**Zigbee物联系统**

**版本：V1.0**

**使用说明书**

深圳市鸿哲智能系统工程有限公司

**目 录**

[一. 产品说明 2](#_Toc471736180)

[1.1 产品简介 2](#_Toc471736181)

[1.2 产品图示 2](#_Toc471736182)

[二．通信与设置 4](#_Toc471736183)

[2.1 通信结构框图 4](#_Toc471736184)

[2.2 通信方式 6](#_Toc471736185)

[2.3 参数设置 6](#_Toc471736186)

[三. 服务器与主控制器之间的通信通信协议 15](#_Toc471736187)

[3.1. 帧类型 15](#_Toc471736188)

[3.2. 帧格式 16](#_Toc471736189)

[3.3. 帧类型说明 20](#_Toc471736190)

[四. 主控制器与从控制器之间的通信协议 27](#_Toc471736191)

[4.1. 帧类型 27](#_Toc471736192)

[4.2. 帧格式 27](#_Toc471736193)

[4.3. 帧类型说明 31](#_Toc471736194)

[五. 主\从控制器与电脑参数配置协议 40](#_Toc471736195)

[5.1. 帧类型 40](#_Toc471736196)

[5.2. 帧格式 40](#_Toc471736197)

[5.3. 帧类型说明 41](#_Toc471736198)

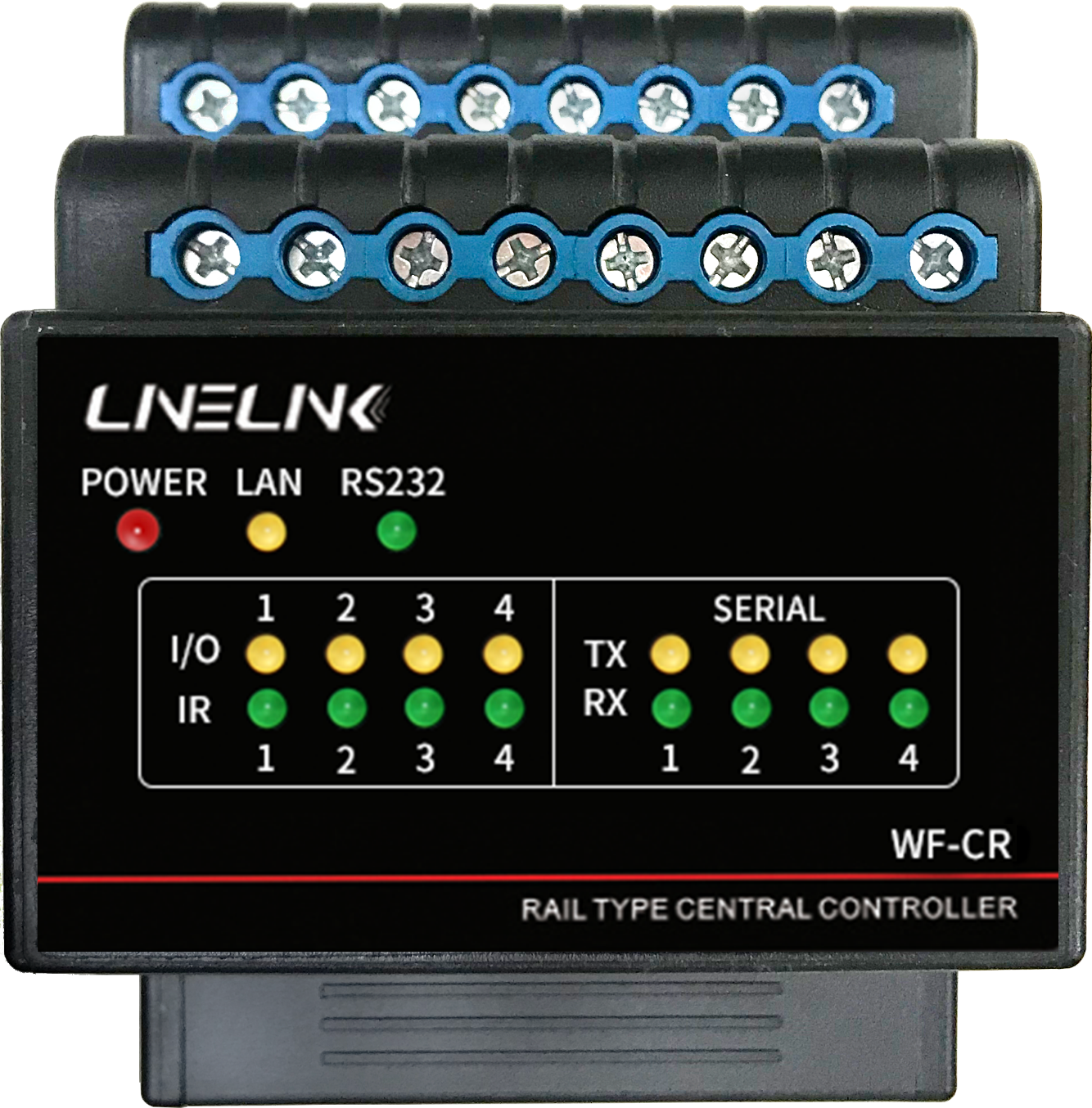
[5.4. 错误应答帧 51](#_Toc471736199)

# 一. 产品说明

## 1.1 产品简介

本物联系统主要用以组建大规模ZIGBEE无线传感网络，从而对于大范围以内的无线红外设备（如空调，电视等）以及有线控制设备（投影机，DVD等）进行控制，并且可同时对于环境温湿度等相关数据进行监测。ZIGBEE设备具有稳定，低功耗，远距离等特点，绿色环保，并易于安装，配合电脑软件可以对于大规模的系统进行整体的操作与监测。

## 1.2 产品图示



WF-CR——物联导轨中控



WF-8REL16A——物联导轨继电器



WF-GW——物联网关



WF-SIM0203——感应探测器

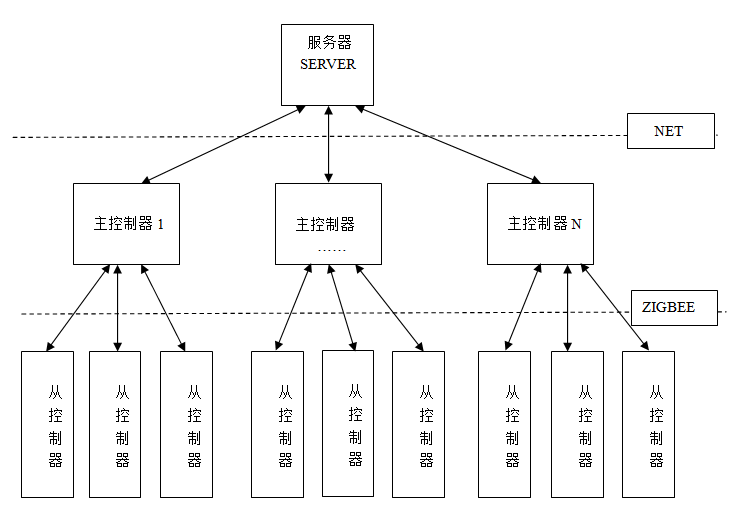
# 二．通信与设置

## 2.1 通信结构框图

服务器(SERVER)与主控制器(HOST)之间采用网络进行通信，该网络通过有线网络进行连接。

主控制器(HOST)与从控制器(DEVICE)之间采用无线ZIGBEE模式进行通信。

其通信结构框图如下：



在网络结构中，每个主控制器(HOST)均分配有一个唯一的IP地址，服务器

通过该IP地址识别不同的主控制器(HOST)。

在ZIGBEE结构中，1个主控制器(HOST)和多个从控制器(DEVICE)可以组成一个小型的ZIGBEE网络(比如1个教室内部)，组成这个小型的ZIGBEE网络

的所有模块必须分配有一个相同的私有网络号(PANID)，在该ZIGBEE网络中的每个模块又分配一个唯一的ID号(模块地址)，主控制器(HOST)内部ZIGBEE模块的ID号(地址)为0x0000，其它从控制器(DEVICE)的ID号从0x0001开始分配。

