a predicted value of zero defined. The difference is usually a short value and results so in a short integer in the following symbol encoding.

1.8 Run / Size Symbols Encoding

The Zigzag ordered values are transformed in a Run/Size (DC only Size) symbol and variable length integers bits (VLI). The Size symbol is taken from the given table:

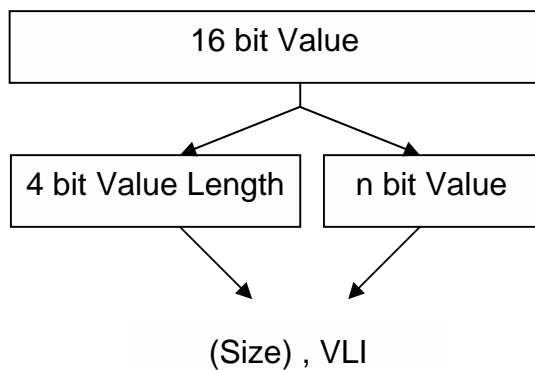
Table 1: Size Symbols

Size	Values range
0	0
1	-1, 1
2	-3 ... -2, 2 ... 3
3	-7 ... -4, 4 ... 7
4	-15 ... -8, 8 ... 15
5	-31 ... -16, 16 ... 31
6	-63 ... -32, 32 ... 63
7	-127 ... -64, 64 ... 127
8	-255 ... -128, 128 ... 255
9	-511 ... -256, 256 ... 511
10	-1023 ... -512, 512 ... 1023
11	-2047 ... -1024, 1024 ... 2047
12	-4095 ... -2048, 2048 ... 4095
13*	-8191 ... -4096, 4096 ... 8191
14*	-16383 ... -8192, 8192 ... 16383
15*	-32767 ... -16384, 16384 ... 32767

*Defined only for 12 bit precision.

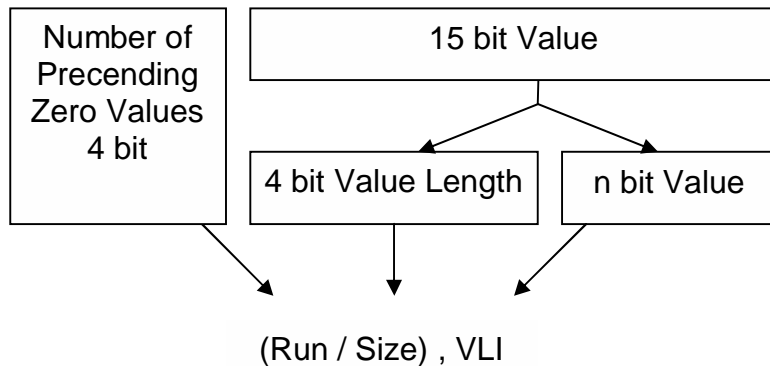
The Size symbol represents the number of the VLI bits. The VLI bits are generated by adding a one, if the value is negative. Then the VLI bits are truncate to n lower bits, where n is the Size value.

Figure 7: DC Coefficient Size, VLI Symbols



The AC values are coded in a Run/Size and VLI symbol, but only the non-zero values. If the value is zero, then only the Run part of the next non-zero value is incremented. Because the Run part is four bit, only 15 consecutive zeros can be coded in the Run/Size symbol. If 16 consecutive zeros appear, the special Zero Run Length (ZRL) symbol is inserted. When at one point all remaining values are zero, then the end of block (EOB) symbol is inserted, and the block is finished.

Figure 8: AC Coefficients Run/Size, VLI Symbols



1.9 Huffman Coding

The Run/Size symbols get a code from the Huffman table. The idea behind the Huffman coding is, to use codes with variable length. Symbols with a large number of occurs get short codes and symbols with less occurs get longer codes.

Table 2: AC Run / Size Symbols

		Size														
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E
Run	0	EOB	0/1	0/2	0/3	0/4	0/5	0/6	0/7	0/8	0/9	0/10	0/11*	0/12*	0/13*	0/14*
	1		1/1	1/2	1/3	1/4	1/5	1/6	1/7	1/8	1/9	1/10	1/11*	1/12*	1/13*	1/14*
	2		2/1	2/2	2/3	2/4	2/5	2/6	2/7	2/8	2/9	2/10	2/11*	2/12*	2/13*	2/14*
	3		3/1	3/2	3/3	3/4	3/5	3/6	3/7	3/8	3/9	3/10	3/11*	3/12*	3/13*	3/14*
	4		4/1	4/2	4/3	4/4	4/5	4/6	4/7	4/8	4/9	4/10	4/11*	4/12*	4/13*	4/14*
	5		5/1	5/2	5/3	5/4	5/5	5/6	5/7	5/8	5/9	5/10	5/11*	5/12*	5/13*	5/14*
	6		6/1	6/2	6/3	6/4	6/5	6/6	6/7	6/8	6/9	6/10	6/11*	6/12*	6/13*	6/14*
	7		7/1	7/2	7/3	7/4	7/5	7/6	7/7	7/8	7/9	7/10	7/11*	7/12*	7/13*	7/14*
	8		8/1	8/2	8/3	8/4	8/5	8/6	8/7	8/8	8/9	8/10	8/11*	8/12*	8/13*	8/14*
	9		9/1	9/2	9/3	9/4	9/5	9/6	9/7	9/8	9/9	9/10	9/11*	9/12*	9/13*	9/14*
	10		10/1	10/2	10/3	10/4	10/5	10/6	10/7	10/8	10/9	10/10	10/11*	10/12*	10/13*	10/14*
	11		11/1	11/2	11/3	11/4	11/5	11/6	11/7	11/8	11/9	11/10	11/11*	11/12*	11/13*	11/14*
	12		12/1	12/2	12/3	12/4	12/5	12/6	12/7	12/8	12/9	12/10	12/11*	12/12*	12/13*	12/14*
	13		13/1	13/2	13/3	13/4	13/5	13/6	13/7	13/8	13/9	13/10	13/11*	13/12*	13/13*	13/14*
	14		14/1	14/2	14/3	14/4	14/5	14/6	14/7	14/8	14/9	14/10	14/11*	14/12*	14/13*	14/14*
	15	ZRL	15/1	15/2	15/3	15/4	15/5	15/6	15/7	15/8	15/9	15/10	15/11*	15/12*	15/13*	15/14*

*Defined only for 12 bit precision.

