WiMi-net 通讯格式说明书

微网智控(北京)科技有限公司

目　录

1. 二进制的通讯格式 1

1.1 二进制流接口格式 7
1.2 二进制流接口 CRC 计算方式 7
1.3 名词术语和缩写简记符 9
1.4 32 位系统上的编程注意事项 9
1.5 关于缺省参数和配置 9

2. 公共参数配置命令 10

2.1 节点工作模式 10
2.2 网络服务标示 11
2.3 16 位网络地址 12
2.4 64 位 MAC 地址 13
2.5 节点的工作频带和信道 14
2.6 配置参数恢复出厂配置 15
2.7 节点的对码状态 16
2.8 射频输出链路配置 17
2.9 射频输入链路配置 18

3. 主站参数配置命令 19

3.1 主站的信道配置模式 19
3.2 删除休眠节点 20
3.3 删除对码节点（按照 16 位网络地址存储格式） 21
3.4 删除对码节点（按照 64 位 MAC 地址存储格式） 22
3.5 查询休眠节点的成员数量 23
3.6 查询休眠节点的成员状态 24

4. 从站/中继参数配置命令 26

4.1 设置从站的漫游切换 26
4.2 从站/中继的固定接入点 27
4.3 休眠与唤醒控制 28
4.4 读取当前网络的信号强度 29

5. 数据传输命令 30

5.1 发送数据 30
5.2 接收数据 33
5.3 查询发送缓冲区 34
5.4 主机的工作状态信息 35
5.5 全局报文描述符 35

6. 全局网络拓扑结构 37

6.1 查询特定主机的全局注册路径 37
6.2 查询特定主机的硬件 MAC 地址 38
6.3 查询最大的注册客户端的数量 39

6.4 查询注册节点的运行状态信息 40
7. 局域拓扑关系发现 44
7.1 读取本地局域拓扑的节点信息 44
8. 本地网络接入控制 48
8.1 设置最大和最小注册路由等级 48
9. 本地 I/O 状态提示 49
9.1 设置本地 I/O 事件与状态提示 49
10. 事件、通知与消息 50
11. 系统管理 53
11.1 系统软件复位 53
12. 附录-1：CRC16 算法 53
13. 附录-2：CRC32 算法 56

1. 二进制的通讯格式

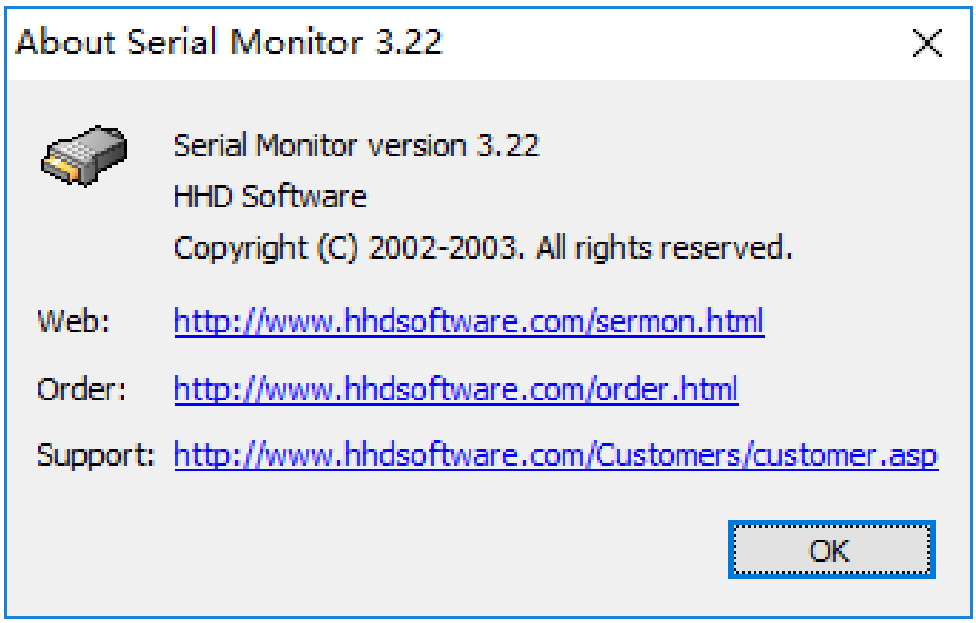
PC 和无线模块的接口遵循一个一个特定结构的二进制数据流。用结构体描述如下。

```
// ------------------------------------------------------------------------
// DESCRIPTION:
// ------------------------------------------------------------------------
typedef struct _NodeMsg
{
   // ---------------------------------------------------------------------
   // DESCRIPTION:
   // ---------------------------------------------------------------------
   unsigned char                                          m_iHeader;

   // ---------------------------------------------------------------------
   // DESCRIPTION:
   // ---------------------------------------------------------------------
   unsigned char                                          m_iOpCode;

   // ---------------------------------------------------------------------
   // DESCRIPTION:
   // ---------------------------------------------------------------------
   unsigned char                                          m_iValueA;

   // ---------------------------------------------------------------------
   // DESCRIPTION:
   // ---------------------------------------------------------------------
   unsigned char                                          m_iValueB;

   // ---------------------------------------------------------------------
   // DESCRIPTION:
   // ---------------------------------------------------------------------
   unsigned char                                          m_iValueC;

   // ---------------------------------------------------------------------
   // DESCRIPTION:
   // ---------------------------------------------------------------------
   unsigned char                                          m_iValueD;

   // ---------------------------------------------------------------------
   // DESCRIPTION:
   // ---------------------------------------------------------------------
   unsigned char                                          m_iAmount;

   // ---------------------------------------------------------------------
   // DESCRIPTION:
   // ---------------------------------------------------------------------
   unsigned short                                         m_iCRCode;

   // ---------------------------------------------------------------------
   // DESCRIPTION:
   // ---------------------------------------------------------------------
   unsigned char                                          m_pBuffer[245];

} NodeMsg;
```

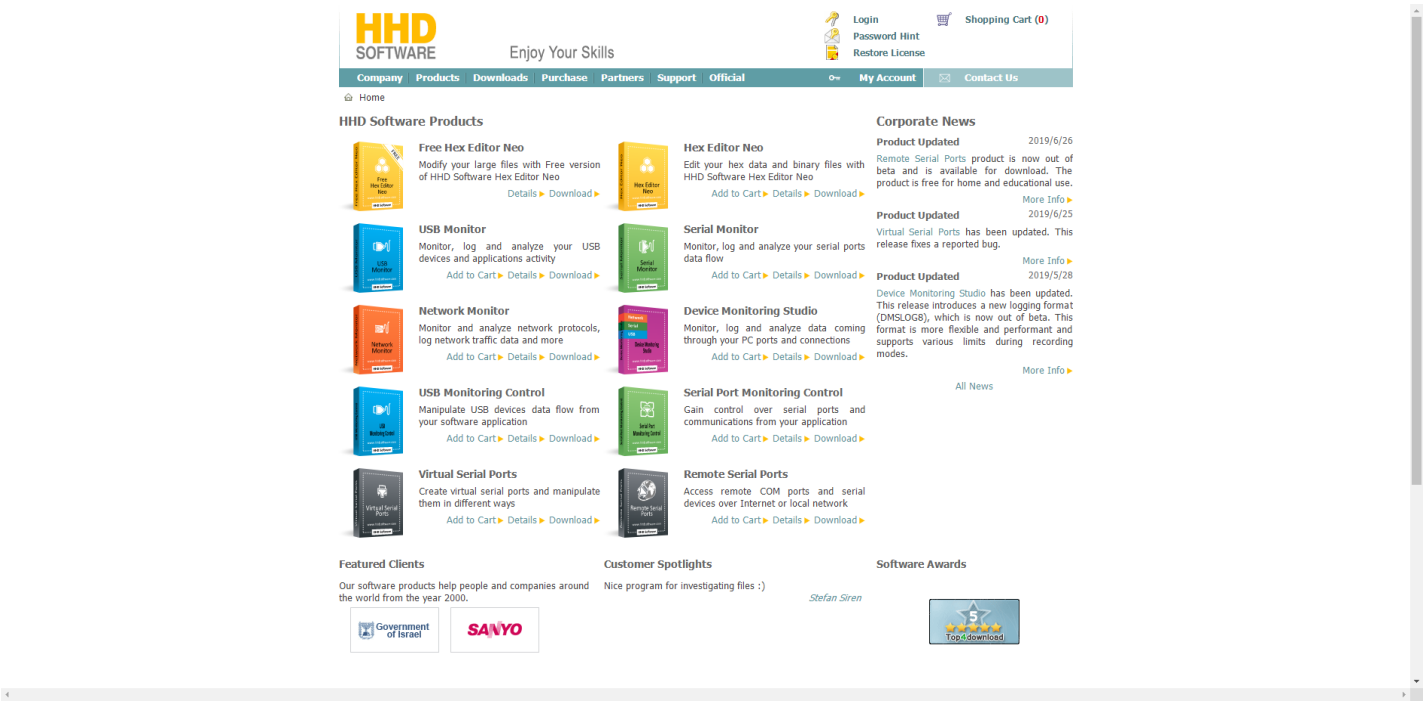
为了便于分析和理解数据通讯格式，建议在 Windows PC 平台上安装串口监控软件，进行内核态的抓包分析。

