5.点播通讯-无线通讯

实验内容：

1. 实验多终端通讯
2. 掌握点对点通讯

提示：如果 3 个模块，可将其中一个节点做路由器，再观察现象更清楚。 实现现象：

将程序分别下载到协调器、终端，连接串口。如果 3 个模块，可将其中一个做路由器，上电可以看到只有协调器在一个周期内收到信息。也就是说路由器和终端均与地址为 0x00（协调器）的设备通信，不与其他设备通信。确定通信对象的就是节点的短地址，实现点对点传输。

实验详解：

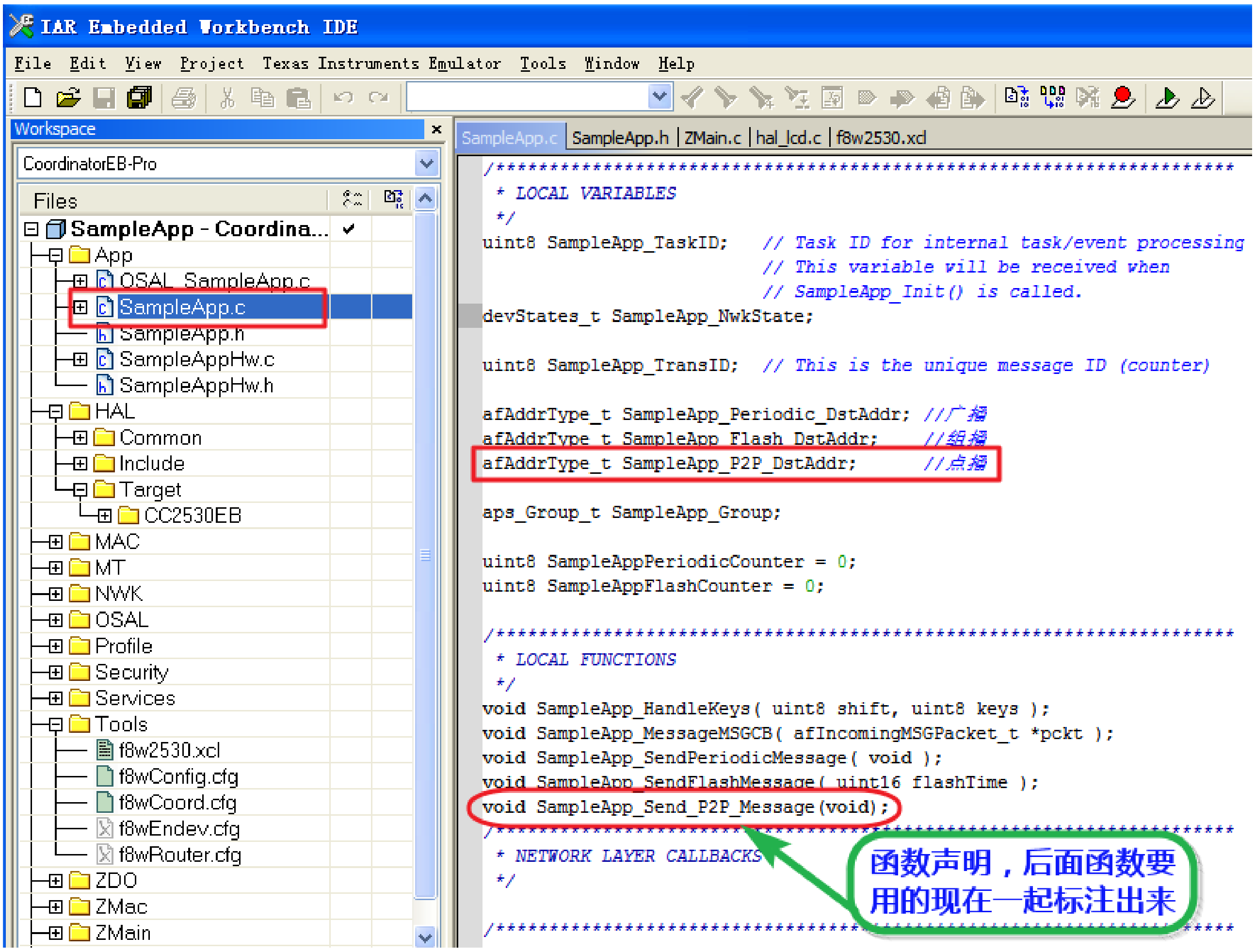
我们在“**3.广播组网-无线数据传输”例子中，**通过简单的修改即可完成点播实验。相同的工 程阅读代码更容易，等大家掌握到一定程度后，我们再换别的例子讲解。

