void HalLedInit (void)

{

#if (HAL\_LED == TRUE)

/\* Initialize all LEDs to OFF \*/

HalLedSet (HAL\_LED\_ALL, HAL\_LED\_MODE\_OFF);

#endif /\* HAL\_LED \*/

#ifdef BLINK\_LEDS

/\* Initialize sleepActive to FALSE \*/

HalLedStatusControl.sleepActive = FALSE;//处于非休眠状态

#endif

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* @fn HalLedSet

\*

\* @brief Tun ON/OFF/TOGGLE given LEDs

\*

\* @param led - bit mask value of leds to be turned ON/OFF/TOGGLE

\* mode - BLINK, FLASH, TOGGLE, ON, OFF

\* @return None

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

uint8 HalLedSet (uint8 leds, uint8 mode)

{

#if (defined (BLINK\_LEDS)) && (HAL\_LED == TRUE)

uint8 led;

HalLedControl\_t \*sts;

switch (mode)

{

case HAL\_LED\_MODE\_BLINK:

/\* Default blink, 1 time, D% duty cycle \*/

HalLedBlink (leds, 1, HAL\_LED\_DEFAULT\_DUTY\_CYCLE, HAL\_LED\_DEFAULT\_FLASH\_TIME);

break;

case HAL\_LED\_MODE\_FLASH:

/\* Default flash, N times, D% duty cycle \*/

HalLedBlink (leds, HAL\_LED\_DEFAULT\_FLASH\_COUNT, HAL\_LED\_DEFAULT\_DUTY\_CYCLE, HAL\_LED\_DEFAULT\_FLASH\_TIME);

break;

case HAL\_LED\_MODE\_ON:

case HAL\_LED\_MODE\_OFF:

case HAL\_LED\_MODE\_TOGGLE:

led = HAL\_LED\_1;//因为它的1是处在最低位，其他的灯的1的位置都可以通过后面的

//左移语句led <<= 1来实现，这样就可以逐个来判断所有的灯是否有操作

leds &= HAL\_LED\_ALL;//保留leds的低四位，高四位清0

sts = HalLedStatusControl.HalLedControlTable;//存放每个LED的状态信息（从HAL\_LED\_1开始）

while (leds)//有LED灯要处理时（低四位有“1”则表示有灯要处理）

{

if (leds & led)//逐个LED灯进行判断，它的mode需要保存在sts->mode里（状态要么是开，要么是关），

{ //对没有操作的等来说，它的sts->mode保持为原理的值

if (mode != HAL\_LED\_MODE\_TOGGLE)

{

sts->mode = mode; /\* ON or OFF \*/

}

else

{

sts->mode ^= HAL\_LED\_MODE\_ON; /\* Toggle \*/

}

HalLedOnOff (led, sts->mode);//要么亮，要么灭

leds ^= led; //操作完了此灯，则它对应位置上的“1”就被会清零，说白了，

} //就是处理完一个灯，就清一个零，直到leds全部变成0，

//这时意味着全部的灯都处理完了

led <<= 1; //处理完了当前的LED，得要判断下一个LED

sts++; //指向下个LED

}

break;

default:

break;

}

#elif (HAL\_LED == TRUE)

LedOnOff(leds, mode);

#else

// HAL LED is disabled, suppress unused argument warnings

(void) leds;

(void) mode;

#endif /\* BLINK\_LEDS && HAL\_LED \*/

return ( HalLedState );// 该变量的值在函数HalLedOnOff被修改，

//它保存了每个LED的状态（灭或者亮）

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* @fn HalLedBlink

\*

\* @brief Blink the leds

\*

\* @param leds - bit mask value of leds to be blinked

\* numBlinks - number of blinks

\* percent - the percentage in each period where the led

\* will be on

\* period - length of each cycle in milliseconds

\*

\* @return None

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void HalLedBlink (uint8 leds, uint8 numBlinks, uint8 percent, uint16 period)//闪烁次数numBlinks=0，

{ //则为无数次闪烁

#if (defined (BLINK\_LEDS)) && (HAL\_LED == TRUE)

uint8 led;

