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

$Id: rgbalterencoding.c,v 1.8 2001/10/02 21:15:02 guyc Exp $

Copyright (c) 2001 BeComm Corporation

Filename:

    rgbalterencoding.c

Abstract:

    Transforms the image's pixels from their current encoding to a new
one.  This
    is useful for compressing an image and altering the pixel format
    for the end display device.

    A new encoding is specified as a path attribute with the name
    "newencoding".  It should be a 32 Integer.

Owner:

    Andy Kutner (andyk)

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

#define SOS_DEBUG_ZONE "/beads/rgbalterencoding"
#include <sosstrings.h>
#include <sosmultimedia.h>

/*++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
+++++
MACROS
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
+++*/
#define RED_SIZE(A)   ((A) & 0xFF)
#define GREEN_SIZE(A) (((A) & 0xFF00) >> (SOS_BITSPERBYTE))
#define BLUE_SIZE(A)  (((A) & 0xFF0000) >> (2 * SOS_BITSPERBYTE))
#define TOTAL_SIZE(A) (((A) & 0xFF000000) >> (3 * SOS_BITSPERBYTE))
#define BLANK_SIZE(A) (TOTAL_SIZE(A) - (RED_SIZE(A) + GREEN_SIZE(A) +
BLUE_SIZE(A)))
/* shift the total size right by 3 bytes.  then shift it right by 3
```

```
40 #define TOTAL_SIZE_BYTES(A) (((A) & 0xFF000000) >> (3 * SOS_BITSPERBYTE +
… 3))
41
42 /*++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
… +++++
43 Named Constants
44 ------------------------------------------------------------------------
… ---*/
45
46 /* Name of bead */
47 static const char BEAD_NAME[] = "rgbalterencoding";
48
49 static const char VIDEOCONTEXT_CLASS_NAME[] = "rgbcontext";
50
51 static const char NEW_ENCODING[] = "newencoding";
52
53 /*++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
… +++++
54 Structs
55 ------------------------------------------------------------------------
… ---*/
56 typedef struct _RGBALTERENCODING_CONTEXT {
57     SOS_IVIDEOCONTEXT *   InVideoContext;
58     SOS_IVIDEOCONTEXT *   OutVideoContext;
59     SOS_VIDEO_FORMAT      OutFormat;
60     SOS_UINT32            OutBufferSize;
61
62     SOS_UINT32            RedMask;
63     SOS_UINT32            GreenMask;
64     SOS_UINT32            BlueMask;
65
66     SOS_UINT32            RedRightShift;
67     SOS_UINT32            GreenRightShift;
68     SOS_UINT32            BlueRightShift;
69
70     SOS_UINT32            RedLeftShift;
71     SOS_UINT32            GreenLeftShift;
72     SOS_UINT32            BlueLeftShift;
73 } RGBALTERENCODING_CONTEXT;
74
75 /*++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
… +++++
76 Context Stuff
77 ------------------------------------------------------------------------
```

```
78
79  static
80  void
81  RgbScale_ContextDestroy(
82      RGBALTERENCODING_CONTEXT *        Context
83  )
84  {
85      if (Context) {
86          SOS_Interface_Release(Context->InVideoContext);
87          SOS_Interface_Release(Context->OutVideoContext);
88          SOS_Mem_Free(Context);
89      }
90  }
91
92  static
93  RGBALTERENCODING_CONTEXT *
94  RgbScale_ContextCreate(
95      void
96  )
97  {
98      RGBALTERENCODING_CONTEXT *context;
99
100     context = SOS_Mem_Alloc(sizeof(*context));
101     if (context) {
102         SOS_memset(context, 0, sizeof(*context));
103
104         context->OutVideoContext = SOS_Interface_CreateFromClassName(
105             VIDEOCONTEXT_CLASS_NAME,
106             SOS_IVIDEOCONTEXT_ID
107         );
108
109         SOS_ASSERT_ASSUMPTION(
110             context->OutVideoContext!=NULL,
111             "Couldn't create video context"
112         );
113
114         if (!context->OutVideoContext){
115             SOS_Mem_Free(context);
116             context = NULL;
117         }
118     }
119     return context;
120 }
121 /*++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
122 Initialization
123 ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
… +++*/
124
125
126 /*++
127 Routine Name:
128
129     GetInVideoContext
130
131 Routine Description:
132
133     Retrieve the video context for the incoming video stream.
134
135 Parameters:
136
137     RGBALTERENCODING_CONTEXT*     Context - [in/out]
138         A session context should already be allocated.  This routine
139         filles in the InVideoContext variable.
140
141     SOS_PATH*             Path    - [in]
142         The path this session will receive images on.
143
144 Return Value:
145
146     SOS_STATUS -
147         SOS_Success on success.
148         Any other value means an error occured.
149
150 --*/
151
152 static
153 SOS_STATUS
154 GetInVideoContext(
155     RGBALTERENCODING_CONTEXT*     Context,
156     SOS_PATH*             Path
157 )
158 {
159     SOS_STATUS status;
160     SOS_REGOBJECT *contextObject;
161
162     SOS_DEBUGOUT_FUNC_TRACE("SetInVidoContext\n");
163
164     status = SOS_Path_AttributeGet(
```

```
166         SOS_VIDEOCONTEXT_NAME,
167         &contextObject
168     );
169 
170     SOS_ASSERT_SOFT_ERROR( SOS_SUCCEEDED(status), "Path context does not
…  contain a video context");
171 
172     if (SOS_SUCCEEDED(status)) {
173 
174         status = SOS_RegObject_InterfaceGet(
175             contextObject,
176             SOS_IVIDEOCONTEXT_ID,
177             (void**)&(Context->InVideoContext)
178         );
179 
180         SOS_ASSERT_SOFT_ERROR( SOS_SUCCEEDED(status), "Video context does
…  not support required interface");
181         SOS_RegObject_Release(contextObject);
182     }
183 
184     return status;
185 }
186 
187 /*++
188 Routine Name:
189 
190     SetOutVideoContext
191 
192 Routine Description:
193 
194     Retrives the new encoding from the Path and then creates a set of
195     mask's and shifts used to coerse the data from the input format to
196     the output format.
197 
198 Parameters:
199 
200     RGBALTERENCODING_CONTEXT*     Context  - [in]
201         The session context.
202 
203     SOS_PATH*             Path     - [in]
204         The Path that we are receiving messages on.
205 
206     SOS_VIDEO_FORMAT*     InFormat - [in]
207         The Pixel format of the incoming data.
```

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.