ZigBee 四种绑定方式在 TI Z-Stack 协议栈中应用

KuangJunBin：本文是作者根据 TI Z-Stack 开发文档，ZigBee Specification-2007，《Zigbee Wireless Networking》等英文资料整合和翻译而成，采用中英双语对照方便读者理解，文中翻译不当之处，望广大同行不吝赐教。推广 ZigBee 技术，提高国内电子行业的国际影响力，是我们无线通讯工程师的愿景。本文欢迎转载，请保留作者信息和出处，作为支持我继续努力前行的动力，谢谢！

ZigBee2006 版本中规定，在全部节点中实现绑定机制，并将其称为源绑定。绑定机制允许一个应用服务在不知道目标地址的情况下向对方（的应用服务）发送数据包。发送时使用的目标地址将由应用支持子层从绑定表中自动获得，从而能使消息顺利被目标节点的一个或多个应用服务，乃至分组接收。

Binding Table

绑定表

1. 绑定表存放的位置是内存中预先定义的块，如果编译选项 NV_RESTORE 被激活，也能保存在 Flash 里。
2. 绑定表放置在源节点（需要激活编译选项 REFLECTOR）。
3. 绑定表的条目把需要发送的消息映射到它们的目标地址上。
4. 绑定表中每个条目包括以下内容：
5. 绑定表条目结构体的定义

```
typedef struct
{
uint16 srcIdx; //源地址索引
uint8 srcEP; //源端点
uint8 dstGroupMode; //指定寻址模式
uint16 dstIdx; //目标地址索引或者分组号
uint8 dstEP; //目标端点
uint8 numClusterIds; //在簇标识符表中簇标识符的个数
uint16 clusterIdList[MAX_BINDING_CLUSTER_IDS]; //簇标识符表
}BindingEntry_t;
```

Simple Description --- How to bind devices

概述---怎样绑定节点

绑定指的是两个节点在应用层上建立起来的一条逻辑链路。在同一个节点上可以建立多个绑定服务，分别对应不同种类的数据包。此外，绑定也允许有多个目标节点（一对多绑定）。

举个例子，在一个灯光网络中，有多个开关和灯光设备，每一个开关可以控制一个或以上的灯光设备。在这种情况下，需要在每个开关中建立绑定服务。这使得开关中的应用服务在不知道灯光设备确切的目标地址时，可以顺利地向灯光设备发送数据包。

一旦在源节点上建立了绑定，其应用服务即可向目标节点发送数据，而不需指定目标地址了（调用 zb_SendDataRequest()，目标地址可用一个无效值 0xFFFE 代替）。这样，协议栈将会根据数据包的命令标识符，通过自身的绑定表查找到所对应的目标设备地址。

在绑定表的条目中，有时会有多个目标端点。这使得协议栈自动地重复发送数据包到绑定表指定的各个目标地址。同时，如果在编译目标文件时，编译选项 NV_RESTORE 被打开，协议栈将会把绑定条目保存在非易失性存储器里。因此当意外重启（或者节点电池耗尽需要更换）等突发情况的发生时，节点能自动恢复到掉电前的工作状态，而不需要用户重新设置绑定服务。

配置设备绑定服务，有两种机制可供选择。如果目标设备的扩展地址（64 位地址）已知，可通过调用 zb_BindDeviceRequest()建立绑定条目。如果目标设备的扩展地址未知，可实施一个“按键”策略实现绑定。这时，目标设备将首先进入一个允许绑定的状态，并通过 zb_AllowBindResponse()对配对请求作出响应。然后，在源节点中执行 zb_BindDeviceRequest()（目标地址设为无效）可实现绑定。

此外，使用节点外部的委托工具（通常是协调器）也可实现绑定服务。请注意，绑定服务只能在“互补”设备之间建立。那就是，只有分别在两个节点的简单描述结构体（simple descriptor structure）中，同时注册了相同的命令标识符（command_id）并且方向相反（一个属于输出指令“output”，另一个属于输入指令“input”），才能成功建立绑定。

There are 4 ways to build a binding table:
建立一个绑定表格有四种方法可供选择：

第一种方法：自动绑定

使用函数：ZDP_MatchDescReq()

一、 负责发送消息的设备在网络上广播带有如下参数的“个人公告”（Personal Advertisement）：
（1） 地址，配置文件标识符，簇集合列表；
（2）描述符匹配请求- ZDP_MatchDescReq()。
二、 匹配的设备会作出响应。
三、 由 ZDO 处理和验证响应
四、 负责发送消息的设备建立绑定表并保存绑定记录。
五、 这种方法有时也称“服务发现”，“自动找寻”或者“自动匹配”。

