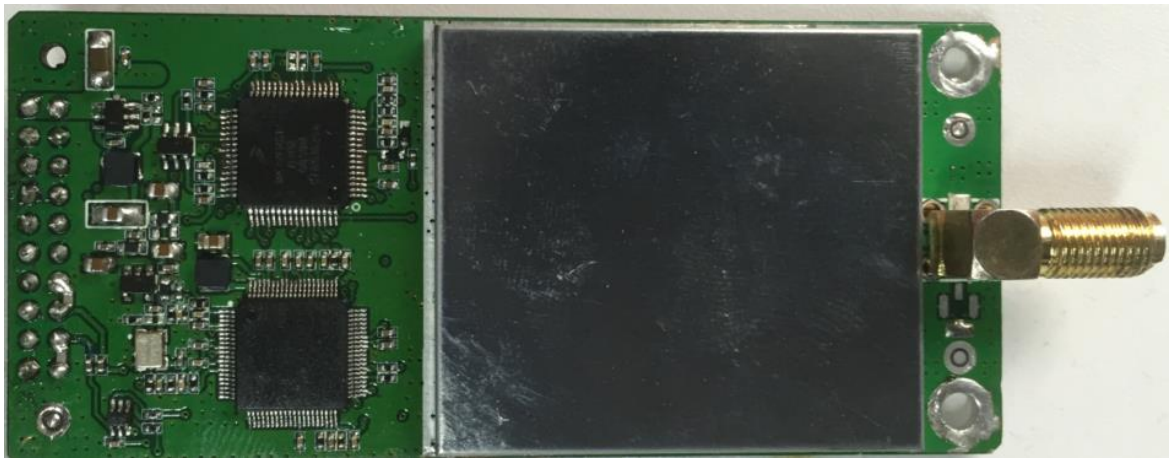
深圳市尚瑞思电子有限公司

DMR-5WUF产品手册

DMR 数字全双工



概述

1.1 简介

DMR-5WUF是深圳市尚瑞思电子有限公司全新推出的一款双时隙DMR数字模块，是行业内唯一款体积最小，支持模拟常规，数字常规的全双工专业通信模块。具有接口简单，高性能，满足国内外相关认证，如型号核准，CE,FCC等。简单易用的标准串口接口，灵活实用的数据业务和操作，能快速的嵌入到各种设备中。该模块仅需外接天线、麦克风、语音功放即可组成一台完整的DMR数字对讲设备。

1.2 应用领域

电梯对讲通信
楼宇小区安防系统
煤矿，海运

2 特性

- 频率范围： 136~174MHZ，220~260MHZ,350~390MHZ,400~480MHz，
- 频率间隔：数字 12.5KHZ，模拟 12.5KHZ、25KHZ
- 射频输出功率：高功率 5W，低功率 1W

- 支持强插/强拆功能
- 支持AMBE3000、WT3000、AMBE1000、SELP、AVDS 等多种声码器
-
- 供电电压：12V
- 高接收灵敏度：≤-122dBm
- 支持组呼、全呼、单呼及全双工语音通信
- 支持确认、非确认短信通信、支持状态短消息
- 支持主叫/被叫检测
- 支持呼叫提示
- 支持远程监听
- 支持直通、中继模式的语音、短信应用

3 尺寸及引脚

DMR-5WUF 板子如图 1 所示，其尺寸为 50mm×90mm。其中 J1 为信号接口，J2 为天线口。J1 的定义如表 1 所示。

图 1 HR_DMR-5WUF 模块板

表 1 J1 接口管脚定义

管脚号	管脚名称	管脚类别	功能描述
1	VBAT	POWER	电源
2	VBAT	POWER	电源
3	GND	GND	地
4	GND	GND	地
5	UART_TX	DO	异步串口（模块发送数据口）
6	UART_RX	DI	异步串口（模块接收数据口）
7	HANGUP	DI	输出喇叭控制使能
8	CALL	DI	开始呼叫，低电平脉冲（大于 20ms）触发

9	CTRL_D0	DIO	预留
10	CTRL_D1	DIO	预留
11	CTRL_D2	DIO	预留
12	CTRL_D3	DIO	预留
13	GND	GND	地
14	NC	NC	悬空
15	MIC_IN	AI	麦克风信号输入
16	GND	GND	地
17	LINEOUT	AO	接收音频信号输出
18	GND	GND	地
19	GND	GND	地
20	GND	GND	地

注 1：在上电时，CALL 管脚需拉高；

注 2：CTRL_D0、CTRL_D1、CTRL_D2、CTRL_D3 这 4 个管脚为预留管脚，建议如果连接到外置 CPU 的 GPIO 上时，在上电时，需拉高。

4 典型应用电路框图

典型应用电路框图如图 2 所示。DMR-5WUF 外接一个主控 MCU、音频功放及喇叭、麦克风即可工作。工作时，可以通过 MCU 写串口命令及配置 Call、HangUp 管脚进行收发控制。