下面是一款抓包软件的使用介绍。

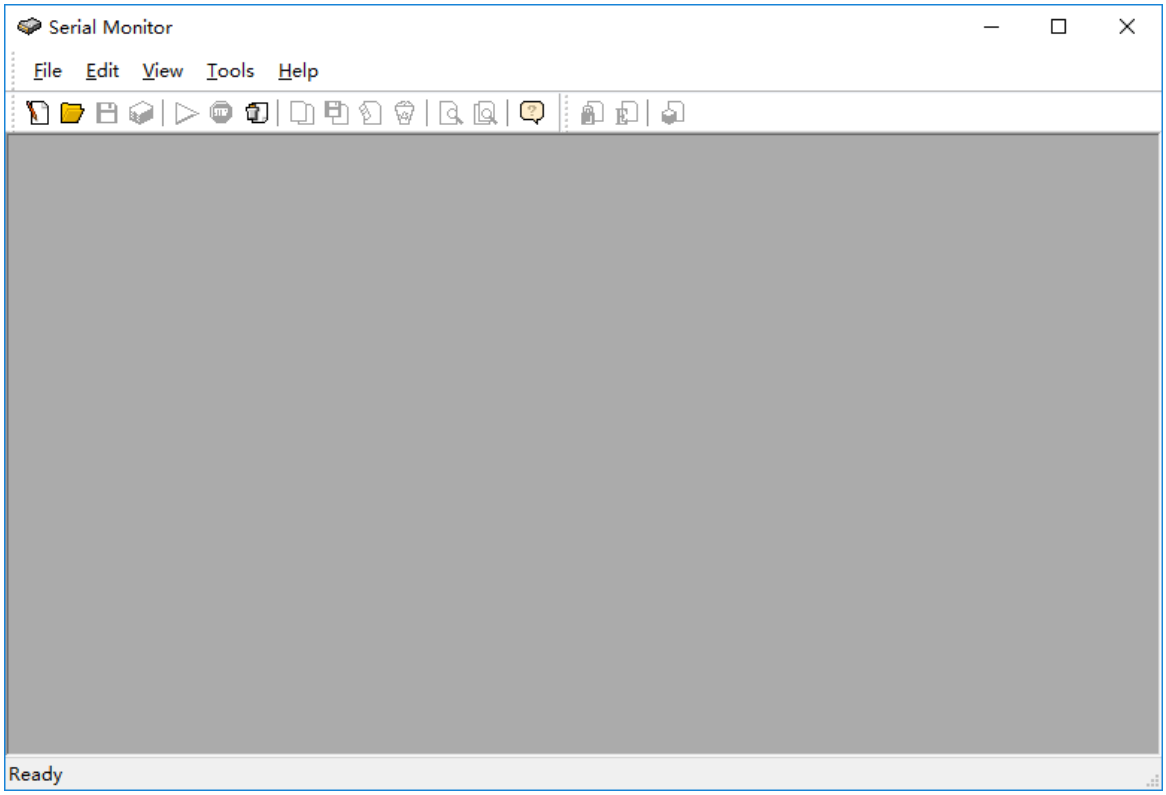
版权信息：



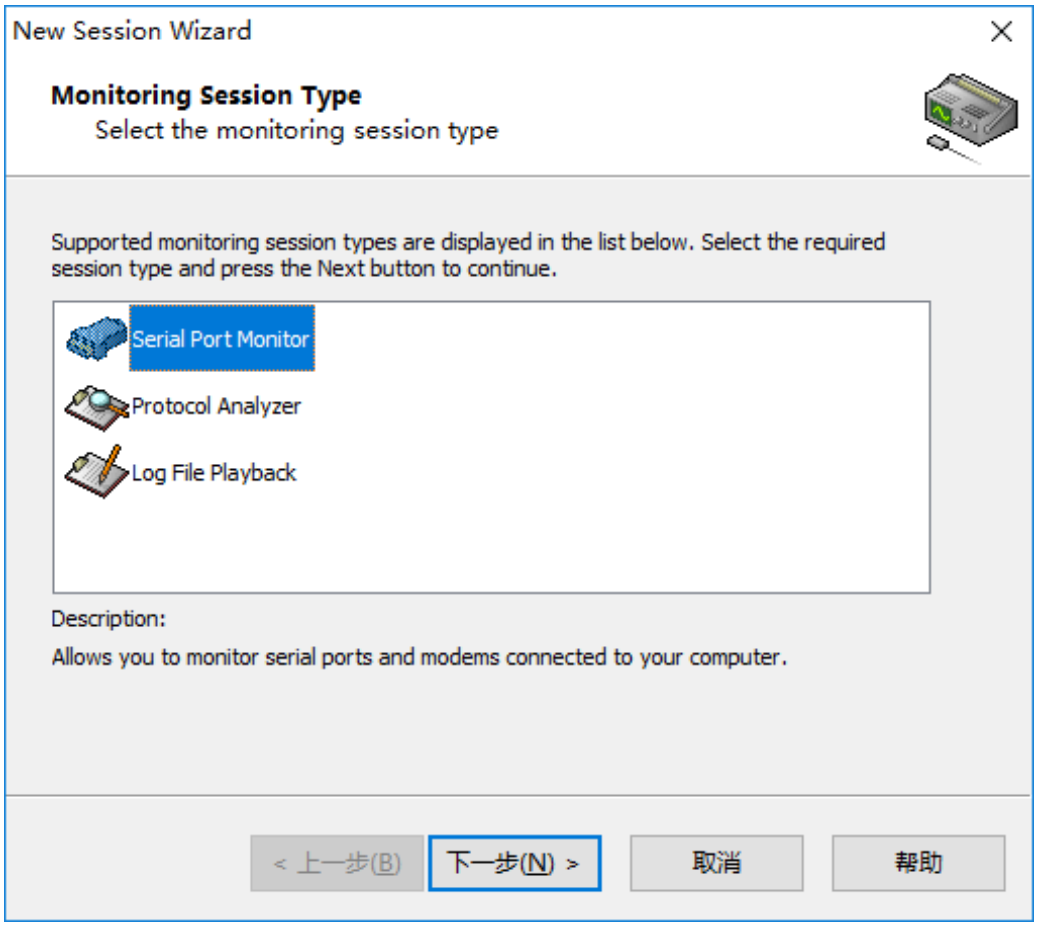
网站页面：https://www.hhdsoftware.com/



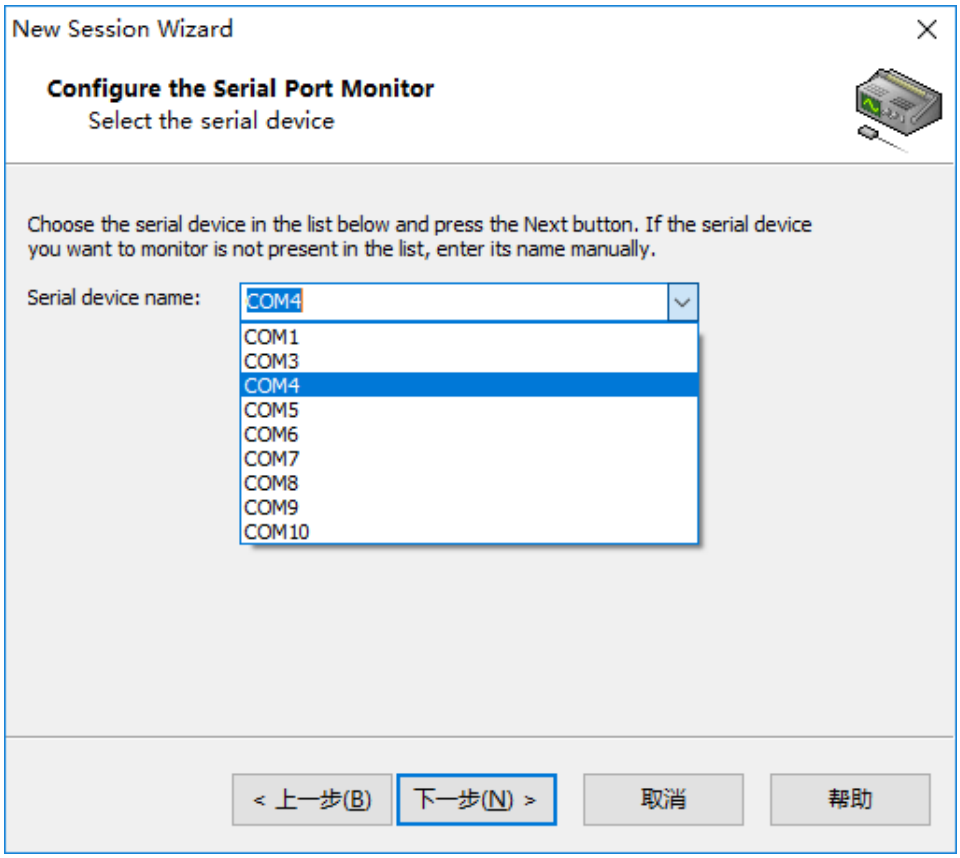
启动界面：


Serial Monitor
File Edit View Tools Help
Ready

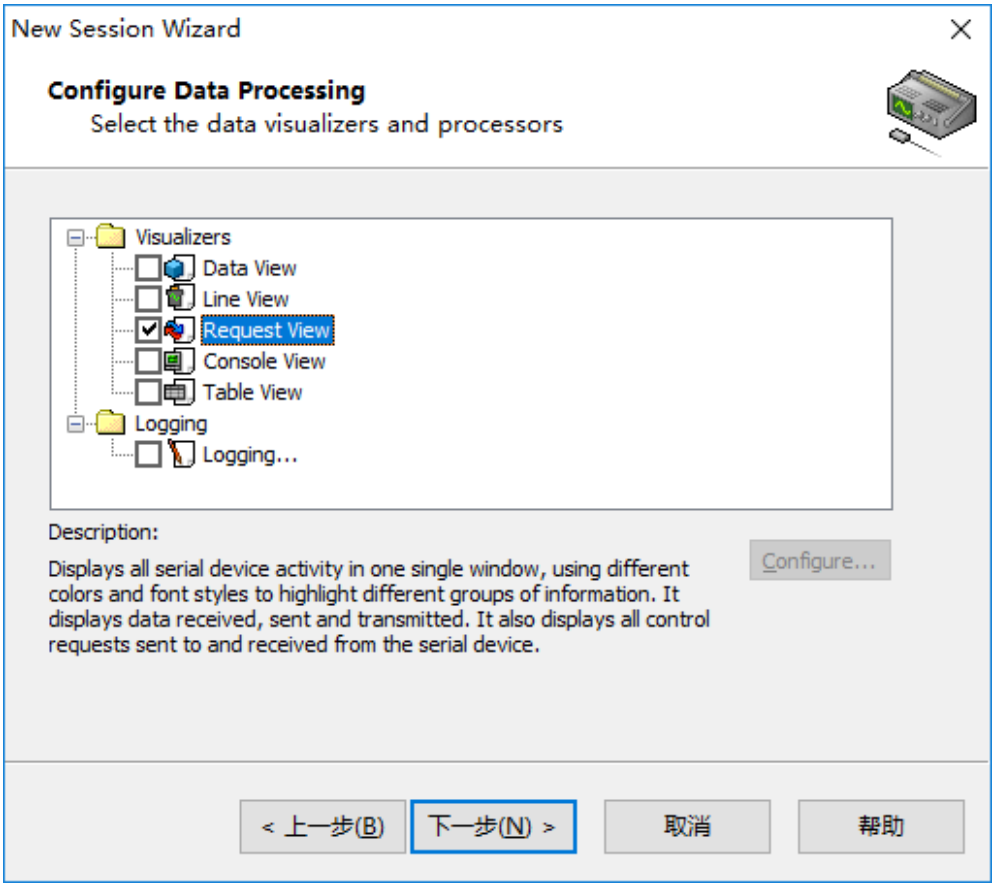
新建会话任务：


New Session Wizard
Monitoring Session Type
Select the monitoring session type
Supported monitoring session types are displayed in the list below. Select the required session type and press the Next button to continue.
Serial Port Monitor
Protocol Analyzer
Log File Playback
Description:
Allows you to monitor serial ports and modems connected to your computer.
< 上一步(B)
下一步(N) >
取消
帮助

