数据结构以及框架命令说明

1. 数据结构

a.发送和接收的命令串的数据结构

```
/**
  * @brief 发送和接收的命令串数据结构
  * @brief 最终使用的12字节数据结构
  */
typedef struct{
  uint8            CMD;       /*命令Command     */
  uint8            ClusterID; /*ClusterID Flag */
  SendNode_t       SendNode;  /*节点信息        */
}Send_Recv_CMDA_t;
```

b.zigbee 网络内的数据结构

```
/*定义zigbee网络内的数据结构*/
typedef UnionInfo_mac ZigbeeData_t;
```

c.可以进行传递的信息的数据信息

```
/**
  * @brief 需要传递的信息，包括节点信息和地址信息等，长度为4个字节
  */
typedef union{
    NetParamt_t NetParamt;   //网络参数
    NodeInfo_t  NodeInfo;    //节点信息
    pNodeInfo_t pNodeInfo;   //父节点信息
    BindCtrl_t  BindCtrl;    //绑定控制和管理相关参数
    LState_t    LState;      //灯参数和变量
    Curtain_t   Curtain;     //窗帘参数和变量
    ColorL_t    ColorL;      //彩灯参数和变量
}UnionInfo_t;
```

d.UnionInfo_t 里面的每一个详细项

```
/**
  * @brief  网络参数
  * @brief  信道、发射功率、panID
  */
typedef struct{
    uint8  Channel;          //信道
    uint8  Txpower;          //发射功率
    uint16 panID;            //panID
}NetParamt_t;

/**
  * @brief  节点信息
  * @brief  节点类型、节点应用类型
  */
typedef struct {
    uint8  NodeType;         //NodeType
    uint8  NodeAppType;      //Node App Type
    uint8  Reserved[2];      //保留位-供需使用
}NodeInfo_t;
/**
  * @brief 父节点信息
  * @brief 父节点类型、父节点应用类型、父节点短地址
  */
typedef struct {
    uint8  pNodeType;        //父节点类型
    uint8  pNodeAppType;     //父节点应用类型
    uint16 pNwkSrc;          //父节点Nwk
}pNodeInfo_t;

/**
  * @brief 绑定控制和管理相关参数
  * @brief 短地址和clusterID
  */
typedef struct{
    uint16 NwkDest;
    uint16 ClusterID;
}BindCtrl_t;
```

```c
/**
  * @brief 灯参数和变量
  */
typedef struct
{
    uint16 ClusterID;        //控制灯命令
    uint8  CtrlVal;          //控制灯的值
    uint8  CtrlMode;         //控制灯的模式 ： 0-瞬间变亮 ； 1-逐渐变亮
}LState_t;

/**
  * @brief 窗帘参数和变量
  */
typedef struct{
    uint8  Reserved[4];      //保留位-供需使用
}Curtain_t;

/**
  * @brief 彩灯参数和变量
  * @brief 以类似结构向后扩展
  */
typedef struct{
    uint8  CtrlMode;         //控制模式 ： 0-瞬间变亮 ； 1-逐渐变亮
    uint8  Red;              //红色值
    uint8  Blue;             //蓝色值
    uint8  Green;            //绿色值
}ColorL_t;
```

2. Zigbee 网络内的应用类型

```c
/**
  * @brief Zigbee网络内所有节点的节点应用类型定义
  */
#define NODE_APP_TYPE_LIGHT             0x01  //灯类型
#define NODE_APP_TYPE_CLIGHT            0x02  //彩灯类型
#define NODE_APP_TYPE_CURTAIN           0x03  //窗帘类型
#define NODE_APP_TYPE_GETEWAY           0x04  //网关类型
```

3. 轮询事件处理层

```c
/**
  * @fun    SoftWareEventInLoop
  * @brief  事件在环执行
  * @param  None
  * @retval None
  */
void SoftWareEventInLoop(void)
{
    uint8  hdlstate=0;  /*处理状态*/

    //-----------------------------------/*软件在环串口事件*/
    if(LoopEvents & YM_EVENT_SERIAL){
        YM_SerialRecv((uint8*)&srcmd);
        LoopEventSetBit(YM_EVENT_CMD);  /*置位命令执行事件*/
        hdlstate=1;
        if(hdlstate){
            LoopEvents ^= YM_EVENT_SERIAL;
            hdlstate ^= hdlstate;
        }
    }

    //-----------------------------------/*软件在环命令事件*/
    if(LoopEvents & YM_EVENT_CMD){
        CmdHandler();
        LoopEventSetBit(YM_EVENT_SERIAL);
        hdlstate=1;
        if(hdlstate){
            LoopEvents ^= YM_EVENT_CMD;
            hdlstate ^= hdlstate;
        }
    }
```

```
    //-----------------------------------/*软件在环测试事件*/
    if(LoopEvents & YM_EVENT_TEST){


        if(hdlstate){
            LoopEvents ^= YM_EVENT_TEST;
            hdlstate ^= hdlstate;
        }
    }

    //-----------------------------------/*软件在环定时事件*/
    if(LoopEvents & YM_EVENT_TIMEING){
        hdlstate=TimeingLoopEvent();
        if(hdlstate){
            LoopEvents ^= YM_EVENT_TIMEING;
            hdlstate ^= hdlstate;
        }
    }
}
```