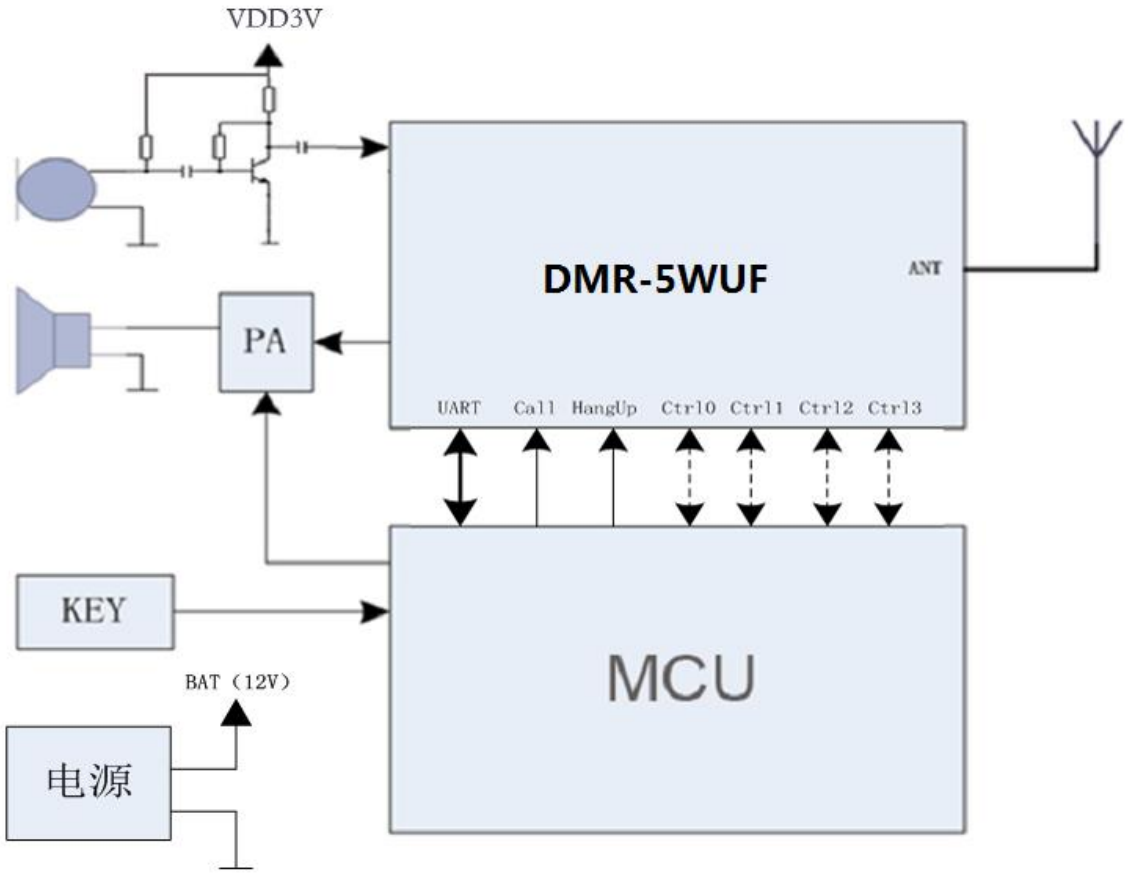


图 2 典型应用电路框图

5 技术参数

5.1 电气特性

表 2 电气特性

参数	条件	最小值	典型值	最大值	单位
供电电压		9	12	14	V
工作温度		-25		70	℃
模块启动时间		100			ms
串口速率			57600		bps
麦克风输入电压		0.5		1	Vpp
Lineout 输出电压				1	Vpp

5.2 指标特性

表 3 指标特性

参数	条件	最小值	典型值	最大值	单位
工作频率	136~174MHZ	136		174	MHz
工作频率	220~260MHZ	220		260	MHz
工作频率	350~390MHZ	350		390	MHz
工作频率	400~480MHz	400		480	MHz
信道间隔			12.5		kHz
天线阻抗			50		Ω
接收					
灵敏度			-122		dBm
邻道选择性		60			dB
发射					
发射功率（低）		1	1.5	2	W
发射功率（高）		5	6	8	W
邻道功率比		60	62	65	dB