第二种方法：辅助绑定

使用函数：ZDP_BindReq()

ZigBee 设备对象绑定请求-一种告诉目标设备建立绑定记录的委托工具，也称辅助绑定。

任何一个设备或应用服务，都能通过无线信道向网络上的另一个设备发送一个 ZDO 消息，帮助其建立一个绑定记录。这称为辅助绑定，在消息发向的设备上会建立一个绑定条目。

委托绑定的申请：

任一个应用服务，通过 ZDP_BindReq()[defined in ZDProfile.h] 向绑定记录所需要的应用服务入口提供参数（地址和端点）以及簇标识号（cluster ID），即可启动委托绑定的申请。第一个参数（消息发送目标地址）是绑定源节点的短地址（即保存绑定记录的节点地址，这是因为 ZDP 需委托应用框架 AF 辅助实现绑定，如果节点本身是 REFLECTOR，并且希望保存绑定记录，则此消息发送的目标地址就是本地的 AF，这与目标节点的地址 DestinationAddr of Receiving device 不同）。

注意事项：

确保[ZDConfig.h]中 ZDO_BIND_UNBIND_REQUEST 特性已经打开！

你可以通过 ZDP_UnbindReq()（使用相同参数）来移除绑定记录。

被请求辅助绑定的目标设备会返回的 ZDO 申请绑定或者解除绑定的应答消息。此 ZDO 消息会被解析并通过调用 ZDApp_BindRsp() 或 ZDApp_UnbindRsp()告知 ZDApp.c 此次请求的结果。

对于申请绑定的应答消息，从协调器返回的状态可能有 ZDP_SUCCESS,ZDP_TABLE_FULL or ZDP_NOT_SUPPORTED。

对于解除绑定的应答消息，从协调器返回的状态可能有 ZDP_SUCCESS,ZDP_NO_ENTRY or ZDP_NOT_SUPPORTED。

绑定是由外部的设备发起（“外部”的意思是发起绑定的不是绑定的对象之一）。

外部设备应用程序以两个应用服务（地址和端点）和簇标识符作为参数调用 ZDP_BindReq ()发起绑定。第一个参数就是绑定记录保存的设备地址。

确保编译选项 REFLECTOR 已经打开！

函数解析：

ZDP_BindReq()实际上是调用 ZDP_BindUnbindReq()的一个宏。这一调用会产生并发送一个绑定的请求，使得 ZigBee 协调器根据簇标识号 clusterID 对相应的应用服务实施绑定。

函数原型：

afStatus_t ZDP_BindReq(zAddrType_t*dstAddr,byte*SourceAddr,byte SrcEPIntf,byte ClusterID,byte*DestinationAddr,byte DstEPIntf,byte SecuritySuite);

参数细节：

DstAddr-消息发送地址 （负责绑定的设备地址）

SourceAddr–源节点的 64 位 IEEE 地址

SrcEPIntf–源节点应用服务的端点

ClusterID–需要绑定的簇标识符

DestinationAddr–目标节点的 64 位 IEEE 地址

DstEPIntf–目标节点应用服务的端点

SecuritySuite-安全机制模式

返回值：afStatus_t–此函数需要借助 AF 发送（AF_DataRequest()）生成的消息，因此返回值是 AF 状态值。

第三种方法：集中式绑定

使用函数：ZDApp_SendEndDeviceBindReq()

ZigBee 设备对象终端节点绑定请求-两个设备可向协调器告知他们想建立一个绑定表记录。协调器通过安排配对并分别在这两个设备上建立绑定表条目，也称集中式绑定。

这一机制规定在指定的时限内，通过按键或者其他类似动作对指定的设备实施绑定。在规定的时限内，协调器负责收集终端设备绑定请求消息，然后根据相同的配置文件标识号和簇标识号建立相应的绑定表格条目。默认的终端节点绑定时限(APS_DEFAULT_MAXBINDING_TIME)是 16 秒(在 nwk_globals.h 中定义)，若要修改可在 f8wConfig.cfg 中新增数值。

所有例子的应用服务中都有一个响应按键事件的函数（例如，TransmitApp.c 中的 TransmitApp_HandleKeys()）。这一响应函数调用 ZDApp_SendEndDeviceBindReq()[在 ZDApp.c 中]收集该应用服务端点的所有信息，然后再调用 ZDP_EndDeviceBindReq()[在 ZDProfile.c 中]把信息发送给协调器。或者，像 SampleLight 和 SampleSwitch 例程中，按键后直接调用 ZDP_EndDeviceBindReq()，仅把与开关灯函数相关的簇标识号发送出去。

