

**MacBee路灯RF协议规范**

**版本号V004**

**深圳市银河风云网络系统股份有限公司**

**Shenzhen GALAXYWIND Network Systems Co., Ltd.**

地址：深圳市高新技术产业园区北区新西路5号银河风云大厦

Address: GALAXYWIND building, No.5 Xinxi road, Shenzhen High-Tech Industry Park, NanShan district, China

邮编 P.C.: 518055

电话 Tel: +86-755-83400088

传真 FAX: +86-755-86139063

网站 Web: http://[www.galaxywind.com](http://www.galaxywind.com/)

客服 Hotline: 4000009879

企业QQ: 4000009879

微信公众号 WeChat:



**版权声明**

版权所有©深圳市银河风云网络系统股份有限公司2017。深圳市银河风云网络系统股份有限公司（以下简称"银河风云"）对本资料进行版权声明，未经银河风云书面许可，任何单位及个人不得以任何方式或理由对本资料的任何内容进行复制、修改、抄录、传播。

**商标声明**

"银河风云"，"GALAXYWIND"等是深圳市风云实业有限公司的注册商标，本文档提及的其他商标、服务标志、注册商标及注册服务标志均为其所有者拥有。

**免责声明**

本资料会进行不定期更新，本资料中的所有陈述、信息和建议不构成任何明示或暗示的担保。

目录

[目录 1](#_Toc23555)

[前言 3](#_Toc23760)

[1 Macbee路灯协议设计需求 5](#_Toc2893)

[2 Macbee路灯系统设计 6](#_Toc604)

[2.1 系统架构图 6](#_Toc5647)

[2.2 信号传输抽象说明 6](#_Toc12038)

[3 几个比较关注的问题 7](#_Toc27615)

[3.1 路灯位置定位 7](#_Toc21062)

[3.2 通信安全 7](#_Toc27562)

[3.3 由于报文采用中继方式传递，如果中间某个设备坏了，是否后面的设备就会掉线？ 7](#_Toc3937)

[3.4 多主机网络 7](#_Toc4997)

[3.5 协议流程状态机 8](#_Toc15475)

[3.6 保活和状态同步 8](#_Toc22538)

[4 Macbee报文定义 9](#_Toc24277)

[4.1 报文类型定义 9](#_Toc10157)

[4.2 各种报文类型对应的报文头部定义： 9](#_Toc14245)

[4.3 报文格式定义： 10](#_Toc432)

[4.3.1 加入网络 10](#_Toc18797)

[4.3.2 绑定 11](#_Toc1424)

[4.3.3 认证 12](#_Toc18503)

[4.3.4 邻居间通信 12](#_Toc18225)

[向邻居广播状态(Type5, cry=1, ack=0, cmd=13): 12](#_Toc12276)

[4.3.5 保活 13](#_Toc21777)

[4.3.6 其他命令交互 13](#_Toc22456)

[5 路灯升级流程 19](#_Toc2364)

[6 路灯信道设置 20](#_Toc23029)

[6.1 设备端处理流程 20](#_Toc17128)

[6.2 网关处理流程 21](#_Toc5607)

[6.3 报文定义： 21](#_Toc21977)

[7 透传报文定义 22](#_Toc999)

[7.1 亮度、冷暖光控制（sub\_cmd = TT\_SUBCMD\_SET\_LIGHT） 22](#_Toc9727)

[7.2 添加/修改定时器（sub\_cmd = TT\_SUBCMD\_SET\_TIMER） 23](#_Toc6432)

[7.3 删除定时器（sub\_cmd = TT\_SUBCMD\_DEL\_TIMER） 24](#_Toc22767)

[8 Report报文定义 25](#_Toc7585)

[8.1 设备能力（sub\_cmd = TTRPT\_SUBCMD\_DEV\_CPB） 26](#_Toc5148)

[8.2 负载信息（sub\_cmd = TTRPT\_SUBCMD\_DEV\_LOAD\_INFO） 26](#_Toc12325)

[8.3 异常上报（sub\_cmd = TTRPT\_SUBCMD\_DEV\_EXCEPT） 27](#_Toc13833)

[8.4 定时器信息（sub\_cmd = TTRPT\_SUBCMD\_DEV\_TIM\_INFO） 28](#_Toc19732)

[9 附件1 Macbee路灯协议技术指标 32](#_Toc10689)

