**ZigBee**网络中存在三种逻辑设备类型：Coordinator（协调器）、Router（路由器）、End-Device（终端设备），ZigBee网络由一个协调器以及多个路由器和多个终端设备组成。

协调器：协调器负责启动整个网络，它也是网络的第一个设备。协调器选择一个信道和一个网络ID（Personal Area Network ID，PAN ID），随后启动整个网络。一旦这些都完成后，协调器的工作就像一个路由器。

路由器：路由器的功能主要是允许其他设备加入网络。

终端设备：终端设备一般就是作为终端的数据获取和设备控制，终端设备没有特定的维持网络结构的责任。

**信道**：ZigBee采用直接序列扩频(DSSS)工作在工业科学医疗(ISM)频段，在2.4GHz频段上，IEEE802.15.4/ZigBec规定了16个信道，每个信道频带宽度为5MHz。信道编号为11~26，中心频率2405MHz~2480MHz

PANID：16位的ID值用来标识唯-一个ZigBee网络，主要是用于区分网络，使得同一地区可以同时存在多个ZigBee网络。其取值范围是0x0000~0xFFFF。当设置为0xFFFF时，协调器可以随机获取一个16位的PANID建立一个网络。路由器或者终端设备可以加入任意一个已设定信道上的网络而不去关心PANID。PANID用于在逻辑上区分同一地区或者同一信道上的ZigBee节点，在不同地区或者同一地区不同的信道可以使用同一PANID。

**地址**：ZigBee设备有两种类型的地址。一种是64位IEEE地址，即MAC地址，另一种是16位网络地址（也叫短地址）。64位地址是全球唯一的地址，设备将在它的生命周期中一直拥有它。它通常由制造商或者被安装时设置。这些地址由IEEE来维护和分配。16位地址为网络地址,是当设备加入网络后分配的,协调器按照一定的算法进行分配，它在网络中是唯一的，用来在网络中鉴别设备和发送数据。

**数据传送方式：**

单点传送-->单点传送是指将数据包发送给一个已经知道网络地址的网络设备。

间接传送-->间接传送模式是指当应用程序不知道数据包的目标设备在哪里的时候使用的模式。从发送设备的栈的绑定表中查找目标设备。这种特点称为源绑定。当数据向下发送到达栈中，从绑定表中查找并且使用该目标地址。这样，数据包将被处理成为一个标准的单点传送据包。如果在绑定表中找到多个设备，则向每个设备都发送一个数据包的拷贝。

广播传送-->广播传送模式是指当应用程序需要将数据包发送给网络中的每一个设备时,使用的据传送模式。

**端点**：端点EndPoint是为实现一个设备描述而定义的一组群集，定义了一个设备内的一个通信实体，一个特定应用通过它被执行。ZDO（设备对象）的Endpoint为0，其他应用程序的Endpoint为1~240，241~255保留未用。关于EndPoint的理解就是虚拟链路。

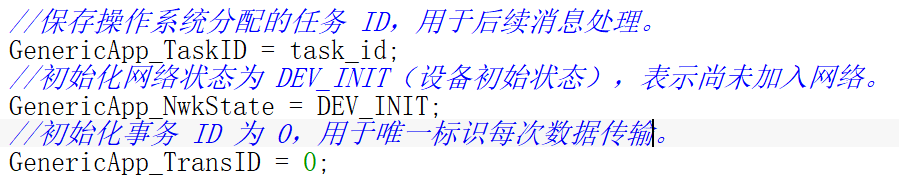
**簇：**一个应用规范内的所有设备，通过簇的方式彼此进行通信。可被输入给一个设备也可从一个设备被输出。簇的作用主要在于发送方和接收方关于通信的一种约定，接收方根据接收到的信息的簇ID来判定要对接收到的信息进行怎样的处理。