这一消息将会被协调器接收[ZDP_IncomingData()in ZDProfile.c]和解析[ZDO_ProcessEndDeviceBindReq()in ZDObject.c]，然后让回调函数 ZDApp_EndDeviceBindReqCB()[in ZDApp.c]调用 ZDO_MatchEndDeviceBind()[ZDObject.c]处理这一请求。

当协调器接收到第一个绑定请求时，他会在一定的时限内保留这一请求并等待第二个请求的出现。（默认的最长时间间隔是 16 秒）。

一旦协调器接收到两个需要匹配的终端设备绑定请求时，它就会启动绑定过程，为发出请求的设备建立源绑定条目。假设在 ZDO 终端设备绑定请求中找到匹配，协调器将采取以下步骤：

1. 协调器发送一个 ZDO 解除绑定请求给第一个设备。终端设备绑定是一个切换过程，所以解除绑定请求需要发送给第一个设备，以便移除一个已有的绑定条目。
2. 等待 ZDO 解除绑定的应答，如果返回的状态是 ZDP_NO_ENTRY，协调器可以发送一个 ZDO 绑定请求，在源设备（ZDP_EndDeviceBindReq()第一个参数指定的地址）中建立绑定条目。假如此时返回的状态是 ZDP_SUCCESS，可继续处理第一个设备的簇标识符（解除绑定指令已经移除了绑定条目，即已经切换完成）。
3. 等待 ZDO 绑定应答。收到以后，继续处理第一个设备的下一个簇标识符。
4. 等第一个设备完成了以后，在第二个设备上实行同样的过程。
5. 等第二个设备也完成了，协调器向两个设备发送 ZDO 终端设备绑定应答消息。

注意打开编译选项：REFLECTOR 和 ZDO_COORDINATOR

ZDApp_SendEndDeviceBindReq()

优点：

1. 绑定信息保存在网络反射设备（例如协调器、路由器）中，可以节省目标设备的内存空间。
2. 网络反射设备总是处于监听网络的状态。所以，如果其中一个被绑定的节点广播网络地址改变的消息，网络反射设备就可以马上更新相应的绑定表条目。这样，其他被绑定的节点即使处于休眠状态（没有收到该节点网络地址改变的消息），随后向该节点（网络地址已改变）发送的消息，（在）网络反射设备（协助下）仍能准确定位。

缺点：

1. 一个与多个设备绑定的节点不能只向一个或若干个配对的设备发送消息。网络反射设备会向全部已绑定的设备本别发送单播消息。
2. 发送消息的设备无法收到目标设备接收情况的通告。（没有像 AF_ACK_REQUEST 标志位那样返回接收情况的功能！）
3. 所有的消息必须经过网络反射设备传输，降低了网络的带宽。

进一步分析：

与六个设备绑定的某个设备，向网络反射器发送一个消息后，会导致反射器发送六个单播消息。假设一个网络被分成两个相等的地理区域 A 和 B，网络反射器在两区之间的中央。如果发送消息的设备在 A 区的深处，接收消息的（六个）设备在 B 区的深处，那么每次通过绑定（向反射器）发送一个消息，A 区的网络流量将会是对六个接收设备分别发送消息时的六分之一。（这是优点！）但如果发送和接收的设备都邻近在一个区的深处（假设离反射器很远），那么（其中一个设备通过反射器的绑定功能想其他设备发送一个消息）该区的网络流量将会是对六个接收设备分别发送单跳消息的许多倍。（这是缺点！）

第四种方法：应用 API 函数绑定

设备的应用服务-设备上的一个应用服务可以建立或者维护一个绑定表。进入设备上绑定条目的另一种方法是由应用服务本身去管理绑定表。

这意味着应用服务通过调用以下的绑定表管理函数，可以在本地进入或者移除绑定表的条目。

管理绑定表使用的 API：

bindAddEntry()–绑定表中加条目

bindRemoveEntry()–绑定表中移除条目

bindRemoveClusterIdFromList()–从一个已有的绑定表条目中移除一个簇标识符

bindAddClusterIdToList()–在一个已有的绑定表条目中加入一个簇标识符

bindRemoveDev()–移除某目标地址的所有条目

bindRemoveSrcDev()–移除某源地址的所有条目（协调器中）

bindUpdateAddr()–更新条目到新的地址

bindFindExisting()–查找一个绑定条目

bindIsClusterIDinList()–在绑定条目中查找一个已有的簇标识符

bindNumBoundTo()–某一地址（源地址或目标地址）绑定条目的个数

bindNumOfEntries()–绑定表条目的个数

bindCapacity()–允许的最大绑定条目数

BindWriteNV()–在 NV 中保存新的绑定表

Which Binding Method To Use?