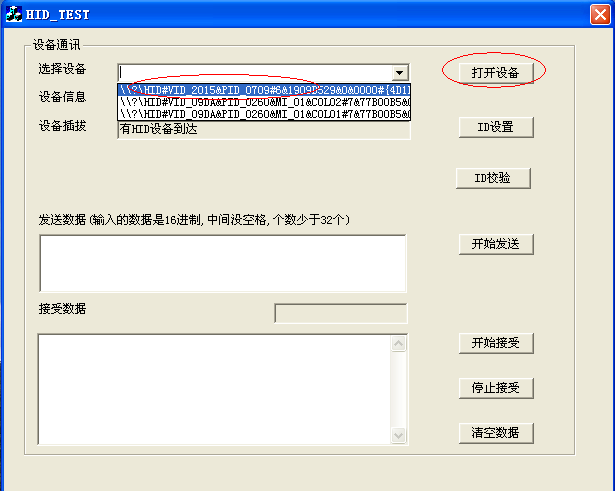
## 2.2 通信方式

服务器(SERVER)与主控制器(HOST)的通信为主从方式，服务器(SERVER)为主机(网络服务器)，主控制器(HOST)为从机(网络客户端)。一般情况下，通信都是由服务器发起，主控制器进行被动应答。服务器在2S内接收不到主控制器应答或者应答信息错误，则认为本次通信失败。

主控制器(HOST)与从控制器(DEVICE)之间的通信也为主从方式，主控制器为主机，从控制器为从机。一般情况下，通信都是由主控制器发起，从控制器进行被动应答。主控制器在500mS内接收不到从控制器应答或者应答信息错误，则认为本次通信失败。

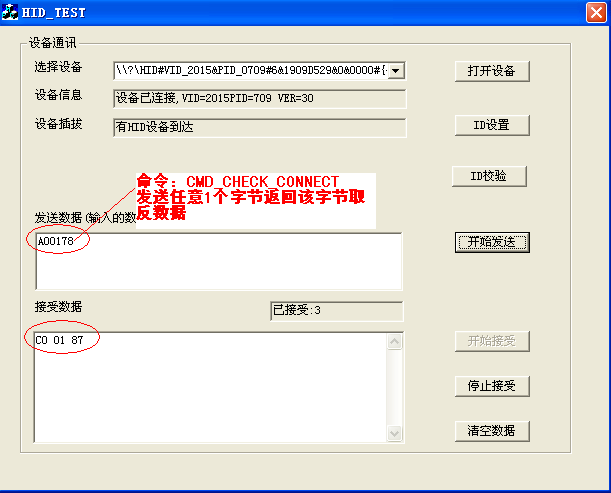
## 2.3 参数设置

Zigbee主机与从机设备的zigbee相关参数使用HID\_TEST软件进行设置，软件使用方法如下图示：

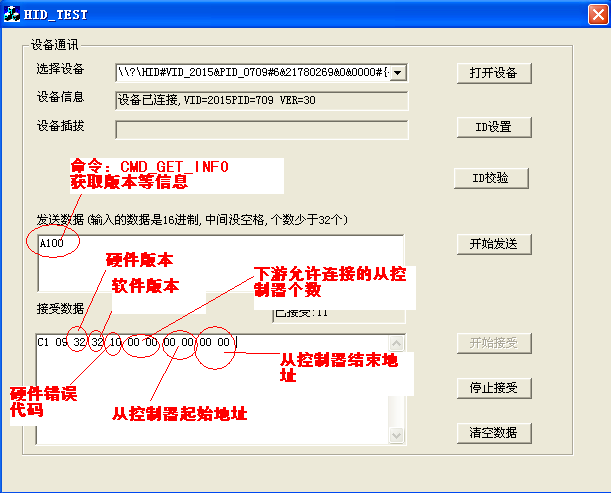


HID\_TEST使用图示1

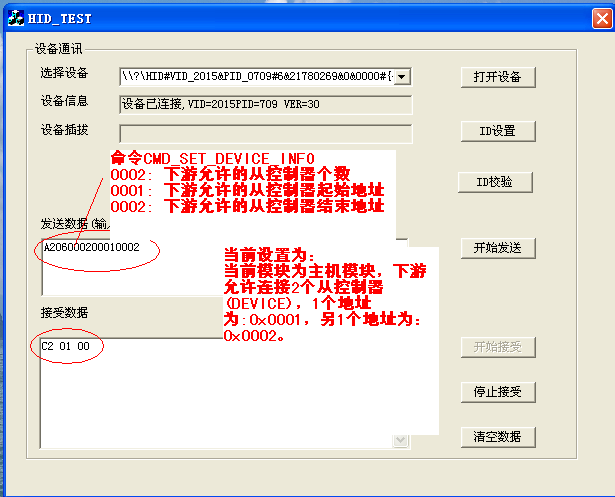
使用HID\_TEST软件时首先按上述图示1所示，当显示如下字样时表示设备正常连接首先选定设备，设备标有HID#VID\_2015&PID字样，选中设备之后，设备信息一栏会显示已选中设备的简要信息如下：，点击打开设备，在点击开始接受按钮，开始接受按钮变为不可点击状态即可开始发送数据对于设备参数进行设置，在接受区可看到返回的数据。具体设置指令按后续图例所示：（发送的数据不区分大小写，16进制，字节之间不加空格）



