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

Copyright (c) 2001 BeComm Corporation

Filename:

    sampleclock.c

Group Name:

    todo

Group Overview:

    todo

Owner:

    Guy Carpenter (guyc) 16-Aug-2001

--------------------------------------------------------------------------
---*/
#define SOS_DEBUG_ZONE "/classes/sampleclock"

#include <sosstrings.h>
#include <sosmultimedia.h>

SOS_SOURCE_VERSION (
    "$Id: sampleclock.c,v 1.13 2001/10/23 16:40:01 guyc Exp $"
);

/*++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
+++++
Definitions
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
+++*/
#define SAMPLECLOCK_CLASS_NAME "sampleclock"
#define SAMPLECLOCK_SCHEME_NAME "sampleclock"

#define LOCK(C)   SOS_Lock_Acquire(C->Lock)
#define UNLOCK(C) SOS_Lock_Release(C->Lock)
/*++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
+++++
globals
```

```
41 ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
… +++*/
42 static
43 SOS_REGOBJECTCLASS *  g_SampleClockClass = NULL;
44
45 static
46 SOS_STRINGTABLE *     g_NamedClockTable = NULL;
47
48 static
49 SOS_LOCK *            g_NamedClockTableLock = NULL;
50
51 /*++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
… +++++
52 Macros
53 ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
… +++*/
54
55 #define RESOLVE(O)    SOS_REGOBJECT_RESOLVE(O,g_SampleClockClass)
56 #define ISVALID(O)    SOS_REGOBJECT_ISVALID(O,g_SampleClockClass)
57 #define IRESOLVE(I)   SOS_INTERFACE_RESOLVE(I,g_SampleClockClass)
58 #define IISVALID(I)   SOS_INTERFACE_ISVALID(I,g_SampleClockClass)
59
60 /*++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
… +++++
61 Types
62 ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
… +++*/
63
64 typedef struct _SAMPLECLOCK {
65     SOS_CLOCK_TICK              Time;
66     SOS_UINT32                  Sample;
67     SOS_UINT32                  Frequency;
68     SOS_UINT32                  Divisor;
69     SOS_BOOLEAN                 IsSet;
70     SOS_LOCK *                  Lock;
71     char *                      Name;
72 } SAMPLECLOCK;
73
74 /*++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
… +++++
75 Basic Object Methods
76 ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
… +++*/
77 static
```

```
79  SampleClock_Init(
80      SOS_REGOBJECT *     Object
81  )
82  {
83      SOS_STATUS status = SOS_Success;
84
85      SAMPLECLOCK *sampleclock = SOS_Mem_Alloc(sizeof(SAMPLECLOCK));
86      SOS_memset(sampleclock, 0, sizeof(SAMPLECLOCK));
87
88      /* lock the sampleclock until the first call to set */
89      sampleclock->Lock = SOS_Lock_Create();
90
91      SOS_RegObject_DataPut(Object, sampleclock);
92
93      return status;
94  }
95
96  static
97  void
98  SampleClock_Uninit(
99      SOS_REGOBJECT *     Object
100 )
101 {
102     if (ISVALID(Object)) {
103         SAMPLECLOCK *sampleclock = RESOLVE(Object);
104
105         if (sampleclock->Name) {
106             SOS_StringTable_ElementRemove(
107                 g_NamedClockTable,
108                 sampleclock->Name,
109                 NULL
110             );
111         }
112
113         SOS_Lock_Destroy(sampleclock->Lock);
114         SOS_Mem_Free(sampleclock->Name);
115         SOS_Mem_Free(sampleclock);
116     }
117 }
118
119 /*++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
… +++++
120 Support functions
121 ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
    ++++/
```

```
122 static
123 void
124 FactorReduce(
125     SOS_UINT32    *Numerator,
126     SOS_UINT32    *Denominator
127 )
128 {
129     SOS_UINT32 numerator = *Numerator;
130     SOS_UINT32 denominator = *Denominator;
131     SOS_UINT32 factor=2;
132 
133     while (factor<=numerator && factor<=denominator) {
134         if (!(numerator%factor) &&
135             !(denominator%factor)) {
136             numerator/=factor;
137             denominator/=factor;
138         } else {
139             if (factor>2) factor++;
140             factor++;
141         }
142     }
143 
144     SOS_DEBUGOUT_MINOR_EVENT(
145         "Rationalized %lu/%lu to %lu/%lu\n",
146         *Numerator,
147         *Denominator,
148         numerator,
149         denominator
150     );
151 
152     *Numerator = numerator;
153     *Denominator = denominator;
154 }
155 
156 /*++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
… +++++
157 
158 ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
… +++*/
159 static
160 SOS_STATUS
161 SampleClock_NameSet(
162     SOS_REGOBJECT *            ClockObject,
163     const char *               Name
```

```
165 {
166     SAMPLECLOCK * clock = RESOLVE(ClockObject);
167     SOS_STATUS status;
168     if (!Name || !*Name) {
169         /* clock is unnamed */
170         status = SOS_Success;
171     } else if (clock && !clock->Name) {
172         clock->Name = SOS_strdup(Name);
173         if (clock->Name) {
174             status = SOS_StringTable_ElementPut(
175                 g_NamedClockTable,
176                 clock->Name,
177                 ClockObject
178             );
179         } else {
180             status = SOS_ErrorResourceAllocation;
181         }
182     } else {
183         status = SOS_ErrorParameter;
184     }
185 
186     return status;
187 }
188 
189 static
190 SOS_STATUS
191 SampleClock_GetByName(
192     const char *                Name,
193     SOS_REGOBJECT **            ClockObject
194 )
195 {
196     SOS_STATUS status;
197 
198     status = SOS_StringTable_ElementPeek(
199         g_NamedClockTable,
200         Name,
201         (void**)ClockObject
202     );
203 
204     if (SOS_SUCCEEDED(status)) {
205         SOS_RegObject_Reference(*ClockObject);
206     }
207 
208     return status;
```

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.