

```

1  % Modelling aspects of US'841, US'659, Q15j28 / lcodec4.0
2  % Iain Richardson 1 August 24 (v2)
3  % Note on pixel array structures:
4  % pixin(1) (2) (3) (4) | (5) (6) (7) (8) % Matlab array
5  %      p3 p2 p1 p0 | q0 q1 q2 q3 % 841/659 Patent
6  %      a  b  c  d | e  f  g  h % Q15j28
7  %      li[0] [1] [2] [3] | [4] [5] [6] [7] % lcodec4.0
8
9  if ~exist('qp', 'var')
10     % Set a default QP if none has been set
11     qp = 12;
12 end
13 if ~exist('UseAltThreshold', 'var')
14     % Set a value if not yet defined
15     UseAltThreshold = 0;
16 end
17 if ~exist('Uselcodec4Diff', 'var')
18     % Set a value if not yet defined, v2
19     Uselcodec4Diff = 0;
20 end
21 if ~exist('FIDblocks', 'var')
22     % Set a value if not yet defined, v2
23     FIDblocks = 1; % Print to terminal output
24 end
25 if ~exist('FIDstats', 'var')
26     % Set a value if not yet defined, v2
27     FIDstats = 1; % Print to terminal output
28 end
29
30 % Blocks of sample values. Set up a gradient with random variations.
31 NoiseScale = 12; % Make this larger to add more variation in the blocks
32 pixin = zeros(8,8); % Two adjacent 8x8 blocks, considering just the RH four columns of
33 % block 1 and the LH four columns of block 2
34 fprintf(FIDblocks, 'Input blocks:\n');
35 startpix = 75+round(rand*50); % random value between 50 and 100
36 steppix = round((rand*NoiseScale)-(NoiseScale/2)); % stepping up or down
37 for i=1:8 % Fill the lines
38     pixin(i,1) = startpix + round((rand*(NoiseScale/2))-(NoiseScale/4));
39     startpix = pixin(i,1);
40     for j=2:8
41         pixin(i,j) = pixin(i,j-1) + steppix + round((rand*(NoiseScale/2))-(NoiseScale/4));
42     end
43     for j=1:4 % Print out the lines of the input blocks
44         fprintf(FIDblocks, '%3d ', pixin(i,j));
45     end
46     fprintf(FIDblocks, ' | ');
47     for j=5:8
48         fprintf(FIDblocks, '%3d ', pixin(i,j));
49     end
50     fprintf(FIDblocks, '\n');
51 end
52 %close all; surf(pixin); axis ij; shading flat; % Uncomment this line to plot the
53 % input blocks as a surface
54
55 % 1. Model Q15-J-28 method
56 Strength1 = 2; % Can range from 0 to 3
57 Strength2 = 2;
58 % Lim(QP) where QP=0 to 31, Q15j28 at p19
59 LimQParray = [7 8 9 10 11 12 14 15 17 20 22 25 28 31 35 39 44 49 55 62 69 78 88 98 110
60 124 139 156 175 197 221 248];
61
62 j28out = zeros(8,8); % Output block
63 fprintf(FIDstats, 'Q15J28 parameters, QP=%d, Strength1=%d, Strength2=%d:\n', qp,
64 Strength1, Strength2);

```

```

61 Diff_de = zeros(1,8); % Diff for pixels d and e
62 Diff_c = zeros(1,8); % Diff for pixel c
63 Diff_f = zeros(1,8); % Diff for pixel f
64 ModPixels1 = zeros(1,8); % Number of pixels with non-zero Diffs
65 Act11_block = zeros(1,8); % Record Act11 v2
66 Act22_block = zeros(1,8); % Record Act22 v2
67 ActEdge_block = zeros(1,8); % Record ActEdge v2
68
69 fprintf(FIDblocks, 'Q15J28 filtered blocks:\n');
70
71 for i=1:8 % For each line across both blocks
