#171: _PackBits Data Format

Revised by: Guillermo Ortiz February 1989
Written by: Cameron Birse November 1987

This Technical Note describes the format of data packed by the Toolbox utility `_PackBits`. Although you can simply unpack these files using the `_UnPackBits` call, we provide this information here for the terminally curious and for those manipulating MacPaint documents or PICT files by hand.
Warning: This information is subject to change.
Changes since March 1988: Added information on PICT files which contain data from the `_PackBits` routine.

First there is a byte which specifies whether or not the the data is packed, and this byte is also the count byte. It is a negative number if the data is packed (i.e., the high bit is set). If the high bit is set, then that complete byte is a two's complement number that tells you how many bytes were packed. If it is a positive number, then it is simply a zero-based count of how many discrete data bytes exist. Consider the following example:

Unpacked data:

```
AA AA AA 80 00 2A AA AA AA AA 80 00 2A 22 AA AA AA AA AA AA AA AA AA AA
```

After being packed by `_PackBits`:

```
FE AA               ; (-(-2)+1) = 3 bytes of the pattern $AA
02 80 00 2A         ; (2)+1 = 3 bytes of discrete data
FD AA               ; (-(-3)+1) = 4 bytes of the pattern $AA
03 80 00 2A 22      ; (3)+1 = 4 bytes of discrete data
F7 AA               ; (-(-9)+1) = 10 bytes of the pattern $AA
```

or

```
FE AA 02 80 00 2A FD AA 03 80 00 2A 22 F7 AA
*     *           *     *              *
```

The bytes with the asterisk (*) under them are the count or flag bytes. `_PackBits` will only pack the data when there are 3 or more consecutive bytes with the same data, otherwise it just copies the data byte for byte (and adds the count byte).

Note: The data associated with some PICT opcodes, $0098 (`PackBitsRect`) and $0099 (`PackBitsRgn`), contain `PixData` which is basically made of `_PackBits` data. It should be noted, though, that the format for `PixData` includes a `byteCount` or length in addition to the data described in this note.

For example, the following is the result of decoding a part of a sample PICT:

```
01CE 0000 0000 02D0 0240                    /* Size and bounds, word + Rect      */
11                                          /* pict 1, byte                          */
01                                          /* version, byte                     */
A0 00 82                                    /* short message, byte + 2 byte data */
01 000A 0000 0000 02D0 0240                 /* clip rgn, byte opcode + Rgn data  */
98                                          /* PackBitsRect, byte opcode         */
000A                                        /* row bytes, word                   */
00C2 007E 00EB 00C6                         /* bounds, Rect                          */
00C2 007E 00EB 00C5                         /* source rect,       Rect               */
0099 0090 00C2 00D7                         /* dest rect, Rect                   */
0001                                        /* mode,        Word                     */

/* Now we have the  scan line data
   which is packed into the following:

[ byteCount for current scan line ] [ data as defined above ]

If rowBytes > 250 then byteCount is a word else is a byte;
*/

/* first scan line */
04                                          /* number of bytes(byte since less than 250}     */
F8 00 00 67                                 /* -(-8)+1 of '00' and (0)+1 of '67'=10  */

/* second scan line */
05                                          /* number of bytes                           */
F9 00 01 01 6D                              /* -(-7) + 1 '00' and (1) + 1 '01 6D' = 10 */

/* The same math applies to all the following scan lines */
/* third line */
0A                                          /* ten bytes                                 */
04 00 00 03 FF FC FD 00 00 A7               /* '00 00 03 FF FC' '00 00 00 00' 'A7' = 10      */

/* next line */
0A                                          /* ten bytes                                 */
04 00 00 02 3F F2 FD 00 00 67               /* '00 00 02 3F F2''00 00 00 00''67' = 10  */

/* last line */
08                                          /* number of bytes                           */
04 00 00 02 3E 31 FC 00                     /* '00 00 02 3E 31' '00 00 00 00 00' = 10  */

/* the very last line */
0B                                          /* bytes                                     */
04 00 00 02 3E 31 FE 00 01 01 EA            /* '00 00 02 3E 31''00 00 00' '01 EA' = 10 */

/* more data follows */
```

Further Reference:

- *Inside Macintosh*, Volume I-465, The Toolbox Utilities
- *Inside Macintosh*, Volume V-39, Color QuickDraw
- Technical Note #86, MacPaint Document Format