前言

**概述**

**产品版本**

与本文档相对应的产品版本如下所示：

|  |  |
| --- | --- |
| **产品名称** | **产品版本** |
| **MacBee路灯RF协议规范** | 0.0.2 |

**阅读对象**

本文档主要适用于以下用户：

结构设计工程师【√】

硬件开发工程师【√】

软件开发工程师【√】

系统规划工程师【√】

系统维护工程师【√】

**约定**

**符号约定**

文档中可能出现的符号定义如下所示：

|  |  |
| --- | --- |
| **符号** | **说明** |
| 警告 | 表示有中度或低度潜在危险，如果不能避免，可能导致较严重结果。 |
| 注意 | 表示有潜在风险，如果忽视这些文本，可能导致不可预知的结果。 |
| 备注 | 表示对正文的强调和补充。 |

**通用格式约定**

|  |  |
| --- | --- |
| **格式** | **说明** |
| 黑体 | 中文：1 -6级标题，图表/表格标题采用黑体 |
| Arial | 英文：1 -6级标题，图表/表格标题采用Arial |
| 宋体 | 中文：正文采用宋体 |
| Times New Roman | 英文：正文采用Times New Roman |

**修改记录**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **修订时间** | **修订内容** | **相关章节** | **修订版本** | **修改人** |
| 2017/8/21 | 初稿 |  | V001 | 孙世玮 |
| 2017/9/29 | 增加信道设置模块 |  | V002 | 孙世玮 |
| 2017/10/25 | 增加定时器同步处理 |  | V003 | 孙世玮 |
| 2017/11/7 | 增加report报文定义 |  | V004 | 孙世玮 |
|  |  |  |  |  |

# Macbee路灯协议设计需求

1、传统路灯存在诸多问题

1）传统路灯基本上采用手动、光控、钟控等方式，易受季节、天气自然环境和人为因素影响；

2）传统路灯无法远程修改开关灯时间：不能根据实际情况(天气突变，重大事件，节日)及时校时和修改开关灯时间，也无法进行LED 灯调光，无法实现二次节能。

3）传统路灯不具备路灯状态监测：故障依据主要来源于巡视人员上报和市民投诉，缺乏主动性、及时性和可靠性，不能实时、准确、全面地监控全城的路灯运行状况。

4）传统路灯单一依靠普通人工巡检：管理部门缺乏统一调度的能力，只能以逐个配电柜为单元进行调整 ，不仅费时费力，而且增加了人为误操作的可能性。

2、针对传统路灯的问题，市面上慢慢出现了智慧路灯解决方案，可实现路灯设施管理、故障报警、用电监测、路灯控制和移动终端应用等功能。但这些功能依靠的通信基础也存在或多或少的问题。

目前的3种智慧路灯方案存在的问题：

1）基于电力载波（PLD）通信方式对路灯进行控制管理

1. 安装成本高，需安装单灯控制器集中控制器和电力设备
2. 通信不能跨越配电变压器；
3. 三相电力线间有很大信号损失，一般电力载波信号只能在单相电力线上传输。
4. 稳定性差，当电力线上负荷很重时，电力线会对载波信号造成高削减，造成通信距离缩短，很容易出现控制失败现象。
5. 保密性低。

2）基于WIFI对路灯进行控制管理

1. 需要wifi无缝覆盖，成本高，
2. 或者只覆盖集中控制点，无法对每个路灯做到单独控制和故障监测，更加无法为后续扩展的车流量监测、光感监测等智能化功能提供通信基础。

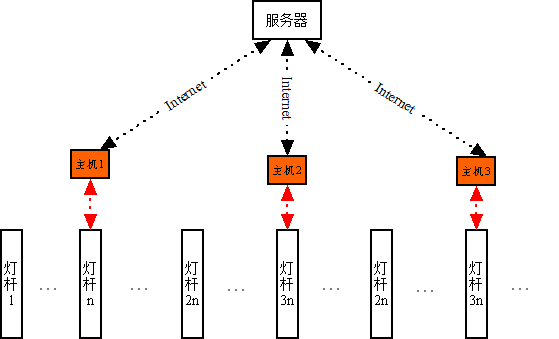
3）基于zigbee通信方式对路灯进行控制管理

1. 1、每个集中控制器下可配置的控制节点有限，最多支持16跳
2. 2、每个集中控制器相对独立，其中一个出现故障，其对应下的控制节点将失去控制
3. 3、Zigbee属于国外专利，且成本较高。