72     a = pixin(i,1); b = pixin(i,2); c = pixin(i,3); d = pixin(i,4); e = pixin(i,5); f =
        pixin(i,6); g = pixin(i,7); h = pixin(i,8);
73
74     % Calculate S parameters, Q15J28 4.5.2
75     S0 = abs(4*(c-f) + 12*(e-d)) + abs(9*(c-d-e+f));
76     S01 = 12*abs(b - (2*c) + d); % Assumed use of b, c, d instead of a, b, c as stated
        in Q15J28 at 19
77     S11 = abs(4*(a-d) + 12*(c-b)) + abs(9*(a-b-c+d));
78     S02 = 12*abs(e - (2*f) + g);
79     S12 = abs(4*(e-h)+12*(g-f)) + abs(9*(e-f-g+h));
80
81     % Find ActOverall, 4.5.3
82
83     ActOverall=0;
84     LimQP = LimQPararray(qp+1); % Lim(qp)
85     for ii=3:-1:0
86         val = LimQP*(2^ii);
87         if (S0<val)
88             ActOverall=3-ii;
89         end
90     end
91     if (Strength1==3) | (Strength2==3)
92         ActOverall = max(ActOverall, 1); % Clip to range 1,3 if either Strength is 3
93     end
94
95     % Find Act1, Act2
96     Act1 = 1;
97     if (S01 < LimQP)
98         Act1 = Act1 + 1;
99     end
100    if (S11 < LimQP)
101        Act1 = Act1 + 1;
102    end
103    Act11 = min(ActOverall, Act1);
104
105    Act2 = 1;
106    if (S02 < LimQP)
107        Act2 = Act2 + 1;
108    end
109    if (S12 < LimQP)
110        Act2 = Act2 + 1;
111    end
112    Act22 = min(ActOverall, Act2);
113    Act11block(i) = Act11; % v2
114    Act22block(i) = Act22; % v2
115
116    %Modified d, e, Q15J28 section 4.5.5
117
118    ActEdge = round((Act11 + Act22)/2);
119    ActEdgeblock(i) = ActEdge; % v2
120
121    ClipEdge = round((ClipQP(qp, Strength1) + ClipQP(qp, Strength2) + Act1 + Act2 -2) / 2
        );
122    Diff = 0;

```

```

123
124     if (ActEdge == 1) || (ActEdge==2)
125         Diff = round((3*(c-f) - 8*(d-e))/16);
126     end
127     if (ActEdge == 3)
128         Diff = round((b + c - (8*(d-e)) -f -g)/16);
129     end
130     Diff = max(-ClipEdge, min(ClipEdge, Diff));
131
132     dresult = Clip8(d+Diff);
133     dresult0 = d+Diff; % lcodec4.0 image.c at 485, no clipping
134     ertest = Clip8(e-Diff);
135     ertest0 = e-Diff; % lcodec4.0 image.c at 486, no clipping
136
137     Diff_de(i) = Diff;
138
139     % Modified c, Q15J28 4.5.6.1 / lcodec4.0 image.c at 489-495
140     Diff = 0;
141     if (UseIcodec4Diff==0) % Use Q15J28 calculation
142         if (Act11 == 2)
143             Diff = round((5*(b-(2*c)+d))/16); % Q15J28 equation at 4.5.6.1
144         end
145         if (Act11 == 3)
146             Diff = round((3*(a+b-(4*c)+d+e))/16);
147         end
148     else % Use 'lcodec4.0 calculation', Diff equation based on dresult0, v2
149         if (Act11 == 2)
150             Diff = round((5*(b-(2*c)+dresult0))/16); % lcodec4.0 image.c at 491
151         end
152         if (Act11 == 3)
153             Diff = round((3*(a+b-(4*c)+dresult0+ertest0))/16); % lcodec4.0 image.c at
154             493
155         end
156     end
157     Diff = max(-ClipQP(qp, Strength1), min(ClipQP(qp, Strength1), Diff));