打开..\Zigbee 资料\ 5. ZigBee 管理系统\ 5.点播通讯-无线通讯 \ZStack-2.5.1a\Projects\zstack\Samples\SampleApp\CC2530DB\SampleApp.eww 工程。在开始之前我们先了解下面两个重要结构：

```
typedef enum                                      //个枚举类型
{
   afAddrNotPresent = AddrNotPresent,
   afAddr16Bit          = Addr16Bit,           //点播方式
   afAddr64Bit          = Addr64Bit,
   afAddrGroup          = AddrGroup,           //组播方式
   afAddrBroadcast    = AddrBroadcast         //广播方式
} afAddrMode_t;

typedef struct
{
   union
   {
      uint16          shortAddr;                  //短地址
      ZLongAddr_t extAddr;                    //IEEE 地址
   } addr;
   afAddrMode_t addrMode;                      //传送模式
   byte endPoint;                                //端点号
   uint16 panId;                                 // used for the INTER_PAN feature
} afAddrType_t;
```

1. 找到 afAddrType_t SampleApp_Periodic_DstAddr;代码下面增加一行代码如下：



```
/*********************************************************************
 * LOCAL VARIABLES
 */
uint8 SampleApp_TaskID;   // Task ID for internal task/event processing
                          // This variable will be received when
                          // SampleApp_Init() is called.
devStates_t SampleApp_NwkState;

uint8 SampleApp_TransID;  // This is the unique message ID (counter)

afAddrType_t SampleApp_Periodic_DstAddr; //广播
afAddrType_t SampleApp_Flash_DstAddr;    //组播
afAddrType_t SampleApp_P2P_DstAddr;      //点播

aps_Group_t SampleApp_Group;

uint8 SampleAppPeriodicCounter = 0;
uint8 SampleAppFlashCounter = 0;

/*********************************************************************
 * LOCAL FUNCTIONS
 */
void SampleApp_HandleKeys( uint8 shift, uint8 keys );
void SampleApp_MessageMSGCB( afIncomingMSGPacket_t *pckt );
void SampleApp_SendPeriodicMessage( void );
void SampleApp_SendFlashMessage( uint16 flashTime );
void SampleApp_Send_P2P_Message(void);
/*********************************************************************
 * NETWORK LAYER CALLBACKS
 */

/*********************************************************************
```

函数声明，后面函数要用的现在一起标注出来

2. 搜索 afAddrGroup，在它下增加对 SampleApp_P2P_DstAddr 配置，可直接复制广播的配置 修改即可，增加后如下：

```
// Setup for the periodic message's destination address
// Broadcast to everyone
SampleApp_Periodic_DstAddr.addrMode = (afAddrMode_t)AddrBroadcast;//广播
SampleApp_Periodic_DstAddr.endPoint = SAMPLEAPP_ENDPOINT;
SampleApp_Periodic_DstAddr.addr.shortAddr = 0xFFFF;

// Setup for the flash command's destination address - Group 1
SampleApp_Flash_DstAddr.addrMode = (afAddrMode_t)afAddrGroup;//组播
SampleApp_Flash_DstAddr.endPoint = SAMPLEAPP_ENDPOINT;
SampleApp_Flash_DstAddr.addr.shortAddr = SAMPLEAPP_FLASH_GROUP;

SampleApp_P2P_DstAddr.addrMode = (afAddrMode_t)Addr16Bit; //点播
SampleApp_P2P_DstAddr.endPoint = SAMPLEAPP_ENDPOINT;
SampleApp_P2P_DstAddr.addr.shortAddr = 0x0000;          //发给协调器
```