要想实现两个终端设备与路由器之间的点对点通信我们需要重新整理一下程序的结构原理。

首先我们的主要文件就是GenericApp.c

在GenericApp.c中我们主要关注两个函数GenericApp\_Init和GenericApp\_ProcessEvent

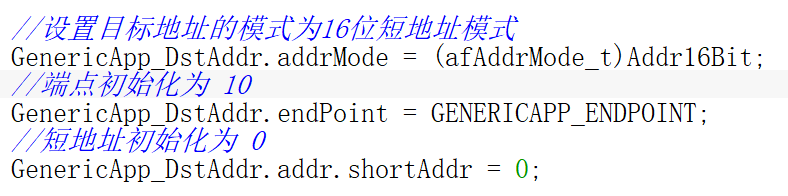
先来看GenericApp\_Init函数



配置串口用于后续信息的打印

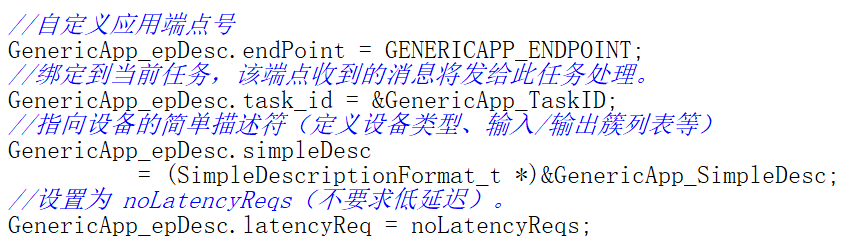


设置目标地址（终端设备）信息，用于后续与目标设备进行通信

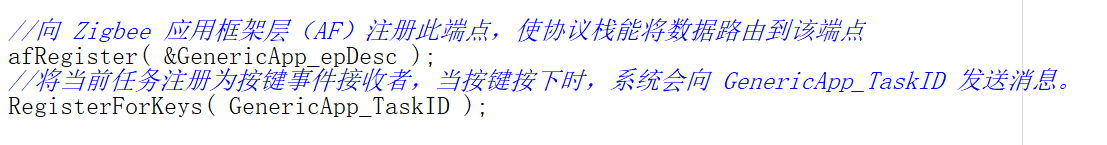


对于自身端点的描述

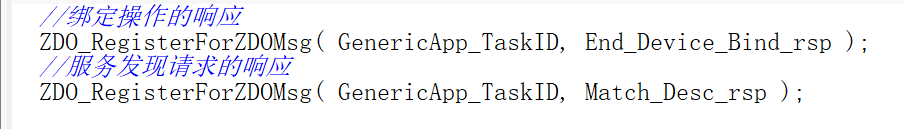
设置自身的端点号，每个端点代表一个独立的应用功能



将自身的端点描述进行注册

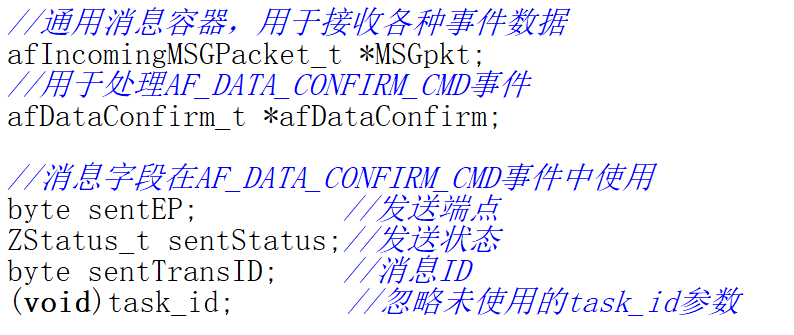


如果端点设备（终端节点）发出绑定请求，或者查询服务的请求，会触发向任务中触发 相应的事件进行处理



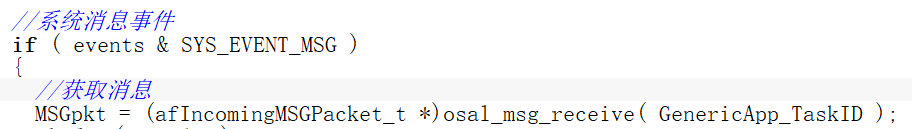
在GenericApp\_ProcessEvent函数中就是对该任务各种各样的事件进行处理，这些事件触发后都会进入该函数。

先定义好之后需要的内容

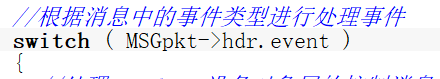


然后判断事件，事件分为两大类，一类是系统消息事件，一类是自己定义的事件（例如之前的LED闪烁）

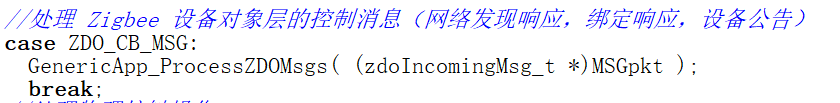
如果是系统消息事件，先获取系统消息，系统消息是一个结构体（结构体中包含了一些系统事件），里面包含了很多内容，不像我们自己定义的事件只是一个常量