The codes from the Huffman table and the belonging VLI bits (and all following) are threaded to a bit-stream. The bit-stream is divided into bytes with a length of 8 bit. If a byte contain only 1s (FFh), then a zero byte is inserted behind, to avoid the misinterpretation with a marker. After the very last block, the remaining bits are filled up with ones, to have a complete byte.

2.3 SOF Start of Frame

The **Start Of Frame** segment defines the geometric parameters of the image

SOF ₁ marker	16 bit	xx, xx	FFh, C0h [*]
Segment length	16 bit	xx, xx	8 + num components * 3
Sample precision	8 bit	xx	08h ^{**}
Number of lines	16 bit	xx, xx	
Number of samples / line	16 bit	xx, xx	
Number of components	8 bit	xx	1 for mono chrome, 3 for color

For each component:

Component identifier	8 bit	xx	0 for Y, 1 for Cb, 2 for Cr
Horizontal sampling factor	4 bit	x	1 for luminance, 2 for chrominance
Vertical sampling factor	4 bit	x	1
Quantization table selector	8 bit	xx	0 for luminance, 1 for chrominance

^{*}C1h for 12 bit precision.

^{**}0Ch for 12 bit precision.

2.4 DHT Define Huffman Table

The **Define Huffman Table** segment defines the Huffman tables, used by the encoder.

DHT marker	16 bit	xx, xx	FFh, C4h
Segment length	16 bit	xx	2 + 276

For each table:

Table class	4 bit	x	0 for DC table, 1 for AC table
Table identifier	4 bit	x	0 for luminance, 1 for chrominance
Number of codes length	16 x 8 bit	xx,..., xx	
Codes	n x 8 bit	xx,..., xx	n = 16 for DC, 226 for AC

2.5 SOS Start of Scan

The **Start Of Scan** segment defines the scan parameter, followed by the compressed bit-stream.

SOS marker	16 bit	xx, xx	FFh, DAh
Segment length	16 bit	xx, xx	6 + Num Components * 2
Number of component	8 bit	xx	1 for mono chrome, 3 for color

For each component:

Component selector	8 bit	xx	0 for Y, 1 for Cb, 2 for Cr
DC table selector	4 bit	x	0 for Y, 1 for Cb and Cr
AC table selector	4 bit	x	0 for Y, 1 for Cb and Cr
Start of spectral selection	8 bit	xx	00
End of spectral selection	8 bit	xx	3Fh
Successive appr. bit pos high	4 bit	x	0
Successive appr. bit pos low	4 bit	x	0

Image bit-stream n x 8 bit xx, ..., xx

2.6 DNL Define Number of Lines

The **Define Number of Lines** segment redefines the image height. It is optional and can be used when the number of lines is not defined at image start. Unfortunately most of the JPEG viewer programs ignore this segment.

DNL marker	16 bit	xx, xx	FFh, DCh
Segment length	16 bit	00, 04	Fix length of 4
Number of lines	16 bit	xx, xx	1... 65535

2.7 EOI End Of Image

The **End Of Image** segment defines the end of an image and has no further parameter.

EOI marker	16 bit	xx, xx	FFh, D9h
------------	--------	--------	----------

3. Encoding Example

This example shows the way of an 8x8 block with sampling values to a JPEG compressed bit-stream.

Here an 8x8 block with the original luminance sampling, as unsigned values.

Figure 9: Block with Original Samples

139	144	149	153	155	155	155	155
144	151	153	156	159	156	156	156
150	155	160	163	158	156	156	156
159	161	162	160	160	159	159	159
159	160	161	162	162	155	155	155
161	161	161	161	160	157	157	157
162	162	161	163	162	157	157	157
162	162	161	161	163	158	158	158

Now, the sampling values are transformed from the time domain to the frequency domain by a discrete cosine transformation.

Figure 11: Coefficients After DCT

236	-1	-12	-5	2	-2	-3	1
-23	-18	-6	-3	-3	0	0	-1
-11	-9	-2	2	0	-1	-1	0
-7	-2	0	2	1	0	0	0
-1	-1	2	2	0	-1	1	1
2	0	2	0	-1	2	1	-1
-1	0	0	-2	-1	2	1	-1
-3	2	-4	-2	2	1	-1	0

After Quantization, most of the higher frequency coefficients are zero.

Figure 13: Quantized Coefficients

15	0	-1	0	0	0	0	0
-2	-1	0	0	0	0	0	0
-1	-1	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

As first, the unsigned sampling values are level shifted to signed values, by subtracting 128:

Figure 10: Block with Level Shifted Samples

11	16	21	25	27	27	27	27
16	23	25	28	31	28	28	28
22	27	32	35	30	28	28	28
31	33	34	32	32	31	31	31
31	32	33	34	34	27	27	27
33	33	33	33	32	29	29	29
34	34	33	35	34	29	29	29
34	34	33	33	35	30	30	30

For quantization, the default luminance table is used.

Figure 12: Quantization Table

16	11	10	16	24	40	51	61
12	12	14	19	26	58	60	55
14	13	16	24	40	57	69	56
14	17	22	29	51	87	80	62
18	22	37	56	68	109	103	77
24	35	55	64	81	104	113	92
49	64	78	87	103	121	120	101
72	92	95	98	112	100	103	99

The quantized values are rearranged in Zigzag order.

Figure 14: Values in Zigzag Order

15, 0, -2, -1, -1, -1, 0, 0, -1, 0 ... 0

Now, the values are transformed in Run/Size symbols and variable length integers (VLI). For the DC coefficient we assumed, that the previous DC coefficient was 12. The codes for the symbols are taken from the default luminance Huffman table.

Figure 15: Huffman Encoding

DC-Value	Difference	(Size),VLI	Code,VLI
15	15 - 12 = 3	(2), 11	011 11
AC-Values		(Run, Size),VLI	Code,VLI
0,-2		(1, 2), 01	11011, 01
-1		(0, 1), 0	00, 0
-1		(0, 1), 0	00, 0
-1		(0, 1), 0	00, 0
0, 0, -1		(2, 1), 0	11100, 0
0 ... 0		EOB	1010

The code and VLI bits are threaded to a finally bit-stream of 31 bit, where represents the 64 byte block.

01111 1101101 000 000 000 111000 1010

The compression rate for this example is $(64 * 8 \text{ bit} / 31 \text{ bit}) 16.5$.

