IFF Icon Data Encoder

```
585   int i;
586   numbits = 0;
587
588   //copy
589   {
590       if((i = copy) > 128) i = 128;
591       bitbuf = (bitbuf<<8) | (i-1); //RLE control byte
592       byte = bitbuf;
593       Printf ("EncodeBytes - RLE upper bitbuf byte: %d\n", byte);
594   }
595   //loop
596   {
597       if((i = equal) > 128) i = 128;
598       bitbuf = (bitbuf<<8) | (256-(i-1));  //RLE control byte
599       byte = bitbuf;
600       Printf ("EncodeBytes - RLE upper bitbuf byte: %d\n", byte);
601       i = 1 //2-byte pairs only have an RLE byte + loop/copy byte
602   }
603   out[idx++] = (bitbuf >> numbits); //process the byte
604
605   while(i--) //process one time for each i, loop or copy
606   {
607       numbits += depth; //increment bit index by depth
608       bitbuf = (bitbuf<<depth) | src[i++]; //load the byte
609       Printf ("EncodeBytes - RLE lower bitbuf byte: %d\n", src[i++]);
610       Printf ("EncodeBytes - RLE numbits byte: %d\n", numbits);
611
612       if(numbits >= 8) //process loop or copy bytes
613       {
614           numbits -= 8; //adjust bit index
615           out[idx++] = (bitbuf >> numbits);  //process the byte
616           byte = (bitbuf >> numbits);
617           Printf ("EncodeBytes - RLE lower bitbuf byte: %d\n", byte);
618           Printf ("EncodeBytes - RLE new numbits: %d\n", numbits);
619       }
620   }
```

//input bytes
10000001, 00000, 11010000, 00000, 11111111, 00001, 11111111, 00010, 11110000, 00011,
[129]128 0 [208]49 0 [255]2 1 [255]2 2 [240]17 3

//output bytes
129 06 128 63 195 254 47 01 255
10000001, 00000110, 10000000, 00111111, 11000011, 11111110, 00101111, 00000001, 11111111,

//32bit bit buffer operations

upper bitbuf = 10000001 load 129 >>0=**129**, numbits = 0

lower bitbuf = 10000001 00000 load 0, numbits = 5

upper bitbuf = 10000001 00000 11010000 >>5=00000110=**6** load 208, numbits = 5

lower bitbuf = 10000001 00000 11010000 00000 load 0, numbits = 10-8=2 //if(numbits >= 8) (numbits-8)

lower bitbuf = 10000001 00000 11010000 00000 >>2=10000000=**128**, numbits = 2

upper bitbuf = 10000001 00000000 00000 11010000 00000 11111111 load 255, >>2=00 111111=**63**, numbits = 2

lower bitbuf = 00000 11010000 00000 11111111 00001 load 1, numbits = 7

upper bitbuf = 11010000 00000 11111111 00001 11111111, load 255 >>7=11 00001 1=**195**, numbits = 7

lower bitbuf = 00000 11111111 00001 11111111 00010, load 2 >>4=1111111 0=**254** numbits = 12-8=4

//**********//

EncodeBytes (convert bitstream of data)

The encoder bounces back and forth between upper and lower bitbuf to process the rle data based on numbits (bit index).The upper bitbuf loads the 8bit rle control byte, followed immediately by buf[idx++] = (bitbuf >> numbits); which writes the byte to the output buffer based on numbits (bit index).

The lower bitbuf performs two tasks. It uses numbits += depth; which increments bit index before loading a byte. For the first task if numbits (bit index) is less than 8 it bounces back to the upper bitbuf without processing. Processing the current byte will be done in the upper bit buffer after loading another RLE byte. For the second task if numbits (bit index) is greater or equal to 8 it adjusts bit index using if(numbits >= 8) (numbits-8). So if the bit index is 10 it becomes 2. After the bit adjust it then processes the byte in the lower bit buffer using buf[idx++] = (bitbuf >> numbits); Then it bounces back to the upper bitbuf to load another 8bit rle control byte to continue processing.

//**********//