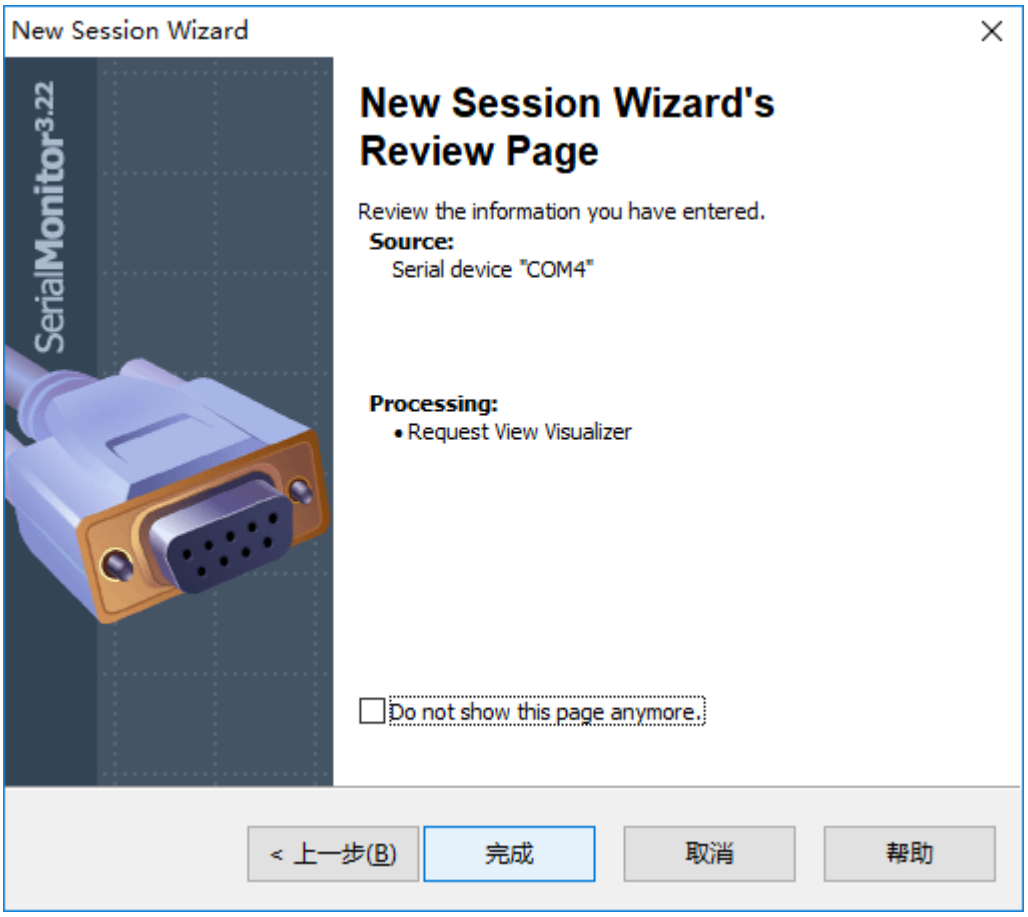
选择监控的串口



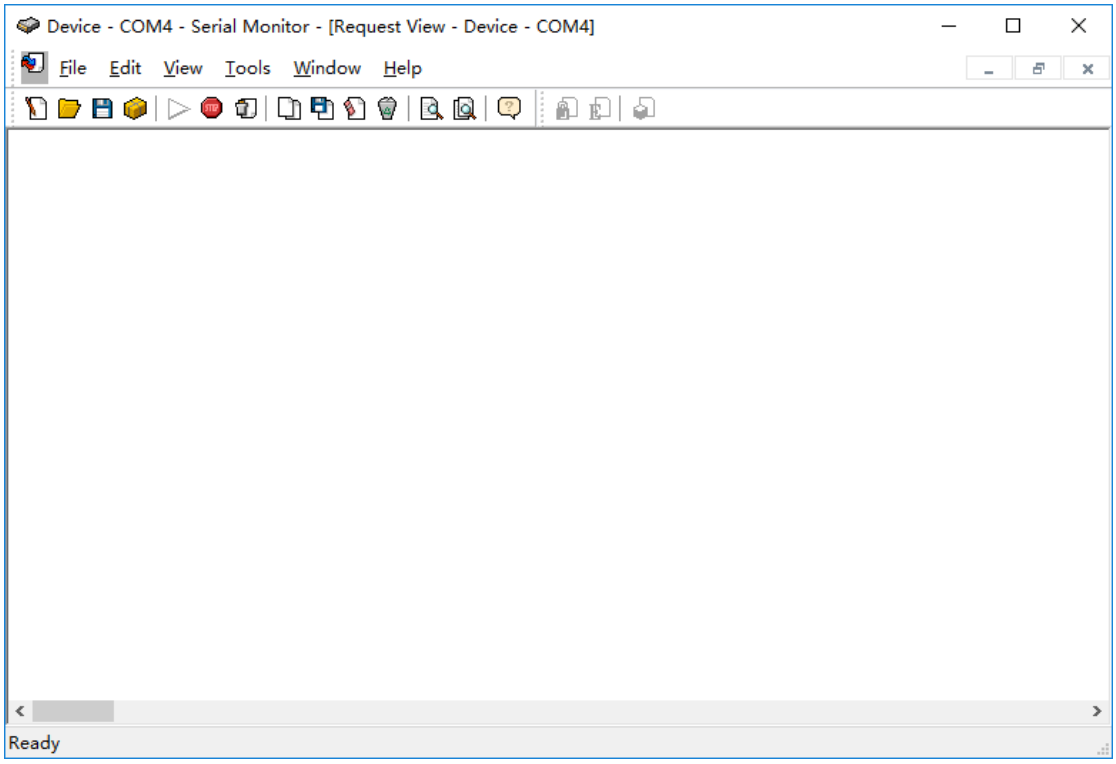
选择格式化输出模式：



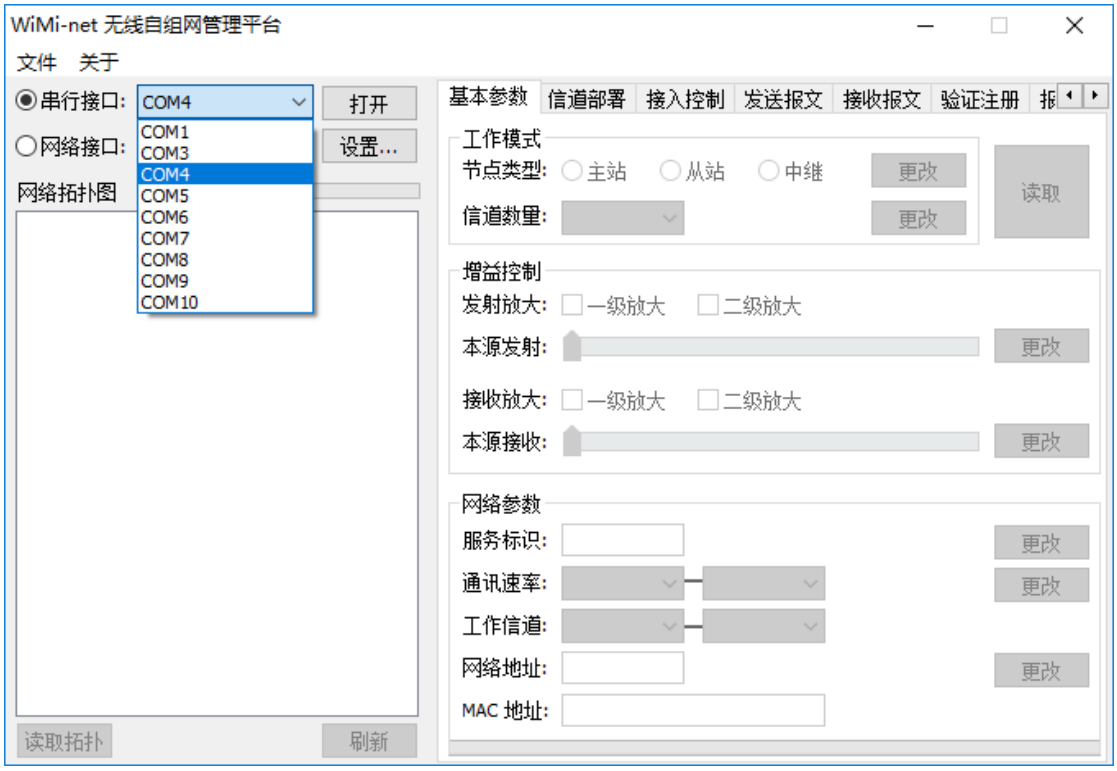
完成会话，开始进入抓包流程



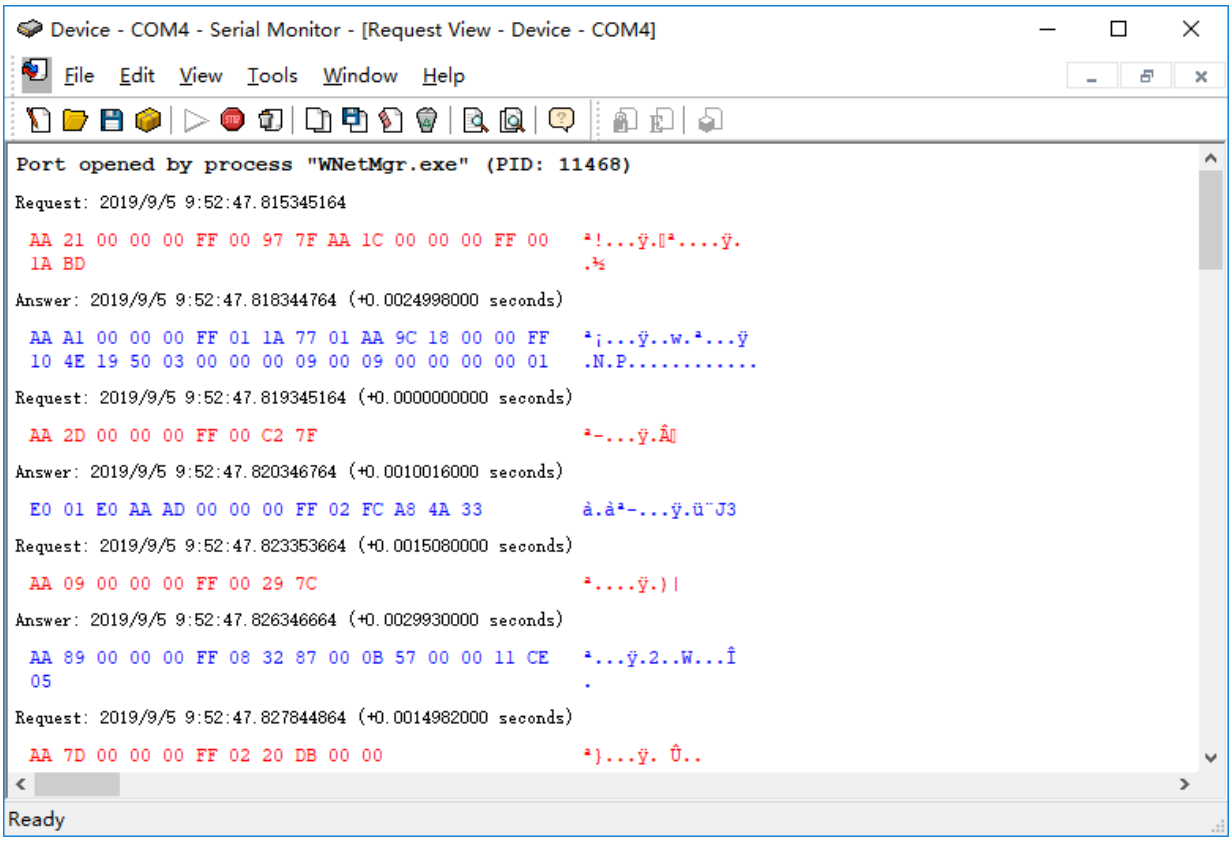
空白的抓包任务界面



启动“WiMi-net 无线自组网管理平台”应用软件，打开监控的串口



查看串口抓包的输出如下图所示，鼠标点击页面，即停止显示刷新。



1.1 二进制流接口格式

	平面化表示与结构化表示的对应关系		
字节	参数	数值	说明
1	m_iHeader	0XAA	报文的同步字节，用于检测报文的起始位置，常量 0XAA
2	m_iOpCode		该报文的命令字，取值范围 0X00-0XFF
3	m_iValueA		报文命令字的第一个参数，具体意义取决于 m_iOpCode
4	m_iValueB		报文命令字的第二个参数，具体意义取决于 m_iOpCode
5	m_iValueC		报文命令字的第三个参数，具体意义取决于 m_iOpCode
6	m_iValueD		报文命令字的第四个参数，具体意义取决与 m_iOpCode
7	m_iAmount		扩展部分的数据报文 m_pBuffer 的长度，取值范围 0X00-0XF5
8	m_iCRCode.MSB	0X##	整个报文的 16 位 CRC 校验和的高字节
9	m_iCRCode.LSB	0X##	整个报文的 16 位 CRC 校验和的低字节
10	m_pBuffer		扩展部分的数据报文，最多可容纳 245(0XF5)个字节
备注-1	CRC 计算从第一个字节开始 m_iHeader，至 m_pBuffer 的最后一个字节结束；计算之前需要将 CRC 高低字节全部清零，然后将计算结果写到 CRC 的位置，保持高字节在前，低字节在后		
备注-2	请求命令的应答命令，其命令字 = 请求命令 + 0X80		
备注-3	报文的同步字节 m_iHeader 是一个固定的常数，取值 0XAA		
备注-4	如果命令没有用到参数，默认将参数 m_iValueA，m_iValueB， m_iValueC， m_iValueD 填写为 0X00		
备注-5	如果命令没有扩展部分，则 m_pBuffer 区域不要填写，同时将 m_iAmount 字段填写为 0X00		
示例-1	请求命令：AA 26 00 00 00 00 01 F8 24 01 应答命令：AA A6 00 00 00 00 02 F3 BB 01 00		

1.2 二进制流接口 CRC 计算方式

（1）多项式：f（X）=1+X^2+X^{15}+X^{16}，该多项式和工业标准的 Modbus-RTU 完全一样

（2）发送端 CRC 计算：先把 m_iCRCode 位置清零，然后计算整个报文（从第一个字节 m_iHeader 开始至 m_pBuffer 的最后一个字节结束）的 CRC 数据，注意高字节在前（MSB）

（3）接收端 CRC 验算：首先从报文中读取 CRC，然后将报文的 m_iCRCode 清零，计算出整个报文（同上）的 CRC，比较这两个 CRC 验证报文的正确性。

（4）应答命令字：请求命令字 + 0X80

（5）CRC算法：工业标准CRC-16，请参考：“H*CRC-16 查表算法*H”和“H*CRC-16 查表算法*H”。

CRC 的计算可以借助如下的工具软件—WiMi-net 无线自组网管理平台

WiMi-net 无线自组网管理平台软件有两种显示模式，上图中红色框住的部分是结构化显示模式，绿色部分是平面显示的模式。为了测试指令是否正确，可以将平面化的指令复制出来，在串口调试助手中进行测试，看看是否有回应。如果可以收到应答，那么说明 CRC 的计算是正确的。

采用串口助手测试的时候，切记要正确设置数据的格式：

(1) 发送需要采用十六进制(勾选“HEX 发送”)

(2) 接收显示需要采用十六进制(勾选“HEX 显示”)

1.3 名词术语和缩写简记符

(1) MSB：Most Significant Byte，双字节变量的高字节 ，即 B8-B15

(2) LSB：Least Significant Byte，双字节变量的低字节，即 B0-B7

(3) Addr：无线节点的 16 位网络地址

(4) MAC：无线节点的 64 位 MAC 地址

1.4 32 位系统上的编程注意事项

(1) 如果采用 C 语言的结构体访问内存数据，需要调整到单字节边界（默认为 4 字节边界）

(2) 系统的多字节变量默认是 Big-Endian，也就是大端在前，ARM 和 X86 都是 Little-Endian，需要进行字节顺序转换

(3) CRC-16 的计算多项式和 Modbus 完全一样，但是其位置不一样，作用范围也不一样

(4) CRC-16 在计算之前需要采用默认的数值 0X00 作为填充。

1.5 关于缺省参数和配置