HalLedControl\_t \*sts;//保存每个LED的相关控制信息

if (leds && percent && period)//有LED需要操作并且不能为灭状态

{ //同时闪烁周期不能为0

if (percent < 100)//有亮有灭（即有闪烁）

{

led = HAL\_LED\_1;//为后面从第一个LED开始逐个判断做准备

leds &= HAL\_LED\_ALL;//只保留低四位，高四位清0

sts = HalLedStatusControl.HalLedControlTable;//取出闪烁之前每个LED相关控制信息

while (leds)//有LED需要操作

{

if (leds & led)//leds是从HAL\_LED\_1逐个开始判断，看看leds里有哪些LED需要操作的

{

/\* Store the current state of the led before going to blinking if not already blinking \*/

if(sts->mode < HAL\_LED\_MODE\_BLINK )//如果是关或者是开的状态

preBlinkState |= (led & HalLedState);//HalLedState里面的1表示这个灯当前的状态是亮的，是0表示是灭的

//preBlinkState可以保存所有LED闪烁之前的状态，在函数HalLedUpdate会用到

sts->mode = HAL\_LED\_MODE\_OFF; /\* Stop previous blink \*///停掉前面的闪烁，意味着后面的闪烁是以亮开始的

sts->time = period; /\* Time for one on/off cycle \*///闪烁周期

sts->onPct = percent; /\* % of cycle LED is on \*///亮的百分比

sts->todo = numBlinks; /\* Number of blink cycles \*///闪烁的次数

if (!numBlinks) sts->mode |= HAL\_LED\_MODE\_FLASH; /\* Continuous \*///若为0，即闪烁无数次 0x04

sts->next = osal\_GetSystemClock(); /\* Start now \*///得到当前系统时间

sts->mode |= HAL\_LED\_MODE\_BLINK; /\* Enable blinking \*///允许闪烁 0x02

leds ^= led; //通过将“1”清0，来表示处理完了当前的LED

}

led <<= 1; //判断下一个LED

sts++; //LED的信息指向下一个LED

}

// Cancel any overlapping timer for blink events

osal\_stop\_timerEx(Hal\_TaskID, HAL\_LED\_BLINK\_EVENT); //停掉所有对闪烁事件的定时，因为闪烁要立即执行

osal\_set\_event (Hal\_TaskID, HAL\_LED\_BLINK\_EVENT); //立即去Hal层执行闪烁事件（会调用函数HalLedUpdate），

} //不会等待

else //百分比不小于100%，则表示亮

{

HalLedSet (leds, HAL\_LED\_MODE\_ON); /\* >= 100%, turn on \*/

}

}

else //百分比为0，则表示灭

{

HalLedSet (leds, HAL\_LED\_MODE\_OFF); /\* No on time, turn off \*/

}

#elif (HAL\_LED == TRUE) //没有闪烁，则要么为亮，要么为灭

percent = (leds & HalLedState) ? HAL\_LED\_MODE\_OFF : HAL\_LED\_MODE\_ON;

HalLedOnOff (leds, percent); /\* Toggle \*/

#else

// HAL LED is disabled, suppress unused argument warnings

(void) leds;

(void) numBlinks;

(void) percent;

(void) period;

#endif /\* BLINK\_LEDS && HAL\_LED \*/

}

#if (HAL\_LED == TRUE)

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* @fn HalLedUpdate

\*

\* @brief Update leds to work with blink

\*

\* @param none

\*

\* @return none

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void HalLedUpdate (void)

{

uint8 led;

uint8 pct; //亮或灭的百分比

uint8 leds;

HalLedControl\_t \*sts; //保存每个LED的控制信息

uint32 time; //保存当前系统时间

uint16 next; //保存wait的值

uint16 wait; //保存当前状态（亮或灭）的持续时间

next = 0;

led = HAL\_LED\_1;

leds = HAL\_LED\_ALL; //要处理所有的LED(不管是任何mode)

sts = HalLedStatusControl.HalLedControlTable; //取出LED操作之前的控制信息 HalLedStatusControl.HalLedControlTable;

/\* Check if sleep is active or not \*/

if (!HalLedStatusControl.sleepActive)//如果LED为非休眠状态

{

while (leds)

{

if (leds & led)

{

if (sts->mode & HAL\_LED\_MODE\_BLINK) //mode里有闪烁的话 0x02

{

time = osal\_GetSystemClock();//得到当前系统时间

if (time >= sts->next)//只要晚于前面的系统时间，则表示时间到了

{

if (sts->mode & HAL\_LED\_MODE\_ON)//前一状态为亮，则当前状态就为灭（有可能是闪烁无数次）0x01

{

pct = 100 - sts->onPct; /\* Percentage of cycle for off \*/

sts->mode &= ~HAL\_LED\_MODE\_ON; /\* Say it's not on \*/

HalLedOnOff (led, HAL\_LED\_MODE\_OFF); /\* Turn it off \*/

if (!(sts->mode & HAL\_LED\_MODE\_FLASH))// 在HalLedBlink函数中 if (!numBlinks) sts->mode |= HAL\_LED\_MODE\_FLASH;