针对以上问题，提出Macbee智慧路灯很有必要。

# Macbee路灯系统设计

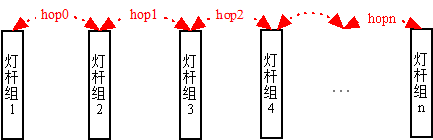
## 系统架构图



说明：1、主机通过**移动数据网络**或者wifi通信接入互联网

2、主机与灯杆之间，灯杆与灯杆之间通过macbee协议通信。

## 信号传输抽象说明



说明：

1. 由于控制信号是无线，所以上图所示的灯杆组1个灯杆组2没有特定的物理界限。仅仅是为描述信号传输过程；
2. 所有报文通过每个路灯中继的方式传输，所以控制响应时间跟路灯与主机之间的距离有很大关系。
3. 每个hop时间间隔为62.5-187.5ms；

# 几个比较关注的问题

## 路灯位置定位

每个路灯在生产时都会产生一个全球唯一的序列号和其他相关信息，我们将会把这些信息打印成二维码固定到路灯/路灯控制器上，在路灯安装时，通过具有定位功能的手机用专用APP扫描对应的二维码，APP会自动把当时的位置信息以及路灯信息上传到指定云服务器，这样便实现了路灯的位置定位。为保证位置的准确性，**app可以通过自动定位和人工确认相结合的方式来采集位置信息**。

## 通信安全

每一个路灯/路灯控制器在出厂时会有一个和云服务器对应的私有秘钥。在路灯正式接入服务器后，服务器会再次对整条管理的道路分配一个公有的秘钥。

1) 路灯入网时服务器会通过广播地址给路灯下发加密后的公共秘钥，只有对应的路灯才能通过私有秘钥解开。

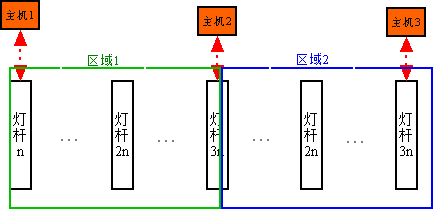
2) 路灯入网后的所有RF通信报文都通过公有秘钥加密，只有属于该管理道路的设备才能正确解密此报文。

3) 加密算法采用TEA加密算法，秘钥长度为128位。

## 由于报文采用中继方式传递，如果中间某个设备坏了，是否后面的设备就会掉线？

答案是否定的，由于无线信号在不中继的情况下可以覆盖100-200m，灯杆的安装间距最大只有40m，如果其中一个灯坏了，后面的灯还可以收到这个无线覆盖区域的其他灯发送的中继报文。

## 多主机网络



1. 为了尽量避免RF信号冲突，区域1的RF报文不会中继到区域2，区域2的RF报文也不会中继到区域1。
2. 如果主机2坏了，区域1和区域2会自动变成一个区域，整个通信仍然正常。

3、考虑到道路的健壮性，建议安装主机的位置冗余安装一个主机，并且保证在道路的两端都安装上主机。

4、考虑到整条道路的响应时效性，建议每10KM至少有安装一个主机。

## 协议流程状态机

新设备->入网->绑定->认证->正常通信

## 保活和状态同步

由于路灯管理中心需要知道当前路灯的工作状态，所以我们把路灯的状态放在保活报文中同步到服务器。由于路灯应用的特殊性，一条管理道路中的大部分灯的亮度状态是一致的，所以我们定义了专门针对路灯的保活协议。

具体保活流程：

1. 所有设备每隔30秒向邻近设备报告一致自己状态，如果设备状态发生改变则清除缓存的邻居设备状态，同时3秒内随机向邻近设备再次报告自己状态。这样保证了自己和邻居都缓存了邻居的最新状态。
2. 设备在主动或者被动上报保活报文时，附带上和自己状态一样的邻居设备信息。这样一条保活报文就携带了几个设备的状态。
3. 设备定时检查自己缓存的邻居状态，如果超过老化时间，则清除设备缓存信息。
4. 设备主动上报保活时间协商策略：保留。

# Macbee报文定义

## 报文类型定义

Type0：SRV2NET（服务器->路灯）Broadcast（广播）

Type1：SRV2NET（服务器->路灯）Multicast or Unicast（组播/单播）

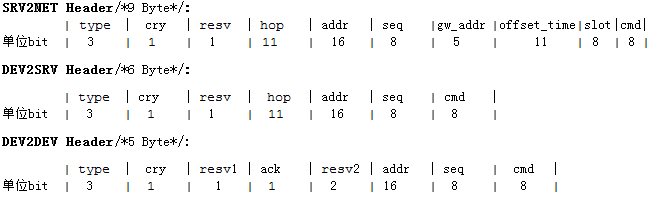
Type2：DEV2SRV（路灯->服务器）unicast ack（路灯回复服务器）