协调器的地址规定为 0x0000

3. 增加发送函数，相信现在大家对下面的函数应该很熟悉了。

```
void SampleApp_Send_P2P_Message( void )
{
  uint8 data[11]="0123456789";
  if ( AF_DataRequest( &SampleApp_P2P_DstAddr, &SampleApp_epDesc,
                       SAMPLEAPP_P2P_CLUSTERID,
                       10,
                       data,
                       &SampleApp_TransID,
                       AF_DISCV_ROUTE,
                       AF_DEFAULT_RADIUS ) == afStatus_SUCCESS )
  {
  }
  else
  {
    // Error occurred in request to send.
  }
}
```

其中 SampleApp_P2P_DstAddr 是我们之前自己定义的，SAMPLEAPP_P2P_CLUSTERID 是在 SampleApp.h 中增加的，如下：

```
  #define SAMPLEAPP_PERIODIC_CLUSTERID     1
  #define SAMPLEAPP_FLASH_CLUSTERID        2
  #define SAMPLEAPP_P2P_CLUSTERID          3
```

4. 搜索 SampleApp_ProcessEvent，找到 if (events & SAMPLEAPP_SEND_PERIODIC_MSG_EVT) 修改成如下代码：

```
  if ( events & SAMPLEAPP_SEND_PERIODIC_MSG_EVT )
  {
    // Send the periodic message
    //SampleApp_SendPeriodicMessage();  //注释原来的发送函数
    SampleApp_Send_P2P_Message();       //增加点播的发送函数

    // Setup to send message again in normal period (+ a little jitter)
    osal_start_timerEx( SampleApp_TaskID, SAMPLEAPP_SEND_PERIODIC_MSG_EVT,
        (SAMPLEAPP_SEND_PERIODIC_MSG_TIMEOUT + (osal_rand() & 0x00FF)) );

    // return unprocessed events
    return (events ^ SAMPLEAPP_SEND_PERIODIC_MSG_EVT);
  }
```

5. 在接收方面，搜索找到 SampleApp_MessageMSGCB，我们进行如下修改（增加红色部分）：

```
void SampleApp_MessageMSGCB( afIncomingMSGPacket_t *pkt )
{
  uint16 flashTime;

  switch ( pkt->clusterId )
  {
    case SAMPLEAPP_P2P_CLUSTERID:
      HalUARTWrite(0, "Rx:", 3);          //提示接收到数据
      HalUARTWrite(0, pkt->cmd.Data, pkt->cmd.DataLength); //串口输出接收到的数据
```