4. 定时事件处理层

```
/**
  * @fun    SoftWareTimeingEventInLoop
  * @brief  定时事件在环执行
  * @param  None
  * @retval None
  */
void SoftWareTimeingEventInLoop(void)
{
    static uint16 HeartBeatLt=0;/*定义心跳事件上一次状态的时间*/
    static uint16 SensorLt=0;    /*定义传感器上一次状态的时间   */
    static uint16 GetNodeIdLt=0;/*定义获取NodeID上一次时间值   */
    static uint16 NodeInitLt=0; /*定义节点初始化的上一次时间值*/

    static uint8  InitState=0;
    uint16  newTime;
    newTime=halCommonGetInt16uMillisecondTick();/*获得当前运行时间*/
    //----------------------------------------
    if(TimeDifference(newTime,HeartBeatLt)>HEART_BEAT_TIME){
        LoopEventSetBit(YM_EVENT_TIMEING);
        HeartBeatLt=newTime;
    }

    //----------------------------------------
    if(TimeDifference(newTime,SensorLt)>SENSOR_RET_TIME){
        LoopEventSetBit(YM_EVENT_TIMEING);
        SensorLt=newTime;
    }

    //----------------------------------------
    if(TimeDifference(newTime,GetNodeIdLt)>GET_NODETYPE_TIME){
        LoopEventSetBit(YM_EVENT_TIMEING);
        LoopTimeEventSetBit(YM_T_EVENT_GET_NODETYPE);
        GetNodeIdLt=newTime;
    }
    //----------------------------------------
    if(TimeDifference(newTime,NodeInitLt)>NODE_INIT_TIME){
        if(!InitState){
            LoopTimeEventSetBit(YM_T_EVENT_INIT);
        }
        InitState=1;
        LoopEventSetBit(YM_EVENT_TIMEING);
        NodeInitLt=newTime;
    }
}
```

```
/**
  * @fun    TimeingLoopEvent
  * @brief  定时循环事件执行函数
  * @param  None
  * @retval state of handle event
  */
uint8 TimeingLoopEvent(void)
{
    uint8 state=0;
    uint8 hdlstate=0;

    //----------------------------------/*心跳包事件      */
    if(LoopTimeEvents & YM_T_EVENT_HEARTBEAT){

        hdlstate=1;
        if(hdlstate){
            LoopTimeEvents ^= YM_T_EVENT_HEARTBEAT;
            hdlstate ^= hdlstate;
        }
    }
    //----------------------------------/*获取NodeType事件*/
    if(LoopTimeEvents & YM_T_EVENT_GET_NODETYPE){
        GetNodeTypeCheckNetwork();
        hdlstate=1;
        if(hdlstate){
            LoopTimeEvents ^= YM_T_EVENT_GET_NODETYPE;
            hdlstate ^= hdlstate;
        }
    }
    //----------------------------------/*定时初始化事件*/
    if(LoopTimeEvents & YM_T_EVENT_INIT){
        YM_DebugSerialStrOutput("YM_T_EVENT_INIT");
        GetNodeNwk();
        hdlstate=1;
        if(hdlstate){
            LoopTimeEvents ^= YM_T_EVENT_INIT;
            hdlstate ^= hdlstate;
        }
    }
    if(!LoopTimeEvents) state=1;
    return state;
}
```