(1) WiMi-net 基于自组网技术设计，所有的站点，包括主站，从站，和中继，都有默认的出厂配置参数，通常情况下无需人工配置

(2) 从站和中继的网络（PAN）标示可以通过对码的方式予以获取

(3) 从站和中继节点的工作频带，信道，网络路径，时槽等参数会通过自组网机制自动获取，不需要用户配置

2. 公共参数配置命令

2.1 节点工作模式

指令	节点工作模式				
字节	参数	设置请求	读取请求	设置应答	读取应答
1	m_iHeader	0XAA	0XAA	0XAA	0XAA
2	m_iOpCode	0X20	0X21	0XA0	0XA1
3	m_iValueA	0X00	0X00	0X00	0X00
4	m_iValueB	0X00	0X00	0X00	0X00
5	m_iValueC	0X00	0X00	0X00	0X00
6	m_iValueD	0X00	0X00	0X00	0X00
7	m_iAmount	0X01	0X00	0X01	0X01
8	m_iCRCode.MSB	0X##	0X##	0X##	0X##
9	m_iCRCode.LSB	0X##	0X##	0X##	0X##
10	m_pBuffer[0]	工作模式		工作模式	工作模式
参数-1	工作模式： 0X00：从站 0X01：主站 0X02：中继				
设置示例	请求：AA 20 00 00 00 00 01 58 0F 01 应答：AA A0 00 00 00 00 02 2E 1A 01 07				
读取示例	请求：AA 21 00 00 00 FF 00 97 7F 应答：AA A1 00 00 00 FF 02 E4 5F 01 07				

2.2 网络服务标示

指令	网络服务标示				
字节	参数	设置请求	读取请求	设置应答	读取应答
1	m_iHeader	0XAA	0XAA	0XAA	0XAA
2	m_iOpCode	0X10	0X11	0X90	0X91
3	m_iValueA	0X00	0X00	0X00	0X00
4	m_iValueB	0X00	0X00	0X00	0X00
5	m_iValueC	0X00	0X00	0X00	0X00
6	m_iValueD	0X00	0X00	0X00	0X00
7	m_iAmount	0X02	0X00	0X02	0X02
8	m_iCRCode.MSB	0X##	0X##	0X##	0X##
9	m_iCRCode.LSB	0X##	0X##	0X##	0X##
10	m_pBuffer[0]	PAN.MSB		PAN.MSB	PAN.MSB
11	m_pBuffer[1]	PAN.LSB		PAN.LSB	PAN.LSB
备注-1	PAN：Personal Area Network，简称为网络标识，16 位数值 PAN.MSB：网络标识 PAN 的高字节 PAN.LSB：网络标识 PAN 的低字节				
设置示例	请求：AA 10 00 00 00 00 02 39 10 13 87 应答：AA 90 00 00 00 00 02 D1 17 13 87				
读取示例	请求：AA 11 00 00 00 00 00 97 4C 应答：AA 91 00 00 00 00 02 14 46 13 87				

2.3 16 位网络地址

指令	16 位网络地址				
字节	参数	设置请求	读取请求	设置应答	读取应答
1	m_iHeader	0XAA	0XAA	0XAA	0XAA
2	m_iOpCode	0X2C	0X2D	0XAC	0XAD
3	m_iValueA	0X00	0X00	0X00	0X00
4	m_iValueB	0X00	0X00	0X00	0X00
5	m_iValueC	0X00	0X00	0X00	0X00
6	m_iValueD	0X00	0X00	0X00	0X00
7	m_iAmount	0X02	0X00	0X02	0X02
8	m_iCRCode.MSB	0X##	0X##	0X##	0X##
9	m_iCRCode.LSB	0X##	0X##	0X##	0X##
10	m_pBuffer[0]	NET.MSB		NET.MSB	NET.MSB
11	m_pBuffer[1]	NET.LSB		NET.LSB	NET.LSB
备注-1	NET：节点的网络地址，16 位数值 NET.MSB：节点地址 NET 的高字节 NET.LSB：节点地址 NET 的低字节 节点地址的编码范围：0X0001~0XFFF9，其中 (1) 0X0000 是虚拟地址，代表主站，不能用于节点编址 (2) 0XFFFF 是虚拟地址，代表全网广播，不能用于节点编址 (3) 0XFFFA~0XFFFE 预留地址，不能用于节点编址				
设置示例	**请求：**AA 2C 00 00 00 00 02 E5 F0 93 A8 **应答：**AA AC 00 00 00 00 02 0D F7 93 A8				
读取示例	**请求：**AA 2D 00 00 00 00 00 D6 4F **应答：**AA AD 00 00 00 00 02 C8 A6 93 A8				