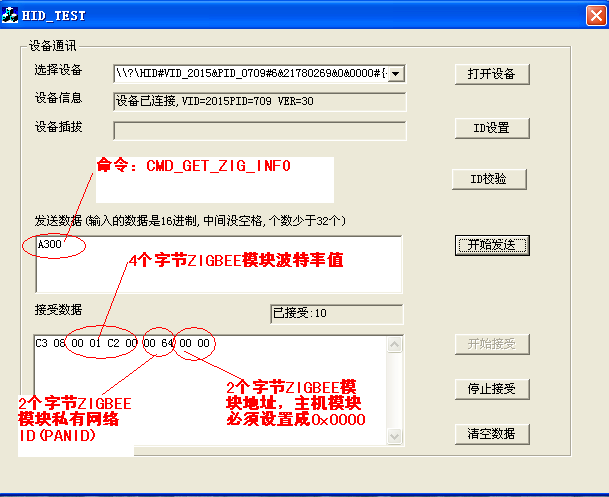
HID\_TEST使用图示2



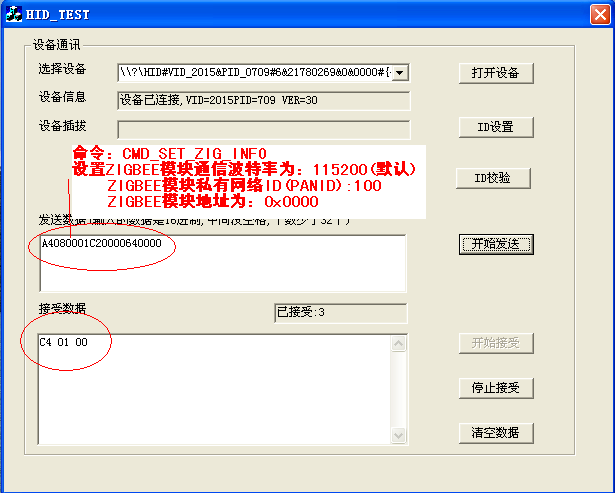
HID\_TEST使用图示3



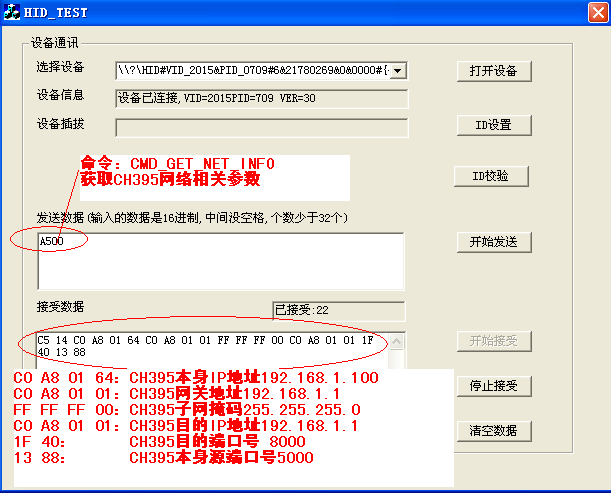
HID\_TEST使用图示4



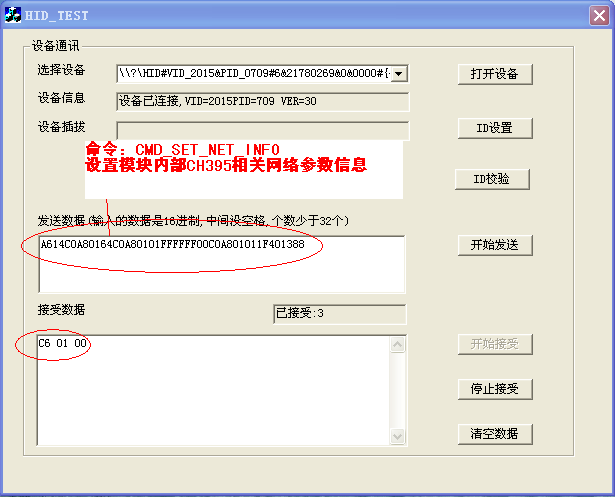
HID\_TEST使用图示5



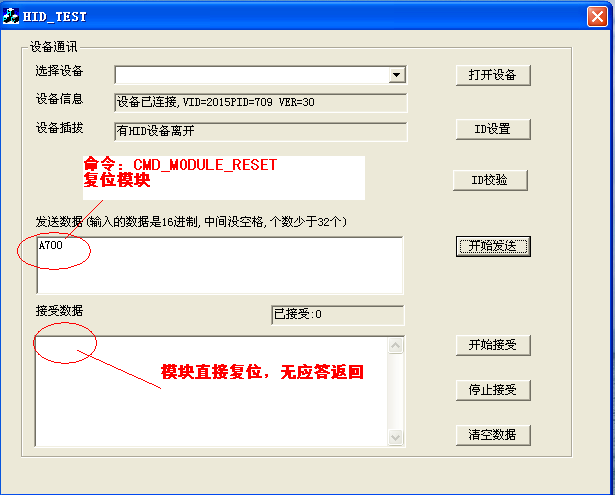
HID\_TEST使用图示6



HID\_TEST使用图示7



HID\_TEST使用图示8



HID\_TEST使用图示9

# 三. 服务器与主控制器之间的通信通信协议

## 3.1. 帧类型

服务器(SERVER)与主控制器(HOST)的通信以帧为单位，即通信的每包数据称为1个帧。通信帧分为两种类型，由服务器发送到主控制器的称为：命令帧；由主控制器发送到服务器的称为：应答帧。

接收方收到对方发送的帧后，先判断该帧是否正确，包括帧头、帧类型、长度、帧尾，如果其中有异常，则按照帧类型中的“错误应答帧”进行返回，如果校验正常，则执行该帧。

注：为简化通信协议，以主控制器和从控制器之间的通信协议为主，网络通信协议实际上是嵌套在该协议之上的。也就是基本上仅仅在HOST和DEVICE通信协议基础上再增加了一层而已。网络帧的DATA部分就是HOST和DEVICE之间的通信数据。

## 3.2. 帧格式

命令帧及应答帧的数据格式如下(大端格式)：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 序号 | 1 | 2 | 3 | 4 | 5 |
| 格式 | 帧头 | 帧类型 | 数据包长度 | 数据 | 帧尾 |
| 符号 | **SOF** | **TYPE** | **LENGTH** | **DATA** | **EOF** |
| 字节数 | 5 | 1 | 2 | N | 5 |

各个部分具体意义见下面叙述：

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 符号 | 字节数 | 简单说明 |
| 1 | SOF | 5 | **帧头**  每帧网络数据均以此5个字节开头，用于表示一帧的开始，其值依次为：0xA0、0xA1、0xA2、0xA3、0xA4。 |
| 2 | TYPE | 1 | **帧类型**  具体帧类型及对应的码值见“帧类型码表”。 |
| 3 | LENGTH | 2 | **数据包长度**  表示后续数据部分(DATA部分)的字节数，占2个字节，高字节在前。 |
| 4 | DATA | N | **数据包**  命令帧：命令数据信息(CMD DATA)；  应答帧：应答数据信息(RESP DATA)； |
| 5 | EOF | 5 | **帧尾**  每帧网络数据均以此5个字节结尾，用于表示一帧  的结束，其值依次为：0xB0、0xB1、0xB2、0xB3、  0xB4。 |

帧类型码表

|  |  |  |  |
| --- | --- | --- | --- |
| **序号** | **标志** | **码值** | **说明** |
| 1 | NET\_TYPE\_MID\_TRANS | 0x00 | 当前帧为：**转发帧**  该帧由服务器发送给主控制器(HOST)，再由主控制器转发给从控制器(DEVICE)，DEVICE执行完毕之后，将执行结果或者数据结果发送给HOST，最后由HSOT发送给SERVER。  转发帧中的数据部分包含了真正涉及到的主控制器与从控制器之间的通信命令及数据。  注：大部分帧均为转发帧。 |
| 2 | NET\_TYPE\_GET\_ALL\_INFO | 0x01 | 当前帧为：**获取信息帧**  服务器通过该帧可以获取到以下信息：  (1)、主控制器(HOST)当前硬件版本号；  (2)、主控制器(HOST)当前软件版本号；  (3)、主控制器(HOST)下游最多可以连接的从控制器个数及每个从控制器的ID号；  (4)、主控制器(HOST)下游当前实际连接的从控制器ID号；  该帧由主控制器本身执行，不下传给下游的从控制器。 |
| 3 | NET\_TYPE\_HEARTBEAT | 0x02 | 当前帧为：**心跳帧**  该帧由服务器发送给主控制器，由主控制器进行应答，每隔1分钟发送1次，主要用于判断主控制器与服务器直接连接是否正常，以及检测下游从控制器连接状态是否正常。  注：心跳帧主要用于长期判断服务器和主控制器(HOST)是否通信正常。 |
| 4 | NET\_TYPE\_ ERR\_RESP | 0xE? | 当前帧为：**错误应答帧**  主控制或服务器接收到对方的帧后，会对帧进行校验等判断，如果该帧存在异常或执行失败，则通过该帧向对方进行应答  应答码 = 命令码 | 0xE0。 |
| 5 | NET\_TYPE\_RESP | 0x8? | 当前帧为：**应答帧**  主控制器(HOST)执行完毕或从控制器(DEVICE) 执行完毕后，向服务器进行应答。  应答码 = 命令码 | 0x80。 |
| 6 | NET \_TYPE\_RS232\_ TRANS | 0x03 | 当前帧为：**串口转发帧**  该帧由服务器发送给主控制器(HOST)，再由HOST通过串口转发输出给投影机等第三方设备，获取到反馈后，再由HOST发送给SERVER。  **串口转发帧**中数据包含了服务器与第三方投影机等设备的通讯命令及数据，主要用来控制投影机的命令和读取投影机的灯泡寿命。 |

## 3.3. 帧类型说明

**3.3.1** **NET\_TYPE\_MID\_TRANS**

该帧由服务器发送给主控制器(HOST)，再由主控制器转发给从控制器(DEVICE)，DEVICE执行完毕之后，再将执行结果或者数据结果发送给HOST，最后由HSOT发送给SERVER。

主控制器(HOST)只是充当数据上传、下传转发角色。该帧的DATA部分数据实际上就是HOST和DEVICE之间的通信数据，具体见第4章节内容。

服务器🡪主控制器：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 格式 | 帧头 | 帧类型 | 数据包长度 | 数据 | 帧尾 |
| 符号 | **SOF** | **TYPE** | **LENGTH** | **DATA** | **EOF** |
| 数据 | 0xA0-0xA4 | 0x00 | 2个字节 | N个字节 | 0xB0-0xB4 |

其中：

TYPE：占1个字节，**转发帧**，其码值为0x00；

LENGTH：占2个字节，具体值根据后面DATA数据区字节数而定，高字节在前，假设DATA区有100个字节数据，则该2个字节依次为：0x00、0x64；

DATA：占N个字节，具体根据实际而定。注：DATA区中的数据内容即下章中主控制器(HOST)和从控制器(DEVICE)相关通信的内容。

主控制器🡪服务器：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 格式 | 帧头 | 帧类型 | 数据包长度 | 数据 | 帧尾 |
| 符号 | **SOF** | **TYPE** | **LENGTH** | **DATA** | **EOF** |
| 数据 | 0xA0-0xA4 | 0x80 | 2个字节 | N个字节 | 0xB0-0xB4 |