我们应该选择哪一种绑定方式？

Automatic 自动绑定的特点：

+no user interaction required 无需用户交互

+no tool cost 没有工具成本

-development time knowledge 开发时，需要知道详细应用？

-non-configurable 不可灵活配置

Assisted 辅助绑定的特点：

+install-time decisions(site-specific knowledge) 安装时决定（具体现场知识）

+analysis,maintenance,modification,visualization 分析，维修，改装，可视化

can be under installers control 可用安装程序控制

-cost of tool 工具成本

Centralized 集中绑定的特点：

+allows user to decide 允许用户决定

+cost of tool minimal 成本最小的工具

-few,if any,configurable parameters 可配置参数很少

-requires a user interface on each device 用户需要介入每个设备

Application 应用 API 函数绑定的特点：

+maximum flexibility 最大的灵活性

-you must write all the code 你必须编写所有代码

来自 TI E2E 社区的进一步讨论：

一、“终端设备绑定请求”这一命名有误导的嫌疑。这一请求不仅仅适用于终端设备，而且适用于对希望在协调器上绑定的两个设备中匹配的簇实施绑定。一旦这个函数被调用，将假设 REFLECTOR 这一编译选项在所有希望使用这一服务的节点中都已经打开。具体操作如下：

（1） (Bind Req) Device 1 --> Coordinator <--- Device 2 (Bind Req)

协调器首先找出包含在绑定请求中的簇，然后对比每一设备的 IEEE 地址，如果簇可以匹配，而且这几个设备没有已经存在的绑定表，那他将发送一个绑定应答给每一个设备。

（2） Device 1 <--- NWK Addr Req ------ Coordinator ------- NWK addr Req ----> Device 2

（3） Device 1 ----> NWK Addr Rsp ---> Coordinator <---- NWK addr Rsp <--- Device 2

（4） Device 1 <----- Bind Rsp <----- Coordinator -----> Bind Rsp ----> Device 2

二、“描述符匹配”为源设备的服务发现提供了一种灵巧的方法。下面是具体的操作，这一过程并没有通过协调器。

（1） Device 1 ----> Match Descriptor request (broadcast or unicast) Device 2

（2） Device 1 <---- Match Descriptor response (if clusters, application profile id match) that includes src endpoint, src address <---- Device 2

1 号设备需要维护一个端点和地址的记录。

许多应用服务最终都会使用第二种方法。

3、绑定

绑定是控制信息从一个应用层到另一个应用层流动的一种机制。在 zigbee06 版本中，绑定机制在所有的设备中被执行。 绑定允许应用层发送信息不需要带目的地址，APS 层确定目的地址从他的绑定表格中，然后在信息前端加上这个目的地址或组。 注意：在 zigbee1.0 版本中，所有绑定条目存储在协调器中。现在所有绑定条目存储在发送数据的设备中。

3.1 绑定一个绑定表格

有三种方式建立一个绑定表格：

ZDO 绑定请求 -- 一个试运转工具能告诉这个设备制作一个绑定报告。

ZDO 终端设备绑定请求 --一 个设备能告诉协调器他们想建立绑定表格报告。该协调器将使协调并在这两个设备上创建绑定表格条目

设备应用 – 在设备上的应用能建立或管理一个绑定表格 。

任何一个设备或应用能在网络中发送一个 ZDO 信息到另一个设备，用来建立一个绑定报告。这是调用绑定帮助并且它将建立一个绑定条目为发送设备。

3.1.1 ZDO 绑定请求

通过调用函数 ZDP_BindReq()（在 ZDProfile.h）发送一个绑定请求。第一个参数（dstAddr）是绑定的源地址的短地址。 这之前应该确定允许绑定，在 ZDConfig.h 文件中有参数[ZDO_BIND_UNBIND_REQUEST]允许绑定。 能用同样的参数调用函数 ZDP_UnbindReq()移除绑定。 目标设备将调用函数 ZDApp_BindRsp()或 ZDApp_UnbindRsp()，反馈绑定或移除绑定的响应，返回其操作状态为 ZDP_SUCCESS, ZDP_TABLE_FULL 或 ZDP_NOT_SUPPORTED.

