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

Copyright (c) 2001 BeComm Corporation

Filename:

    clocksync.c

Group Name:

    todo

Group Overview:

    Used in conjunction with timesync.

    Used to propogate a master/render clock pair over
    a network link.

    Passes the following elements:
    - epoch from the master clock.
    - frequency/divisor from the render clock. (NO LONGER REQUIRED,
REMOVED)

    This is enough to manage timed delivery of video.

    NOTE: Transports only in forward direction (currently) - updates to
the
    render clock are not propogated backwards.

Owner:

    Guy Carpenter (guyc) 16-Aug-2001

--------------------------------------------------------------------------
----*/
#define SOS_DEBUG_ZONE "/beads/clocksync"
#include <sosstrings.h>
#include <sosmultimedia.h>
#include "timesync.h"

SOS_SOURCE_VERSION("$Id: clocksync.c,v 1.11 2001/10/23 17:11:25 guyc Exp
$");
```

```
41 /*++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
… ++++++
42 Named Constants
43 ------------------------------------------------------------------------
… ----*/
44
45 /*
46  * Name of bead
47  */
48 static
49 const char BEAD_NAME[]   =   "clocksync";
50 const char LOOPBACK_CONTENTTYPE[] = "clocksync/loopback";
51
52 #define TIMER_INTERVAL     4000
53 #define TIMER_INITIAL_DELAY 200
54
55 /*++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
… ++++++
56 Masks
57 ------------------------------------------------------------------------
… ----*/
58
59 /*
60  * Order here is important: we use the bit
61  * positions to efficiently compute the length.
62  */
63 #define FLAG_MASTERSHIFT  16
64 #define FLAG_RENDERSHIFT  18
65 #define MASK_SIZE          3
66 #define MASK_LENGTH       0xFFFF
67 #define FLAG_MASTEREPOCH  1<<(FLAG_MASTERSHIFT)
68 #define FLAG_MASTERFREQ   1<<(FLAG_MASTERSHIFT+1)
69 #define FLAG_RENDEREPOCH  1<<(FLAG_RENDERSHIFT)
70 #define FLAG_RENDERFREQ   1<<(FLAG_RENDERSHIFT+1)
71 #define FLAG_LOOPBACKREQ  1<<20
72 #define FLAG_DONTFORWARD  1<<21
73 #define FLAG_UNCONDITIONAL_MASK (FLAG_LOOPBACKREQ | FLAG_DONTFORWARD)
74 #define CONSTANT_LENGTH     2 /* hostid and length+flags in 2x32 bits */
75 #define MAX_VARIABLE_LENGTH 6 /* up to 6 other 32-bit values          */
76
77 /*++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
… ++++++
78 Structs
79 ------------------------------------------------------------------------
   */
```

```
80
81  typedef struct STREAM_CONTEXT {
82      SOS_UINT32          HostId;
83      SOS_CLOCK_TICK      StartTime;
84      SOS_ISAMPLECLOCK * IMasterClock;
85      SOS_ISAMPLECLOCK * IRenderClock;
86      SOS_TIMER_ID        TimerId;
87      SOS_BOOLEAN         RenderUpdate;
88      SOS_PATH *          ReturnPath;
89  } STREAM_CONTEXT;
90
91  /*++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
…  ++++++
92  Context create and destory
93  ----------------------------------------------------------------------
…  ----*/
94
95  /*++
96  Routine Name:
97
98      ClockSync_ContextCreate
99
100 Routine Description:
101
102     Attempt to allocate a ClockSync session and all the resources it
…  requires.
103
104 Parameters:
105
106     None.
107
108 Return Value:
109
110     UPDATE_CONTEXT * -
111         A valid pointer to a session context on success.
112
113         A NULL pointer if an error occured.
114
115 --*/
116 static
117 STREAM_CONTEXT *
118 ClockSync_ContextCreate(
119     void
120 )
```

```
122     STREAM_CONTEXT *context;
123
124     SOS_DEBUGOUT_FUNC_TRACE("ClockSync_ContextCreate called\n");
125
126     context = SOS_Mem_Alloc(sizeof(*context));
127     if (context) {
128         SOS_memset(context, 0, sizeof(*context));
129         TimeSync_HostIdGet(&context->HostId);
130         context->StartTime = SOS_Clock_TickGet();
131 //        context->RenderUpdate = SOS_True;  // REVISIT - set via
    … configuration
132
133     }
134
135     return context;
136 }
137
138 void
139 ClockSync_ContextDestroy(
140     STREAM_CONTEXT *        Context
141 )
142 {
143     if (Context) {
144         SOS_Interface_Release(Context->IMasterClock);
145         SOS_Interface_Release(Context->IRenderClock);
146
147         SOS_Mem_Free(Context);
148     }
149 }
150
151 /*++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
    ++++++
152 Edge Handlers
153 ----------------------------------------------------------------------
    ----*/
154
155 /*++
156 Routine Name:
157
158     ClockSync_KeyCreate
159
160 Routine Description:
161
162     Creates a new key for each session.
```

```
164 Parameters:
165
166     SOS_PATH     *Path    - [in]
167         The new Path to create a Key for
168
169     SOS_MESSAGE  *Message - [in]
170
171 Return Value:
172
173     SOS_STATUS -
174         SOS_PathBuild_Stop on success.
175
176         SOS_Error if something goes wrong.
177 --*/
178 static
179 SOS_STATUS
180 ClockSync_KeyCreate(
181     SOS_PATH     *Path,
182     SOS_MESSAGE  *Message
183 )
184 {
185     static SOS_UINT32 s_UniqueId = 0;
186     SOS_REGOBJECT*  uniqueSessionKey;
187
188     uniqueSessionKey = SOS_UInt32_Create(s_UniqueId++);
189     SOS_Path_SessionKeySet(Path, uniqueSessionKey);
190     SOS_RegObject_Release(uniqueSessionKey);
191
192     return SOS_PathBuild_Stop;
193 }
194
195 static
196 SOS_ISAMPLECLOCK *
197 FindClock(
198     SOS_PATH *   Path,
199     const char * Name
200 )
201 {
202     SOS_REGOBJECT *clock;
203     SOS_ISAMPLECLOCK *iSampleClock = NULL;
204
205     if (SOS_SUCCEEDED(
206             SOS_Path_AttributeGet(
207                 Path,
```

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.