2.4 64 位 MAC 地址

指令	64 位 MAC 地址				
字节	参数	设置请求	读取请求	设置应答	读取应答
1	m_iHeader	0XAA	0XAA	0XAA	0XAA
2	m_iOpCode	0X08	0X09	0X88	0X89
3	m_iValueA	0X00	0X00	0X00	0X00
4	m_iValueB	0X00	0X00	0X00	0X00
5	m_iValueC	0X00	0X00	0X00	0X00
6	m_iValueD	0X00	0X00	0X00	0X00
7	m_iAmount	0X08	0X00	0X08	0X08
8	m_iCRCode.MSB	0X##	0X##	0X##	0X##
9	m_iCRCode.LSB	0X##	0X##	0X##	0X##
10	m_pBuffer[0]	MAC_ID[0]		MAC_ID[0]	MAC_ID[0]
11	m_pBuffer[1]	MAC_ID[1]		MAC_ID[1]	MAC_ID[1]
12	m_pBuffer[2]	MAC_ID[2]		MAC_ID[2]	MAC_ID[2]
13	m_pBuffer[3]	MAC_ID[3]		MAC_ID[3]	MAC_ID[3]
14	m_pBuffer[4]	MAC_ID[4]		MAC_ID[4]	MAC_ID[4]
15	m_pBuffer[5]	MAC_ID[5]		MAC_ID[5]	MAC_ID[5]
16	m_pBuffer[6]	MAC_ID[6]		MAC_ID[6]	MAC_ID[6]
17	m_pBuffer[7]	MAC_ID[7]		MAC_ID[7]	MAC_ID[7]
备注-1	MAC：节点的 MAC 地址，64 位数值 MAC_ID[0] ~ MAC_ID[7]：8 字节的 MAC 地址，高字节在前，低字节在后				
备注-2	MAC 地址通常是只读的，不允许修改，仅在部分型号的产品上允许设置				
设置示例	请求：AA 08 00 00 00 00 08 CE EB 24 E4 C3 03 55 6B 59 17 应答：AA 88 00 00 00 00 08 6E 8A 24 E4 C3 03 55 6B 59 17				
读取示例	请求：AA 09 00 00 00 00 00 3D 4C 应答：AA 89 00 00 00 00 08 FE 4B 24 E4 C3 03 55 6B 59 17				

2.5 节点的工作频带和信道

指令	节点的工作频带和信道				
字节	参数	设置请求	读取请求	设置应答	读取应答
1	m_iHeader	0XAA	0XAA	0XAA	0XAA
2	m_iOpCode	0X12	0X13	0X92	0X93
3	m_iValueA	0X00	0X00	0X00	0X00
4	m_iValueB	0X00	0X00	0X00	0X00
5	m_iValueC	0X00	0X00	0X00	0X00
6	m_iValueD	0X00	0X00	0X00	0X00
7	m_iAmount	0X02	0X00	0X02	0X02
8	m_iCRCode.MSB	0X##	0X##	0X##	0X##
9	m_iCRCode.LSB	0X##	0X##	0X##	0X##
10	m_pBuffer[0]	Band		Band	Band
11	m_pBuffer[1]	Channel		Channel	Channel
备注-1	Band：工作频带，双频带硬件平台取值范围 0~1，四频带硬件平台取值范围 0~3 Channel：工作信道，32 信道配置取值范围 0~31，64 信道配置取值范围 0~63 具体的工作频带和信道范围取决于硬件设计和固件配置，不同产品或者硬件平台之间参数可能不相同				
设置示例	**请求：**AA 12 00 00 00 00 02 A1 FC 00 00 **应答：**AA 92 00 00 00 00 02 49 FB 00 00				
读取示例	**请求：**AA 13 00 00 00 00 00 4E CD **应答：**AA 93 00 00 00 00 02 8C AA 00 00				

2.6 配置参数恢复出厂配置

指令	配置参数恢复出厂配置		
字节	参数	请求指令	应答指令
1	m_iHeader	0XAA	0XAA
2	m_iOpCode	0X0F	0X8F
3	m_iValueA	0X00	0X00
4	m_iValueB	0X00	0X00
5	m_iValueC	0X00	0X00
6	m_iValueD	0X00	0X00
7	m_iAmount	0X00	0X00
8	m_iCRCode.MSB	0X##	0X##
9	m_iCRCode.LSB	0X##	0X##
备注	清除节点的配置参数，对码信息和休眠节点信息		
请求示例	AA 0F 00 00 00 00 00 17 CC		
应答示例	AA 8F 00 00 00 00 00 77 C4		

2.7 节点的对码状态

指令	节点的对码状态				
字节	参数	设置请求	读取请求	设置应答	读取应答
1	m_iHeader	0XAA	0XAA	0XAA	0XAA
2	m_iOpCode	0X15	0X24	0X95	0XA4
3	m_iValueA	0X00	0X00	0X00	0X00
4	m_iValueB	0X00	0X00	0X00	0X00
5	m_iValueC	0X00	0X00	0X00	0X00
6	m_iValueD	0X00	0X00	0X00	0X00
7	m_iAmount	0X01	0X00	0X01	0X01
8	m_iCRCode.MSB	0X##	0X##	0X##	0X##
9	m_iCRCode.LSB	0X##	0X##	0X##	0X##
10	m_pBuffer[0]	Pair.Status		Pair.Status	Pair.Status
备注-1	Pair.Status：节点的对码状态 0X00：取消当前的对码行为 0X01：从站启动向主站对码请求 0X02：主站开启从站对码窗口 0X03：从当前的网络中注销并退出				
设置示例	请求：AA 15 00 00 00 00 01 C9 A5 00 应答：AA 95 00 00 00 00 01 0F C4 00				
读取示例	请求：AA 24 00 00 00 00 00 BC 8F 应答：AA A4 00 00 00 00 01 9E 9D 00				

2.8 射频输出链路配置

指令	射频输出链路配置				
字节	参数	设置请求	读取请求	设置应答	读取应答
1	m_iHeader	0XAA	0XAA	0XAA	0XAA
2	m_iOpCode	0X3A	0X3B	0XBA	0XBB
3	m_iValueA	0X00	0X00	0X00	0X00
4	m_iValueB	0X00	0X00	0X00	0X00
5	m_iValueC	0X00	0X00	0X00	0X00
6	m_iValueD	0X00	0X00	0X00	0X00
7	m_iAmount	0X03	0X00	0X04	0X04
8	m_iCRCode.MSB	0X##	0X##	0X##	0X##
9	m_iCRCode.LSB	0X##	0X##	0X##	0X##
10	m_pBuffer[0]	Chip.Status		Chip.Status	Chip.Status
11	m_pBuffer[1]	TPA0.Status		Chip.IDCode	Chip.IDCode
12	m_pBuffer[2]	TPA1.Status		TPA0.Status	TPA0.Status
	m_pBuffer[3]			TPA1.Status	TPA1.Status
备注-1	Chip.Status：射频芯片的本源发射配置，具体参考芯片规格书，默认填写 0X00 Chip.IDCode：射频芯片的型号识别代码 TPA0.Status：一级功放的开启或者关闭状态 Bit0：使能状态，0X01=开启功放，0X00=关闭功放 Bit7：编辑状态：0X01=可以编辑，0X00=不可编辑（只读） TPA1.Status：二级功放的开启或者关闭状态 Bit0：使能状态，0X01=开启功放，0X00=关闭功放 Bit7：编辑状态：0X01=可以编辑，0X00=不可编辑（只读）				
设置示例	请求：AA 3A 00 00 00 00 03 30 21 00 00 00 应答：AA BA 00 00 00 00 04 C6 E4 00 00 80 80				
读取示例	请求：AA 3B 00 00 00 00 00 F0 CE 应答：AA BB 00 00 00 00 04 05 19 00 00 80 80				

2.9 射频输入链路配置

指令	射频输入链路配置				
字节	参数	设置请求	读取请求	设置应答	读取应答
1	m_iHeader	0XAA	0XAA	0XAA	0XAA
2	m_iOpCode	0X34	0X35	0XB4	0XB5
3	m_iValueA	0X00	0X00	0X00	0X00
4	m_iValueB	0X00	0X00	0X00	0X00
5	m_iValueC	0X00	0X00	0X00	0X00
6	m_iValueD	0X00	0X00	0X00	0X00
7	m_iAmount	0X03	0X00	0X04	0X04
8	m_iCRCode.MSB	0X##	0X##	0X##	0X##
9	m_iCRCode.LSB	0X##	0X##	0X##	0X##
10	m_pBuffer[0]	Chip.Status		Chip.Status	Chip.Status
11	m_pBuffer[1]	LNA0.Status		Chip.IDCode	Chip.IDCode
12	m_pBuffer[2]	LNA1.Status		LNA0.Status	LNA0.Status
	m_pBuffer[3]			LNA1.Status	LNA1.Status
备注-1	Chip.Status：射频芯片的本源接收配置，具体参考芯片规格书，默认填写 0X00 Chip.IDCode：射频芯片的型号识别代码 LNA0.Status：一级低噪放的开启或者关闭状态 Bit0：使能状态，0X01=开启低噪放，0X00=关闭低噪放 Bit7：编辑状态：0X01=可以编辑，0X00=不可编辑（只读） LNA1.Status：二级低噪放的开启或者关闭状态 Bit0：使能状态，0X01=开启低噪放，0X00=关闭低噪放 Bit7：编辑状态：0X01=可以编辑，0X00=不可编辑（只读）				
设置示例	请求：AA 34 00 00 00 00 03 D8 15 00 00 00 应答：AA B4 00 00 00 00 04 B1 0C 00 00 80 00				
读取示例	请求：AA 35 00 00 00 00 00 7C 4F 应答：AA B5 00 00 00 00 04 72 F1 00 00 80 00				

3. 无线主站参数配置命令

3.1 无线主站的信道配置模式

指令	无线主站的信道配置模式				
字节	参数	设置请求	读取请求	设置应答	读取应答
1	m_iHeader	0XAA	0XAA	0XAA	0XAA
2	m_iOpCode	0X4B	0X4C	0XCB	0XCC
3	m_iValueA	0X00	0X00	0X00	0X00
4	m_iValueB	0X00	0X00	0X00	0X00
5	m_iValueC	0X00	0X00	0X00	0X00
6	m_iValueD	0X00	0X00	0X00	0X00
7	m_iAmount	0X01	0X00	0X01	0X01
8	m_iCRCode.MSB	0X##	0X##	0X##	0X##
9	m_iCRCode.LSB	0X##	0X##	0X##	0X##
10	m_pBuffer[0]	Channel.Mode		Channel.Mode	Channel.Mode
备注-1	Channel.Mode：信道配置模式 0X00:固定信道 0X01:出厂第一次配置搜索 0X02:永远搜索				
设置示例	请求：AA 4B 00 00 00 00 01 AA 15 00 应答：AA CB 00 00 00 00 01 6C 74 00				
读取示例	请求：AA 4C 00 00 00 00 00 32 88 应答：AA CC 00 00 00 00 01 5C 52 00				