4. The JE4XX JPEG Encoder

4.1 Technical Features

- Parallel compression of 4 unsynchronized images
- Marker generation included
- JPEG file output
- Baseline encoder
- Compliant with Baseline ISO/IEC 10918-1
- Block building RAM included, no external RAM needed
- Monochrome or color (YCbCr 4:2:2)
- Up to 2048 samples per line
- Line by line sample input
- Motion-JPEG capability
- 8-bit/sample
- 2 Quantization tables
- 4 fixed Huffman tables (2 DC and 2 AC)
- Predefined luminance and chrominance tables
- Fully synchronous design
- Fully stall able design
- Simple CPU interface for Quantization table reprogramming
- Different clocks for encoder and CPU interface
- Single clock cycle per sample encoding
- No pause cycles between blocks

4.2 Differences Between the JE430, JE440 and JE490 Cores

- The JE430 is a netlist version for Xilinx Spartan-3 and Virtex 2 to 4 FPGAs
- The JE440 is a netlist version for Altera Stratix and Stratix-II FPGAs
- The JE490 is a VHDL source code version for all FPGA vendors; it was tested with Xilinx and Altera FPGAs.

4.3 Needed Resources

Table 3: Needed Resources for JE430

Device	Samples per Row	Slice	Flip-Flops	LUTs	Block RAMs	DSP Elements	IOBs	Performance
XC3S2000-4	2048*	4650	4550	7350	39	15	221	25 MHz+
XC3S2000-5	2048*	4650	4550	7350	39	15	221	30 MHz+
XC4VSX35-10	2048*	4500	4450	7300	39	15	221	39 MHz+
XC4VSX35-12	2048*	4500	4450	7300	39	15	221	53 MHz+

* Contact us for more samples per row.

+ Maximal sample clock, the encoder clock is four times higher.

Table 4: Needed Resources for JE440

Device	Samples per Row	Logic Elements	M512 RAMs	M4K RAMs	M RAMs	DSP Elements	IOBs	Performance
EP1S25-C8	2048*	6800	20	32	1	24	221	27 MHz+
EP1S25-C5	2048*	6800	20	32	1	24	221	39 MHz+
EP2S30-C5	2048*	6700	20	32	1	24	221	51 MHz+
EP2S30-C3	2048*	6700	20	32	1	24	221	62 MHz+

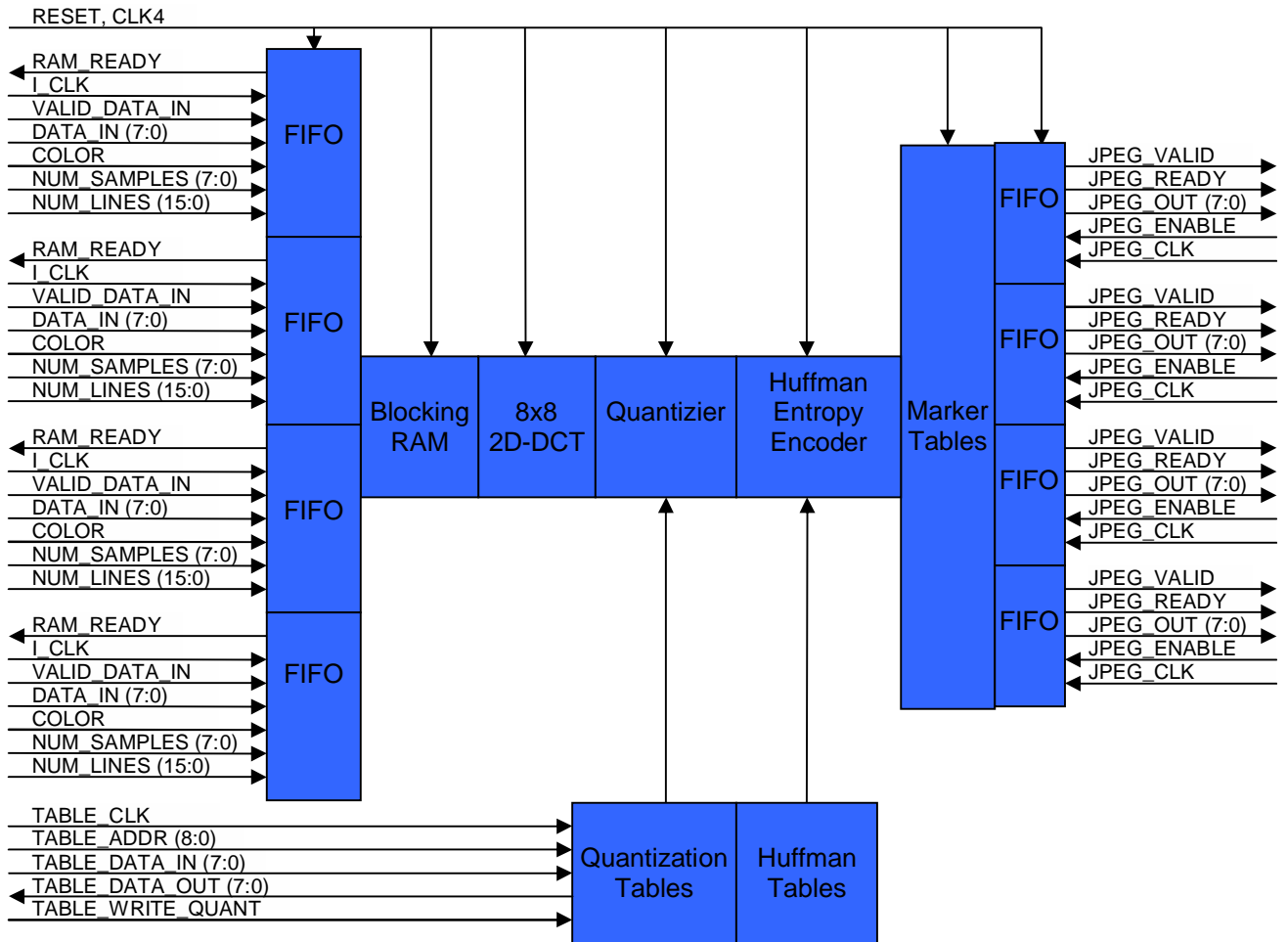
* Contact us for more samples per row.

+ Maximal sample clock, the encoder clock is four times higher.