Type3：DEV2SRV（路灯->服务器）unicast report（路灯主动上报）

Type4：SRV2NET（服务器->路灯）unicast report ack（服务器回复路灯）

Type5：DEV2DEV（路灯->路灯）

## 各种报文类型对应的报文头部定义：



Header定义代码参考附件1

Macbee路灯协议技术指标见附件2

## 报文格式定义：

约定：以下报文格式描述时没有写报文header部分，只写了报文类型（Type0、Type1、Type2、Type3、Type4、Type5）,实际报文需要包含header部分。

### 加入网络

#### 申请加入(Type5, cry=0, ack=0, cmd=1):

|  |  |  |  |
| --- | --- | --- | --- |
| byte0 | byte1 | byte2 | byte3 |
| flag |  |  |  |
| sn[0] | sn[1] | sn[2] | sn[3] |
| sn[4] | sn[5] | sn[6] | sn[7] |
| resv |  |  |  |
| net\_id[0] | net\_id[1] | net\_id[2] | net\_id[3] |

长度：14

flag[0] 1: 已绑定, 有 Net ID

flag[0] 0: 未绑定, 无 Net ID

#### 申请加入应答(Type5, cry=0, ack=1, cmd=1):

|  |  |  |  |
| --- | --- | --- | --- |
| byte0 | byte1 | byte2 | byte3 |
| flag | slot | max slot |  |
| FHSS func | | offset time | |
| utc | | | |
| sn[0] | sn[1] | sn[2] | sn[3] |
| sn[4] | sn[5] | sn[6] | sn[7] |
| result | rand(opt)| | |  |
| HMAC(opt) | | | |

长度：26

flag[0] 1: 已绑定, 有 rand, HAMC(用 session key 计算)

flag[0] 0: 未绑定, 无 rand, HAMC

result 0: 成功

result !=0: 失败

收到HMAC后, 使用自己的session key计算HMAC, 吻合且有 DEV addr 就跳过认证.

#### 入网成功(类型5, cry=0, ack=0, cmd=2):

|  |  |  |  |
| --- | --- | --- | --- |
| byte0 | byte1 | byte2 | byte3 |
| sn[0] | sn[1] | sn[2] | sn[3] |
| sn[4] | sn[5] | sn[6] | sn[7] |

长度：8

### 绑定

#### 绑定请求(Type1, cmd=5):

|  |  |  |  |
| --- | --- | --- | --- |
| byte0 | byte1 | byte2 | byte3 |
| resv |  |  |  |
| net\_id[0] | net\_id[1] | net\_id[2] | net\_id[3] |
| dev\_addr | |  |  |
| sn[0] | sn[1] | sn[2] | sn[3] |
| sn[4] | sn[5] | sn[6] | sn[7] |
| random1 | | | |

长度：19

#### 绑定提问(Type2, cmd=6):

|resv|DEV addr|random2|

| 1 | 2 | 4 |

|  |  |  |  |
| --- | --- | --- | --- |
| byte0 | byte1 | byte2 | byte3 |
| resv |  |  |  |
| dev\_addr | |  |  |
| random1 | | | |

长度：7

#### 绑定答案(Type1 cmd=7):

|crypt method|DEV addr|answer|

| 1 | 2 | 16 |

长度：19绑定结果(Type2, cmd=8):

|  |  |  |  |
| --- | --- | --- | --- |
| byte0 | byte1 | byte2 | byte3 |
| result |  |  |  |
| dev\_addr | |  |  |

长度：3

result==0: 成功

result!=0: 失败

### 认证

#### 认证请求(Type3, cmd=9):

|  |  |  |  |
| --- | --- | --- | --- |
| byte0 | byte1 | byte2 | byte3 |
| resv | dev\_type | ext\_type |  |
| sn[0] | sn[1] | sn[2] | sn[3] |
| sn[4] | sn[5] | sn[6] | sn[7] |
| random1 | | | |

长度：15

#### 认证结果(Type4, cmd=12):

|result|crypt\_method|session key|

| 1 | 1 | 16|

长度：18

result==0: 成功, 有 session key, 使用算法加密

result!=0: 失败, 无 session key

### 邻居间通信

### 向邻居广播状态(Type5, cry=1, ack=0, cmd=13):

|DEV addr|TLV|TLV|...

| 2 |N1 |N2 |

每个TVL是一种类型的状态, 可以有多个(但是不建议).