158     cresult = Clip8(c+Diff);
159     Diff_c(i) = Diff;
160
161     % Modified f, Q15J28 4.5.6.2 / lcodec4.0 image.c at 497-503
162     Diff = 0;
163     if (UseIcodec4Diff==0) % Use Q15J28 calculation
164         if (Act22 == 2)
165             Diff = round((5*(e-(2*f)+g))/16);
166         end
167         if (Act22 == 3)
168             Diff = round((3*(d+e-(4*f)+g+h))/16);
169         end
170     else % Use lcodec4.0 calculation, Diff equation based on ertest0, v2
171         if (Act22 == 2)
172             Diff = round((5*(ertest0-(2*f)+g))/16); % lcodec4.0 image.c at 499
173         end
174         if (Act22 == 3)
175             Diff = round((3*(dresult0+ertest0-(4*f)+g+h))/16); % lcodec4.0 image.c at
176             501
177         end
178     end
179     Diff = max(-ClipQP(qp, Strength2), min(ClipQP(qp, Strength2), Diff));
180     fresult = Clip8(f+Diff);
181     Diff_f(i) = Diff;
182
183     % Print out a line of filtered pixels
184     j28out(i,:) = [a b cresult dresult ertest fresult g h];
185     for j=1:4
186         fprintf(FIDblocks, '%3d ', j28out(i,j));
187     end

```

```

186     fprintf(FIDblocks, ' | ');
187     for j=5:8
188         fprintf(FIDblocks, '%3d ', j28out(i,j));
189     end
190     fprintf(FIDblocks, '\n');
191     ModPixels1(i) = 0; % Total number of pixels with non-zero Diff
192     if not(isequal(d, dresult))
193         ModPixels1(i) = ModPixels1(i) + 2; % If d has non-zero Diff then e has non-zero
194         Diff, 4.5.5
195     end
196     if not(isequal(c, cresult))
197         ModPixels1(i) = ModPixels1(i) + 1;
198     end
199     if not(isequal(f, fresult))
200         ModPixels1(i) = ModPixels1(i) + 1;
201     end
202     % Log parameters for every row of pixels: v2
203     fprintf(FIDstats, 'Row:%d S0:%d S01:%d S11:%d S02:%d S12:%d ActOverall:%d Act1:%d
204     Act2:%d Act11:%d Act22:%d ActEdge:%d ClipEdge:%d\n', i, S0, S01, S11, S02, S12,
205     ActOverall, Act1, Act2, Act11, Act22, ActEdge, ClipEdge);
206 end % i loop
207
208 fprintf(FIDstats, '# Pixels with non-zero Diff :%2d %2d %2d %2d %2d %2d %2d %2d',
209 ModPixels1);
210 fprintf(FIDstats, '\nDiff_de:           %2d %2d %2d %2d %2d %2d %2d %2d', Diff_de);
211 fprintf(FIDstats, '\nDiff_c:           %2d %2d %2d %2d %2d %2d %2d %2d', Diff_c);
212 fprintf(FIDstats, '\nDiff_f:           %2d %2d %2d %2d %2d %2d %2d %2d', Diff_f);
213
214 % 2. Model aspects of '841 patent and '659 patent
215
216 % Beta table, Table 1, '841 col 16
217 BetaQArray = zeros(1,56); % Index 1 corresponds to QP=0, index 56 corresponds to QP=56
218 BetaQArray(17:29)=(6:18);
219 BetaQArray(30:52)=(20:2:64);
220 BetaQArray(53:56)= [64 64 64 64];
221
222 % Tc array, '659 col 11
223 TcArray = zeros(1, 56);
224 TcArray(1:18)=0;
225 TcArray(19:27)=1;
226 TcArray(28:31)=2;
227 TcArray(32:35)=3;
228 TcArray(36:38)=4;
229 TcArray(39:56)=[5 5 6 6 7 8 9 9 10 10 11 11 12 12 13 13 14 14];
230 Tc2Array = round(TcArray/2); % '659 14:20