DMR-5WUF 可以通过串口协议配置模块进行接收、发射等功能，具体如串口协议所示。也能够通过配置 CALL 管脚来进行控制发射。

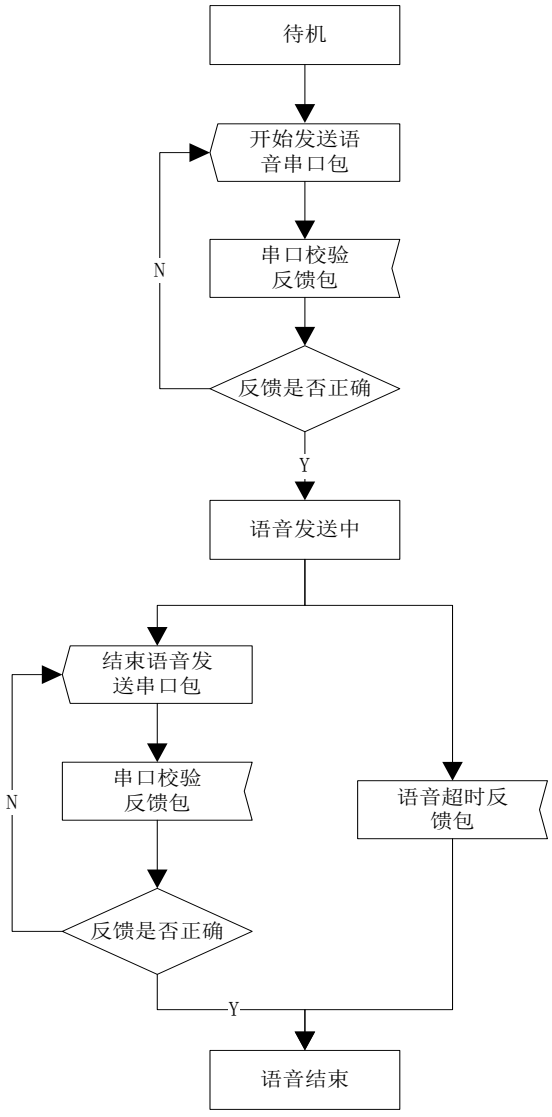
5.3 语音发送

为方便用户使用，可以通过配置 CALL 管脚来进行语音发送（该功能也可以通过串口协议来实现）。当使用 PTT 管脚控制时，操作顺利如下：

- 通过串口命令写入信道切换配置包切换到所需的信道。
- CALL 配置如图所示，将 CALL 拉低，启动发射； CALL 管脚拉高，结束发射。

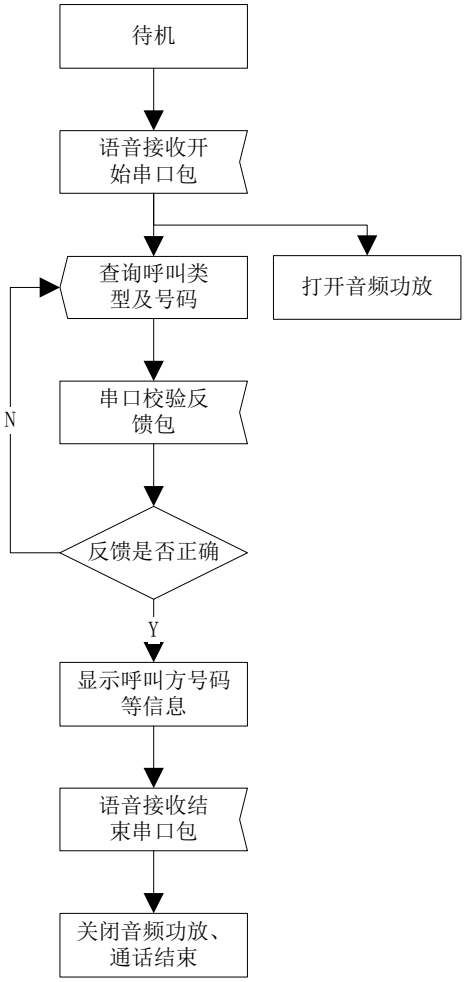
图 3 发射管脚配置时序

5.3.1 语音发送流程



5.4 语音接收

语音接收流程见所示。



5.5 短信收发

短信收发流程见所示。

5.5.1 非确认短信发送流程

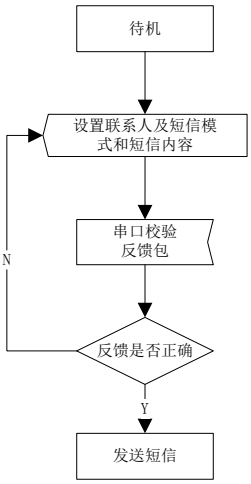


图 4 非确认短信发送流程

外置 CPU 处理非确认短信发送流程如图 4 所示。

首先，由外置 CPU 向模块写入串口包，设置短信联系人及短信模式和短信内容；模块会对写入的串口包进行校验，若校验不通过，则给出错误的反馈包，如果校验通过，则发送短信。

5.5.2 非确认短信接收流程

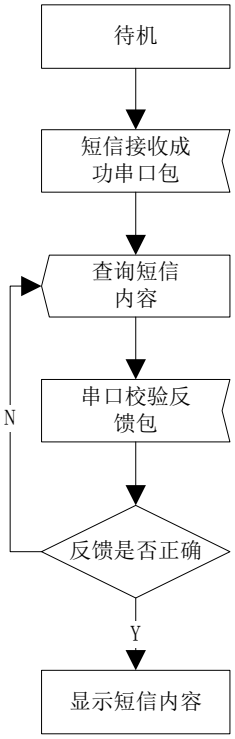


图 5 非确认短信接收流程

外置 CPU 处理非确认短信发送流程如图 5 所示。

外置 CPU 接收到短信接收成功的串口包后向模块查询短信内容的串口包。模块对接收到的查询短信内容的串口包进行校验，若校验正确，则将接收到的短信内容及短信发送方地址发送给外置 CPU；若校验不正确，则向外置 CPU 发送校验不正确的串口反馈包。