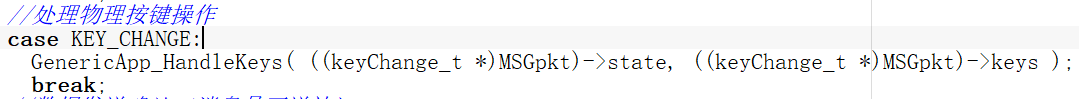


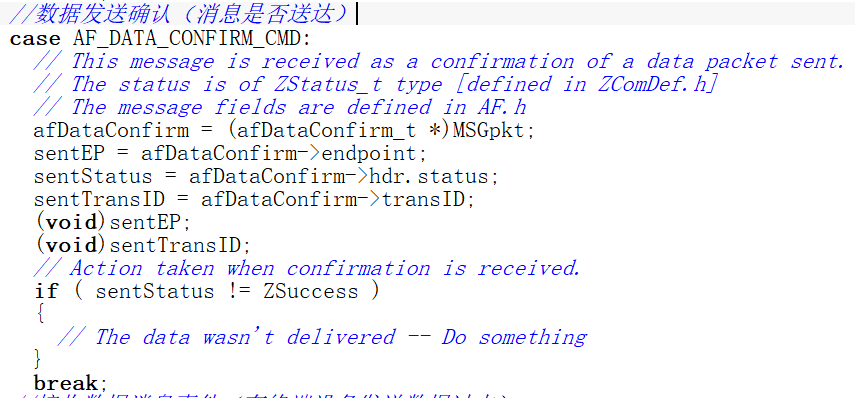
接着根据消息中系统事件的类型执行相应的操作



系统事件有一下几种





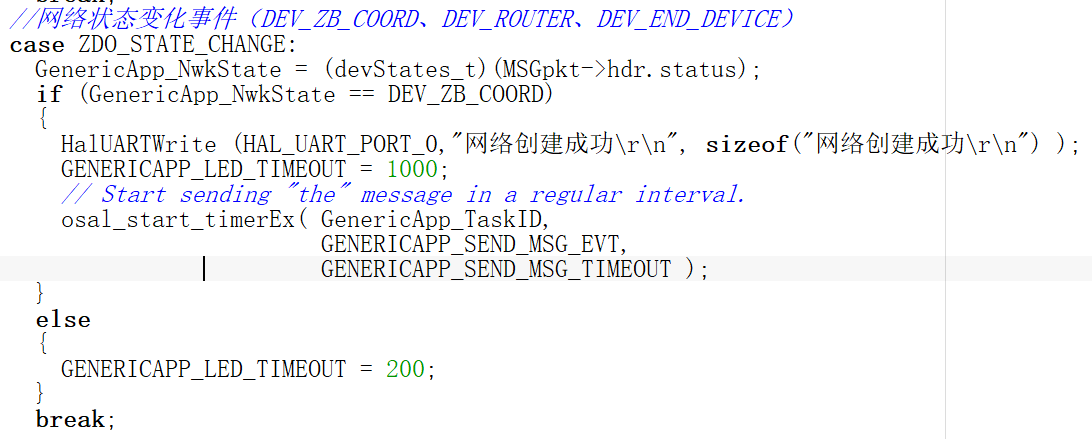


以上三个事件不需要我们进行编译处理，模板默认的处理方式就可以

接收数据事件，也就是说设备收到了其他设备发送来了数据，该事件出发后会区调用数据的接收和解析函数，后续会再次讲解这个函数



网络状态发生改变会触发下面这个事件，对于协调器来说当网络创建启动成功后还会触发该事件，对于终端设备来说当加入网络成功会触发这个事件，在这个事件中我们可以改变LED的闪烁状态和打印相关消息来进行指示



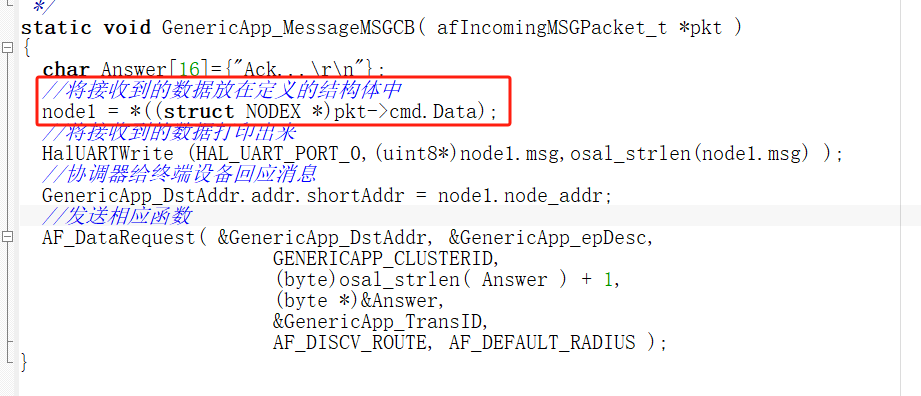
周期性发送数据事件，这个事件是模板给出的一个例子，性质与我们写的LED闪烁事件相同，周期性给终端设备发送数据，这个事件的开启是在网络状态改变事件中，只有网络成功后才会开始周期性触发此事件（当然也可以在其他条件下开始此事件）



