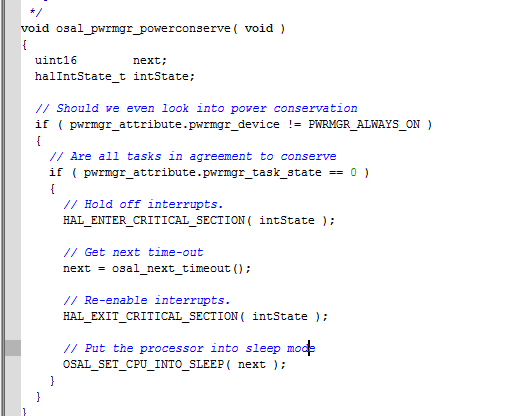
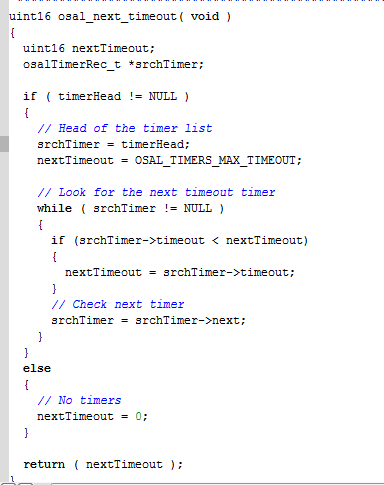
1. 验证节点确实进入休眠，在OSAL\_PwrMgr.c中有函数如下：



其中next就是休眠的时间。osal\_next\_timeout()的定义如下：

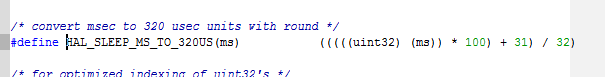


关于最大休眠时间，如下所示：

#define OSAL\_TIMERS\_MAX\_TIMEOUT 0xFFFF

因为这个寄存器有16位，所以最大休眠时间是65535ms，约一分钟多一点。

但是在hal\_sleep.c中调用halSleep(osal\_timeout )的时候又将传入的这个osal\_timeout用HAL\_SLEEP\_MS\_TO\_320US(osal\_timeout);切割了，具体实现如下：



具体切割成多少不明。

所以休眠暂时无法调整。

1. 可以实现”定时发一次包”，具体实现是：

// (setup in SampleApp\_Init()).

if ( events & SAMPLEAPP\_SEND\_PERIODIC\_MSG\_EVT )

{

// Send the periodic message

SampleApp\_SendPeriodicMessage();

// Setup to send message again in normal period (+ a little jitter)

osal\_start\_timerEx( SampleApp\_TaskID, NISL\_ZLC\_COUNT\_EVT,

(SAMPLEAPP\_SEND\_PERIODIC\_MSG\_TIMEOUT + (osal\_rand() & 0x00FF)) );

// return unprocessed events

return (events ^ SAMPLEAPP\_SEND\_PERIODIC\_MSG\_EVT);

}

//Defined by Lichao Zhang

if ( events & NISL\_ZLC\_COUNT\_EVT )

{

if(flag < 3)

{

flag++;

osal\_start\_timerEx( SampleApp\_TaskID, NISL\_ZLC\_COUNT\_EVT,

(SAMPLEAPP\_SEND\_PERIODIC\_MSG\_TIMEOUT + (osal\_rand() & 0x00FF)) );

}

else

{

flag = 0;

osal\_start\_timerEx( SampleApp\_TaskID,

SAMPLEAPP\_SEND\_PERIODIC\_MSG\_EVT,

SAMPLEAPP\_SEND\_PERIODIC\_MSG\_TIMEOUT );

}

return (events ^ NISL\_ZLC\_COUNT\_EVT);

}

其中基本原理是，如果没到约定时间，我就在时间片里面执行等待事件，如果到约定时间，那么就在下个时间片里面发送数据。时间片的长度由我们自己定义。