5.6 功放开关

- 当模块需要出声音的时候，模块会给出 HANGUP 管脚一个上升沿脉冲，当关闭声音输出的时候，模块会给出 HANGUP 管脚一个下降沿脉冲。即常规模式下 HANGUP 管脚为低电平，播放声音的时候为高电平。HANGUP 配置如下图所示。

6 串口协议

模块支持通过串口进行语音、短信等功能的收发配置。串口协议包格式如图 6 所示，协议字段定义如表 4 所示。

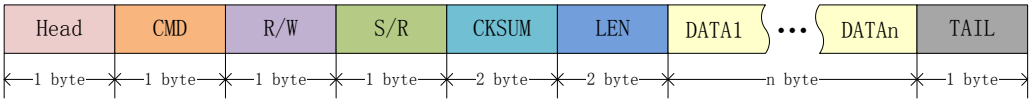


图 6 串口协议包格式

表 4 串口协议字段定义

Offset	Flag	Length	Comment	Detail
0	Head	1	包头	0x68
1	CMD	1	指令	0x22：设置数字信道 0x23：设置模拟信道 0x24：查询数字信道信息 0x25：查询模拟信道信息 0x26：设置发射信息 0x27：查询模块是否初始化完成 0x28：设置增强功能 0x29：设置加密功能 0x2A：设置 MIC 增益 0x2B：查询数字语音接收信息 0x2C：发送短信

				0x2D：获取短信 0x2E：设置音量 0x2F：设置监听 0x30：设置静噪 0x31：设置省电模式 0x32：查询信号强度 0x33：设置中继脱网模式 0x34:查询版本号 0x3C:传输中断功能
2	R/W	1	操作方式	0x00：读； 0x01：写； (外部 CPU 发为写，外部 CPU 收为读) 0x02：主动发送
3	S/R	1	设置/回答指令	设置： 0x01：表示开始设置 回答： 0x00 设置成功 0x01 设置失败 0x02 校验错误
4、5	CKSUM	2	检验和	
6、7	LEN	2	数据段长度	DATA 数据段长度，若无数据段信息，则 LEN 值为 0x00
8	DATA	len	数据段信息	
	TAIL	1	包尾	0x10

检测和算法

```
uint16 PcCheckSum(uint8 * buf, int16 len)
{
    uint32 sum=0;
    while(len >1)
    {
        sum += 0xFFFF & (*buf<<8|*(buf+1));
        buf+=2;
        len-=2;
    }
    if (len)
    {
        sum += (0xFF & *buf)<<8;
    }
    while (sum>>16)
    {
```

```
            sum = (sum & 0xFFFF)+(sum >> 16);
        }
        return( (uint16) sum ^ 0xFFFF);
}
```

注：所有的数据都采用小端模式

6.01 设置数字组命令

```
typedef struct
{
    uint32   rx_freq;        //接收频率          4000000000-480000000HZ
    uint32   tx_freq;        //发射频率          4000000000-480000000HZ
    uint32    localID;       //本机 ID      1-16776415

    uint32 GroupList[32];    //接收组列表

    uint32   tx_contact;     //联系人号码 1-16776415
    uint8    ContactType;        //联系人类型 0:个呼 1:组呼 2:全呼 3：双工

    uint8    power;          //0:低功率 1：高功率
    uint8    cc;                 //色码    0~15

    uint8    InboundSlot;        //0:时隙 1      1: 时隙 2
    uint8    OutboundSlot;   //0:时隙 1      1: 时隙 2

    uint8    ChannelMode;        //0 直通模式 4 真双时隙

    uint8    EncryptSw;          //加密开关 1:enable 2:disable
    uint8    EncryptKey[8];  //密钥

    uint8 pwrsave;           //省电开关 2:disable 1:enable
    uint8 volume;            //音量 1-9
    uint8 mic;                  //mic 增益 0~2
    uint8 relay;             //中继脱网 2:disable 1:enable
} DB_DIGITAL_INFO;
```

Offset	Flag	Length	Comment	Detail
0	Head	1	包头	0x68
1	CMD	1	指令	0x22
2	R/W	1	操作方式	0x01（写模式）
3	S/R	1	设置/回答指令	0x01：设置
4，5	CKSUM	2	检验和	（16bit 校验和值）

6，7	LEN	2	数据段长度	数字结构体长度
0	DATA	1	数据段信息	根据需要填写数字结构体，结构体后面的注释为变量的范围
	TAIL	1	包尾	0x10