3.2 删除休眠节点

指令	删除休眠节点		
字节	参数	请求指令	应答指令
1	m_iHeader	0XAA	0XAA
2	m_iOpCode	0X23	0XA3
3	m_iValueA	0X00	0X00
4	m_iValueB	0X00	0X00
5	m_iValueC	0X00	0X00
6	m_iValueD	0X00	0X00
7	m_iAmount	0X02 * N（节点个数）	0X00
8	m_iCRCode.MSB	0X##	0X##
9	m_iCRCode.LSB	0X##	0X##
10	m_pBuffer[0]	删除节点 1 网络地址高字节	
11	m_pBuffer[1]	删除节点 1 网络地址低字节	
12	m_pBuffer[2]	删除节点 2 网络地址高字节	
13	m_pBuffer[3]	删除节点 2 网络地址低字节	
14	m_pBuffer[4]	依此类推 ……	
15	m_pBuffer[5]	依此类推 ……	
备注	如果 m_iAmount = 0X00，则清除所有的休眠节点		
请求示例	AA 23 00 00 00 00 04 64 25 34 F1 7E 50		
应答示例	AA A3 00 00 00 00 00 3A C6		

3.3 删除对码节点（按照 16 位网络地址存储格式）

指令	删除对码节点（按照 16 位网络地址存储格式）		
字节	参数	请求指令	应答指令
1	m_iHeader	0XAA	0XAA
2	m_iOpCode	0X43	0XC3
3	m_iValueA	0X00	0X00
4	m_iValueB	0X00	0X00
5	m_iValueC	0X00	0X00
6	m_iValueD	0X00	0X00
7	m_iAmount	0X02 * N（节点个数）	0X02 * N（节点个数）
8	m_iCRCode.MSB	0X##	0X##
9	m_iCRCode.LSB	0X##	0X##
10	m_pBuffer[0]	删除节点 1 网络地址高字节	删除节点 1 网络地址高字节
11	m_pBuffer[1]	删除节点 1 网络地址低字节	删除节点 1 网络地址低字节
12	m_pBuffer[2]	删除节点 2 网络地址高字节	删除节点 2 网络地址高字节
13	m_pBuffer[3]	删除节点 2 网络地址低字节	删除节点 2 网络地址低字节
14	m_pBuffer[4]	依此类推 ……	依此类推 ……
15	m_pBuffer[5]	依此类推 ……	依此类推 ……
备注	如果 m_iAmount = 0X00，则清除所有的对码节点		
请求示例	AA 43 00 00 00 00 04 E4 A5 34 F1 7E 50		
应答示例	AA C3 00 00 00 00 04 9A A7 34 F1 7E 50		

3.4 删除对码节点（按照 64 位 MAC 地址存储格式）

指令	删除对码节点（按照 64 位 MAC 地址存储格式）		
字节	参数	请求指令	应答指令
1	m_iHeader	0XAA	0XAA
2	m_iOpCode	0X41	0XC1
3	m_iValueA	0X00	0X00
4	m_iValueB	0X00	0X00
5	m_iValueC	0X00	0X00
6	m_iValueD	0X00	0X00
7	m_iAmount	0X08 * N（节点个数）	0X00
8	m_iCRCode.MSB	0X##	0X##
9	m_iCRCode.LSB	0X##	0X##
10	m_pBuffer[0]	删除节点 1 的 MAC_ID[0]	
11	m_pBuffer[1]	删除节点 1 的 MAC_ID[1]	
12	m_pBuffer[2]	删除节点 1 的 MAC_ID[2]	
13	m_pBuffer[3]	删除节点 1 的 MAC_ID[3]	
14	m_pBuffer[4]	删除节点 1 的 MAC_ID[4]	
15	m_pBuffer[5]	删除节点 1 的 MAC_ID[4]	
16	m_pBuffer[6]	删除节点 1 的 MAC_ID[6]	
17	m_pBuffer[7]	删除节点 1 的 MAC_ID[7]	
18	m_pBuffer[8]	依此类推 ……	
19	m_pBuffer[9]	依此类推 ……	
备注	如果 m_iAmount = 0X00，则清除所有的对码节点		
请求示例	AA 41 00 00 00 00 08 68 1C 24 E4 C3 03 55 6B 59 17		
应答示例	AA C1 00 00 00 00 00 CB 41		

3.5 查询休眠节点的成员数量

指令	查询休眠节点的成员数量		
字节	参数	请求指令	应答指令
1	m_iHeader	0XAA	0XAA
2	m_iOpCode	0X3C	0XBC
3	m_iValueA	0X00	0X00
4	m_iValueB	0X00	0X00
5	m_iValueC	0X00	0X00
6	m_iValueD	0X00	0X00
7	m_iAmount	0X00	0X03
8	m_iCRCode.MSB	0X##	0X##
9	m_iCRCode.LSB	0X##	0X##
10	m_pBuffer[0]		Count.MSB
11	m_pBuffer[1]		Count.LSB
12	m_pBuffer[2]		Unit.Size
备注	Count.MSB：休眠节点个数高字节 Count.LSB：休眠节点个数低字节 Unit.Size：单个节点配置信息的大小，单位为字节		
请求示例	AA 3C 00 00 00 08 00 76 8D		
应答示例	AA BC 00 00 00 08 03 D9 D1 00 16 05		

3.6 查询休眠节点的成员状态

指令	查询休眠节点的成员状态		
字节	参数	请求指令	应答指令
1	m_iHeader	0XAA	0XAA
2	m_iOpCode	0X3D	0XBD
3	m_iValueA	Index.MSB	Index.MSB
4	m_iValueB	Index.LSB	Index.LSB
5	m_iValueC	Request.Node.Size	Respond.Node.Size
6	m_iValueD	0X00	0X00
7	m_iAmount	0X00	Unit.Size * Respond.Node.Size
8	m_iCRCode.MSB	0X##	0X##
9	m_iCRCode.LSB	0X##	0X##
10	m_pBuffer[0]		结构体成员信息，参考定义
11	m_pBuffer[1]		依此类推 ……
12	m_pBuffer[2]		依此类推 ……
备注	Index.MSB：休眠节点成员内部序号的高字节 Index.LSB：休眠节点成员内部序号的低字节 Request.Node.Size：本次操作需要读取的休眠节点数量，单位为字节 Respond.Node.Size：本次操作实际读取的休眠节点数量，单位为字节 Unit.Size：单个休眠节点的结构体大小，单位为字节		
请求示例	AA 3D 00 00 FF 08 00 AE 58		
应答示例	AA BD 00 00 16 08 6E FA 63 72 69 00 00 02 3E F3 00 40 03 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		

休眠节点信息的 C 语言结构体信息如下：

```
// ---------------------------------------------------------------------------
// DESCRIPTION:
// ---------------------------------------------------------------------------
typedef struct _WOR_BandChannel_
{
   // ---------------------------------------------------------------------------
   // DESCRIPTION: The wake on radio band
   // ---------------------------------------------------------------------------
   unsigned char                                                m_iBand;

   // ---------------------------------------------------------------------------
   // DESCRIPTION: The wake on radio channel
   // ---------------------------------------------------------------------------
   unsigned char                                                m_iChannel;

} WOR_BandChannel;

// ---------------------------------------------------------------------------
// DESCRIPTION:
// ---------------------------------------------------------------------------
typedef union _Wakeup_Profile_
{
   // ---------------------------------------------------------------------------
   // DESCRIPTION:
   // ---------------------------------------------------------------------------
   unsigned short                                               m_iParent;

   // ---------------------------------------------------------------------------
   // DESCRIPTION: The wake on radio band
   // ---------------------------------------------------------------------------
   WOR_BandChannel                                              m_WORadio;

} Wakeup_Profile;

// ---------------------------------------------------------------------------
// DESCRIPTION:
// ---------------------------------------------------------------------------
typedef struct _WiMinet_WakeUp_
{
   // ---------------------------------------------------------------------------
   // DESCRIPTION: The device address
   // ---------------------------------------------------------------------------
   unsigned short                                               m_iTxAddr;

   // ---------------------------------------------------------------------------
   // DESCRIPTION: The wakeup profile
   // ---------------------------------------------------------------------------
   Wakeup_Profile                                               m_Profile;

   // ---------------------------------------------------------------------------
   // DESCRIPTION: The WOR packet counter
   // ---------------------------------------------------------------------------
   unsigned char                                                m_iNative:0X01;
```

```
    // ------------------------------------------------------------------------
    // DESCRIPTION: The WOR packet counter
    // ------------------------------------------------------------------------
    unsigned char                                                m_iAmount:0X07;

} WiMinet_WakeUp;
```

4. 无线从站/无线中继参数配置命令

4.1 设置无线从站的漫游切换

指令	删除休眠节点		
字节	参数	请求指令	应答指令
1	m_iHeader	0XAA	0XAA
2	m_iOpCode	0X44	0XC4
3	m_iValueA	0X00	0X00
4	m_iValueB	0X00	0X00
5	m_iValueC	0X00	0X00
6	m_iValueD	0X00	0X00
7	m_iAmount	0X01	0X00
8	m_iCRCode.MSB	0X##	0X##
9	m_iCRCode.LSB	0X##	0X##
10	m_pBuffer[0]	Roam.Status	
备注	Roam.Status：漫游切换的状态 0X00：关闭 0X01：开启		
请求示例	AA 44 00 00 00 00 01 5A 54 00		
应答示例	AA C4 00 00 00 00 00 F4 81		

4.2 无线从站/无线中继的固定接入点

指令	无线从站/无线中继的固定接入点				
字节	参数	设置请求	读取请求	设置应答	读取应答
1	m_iHeader	0XAA	0XAA	0XAA	0XAA
2	m_iOpCode	0X0A	0X0B	0X8A	0X8B
3	m_iValueA	0X00	0X00	0X00	0X00
4	m_iValueB	0X00	0X00	0X00	0X00
5	m_iValueC	0X00	0X00	0X00	0X00
6	m_iValueD	0X00	0X00	0X00	0X00
7	m_iAmount	0X02	0X00	0X02	0X02
8	m_iCRCode.MSB	0X##	0X##	0X##	0X##
9	m_iCRCode.LSB	0X##	0X##	0X##	0X##
10	m_pBuffer[0]	Parent.MSB		Parent.MSB	Parent.MSB
11	m_pBuffer[1]	Parent.LSB		Parent.LSB	Parent.LSB
备注-1	Parent.MSB：固定接入点的 16 位网络地址高字节 Parent.LSB：固定接入点的 16 位网络地址低字节 地址 0X00 和 0XFFFA~0XFFFF 属于虚拟地址，如果填写该地址，则自动搜索；如果是物理地址 0X0001~0XFFF9 则锁定在接入点上				
设置示例	请求：AA 0A 00 00 00 00 02 6E 7D FF FF 应答：AA 8A 00 00 00 00 02 86 7A FF FF				
读取示例	请求：AA 0B 00 00 00 00 00 E4 CD 应答：AA 8B 00 00 00 00 02 43 2B FF FF				

