

/* COMP_RLE description:

byte N:

+N: N bytes of uncompressed data follow

0: generic escape code (see below)

-1: following byte is byte-count for word/long data (pos/neg)
-1 \$ab \$cd \$ef \$gh \$ij
\$ab > 0 means \$cdef is word data repeated \$ab times
\$ab < 0 means \$cdefghij is long data repeated -\$ab
\$ab == 0 reserved

-2: next byte is byte-count of pixels to skip, count of 0 means use LUT
encoding
-2 \$ab \$cd \$ef \$gh
\$ab == 0 means use LUT encoding for this scan line
\$cd is table size, or zero for same as last table
the table is found as the next \$cd bytes
\$ab != 0 skip \$ab pixels

-N: repeat the following byte N times

Generic escape code: 0 \$ab \$cd \$ef \$gh \$ij \$kl

case \$a of:

%0000 (0) - \$bcd is a count for uncompressed data
%0001 (1) - \$bcd is a count for byte-data \$ef
%0010 (2) - \$bcd is a count for word-data \$efgh
%0011 (3) - \$bcd is a count for long-data \$efghijkl
%0100 (4) - \$bcd is a count of the number of pixels to skip
%0101 (5) - \$bcd is a count of the number of scan lines to skip
(0 means go directly to next scan line)
%0110 (6) - fill scan lines
switch (\$b)
%00: fill next \$cd scan lines with \$ef
%01: fill next \$cd scan lines with \$efgh
%11: fill next \$cd scan lines with \$efghijkl

%1111 (15) - END OF BITMAP

UNASSIGNED (Lempel-Ziv section)

\$cd is a count for repeated run of the previous \$b bytes
copy data from above: \$b is dx offset (-8 to 7)
\$cd is the byte count
\$ef is # of lines back (0 = same line)
copy data from above: \$b contains 2 sets of 2 high bits for x1 and d1
\$cd is # of lines back (0 = same line)
\$ef is x1, the start byte
\$gh is d1, the count byte
copy the next \$cd bytes from the scan line above (0 = till end of scan

line)

*/

Tuesday, September 25, 1990 2:39 PM

```

DIFF_NONE = 0, /* use MOVE, don't depend on previous frame data (full rect ext
DIFF_MOVE,    /* use MOVE, does depend on previous (skips or non-full extent
DIFF_XOR      /* use XOR, does depend on previous frame (obviously) */
} Diff;

```

```

/* compression types (up to 8 different kinds) */

```

```

typedef enum {
    COMP_NONE,          /* uncompressed bitmap */
    COMP_RLE,           /* CAVF comprehensive RLE format */
    COMP_PICT,          /* standard PICT file */
    COMP_PICT_LUT,      /* PICT, using lut from within */
    COMP_4BIT           /* histogram-computed 4-bit codes (MacWrite style) */
} Comp;

```

```

/* RLE description:

```

```

byte N:

```

```

+N:      N bytes of uncompressed data follow
0:      generic escape code (see below)
-1:     next byte is byte-count followed by word-length data
-2:     next byte is byte-count of pixels to skip, count of 0 and 1 are reserved
        count 0: rest of scan-line is uncompressed data
        count 1: 2
-N:     repeat the following byte N times
-128:   repeat the following byte out to the completion of this scan line

```

```

Generic escape code: $00 $ab $cd $ef $gh $ij $kl

```

```

case $a of:
%1000 - $bcd is a pixel skip count
%1001 - $bcd is a count for byte-data $ef
%1010 - $bcd is a count for word-data $efgh
%1011 - $bcd is a count for long-data $efghijkl
%1100 - $cd is a count for repeated run of the following $b bytes
%0001 - copy data from above:  $b is dx offset (-8 to 7)
                                $cd is the byte count
                                $ef is # of lines back (0 = same line)
%0010 - copy data from above:  $b contains 2 sets of 2 high bits for x1 and d1
                                $cd is # of lines back (0 = same line)
                                $ef is x1, the start byte
                                $gh is d1, the count byte
%0011 - copy the next $cd bytes from the scan line above (0 = till end of scan
%0100 - fill next $cd scan lines with byte value $ef

```

```

*/

```

```

typedef struct {

```

```

    Opcode      type;          /* BITMAP */
    int         left, top;
    int         width, height; /* area of change */
    Byte        bw : 1;        /* black and white (one-bit) */
    Byte        diff_type : 3; /* Differencing type */
    Byte        comp_type : 4; /* Compression type */
    Byte        data[];

```

```

} OpBitmap;

```

0 - bcd non-run
 1 - bcd bytewise
 2 - " word run
 3 - bcd long run
 4 - bcd skip count

5 - variable length run

15 - END of Data

5 - skip \$cd scanlines

5 - fill next \$cd scan lines with

\$b=00 byte value that follows
 01 word
 10 long

zero means next byte is count for long-word run

next word is bits for LUT entries that change followed by LUT values in that order

word long

NIBBLES 'n BITS

1-7 → That many nibbles of uncompressed data follow

8 → 3 → Repeat following nibble this many times.
 following nibble ~~word~~ count (> 0 word, < 0 lw)

-2 ~~following nibble~~ ~~count~~ skip

0 escape code

following nibble

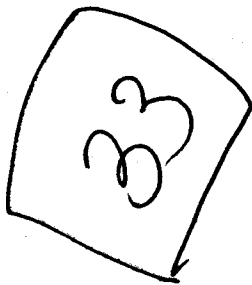
byte follows

non run count
 byte run count
~~word~~
 long

skip count
 skip lines (0 → to end)

nibble to repeat in
 BOTTOM of following byte

if 1 →



16 381 @ 000
 358 000
 405

0xffff

==

7 row