4.4 Functional Description

The JE4XX is a stand alone image compressor using the JPEG “baseline system”. The core is a full synchronous design and has the capability to be stalled from the samples input and from the compressed image output side.

Figure 16: Encoder Structure



An additional interface, the “Table Programming Interface”, is used to change the Quantization tables. The “Table Programming Interface” has an own clock to minimize the glue logic for connecting the core to a controller. Usually there is no need to change the tables, only if the compression rate is to change.

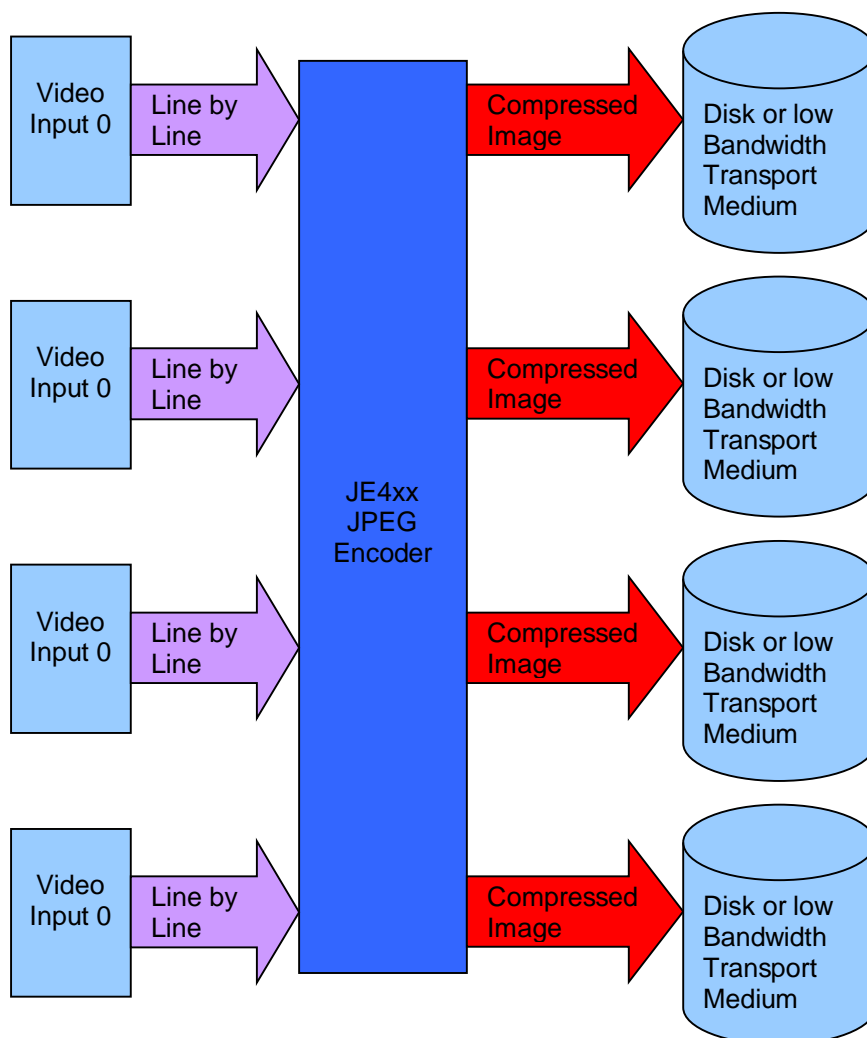
Each incoming line of samples is stored in the internal RAM. When 8 lines are written, the samples are read out in 8x8 blocks. These blocks go through the two-dimensional discrete cosine transformation (2D-DCT) and will be transformed into the 64 coefficients of the frequency domain. Now, the coefficients will be read out in a Zigzag order and quantized by dividing through the selected Quantization table. The first (DC) quantized coefficient is differentially coded, using the most recently DC coefficient from the same component. Now all coefficients are run-length coded to Run/Size symbols (the DC coefficient only to a size symbol). The symbols are Huffman coded using the code from the selected Huffman table. The resulting bit-

stream of Huffman codes is divided into parts with 8 bit and embedded between all needed markers. Finally, the JPEG file is stored in the output register.

4.5 Typical Application

In a typical application, first the video signal is digitized (if analog). The digitized pixels are written directly into the encoder. There is no need to reordering the pixel in 8x8 blocks, no external RAM is needed. The encoder outputs the bit-stream of the compressed image with all markers. The bit-stream can be transmitted over a system with limited bandwidth like a network or USB bus.