5. 从串口获取的命令与数据解析处理

```
/*命令解析函数*/
void CmdHandler(void)
{
    void (*pf)(Send_Recv_CMDA_t *srcd)=(void*)0;
    switch(srcmd.CMD){
        case CMD_NODE_INFO_LOOKUP:  /*节点信息查询*/
          pf=Abs_NodeInfoLookup;
          YM_DebugSerialStrOutput("CMD_NODE_INFO_LOOKUP\n");
          break;
        case CMD_NETWORK_OPRT:      /*网络操作*/
          pf=Abs_NetWorkOprt;
          YM_DebugSerialStrOutput("CMD_NETWORK_OPRT\n");
          break;
        case CMD_LIGHT_OPRT:        /*灯控制操作*/
          pf=Abs_LightOprt;
          YM_DebugSerialStrOutput("CMD_LIGHT_OPRT\n");
          break;
        case CMD_CURTAIN_OPRT:      /*窗帘控制操作*/
          pf=Abs_CurtainOprt;
          YM_DebugSerialStrOutput("CMD_CURTAIN_OPRT\n");
          break;
        default:
          pf=(void*)0;
          break;
    }
    if(pf) pf(&srcmd);
    clcstr((uint8*)&srcmd,sizeof(Send_Recv_CMDA_t));
}
```

```
/*节点信息查询命令处理*/
static void Abs_NodeInfoLookup(Send_Recv_CMDA_t *srcd);

/*网络操作命令处理*/
static void Abs_NetWorkOprt(Send_Recv_CMDA_t *srcd);

/*灯操作命令处理*/
static void Abs_LightOprt(Send_Recv_CMDA_t *srcd);

/*窗帘操作命令处理*/
static void Abs_CurtainOprt(Send_Recv_CMDA_t *srcd);
```

6. 从无线获取的命令与数据解析处理

```
/**
  * @fun     YM_AfMessageRecvCmdHandler
  * @brief   无线信息命令处理函数
  * @param   incomingMessage :指向接收入的无线信息变量的指针
  * @retval None
  */
void YM_AfMessageRecvCmdHandler(EmberAfIncomingMessage* incomingMessage)
{
    uint8   cmd=0;
    void (*pf)(EmberAfIncomingMessage *pkt)=(void*)0;
    cmd|=clusterID>>8;  /*提取出命令*/
    switch(cmd){
        case CMD_NODE_INFO_LOOKUP:  /*节点信息查询*/
          pf=AbsAf_NodeInfoLookup;
          YM_DebugSerialStrOutput("AbsAf_NodeInfoLookup\n");
          break;
        case CMD_NETWORK_OPRT:      /*网络操作*/
          pf=AbsAf_NetWorkOprt;
          YM_DebugSerialStrOutput("AbsAf_NetWorkOprt\n");
          break;
        case CMD_LIGHT_OPRT:        /*灯控制操作*/
          pf=AbsAf_LightOprt;
          YM_DebugSerialStrOutput("AbsAf_LightOprt\n");
          break;
        case CMD_CURTAIN_OPRT:      /*窗帘控制操作*/
          pf=AbsAf_CurtainOprt;
          YM_DebugSerialStrOutput("AbsAf_CurtainOprt\n");
          break;
        default:
          pf=(void*)0;
          break;
    }
    if(pf) pf(incomingMessage);
}
```

```
/*节点信息查询*/
static void AbsAf_NodeInfoLookup(EmberAfIncomingMessage *pkt);

/*网络操作*/
static void AbsAf_NetWorkOprt(EmberAfIncomingMessage *pkt);

/*灯控制操作*/
static void AbsAf_LightOprt(EmberAfIncomingMessage *pkt);

/*窗帘控制操作*/
static void AbsAf_CurtainOprt(EmberAfIncomingMessage *pkt);
```

7. 无线数据的发送函数

```
/**
  * @fun     YM_AfMessageSendHandler
  * @brief   无线信息数据发送处理函数
  * @param   AfSendData :指向要发送的无线信息变量的指针
  * @retval None
  */
void YM_AfMessageSendHandler(AfSendData_t *AfSendData)
{
    uint8   messageTag;
    EmberApsFrame  apsFram;
    apsFram.profileId=PROFILE_ID;
    apsFram.clusterId=AfSendData->ClusterID;
    apsFram.sourceEndpoint=ENDPOINT;
    apsFram.destinationEndpoint=ENDPOINT;
    apsFram.options=EMBER_APS_OPTION_RETRY;
    emAfSend(AfSendData->msgType,
             AfSendData->Nwk,
             &apsFram,
             AfSendData->DataLength,
             AfSendData->SendData,
             &messageTag);
}
```

```
/*定义无线数据发送数据类型*/
typedef struct{
    uint16  Nwk;
    uint16  ClusterID;
    uint8   *SendData;
    uint8   DataLength;
    uint8   msgType;
}AfSendData_t;

/*无线信息数据发送处理函数*/
extern void YM_AfMessageSendHandler(AfSendData_t *AfSendData);

/*无线信息命令处理函数*/
extern void YM_AfMessageRecvCmdHandler(EmberAfIncomingMessage* incomingMessage);
```