反馈包

Offset	Flag	Length	Comment	Detail
0	Head	1	包头	0x68
1	CMD	1	指令	0x22
2	R/W	1	操作方式	0x00（读模式）
3	S/R	1	设置/回答指令	0x00 设置成功 0x01设置失败 0x02 校验错误
4，5	CKSUM	2	检验和	（16bit 校验和值）
6，7	LEN	2	数据段长度	0x00,0x00
8	TAIL	1	包尾	0x10

6.02 设置模拟组命令

```
typedef struct
{
    uint32   rx_freq;   //接收频率     4000000000-480000000HZ
    uint32   tx_freq;   //发射频率     4000000000-480000000HZ

    uint8 band;              //0:窄带  1:宽带
    uint8 power;        //0:低功率  1:高功率

    uint8 sq;           //SQ 等级  0~9
    uint8 rx_type;      //|0:载波  |1:ctcsss |2:正向 DCS |3:反向 DCS
    uint8 rx_subcode;   //|      0    |    0~50   |    0~82    |   0~82
    uint8 tx_type;      //|0:载波  |1:ctcsss |2:正向 DCS |3:反向 DCS
    uint8 tx_subcode;   //|      0    |    0~50   |    0~82    |   0~82

    uint8 pwrsave;      //省电开关  2:disable 1:enable
    uint8 volume;       //音量  1-9
    uint8 monitor;      //监听开关  2:disable 1:enable
    uint8 relay;        //中继脱网  2:disable 1:enable
} DB_ANALOG_INFO;
```

Offset	Flag	Length	Comment	Detail
0	Head	1	包头	0x68
1	CMD	1	指令	0x23
2	R/W	1	操作方式	0x01（写模式）
3	S/R	1	设置/回答指令	0x01：设置
4，5	CKSUM	2	检验和	（16bit 校验和值）
6，7	LEN	2	数据段长度	模拟结构体长度
0	DATA	1	数据段信息	根据需要填写模拟结构体，结构体后面的注释为变量的范围
	TAIL	1	包尾	0x10

反馈包

Offset	Flag	Length	Comment	Detail
0	Head	1	包头	0x68
1	CMD	1	指令	0x23
2	R/W	1	操作方式	0x00（读模式）
3	S/R	1	设置/回答指令	0x00 设置成功 0x01设置失败 0x02 校验错误
4，5	CKSUM	2	检验和	（16bit 校验和值）
6，7	LEN	2	数据段长度	0x00,0x00
8	TAIL	1	包尾	0x10

6.03 设置发射命令

Offset	Flag	Length	Comment	Detail
0	Head	1	包头	0x68
1	CMD	1	指令	0x26
2	R/W	1	操作方式	0x01（写模式）
3	S/R	1	设置/回答指令	0x01：设置
4，5	CKSUM	2	检验和	（16bit 校验和值）
6，7	LEN	2	数据段长度	0x00 0x01
0	DATA	1	数据段信息	1：发送开始 2：发送结束
	TAIL	1	包尾	0x10

反馈包

Offset	Flag	Length	Comment	Detail
0	Head	1	包头	0x68

1	CMD	1	指令	0x26
2	R/W	1	操作方式	0x00（读模式）
3	S/R	1	设置/回答指令	0x00 设置成功 0x01设置失败 0x02 校验错误
4，5	CKSUM	2	检验和	（16bit 校验和值）
6，7	LEN	2	数据段长度	0x00,0x00
8	TAIL	1	包尾	0x10

6.04 设置增强功能

Offset	Flag	Length	Comment	Detail
0	Head	1	包头	0x68
1	CMD	1	指令	0x28
2	R/W	1	操作方式	0x01（写模式）
3	S/R	1	设置/回答指令	0x01：设置
4，5	CKSUM	2	检验和	（16bit 校验和值）
6，7	LEN	2	数据段长度	0x00,0x05
8	FUN	1	增强功能	0x01：对讲机检测 0x02：呼叫提示 0x03：远程监听 0x04：对讲机遥毙 0x05：对讲机激活
9,10,11,12	CallNum	4	联系人号码	1-16776415
12	TAIL	1	包尾	0x10

反馈包

Offset	Flag	Length	Comment	Detail
0	Head	1	包头	0x68
1	CMD	1	指令	0x29
2	R/W	1	操作方式	0x00（读模式）
3	S/R	1	设置/回答指令	0x00 设置成功 0x01设置失败 0x02 校验错误
4，5	CKSUM	2	检验和	（16bit 校验和值）
6，7	LEN	2	数据段长度	0x00,0x00
8	TAIL	1	包尾	0x10

6.05 设置加密功能

Offset	Flag	Length	Comment	Detail
0	Head	1	包头	0x68
1	CMD	1	指令	0x29
2	R/W	1	操作方式	0x01（写模式）
3	S/R	1	设置/回答指令	0x01：设置
4，5	CKSUM	2	检验和	（16bit 校验和值）
6，7	LEN	2	数据段长度	0x00,0x09
8	SWITCH	1	加密开关	1:enable 2:disable
9~16	DATA	8	数据	密钥 0~7 字节
17	TAIL	1	包尾	0x10

