```
**********************************************************
   14394 Mon Apr 28 11:07:49 2014
new/usr/src/uts/common/sys/time.h
4809 NANOSEC should be 'long long' to avoid integer overflow bugs
4810 spa_async_tasks_pending suffers from an integer overflow bug
4811 in.mpathd: tv2ns suffers from an integer overflow bug
Reviewed by: Marcel Telka <marcel.telka@nexenta.com>
**********************************************************
______unchanged_portion_omitted_

 221 #define ITIMERVAL_OVERFLOW(itv)                         \
 222         (TIMEVAL_OVERFLOW(&(itv)->it_interval) ||       \
 223         TIMEVAL_OVERFLOW(&(itv)->it_value))

 225 #endif  /* _SYSCALL32 */
 226 #endif  /* _ASM */
 227 #endif /* !defined(__XOPEN_OR_POSIX) || defined(_XPG4_2) ... */


 230 #if !defined(__XOPEN_OR_POSIX) || defined(__EXTENSIONS__)
 231 /*
 232  *      Definitions for commonly used resolutions.
 233  */
 234 #define SEC             1
 235 #define MILLISEC        1000
 236 #define MICROSEC        1000000
 237 #define NANOSEC         1000000000LL
 237 #define NANOSEC         1000000000

 239 #define MSEC2NSEC(m)    ((hrtime_t)(m) * (NANOSEC / MILLISEC))
 240 #define NSEC2MSEC(n)    ((n) / (NANOSEC / MILLISEC))

 242 #endif /* !defined(__XOPEN_OR_POSIX) || defined(__EXTENSIONS__) */

 244 #ifndef _ASM

 246 /*
 247  * Time expressed as a 64-bit nanosecond counter.
 248  */
 249 typedef longlong_t      hrtime_t;

 251 #ifdef _KERNEL

 253 #include <sys/time_impl.h>
 254 #include <sys/mutex.h>

 256 extern int tick_per_msec;       /* clock ticks per millisecond (may be zero) */
 257 extern int msec_per_tick;       /* milliseconds per clock tick (may be zero) */
 258 extern int usec_per_tick;       /* microseconds per clock tick */
 259 extern int nsec_per_tick;       /* nanoseconds per clock tick */

 261 /*
 262  * Macros to convert from common units of time (sec, msec, usec, nsec,
 263  * timeval, timestruc) to clock ticks and vice versa.
 264  */
 265 #define TICK_TO_SEC(tick)       ((tick) / hz)
 266 #define SEC_TO_TICK(sec)        ((sec) * hz)

 268 #define TICK_TO_MSEC(tick)      \
 269         (msec_per_tick ? (tick) * msec_per_tick : (tick) / tick_per_msec)
 270 #define MSEC_TO_TICK(msec)      \
 271         (msec_per_tick ? (msec) / msec_per_tick : (msec) * tick_per_msec)
 272 #define MSEC_TO_TICK_ROUNDUP(msec)      \
 273         (msec_per_tick ? \
 274         ((msec) == 0 ? 0 : ((msec) - 1) / msec_per_tick + 1) : \
 275         (msec) * tick_per_msec)
```



```
 277 #define TICK_TO_USEC(tick)              ((tick) * usec_per_tick)
 278 #define USEC_TO_TICK(usec)              ((usec) / usec_per_tick)
 279 #define USEC_TO_TICK_ROUNDUP(usec)      \
 280         ((usec) == 0 ? 0 : USEC_TO_TICK((usec) - 1) + 1)

 282 #define TICK_TO_NSEC(tick)              ((hrtime_t)(tick) * nsec_per_tick)
 283 #define NSEC_TO_TICK(nsec)              ((nsec) / nsec_per_tick)
 284 #define NSEC_TO_TICK_ROUNDUP(nsec)      \
 285         ((nsec) == 0 ? 0 : NSEC_TO_TICK((nsec) - 1) + 1)

 287 #define TICK_TO_TIMEVAL(tick, tvp) {    \
 288         clock_t __tmptck = (tick);      \
 289         (tvp)->tv_sec = TICK_TO_SEC(__tmptck);  \
 290         (tvp)->tv_usec = TICK_TO_USEC(__tmptck - SEC_TO_TICK((tvp)->tv_sec)); \
 291 }
______unchanged_portion_omitted_
```