4.3 休眠与唤醒控制

指令	休眠与唤醒控制		
字节	参数	请求指令	应答指令
1	m_iHeader	0XAA	0XAA
2	m_iOpCode	0X0E	0X8E
3	m_iValueA	0X00	0X00
4	m_iValueB	0X00	0X00
5	m_iValueC	0X00	0X00
6	m_iValueD	0X00	0X00
7	m_iAmount	0X01	0X00
8	m_iCRCode.MSB	0X##	0X##
9	m_iCRCode.LSB	0X##	0X##
10	m_pBuffer[0]	IO.Control	
备注-1	IO.Control：休眠，唤醒与复位控制状态 0X00：唤醒处理器 0X01：休眠处理器 0XFF：复位重新启动		
备注-2	该指令部分参数没有回应指令		
请求示例	AA 0E 00 00 00 00 01 39 1B 00		
应答示例	AA 8E 00 00 00 00 01 FF 7A 00		

4.4 读取当前网络的信号强度

指令	读取当前网络的信号强度		
字节	参数	请求指令	应答指令
1	m_iHeader	0XAA	0XAA
2	m_iOpCode	0X7E	0XFE
3	m_iValueA	0X00	0X00
4	m_iValueB	0X00	0X00
5	m_iValueC	0X00	0X00
6	m_iValueD	0X00	0X00
7	m_iAmount	0X00	0X04
8	m_iCRCode.MSB	0X##	0X##
9	m_iCRCode.LSB	0X##	0X##
10	m_pBuffer[0]		RSSI.digital
11	m_pBuffer[1]		LNA.Status
12	m_pBuffer[2]		RSSI.dBm
13	m_pBuffer[3]		NET.Status
备注	RSSI.digital：当前的数字化信号强度（已经包含了 LNA 或衰减器的增益） LNA.Status：LNA 或衰减器的当前配置状态 RSSI.dBm：原始的物理化信号强度（已经去掉了 LNA 或衰减器的增益） NET.Status：当前的网络连接状况： 0X00：脱机状态 0X01：联机状态		
请求示例	AA 7E 00 00 00 00 00 FF 0A		
应答示例	AA FE 00 00 00 00 04 3A 90 DD 01 E8 01		

5. 数据传输命令

5.1 发送数据

指令	发送数据		
字节	参数	请求指令	应答指令
1	m_iHeader	0XAA	0XAA
2	m_iOpCode	0X1D	0X9D
3	m_iValueA	Object.MSB	Quantum.Size
4	m_iValueB	Object.LSB	Write.Size
5	m_iValueC	Attribute	Attribute
6	m_iValueD	SerialNumber	SerialNumber
7	m_iAmount	Payload.Size	0X10
8	m_iCRCode.MSB	0X##	0X##
9	m_iCRCode.LSB	0X##	0X##
10	m_pBuffer[0]	Payload[0]	Host.Status
11	m_pBuffer[1]	Payload[1]	NET.Status
12	m_pBuffer[2]	Payload[2]	IOS.Head
13	m_pBuffer[3]	Payload[3]	IOS.Used.MSB
14	m_pBuffer[4]	Payload[4]	IOS.Used.LSB
15	m_pBuffer[5]	Payload[5]	IOS.Free.MSB
16	m_pBuffer[6]	Payload[5]	IOS.Free.LSB
17	m_pBuffer[7]	Payload[7]	IOS.Size.MSB
18	m_pBuffer[8]	Payload[8]	IOS.Size.LSB
19	m_pBuffer[9]	Payload[9]	TCP.Head
20	m_pBuffer[10]	Payload[10]	TCP.Used.MSB
21	m_pBuffer[11]	Payload[11]	TCP.Used.LSB
22	m_pBuffer[12]	Payload[12]	TCP.Free.MSB
23	m_pBuffer[13]	Payload[13]	TCP.Free.LSB
24	m_pBuffer[14]	依此类推 ……	TCP.Size.MSB
25	m_pBuffer[15]	依此类推 ……	TCP.Size.LSB

备注-1	Object.MSB：目标节点的 16 位地址的高字节 Object.LSB：目标节点的 16 位地址的低字节 如果从节点向主节点发送数据，有两种写法: 1) 直接写目标主站的物理地址，比如 0X0001，0XFF00 等真实地址 2) 写主站的虚拟地址 0X0000，改地址不随主站地址更改而变化，便于程序设计的一致性
备注-2	Attribute：报文的属性 SerialNumber：报文的序号 Payload.Size：用户数据静荷的长度 Quantum.Size：协议栈定长数据大小 Write.Size：本次实际写入的报文的长度 Payload[0~N]：用户静荷数据字节
备注-3	Host.Status：模块当前的运行状态 NET.Status：模块当前的网络连接状态
备注-4	IOS.Head：模块串口或者网口的接收缓冲区第一个字节 IOS.Used.MSB：模块串口或者网口的接收缓冲区当前占用的大小高字节 IOS.Used.LSB：模块串口或者网口的接收缓冲区当前占用的大小低字节 IOS.Free.MSB：模块串口或者网口的接收缓冲区当前剩余的大小高字节 IOS.Free.LSB：模块串口或者网口的接收缓冲区当前剩余的大小低字节 IOS.Size.MSB：模块串口或者网口的接收缓冲区当前总数的大小高字节 IOS.Size.LSB：模块串口或者网口的接收缓冲区当前总数的大小低字节 该数据块描述模块的串口或者网口的接收缓冲区的状况，其中 IOS.Size = IOS.Used + IOS.Free，**也就是说总大小 = 剩余字节 + 占用字节**
备注-5	TCP.Head：模块的发送缓冲区第一个字节 TCP.Used.MSB：模块的发送缓冲区当前占用大小高字节 TCP.Used.LSB：模块串的发送缓冲区当前占用大小低字节 TCP.Free.MSB：模块的发送缓冲区当前剩余的大小高字节 TCP.Free.LSB：模块的发送缓冲区当前剩余的大小低字节 TCP.Size.MSB：模块的发送缓冲区当前总数的大小高字节 TCP.Size.LSB：模块的发送缓冲区当前总数的大小低字节 该数据块描述模块的发送缓冲区的状况，其中 TCP.Size = TCP.Used + TCP.Free，**也就是说总大小 = 剩余字节 + 占用字节**
备注-6	Attribute **的属性定义：**

	1) 0X00 是以 TCP 形式发送数据报文，报文主体，代表报文的中间部分，不是头尾 2) 0X01 是以 TCP 形式发送数据报文，报文头部，代表报文的第一个片段 3) 0X02 是以 TCP 形式发送数据报文，报文尾部，代表报文的最后一个片段 4) 0X03 是以 TCP 形式发送数据报文，单个报文，最大写入长度 245 字节 5) 0X10 是 2.4Kbps 的语音编码报文，TCP 连接，UDP 传送 6) 0X11 是 4.8Kbps 的语音编码报文，TCP 连接，UDP 传送 7) 0X12 是 9.6Kbps 的语音编码报文，TCP 连接，UDP 传送 8) 0X13 是远程语音的播放请求命令，TCP 连接，UDP 传送 9) 0X14 是以 UDP 形式发送，要求接收端会送 ACK 确认 10) 0X15 是以 UDP 形式发送，不要求接收端送 ACK 确认 **11) 0X23 是 TCP 形式发送的远程指令，将已经构造好的一个指令的完整报文（包括报文头部 m_iHeader，m_iOpCode，m_iValueA，m_iValueB，m_iValueC，m_iValueD，m_iAmount，m_iCRCode 和报文扩展部分 m_pBuffer）作为本报文的内容（m_pBuffer）发送到目标节点，目标节点会将被封装的报文提取出来并进行执行，将执行的结果返回给发送节点，应答报文属性是 0X27。该指令通常用作远程配置和远程诊断等用途。** 12) 报文属性的 Bit7：开启/关闭本地回显（是否需要以 0X9D 回应），Bit7=1 是关闭本地回显，Bit7=0 开启本地回显 报文属性的 Bit6：要求或者放弃异步通知，Bit6=1 是要求异步通知说明发送成功或者失败，Bit6=0 是放弃异步通知说明发送成功或者失败
备注-7	发送定长的小型数据块，建议采用 UDP（0X14）发送，如需关闭回显可采用 0X94
备注-8	发送少量的较大数据块，建议采用 TCP（0X03）传送，如需关闭回显可采用 0X83
备注-9	发送大型数据块，比如一张 BMP 图片，建议采用多包组合的 TCP 传送，不关闭回显，首包 （0X01）+ 多个扩展包 （0X00） + 尾包（0X02）
备注-9	SerialNumber：报文序列号码 (1) Bit0-Bit6：报文的序号，取值范围 0X00-0X7F。 (2) Bit7：该报文的重复发送指示 Bit7=1：需要接收端比对报文序号 Bit0-Bit7，序号相同则放弃本报文，序号不一致则正常接收 Bit7=0：无条件接收该报文 **注意：SerialNumber 在发送数量不大的报文的时候，可以设置 SerialNumber 为 0X00，也就是无条件接收；在发送通过连续多包的形式发送较大的数据块的时候，才有必要启用 SerialNumber 的报文需要比对功能**
请求示例	AA 1D 7E 50 03 00 0A 45 90 30 31 32 33 34 35 36 37 38 39
应答示例	AA 9D 10 0A 03 00 10 49 21 50 30 11 00 10 3A F0 3B 00 00 00 00 01 00 01 00

5.2 接收数据

指令	接收数据	
字节	参数	输出指令
1	m_iHeader	0XAA
2	m_iOpCode	0X1E
3	m_iValueA	Source.MSB
4	m_iValueB	Source.LSB
5	m_iValueC	Attribute
6	m_iValueD	0X00
7	m_iAmount	Payload.Size
8	m_iCRCode.MSB	0X##
9	m_iCRCode.LSB	0X##
10	m_pBuffer[0]	Payload[0]
11	m_pBuffer[1]	Payload[1]
12	m_pBuffer[2]	依此类推 ……
13	m_pBuffer[3]	依此类推 ……
备注-1	无线模块主动输出，用户不需要做出应答	
备注-2	Source.MSB：原始发送节点的 16 位网络地址高字节 Source.LSB：原始发送节点的 16 位网络地址低字节	
备注-3	Attribute：**报文属性** 如果对端以 0X03 指令的 TCP 形式发送较大的数据块，且无线模块没有做拼包输出的话，用户接收端需要自己做拼包处理，具体如下： 0X01：第一包，代表这是第一包，后面还有会尾包，可能会有中间包 0X00：中间包，中间包可能会有多个，也可能没有 0X02：尾包，如果有第一包，一定会有尾包 0X03：仅此一包，也就是首包 + 尾包（0X03=0X01 + N * 0X00 + 0X02），N 为自然数	
备注-4	Payload.Size：报文长度	
备注-5	Payload[0-N]：报文的静荷部分	
示例报文	AA 1E FD D7 14 00 0A 85 85 30 31 32 33 34 35 36 37 38 39	