再往下就是事件的一些处理函数，我们主要关注两个

第一个是接收数据事件处理函数，在该函数中对其他设备发送来的数据进行接收和解析

在这个函数中的重点就在如何接收这个函数，截图中的红色框内



另外一个就是如何像别的设备发送数据

主要调用了一个函数AF\_DataRequest

这个函数的参数内容为

&GenericApp\_DstAddr目标设备的短地址

&GenericApp\_epDesc源端点的描述（例如协调器向终端设备发送数据，那么该参数就是协调器的端点描述）

GENERICAPP\_CLUSTERID簇ID

byte)osal\_strlen( theMessageData ) + 1发送数据的长度

(byte \*)&theMessageData要发送的数据

&GenericApp\_TransID当前消息ID

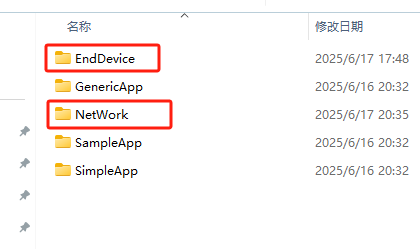
AF\_DISCV\_ROUTE控制数据发送行为（若路由不存在则自动查找路由）

AF\_DEFAULT\_RADIUS限制数据包的最大转发跳数（防止无限循环）

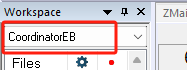
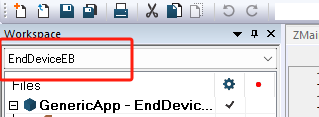


实现终端设备周期性给协调器发送消息，协调器在收到消息后能够给终端设备进行回应

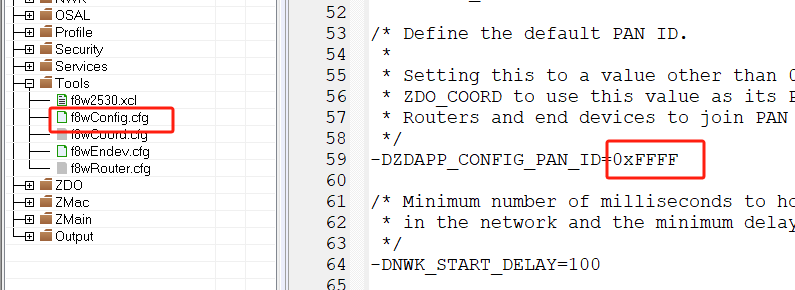
1. 将GenericApp文件夹复制两份，并粘贴到同一路径下，分别重新命名，一个叫做EndDevice作为终端设备的工程，另一个命名为NetWork作为协调器工程使用。



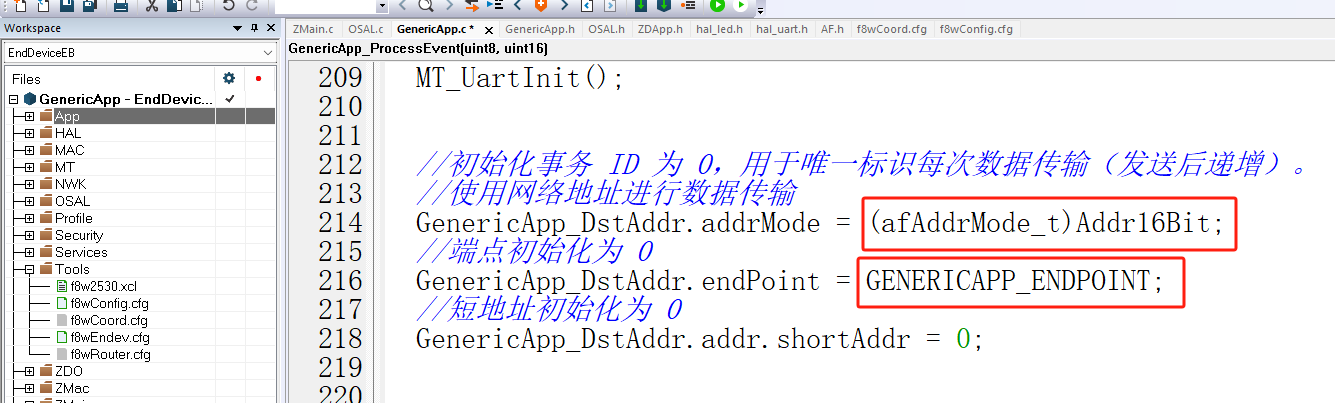
1. 打开这两个工程，更改工作空间，终端设备的工程更改为EndDeviceEB，协调器的工程更改为CoordinatorEB，工作空间不同，相对应的在底层的一些配置也是不同的



