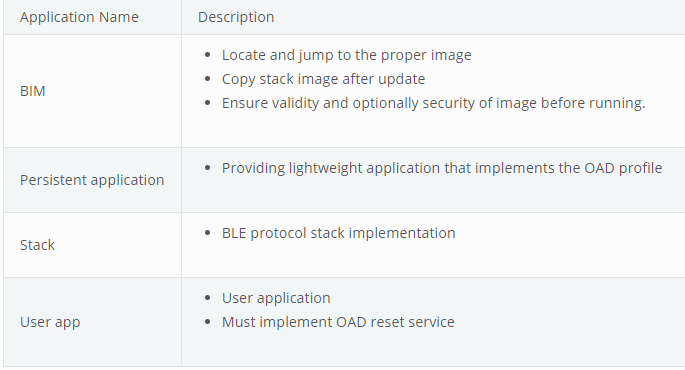
**P&G Tracker项目总结与沉淀**

1.方案选择

本项目平台为CC2640，OAD方案选择片内OAD，flash紧张，故把串口关闭，以节省空间。

2.OAD升级

工程由四部分组成，分别为BIM、Persistent、Stack、User app。



注：

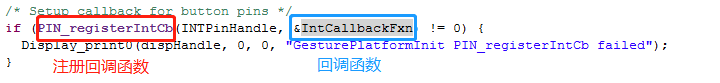
BIM：系统引导程序，开机启动，查找有效的映像，映像有效则拷贝，然后跳转到user app中去执行。

president：进入OAD升级，拥有OAD升级所需的功能。

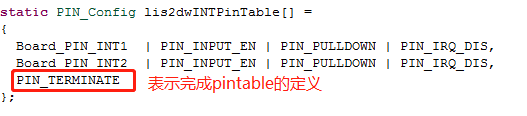
Stack: 协议栈。

User app:用户应用程序，实现用户功能。

3.**PIN\_registerIntCb --- 为一组pin注册回调函数**



4.pintable



5.定时器（clock）事件

(1)创建定时器 Util\_constructClock()



\* @param pClock - pointer to clock instance structure.

\* @param clockCB - callback function upon clock expiration.

\* @param clockDuration - longevity of clock timer in milliseconds

\* @param clockPeriod - if set to a value other than 0, the first

\* expiry is determined by clockDuration. All

\* subsequent expiries use the clockPeriod value.

\* @param startFlag - TRUE to start immediately, FALSE to wait.

\* @param arg - argument passed to callback function.

Util\_constructClock(Clock\_Struct \*pClock, Clock\_FuncPtr clockCB,uint32\_t clockDuration,

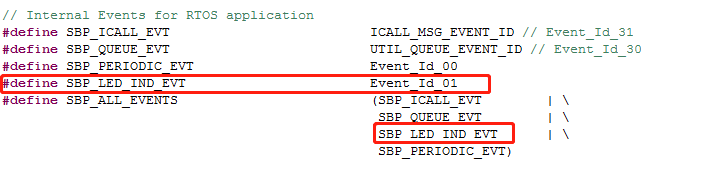
uint32\_t clockPeriod,uint8\_t startFlag,UArg arg);

(2)定义定时器定时周期

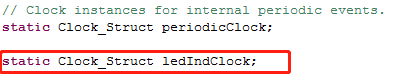


(3)创建定时器事件的优先级

在分配定时器事件的优先级时是按位分配的（协议栈中每个Task用一个16进制数按位代表事件的优先级，共16级）

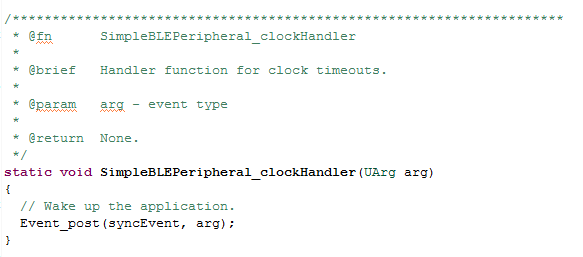


(4)创建clock structure（即一个定时器的数据结构）



(5)定时器超时处理函数

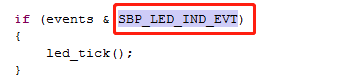
两个定时器可以使用同一个超时处理函数。



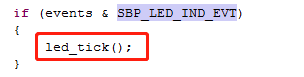
(6)开启定时器 Util\_startClock()

注意需要有各自的中断处理函数。原理是：定时器（属于硬件定时器，定时过程中不占用CPU，不影响协议栈运行）定时到了后设置事件标志位，然后在线程中判断事件标志，事件发生了就处理。PS：flag为false时，为等待状态，必须使用Util\_startClock()函数启动定时器，不然定时器不会工作。flag为true时，立即响应，创建定时器即开启定时器。