反馈包

Offset	Flag	Length	Comment	Detail
0	Head	1	包头	0x68
1	CMD	1	指令	0x29
2	R/W	1	操作方式	0x00（读模式）
3	S/R	1	设置/回答指令	0x00 设置成功 0x01设置失败 0x02 校验错误
4，5	CKSUM	2	检验和	（16bit 校验和值）
6，7	LEN	2	数据段长度	0x00,0x00
8	TAIL	1	包尾	0x10

6.06 设置 MIC 增益

Offset	Flag	Length	Comment	Detail
0	Head	1	包头	0x68
1	CMD	1	指令	0x2A
2	R/W	1	操作方式	0x01（写模式）
3	S/R	1	设置/回答指令	0x01：设置
4，5	CKSUM	2	检验和	（16bit 校验和值）
6，7	LEN	2	数据段长度	0x00,0x01
8	GAIN	1	增益值	范围 0~15
9	TAIL	1	包尾	0x10

反馈包

Offset	Flag	Length	Comment	Detail
0	Head	1	包头	0x68
1	CMD	1	指令	0x2A
2	R/W	1	操作方式	0x00（读模式）
3	S/R	1	设置/回答指令	0x00 设置成功 0x01设置失败 0x02 校验错误
4，5	CKSUM	2	检验和	（16bit 校验和值）
6，7	LEN	2	数据段长度	0x00,0x00
8	TAIL	1	包尾	0x10

6.06 获取数字语音接收信息

Offset	Flag	Length	Comment	Detail
0	Head	1	包头	0x68
1	CMD	1	指令	0x2B
2	R/W	1	操作方式	0x01（写模式）
3	S/R	1	设置/回答指令	0x01：设置
4，5	CKSUM	2	检验和	（16bit 校验和值）
6，7	LEN	2	数据段长度	0x00,0x01
8		1		01
9	TAIL	1	包尾	0x10

反馈包

Offset	Flag	Length	Comment	Detail
0	Head	1	包头	0x68
1	CMD	1	指令	0x2B
2	R/W	1	操作方式	1
3	S/R	1	设置/回答指令	1
4，5	CKSUM	2	检验和	（16bit 校验和值）
6，7	LEN	2	数据段长度	0x00,0x05
8-12	DATA		数据段信息	第 0 字节为呼叫类型： 0x00：个呼 0x01：组呼 0x02：无地址呼 0x03：全呼和广播

				第 1~4 字节为联系人号码
13	TAIL	1	包尾	0x10

6.07 发送短信

Offset	Flag	Length	Comment	Detail
0	Head	1	包头	0x68
1	CMD	1	指令	0x2C
2	R/W	1	操作方式	0x01
3	S/R	1	设置/回答指令	0x01
4，5	CKSUM	2	检验和	（16bit 校验和值；）
6，7	LEN	LEN	数据段长度	
8	Msg Type	1	短信类型	0x01：IP 确认 0x02：IP 非确认 0x03：组呼
9,10,11,12	CallNum	4	联系人号码	联系人号码 4 字节
13	DATA	LEN-4	数据段信息	短信格式为 unicode 编码
	TAIL	1	包尾	0x10

反馈包

Offset	Flag	Length	Comment	Detail
0	Head	1	包头	0x68
1	CMD	1	指令	0x2C
2	R/W	1	操作方式	0x00（读模式）
3	S/R	1	设置/回答指令	0x00 设置成功 0x01设置失败 0x02 校验错误
4，5	CKSUM	2	检验和	（16bit 校验和值）
6，7	LEN	2	数据段长度	0x00,0x00
8	TAIL	1	包尾	0x10

6.08 获取短信

Offset	Flag	Length	Comment	Detail
0	Head	1	包头	0x68
1	CMD	1	指令	0x2D
2	R/W	1	操作方式	0x01（写模式）
3	S/R	1	设置/回答指令	0x01
4，5	CKSUM	2	检验和	
6，7	LEN	2	数据段长度	0x00,0x01
8	DATA	1	查询内容	0x01
9	TAIL	1	包尾	0x10

反馈包

Offset	Flag	Length	Comment	Detail
0	Head	1	包头	0x68
1	CMD	1	指令	0x2D
2	R/W	1	操作方式	0x00（读模式）
3	S/R	1	设置/回答指令	0x01
4，5	CKSUM	2	检验和	16bit 校验和值
6，7	LEN	2	数据段长度	
8-11	CallID	4	短信发送方号码	
12	MsgData	LEN-4	短信内容	短信格式为 unicode 编码
	TAIL	1	包尾	0x10