231
232 % QP and Quantization Step Size:
233 % Array B(QP) from Q15J28 at p18:
234 % Barray = [ 3881, 4351, 4890, 5481, 6154, 6914, 7761, 8718, 9781,
235 10987,12339,13828, 15523, 17435, 19561, 21873, 24552,27656,30847,34870, 38807, 43747,
236 49103, 54683,61694,68745,77615,89113,100253,109366,126635,141533];
237 % Relationship between B(QP) and quantizer step size:
238 % Q15J28 at 18: dequantized level k = level * B(QP)
239 % Q15J28 at 16: inverse transform scales sample values by 676
240 % Q15J28 at 18: sample values are downscaled by 2^20 after inverse transform
241 % Therefore J15Q28 Quant Step Size = (Barray(QP)*676) / 2^20
242 % For QP=0, 1, 2, 3, 4, 5, 6,.. approx step size = 2.5, 2.81, 3.15, 3.53, 3.97, 4.46,
243 5..., doubling every QP+6
244 % JCTVC-B205 document: quantizer step size ranges from 0-51
245 % B205 at _45023 : basic principle is the same as H.264/AVC
246 % Richardson 2010 H.264 book at 191: mapping QP (range 0-51) to (approximate) Qstep
247 % For QP=12, 13, 14, 15, 16, 17, 18, ..., approx step size = 2.5, 2.75, 3.25, 3.5, 4,
248 4.5, 5..., doubling every QP+6

```

```

243 % Therefore JCTVC-B205 QP in the range 12-43 has approximately the same Quantizer Step
    Size as Q15-J-28 QP in the range 0-31
244
245 QPmod = qp+12; % Modify 'Q15-J-28' QP, add 12 to produce approximately the same
    Quantizer Step Size using the JCTVC-B205 framework
246 QPmod = max(0, min(55, QPmod));
247
248 % Calculate Thr1, Thr2, '841 patent 16:59-60
249 MyBeta = BetaQPararray(QPmod+1); % Find the current value of Beta
250 Thr1 = round(MyBeta/6);
251 Thr2 = round(MyBeta/6);
252
253 % Calculate Tc, Tc2, '659 patent col 11, col 14
254 tc = TcArray(QPmod+1);
255 tc2 = Tc2Array(QPmod+1);
256
257 % '841 Embodiment 3' method, calculate dpi and dqi for every line, '841 9:41-59
258 FirstOffset=zeros(1,8);
259 SecondOffset=zeros(1,8);
260 ThirdOffset=zeros(1,8);
261 FirstOffsetClip=zeros(1,8);
262 SecondOffsetClip=zeros(1,8);
263 ThirdOffsetClip=zeros(1,8);
264 ModPixels2 = zeros(1,8); % Number of pixels with non-zero offsets
265 FiltPixels2 = zeros(1,8); % Number of pixels filtered
266 FiltPixelsP = zeros(1,8); % Number of P pixels filtered
267 FiltPixelsQ = zeros(1,8); % Number of Q pixels filtered
268 BlockOutNoClip = zeros(8,8); % Output without Tc/Tc2 clipping
269 BlockOutClip = zeros(8,8); % Output with Tc/Tc2 clipping
270 fprintf(FIDstats, '\n\n''841/' '659 decisions,');
271 if (UseAltThreshold==0)
272     fprintf(FIDstats, ', QP=%d, using Thr1 and Thr2\n', QPmod);
273 else
274     fprintf(FIDstats, 'using S01 and S02 decisions\n');
275 end
276
277 for i=1:8 % For each line across the two blocks
278     p3=pixin(i,1); p2=pixin(i,2); p1=pixin(i,3); p0=pixin(i,4); % p3p2p1p0|
279     q0=pixin(i,5); q1=pixin(i,6); q2=pixin(i,7); q3=pixin(i,8); % |q0q1q2q3
280     p1filt = p1; q1filt = q1; p1filtClip = p1; q1filtClip = q1; % These may be changed
        due to filtering