8. 已定义的命令

```
//*****节点信息查询******************
//************************************
#define CMD_NODE_INFO_LOOKUP        0x01          //CMD

//------------------------------------

#define CID_NODE_TYPE_LOOKUP        0x01   //节点类型查询
#define CID_NODE_ALLNWK_LOOKUP      0x02   //所有节点地址信息查询
#define CID_NODE_NWK_LOOPUP         0x03   //节点 nwk 地址查询-根据 mac 地址
#define CID_NODE_MAC_LOOPUP         0x04   //节点 mac 地址查询-根据 nwk 地址
#define CID_NODE_PANID_LOOPUP       0x05   //节点 PANID 信息查询
#define CID_NODE_EXTPANID_LOOPUP    0x06   //节点 extPANID 信息查询
#define CID_NODE_CHANNEL_LOOPUP     0x07   //节点信道查询
#define CID_NODE_P_NWK_LOOPUP       0x08   //父节点 nwk 地址查询
#define CID_NODE_P_MAC_LOOPUP       0x09   //父节点 mac 地址查询
#define CID_NODE_APPTYPE_LOOKUP     0x0A   //节点应用类型查询
```

```
//*****网络操作**********************
//***********************************
#define CMD_NETWORK_OPRT            0x02        //CMD

//-----------------------------------

#define CID_NETWORK_FORMAT_REQ    0x01   //网络格式化，重启网络使用
#define CID_NETWORK_PERMIT_JOIN   0x02   //允许加入网络
#define CID_NETWORK_PERMIT_LEAVE  0x03   //允许离开网络
#define CID_NETWORK_REMOVE_CHILD  0x04   //移除子节点
#define CID_NETWORK_FOUND         0x05   //网络发现
#define CID_NETWORK_GETPARAMET    0x06   //获取网络参数

//*****灯控制操作**********************
//***********************************
#define CMD_LIGHT_OPRT              0x03        //CMD

//-----------------------------------

#define CID_LIGHT_TURN_ON           0x01   //灯开启
#define CID_LIGHT_TURN_OFF          0x02   //灯关闭

//*****窗帘控制操作********************
//***********************************
#define CMD_CURTAIN_OPRT            0x04        //CMD

//-----------------------------------

#define CID_CURTAIN_OPEN            0x01   //窗帘打开
#define CID_CURTAIN_CLOSE           0x02   //窗帘关闭
#define CID_CURTAIN_STOP            0x03   //窗帘停止

//*****************************公共 ClusterID*********************************
//Zigbee 网络内使用的控制 ClusterID
//-------------------------------------------------------------------------

#define EXEAPP_CID_NODE_TYPE_LOOKUP          0x0101   //节点类型查询
#define EXEAPP_CID_NODE_ALLNWK_LOOKUP        0x0102   //所有节点地址信息查询
#define EXEAPP_CID_NODE_NWK_LOOPUP           0x0103   //节点 nwk 地址查询-根据 mac 地址
#define EXEAPP_CID_NODE_MAC_LOOPUP           0x0104   //节点 mac 地址查询-根据 nwk 地址
#define EXEAPP_CID_NODE_PANID_LOOPUP         0x0105   //节点 PANID 信息查询
#define EXEAPP_CID_NODE_EXTPANID_LOOPUP      0x0106   //节点 extPANID 信息查询
#define EXEAPP_CID_NODE_CHANNEL_LOOPUP       0x0107   //节点信道查询
#define EXEAPP_CID_NODE_P_NWK_LOOPUP         0x0108   //父节点 nwk 地址查询
```

```
#define EXEAPP_CID_NODE_P_MAC_LOOPUP        0x0109   //父节点 mac 地址查询
#define EXEAPP_CID_NODE_APPTYPE_LOOKUP      0x010A   //节点应用类型查询

//------------------------------------------------------------------------

#define EXEAPP_CID_NETWORK_FORMAT_REQ       0x0201   //网络格式化，重启网络使用
#define EXEAPP_CID_NETWORK_PERMIT_JOIN      0x0202   //允许加入网络
#define EXEAPP_CID_NETWORK_PERMIT_LEAVE     0x0203   //允许离开网络
#define EXEAPP_CID_NETWORK_REMOVE_CHILD     0x0204   //移除子节点
#define EXEAPP_CID_NETWORK_FOUND            0x0205   //网络发现
#define EXEAPP_CID_NETWORK_GETPARAMET       0x0206   //获取网络参数

//------------------------------------------------------------------------

#define EXEAPP_CID_LIGHT_TURN_ON            0x0301   //灯开启
#define EXEAPP_CID_LIGHT_TURN_OFF           0x0302   //灯关闭

//------------------------------------------------------------------------

#define EXEAPP_CID_CURTAIN_OPEN             0x0401   //窗帘打开
#define EXEAPP_CID_CURTAIN_CLOSE            0x0402   //窗帘关闭
#define EXEAPP_CID_CURTAIN_STOP             0x0403   //窗帘停止
```

——林铭锋-linails@foxmail.com