[zstack sampleApp通讯方式](http://blog.chinaunix.net/uid-24343357-id-3605104.html)

**Zigbee的通讯方式（即数据发送的形式）主要有三种：广播、点播、组播。**

**一、概念**

**1.1 广播**

**广播就是网络中任意一节点设备发出广播数据，网络中其它的任意节点都能收到。**

**1.2 点播**

**点播（也叫点对点）就是网络中任意一节点对另一个已知网络地址（即短地址）的节点进行数据发送的过程。**

**1.3 组播**

**组播（也叫组网）就是网络中所有节点设备被分组后，网络中任意组的任意一节点都可以对某一已知组号（包扩自身所属的组号）的组进行数据发送的过程。**

**二、相关函数的配置**

**需注意的是执行完自定义事件函数SampleApp\_ProcessEvent()后并不代表主函数结束，程序将会一直待在osal\_start\_system()进行任务轮询。**

**2.1 自定义任务初始化**

**在任务初始化函数osalInitTasks()中添加自己写的任务（事件）的初始化函数。在SampleApp\_Init ()里可以修改Zigbee通讯方式的模式、端点号、目标地址等等，相关的初始化代码如下。**

**// Setup for the periodic message's destination address**

**SampleApp\_Periodic\_DstAddr.addrMode = (afAddrMode\_t)afAddr16Bit;**

**//模式**

**SampleApp\_Periodic\_DstAddr.endPoint = SAMPLEAPP\_ENDPOINT;**

**//端点号**

**SampleApp\_Periodic\_DstAddr.addr.shortAddr = 0x0001;**

**//目标地址**

**// Fill out the endpoint description.填写端点描述**

**SampleApp\_epDesc.endPoint = SAMPLEAPP\_ENDPOINT;**

**SampleApp\_epDesc.task\_id = &SampleApp\_TaskID;**

**SampleApp\_epDesc.simpleDesc = (SimpleDescriptionFormat\_t \*)&SampleApp\_SimpleDesc;**

**SampleApp\_epDesc.latencyReq = noLatencyReqs;**

**// By default, all devices start out in Group 1 SampleApp\_Group.ID = 0x0001;**

**//本设备组号**

**osal\_memcpy( SampleApp\_Group.name, "Group 1", 7 );//组名**

**aps\_AddGroup(SAMPLEAPP\_ENDPOINT, &SampleApp\_Group);//添加新组**

**2.1.1 设置模式、端点号及目标地址**

**SampleApp\_Periodic\_DstAddr.addrMode = (afAddrMode\_t)afAddr16Bit;**

**//通信模式**

**SampleApp\_Periodic\_DstAddr.endPoint = SAMPLEAPP\_ENDPOINT;**

**//端点号**

**SampleApp\_Periodic\_DstAddr.addr.shortAddr = 0x0001;**

**//目标地址 （注：红色部分为需要修改的地方，具体修改如下文所示。）**

**①SampleApp\_Periodic\_DstAddr对应的结构体afAddrType\_t。**

**typedef struct**

**{             union**

**{**

**uint16  shortAddr;                     //目标地址**

**ZLongAddr\_t  extAddr;**

**}addr;**

**afAddrMode\_t  addrMode;                    //传送模式**

**byte  endPoint;                                     //端点号(0~255)**

**uint16  panId;                                       // used for the INTER\_PAN feature**

**}afAddrType\_t;**

**②设置传送模式afAddrMode\_t，只需将addrMode设置成需要的模式。**

**typedef enum {**

**afAddrNotPresent = AddrNotPresent,//绑定后的点播**

**afAddr16Bit      = Addr16Bit,           //点播**

**afAddr64Bit      = Addr64Bit,           //**

**afAddrGroup      = AddrGroup,              //组播**

**afAddrBroadcast  = AddrBroadcast      //广播**

**} afAddrMode\_t;**

**③设置目标地址shortAddr（见表1）。**



**2.1.2 建立新组**

**SampleApp\_Group.ID = 0x0001; //本设备组号**

**osal\_memcpy( SampleApp\_Group.name, "Group 1", 7 );//组名**

**aps\_AddGroup(SAMPLEAPP\_ENDPOINT, &SampleApp\_Group);//往组表添加新组**

**①   SampleApp\_Group对应的结构体aps\_Group\_t**

**typedef struct {**

**uint16 ID;                       //组号**

**uint8  name[APS\_GROUP\_NAME\_LEN]; //组名**

**} aps\_Group\_t;**

**②添加新组aps\_AddGroup()**

**函数原型ZStayus\_t aps\_AddGroup (unit8 endpiont , aps\_Group\_t \*group);**

**endpoint为此终端接收或发送给字段中的组信息，**

**group包含将要加入组表的组号和组名。**

**注意：**

**1、** **单播模式下，需要知道目标地址即接收节点的网络地址，而网络地址是由协调器随机分配的，可通过以下函数获知。  
一般情况下，协调器的网络地址为0x0000,   
第一个与协调器建立联系的节点为0x796F，  
第二个与协调器建立联系的节点为0x7970，依此类推。**