设备随机发送, 控制速率大约每30秒1次. 连续2分钟未收到老化该设备.

### 保活

#### 服务器向路灯查询状态(keeplive)(Type1, cmd=14):

|  |  |  |  |
| --- | --- | --- | --- |
| byte0 | byte1 | byte2 | byte3 |
|  |  |  |  |

长度：0

#### 路灯向服务器发送keeplive应答(Type2, cmd=14):

|len | type | len | value |addr[0]|... |addr[n]|

| 1 | 1 | 1 | 4 | 2 | ... | 2 |

#### 路灯主动上报keeplive(Type3, cmd=24):

|len | type | len | value |addr[0]|... |addr[n]|

| 1 | 1 | 1 | 4 | 2 | ... | 2 |

### 其他命令交互

#### SRV->NET允许或者禁止新设备加入网络(Type0, cmd=18):

|enable|

| 1 |

#### SRV->NET命令透传(Type0或Type1, cmd=15):

|data|

| N |

#### DEV->SRV命令透传应答(对单播才有, Type2, cmd=15):

|data|

| N |

#### DEV->SRV报告透传(Type3, cmd=16):

|report\_idx|data|

| 1 | N |

#### SRV->DEV报告透传应答(Type4 cmd=16):

|report\_idx|

| 1 |

#### SRV->NET同步系统时间(Type0, cmd=23):

|utc |

| 4 |

#### SRV->NET升级报文img\_head(Type0, cmd=252):

|  |  |  |  |
| --- | --- | --- | --- |
| byte0 | byte1 | byte2 | byte3 |
| img\_size | | img\_crc | |
| dev\_type | ext\_type | maj\_min\_ver | rev\_ver |
| img\_type | snd\_time | repeat\_num |  |

长度：11

#### SRV->NET升级报文img\_data(Type0, cmd=253):

|addr\_offset |data |

| 2 | 20 |

长度：22

#### DEV->DEV向邻居请求升级报文img\_head(Type5, ack=0,cmd=252):

|  |  |  |  |
| --- | --- | --- | --- |
| byte0 | byte1 | byte2 | byte3 |
| neigb\_addr | | dev\_type | ext\_type |
| img\_type |  |  |  |

长度：5

#### DEV->DEV升级报文img\_head(Type5, ack=1,cmd=252):

|  |  |  |  |
| --- | --- | --- | --- |
| byte0 | byte1 | byte2 | byte3 |
| neigb\_addr | | rsv |  |
| img\_size | | img\_crc | |
| dev\_type | ext\_type | maj\_min\_ver | rev\_ver |
| img\_type | snd\_time | repeat\_num |  |

长度：14

#### DEV->DEV向邻居请求升级报文img\_data(Type5, ack=0,cmd=253):

|  |  |  |  |
| --- | --- | --- | --- |
| byte0 | byte1 | byte2 | byte3 |
| neigb\_addr | | dev\_type | ext\_type |
| maj\_min\_ver | rev\_ver | img\_type |  |
| img\_size | | addr\_offset | |

长度：11

#### DEV->DEV升级报文img\_data(Type5, ack=1,cmd=253):

|neigb\_addr |img\_type\_ext\_type |addr\_offset |data |

| 2 | 1 | 2 | 20 |

长度：25

**为方便服务器处理，以下几种报文不采用通用的透传cmd，采用单独定义cmd的方式处理**

#### DEV->SRV路灯主动上报版本信息(Type3, cmd=19):

|report\_idx|boot\_maj\_min |boot\_rev |app\_maj\_min |app\_rev|

| 1 | 1 | 1 | 1 | 1 |

#### SRV->NET路灯主动上报版本应答(Type4 cmd=19):

|report\_idx|

| 1 |

#### SRV->NET服务器查询路灯版本信息(Type1, cmd=20):

长度：0

#### DEV->SRV查询路灯版本应答(Type2, cmd=20):

|resv |boot\_maj\_min |boot\_rev |app\_maj\_min |app\_rev|

| 1 | 1 | 1 | 1 | 1 |

#### DEV->SRV路灯主动上报设备信息(Type3, cmd=22):

|report\_idx|data|

| 1 | N |

#### SRV->NET路灯主动上报设备信息应答(Type4 cmd=22):

|report\_idx|

| 1 |

#### DEV->SRV路灯主动上报设备异常 (Type3, cmd=21):

|report\_idx|data|

| 1 | N |

#### SRV->NET路灯主动上报设备异常应答(Type4 cmd=21):

|report\_idx|

| 1 |

#### DEV->SRV路灯主动上报设备定时器信息(Type3, cmd=26):

|report\_idx|data|

| 1 | N |

#### SRV->NET路灯主动上报设备定时器信息应答(Type4 cmd=26):

|report\_idx|

| 1 |

# 路灯升级流程

详细说明参考“基于macbee light link协议的智慧路灯远程升级方法、装置及系统.doc”