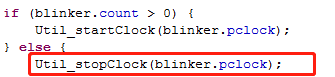
(7)在taskFxn函数中(这里设置为**SimpleBLEPeripheral\_taskFxn**)添加事件



(8)创建定时器中断函数



(9)关闭定时器 Util\_stopClock()



1. 通过更改超时和周期值重新安排时钟 Util\_rescheduleClock()



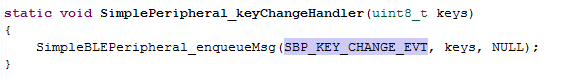
(11)总结

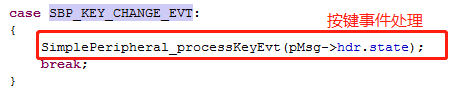
1）创建一个Clock structure 而使用两个不同的事件标志，最终发现只会执行优先级高的，再次证明优先级的用途。

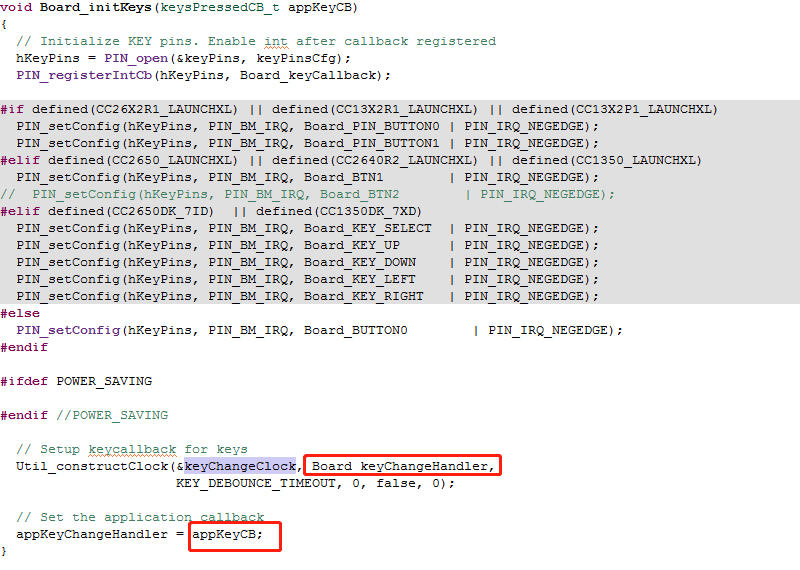
2）定时器定时到了以后，定时器中断函数会立即执行，执行过程中不会被另外一个定时器中断函数中断，直到该中断函数执行完毕，才会执行另外一个。