3.1.2 ZDO 终端设备绑定请求

该机制是用一个按钮按下或其他类似的动作来选择设备在指定时间内被绑定。在规定时间内，该终端设备绑定请求信息被收集到协调器，并创建一个基于模式（profile） ID 和串（cluster） ID 的规定的绑定表格条目。默认的终端设备绑定超时时间（APS_DEFAULT_MAXBINDING_TIME）为 16000（定义在 **ZGlobals.h** 中），但是能被改变

发送绑定请求

在所有的应用例子中有一个处理键盘事件的函数[例如在 TransmitApp.c 文件中的 TransmitApp_HandleKeys()函数]。在该函数中，调用了函数 ZDApp_SendEndDeviceBindReq()[在 ZDApp.c 中]，它将收集应用的终端设备的所有信息并调用函数 ZDP_EndDeviceBindReq() [ZDProfile.c]，发送一个绑定信息到协调器。或者，在 SampleLight 和 SampleSwitch 例子中，直接调用 ZDP_EndDeviceBindReq()函数就实现点亮/关闭灯的功能。 **（严重注意：我在协议里面搜索 TransmitApp_HandleKeys 函数，根本搜索不到？，协议栈似乎没有包含 TransmitApp.c 函数进来）**

接收绑定请求

协调器将接收[ZDP_IncomingData() 在 ZDProfile.c]这些信息并分析处理[ZDO_ProcessEndDeviceBindReq() 在 ZDObject.c]这些信息并调用函数 ZDApp_EndDeviceBindReqCB() [in ZDApp.c]，它将调用 ZDO_MatchEndDeviceBind() [ZDObject.c]处理这个请求，当协调器接收到 2 个匹配终端色后备的绑定请求时，它将启动在绑定设备上创建源绑定条目的处理过程。该协调器有如下处理过程：

解除绑定

1. 发送一个 ZDO 解除绑定请求到第一个设备。终端设备绑定切换处理，所以解除绑定首先被发送到移除一个存在的绑定条目。
2. 等待 ZDO 解除绑定响应，如果响应状态为 ZDP_NO_ENTRY，发送一个 ZDO 绑定请求，在源设备上制作一个绑定条目。如

果该响应为 ZDP_SUCCESS, 为第一个设备继续到 move on to the cluster ID for the first device (the unbind removed the entry – toggle)。

3. 等待 ZDO 绑定响应. When received, move on to the next cluster IDfor the first device。

4. 当第一个设备完成时，对第二个设备做同样的处理。

5. 当第二个设备完成时，发送 ZDO 终端设备绑定响应信息到第一个和第二个设备

3.1.3 设备应用绑定管理

在设备上其他进入绑定条目的方式是应用层管理绑定表格。 意思是说，应用层将调用下列函数进入和移除绑定表格条目：

bindAddEntry() ——增加绑定表格条目

bindRemoveEntry() —— 从绑定表格中移除条目

bindRemoveClusterIdFromList() —— 从一个存在的绑定表格项目中移除一个串 ID 。

bindAddClusterIdToList()——向一个已经存在的绑定记录中增加一个群 ID

bindRemoveDev()——删除所有地址引用的记录

bindRemoveSrcDev()——删除所有源地址引用的记录

bindUpdateAddr()——将记录更新为另一个地址

bindFindExisting()——查找一个绑定表记录

bindIsClusterIdInList()——在表记录中检查一个已经存在的群 ID

bindNumBoundTo()——拥有相同地址(源或者目的)的记录的个数

bindNumEntries()——表中记录的个数

bindCapacity()——最多允许的记录个数

bindWriteNV()——在 NV 中更新表

3.2 配置源绑定

允许绑定源的编译选项 REFLECTOR 在 f8wConfig.cfg 文件中。在文件 f8wConfig.cfg,中查看这两个绑定配置参数（DNWK_MAX_BINDING_ENTRIES & DMAX_BINDING_CLUSTER_IDS)。DNWK_MAX_BINDING_ENTRIES 绑定表格中最大的绑定实体数量参数；DMAX_BINDING_CLUSTER_IDS 是在每个绑定实体中最大的串 ID 数量。 绑定表在静态 RAM 中(未分配)，因此绑定表中记录的个数，每条记录中群 ID 的个数都实际影响着使用 RAM 的数量。每一条绑定记录是 8 字节多(MAX_BINDING_CLUSTER_IDS * 2 字节)。除了绑定表使用的静态 RAM 的数量，绑定配置项目也影响地址管理器中的记录的个数。