注：短信主动发送命令

Offset	Flag	Length	Comment	Detail
0	Head	1	包头	0x68
1	CMD	1	指令	0x2D

2	R/W	1	操作方式	0x02（主动发送）
3	S/R	1	设置/回答指令	0x01
4，5	CKSUM	2	检验和	16bit 校验和值
6，7	LEN	2	数据段长度	
8-11	CallID	4	短信发送方号码	
12	MsgData	LEN-4	短信内容	短信格式为 unicode 编码
	TAIL	1	包尾	0x10

6.09 设置音量

Offset	Flag	Length	Comment	Detail
0	Head	1	包头	0x68
1	CMD	1	指令	0x2E
2	R/W	1	操作方式	0x01（写模式）
3	S/R	1	设置/回答指令	0x01：设置
4，5	CKSUM	2	检验和	（16bit 校验和值）
6，7	LEN	2	数据段长度	0x00,0x01
8	VOL	1	音量值	范围 1~9，其中 1 表示音量最小，9 表示音量最大
9	TAIL	1	包尾	0x10

反馈包

Offset	Flag	Length	Comment	Detail
0	Head	1	包头	0x68
1	CMD	1	指令	0x2E
2	R/W	1	操作方式	0x00（读模式）
3	S/R	1	设置/回答指令	0x00 设置成功 0x01设置失败 0x02 校验错误
4，5	CKSUM	2	检验和	（16bit 校验和值）
6，7	LEN	2	数据段长度	0x00,0x00
8	TAIL	1	包尾	0x10

6.10 设置监听

Offset	Flag	Length	Comment	Detail
0	Head	1	包头	0x68
1	CMD	1	指令	0x2F
2	R/W	1	操作方式	0x01（写模式）
3	S/R	1	设置/回答指令	0x01：设置
4，5	CKSUM	2	检验和	（16bit 校验和值）
6，7	LEN	2	数据段长度	0x00,0x01
8	VOL	1		1：enable 2:disable
9	TAIL	1	包尾	0x10

反馈包

Offset	Flag	Length	Comment	Detail
0	Head	1	包头	0x68
1	CMD	1	指令	0x2F
2	R/W	1	操作方式	0x00（读模式）
3	S/R	1	设置/回答指令	0x00 设置成功 0x01设置失败 0x02 校验错误
4，5	CKSUM	2	检验和	（16bit 校验和值）
6，7	LEN	2	数据段长度	0x00,0x00
8	TAIL	1	包尾	0x10

6.11 设置静噪

Offset	Flag	Length	Comment	Detail
0	Head	1	包头	0x68
1	CMD	1	指令	0x30
2	R/W	1	操作方式	0x01（写模式）
3	S/R	1	设置/回答指令	0x01：设置
4，5	CKSUM	2	检验和	（16bit 校验和值）
6，7	LEN	2	数据段长度	0x00,0x01
8	VOL	1	SQL	0~9
9	TAIL	1	包尾	0x10

反馈包

Offset	Flag	Length	Comment	Detail
0	Head	1	包头	0x68
1	CMD	1	指令	0x30
2	R/W	1	操作方式	0x00（读模式）
3	S/R	1	设置/回答指令	0x00 设置成功 0x01设置失败 0x02 校验错误
4，5	CKSUM	2	检验和	（16bit 校验和值）
6，7	LEN	2	数据段长度	0x00,0x00
8	TAIL	1	包尾	0x10

6.12 设置省电

Offset	Flag	Length	Comment	Detail
0	Head	1	包头	0x68
1	CMD	1	指令	0x31
2	R/W	1	操作方式	0x01（写模式）
3	S/R	1	设置/回答指令	0x01：设置
4，5	CKSUM	2	检验和	（16bit 校验和值）
6，7	LEN	2	数据段长度	0x00,0x01
8	VOL	1		1：enable 2:disable
9	TAIL	1	包尾	0x10

反馈包

Offset	Flag	Length	Comment	Detail
0	Head	1	包头	0x68
1	CMD	1	指令	0x31
2	R/W	1	操作方式	0x00（读模式）
3	S/R	1	设置/回答指令	0x00 设置成功 0x01设置失败 0x02 校验错误
4，5	CKSUM	2	检验和	（16bit 校验和值）
6，7	LEN	2	数据段长度	0x00,0x00
8	TAIL	1	包尾	0x10

6.13 获取信号强度

Offset	Flag	Length	Comment	Detail
0	Head	1	包头	0x68
1	CMD	1	指令	0x32
2	R/W	1	操作方式	0x01（写模式）
3	S/R	1	设置/回答指令	0x01：设置
4，5	CKSUM	2	检验和	（16bit 校验和值）
6，7	LEN	2	数据段长度	0x00,0x01
8	Data	1	数据段信息	0x01
	TAIL	1	包尾	0x10

反馈包

Offset	Flag	Length	Comment	Detail
0	Head	1	包头	0x68
1	CMD	1	指令	0x32
2	R/W	1	操作方式	0x00（读模式）
3	S/R	1	设置/回答指令	0x00 成功 0x01 失败 0x02 校验错误
4，5	CKSUM	2	检验和	（16bit 校验和值）
6，7	LEN	2	数据段长度	
8	RSSI	1	数据段信息	信号强度值
	TAIL	1	包尾	0x10