Figure 17: Block Diagram of a Typical Application



4.6 JE4XX Core Entity

This is the core entity like defined in the VHDL source code:

```

entity JE4XX is port (
    CLK4          : in  std_logic;          -- Main clock for the encoder
    RESET        : in  std_logic;          -- Reset JPEG logic

    -- Samples Input 0
    I_CLK_0      : in  std_logic;          -- Samples input clock
    VALID_DATA_IN_0 : in  std_logic;      -- Sample data is valid
    DATA_IN_0   : in  std_logic_vector ( 7 downto 0); -- Sample data
    NUM_SAMPLES_0 : in  std_logic_vector (12 downto 0); -- Number of samples in a line
    NUM_LINES_0  : in  std_logic_vector (15 downto 0); -- Number of lines in the image
    COLOR_0      : in  std_logic;          -- Color mode (0 = mono)
    RAM_READY_0  : out std_logic;          -- Encoder is ready for new sample

    -- Samples Input 1
    I_CLK_1      : in  std_logic;          -- Samples input clock
    VALID_DATA_IN_1 : in  std_logic;      -- Sample data is valid
    DATA_IN_1    : in  std_logic_vector ( 7 downto 0); -- Sample data
    NUM_SAMPLES_1 : in  std_logic_vector (12 downto 0); -- Number of samples in a line
    NUM_LINES_1  : in  std_logic_vector (15 downto 0); -- Number of lines in the image
    COLOR_1      : in  std_logic;          -- Color mode (0 = mono)
    RAM_READY_1  : out std_logic;          -- Encoder is ready for new sample

    -- Samples Input 2
    I_CLK_2      : in  std_logic;          -- Samples input clock
    VALID_DATA_IN_2 : in  std_logic;      -- Sample data is valid
    DATA_IN_2    : in  std_logic_vector ( 7 downto 0); -- Sample data
    NUM_SAMPLES_2 : in  std_logic_vector (12 downto 0); -- Number of samples in a line
    NUM_LINES_2  : in  std_logic_vector (15 downto 0); -- Number of lines in the image
    COLOR_2      : in  std_logic;          -- Color mode (0 = mono)
    RAM_READY_2  : out std_logic;          -- Encoder is ready for new sample

    -- Samples Input 3
    I_CLK_3      : in  std_logic;          -- Samples input clock
    VALID_DATA_IN_3 : in  std_logic;      -- Sample data is valid
    DATA_IN_3    : in  std_logic_vector ( 7 downto 0); -- Sample data
    NUM_SAMPLES_3 : in  std_logic_vector (12 downto 0); -- Number of samples in a line
    NUM_LINES_3  : in  std_logic_vector (15 downto 0); -- Number of lines in the image
    COLOR_3      : in  std_logic;          -- Color mode (0 = mono)
    RAM_READY_3  : out std_logic;          -- Encoder is ready for new sample

    -- Compressed data Output 0
    JPEG_CLK_0    : in  std_logic;          -- Samples input clock
    JPEG_ENABLE_0 : in  std_logic;          -- Application read the JPEG byte
    JPEG_OUT_0    : out std_logic_vector ( 7 downto 0); -- JPEG stream byte output
    JPEG_VALID_0  : out std_logic;          -- JPEG stream byte is valid
    JPEG_READY_0  : out std_logic;          -- JPEG stream is finished

    -- Compressed data Output 1
    JPEG_CLK_1    : in  std_logic;          -- Samples input clock
    JPEG_ENABLE_1 : in  std_logic;          -- Application read the JPEG byte
    JPEG_OUT_1    : out std_logic_vector ( 7 downto 0); -- JPEG stream byte output
    JPEG_VALID_1  : out std_logic;          -- JPEG stream byte is valid
    JPEG_READY_1  : out std_logic;          -- JPEG stream is finished

    -- Compressed data Output 2
    JPEG_CLK_2    : in  std_logic;          -- Samples input clock
    JPEG_ENABLE_2 : in  std_logic;          -- Application read the JPEG byte
    JPEG_OUT_2    : out std_logic_vector ( 7 downto 0); -- JPEG stream byte output
    JPEG_VALID_2  : out std_logic;          -- JPEG stream byte is valid
    JPEG_READY_2  : out std_logic;          -- JPEG stream is finished

    -- Compressed data Output 3
    JPEG_CLK_3    : in  std_logic;          -- Samples input clock
    JPEG_ENABLE_3 : in  std_logic;          -- Application read the JPEG byte
    JPEG_OUT_3    : out std_logic_vector ( 7 downto 0); -- JPEG stream byte output
    JPEG_VALID_3  : out std_logic;          -- JPEG stream byte is valid
    JPEG_READY_3  : out std_logic;          -- JPEG stream is finished

```

```
-- Tables Programming
TABLE_CLK      : in  std_logic;           -- Clock for table reprogramming
TABLE_WRITE_QUANT : in  std_logic;       -- Write value in Quantization tab
TABLE_ADDR     : in  std_logic_vector (8 downto 0); -- Table address
TABLE_DATA_IN  : in  std_logic_vector (7 downto 0); -- Table write data
TABLE_DATA_OUT : out std_logic_vector (7 downto 0) -- Table read data
);
end JE4XX;
```

4.7 Core Interface

The Core interface is subdivided into three parts: the “Sample Input Interface”, the “Compressed Data Output Interface” and the “Tables Programming Interface”. The “Tables Programming Interface” is only used when the tables are reprogrammed with custom values.

Table 5: Core Entity Signals

Signal Name	Direction	Description
Control interface		
RESET	In	Encoder reset
CLK4	In	Encoder clock
Sample input interface, four times for each channel		
I_CLK_x	In	Sample input clock
VALID_DATA_IN_x	In	Sample is valid
DATA_IN_x (7 : 0)	In	Sample input
COLOR_x	In	Mode, mono chrome or color
NUM_SAMPLES_x (7:0)	In	Number of samples per line in 8 units
NUM_LINES_x (15:0)	In	Number of lines per image
RAM_READY_x	Out	Encoder is ready for new sample
Compressed data output interface, four times for each channel		
JPEG_CLK_x	In	Clock for JPEG file output
JPEG_ENABLE_x	In	Data handshake
JPEG_VALID_x	Out	Output data is valid
JPEG_READY_x	Out	Last data for this field
JPEG_OUT_x (7:0)	Out	Output data
Tables programming Interface		
TABLE_CLK	In	Clock for table reprogramming
TABLE_ADDR (8:0)	In	Table select and addressing
TABLE_DATA_OUT (7:0)	Out	Table read data
TABLE_DATA_IN (7:0)	In	Table write data
TABLE_WRITE_QUANT	In	Write strobe for Quantization table

4.7.1 Control Interface

CLK4

```
CLK4           : in std_logic;           -- Main clock for the encoder
```

This is the main clock for the encoder. The frequency should be at least four times the highest “I_CLK_x”.

RESET

```
RESET         : in std_logic;           -- Reset JPEG logic
```

Synchronic reset of the encoder core, but Quantization tables and Huffman tables are not affected. After power up, or if an image compression is canceled, assert this signal for at least three cycles of the slowest clock.

4.7.2 Sample Input Interface

Each signal is four times there, for each input channel once.

I_CLK

```
I_CLK_0       : in std_logic;           -- Clock for samples input
I_CLK_1       : in std_logic;           -- Clock for samples input
I_CLK_2       : in std_logic;           -- Clock for samples input
I_CLK_3       : in std_logic;           -- Clock for samples input
```

This is the clock for pushing samples into the encoder. Input signals must be valid at the rising edge, output signals are updated with the rising edge, too.

COLOR

```
COLOR_0       : in std_logic;           -- Color mode (0 = mono)
COLOR_1       : in std_logic;           -- Color mode (0 = mono)
COLOR_2       : in std_logic;           -- Color mode (0 = mono)
COLOR_3       : in std_logic;           -- Color mode (0 = mono)
```

This signal defines the kind of the image: a 0 means a monochrome image, a 1 a colored image with chrominance sub sampling (YCbCr 4:2:2). This signal must not change while an image is compressed. When only monochrome images are compressed, assign a static 0 to allow the mapper removing the unused logic.

