ZigBee 地址模式分析 (2010-12-28 16:15)
分类： ZigBee 技术学习

我们知道 ZigBee 设备有两种地址。一种是 64 位 IEEE 地址（物理），即 MAC 地址，另一种是 16 位网络地址。

64 位地址使全球唯一的地址，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。我们刚买到的调和上的 IEEE 地址应该是全部的 F,我们可以通过 TI 的软件 SmartRF Flash Programmer 重新写入一个 IEEE 地址，这就像我们的 PC 上的物理地地类似，在全球范围内物理地址是唯一的。不过在 ZigBee 设备中我们也可以更改这个地址，其实也就不确保全球唯一了，当然，在 PC 上也可以通过软件更改物理地址，不过只要在一个局域网中没有两个相同的物理地址，是一样可以连接互联网。很多学校里的上网帐号就是和物理地址进行绑定的，分配给一台 PC 上的 IP 地址，是不可能在另一个 PC 上使用，除非修改 PC 的物理地址。说多了，其实也就是你应该必须保证在你组成的网络中，不可以写入相同的 IEEE 地址。

16 为网络地址是当设备加入网络后由协调器或路由器分配的。它在网络中是唯一的，用来在网络中鉴别设备和发送数据。在这个版本的中 Z-Stack-1.4.3-1.2.1 中，可以看到下面

```
typedef enum
{
afAddrNotPresent = AddrNotPresent, //绑定
afAddr16Bit = Addr16Bit, //短地址
afAddrGroup = AddrGroup, //组发送
afAddrBroadcast = AddrBroadcast //广播发送
} afAddrMode_t;

typedef struct
{
union
{
uint16 shortAddr;
} addr;
afAddrMode_t addrMode;
byte endPoint;
} afAddrType_t;
```

这里的地址模式只有 4 种，而没有 IEEE 地址的什么事，不过可以想像，在一个 ZigBee 网络中，当 ZigBee 的协调器建立网络成功以后，终端设备或者路由器设备打开电源开关加入网络时，网络成功后协调如何知道它管辖的表具号呢？这时会分配一个 16 位的短地址，当加入网络时终端会发送一个响应给协调器，协调器的短地址是知道的因为是 0x0000，当设备加入成功后，会产生一个 ZDO_STATE_CHANGE_EVT 事件，这个事件就是设备加入网络成功后，并在网络中的身份确定后产生的一个事件，我们可以在这里处理，一些初始化，比如可

以发送终端的短地址，IEEE 地址等，这里协调器接收到以后，可以提取出终端的短地址，其实在终端给协调器发送的每个数据包中，都含有其自身的短地址，如下面的结构体当中的 afAddrType_t srcAddr;协调器在接收到短地址后，就可以知道自己下面管辖的终端结点，或者路由结点有那些了，协调器提取到的短地址可以存放到一个非易失性的存储器中，这时如果要使用直接地址模式，也就是单播的话，就可以在这个表中查找，当然这要和上位机的软件结合起来，起码应该让用户看到这些短地址，这种短地址再和某种应用中的属性对应起来。不知道这种理解的正确性有多大，更确切的说是在实际中的可行性有多大？

```
typedef struct
{
osal_event_hdr_t hdr;
uint16 groupId;
uint16 clusterId;
afAddrType_t srcAddr;
byte endPoint;
byte wasBroadcast;
byte LinkQuality;
byte SecurityUse;
uint32 timestamp;
afMSGCommandFormat_t cmd;
} afIncomingMSGPacket_t;
extern uint8 APSME_LookupNwkAddr( uint8* extAddr, uint16* nwkAddr );
```

参数：

nwkAddr -拥有的短地址，用来查找扩展地址。

extAddr -指向扩展地址存放的缓存

该函数知道扩展的 IEEE 地址，得到网络中的短地址，这里就提供了另一种获得终端设备的短地址的方式，可以这个函数，在网上看到的是，用 APSME_LookupNwkAddr 得到短地址时，只能查找协调器儿子节点，对孙子节点不可访问，对限制了一些应用。不过，可以用绑定来解决这个问题。

```
extern uint8 APSME_LookupExtAddr( uint16 nwkAddr, uint8* extAddr );
```