1. 更改PANID，防止不同组别之间的数据发送错乱，PANID的更改在Tools分组下的f8wConfig.cfg文件中，每个组的PANID不能相同，例如第1组就将PANID更改为0x0001，注意终端设备工程和协调器设备工程都需要更改否则这两个设备就不在一个网络当中

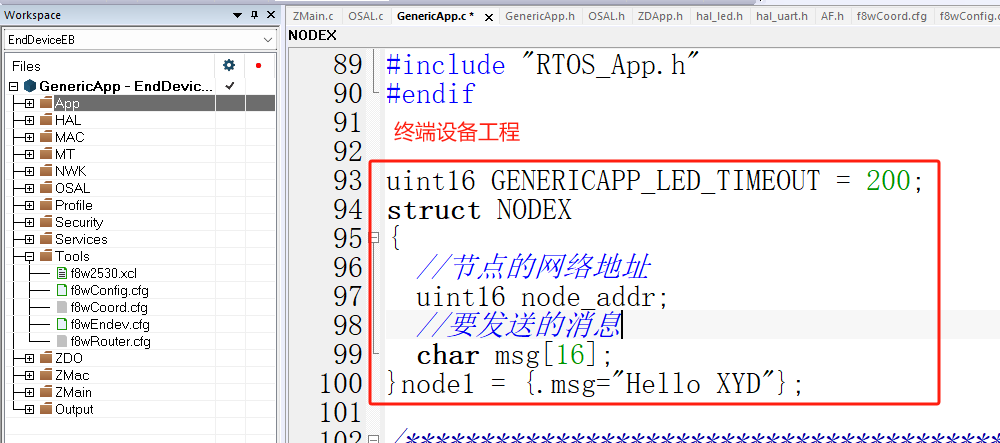


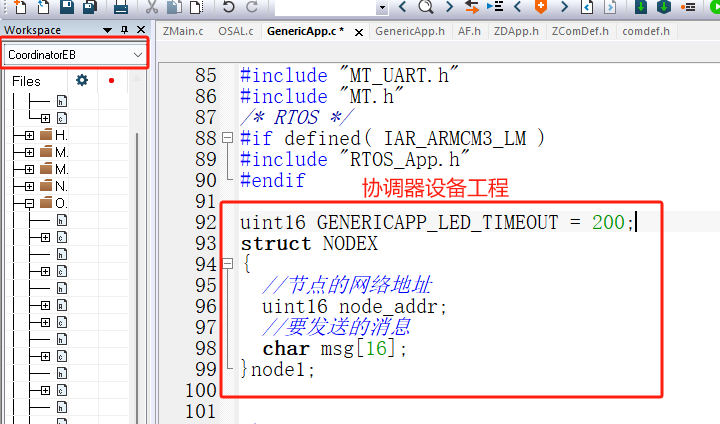
1. 修改两个工程的目标信息（两个工程相同）



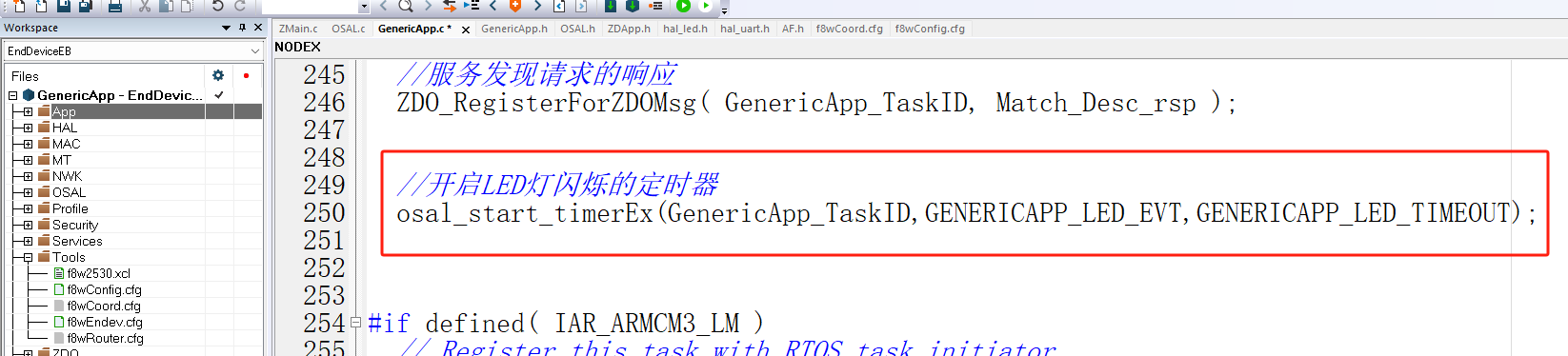
1. 实现LED灯指示网络状态，网络未创建成功或者终端设备未加入网络时，LED灯200ms闪烁一次，网络成功1S闪烁一次。