其中：

TYPE：占1个字节，当前帧为应答帧，其码值为0x80；

LENGTH：占2个字节，具体值根据DATA区真正功能而定；

DATA：占N个字节，具体值根据DATA区真正功能而定。注：DATA区中的数据内容即下章中主控制器(HOST)和从控制器(DEVICE)相关通信的内容。

**3.3.2 NET\_TYPE\_GET\_ALL\_INFO**

获取主控制器(HOST)及下游所有连接的从控制器(DEVICE)的相关信息。该帧由服务器发送给主控制器，由主控制器本身执行。

服务器🡪主控制器：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 格式 | 帧头 | 帧类型 | 数据包长度 | 数据 | 帧尾 |
| 符号 | **SOF** | **TYPE** | **LENGTH** | **DATA** | **EOF** |
| 数据 | 0xA0-0xA4 | 0x01 | 0x00、0x00 | 无 | 0xB0-0xB4 |

其中：

TYPE：占1个字节，获取主控制器与下游所有从控制器信息帧，其码值为0x01；

LENGTH：占2个字节，由于无后续数据，其值为0x00、0x00；

DATA：占0个字节。

主控制器🡪服务器：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 格式 | 帧头 | 帧类型 | 数据包长度 | 数据 | 帧尾 |
| 符号 | **SOF** | **TYPE** | **LENGTH** | **DATA** | **EOF** |
| 数据 | 0xA0-0xA4 | 0x81 | 0x??、0x?? | N个字节 | 0xB0-0xB4 |

其中：

TYPE：占1个字节，当前帧为应答帧，其码值为0x81；

LENGTH：占2个字节，具体值和主控制器设置的下游从控制器个数有关；

DATA：占N个字节，具体值和主控制器设置的下游从控制器个数有关。

如果下游最多可连接的从控制器(DEVICE)个数为10，ID号依次为0x0001---0x000A；当前下游实际连接的从控制器个数为7，ID号分别为：0x0001、

0x0002、0x0004、0x0005、0x0006、0x0007、0x000A；

则LENGTH = 3 + 2 + 2 \* 10 + 2 + 2 \* 7；

DATA区的3 + ( 2 \* 10 ) + ( 2 \* 7 )个字节数据依次为：

1个字节的主控制器(HOST)当前硬件版本号：0x30；

1个字节的主控制器(HOST)当前软件版本号：0x30；

1个字节的主控制器(HOST)当前硬件错误代码：0x00；

2个字节的主控制器(HOST)下游最多可以连接的从控制器个数：0x000A；

2\*10个字节主控制器(HOST)下游当前实际连接的从控制器ID号：0x0001、0x0002、0x0003、0x0004、0x0005、0x0006、0x0007、0x0008、0x0009、0x000A；

2个字节的主控制器(HOST)下游实际连接的从控制器个数：0x0007；

2\*7个字节主控制器(HOST)下游当前实际连接的从控制器ID号：0x0001、0x0002、0x0004、0x0005、0x0006、0x0007、0x000A；

**3.3.3 NET\_TYPE\_HEARTBEAT**

该帧由服务器发送给主控制器，由主控制器进行应答，一般服务器每隔1分钟发送1次，主要用于判断主控制器与服务器直接连接是否正常、主控制器与下游从控制器连接是否正常。

服务器🡪主控制器：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 格式 | 帧头 | 帧类型 | 数据包长度 | 数据 | 帧尾 |
| 符号 | **SOF** | **TYPE** | **LENGTH** | **DATA** | **EOF** |
| 数据 | 0xA0-0xA4 | 0x02 | 0x00、0x01 | 0x78 | 0xB0-0xB4 |

其中：

TYPE：占1个字节，当前帧为心跳帧，其码值为0x02；

LENGTH：占2个字节，由于后续数据占1个字节，该值依次为：0x00、0x01；

DATA：占1个字节，可以是任意数据，主控制器接收到该字节后会将其取反然后作为第一个字节数据返回；

主控制器🡪服务器：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 格式 | 帧头 | 帧类型 | 数据包长度 | 数据 | 帧尾 |
| 符号 | **SOF** | **TYPE** | **LENGTH** | **DATA** | **EOF** |
| 数据 | 0xA0-0xA4 | 0x82 | 0x??、0x?? | N个字节 | 0xB0-0xB4 |

其中：

TYPE：占1个字节，当前帧为应答帧，其码值为0x82；

LENGTH：占2个字节，由于后续数据为1 + NUM个字节，该值为：0x??、0x??，其中NUM为主控制器下游允许连接的从控制器个数；

DATA：占1 + NUM个字节，依次为：

第1个字节为下传数据取反；

第2个字节开始为NUM个下游从控制器连接状态，0x00为连接正常，0x01为连接异常。

**3.3.4 NET\_TYPE\_ERR\_RESP**

主控制或服务器接收到对方的帧后，会对帧进行校验等判断，如果该帧存在异常或执行失败，则通过该帧向对方进行应答。该帧可以是主控制器发向服务器的，也可以是服务器发向主控制器的。

服务器🡪主控制器：

主控制器🡪服务器：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 格式 | 帧头 | 帧类型 | 数据包长度 | 数据 | 帧尾 |
| 符号 | **SOF** | **TYPE** | **LENGTH** | **DATA** | **EOF** |
| 数据 | 0xA0-0xA4 | 0xE? | 0x00、0x01 | 1个字节 | 0xB0-0xB4 |

其中：

TYPE：占1个字节，当前帧错误应答帧，其码值为0xE?；

LENGTH：占2个字节，由于后续数据为1个字节，则该部分值为：0x00、0x01；

DATA：占1个字节，为当前接收到**帧的错误状态**或**执行错误状态**。

错误应答帧错误码表

|  |  |  |  |
| --- | --- | --- | --- |
| **序号** | **标志** | **码值** | **说明** |
| 1 | NET\_RESP\_SUCCESS | 0x00 | **帧正常，执行成功返回** |
| 2 | NET\_RESP\_ERR\_SOF | 0x01 | **帧头错误** |
| 3 | NET\_RESP\_ERR\_TYPE | 0x02 | **帧类型错误**，不在预定义的帧类型中 |
| 4 | NET\_RESP\_ERR\_LENGTH | 0x03 | **长度校验错误** |
| 5 | NET\_RESP\_ERR\_DATA | 0x04 | **数据错误**，不符合当前格式 |
| 6 | NET\_RESP\_ERR\_CHKSUM | 0x05 | **校验和错误** |
| 7 | NET\_RESP\_ERR\_EOF | 0x06 | **帧尾错误** |
| 8 | NET\_RESP\_ERR\_OP | 0x07 | **帧正常，执行失败返回** |

**3.3.5 NET\_TYPE\_RESP**

主控制器(HOST)执行完毕或从控制器(DEVICE) 执行完毕后，向服务器进行应答。

具体见命令帧NET\_TYPE\_MID\_TRANS、命令帧NET\_TYPE\_GET\_ALL\_INFO、命令帧NET\_TYPE\_HEARTBEAT对应的应答帧。

**3.3.6** NET \_TYPE\_RS232\_ TRANS

该帧由服务器发送给主控制器(HOST)，再由HOST通过串口转发给投影机等第三方设备，得到反馈数据后，再由HSOT发送给SERVER对应的应答。

服务器🡪主控制器：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 格式 | 帧头 | 帧类型 | 数据包长度 | 数据 | 帧尾 |
| 符号 | **SOF** | **TYPE** | **LENGTH** | **DATA** | **EOF** |
| 数据 | 0xA0-0xA4 | 0x03 | 2个字节 | N个字节 | 0xB0-0xB4 |

其中：

TYPE：占1个字节，**串口转发帧**，其码值为0x03；

LENGTH：占2个字节，具体值根据后面DATA数据区字节数而定，高字节在前，假设DATA区有100个字节数据，则该2个字节依次为：0x00、0x64；

DATA：占N个字节，具体根据实际而定。注：DATA区中的数据内容根据投影或者第三方串口控制设备相关通信协议代码给出。

主控制器🡪服务器：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 格式 | 帧头 | 帧类型 | 数据包长度 | 数据 | 帧尾 |
| 符号 | **SOF** | **TYPE** | **LENGTH** | **DATA** | **EOF** |
| 数据 | 0xA0-0xA4 | 0x83 | 2个字节 | N个字节 | 0xB0-0xB4 |

其中：

TYPE：占1个字节，当前帧为应答帧，其码值为0x83；

LENGTH：占2个字节，具体值根据DATA区真正功能而定；

DATA：占N个字节，具体值根据DATA区真正功能而定。注：DATA区中的数据内容根据投影或者第三方串口控制设备相关通信协议代码给出。

# 四. 主控制器与从控制器之间的通信协议

## 4.1. 帧类型

主控制器(HOST)与从控制器(DEVICE)之间的通信也为主从方式，主控制器为主机，从控制器为从机。通信以帧为单位，即通信的每包数据称为1个帧。

通信帧分为两种类型，由主控制器发送到从控制器的称为：命令帧；由从控制器发送到主控制器的称为：应答帧。实际上由主控制器(HOST)发送到从控制器(DEVICE)的命令帧可以分为三种：