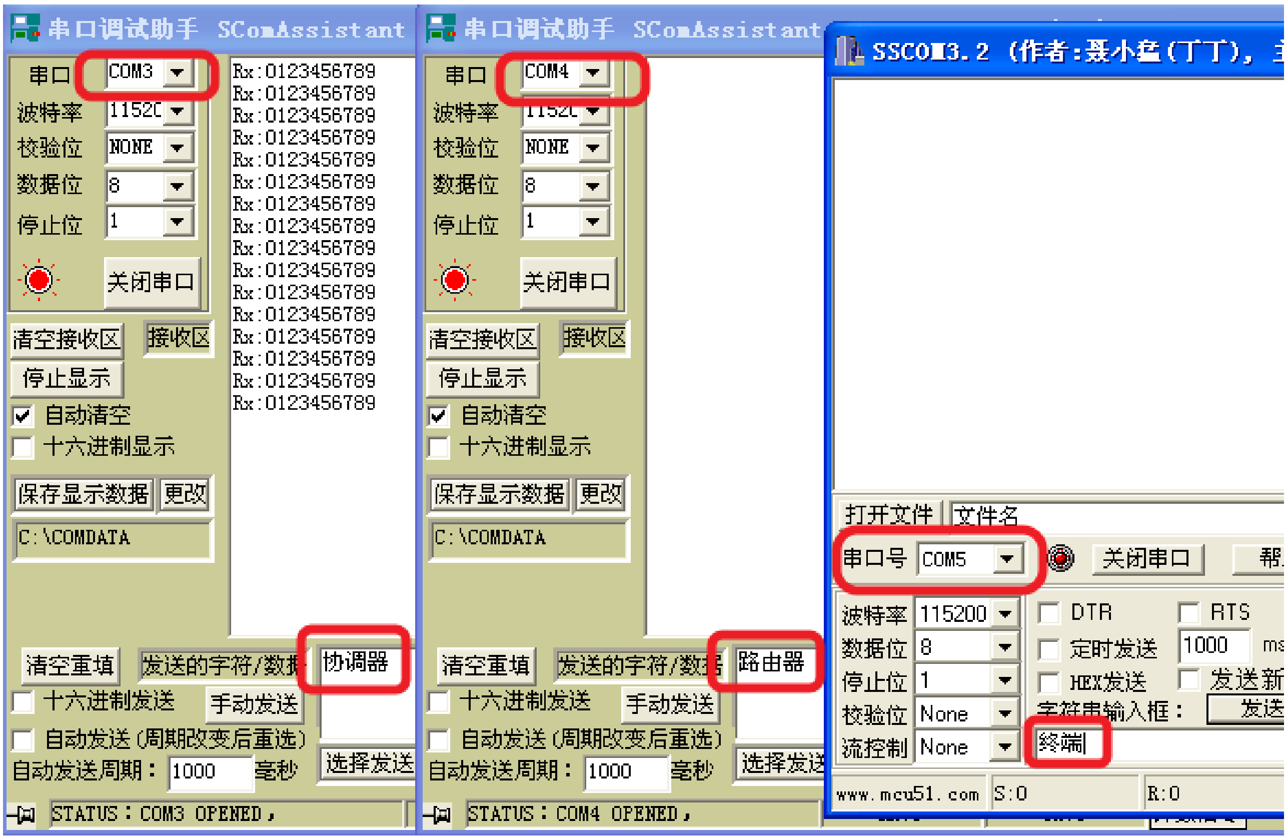
```
        HalUARTWrite(0, "\n", 1);               // 回车换行
        break;
      case SAMPLEAPP_PERIODIC_CLUSTERID:
        break;

      case SAMPLEAPP_FLASH_CLUSTERID:
        flashTime = BUILD_UINT16(pkt->cmd.Data[1], pkt->cmd.Data[2] );
        HalLedBlink( HAL_LED_4, 4, 50, (flashTime / 4) );
        break;
    }
}
```

6. 协调器不需要周期发数据，注释协调器的周期事件

```
      case ZDO_STATE_CHANGE:
        SampleApp_NwkState = (devStates_t)(MSGpkt->hdr.status);
        if ( //(SampleApp_NwkState == DEV_ZB_COORD) ||
                   (SampleApp_NwkState == DEV_ROUTER)
            || (SampleApp_NwkState == DEV_END_DEVICE) )
        {
          // Start sending the periodic message in a regular interval.
          osal_start_timerEx( SampleApp_TaskID,
                              SAMPLEAPP_SEND_PERIODIC_MSG_EVT,
                              SAMPLEAPP_SEND_PERIODIC_MSG_TIMEOUT );
        }
        else
        {
          // Device is no longer in the network
        }
        break;
```

最后别忘了加上图 1 中的函数声明，不然编译报错的。将修改后的程序分别编译、下载到协调器、路由器、终端，如果条件允许都连接串口。可以看到只有协调器一个周期性收到字符串。也就是说路由器和终端均与地址为 0x00（协调器）的设备通信，不语其他设备通信。实现点对点传输。如下图所示：