在两个工程的GenericApp.c中定义一个时间变量用于后续更改，顺便创建一个用于发送和接收数据的结构体。注意两个工程都需要定义和创建

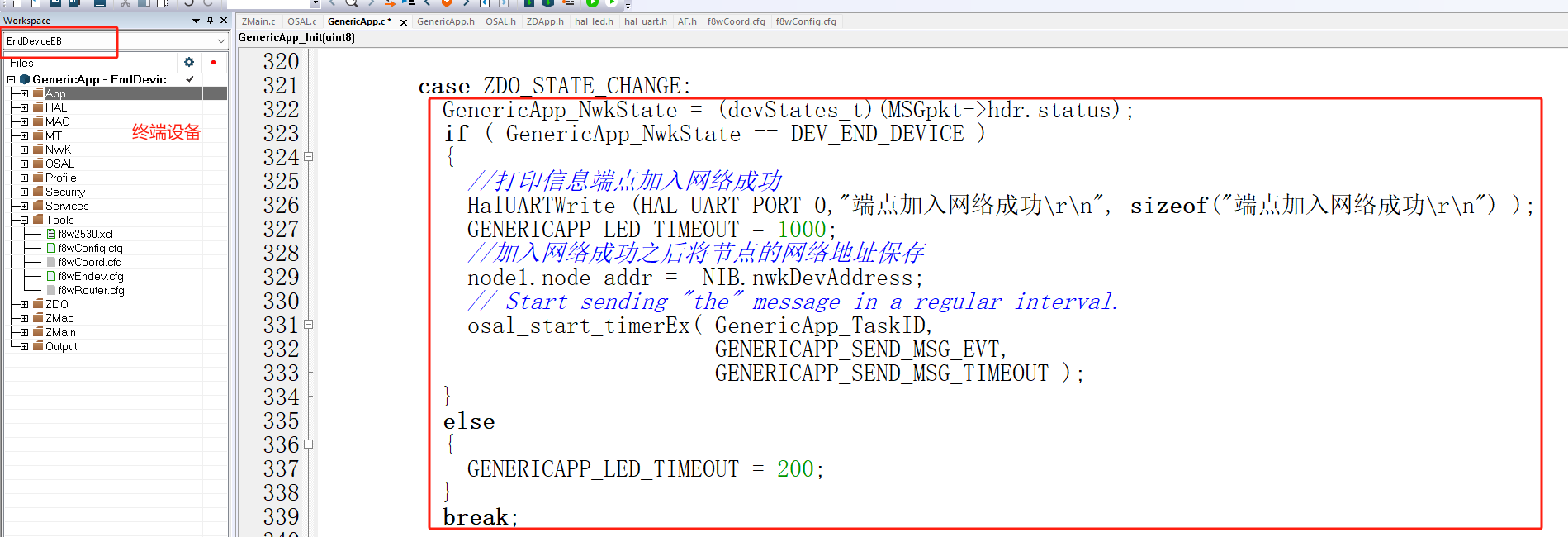


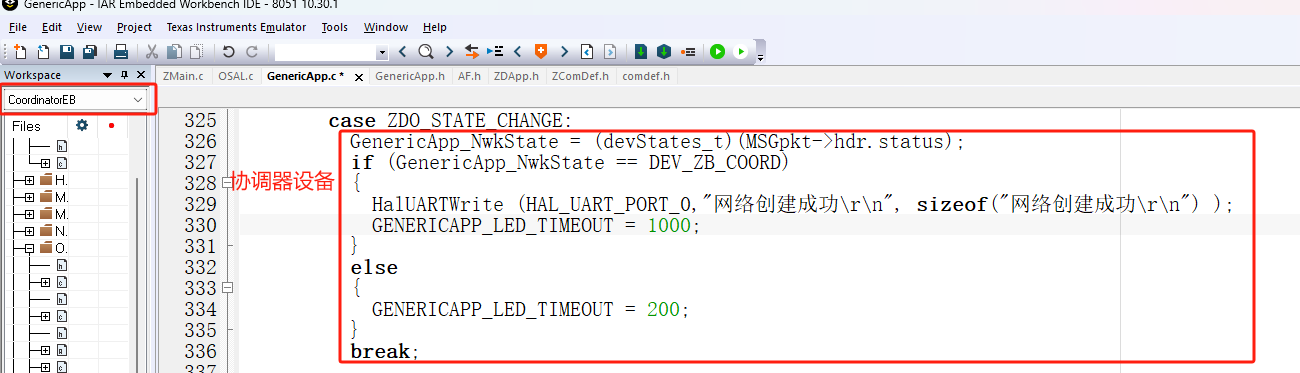


1. 在GenericApp\_Init函数中开启LED闪烁的定时器（注意两个工程都要编写）