(1)、命令包，即发送给某个从控制器(DEVICE)的控制命令(开关灯、空调控制)；

(2)、咨询包，即从某个从控制器(DEVICE)咨询传感器的数据，温度、电量等；

(3)、控制包，比如初始化等控制；

一般情况下，通信都是由主控制器(HOST)发起，从控制器(DEVICE)进行被动应答。主控制器在500mS内接收不到从控制器应答或者应答信息错误，则认为本次通信失败。

接收方收到对方发送的帧后，先判断该帧是否正确，包括帧头、主控制器地址、终端地址、帧类型、长度校验、累加和校验，如果其中有异常，则按照帧类型中的“错误应答帧”进行返回，如果校验正常，则执行该帧。

## 4.2. 帧格式

命令帧及应答帧的数据格式如下：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 序号 | 1 | 2 | 3 | 4 | 5 |
| 格式 | 帧头 | 帧类型 | 长度 | 数据 | 校验和 |
| 符号 | **HEAD** | **F\_TYPE** | **LENGTH** | **INFO** | **SUM** |
| 字节 | 4 | 1 | 1 | N | 1 |

各个部分具体意义见下面叙述：

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 符号 | 字节数 | 简单说明 |
| 1 | HEAD | 4 | **帧头**  每帧数据均以此4个字节开头，用于表示一帧的开始，其值依次为0xAA、地址高字节、地址低字节、0x55。  注：该4个字节的帧头是ZIGBEE模块厂家约定的，必须按此格式发送，否则模块无法识别。 |
| 2 | F\_TYPE | 1 | **帧类型**  由主控制器发送到从控制器，范围：0x00-0x7F；  由从控制器发送到主控制器，范围：0x81-0xFF；  错误应答帧：0x80；  具体帧类型及对应的码值见“帧类型码表”。 |
| 3 | LENGTH | 1 | **数据包长度**  占1个字节，有效范围为：0x00---0xFF |
| 4 | INFO | N | **数据包**  命令帧：命令数据信息(CMD INFO)；  应答帧：应答数据信息(RESP INFO)；  占N个字节，N有效范围为：0x00----0xFF |
| 5 | SUM | 1 | **校验和**  该部分为前面所有数据的累加和，即将前面所有数据进行相加，每次相加时只取最低8位。  SUM = HEAD + F\_TYPE + LENGTH + INFO |

帧类型码表

|  |  |  |  |
| --- | --- | --- | --- |
| **序号** | **标志** | **码值** | **说明** |
| 1 | ZIG\_TYPE\_CHECK\_CONNECT | 0x01 | 当前帧为：**连接检测帧**  该帧主要服务器通过主控制器(HOST)查询下游连接的从控制器(DEVICE)状态。  主控制器本身也会定时通过该命令依次查询下游连接的所有从控制器是否存在，然后统一上报服务器。  从控制器接收到该命令后，需要向主控制器上报本身的硬件版本号、软件版本号、错误状态码。 |
| 2 | ZIG\_TYPE\_IR\_CONTORL | 0x02 | 当前帧为：**红外码控制帧**  该帧主要用于控制从控制器(DEVICE)发送第几个红外编码数据，执行红外控制动作。 |
| 3 | ZIG\_TYPE\_IO\_CONTROL | 0x03 | 当前帧为：**IO输出控制帧**  该帧主要用于控制从控制器(DEVICE)对应的IO口输出高、低电平。 |
| 4 | ZIG\_TYPE\_GET\_SENSOR | 0x04 | 当前帧为：**传感器信息获取帧**  该帧主要用于控制从控制器(DEVICE)读取并返回对应传感器的信息，包括温度、湿度、IO口状态、红外码、电量等。 |
| 5 | ZIG\_TYPE\_XXX | 0x05 | 当前帧为：**预留帧**  具体功能根据需要再增加 |
| 6 | ZIG\_TYPE\_ERR\_RESP | 0xE? | 当前帧为：**错误应答帧**  主控制器(HOST)或从控制器(DEVICE)接收到对方的帧后，会对帧进行校验等判断，如果该帧存在异常或执行失败，则通过该帧向对方进行应答。 |
| 7 | ZIG\_TYPE\_RESP | 0x8? | 当前帧为：**应答帧**  主控制器(HOST)或从控制器(DEVICE)接收到对方的帧后，会对帧进行校验等判断，如果该帧存在正常或执行成功，则通过该帧向对方进行应答。 |

## 4.3. 帧类型说明

**4.3.1 ZIG\_TYPE\_CHECK\_CONNECT**

该帧主要用于主控制器(HOST)查询下游连接的所有从控制器(DEVICE)状态。主控制器会定时查询从控制器连接状态，然后在网络心跳包时上报服务器当前所有DEVICE的连接状态。从控制器接收到该命令后，需要向主控制器上报本身硬件版本号、软件版本号、硬件错误代码。假设主控制器下游最多连接16个从控制器，则主控制器会依次向0x0001---0x0010地址发送该命令，等待各自返回应答数据。

主控制器🡪从控制器器：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 序号 | 1 | 2 | 3 | 4 | 5 |
| 格式 | 帧头 | 帧类型 | 长度 | 数据 | 校验和 |
| 符号 | **HEAD** | **F\_TYPE** | **LENGTH** | **INFO** | **SUM** |
| 数据 | 0xAA、  地址高字节、地址低字节、0x55 | 0x01 | 0x00 | 无 | 0x?? |

其中：

HEAD: 占4个字节，帧头部分，依次为0xAA、地址高字节、地址低字节、0x55；

F\_TYPE：占1个字节，当前帧为连接检测帧，其码值为0x01；

LENGTH：占1个字节，该帧不带后续数据，则该字节为: 0x00；

DATA：占0个字节。

从控制器🡪主控制器：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 序号 | 1 | 2 | 3 | 4 | 5 |
| 格式 | 帧头 | 帧类型 | 长度 | 数据 | 校验和 |
| 符号 | **HEAD** | **F\_TYPE** | **LENGTH** | **INFO** | **SUM** |
| 数据 | 0xAA、  地址高字节、地址低字节、0x55 | 0x81 | 0x03 | 3个字节数据 | 0x?? |

其中：

F\_TYPE：占1个字节，当前帧为应答帧，其码值为0x81；

LENGTH：占1个字节，由于后续数据为3个字节，该值为：0x03；

DATA：占3个字节，依次为：

1个字节当前从控制器硬件版本号；

1个字节当前从控制器软件版本号；

1个字节当前从控制器硬件错误代码；

**4.3.2 ZIG\_TYPE\_IR\_CONTORL**

该帧主要用于主控制器(HOST)控制从控制器(DEVICE)发送第几个红外编码数据，执行红外控制动作。从控制器学习的红外编码数据存储在外接的串口FALSH中，存储按分区进行。从控制器接收到该命令后，从指定的分区中读取红外编码数据进行发送。

主控制器🡪从控制器器：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 序号 | 1 | 2 | 3 | 4 | 5 |
| 格式 | 帧头 | 帧类型 | 长度 | 数据 | 校验和 |
| 符号 | **HEAD** | **F\_TYPE** | **LENGTH** | **INFO** | **SUM** |
| 数据 | 0xAA、  地址高字节、地址低字节、0x55 | 0x02 | 0x01 | 1个字节数据 | 0x?? |

其中：

HEAD: 占4个字节，帧头部分，依次为0xAA、地址高字节、地址低字节、0x55；

F\_TYPE：占1个字节，当前帧为红外码控制帧，其码值为0x02；

LENGTH：占1个字节，该帧带1个字节后续数据，则字节为: 0x01；

DATA：占1个字节，有效范围为：0x01----0xFF，其值表示红外数据编号，比如当前值为0x01，则表示发送第1个学习到的红外编码数据；比如当前值为0x05，则表示发送第5个学习到的红外编码数据；

从控制器🡪主控制器：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 序号 | 1 | 2 | 3 | 4 | 5 |
| 格式 | 帧头 | 帧类型 | 长度 | 数据 | 校验和 |
| 符号 | **HEAD** | **F\_TYPE** | **LENGTH** | **INFO** | **SUM** |
| 数据 | 0xAA、  地址高字节、地址低字节、0x55 | 0x82 | 0x01 | 1个字节数据 | 0x?? |