该函数知道网络地址，得到网络中的扩展地址。

上面两个函数是，快速查询（不启动无线查询，而是根据已存储于地址管理器中的网络（物理）地址查询物理（网络）地址）：

```
/*********************************************************************
*
* @fn ZDP_NwkAddrReq
*
* @brief This builds and send a NWK_addr_req message. This
* function sends a broadcast message looking for a 16
* bit address with a 64 bit address as bait.
*
* @param IEEEAddress - looking for this device
* @param SecurityEnable - Security Options
*
```

```
* @return afStatus_t
*/
afStatus_t ZDP_NwkAddrReq( byte *IEEEAddress, byte ReqType,
byte StartIndex, byte SecurityEnable )
```
根据已知网络地址查询远程设备物理地址，作为一个广播信息发送给网络中的所有设备：这个函数也是知道 IEEE 地址，对短地址进行寻找，这个不存在上面所说的限制，
```
/*********************************************************************
*
* @fn ZDP_IEEEAddrReq
*
* @brief This builds and send a IEEE_addr_req message. This
* function sends a unicast message looking for a 64
* bit IEEE address with a 16 bit address as bait.
*
* @param ReqType - ZDP_IEEEADDR_REQTYPE_SINGLE or
* ZDP_IEEEADDR_REQTYPE_EXTENDED
* @param SecurityEnable - Security Options
*
* @return afStatus_t
*/
afStatus_t ZDP_IEEEAddrReq( uint16 shortAddr, byte ReqType,
byte StartIndex, byte SecurityEnable )
```
这个函数，知道短地址，获取网络中的 IEEE 地址。

这个函数，在绑定时经常会使用，因为在绑定中有两种方式，一种是知道 IEEE 地址，另一种是未知 IEEE 地址的绑定。这里会使用到 ZDP_NwkAddrReq() 函数。这两个函数的返回值并不是得到的地址值，而是状态值，返回的地址值应该是存放在了全局变量中，
```
static uint8 ZDP_Buf[ ZDP_BUF_SZ ];
static uint8 *ZDP_TmpBuf = ZDP_Buf+1;
```
应该说在一个网络中 IEEE 地址是可以事先知道的，可以通过 TI 的软件 SmartRF Flash Programmer 进行读取，当知道了 IEEE 地址后我们就可以单播发送到一个指定的终端了。这样就不需要终端主动上报给协调器了。应该利用绑定是最灵活的一种方式，不过这种方式在一个网络中具体的应用还有一点模糊，也是下一个要解决的问题。在 Z-Stack-1.4.2--1.1.0 中还有这么一个函数
```
#define ZDApp_AutoFindDestination( endPoint )
ZDApp_AutoFindDestinationEx( endPoint, (uint8 *)0 )
```
自动寻找目的设备，而在 Z-Stack-1.4.3-1.2.1 中，这两个函数都已经找不到了。这里还有另外几个比较特殊的网络地址，

0xFFFF -这个一个对全网络中设备进行广播的广播地址

0xFFFD -如果在命令中将目标地址设为这个地址的话那么只对打开了接收的设备进行广播

0xFFFC -广播到协调器和路由器

0xFFFE -如果目的地址为这个地址的话，那么应用层将不指定目标设备，而是通过协议栈读取绑定表来获得相应目标设备的短地址

此外的 0x0000 到 0xFFF8 都是有效的目的地址。每一个地址就只是了一个目标设备。

1．单点传送(Unicast)

Uicast 是标准寻址模式，它将数据包发送给一个已经知道网络地址的网络设备。将 afAddrMode 设置为 Addr16Bit 并且在数据包中携带目标设备地址。也就是上面说的使用网络短地址进行发送。这里的关键应该是在网络形成后，如何获得网络中的设备短地址。

2. 间接传送(Indirect)

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。将模式设置为 AddrNotPresent 并且目标地址没有指定。取代它的是从发送设备的栈的绑定表中查找目标设备。这种特点称之为源绑定。

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数据包。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。