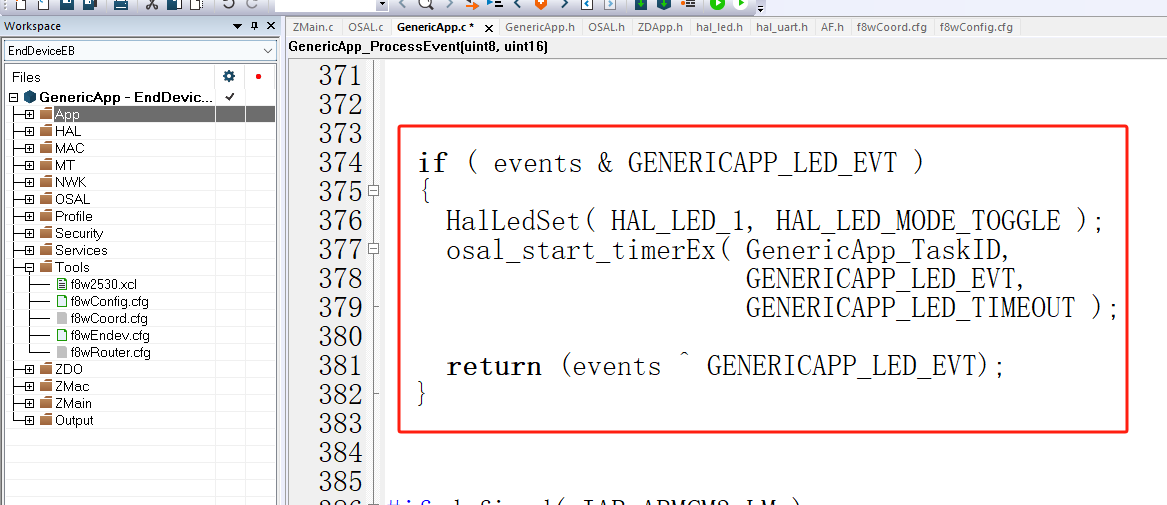


1. 当网络状态发送改变时打印信息并改变LED灯定时器的时间（两个工程不同注意甄别）



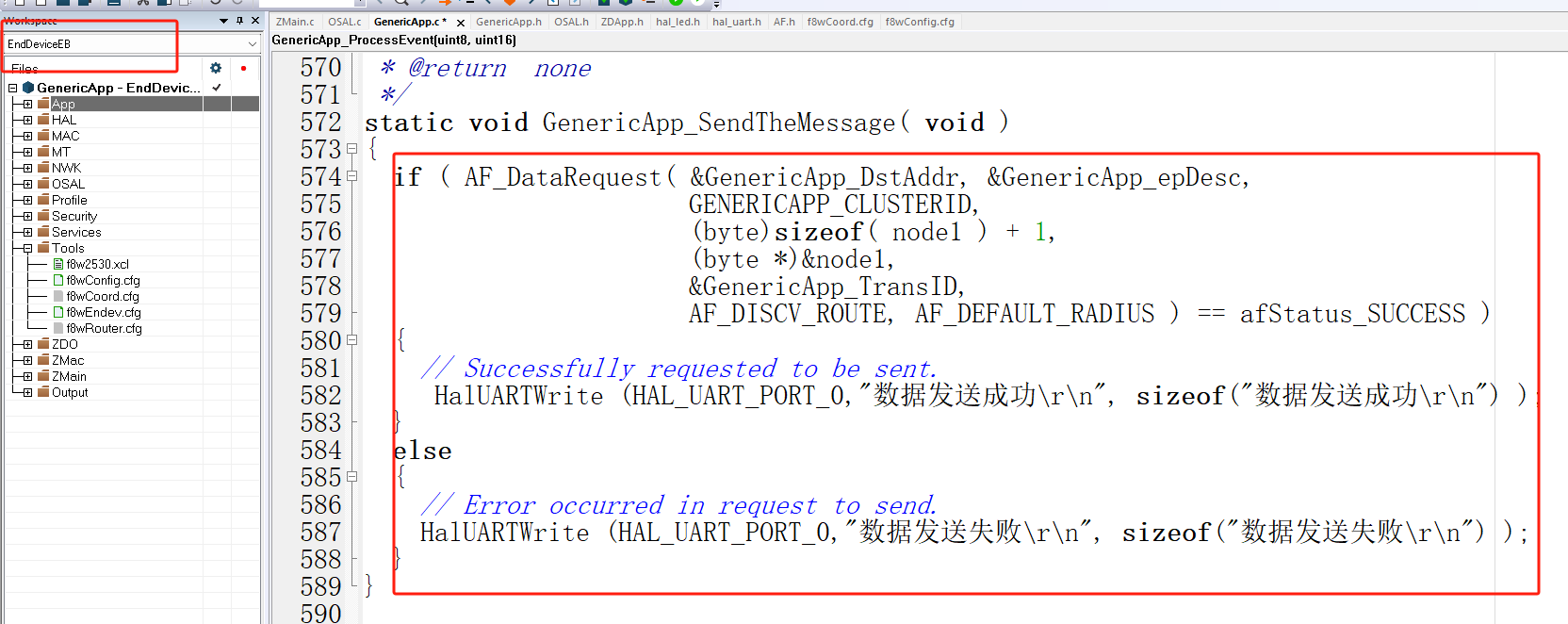


1. 在GenericApp\_ProcessEvent函数中处理LED事件（两个工程相同）