其中：

F\_TYPE：占1个字节，当前帧为应答帧，其码值为0x82；

LENGTH：占1个字节，由于后续数据为1个字节，该值为：0x01；

DATA： 占1个字节，返回一个字节的当前命令执行状态；

**4.3.3 ZIG\_TYPE\_IO\_CONTROL**

该帧主要用于主控制器(HOST)控制从控制器(DEVICE)对应的IO口输出高、低电平。当前从控制器有3个IO口可以进行输出高低电平控制。

主控制器🡪从控制器器：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 序号 | 1 | 2 | 3 | 4 | 5 |
| 格式 | 帧头 | 帧类型 | 长度 | 数据 | 校验和 |
| 符号 | **HEAD** | **F\_TYPE** | **LENGTH** | **INFO** | **SUM** |
| 数据 | 0xAA、  地址高字节、地址低字节、0x55 | 0x03 | 0x02 | 2个字节数据 | 0x?? |

其中：

HEAD: 占4个字节，帧头部分，依次为0xAA、地址高字节、地址低字节、0x55；

F\_TYPE：占1个字节，当前帧为IO输出帧，其码值为0x03；

LENGTH：占1个字节，该帧带2个字节后续数据，则字节为: 0x02；

DATA：占2个字节

注：(1) 第一个字节为0：表示控制带开关的电柜模块，这时，第二个字节每一位（bit）对应一个开关，0为关，1为开。

（2）第一个字节为1,2,3：表示控制IO口，第一个字节IO口号有效范围：0x01---0x03；第二个字节状态：0开 1 关。

（3）第一个字节为4：表示控制投影机,第二个字节表示对投影机的控制动作 1：开投影机 2：关投影机 。

从控制器🡪主控制器：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 序号 | 1 | 2 | 3 | 4 | 5 |
| 格式 | 帧头 | 帧类型 | 长度 | 数据 | 校验和 |
| 符号 | **HEAD** | **F\_TYPE** | **LENGTH** | **INFO** | **SUM** |
| 数据 | 0xAA、  地址高字节、地址低字节、0x55 | 0x83 | 0x01 | 1个字节数据 | 0x?? |

其中：

F\_TYPE：占1个字节，当前帧为应答帧，其码值为0x83；

LENGTH：占1个字节，由于后续数据为1个字节，该值为：0x01；

DATA： 占1个字节，返回一个字节的当前命令执行状态；

**4.3.4 ZIG\_TYPE\_GET\_SENSOR**

该帧主要用于主控制器控制从控制器(DEVICE)读取并返回对应传感器的信息，包括温度、湿度、IO口状态、红外码、电量等。

主控制器🡪从控制器器：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 序号 | 1 | 2 | 3 | 4 | 5 |
| 格式 | 帧头 | 帧类型 | 长度 | 数据 | 校验和 |
| 符号 | **HEAD** | **F\_TYPE** | **LENGTH** | **INFO** | **SUM** |
| 数据 | 0xAA、  地址高字节、地址低字节、0x55 | 0x04 | 0x01 | 1个字节数据 | 0x?? |

其中：

F\_TYPE：占1个字节，当前帧为传感器信息获取帧，其码值为0x04；

LENGTH：占1个字节，该帧带1个字节后续数据，则该字节为:0x01；

DATA：占1个字节，表示当前要获取的传感器类型，

0x00：表示获取温度值；

0x01：表示获取湿度值；

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 序号 | 1 | 2 | 3 | 4 | 5 |
| 格式 | 帧头 | 帧类型 | 长度 | 数据 | 校验和 |
| 符号 | **HEAD** | **F\_TYPE** | **LENGTH** | **INFO** | **SUM** |
| 数据 | 0xAA、  地址高字节、地址低字节、0x55 | 0x84 | 0x05 | 5个字节数据 | 0x?? |

F\_TYPE：占1个字节，当前帧为应答帧，其码值为0x84；

LENGTH：占1个字节，后续数据为5个字节，其值为：0x05；

DATA：占5个字节，依次为：

1个字节的传感器类型，有效范围为：0x00---0x01，见上面描述；

2个字节的对应传感器值，如果当前为IO口状态，前1个字节固定为0x00，后1个字节当前IO口电平。采用BIT0、BIT1、BIT2表示对于3个IO

口状态，0为低电平，1为高电平。

2 个字节的对应 DEVICE ID,高位在前，低位在后。

0x02：表示获取IO口电平值；

0x04：表示获取电量值；

从控制器🡪主控制器：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 序号 | 1 | 2 | 3 | 4 | 5 |
| 格式 | 帧头 | 帧类型 | 长度 | 数据 | 校验和 |
| 符号 | **HEAD** | **F\_TYPE** | **LENGTH** | **INFO** | **SUM** |
| 数据 | 0xAA、  地址高字节、地址低字节、0x55 | 0x84 | 0x03 | 3个字节数据 | 0x?? |

F\_TYPE：占1个字节，当前帧为应答帧，其码值为0x84；

LENGTH：占1个字节，后续数据为3个字节，其值为：0x03；

DATA：占3个字节，依次为：

1个字节的传感器类型，有效范围为：0x02或 0x04，02对应继电器状态，04对应电量值见上面描述；

2个字节的对应传感器值，为继电器状态时，依次为00，0X，X为继电器状态的2进制表示形式，例如X = 3，则为0011B（二进制0011），则代表3、4继电器开，1、2继电器关例如X = 4，则为0100B，代表2继电器开，1、3、4继电器关；为电量值时依次为高位，低位，以16进制表示，例如12 33表示电量数值为0x1233 = 4659。

**4.3.5 ZIG\_TYPE\_ERR\_RESP**

主控制或从控制器接收到对方的帧后，会对帧进行校验等判断，如果该帧存在异常或执行失败，则通过该帧向对方进行应答。

主控制器🡪从控制器：

从控制器🡪主控制器：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 序号 | 1 | 2 | 3 | 4 | 5 |
| 格式 | 帧头 | 帧类型 | 长度 | 数据 | 校验和 |
| 符号 | **HEAD** | **F\_TYPE** | **LENGTH** | **INFO** | **SUM** |
| 数据 | 0xAA、  地址高字节、地址低字节、0x55 | 0xE? | 0x01 | 1个字节数据 | 0x?? |

其中：

F\_TYPE：占1个字节，当前帧为错误应答帧，其码值为0x80；

LENGTH：占1个字节，后续数据为1个字节；

DATA：占1个字节，该字节为应答帧错误码，具体见下表：

错误应答帧错误码表

|  |  |  |  |
| --- | --- | --- | --- |
| **序号** | **标志** | **码值** | **说明** |
| 1 | ZIG\_RESP\_SUCCESS | 0x00 | **帧正常，执行成功返回** |
| 2 | ZIG \_RESP\_ERR\_HEAD | 0x01 | **帧头错误** |
| 3 | ZIG \_RESP\_ERR\_ADDR | 0x02 | **从控制器地址错误** |
| 4 | ZIG \_RESP\_ERR\_F\_TYPE | 0x03 | **帧类型错误**，不在预定义的帧类型中 |
| 5 | ZIG \_RESP\_ERR\_LENGTH | 0x04 | **数据长度错误** |
| 6 | ZIG \_RESP\_ERR\_INFO | 0x05 | **数据错误**，不符合本包数据格式 |
| 7 | ZIG \_RESP\_ERR\_CHKSUM | 0x06 | **校验和错误** |
| 8 | ZIG \_RESP\_ERR\_VER | 0x07 | **版本错误**，可能不兼容 |
| 9 | ZIG \_RESP\_ERR\_OP | 0x08 | **帧正常，执行失败返回** |

# 五. 主\从控制器与电脑参数配置协议

每个主控制器或者从控制器在正式使用前，都必须通过USB接口与电脑连接，采用电脑端的软件进行参数配置，这些参数包括主从控制内部的网络相关参数、ZIGBEE模块相关参数等。配置的参数保存在模块主控芯片内部的FLASH中，永久保存，掉电不丢失。

## 5.1. 帧类型

主从控制器的USB接口与PC机连接，在电脑的设备管理器中会多出一个标准的USB自定义HID类设备，不需要安装任何驱动即可实现数据的上、下传通信，从而实现参数配置功能。