281     dpi = abs(p2 - (2*p1) + p0); % '841 claim 5
282     dqi = abs(q2 - (2*q1) + q0); % '841 claim 5
283     FiltPixels2(i)=2; % Count of pixels to filter
284     % Calculate offsets
285     FirstOffset(i) = round(((9 * (q0-p0)) - (3 * (q1-p1)))/16);
286     FirstOffsetClip(i) = max(-tc, min(tc, FirstOffset(i))); % Clip first offset to (-tc,
        tc)
287     SecondOffset(i) = round((p0 + p2 - (2*p1) + (2*FirstOffset(i)))/4);
288     SecondOffsetClip(i) = max(-tc2, min(tc2, SecondOffset(i))); % Clip to (-tc2, tc2)
289     ThirdOffset(i) = round((q0 + q2 - (2*q1) - (2*FirstOffset(i)))/4);
290     ThirdOffsetClip(i) = max(-tc2, min(tc2, ThirdOffset(i))); % Clip to (-tc2, tc2)
291     % Calculate filtered output, with and without clipping
292     p0filt = p0 + FirstOffset(i); % '659 claim 4, 841 claim 10
293     q0filt = q0 - FirstOffset(i); % '659 claim 4, 841 claim 10
294     if not(isequal(FirstOffset(i),0))
295         ModPixels2(i) = ModPixels2(i)+2; % If FirstOffset is non-zero, both p0 and q0
            values are modified
296     end
297     % Decision whether or not to filter p1, q1
298     if (UseAltThreshold==0)
299         pldecision = (dpi < Thr1); % '1' if dpi < Thr1, '0' otherwise
300         qldecision = (dqi < Thr2); % '1' if dqi < Thr1, '0' otherwise
301     else % Use the 'Q15J28 S01 and S02' decisions to determine whether to filter p1, q1
302         S01b = 12*abs(p2 - (2*p1) + p0); % Assumed use of b, c, d instead of a, b, c as

```

```

303         stated in Q15J28 at 19: S01=12*abs(b - (2*c) + d);
304         S02b = 12*abs(q0 - (2*q1) + q2); % Recalculate S01, S02 : S02=12*abs(e - (2*f) +
305         g);
306         LimQP = LimQPararray(qp+1); % Lim(qp), based on the 'Q15J28 qp'
307         pldecision=(S01 < LimQP);
308         qldecision=(S02 < LimQP);
309     end
310     if pldecision % Apply filtering to p1
311         plfilt = p1 + SecondOffset(i); % 659 claim 4, 841 claim 10
312         if not(isequal(SecondOffset(i),0))
313             ModPixels2(i) = ModPixels2(i)+1; % p1 value is changed
314         end
315         FiltPixels2(i) = FiltPixels2(i)+1; % Filter an extra pixel in block P
316         FiltPixelsP(i) = FiltPixelsP(i)+1;
317     end
318     if qldecision % Apply filtering to q1
319         qlfilt = q1 + ThirdOffset(i); % 659 claim 4, 841 claim 10
320         if not(isequal(ThirdOffset(i),0))
321             ModPixels2(i) = ModPixels2(i)+1; % q1 value is changed
322         end
323         FiltPixels2(i) = FiltPixels2(i)+1; % Filter an extra pixel in block Q
324         FiltPixelsQ(i) = FiltPixelsQ(i)+1;
325     end
326     p0filtClip = p0 + FirstOffsetClip(i); % '659 claim 2
327     q0filtClip = q0 - FirstOffsetClip(i); % '659 claim 2
328     if (dpi < Thr1)
329         plfiltClip = p1 + SecondOffsetClip(i); % 659 claim 6
330     end
331     if (dqi < Thr2)
332         qlfiltClip = q1 + ThirdOffsetClip(i); % 659 claim 6
333     end
334     BlockOutNoClip(i,:) = [p3 p2 plfilt p0filt q0filt qlfilt q2 q3];