上一个版本的 ZigBee(ZigBee04)，有一个选项可以讲绑定表保存在协调器(Coordinator)当中。发送设备将数据包发送给协调器，协调器查找它栈中的绑定表，然后将数据发送给最终的目标设备。这个附加的特性叫做协调器绑定(Coordinator Binding)。这个版本中的绑定有四种方式，在上几篇文章中也有介绍。绑定的方式应该比使用短地址进行发送来的更灵活一些，TI 的几个例子也都已经调试成功，其中有两个例程都是使用的绑定方式。上面有篇文章也重点分析了两种绑定例子。其中一种是经过协调器的绑定，一种是不经过协调器的绑定。

3 广播传送(broadcast)

当应用程序需要将数据包发送给网络的每一个设备时，使用这种模式。地址模式设置为 AddrBroadcast。目标地址可以设置为下面广播地址的一种：

NWK_BROADCAST_SHORTADDR_DEVALL(0xFFFF)——数据包将被传送到网络上的所有设备，包括睡眠中的设备。对于睡眠中的设备，数据包将被保留在其父亲节点直到查询到它，或者消息超时(NWK_INDIRECT_MSG_TIMEOUT 在 f8wConifg.cfg 中)。

NWK_BROADCAST_SHORTADDR_DEVRXON(0xFFFD)——数据包将被传送到网络上的所有在空闲时打开接收的设备(RXONWHENIDLE)，也就是说，除了睡眠中的所有设备。

NWK_BROADCAST_SHORTADDR_DEVZCZR(0xFFFC)——数据包发送给所有的路由器，包括协调器。

这种发送的话，就和很多的无线发送方式是一样的，现在的 CC1110 芯片就是采用的广播发送，数据发送出去后，在射频范围之内所以的 CC1110 无线芯片都可以接收到。上面几种特殊的网络地址，在上面也有简单的介绍。

4 组寻址(Group Addressing)

当应用程序需要将数据包发送给网络上的一组设备时，使用该模式。地址模式设置为 afAddrGroup 并且 addr.shortAddr 设置为组 ID。

在使用这个功能呢之前，必须在网络中定义组。(参见 Z-stack API 文档中的 aps_AddGroup()函数)。

注意组可以用来关联间接寻址。再绑定表中找到的目标地址可能是是单点传送或者是一个组地址。另外，广播发送可以看做是一个组寻址的特例。

下面的代码是一个设备怎样加入到一个 ID 为 1 的组当中：

```
aps_Group_t group;
// Assign yourself to group 1
group.ID = 0x0001;
group.name[0] = 0; // This could be a human readable string
aps_AddGroup( SAMPLEAPP_ENDPOINT, &group );
```

在 SampleApp 例程中我们可以看到下面关于组的定义。

```
SampleApp_Group.ID = 0x0001;
osal_memcpy( SampleApp_Group.name, "Group 1", 7 );
aps_AddGroup( SAMPLEAPP_ENDPOINT, &SampleApp_Group );
```

将组加入到组列表中，其中参数 SAMPLEAPP_ENDPOINT 表示接收发级该组的信息时通过的端口。

定义目的地址 Sample_GrpDstAddr.在发送节点中初始化函数中设置目的地址的地址模式和短地址

```
Static afAddrType_t Sample_GrpDstAddr;
Sample_GrpDstAddr.addrMode = (afAddrMode_t)afAddrGroup;
Sample_GrpDstAddr.endPoint =SAMPLEAPP_ENDPOINT //端口
Sample_GrpDstAddr.addr.shorAddr = SampleApp_Group.ID //组 ID
```

调用下面的函数，给属于 SampleApp_Group 的设备发送数据

```
AF_DataRequest( & Sample_GrpDstAddr, &SampleApp_epDesc,
SAMPLEAPP_FLASH_CLUSTERID,
3,
buffer,
&SampleApp_TransID,
AF_DISCV_ROUTE,
AF_DEFAULT_RADIUS ) == afStatus_SUCCESS )
```

这样发送数据的方式也是基本就这些，不管是那种设备都会使用这四种中的其中之一发送数据。其实在这中间绑定的方式是最复杂的，复杂性也带来了一定的灵活性。下面应该是整理一下绑定的过程。其中绑定的具体实现过程也已经在上面的文章中有介绍。只是一个较大的网络中，各种设备的加入、离开对网络通信的影响如何，还有待进一步的研究。