VALID_DATA_IN

```
VALID_DATA_IN_0 : in std_logic;        -- Sample data is valid
VALID_DATA_IN_1 : in std_logic;        -- Sample data is valid
VALID_DATA_IN_2 : in std_logic;        -- Sample data is valid
VALID_DATA_IN_3 : in std_logic;        -- Sample data is valid
```

This is the sample qualifier. Assert this signal, when a new sample is valid. Sample data will be accepted, when “RAM_READY_x” is asserted, too.

DATA_IN

```
DATA_IN_0      : in std_logic_vector (7 downto 0); -- Sample data
DATA_IN_1      : in std_logic_vector (7 downto 0); -- Sample data
DATA_IN_2      : in std_logic_vector (7 downto 0); -- Sample data
DATA_IN_3      : in std_logic_vector (7 downto 0); -- Sample data
```

This is the sample input port. The format is 8 bit signed with values from -128 to 127. Samples order is from the left to the right of the top line until the bottom line. The data will be accepted, when “VALID_DATA_IN_x” and “RAM_READY_x” are asserted.

NUM_SAMPLES

```
NUM_SAMPLES_0  : in std_logic_vector (7 downto 0); -- Number of samples in a line
NUM_SAMPLES_1  : in std_logic_vector (7 downto 0); -- Number of samples in a line
NUM_SAMPLES_2  : in std_logic_vector (7 downto 0); -- Number of samples in a line
NUM_SAMPLES_3  : in std_logic_vector (7 downto 0); -- Number of samples in a line
```

This signal determinates the number of samples in a line. Because the value must be a multiple of 8, the lower three bits are truncated (divide by 8).

NUM_LINES

```
NUM_LINES_0    : in std_logic_vector (15 downto 0); -- Number of lines in the image
NUM_LINES_1    : in std_logic_vector (15 downto 0); -- Number of lines in the image
NUM_LINES_2    : in std_logic_vector (15 downto 0); -- Number of lines in the image
NUM_LINES_3    : in std_logic_vector (15 downto 0); -- Number of lines in the image
```

This signal determinates the number of lines in the image and is used for the internal line counter and for the header with the markers. The value must be a multiple of 8, the maximal value is 65528.

RAM_READY

```
RAM_READY_0    : out std_logic; -- RAM is ready for new sample
RAM_READY_1    : out std_logic; -- RAM is ready for new sample
RAM_READY_2    : out std_logic; -- RAM is ready for new sample
RAM_READY_3    : out std_logic; -- RAM is ready for new sample
```

This signal is asserted, when the core is ready to receive new data. When “JPEG_ENABLE_x” stays always at 1, then usually this signal will not be deasserted. A stalling of any channel will stall all four channels!

4.7.3 Compressed Data Output Interface

Each signal is four times there, for each input channel once.

JPEG_CLK

```
JPEG_CLK_0      : in std_logic;          -- Clock for the JPEG file output
JPEG_CLK_1      : in std_logic;          -- Clock for the JPEG file output
JPEG_CLK_2      : in std_logic;          -- Clock for the JPEG file output
JPEG_CLK_3      : in std_logic;          -- Clock for the JPEG file output
```

This is the clock for JPEG file output. Input signals must be valid at the rising edge, output signals are updated with the rising edge, too. The frequency should be the same, like the “I_CLK_x”.

JPEG_ENABLE

```
JPEG_ENABLE_0   : in std_logic;          -- Application read the JPEG byte
JPEG_ENABLE_1   : in std_logic;          -- Application read the JPEG byte
JPEG_ENABLE_2   : in std_logic;          -- Application read the JPEG byte
JPEG_ENABLE_3   : in std_logic;          -- Application read the JPEG byte
```

Assert this signal, when the application is ready to get the data byte. A valid data byte stays on the output until this signal is asserted. If not asserted while “JPEG_VALID_x” is asserted, the output interface will be stalled, but the remaining encoder still works until the output FIFO (64 items) is filled up. After this time, all four channels are stalled!

JPEG_OUT

```
JPEG_OUT_0      : out std_logic_vector (7 downto 0); -- JPEG stream byte output
JPEG_OUT_1      : out std_logic_vector (7 downto 0); -- JPEG stream byte output
JPEG_OUT_2      : out std_logic_vector (7 downto 0); -- JPEG stream byte output
JPEG_OUT_3      : out std_logic_vector (7 downto 0); -- JPEG stream byte output
```

Compressed data output stream with a size of 8 bit. Data is valid, when the signal “JPEG_VALID_x” is asserted and stays unchanged until “JPEG_ENABLE_x” is asserted. The data stream includes all the markers that be needed to have a valid JPEG file.

JPEG_VALID

```
JPEG_VALID_0    : out std_logic;          -- JPEG stream byte is valid
JPEG_VALID_1    : out std_logic;          -- JPEG stream byte is valid
JPEG_VALID_2    : out std_logic;          -- JPEG stream byte is valid
JPEG_VALID_3    : out std_logic;          -- JPEG stream byte is valid
```

This is the output data qualifier. When asserted, the output data is valid and stay valid until “JPEG_ENABLE_x” is asserted. If “JPEG_ENABLE_x” is already asserted, the output data is valid only for one clock cycle.

JPEG_READY

```

JPEG_READY_0      : out std_logic;           -- JPEG stream is finished
JPEG_READY_1      : out std_logic;           -- JPEG stream is finished
JPEG_READY_2      : out std_logic;           -- JPEG stream is finished
JPEG_READY_3      : out std_logic;           -- JPEG stream is finished

```

This signal will be asserted, when the last byte from the last block of the entire image is valid. This means, the image is completely compressed.

4.7.4 Tables Programming Interface

TABLE_CLK

```

TABLE_CLK          : in std_logic;           -- Clock for table reprogramming

```

This is the clock for the Tables Programming Interface. Input signals must be valid at the rising edge, output signals are updated with the rising edge, too.

TABLE_WRITE_QUANT

```

TABLE_WRITE_QUANT : in std_logic;           -- Write value in Quantization tab

```

Write enable for the Quantization tables. Assert this signal for one clock cycle. Address and data must be valid when asserted. In color mode, both tables must be written.

TABLE_ADDR

```

TABLE_ADDR         : in std_logic_vector (8 downto 0); -- Table address

```

These are the address lines for the access to the Quantization tables.

A5 ... A0 selects one of the 64 elements from the table.

A6 select between the Luminance and the Chrominance table (0=Luminance).