335     BlockOutClip(i,:) = [p3 p2 plfiltClip p0filtClip q0filtClip qlfiltClip q2 q3];
336 end
337 fprintf(FIDblocks, '841/659 filtered blocks, no clipping:\n');
338 for i=1:8
339     for j=1:4 % Print filtered blocks
340         fprintf(FIDblocks, '%3d ', BlockOutNoClip(i,j));
341     end
342     fprintf(FIDblocks, ' | ');
343     for j=5:8
344         fprintf(FIDblocks, '%3d ', BlockOutNoClip(i,j));
345     end
346     fprintf(FIDblocks, '\n');
347 end
348 fprintf(FIDblocks, '841/659 filtered blocks, with clipping:\n');
349 for i=1:8
350     for j=1:4 % Print filtered blocks
351         fprintf(FIDblocks, '%3d ', BlockOutClip(i,j));
352     end
353     fprintf(FIDblocks, ' | ');
354     for j=5:8
355         fprintf(FIDblocks, '%3d ', BlockOutClip(i,j));
356     end
357     fprintf(FIDblocks, '\n');
358 end
359 % Print statistics
360 fprintf(FIDstats, '# Pixels with non-zero offset :%2d %2d %2d %2d %2d %2d %2d %2d',
361 ModPixels2);
362 fprintf(FIDstats, '\n# Filtered pixels:%2d %2d %2d %2d %2d %2d %2d %2d', FiltPixels2);
363 fprintf(FIDstats, '\n# Filtered P pix: %2d %2d %2d %2d %2d %2d %2d %2d', FiltPixelsP);
364 fprintf(FIDstats, '\n# Filtered Q pix: %2d %2d %2d %2d %2d %2d %2d %2d', FiltPixelsQ);

```

```

365 fprintf(FIDstats, '\nFirstOffset:      %2d %2d %2d %2d %2d %2d %2d %2d', FirstOffset);
366 fprintf(FIDstats, '\nSecondOffset:     %2d %2d %2d %2d %2d %2d %2d %2d', SecondOffset);
367 fprintf(FIDstats, '\nThirdOffset:       %2d %2d %2d %2d %2d %2d %2d %2d', ThirdOffset);
368 fprintf(FIDstats, '\nFirstOffsetClip:   %2d %2d %2d %2d %2d %2d %2d %2d', FirstOffsetClip
);
369 fprintf(FIDstats, '\nSecondOffsetClip:  %2d %2d %2d %2d %2d %2d %2d %2d', SecondOffsetClip
);
370 fprintf(FIDstats, '\nThirdOffsetClip:   %2d %2d %2d %2d %2d %2d %2d %2d', ThirdOffsetClip
);
371
372 fprintf(FIDstats, '\n\n');
373
374 % End of main script
375
376 % ClipQP function, Q15J28 p20
377 function out = ClipQP(qp_in, strengthval)
378
379 % ClipQP tables, p20, indexed by QP 0-31, assumed that there are suitable values for
QP=28
380 ClipQP1 = [0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 3 3];
381 ClipQP2 = [0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 3 3 3 3 4 4 5 5];
382 ClipQP3 = [0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 2 2 2 2 3 3 3 4 4 5 5 5 6 7 8 9];
383
384 if (qp_in<0) || (qp_in>31) || (strengthval <1) || (strengthval>3)
385     fprintf(1, 'Error: QP %d, strength %d is out of range/n', qp_in, strengthval);
386 end
387
388 out0 = 0;
389 if (strengthval == 1)
390     out0 = ClipQP1(qp_in+1);
391 end
392 if (strengthval == 2)
393     out0 = ClipQP2(qp_in+1);
394 end
395 if (strengthval == 3)
396     out0 = ClipQP3(qp_in+1);
397 end
398
399 out = out0;
400
401 end % Function ClipQP
402
403 % Clip8 function, clip to range 0-255
404 function out = Clip8(in)
405
406 out = min(255, max(0, in));
407
408 end % Function Clip8
409
410

```