6.14 设置中继脱网功能

Offset	Flag	Length	Comment	Detail
0	Head	1	包头	0x68
1	CMD	1	指令	0x33
2	R/W	1	操作方式	0x01（写模式）
3	S/R	1	设置/回答指令	0x01：设置
4，5	CKSUM	2	检验和	（16bit 校验和值）
6，7	LEN	2	数据段长度	0x00,0x01
8	VOL	1		1：enable 2:disable
9	TAIL	1	包尾	0x10

反馈包

Offset	Flag	Length	Comment	Detail

0	Head	1	包头	0x68
1	CMD	1	指令	0x33
2	R/W	1	操作方式	0x00（读模式）
3	S/R	1	设置/回答指令	0x00 设置成功 0x01设置失败 0x02 校验错误
4，5	CKSUM	2	检验和	（16bit 校验和值）
6，7	LEN	2	数据段长度	0x00,0x00
8	TAIL	1	包尾	0x10

6.15 获取版本号

Offset	Flag	Length	Comment	Detail
0	Head	1	包头	0x68
1	CMD	1	指令	0x34
2	R/W	1	操作方式	0x01（写模式）
3	S/R	1	设置/回答指令	0x01：设置
4，5	CKSUM	2	检验和	（16bit 校验和值）
6，7	LEN	2	数据段长度	0x00,0x01
8	DATA	1	数据	0x01
9	TAIL	1	包尾	0x10

反馈包

Offset	Flag	Length	Comment	Detail
0	Head	1	包头	0x68
1	CMD	1	指令	0x34
2	R/W	1	操作方式	0x00（读模式）
3	S/R	1	设置/回答指令	0x00 成功 0x01 失败 0x02 校验错误
4，5	CKSUM	2	检验和	（16bit 校验和值）
6，7	LEN	2	数据段长度	0x00,0x12
8-25	DATA	18	软件版本号	0~17 字节：软件版本号
26	TAIL	1	包尾	0x10

6.17 传输中断功能

Offset	Flag	Length	Comment	Detail
0	Head	1	包头	0x68
1	CMD	1	指令	0x3C
2	R/W	1	操作方式	0x01（写模式）
3	S/R	1	设置/回答指令	0x01：设置
4，5	CKSUM	2	检验和	（16bit 校验和值）
6，7	LEN	2	数据段长度	0x00,0x01
8	DATA	1	数据	1:enable 2:disable
9	TAIL	1	包尾	0x10

反馈包

Offset	Flag	Length	Comment	Detail
0	Head	1	包头	0x68
1	CMD	1	指令	0x3C
2	R/W	1	操作方式	0x00（读模式）
3	S/R	1	设置/回答指令	0x00 设置成功 0x01设置失败 0x02 校验错误
4，5	CKSUM	2	检验和	（16bit 校验和值）
6，7	LEN	2	数据段长度	0x00,0x00
8	TAIL	1	包尾	0x10
Offset	Flag	Length	Comment	Detail

附录 串口校验算法

```
uint16 PcCheckSum(uint8 * buf, int16 len)
{
    uint32 sum=0;
    while(len >1)
    {
        sum += 0xFFFF & (*buf<<8|*(buf+1));
        buf+=2;
        len-=2;
    }
```

```
    if (len)
    {
        sum += (0xFF & *buf)<<8;
    }
    while (sum>>16)
    {
        sum = (sum & 0xFFFF)+(sum >> 16);
    }
    return( (uint16) sum ^ 0xFFFF);
}
```

文档中的《附录串口校验算法》，关于 PcCheckSum(uint8 * buf, int16 len)函数两个疑问：

1.参数 buf，是包头的起始位置还是数据段的起始位置，len 是整包长度，还是数据段的长度？

2.如果是计算的整包的校验和，在计算校验和是，是否需要把第 4，5 位（校验和的位置）置“0”，再来计算校验和？

```
 uint8 CheckCkSum(uint16 len,uint8 buf[])
{
   uint16 sum,cksum;
   sum = buf[Pcksum];//Pcksum  为 4
   buf[Pcksum] = 0;
   sum = (sum<<8);
   sum = sum + buf[Pcksum+1];
   buf[Pcksum+1] = 0;
            cksum = PcCheckSum(buf,len);
            buf[Pcksum+1] = (sum&0xff);
            buf[Pcksum] = (sum>>8)&0xff;
if(sum == cksum)
{
   //AckToPC(buf[Pcmd],ok);
   return 1;
}
else
{
   //AckToPC(buf[Pcmd],ChkError);
   return 0;
}
}

//这个是检验的接口，可以参考下代码。
```

从中可以看到

文档中的《附录串口校验算法》，关于 PcCheckSum(uint8 * buf, int16 len)函数有两个疑问：

1buf　从包头的起始，len 是整包长度

2.是的，，需要把第 4，5 位（校验和的位置）置“0”，再来计算校验和