Step1：服务器发送广播报文”升级img\_header”到net

Step2:设备判断img\_header的正确性，

如果需要升级，则檫除对应镜像有有效标识位，然后根据需要跳转到特定镜像，同时将对应img保存的版本信息置0。

升级app，则跳转到boot，升级boot，则跳转到app

Step3：再次收到img\_header后，檫除对应img，记录升级结束理论时间

Step4：收到img\_data后，将数据写到对应flash位置

Step5：升级完成后，设备自动重启

Step6：如果step5不成立，但是升级结束时间到了，则主动向邻居请求img-data，

Step7：如果step4不成立，但是启动后已经等待了35秒，则主动向邻居请求img\_header

其他：如果出现img校验失败，则檫除img对应flash区域，然后主动向邻居请求img-data

如果升级中途重新受到img\_header报文,则与设备缓存的img——header比较，如果相同，则只修改升级结束理论时间，否则，修改结束时间的同时，将img对应flash区域檫除。

# 路灯信道设置

## 设备端处理流程



## 网关处理流程



## 报文定义：

#### SRV->NET同步或者设置道路RF信道(Type0, cmd=25):

|sub\_cmd |channel| time | pad |

| 1 | 1 | 1 | 1 |

sub\_cmd;/\*1:同步；2:设置\*/

time;/\*生效延迟时间\*/

# **透传报文定义**

**RV->NET命令透传(Type0或Type1, cmd=15):**

|data|

| N |

Data最大长度为20Byte

上面“4.3.6其他命令交互” 已经对该命令做出了定义，下面所有报文都是对“data”部分的扩展

**注意：以下内容只是统一定义，不是每一种设备都要实现此功能，视具体设备而定**

通用命令格式

|sub\_cmd| len | value |

| 1 | 1 | N |

Len为value的长度。

Subcmd定义

/\*TT\_SUBCMD SRV->DEV\*/

enum{

TT\_SUBCMD\_SET\_LIGHT = 1, /\*light ctrl\*/

TT\_SUBCMD\_SET\_TIMER = 2, /\*set timer\*/

TT\_SUBCMD\_DEL\_TIMER = 3, /\*del timer\*/

// TT\_SUBCMD\_SET\_GROUP = xx,/\*set group\*/

};/\*TT\_SUBCMD no ack\*/

## 亮度、冷暖光控制（sub\_cmd = TT\_SUBCMD\_SET\_LIGHT）

|  |  |  |  |
| --- | --- | --- | --- |
| byte0 | byte1 | byte2 | byte3 |
| sub\_cmd | len | mask | W\_lumin\_ch1 |
| W\_lumin\_ch2 | W\_lumin\_ch3 |  |  |

Mask: bit0:ch1\_lumin, bit1:ch1\_W bit2:ch2\_lumin, bit3:ch2\_W bit4:ch3\_lumin, bit5:ch3\_W

W\_lumin: bit7-bit4: 冷光百分比（0-10），bit3-bit0:亮度等级（0-10）

## 添加/修改定时器（sub\_cmd = TT\_SUBCMD\_SET\_TIMER）

|  |  |  |  |
| --- | --- | --- | --- |
| byte0 | byte1 | byte2 | byte3 |
| sub\_cmd | len | id | enable |
| type | week |  |  |
| start\_time | | | |
| duration | |  |  |
| mask | W\_lumin\_ch1 | W\_lumin\_ch2 | W\_lumin\_ch3 |

id:定期器ID

有效值范围（0-9），相当于最多可定义10个定时器。

添加和修改定时器时，Id由服务器分配，没有可靠保证，所以建议多发送几次

type:定时器的类型（设备端和手机端约定类型）

保留字节

enable:是否使能

使能为开启，反之为关闭

week：UTC星期

最高位bit7：0表示一次性定时器，1表示周期性定时器

bit0 - bit6对应UTC周日至周六

当bit7 == 1时，bit0 - bit6置1代表重复，可以多选。（例如：week == 0xFF，表示每天都重复）

当bit7 == 0时，bit0 - bit6可以全为0，此时设备段在添加时自动填写此字段；也可以置其中某一位为1。（例如：week == 0x04，表示此定时器在周二执行）