{ //当闪烁总次数numBlinks不为0的话，表示不是无数次闪烁

sts->todo--; /\* Not continuous, reduce count \*///这条语句表示闪烁次数减一

}

}

else if ( (!sts->todo) && !(sts->mode & HAL\_LED\_MODE\_FLASH) )//前一状态为灭并且不是无数次闪烁

{ //同时当前已经闪烁结束

sts->mode ^= HAL\_LED\_MODE\_BLINK; /\* No more blinks \*/ //后面就不能再闪烁了

}

else //前一状态为灭且还没闪烁完（或者是无数次闪烁），则当前状态就为亮

{

pct = sts->onPct; /\* Percentage of cycle for on \*/

sts->mode |= HAL\_LED\_MODE\_ON; /\* Say it's on \*/ 0x01

HalLedOnOff (led, HAL\_LED\_MODE\_ON); /\* Turn it on \*/

}

if (sts->mode & HAL\_LED\_MODE\_BLINK) //还没闪完

{

wait = (((uint32)pct \* (uint32)sts->time) / 100); //当前状态（或亮或灭）的持续时间

sts->next = time + wait; //下一时刻是从当前的时刻time再加一个持续时间wait

}

else //当闪烁完了，下一步，灯的状态就要恢复到闪烁前的状态

{

/\* no more blink, no more wait \*/

wait = 0; //既然烁完了，就不需要持续时间了

/\* After blinking, set the LED back to the state before it blinks \*/

HalLedSet (led, ((preBlinkState & led)!=0)?HAL\_LED\_MODE\_ON:HAL\_LED\_MODE\_OFF); //(preBlinkState & led)!=0说明led

//这个灯在闪烁之前的状态是亮的，即是恢复闪烁之前的状态

//preBlinkState变量的值来自于函数HalLedBlink的一执行语句

/\* Clear the saved bit \*/

preBlinkState &= (led ^ 0xFF);//当前的led闪烁完之后，即用语句(preBlinkState & led)回到之前状态，

} //接着为了表示该led的闪烁处理完了，就要将它对应位置上的“1”清0

}

else //若定时时间还未到，则继续等待

{

wait = sts->next - time; /\* Time left \*/

}

if (!next || ( wait && (wait < next) ))

{

next = wait;//下一步要等待next时间才能触发HAL\_LED\_BLINK\_EVENT事件

}

}

leds ^= led; //此LED处理完，则对应的“1”要清0

}

led <<= 1; //判断下一个LED

sts++; //指向下一个LED的控制信息

}

if (next) //若LED的状态需要持续时间，则让这些LED的状态持续next时间

{

osal\_start\_timerEx(Hal\_TaskID, HAL\_LED\_BLINK\_EVENT, next); /\* Schedule event \*///让这些LED的状态持续next时间后转为下一状态

}

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* @fn HalLedOnOff

\*

\* @brief Turns specified LED ON or OFF

\*

\* @param leds - LED bit mask

\* mode - LED\_ON,LED\_OFF,

\*

\* @return none

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void HalLedOnOff (uint8 leds, uint8 mode)

{

if (leds & HAL\_LED\_1)

{

if (mode == HAL\_LED\_MODE\_OFF)

{

HAL\_TURN\_OFF\_LED1();

}

else

{

HAL\_TURN\_ON\_LED1();

}

}

if (leds & HAL\_LED\_2)

{

if (mode == HAL\_LED\_MODE\_ON)

{

HAL\_TURN\_ON\_LED2();

}

else

{

HAL\_TURN\_OFF\_LED2();

}

}

if (leds & HAL\_LED\_3)

{

if (mode == HAL\_LED\_MODE\_ON)

{

HAL\_TURN\_ON\_LED3();

}

else

{

HAL\_TURN\_OFF\_LED3();

}

}

if (leds & HAL\_LED\_4)

{

if (mode == HAL\_LED\_MODE\_ON)

{

HAL\_TURN\_ON\_LED4();

}

else

{

HAL\_TURN\_OFF\_LED4();

}

}

/\* Remember current state \*/

if (mode)

{

HalLedState |= leds;//哪些灯是开的，哪些灯对应就被置为1

} //HalLedState里面的1表示这个灯当前的状态是亮的，是0表示是灭的

else

{

HalLedState &= (leds ^ 0xFF);//哪些灯是关的，就将这些灯对应的1清零

}

}

#endif /\* HAL\_LED \*/