串口 COM3
波特率 1152C
校验位 NONE
数据位 8
停止位 1
C:\COMDATA
自动发送周期：1000 毫秒

```
Rx:0123456789
Rx:0123456789
Rx:0123456789
Rx:0123456789
Rx:0123456789
Rx:0123456789
Rx:0123456789
Rx:0123456789
Rx:0123456789
Rx:0123456789
Rx:0123456789
Rx:0123456789
Rx:0123456789
Rx:0123456789
Rx:0123456789
Rx:0123456789
```

协调器

STATUS：COM3 OPENED，

串口 COM4
波特率 1152C
校验位 NONE
数据位 8
停止位 1
C:\COMDATA
自动发送周期：1000 毫秒

路由器

STATUS：COM4 OPENED，

串口号 COM5
波特率 115200
数据位 8
停止位 1
校验位 None
流控制 None
1000 ms

终端

S:0 R:0

zigbee 中常用的结构体

数据发送： AF_DataRequest//

数据发送函数

AF_DataRequest(

afAddrType_t   *dstAddr, //目的地址结构体变量（含端点）

endPointDesc_t *srcEP,   //设备端点描述符（源端点描述）

unit16   cID,   //串 ID   （命令）

unit16   len,   //有效数据长度

unit8   *buf,   //数据

unit8   *transID,

unit8   *options,

unit8   radius,

)

例子：

```
AF_DataRequest( &SampleApp_Flash_DstAddr,
           &SampleApp_epDesc,
             SAMPLEAPP_FLASH_CLUSTERID,
              2,
              buffer,
             &SampleApp_TransID,
             AF_DISCV_ROUTE,
             AF_DEFAULT_RADIUS )
```

```
typedef struct// afAddrType_t;目的地址结构体变量
{
 union
   {
   uint16       shortAddr;
   ZLongAddr_t extAddr;
   }addr;
  afAddrMode_t addrMode;//afAddrMode_t 是一个枚举类型模式参数
 byte endPoint;//指定的端点号  端点 241—254 保留端点  范围  1-240
 uint16 panId;   // used for theINTER_PAN feature
} afAddrType_t;      //目的地址结构体变量（含端点）
```

```
例：afAddrType_t    SampleApp_Flash_DstAddr;
      /*  设置闪烁命令的目的地址（发送给组 1 的所有成员）*/
 SampleApp_Flash_DstAddr.addrMode= (afAddrMode_t) afAddrGroup;
 SampleApp_Flash_DstAddr.endPoint = SAMPLEAPP_ENDPOINT;
 SampleApp_Flash_DstAddr.addr.shortAddr = SAMPLEAPP_FLASH_GROUP;
```

```
typedef   enum//afAddrMode_t 数据传送类型
{
afAddrNotPresent   =   AddrNotPresent,     //间接传送(Indirect)
afAddr16Bit   =   Addr16Bit,                //指定地址单点传送(Unicast)   16 位
afAddrGroup   =   AddrGroup,                //组寻址(Group Addressing)
afAddrBroadcast   =   AddrBroadcast         //广播传送(broadcast)
} afAddrMode_t;//数据传送类型
```

```
typedef struct// endPointDesc_t;设备端点描述符
{
 byte   endPoint;
 byte *task_id;          // Pointer to location of theApplication task ID.
 SimpleDescriptionFormat_t *simpleDesc;        //设备简单描述符
 afNetworkLatencyReq_t   latencyReq;
} endPointDesc_t;//设备端点描述符  例：
      endPointDesc_tSampleApp_epDesc;
      /*  填充端点描述符  */
  SampleApp_epDesc.endPoint =SAMPLEAPP_ENDPOINT;
  SampleApp_epDesc.task_id =&SampleApp_TaskID;
  SampleApp_epDesc.simpleDesc
           =(SimpleDescriptionFormat_t *)&SampleApp_SimpleDesc;
  SampleApp_epDesc.latencyReq= noLatencyReqs;
```

typedef struct// zAddrType_t;地址变量（长地址或者短地址）

```
{
 union
  {
  uint16          shortAddr;
  ZLongAddr_t    extAddr;
  }addr;
 byte addrMode;
} zAddrType_t;   //地址变量（长地址或者短地址）
```

```
typedef struct// aps_Group_t;组结构体
{
 uint16 ID;                    // Unique to this table
 uint8   name[APS_GROUP_NAME_LEN];// Human readable name of group
} aps_Group_t;   //组结构体
```