Start\_time:

**单位：分钟（网络序）**

就是一周第一次执行的时间点绝对utc时间。

Duration:持续分钟数\*\*\*\*\*\*\*\*设备端没有采用，用的是单点定时器

取值0 - 1439（不超过一天的时间）

duration == 0时，表示此定时器为单点定时器，否则为时间段定时器

Mask: bit0:ch1\_lumin, bit1:ch1\_W bit2:ch2\_lumin, bit3:ch2\_W bit4:ch3\_lumin, bit5:ch3\_W

W\_lumin: bit7-bit4: 冷光百分比（0-10），bit3-bit0:亮度等级（0-10）

备注：timer信息更新后需要report到服务器

## 删除定时器（sub\_cmd = TT\_SUBCMD\_DEL\_TIMER）

|  |  |  |  |
| --- | --- | --- | --- |
| byte0 | byte1 | byte2 | byte3 |
| sub\_cmd | len | Del\_all | id |
| id | …… | | id |

Len最大为18；

Del\_all: 为1时，删除所有定时器，其他情况为0；

Id：需要删除的id号；

# Report**报文**定义

**RV->NET命令透传（Type3, cmd= 16 （RF\_CMD\_TRANS\_REPORT））:**

|report\_idx| data |

| 1 | N |

report\_idx由底层统一控制，用来保证消息可靠性，在应用层交互中不可见。

Data最大长度为24Byte

上面“4.3.6其他命令交互” 已经对该命令做出了定义，下面所有报文都是对“data”部分的扩展

**注意：以下内容只是统一定义，不是每一种设备都要实现此功能，视具体设备而定**

通用命令格式

|sub\_cmd| len | value |

| 1 | 1 | N |

Len为value的长度。

Subcmd定义

/\*TT\_REPORT\_SUBCMD DEV->SRV need srv ack\*/

enum{

TTRPT\_SUBCMD\_DEV\_CPB = 1, /\*Device capability description \*/

TTRPT\_SUBCMD\_DEV\_LOAD\_INFO = 2, /\*负载信息\*/

TTRPT\_SUBCMD\_DEV\_EXCEPT = 3, /\*设备异常\*/

TTRPT\_SUBCMD\_DEV\_TIM\_INFO = 4, /\*定时器信息\*/

};/\*TT\_REPORT\_SUBCMD need srv ack\*/

## 设备能力（sub\_cmd = TTRPT\_SUBCMD\_DEV\_CPB）

|  |  |  |  |
| --- | --- | --- | --- |
| byte0 | byte1 | byte2 | byte3 |
| sub\_cmd | len |  |  |
| flag | | | |
| light\_num | rsv | rsv | rsv |

flag定义： BIT0: 是否支持冷暖调光; BIT1: 是否支持定时器; BIT2: 是否电池供电;

light\_num：支持路数，从1开始

## 负载信息（sub\_cmd = TTRPT\_SUBCMD\_DEV\_LOAD\_INFO）

|  |  |  |  |
| --- | --- | --- | --- |
| byte0 | byte1 | byte2 | byte3 |
| sub\_cmd | len |  |  |
| load100\_i\_ch1 | | load100\_v\_ch1 | |
| load100\_pf\_ch1 |  |  |  |
| load100\_i\_ch2 | | load100\_v\_ch2 | |
| load100\_pf\_ch2 |  |  |  |
| load100\_i\_ch3 | | load100\_v\_ch3 | |
| load100\_pf\_ch3 |  |  |  |

oad100\_i\_ch1：第一路负载亮度为100%时的电流

load100\_v\_ch1：第一路负载亮度为100%时的电压

load100\_pf\_ch1：第一路负载亮度为100%时的功率因数

## 异常上报（sub\_cmd = TTRPT\_SUBCMD\_DEV\_EXCEPT）

|  |  |  |  |
| --- | --- | --- | --- |
| byte0 | byte1 | byte2 | byte3 |
| sub\_cmd | len |  |  |
| count |  |  |  |
| exception\_code | valid | exception\_code | valid |
| … | | exception\_code | valid |

Count：异常计数

exception\_code：异常编号

Bit0-bit3:异常定义

bit4-bit7:表示通道：0表示通用异常；1表示第一路；2表示第二路；3：表示第3路；

Valid：0异常消失，1异常产生

/\*exception\_code\*/

enum{

LAMP\_EXCEPTION\_V = 1,//电压异常

LAMP\_EXCEPTION\_A = 2,//电流异常

};

