1. 使用涂鸦协议发送一条命令包括下面步骤，会存在着一个状态管理，现在采用的是

Switch case 代码结构来进行编码

1. 启动命令发送
2. 等待设备控制器应答命令,并进行计时，如果超时没有收到命令应答，再次启动命令

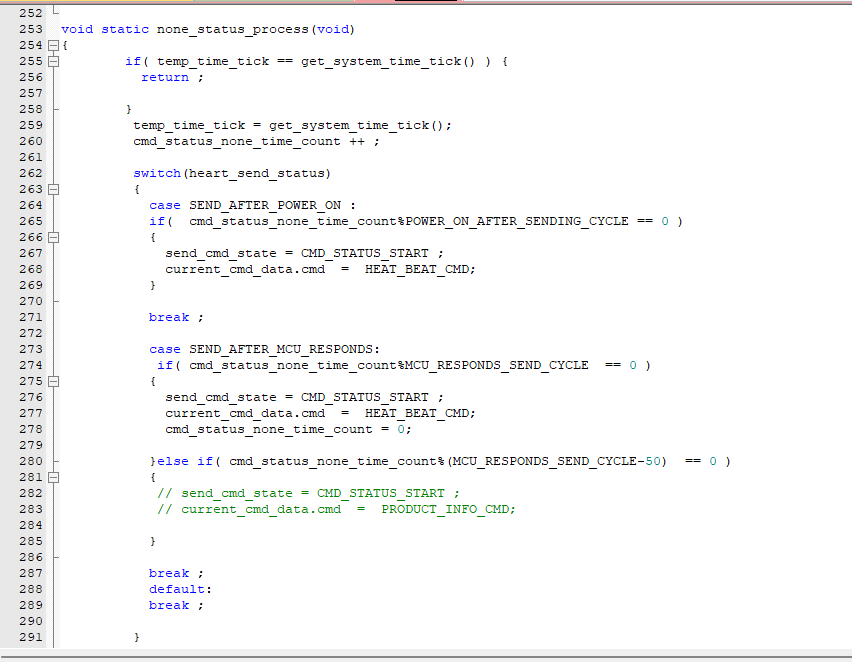
发送，连续发送三次都没有收到应答，认为通讯超时

下图是发送命令的流程

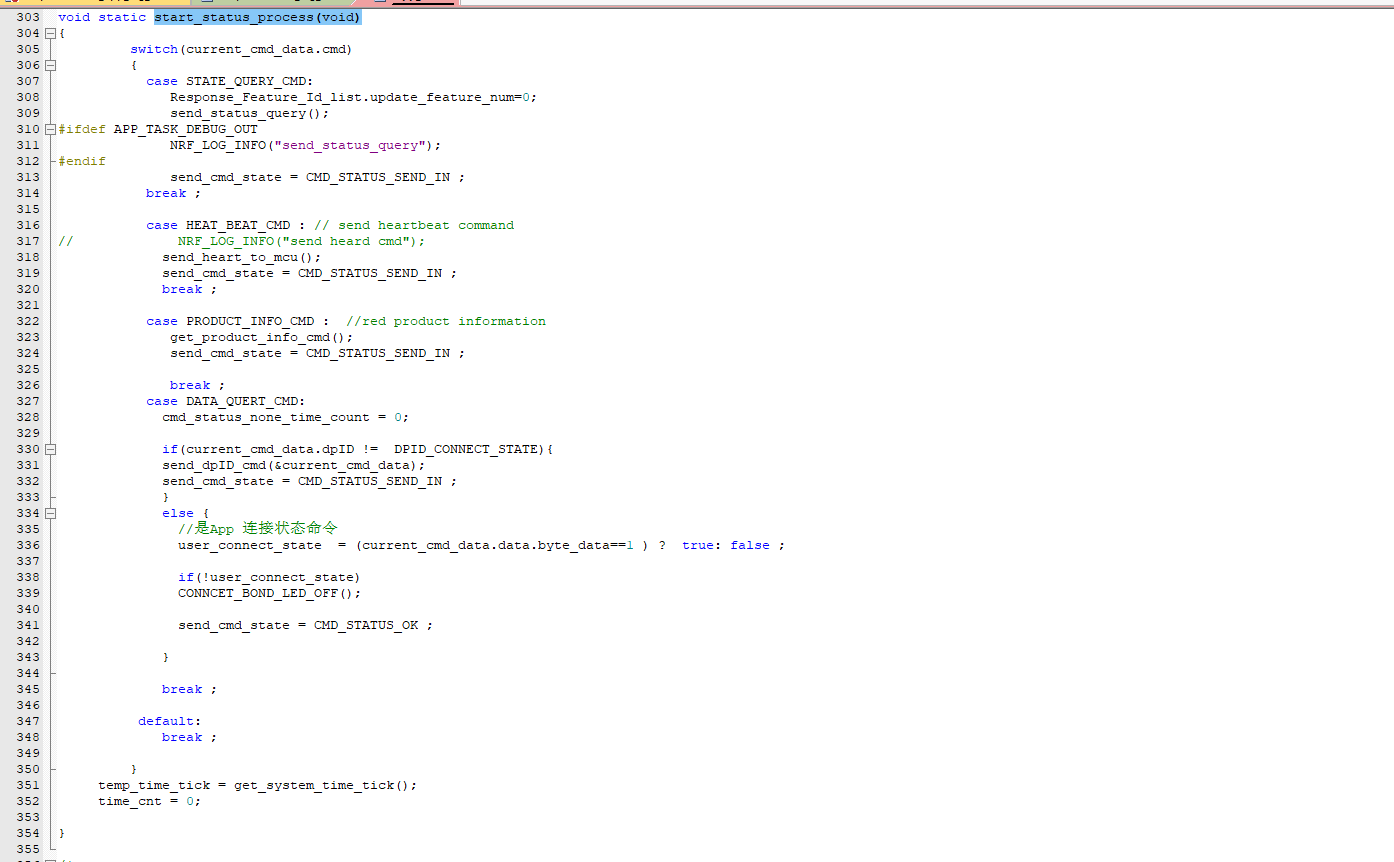


None\_state\_process() 没有用户命令发送时需要处理的流程,

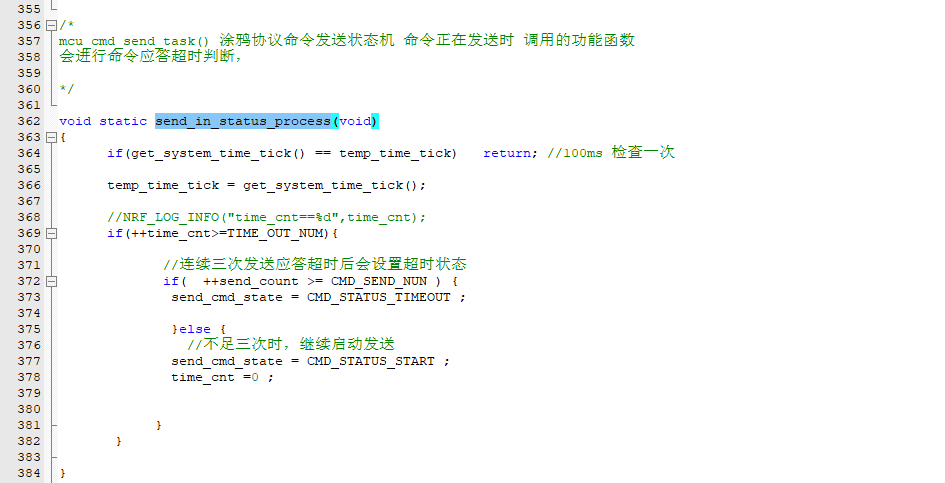
这个流程里包含了心跳处理，这里面其实也包含了一个状态机



start\_status\_process(void) 启动命令发送的处理,这里面对各种不同的命令进行了处理，06设置命令，08查询命令，心跳命令等



send\_in\_status\_process() 命令发送中 ,进行命令超时判断



timeout\_status\_process(void) 对命令超时状态进行了处理,这里面包含了心跳命令,06设置命令，08状态查询命令



ok\_status\_process(void) ，命令应答成功时的处理，针对各种不同的命令进行处理



我现在想使用的是一个轻量极的函数指针状态机框架，想对上面流程进行代码优化，让代码的结构更清晰，方便阅读，也就是对app\_task.c里的代码进行优化

函数指针状态机代码



一个控制指示灯闪烁的例程