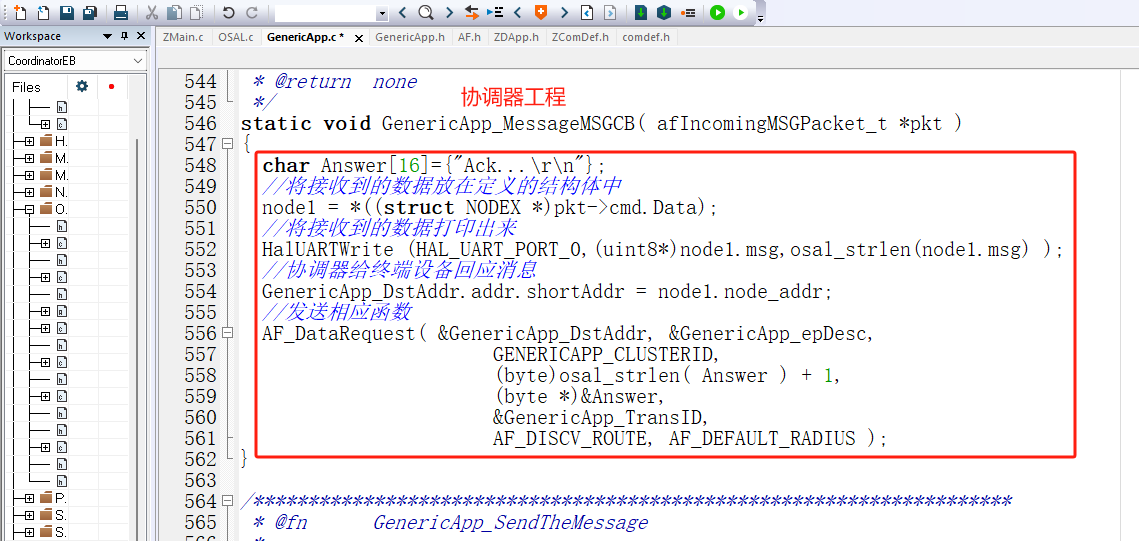


到这里之后将程序编译下载，即可在串口助手上查看到网络连接状态的信息

1. 在终端设备工程中编写数据发送事件代码



1. 在协调器工程中编写数据接收和显示，以及发送响应消息给终端设备代码



1. 编写终端设备接收响应数据并显示代码