## 定时器信息（sub\_cmd = TTRPT\_SUBCMD\_DEV\_TIM\_INFO）

|  |  |  |  |
| --- | --- | --- | --- |
| byte0 | byte1 | byte2 | byte3 |
| sub\_cmd | len |  |  |
| all\_timer\_crc | | tim0\_crc | Tim1\_crc |
| Tim2\_crc | Tim3\_crc | Tim4\_crc | Tim5\_crc |
| Tim6\_crc | Tim7\_crc | Tim8\_crc | Tim9\_crc |

附件1 macbee header定义

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* header \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//common header

typedef struct{

#if \_\_BYTE\_ORDER == \_\_BIG\_ENDIAN

u\_int16\_t type:3;

u\_int16\_t cry:1;

u\_int16\_t resv:1;

u\_int16\_t hop:11;

#elif \_\_BYTE\_ORDER == \_\_LITTLE\_ENDIAN

u\_int16\_t hop:11;

u\_int16\_t resv:1;

u\_int16\_t cry:1;

u\_int16\_t type:3;

#endif

u\_int16\_t addr;

u\_int8\_t seq;

}com\_hd\_t;/\*5 byte\*/

//type=0. broadcast: server -> all devs

//type=1. multicast or unicast: server -> devs or dev

//type=4. unicast report ack: server -> dev

typedef struct{

com\_hd\_t ch;/\*5 Byte\*/

#if \_\_BYTE\_ORDER == \_\_BIG\_ENDIAN

u\_int16\_t gw\_addr:5;

u\_int16\_t offset\_time:11;/\*unit:rtc\_tick\*/

#elif \_\_BYTE\_ORDER == \_\_LITTLE\_ENDIAN

u\_int16\_t offset\_time:11;

u\_int16\_t gw\_addr:5;

#endif

u\_int8\_t slot;

u\_int8\_t cmd;

}srv2net\_hd\_t;/\*9 Byte\*/

//type=2. unicast ack: dev -> server

//type=3. unicast report: dev -> server

typedef struct{

com\_hd\_t ch;

u\_int8\_t cmd;

}dev2srv\_hd\_t;

//type5. neighbor communication: dev -> dev

typedef struct{

#if \_\_BYTE\_ORDER == \_\_BIG\_ENDIAN

u\_int8\_t type:3;

u\_int8\_t cry:1;

u\_int8\_t resv1:1;

u\_int8\_t ack:1;

u\_int8\_t resv2:2;

#elif \_\_BYTE\_ORDER == \_\_LITTLE\_ENDIAN

u\_int8\_t resv2:2;

u\_int8\_t ack:1;

u\_int8\_t resv1:1;

u\_int8\_t cry:1;

u\_int8\_t type:3;

#endif

u\_int16\_t addr;

u\_int8\_t seq;

u\_int8\_t cmd;

}dev2dev\_hd\_t;/\*5字节\*/

# 附件1 Macbee路灯协议技术指标

具体功能：

1. 每个灯杆在安装时实现gps位置上报
2. 一键实现自动组网。
3. 定时开关（亮度可调）、

2、异常主动上报（电流、电压异常）

3、分组控制（还未实现）

**表1 各种功能响应时间**

|  |  |
| --- | --- |
| 无线信号距离（无中继） | 100-200米 |
| Hop增加一次时间  （无线信号中继一次时间） | 62.5-187.5ms |
| 开/关灯最小响应时间 | Hop\*62.5ms  以无线距离100米计算，10Km响应时间为6.25秒 |
| 开/关灯最大响应时间 | Hop\*187.5ms  以无线距离100米计算，10Km响应时间为18.75秒 |
| 离线时间（以最差计算） | 5min+系统(hop\_max/10+1)\*20\*3 + hop\_dev\*5  以无线距离100米计算，10Km响应时间为300+600秒 |
| 上线时间（以最差计算） | 系统最大(hop\_max/10+1)\*20  以无线距离100米计算，10Km响应时间为200秒 |
| 故障上报时间（以最差计算） | 系统最大(hop\_max/10+1)\*20  以无线距离100米计算，10Km响应时间为200秒 |
| 单网关容量 | 25Km±1Km至少安装一个网关，建议每10Km安装一个网关。 |
| 网关总容量 | 一条道路最多可容纳31个网关 |
| 系统容量 | 一条道路最大可容纳3万个路灯控制器 |
| 设备升级时间 | 10min-30min |