SimpleDescriptionFormat_t;/*设备的简单描述符 */

```
 typedef struct
{
  byte EndPoint;                           //EP ID (EP=End Point)
  uint16 AppProfId;                  // profile ID（剖面 ID）
  uint16 AppDeviceId;                 // Device ID
  byte AppDevVer:4;                 //Device Version 0x00  为  Version 1.0
  byte Reserved:4;                  // AF_V1_SUPPORT uses for AppFlags:4.
  byte AppNumInClusters;                //终端支持的输入簇的个数
  cId_t*pAppInClusterList;        //指向输入 Cluster ID 列表的指针
  byte AppNumOutClusters;          //输出簇的个数
  cId_t *pAppOutClusterList;      //指向输出 Cluseter ID 列表的指针
} SimpleDescriptionFormat_t;
```

例子：

```
const SimpleDescriptionFormat_tSampleApp_SimpleDesc =
{
 SAMPLEAPP_ENDPOINT,              //   端点号
 SAMPLEAPP_PROFID,                // Profile ID
 SAMPLEAPP_DEVICEID,              //   设备 ID
 SAMPLEAPP_DEVICE_VERSION,        // 设备版本
 SAMPLEAPP_FLAGS,                 //   标识
 SAMPLEAPP_MAX_CLUSTERS,          // 输入簇的数量
 (cId_t *)SampleApp_ClusterList,  //   输入簇列表
 SAMPLEAPP_MAX_CLUSTERS,          //   输出簇的数量
 (cId_t *)SampleApp_ClusterList   //   输出簇列表
};
```

数据接收:

```
typedef struct   //afIncomingMSGPacket_t
{
 osal_event_hdr_t   hdr;      /* OSAL Message header */
 uint16   groupId;           /* Message's group ID - 0 if not set*/
 uint16   clusterId;         /* Message's cluster ID */
 afAddrType_t   srcAddr;      /* Source Address, if endpoint isSTUBAPS_INTER_PAN_EP,
                              it's an InterPANmessage */
 uint16   macDestAddr;        /* MAC header destination short address*/
 uint8   endPoint;            /* destination endpoint */
 uint8   wasBroadcast;        /* TRUE if network destination was abroadcast address */
 uint8   LinkQuality;         /* The link quality of the receiveddata frame */
 uint8   correlation;         /* The raw correlation value of thereceived data frame */
 int8    rssi;                /* The received RF power inunits dBm */
 uint8   SecurityUse;         /* deprecated */
 uint32 timestamp;            /*receipt timestamp from MAC */
 afMSGCommandFormat_t   cmd; /*Application Data */
} afIncomingMSGPacket_t;
//afIncomingMSGPacket_t   gtwRxFromNode;

// Generalized MSG Command Format
typedef struct   // afMSGCommandFormat_t;
```

```
{
 byte   TransSeqNumber;
 uint16 DataLength;            // Number of bytes in TransData
 byte  *Data;
} afMSGCommandFormat_t;

typedef struct //osal_event_hdr_t;
{
 uint8  event;
 uint8  status;
} osal_event_hdr_t;


typedef struct  //afDataConfirm_t;
{
 osal_event_hdr_t hdr;
 byte endpoint;
 byte transID;
}afDataConfirm_t;

typedef struct // ZDO_ActiveEndpointRsp_t;
{
 uint8   status;
 uint16  nwkAddr;   // Network address of interest
 uint8   cnt;
 uint8   epList[];
} ZDO_ActiveEndpointRsp_t;
```