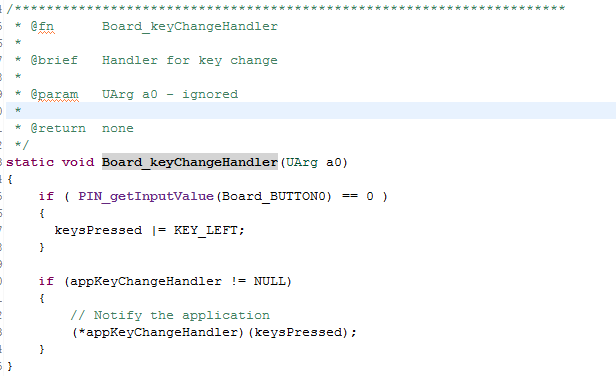
6.协议栈自带的按键驱动





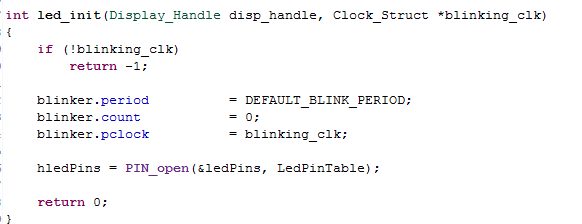




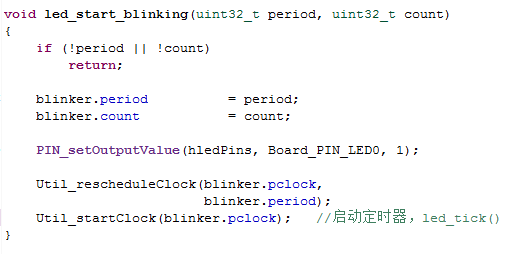


7.led驱动（实现led周期性闪烁---设置每次闪烁的时间和次数）

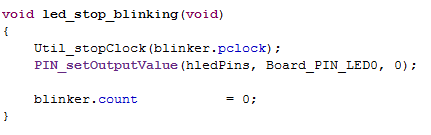
1）初始化



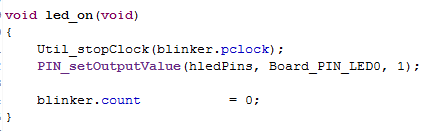
2）设置闪烁次数及其周期



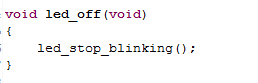
3）停止闪烁



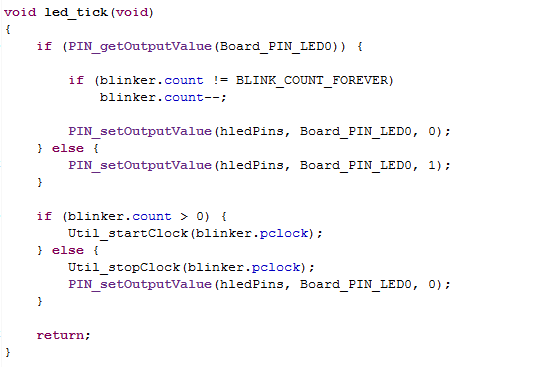
4）打开led



5）关闭led



6）led闪烁



7）驱动程序



8.API函数在peripheral.h中定义

· GAPROLE\_ADVERT\_ENABLED—广播使能。

. GAPROLE\_ADV\_NONCONN\_ENABLED--无连接广播使能

· GAPROLE\_ADVERT\_DATA—包含在广播里的信息。

· GAPROLE\_SCAN\_RSP\_DATA—外设用于回复主机扫描请求的信息。

· GAPROLE\_ADVERT\_OFF\_TIME—表示外设关闭广播持续时间，该值为零表示无限期关闭广播直到下一次广播使能信号到来。

· GAPROLE\_PARAM\_UPDATE\_ENABLE—使能自动更新连接参数，可以让外设连接失败时自动调整连接参数以便重新连接。

· GAPROLE\_MIN\_CONN\_INTERVAL—设置最小连接间隙，缺省值为80个单位（每单位1.25ms）。

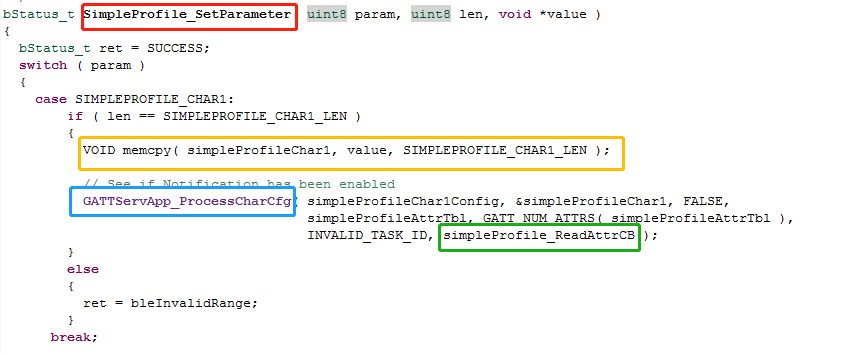
· GAPROLE\_MIN\_CONN\_INTERVAL—设置最大连接间隙，缺省值为3200个单位。

· GAPROLE\_SLAVE\_LATENCY—延时参数，缺省值为零。

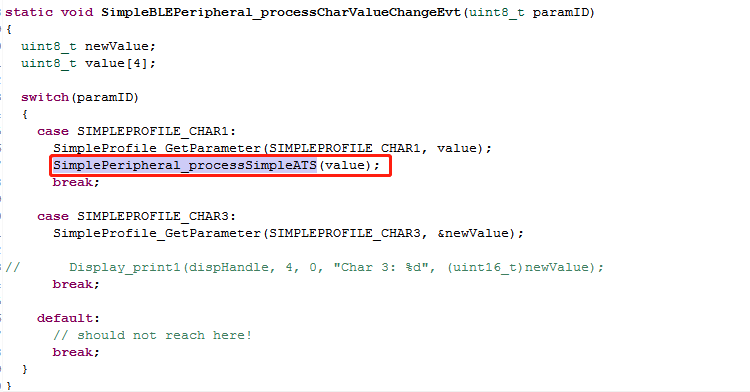
· GAPROLE\_TIMEOUT\_MULTIPLIER--最大耐心等待时间，缺省值为1000个单位。

. GAPROLE\_ADVERT\_ON\_TIME--广播持续时间，超时停止广播

9.simpleGATTProfile.c 里面有个SimpleProfile\_SetParameter(), 这个就是间隔性被调用的函数.里面调用 GATTServApp\_ProcessCharCfg() ,这个函数内部最终会回调simpleProfile\_ReadAttrCB()



应用：APP写入ats指令，app得到产品的响应信息。





10.I2C的读写