数据的传输以帧为单位，即以数据包的形式发送，每帧数据都带有命令码、后续数据长度以及后续数据。如果模块接收到错误帧，会返回错误应答帧或者将其直接丢弃。

以下将先PC机发起的通信帧称为命令帧，模块返回的通信帧称为应答帧。对于命令包，PC机发送之后，需要等待模块返回应答帧，根据应答包来确定本次命令是否执行成功。如果返回错误状态或接收不到应答包，则需要根据情况进行重试或者出错处理。

## 5.2. 帧格式

命令帧及应答帧的数据格式如下(大端格式)：

|  |  |  |  |
| --- | --- | --- | --- |
| **序号** | 1 | 2 | 3 |
| **格式** | **命令码** | **后续数据长度** | **后续数据** |
| **符号** | CMD | LEN | DATA |
| **字节数** | 1个字节 | 1个字节 | N个字节(0-30) |

各个部分具体意义见下面叙述：

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 符号 | 字节数 | 简单说明 |
| 1 | CMD | 1 | **命令码**  该部分占1个字节，不同的命令码包含不同的功能，具体见下面命令码表及详细解释。  计算机发起的命令帧的命令码有效范围：0xA0---0xAF；  模块返回对应应答帧的命令码有效返回为：0xC0---0xCF；  即应答帧命令码 = 命令帧命令码+0x20。 |
| 2 | LEN | 1 | **后续数据长度**  该部分占1个字节，主要用于记录该包实际后续有效数据的长度 |
| 3 | DATA | 1 | **后续数据**  该部分占N个字节，N有效范围为：0----30，即最多可以包含30个字节。 |

## 5.3. 帧类型说明

命令码表：

|  |  |  |  |
| --- | --- | --- | --- |
| **序号** | **命令名称** | **命名码** | **命令说明** |
| 1 | CMD\_CHECK\_CONNECT | 0xA0 | **检测当前模块连接情况** |
| 2 | CMD\_GET\_INFO | 0xA1 | **获取当前模块程序版本等信息** |
| 3 | CMD\_SET\_DEVICE\_INFO | 0xA2 | **设置主控制器下游从控制器个数**  **注：该命令只对主控制器有效** |
| 4 | CMD\_GET\_ZIG\_INFO | 0xA3 | **获取当前模块内部ZIGBEE模块相关参数信息** |
| 5 | CMD\_SET\_ZIG\_INFO | 0xA4 | **设置当前模块内部ZIGBEE模块相关参数信息** |
| 6 | CMD\_GET\_NET\_INFO | 0xA5 | **获取当前模块内部网络芯片CH395相关参数信息** |
| 7 | CMD\_SET\_NET\_INFO | 0xA6 | **设置当前模块内部网络芯片CH395相关参数信息** |
| 8 | CMD\_MODULE\_RESET | 0xA7 | **复位模块** |

**5.3.1 CMD\_CHECK\_CONNECT**

该命令用于计算机检测当前模块与计算机连接是否成功，以及检测模块运行是否正常。一般情况下，在开始运行配置软件时发送一次该命令即可。

计算机🡪模块：

|  |  |  |  |
| --- | --- | --- | --- |
| **格式** | **命令码** | **后续数据长度** | **后续数据** |
| **符号** | CMD | LEN | DATA |
| **数据** | 0xA0 | 0x01 | 1个字节任意，比如0x78 |

该命令帧带1个字节的任意数据，模块接收到该命令帧后，将该数据取反，再通过应答帧返回计算机。

模块🡪计算机：

|  |  |  |  |
| --- | --- | --- | --- |
| **格式** | **命令码** | **后续数据长度** | **后续数据** |
| **符号** | CMD | LEN | DATA |
| **数据** | 0xC0 | 0x01 | 1个字节取反数据，比如0x87 |

例如：

计算机发向模块的命令帧为：0xA0、0x01、0x78；则模块发向计算机的应答帧为：0xC0、0x01、0x87；

**5.3.2 CMD\_GET\_INFO**

该命令用于计算机获取模块的硬件版本号、程序版本号、硬件错误状态、主控制器下游从控制器个数以及地址范围区间等信息。

计算机🡪模块：

|  |  |  |  |
| --- | --- | --- | --- |
| **格式** | **命令码** | **后续数据长度** | **后续数据** |
| **符号** | CMD | LEN | DATA |
| **数据** | 0xA1 | 0x00 | 无数据 |

该命令不带任何参数。

模块🡪计算机：

|  |  |  |  |
| --- | --- | --- | --- |
| **格式** | **命令码** | **后续数据长度** | **后续数据** |
| **符号** | CMD | LEN | DATA |
| **数据** | 0xC1 | 0x09 | 9个字节后续数据 |

返回的5个字节后续数据依次为：

1个字节模块硬件版本号；

1个字节模块程序版本号；

1个字节模块当前硬件错误状态；

2个字节模块下游从控制器个数(高字节在前，如果当前模块为从控制器，则该2个字节必须是0x00、0x00)；

2个字节模块下游从控制器地址(ADDRID)范围区间开始值(高字节在前，如果当前模块为从控制器，则该2个字节必须是0x00、0x00)；

2个字节模块下游从控制器地址(ADDRID)范围区间结束值(高字节在前，如果当前模块为从控制器，则该2个字节必须是0x00、0x00)；

举例：如果当前当前硬件版本为V1.3，当前程序版本为V1.0，当前模块为主控制器，无硬件错误，下游最多允许连接100个从控制器模块，并且从控制器模块的地址范围区间为：0x0001----0x0064，则返回的9个字节数据依次为：

0x33、0x30、0x00、0x00、0x64、0x00、0x01、0x00、0x64。

**5.3.3 CMD\_SET\_DEVICE\_INFO**

该命令用于计算机设置主控制器模块(HOST)下游可连接的从控制器模块(DEVICE)个数(NUM)以及从控制器通信地址范围区间(ADDRID)，高字节在前。

计算机🡪模块：

|  |  |  |  |
| --- | --- | --- | --- |
| **格式** | **命令码** | **后续数据长度** | **后续数据** |
| **符号** | CMD | LEN | DATA |
| **数据** | 0xA2 | 6 | 6个字节 |

该命令带有6个字节的后续数据，依次为：

2个字节的下游从控制器连接个数(NUM)；

2个字节下游从控制器通信地址(ADDRID)范围区间开始值；

2个字节下游从控制器通信地址(ADDRID) 范围区间结束值；

为方便从控制器模块地址分配以及程序处理，从控制器的通信地址按区间进行分配，建议每个子网络(比如一个房间)内的从控制器地址均由0x0001开始依次加1递增进行分配。

比如当前子网络需要连接10个从控制器，则通信地址分配范围为：0x0001----0x000A，即10个从控制器模块的地址分别为：0x0001、0x0002、0x0003、0x0004、0x0005、0x0006、0x0007、0x0008、0x0009、0x000A。

当前命令包所带的后续数据字节依次为：

2个字节的下游从控制器连接个数(NUM)：0x00、0x0A；

2个字节下游从控制器通信地址(ADDRID)范围区间开始值：0x00、0x01；

2个字节下游从控制器通信地址(ADDRID) 范围区间结束值：0x00、0x0A；

模块🡪计算机：

|  |  |  |  |
| --- | --- | --- | --- |
| **格式** | **命令码** | **后续数据长度** | **后续数据** |
| **符号** | CMD | LEN | DATA |
| **数据** | 0xC2 | 0x01 | 1个字节执行状态 |

返回的1个字节后续数据为：该命令执行状态，具体见命令执行状态表。

**5.3.4 CMD\_GET\_ZIG\_INFO**

该命令用于计算机获取模块内部ZIGBEE模块相关参数信息，包括通信波特率(BAUDRATE)、私有网络ID号(PANID)和模块地址(ADDRID)。

计算机🡪模块：

|  |  |  |  |
| --- | --- | --- | --- |
| **格式** | **命令码** | **后续数据长度** | **后续数据** |
| **符号** | CMD | LEN | DATA |
| **数据** | 0xA3 | 0x00 | 无数据 |

该命令不带任何参数。

模块🡪计算机：

|  |  |  |  |
| --- | --- | --- | --- |
| **格式** | **命令码** | **后续数据长度** | **后续数据** |
| **符号** | CMD | LEN | DATA |
| **数据** | 0xC3 | 0x08 | 8个字节后续数据 |