A8 ... A7 selects one of the four channels.

For more information about the tables contents, see chapter “Quantization Tables”.

TABLE_DATA_IN

```

TABLE_DATA_IN      : in std_logic_vector (7 downto 0); -- Table write data

```

Input data bus for writing into the Quantization tables.

For more information about the tables contents, see chapter “Quantization Tables”.

TABLE_DATA_OUT

```
TABLE_DATA_OUT      : out std_logic_vector (7 downto 0); -- Table read data
```

Output data bus with the Quantization tables read data. Data changes one clock after TABLE_ADDR is changed. There is no read enable signal.

For more information about the tables contents, see chapter “Quantization Tables”.

5. Quantization Tables

The Quantization table contains the divisors for the coefficient quantization. There are two tables implemented, predefined after power-up. The RESET signal doesn't affect the table's contents.

The address lines "TABLE_ADDR" are used to select a value in one of the four tables:

A8.. A7	A6	A5 – A0
Channel	Table	Zigzag Index

Because the values in the table are in Zigzag order, a table is needed to convert the indices from a Quantization table in coefficient order. The following table can be used to do this.

Table 6: Zigzag Order

	x0	x1	x2	x3	x4	x5	x6	x7	x8	x9	xA	xB	xC	xD	xE	xF
0x	0	1	8	16	9	2	3	10	17	24	32	25	18	11	04	05
1x	12	19	26	33	40	48	41	34	27	20	13	06	07	14	21	28
2x	35	42	49	56	57	50	43	36	29	22	15	23	30	37	44	51
3x	58	59	52	45	38	31	39	46	53	60	61	54	47	55	62	63

To fill the JE4XX Quantization table from a Divisor table in coefficient order, use the following C code:

```
int Zigzag_Table [64] = {
    0, 1, 8,16, 9, 2, 3,10,
    17,24,32,25,18,11,04,05,
    12,19,26,33,40,48,41,34,
    27,20,13,06,07,14,21,28,
    35,42,49,56,57,50,43,36,
    29,22,15,23,30,37,44,51,
    58,59,52,45,38,31,39,46,
    53,60,61,54,47,55,62,63
};

for (i = 0; i < 64; i++) {
    Quantization_Table [i] = Divisor_Table [ Zigzag_Table [i] ];
}
```

Each table entry has a size of 8 bit and contains the unsigned divisor for quantization. Allowed values are 1 to 255.

D7 – D0
Divisor

5.1 Default Quantization Table for Luminance

This Quantization table is predefined in the core as index 0.
The table is shown in the coefficient order.

Table 7: Default Quantization Table for Luminance

	x0	x1	x2	x3	x4	x5	x6	x7
0x	16	11	10	16	24	40	51	61
1x	12	12	14	19	26	58	60	55
2x	14	13	16	24	40	57	69	56
3x	14	17	22	29	51	87	80	62
4x	18	22	37	56	68	109	103	77
5x	24	35	55	64	81	104	113	92
6x	49	64	78	87	103	121	120	101
7x	72	92	95	98	112	100	103	99

5.2 Default Quantization Table for Chrominance

This Quantization table is predefined in the core as index 1.
The table is shown in the coefficient order.

Table 8: Default Quantization Table for Chrominance

	x0	x1	x2	x3	x4	x5	x6	x7
0x	17	18	24	47	99	99	99	99
1x	18	21	26	66	99	99	99	99
2x	24	26	56	99	99	99	99	99
3x	47	66	99	99	99	99	99	99
4x	99	99	99	99	99	99	99	99
5x	99	99	99	99	99	99	99	99
6x	99	99	99	99	99	99	99	99
7x	99	99	99	99	99	99	99	99

5.3 Changing the Compression Rate

To change the compression rate, change the values of the Quantization table. There is no rule how to do this, but usually the default values are scaled. A Quality parameter (Q) is used to determinate the scaling. A low value results in a low quality image with a high compression rate and a high value results in a high quality image with a low compression rate. Allowed values are 1 to 100; see at our web pages for sample images with various quality values.

Use the following formula to calculate a scaled Quantization table:

Quality : Q (1...100)
 Scaling factor : S
 Item from the default table : XD_i
 Item from the scaled table : XS_i

$S = 50/Q$ for $Q < 50$

$S = 2-Q/50$ for $Q \geq 50$

$XS_i = XD_i * S$

The following C code illustrates the scaling of the Quantization table:

```
int    Q = 75;
double S;
int    i;
int    NewValue;

if (Q <= 50) {
    S = 50.0 / Q;
} else {
    S = 2.0 - Q / 50.0;
}

for (i = 0; i < 64; i++) {
    NewValue = (int)(Default_Quantization_Table[i] * S);
    if (NewValue < 1) {
        NewValue = 1;
    } else {
        if (NewValue > 255) {
            NewValue = 255;
        }
    }
    Scaled_Quantization_Table[i] = (BYTE)NewValue;
}
```

6. Huffman Tables

The Huffman table contains the Huffman code words for the DC and AC coefficients, there are different codes for the DC and the AC coefficients.

6.1 Default Huffman Table for Luminance

This table shows the number of codes for the luminance DC table:

Table 9: Luminance Number of DC Codes

Length	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Number	0h	1h	5h	1h	1h	1h	1h	1h	1h	0h	0h	0h	0h	0h	0h	0h

This table shows the assignment of generated codes to the Size symbols for the luminance DC table:

Table 10: Luminance DC Symbol to Code Assignment

Code	0	1	2	3	4	5	6	7	8	9	10	11
Symbol	0h	1h	2h	3h	4h	5h	6h	7h	8h	9h	Ah	Bh

This table shows the number of codes for the luminance AC table:

Table 11: Luminance Number of AC Codes

Length	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Number	0h	2h	1h	3h	3h	2h	4h	3h	5h	5h	4h	4h	0h	0h	1h	7Dh

This table shows the assignment of generated codes to the Run/Size symbols for the luminance AC table:

Table 12: Luminance AC Symbol to Code Assignment

	x0	x1	X2	x3	x4	x5	x6	X7	x8	x9	xA	xB	xC	xD	xE	xF
0x	01h	02h	03h	00h	04h	11h	05h	12h	21h	31h	41h	06h	13h	51h	61h	07h
1x	22h	71h	14h	32h	81h	91h	A1h	08h	23h	42h	B1h	C1h	15h	52h	D1h	F0h
2x	24h	33h	62h	72h	82h	09h	0Ah	16h	17h	18h	19h	1Ah	25h	26h	27h	28h
3x	29h	2Ah	34h	35h	36h	37h	38h	39h	3Ah	43h	44h	45h	46h	47h	48h	49h
4x	4Ah	53h	54h	55h	56h	57h	58h	59h	5Ah	63h	64h	65h	66h	67h	68h	69h
5x	6Ah	73h	74h	75h	76h	77h	78h	79h	7Ah	83h	84h	85h	86h	87h	88h	89h
6x	8Ah	92h	93h	94h	95h	96h	97h	98h	99h	9Ah	A2h	A3h	A4h	A5h	A6h	A7h
7x	A8h	A9h	AAh	B2h	B3h	B4h	B5h	B6h	B7h	B8h	B9h	BAh	C2h	C3h	C4h	C5h
8x	C6h	C7h	C8h	C9h	CAh	D2h	D3h	D4h	D5h	D6h	D7h	D8h	D9h	DAh	E1h	E2h
9x	E3h	E4h	E5h	E6h	E7h	E8h	E9h	EAh	F1h	F2h	F3h	F4h	F5h	F6h	F7h	F8h
Ax	F9h	FAh														

6.2 Default Huffman Table for Chrominance

This table shows the number of codes for the chrominance DC table:

Table 13: Chrominance Number of DC Codes

Length	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Number	0h	3h	1h	1h	1h	1h	1h	1h	1h	1h	1h	0h	0h	0h	0h	0h

This table shows the assignment of generated code to the Size symbol for the chrominance DC table:

Table 14: Chrominance DC Symbol to Code Assignment

Code	0	1	2	3	4	5	6	7	8	9	10	11
Symbol	0h	1h	2h	3h	4h	5h	6h	7h	8h	9h	Ah	Bh

This table shows the number of codes for the chrominance AC table:

Table 15: Chrominance Number of AC Codes

Length	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Number	0h	2h	1h	2h	4h	4h	3h	4h	7h	5h	4h	4h	0h	1h	2h	77h

This table shows the assignment of generated code to the Run/Size symbol for the chrominance AC table:

Table 16: Chrominance AC Symbol to Code Assignment

	x0	x1	x2	x3	x4	x5	x6	x7	x8	x9	xA	xB	xC	xD	xE	xF
0x	00h	01h	02h	03h	11h	04h	05h	21h	31h	06h	12h	41h	51h	07h	61h	71h
1x	13h	22h	32h	81h	08h	14h	42h	91h	A1h	B1h	C1h	09h	23h	33h	52h	F0h
2x	15h	62h	72h	D1h	0Ah	16h	24h	34h	E1h	25h	F1h	17h	18h	19h	1Ah	26h
3x	27h	28h	29h	2Ah	35h	36h	37h	38h	39h	3Ah	43h	44h	45h	46h	47h	48h
4x	49h	4Ah	53h	54h	55h	56h	57h	58h	59h	5Ah	63h	64h	65h	66h	67h	68h
5x	69h	6Ah	73h	74h	75h	76h	77h	78h	79h	7Ah	82h	83h	84h	85h	86h	87h
6x	88h	89h	8Ah	92h	93h	94h	95h	96h	97h	98h	99h	9Ah	A2h	A3h	A4h	A5h
7x	A6h	A7h	A8h	A9h	AAh	B2h	B3h	B4h	B5h	B6h	B7h	B8h	B9h	BAh	C2h	C3h
8x	C4h	C5h	C6h	C7h	C8h	C9h	CAh	D2h	D3h	D4h	D5h	D6h	D7h	D8h	D9h	DAh
9x	E2h	E3h	E4h	E5h	E6h	E7h	E8h	E9h	EAh	F2h	F3h	F4h	F5h	F6h	F7h	F8h
Ax	F9h	FAh														

7. Hints and Restrictions

7.1 Initialization

- After power up, RESET must be asserted for at least three cycles of the slowest clock.
- After power up or a change of the color mode, the Quantization tables must be written with new values.

7.2 Clocks

- Each channel has its own sample input and JPEG output clock.
- The encoder clock (CLK4) should be at least 4 times higher than the fastest input data rate. If the encoder clock is too slow, then the input interface will be stalled.
- The output data rate should be at least the input data rate. An output data rate that is too slowly, will stall the encoder for **all** channels.

7.3 Image Size

- In monochrome mode is one sample equal to one pixel.
- In color mode are two samples one pixel. The smallest unit is two pixels, which consist of two luminance- and two chrominance samples.
- The number of pixel in a line must be a multiple of 8 in monochrome mode and a multiple of 16 in color mode.
- The lowest number of pixels in a line is 64.
- The highest number of pixels in a line is 2048 in monochrome mode and 1024 in color mode.
- The lowest number of lines is 8.
- The highest number of lines is 65528.
- A change of the image width or the color mode can only be done, after the compression of the previous image is finished.
- The time duration between the first samples of two lines must be at least 6912 cycles of the encoder clock, or the image may be corrupted.

Example: When 1440 samples per line are transferred with each cycle of a 27 MHz I_CLK and the encoder clock is 108 MHz then after a line is completed a pause of 288 I_CLK cycles is needed.

$$(6912 / 108 \text{ MHz} - 1440 / 27 \text{ MHz}) * 27 \text{ MHz} = 288$$

This is usually no problem, because a typical video source has a 64 μs line duration with an active video time of 52 μs . The remaining 12 μs are enough to solve the requirements. If a video source with very low blanking time is used and the encoder clock can't be increased, an encoder core without this restriction can be delivered, but with a double count of needed RAM.

8. Literature and Links

8.1 Documents from the Internet

The JPEG Still Picture Compression Standard. Wallace GK (ed) (PDF document)

JPEG File Interchange Format. Hamilton E (ed) (PDF document)

8.2 Hard Book

JPEG STILL IMAGE DATA COMPRESSION STANDARD. Pennebaker WB, Mitchell JL (eds) Kluwer Academic Publishers Boston, Dordrecht, London, 1993

8.3 Internet Links

JPEG organization

<http://www.jpeg.org/>

NASA Vision Group Publications Menu

<http://vision.arc.nasa.gov/publications/publications.html#ImageCompression>

9. Revision Info

Rev.	Date	Changes
1.0	16.03.2006	Initial revision