```
/*++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
+++++

Copyright (c) 2001 BeComm Corporation

Filename:

    bmpdecoder.c

Group Name:

    todo

Group Overview:

    todo

Owner:

    Guy Carpenter (guyc) 19-Jul-2001

----------------------------------------------------------------------------
---*/

#define SOS_DEBUG_ZONE "/beads/bmpdisplay"

#include <sosstrings.h>
#include "bmpdecoder.h"

SOS_SOURCE_VERSION (
    "$Id: bmpdecoder.c,v 1.1 2001/09/06 21:07:58 guyc Exp $"
);

/*++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
+++++
Types and Structures
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
+++*/



struct _BMP_DECODER {
    BMP_FORMAT          Format;
    SOS_UINT32 *        ColorInfo;
    SOS_BOOLEAN         FormatIsSet;
```

```
42 };
43
44 /* is this word or dword alignment?
45  * V1 used dword, so we will too
46  */
47 #define WORD_ALIGN(X) ((X+1)&(SOS_UINT32_MAX-1))
48 #define DWORD_ALIGN(X) ((X+3)&(SOS_UINT32_MAX-3))
49 #define RLE8_WRITE_PIXEL(P) *dest++=BmpDecoder->ColorInfo[P]
50
51 /*++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
… +++++
52 Decoders
53 ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
… +++*/
54 static
55 SOS_STATUS
56 FrameDecode_Rle8(
57     BMP_DECODER *        BmpDecoder,
58     void *               EncodedFrame,
59     size_t               EncodedFrameSize,
60     void *               DecodedFrame,
61     size_t               DecodedFrameSize
62 )
63 {
64     SOS_STATUS status = SOS_Success;
65     unsigned char *src = EncodedFrame;
66     SOS_UINT32 size = EncodedFrameSize;
67     SOS_UINT32 align;
68     unsigned char count;
69     SOS_UINT32 width = BmpDecoder->Format.Width;
70     SOS_UINT32 height= BmpDecoder->Format.Height;
71     SOS_UINT32 *dest = DecodedFrame;
72
73     /* destEnd is the pixel entry following
74        the legal write area, used to detect overflow */
75     SOS_UINT32 *destEnd = dest+(DecodedFrameSize/sizeof(SOS_UINT32));
76     dest += width * (height-1);
77
78     /*
79      * In debug mode we check the cursor position to make sure it never
80      * gets out of bounds.
81      */
82 #if DEBUG
83 #define CHECK() if (dest<(SOS_UINT32*)DecodedFrame || dest>=destEnd)
    {SOS_DebugStringPrint("Range check failed %p %p
```

```
83… %p\n",DecodedFrame,dest,destEnd);SOS_ASSERT_ASSUMPTION(0,"check");}
84 #else
85 #define CHECK()
86 #endif
87
88     /*
89      * This loop is written with the intention of making easy
90      * to support streaming (incremental processing as data
91      * becomes available).  At this stage it isn't needed.
92      */
93
94 //    SOS_DEBUG_LEVEL_SET(5);
95
96     while (size>=2) {  /* need two bytes minimum for any op */
97         if (src[0]==0) {
98             /*
99              * PROCESS ESCAPES
100             * 0 character is an escape.  Meaning
101             * depends upon following character:
102             * 0: end of line
103             * 1: end of bitmap
104             * 2: delta - following two bytes contain unsigned values
… indicating
105             *    the horizontal and vertical offsets of the next pixel.
106             * default: next byte represents number of following bytes
107             *    each of which contains the index of a single pixel.
… When
108             *    second byte is 2 or less, has same meaning as in encoded
109             *    mode.  In absolute mode, each run must end on 2-byte
110             *    word boundary.
111             */
112            switch (src[1]) {
113            case 0:            /* end of line   */
114            {
115                SOS_UINT32 *start = (SOS_UINT32*)DecodedFrame;
116                size_t delta = dest - start;
117                size_t x = delta % width;
118                size_t y = delta / width;
119                SOS_DEBUGOUT_FUNC_TRACE(
120                    "BMP escape eol %ld %ld\n",
121                    x,y
122                );
123            }
124                src+=2;        /* consume two bytes */
```

```
126                 /* step to start of next line
127                  * need to be careful here.  If we just
128                  * stepped over the end of the line, we dont
129                  * want to skip again.  We assume there
130                  * wont be a eol at position 0
131                  */
132                 {
133 //                      size_t y = ((dest-1-(SOS_UINT32*)DecodedFrame) %
    … width);
134 //                      dest += width - y - 1;
135
136                     size_t x = ((dest-(SOS_UINT32*)DecodedFrame) % width);
137                     dest -= (width<<1) + x;
138                 }
139             {
140                 SOS_UINT32 *start = (SOS_UINT32*)DecodedFrame;
141                 size_t delta = dest - start;
142                 size_t x = delta % width;
143                 size_t y = delta / width;
144                 SOS_DEBUGOUT_FUNC_TRACE(
145                     "  now at %ld %ld\n",
146                     x,y
147                 );
148             }
149
150                 break;
151             case 1:             /* end of bitmap */
152                 SOS_DEBUGOUT_FUNC_TRACE("BMP escape eof\n");
153                 src+=2;         /* consume two bytes */
154                 size-=2;
155                 goto rle8_done;
156             case 2:             /* delta         */
157                 if (size<4) {  /* need 3:0+dx+dy*/
158                     goto rle8_done;
159                 }
160                 SOS_DEBUGOUT_FUNC_TRACE(
161                     "BMP escape delta %d %d\n",
162                     src[2],
163                     src[3]
164                 );
165                 dest += src[2] - width * src[3];
166                 src+=4;
167                 size-=4;
168                 break;
```

```
170                 /* we need size[1] bytes to proceed */
171                 if (size<(SOS_UINT32)(src[1]+2)) {
172                     goto rle8_done;
173                 }
174                 SOS_DEBUGOUT_FUNC_TRACE(
175                     "BMP absolute %d\n",
176                     src[1]
177                 );
178                 count=src[1];
179                 src+=2;
180                 size-=2;
181                 while (count-->0) {
182                     if (dest>=destEnd) {
183                         status = SOS_ErrorFormat;
184                         goto rle8_done;
185                     }
186                     CHECK();
187                     RLE8_WRITE_PIXEL(*src++);
188                     size--;
189                 }
190 
191                 {
192                     size_t length = src-(unsigned char *)EncodedFrame;
193                     align = WORD_ALIGN(length)-length;
194                 }
195 
196 
197                 while (align--) {
198                     size--;
199                     src++;
200                 }
201             }
202         } else {
203             /* first byte is not zero, run length encoding */
204             SOS_DEBUGOUT_FUNC_TRACE(
205                 "BMP repeat %d %d\n",
206                 src[0],
207                 src[1]
208             );
209             count=*src++;
210             size--;
211             while (count-->0) {
212                 if (dest>=destEnd) {
213                     status = SOS_ErrorFormat;
```

DOCKET ALARM

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS
Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS
Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS
Sync your system to PACER to automate legal marketing.