返回的8个字节后续数据依次为：

4个字节当前ZIGBEE模块通信波特率，高字节在前，比如115200bps，则该4个字节为：0x00、0x01、0xC2、0x00；

2个字节当前ZIGBEE模块的私有网络ID号(PANID)，高字节在前，有效范围为：1---65535；

2个字节当前ZIGBEE模块的地址值(ADDRID)，高字节在前；

**5.3.5 CMD\_SET\_ZIG\_INFO**

该命令用于计算机设置模块内部ZIGBEE模块相关参数信息，包括通信波特率(BAUDRATE)、私有网络ID号(PANID)和模块地址(ADDRID)。

计算机🡪模块：

|  |  |  |  |
| --- | --- | --- | --- |
| **格式** | **命令码** | **后续数据长度** | **后续数据** |
| **符号** | CMD | LEN | DATA |
| **数据** | 0xA4 | 0x08 | 8个字节后续数据 |

该命令带8个字节的后续参数，依次为：

4个字节当前ZIGBEE模块通信波特率，高字节在前，比如115200bps，则该4个字节为：0x00、0x01、0xC2、0x00；

2个字节当前ZIGBEE模块的私有网络ID号(PANID)，高字节在前，有效范围为：1---65535；

2个字节当前ZIGBEE模块的地址值(ADDRID)，高字节在前；

注意：如果当前模块为主控制器(HOST)，则ZIGBEE模块地址(ADDRID)必须设置成0x0000；如果当前模块为从控制器(DEVICE)，则ZIGBEE模块地址(ADDRID)必须设置成非0x0000；

模块🡪计算机：

|  |  |  |  |
| --- | --- | --- | --- |
| **格式** | **命令码** | **后续数据长度** | **后续数据** |
| **符号** | CMD | LEN | DATA |
| **数据** | 0xC4 | 0x01 | 1个字节执行状态 |

返回的1个字节后续数据为：该命令执行状态，具体见命令执行状态表。

**5.3.6 CMD\_GET\_NET\_INFO**

该命令用于计算机获取模块内部网络芯片CH395相关参数信息，包括CH395 IP地址(CH395IPAddr)、CH395网关地址(CH395GWIPAddr)、CH395子网掩码(CH395IPMask)、CH395 Socket0 目的IP地址(Socket0DesIP)、CH395 Socket目的端口号(Socket0DesPort )和CH395 Socket源端口号(Socket0SourPort)。

计算机🡪模块：

|  |  |  |  |
| --- | --- | --- | --- |
| **格式** | **命令码** | **后续数据长度** | **后续数据** |
| **符号** | CMD | LEN | DATA |
| **数据** | 0xA5 | 0x00 | 无数据 |

该命令不带任何参数。

模块🡪计算机：

|  |  |  |  |
| --- | --- | --- | --- |
| **格式** | **命令码** | **后续数据长度** | **后续数据** |
| **符号** | CMD | LEN | DATA |
| **数据** | 0xC5 | 0x14 | 20个字节后续数据 |

返回的20个字节后续数据依次为：

4个字节的CH395 IP地址(CH395IPAddr)，该IP地址是模块本身的IP地址，默认值为：192.168.1.100，即该4个字节默认为：0xC0、0xA8、0x01、0x64；

4个字节的CH395网关地址(CH395GWIPAddr)，默认值为：192.168.1.1，即该4个字节默认为：0xC0、0xA8、0x01、0x01；

4个字节的CH395子网掩码(CH395IPMask)，默认值为：255.255.255.0，即该4个字节默认为：0xFF、0xFF、0xFF、0x00；

4个字节的CH395 Socket0 目的IP地址(Socket0DesIP)，该IP地址是模块想要连接的电脑或者服务器本身的IP地址，默认值为：192.168.1.1，即该4个字节默认为 ：0xC0、0xA8、0x01、0x01；

2个字节的CH395 Socket目的端口号(Socket0DesPort )，该目的端口号是模块想要连接的电脑或服务器本身的端口号，默认值为：8000，即该2个字节默认为：0x1F、0x40；

2个字节的CH395 Socket源端口号(Socket0SourPort)，该源端口号是模块本身用于TCPIP通信的端口号，默认值为：5000，即该2个字节默认为：0x13、0x88，该值每次连接后会自动加1，断电后恢复成默认值；

**5.3.7 CMD\_SET\_NET\_INFO**

该命令用于计算机设置模块内部网络芯片CH395相关参数信息，包括CH395 IP地址(CH395IPAddr)、CH395网关地址(CH395GWIPAddr)、CH395子网掩码(CH395IPMask)、CH395 Socket0 目的IP地址(Socket0DesIP)、CH395 Socket目的端口号(Socket0DesPort )和CH395 Socket源端口号(Socket0SourPort)。

计算机🡪模块：

|  |  |  |  |
| --- | --- | --- | --- |
| **格式** | **命令码** | **后续数据长度** | **后续数据** |
| **符号** | CMD | LEN | DATA |
| **数据** | 0xA6 | 0x14 | 20个字节后续数据 |

该命令带的20个字节后续数据依次为：

4个字节的CH395 IP地址(CH395IPAddr)，该IP地址是模块本身的IP地址，默认值为：192.168.1.100，即该4个字节默认为：0xC0、0xA8、0x01、0x64；

4个字节的CH395网关地址(CH395GWIPAddr)，默认值为：192.168.1.1，即该4个字节默认为：0xC0、0xA8、0x01、0x01；

4个字节的CH395子网掩码(CH395IPMask)，默认值为：255.255.255.0，即该4个字节默认为：0xFF、0xFF、0xFF、0x00；

4个字节的CH395 Socket0 目的IP地址(Socket0DesIP)，该IP地址是模块想要连接的电脑或者服务器本身的IP地址，默认值为：192.168.1.1，即该4个字节默认为 ：0xC0、0xA8、0x01、0x01；

2个字节的CH395 Socket目的端口号(Socket0DesPort )，该目的端口号是模块想要连接的电脑或服务器本身的端口号，默认值为：8000，即该2个字节默认为：0x1F、0x40；

2个字节的CH395 Socket源端口号(Socket0SourPort)，该源端口号是模块本身用于TCPIP通信的端口号，默认值为：5000，即该2个字节默认为：0x13、0x88，该值每次连接后会自动加1，断电后恢复成默认值；

模块🡪计算机：

|  |  |  |  |
| --- | --- | --- | --- |
| **格式** | **命令码** | **后续数据长度** | **后续数据** |
| **符号** | CMD | LEN | DATA |
| **数据** | 0xC6 | 0x01 | 1个字节执行状态 |

返回的1个字节后续数据为：该命令执行状态，具体见命令执行状态表。

**5.3.8 CMD\_MODULE\_RESET**

该命令用于计算机复位模块，模块接收到该命令后，会进行软件复位，断开USB连接，从上电处重新开始运行，该命令无应答包反应。

一般在设置完参数之后，可以发送该命令，以便模块以新参数重新工作。

计算机🡪模块：

|  |  |  |  |
| --- | --- | --- | --- |
| **格式** | **命令码** | **后续数据长度** | **后续数据** |
| **符号** | CMD | LEN | DATA |
| **数据** | 0xA7 | 0x00 | 无数据 |

该命令不带任何参数。

模块🡪计算机：

无应答帧

## 5.4. 错误应答帧

模块接收到的命令帧如果存在命令码错误、数据长度错误、参数错误、或者执行失败等问题时，则需要通过错误应答帧进行应答。错误应答帧包含1个字节后续数据，该数据即为命令执行状态。

模块🡪计算机：

|  |  |  |  |
| --- | --- | --- | --- |
| **格式** | **命令码** | **后续数据长度** | **后续数据** |
| **符号** | CMD | LEN | DATA |
| **数据** | 0xEE | 0x01 | 1个字节执行状态 |

返回的1个字节后续数据为：该命令执行状态，具体见命令执行状态表。

表2-命令执行状态表

|  |  |  |
| --- | --- | --- |
| **状态名称** | **状态码** | **状态说明** |
| DEF\_RET\_SUCCESS | 0x00 | 命令执行成功 |
| DEF\_RET\_ERR\_CMD | 0xE1 | 命令码错误 |
| DEF\_RET\_ERR\_LEN | 0xE2 | 数据长度错误 |
| DEF\_RET\_ERR\_PARA | 0xE3 | 参数错误，比如超出范围 |
| DEF\_RET\_ERR\_OP | 0xE4 | 命令执行失败，比如写入失败 |