**NLME\_GetShortAddr()——返回本设备的16位网络地址**

**NLME\_GetCoordShortAddr()——返回本设备的父亲设备的16位网络地址**

**2、组播模式下，协调器无法接收到自己发送的数据；**

**3、组播模式下，终端无法接收到数据，协调器或路由器却可以收到，有可能是下图所示的 FALSE没有改成TRUE；**



**2.2 Zigbee发送数据**

**Zigbee发送数据时调用函数AF\_DataRequest()，只需要了解函数的参数，就可非常灵活的以各种方式来发送数据。**

**afStatus\_t  AF\_DataRequest(**

**afAddrType\_t \*dstAddr,//发送目的地址＋端点地址和传送模式**

**endPointDesc\_t \*srcEP,//源终端的描述(如:操作系统任务ID) uint16 cID, //用于接收器识别的标号（簇ID）**

**uint16 len, //发送数据长度**

**uint8 \*buf, //发送数据缓冲区(即发送的数据)**

**uint8 \*transID, //任务ID号**

**uint8 options, //有效位掩码的发送选项**

**uint8 radius ); //最大传送跳数**

**AF\_DataRequest()函数最终调用APSDE\_DataReq原语，而AF\_DataRequest()函数的调用会触发afDataConfirm()函数,数据的发送结果也由afDataConfirm()函数返回，而AF\_DataRequest()函数返回的值并不是真正的发送结果，返回值afStatus\_t为状态，若返回值等于SUCCESS(即0x00)，则数据发送成功。**

**2.3 Zigbee接收数据**

**若Zigbee接收到数据，则进入自定义事件SampleApp\_ProcessEvent()后，会触发能接收及处理数据包的函数SampleApp\_MessageMSGCB()，在此函数里将处理后的数据通过串口查看。**

**函数SampleApp\_MessageMSGCB(afIncomingMSGPacket\_t \*pkt)接收到的所有数据信息存储在afIncomingMSGPacket\_t \*pkt中。**

**typedef struct {**

**osal\_event\_hdr\_t hdr;     /\* OSAL Message header \*/**

**uint16 groupId;           /\*组号（若未设置则为0）\*/**

**uint16 clusterId;**

**/\*用于识别的标号(簇ID)，应该与数据发送函数中的簇ID一致\*/**

**afAddrType\_t srcAddr;**

**uint16 macDestAddr;    /\* MAC header destination short address \*/**

**uint8 endPoint;           /\*端点号\*/**

**uint8 wasBroadcast;       /\*判断是否为广播地址，若是返回TRUE\*/**

**uint8 LinkQuality;        /\* The link quality of the received data frame \*/**

**uint8 correlation;        /\* The raw correlation value of the received data frame \*/**

**int8  rssi;               /\* 接收到的射频功率单位dBm \*/**

**uint8 SecurityUse;        /\* deprecated \*/**

**uint32 timestamp;         /\* receipt timestamp from MAC \*/**

**afMSGCommandFormat\_t cmd; /\* 接收到的数据 \*/**

**} afIncomingMSGPacket\_t;**

**typedef struct {**

**byte   TransSeqNumber;**

**uint16 DataLength;//接收到的数据长度**

**byte  \*Data;//接收到的数据内容**

**} afMSGCommandFormat\_t;**

**在SampleApp\_MessageMSGCB()函数体内通过识别簇ID（与发送设备的数据发送函数中的簇ID一致），取出数据并串口显示，代码如下。**

**SampleApp\_MessageMSGCB (afIncomingMSGPacket\_t  \*pkt)**

**{…**

**switch ( pkt->clusterId )**

**{**

**case SAMPLEAPP\_PERIODIC\_CLUSTERID: //簇ID len = (char)pkt->cmd.DataLength;**

**//读取数据的长度**

**for(n=0;n<len;n++) < strong="" style="overflow-wrap: break-word;"></len;n++) <>**

**UartTX\_Send\_String(&pkt->cmd.Data[n],1);**

**//串口显示数据**

**… }**

**… }**