5.3 查询发送缓冲区

指令	查询发送缓冲区		
字节	参数	请求指令	应答指令
1	m_iHeader	0XAA	0XAA
2	m_iOpCode	0X1C	0X9C
3	m_iValueA	0X00	Quantum.Size
4	m_iValueB	0X00	0X00
5	m_iValueC	0X00	0X00
6	m_iValueD	0X00	0X00
7	m_iAmount	0X00	0X10
8	m_iCRCode.MSB	0X##	0X##
9	m_iCRCode.LSB	0X##	0X##
10-25	m_pBuffer[0-15]		定义同 0X9D 指令 详见“**主机的工作状态信息**”
备注-1	Quantum.Size：协议栈定长数据大小		
备注-2	**请求指令参数** m_iValueB：未定义，可以取任意数值，默认填写 0X00 m_iValueC：未定义，可以取任意数值，默认填写 0X00 m_iValueD： (1) Bit0~Bit6：报文序号，可以取任意数值，无线模块会同步此序号 (2) Bit7：未定义，默认填写 0X00		
备注-3	**应答指令参数** m_iValueB：同请求指令，未做修改 m_iValueC：同请求指令，未做修改 m_iValueD：同请求指令，未做修改		
请求示例	AA 1C 00 00 00 FF 00 1A BD		
应答示例	AA 9C 10 00 00 FF 10 5A B4 50 30 00 00 00 3B 00 3B 00 00 00 00 01 00 01 00		

5.4 主机的工作状态信息

主机工作状态信息：

(1) Byte00：当前的主机工作状态

(2) Byte01：当前的网络注册状态

(3) Byte02-Byte08（共计 7 个字节）：当前的本地接收缓冲区的信息

Byte02：缓冲区中的第一个字节

Byte03：当前已经占用的缓冲区大小的高字节（MSB）

Byte04：当前已经占用的缓冲区大小的低字节（LSB）

Byte05：当前剩余未用的缓冲区大小的高字节（MSB）

Byte06：当前剩余未用的缓冲区大小的低字节（LSB）

Byte07：总共的缓冲区大小的高字节（MSB）

Byte08：总共的缓冲区大小的低字节（LSB）

(4) Byte09-Byte15（共计 7 个字节）：当前的网络发送缓冲区的信息

Byte09：缓冲区中的第一个字节

Byte10：当前已经占用的缓冲区大小的高字节（MSB）

Byte11：当前已经占用的缓冲区大小的低字节（LSB）

Byte12：当前剩余未用的缓冲区大小的高字节（MSB）

Byte13：当前剩余未用的缓冲区大小的低字节（LSB）

Byte14：总共的缓冲区大小的高字节（MSB）

Byte15：总共的缓冲区大小的低字节（LSB）

说明：发送大型数据包的流程步骤如下：

(1) 通过 0X1C 指令查询发送的缓冲区长度，检查“主机的工作状态信息”的 Byte05-Byte06(**红色字体标注**)，这是实际的可以写入的最大长度，超过了该长度，就会造成缓冲区溢出。如果该长度为零，则等待一段时间后(建议 100ms)再次查询，否则执行步骤(2)

(2) 通过 0X1D 指令写入数据，根据应答指令中的返回信息，检查“主机的工作状态信息” 的 Byte05-Byte06，获得下一次可以写入的最大长度，如果该长度为零，则跳转到步骤(1)

(3) 如果还有数据需要写入，执行步骤（2）或者(1)，直到全部数据写入完成。

5.5 全局报文描述符

发送大型数据块，比如 30K+字节的时候，为了方便接收方显示接收进度并自动进行 CRC 校验，发送端通常会发送一个全局报文描述符的报文，报文属性 0X3F。

该报文的数据结构定义如下：

（1） 数据块的大小，32 位长度

Byte00：数据长度 24-31 位

Byte01：数据长度 16-23 位

Byte02：数据长度 8-15 位

Byte03：数据长度 0-7 位

（2） 数据块的 CRC 校验

Byte04：CRC32 数据的 24-31 位

Byte05: CRC32 数据的 16-23 位

Byte06：CRC32 数据的 8-15 位

Byte07：CRC32 数据的 0-7 位

（3） 业务的流水序列号码

Byte08：业务流水序列号码的 24-31 位

Byte09: 业务流水序列号码的 16-23 位

Byte10：业务流水序列号码的 8-15 位

Byte11：业务流水序列号码的 0-7 位

计算 CRC32 的函数和算法参考附录中的 CRC32 的实现方式。

6. 全局网络拓扑结构

6.1 查询特定主机的全局注册路径

指令	查询特定主机的全局注册路径		
字节	参数	请求指令	应答指令
1	m_iHeader	0XAA	0XAA
2	m_iOpCode	0X7D	0XFD
3	m_iValueA	0X00	0X00
4	m_iValueB	0X00	0X00
5	m_iValueC	0X00	0X00
6	m_iValueD	0X00	0X00
7	m_iAmount	0X02	0X02 * N
8	m_iCRCode.MSB	0X##	0X##
9	m_iCRCode.LSB	0X##	0X##
10	m_pBuffer[0]	Objet.MSB	第 1 个节点的地址高字节
11	m_pBuffer[1]	Objet.LSB	第 1 个节点的地址低字节
12	m_pBuffer[2]		第 2 个节点的地址高字节
13	m_pBuffer[3]		第 2 个节点的地址低字节
14	m_pBuffer[4]		依此类推 ……
15	m_pBuffer[5]		依此类推 ……
备注-1	Objet.MSB：被查询主机的 16 位网络地址高字节 Objet.LSB：被查询主机的 16 位网络地址低字节 N：节点的个数		
请求示例	AA 7D FD D7 00 00 00 02 74		
应答示例	AA FD FD D7 00 00 02 A7 B5 93 A8		

6.2 查询特定主机的硬件 MAC 地址

指令	查询特定主机的硬件 MAC 地址		
字节	参数	请求指令	应答指令
1	m_iHeader	0XAA	0XAA
2	m_iOpCode	0X7A	0XFA
3	m_iValueA	0X00	0X00
4	m_iValueB	0X00	0X00
5	m_iValueC	0X00	0X00
6	m_iValueD	0X00	0X00
7	m_iAmount	0X02	0X08
8	m_iCRCode.MSB	0X##	0X##
9	m_iCRCode.LSB	0X##	0X##
10	m_pBuffer[0]	Objet.MSB	MAC_ID[0]
11	m_pBuffer[1]	Objet.LSB	MAC_ID[1]
12	m_pBuffer[2]		MAC_ID[2]
13	m_pBuffer[3]		MAC_ID[3]
14	m_pBuffer[4]		MAC_ID[4]
15	m_pBuffer[5]		MAC_ID[5]
16	m_pBuffer[6]		MAC_ID[6]
17	m_pBuffer[7]		MAC_ID[7]
备注-1	Objet.MSB：被查询主机的 16 位网络地址高字节 Objet.LSB：被查询主机的 16 位网络地址低字节 MAC_ID[0~7]：被查询主机的硬件 64 位硬件 MAC 地址		
请求示例	AA 7A 00 00 00 00 02 3B 7E FD D7		
应答示例	AA FA 00 00 00 00 08 85 51 00 0B 57 00 00 11 BC 30		

6.3 查询最大的注册客户端的数量

指令	查询最大的注册客户端的数量		
字节	参数	请求指令	应答指令
1	m_iHeader	0XAA	0XAA
2	m_iOpCode	0X4A	0XCA
3	m_iValueA	0X00	0X00
4	m_iValueB	0X00	0X00
5	m_iValueC	0X00	0X00
6	m_iValueD	0X00	0X00
7	m_iAmount	0X00	0X02
8	m_iCRCode.MSB	0X##	0X##
9	m_iCRCode.LSB	0X##	0X##
10	m_pBuffer[0]		Count.MSB
11	m_pBuffer[1]		Count.LSB
备注-1	Count.MSB：最大的注册客户端的数量高字节 Count.LSB：最大的注册客户端的数量低字节		
请求示例	AA 4A 00 00 00 FF 00 0C 38		
应答示例	AA CA 00 00 00 FF 02 4D 6C 00 80		

6.4 查询注册节点的运行状态信息

指令	查询注册节点的运行状态信息		
字节	参数	请求指令	应答指令
1	m_iHeader	0XAA	0XAA
2	m_iOpCode	0X49	0XC9
3	m_iValueA	GetIndex.MSB	ReadIndex.MSB
4	m_iValueB	GetIndex.MSB	ReadIndex.MSB
5	m_iValueC	0X00	0X00
6	m_iValueD	0X00	0X00
7	m_iAmount	0X00	0X1C
8	m_iCRCode.MSB	0X##	0X##
9	m_iCRCode.LSB	0X##	0X##
10~37	m_pBuffer[0~27]		参考 “WiMinet_MeshItem”定义
备注-1	GetIndex.MSB：需要读取记录条目的索引序号（从 0 开始）的高字节 GetIndex.LSB：需要读取记录条目的索引序号（从 0 开始）的低字节		
备注-2	ReadIndex.MSB：实际读取记录条目的索引序号（从 0 开始）的高字节 ReadIndex.LSB：实际读取记录条目的索引序号（从 0 开始）的低字节		
请求示例	AA 49 00 00 00 FF 00 19 78		
应答示例	AA C9 00 00 00 FF 1C B1 8A 01 10 00 0B 57 00 00 13 A0 27 3E F3 01 00 01 00 3C B7 00 00 37 38 01 87 F5 80 F2 80		

是节点的注册信息 WiMinet_MeshItem 的定义说明

1. Byte10(m_pBuffer[0])：节点的唤醒计数器
2. Byte11(m_pBuffer[1])：节点的射频芯片和频带配置
3. Byte12(m_pBuffer[2])：节点的 64 位 MAC 地址字节 0
4. Byte13(m_pBuffer[3])：节点的 64 位 MAC 地址字节 1
5. Byte14(m_pBuffer[4])：节点的 64 位 MAC 地址字节 2
6. Byte15(m_pBuffer[5])：节点的 64 位 MAC 地址字节 3
7. Byte16(m_pBuffer[6])：节点的 64 位 MAC 地址字节 4
8. Byte17(m_pBuffer[7])：节点的 64 位 MAC 地址字节 5

9. Byte18(m_pBuffer[8])：节点的 64 位 MAC 地址字节 6
10. Byte19(m_pBuffer[9])：节点的 64 位 MAC 地址字节 7
11. Byte20(m_pBuffer[10])：节点的 16 位网络地址低字节
12. Byte21(m_pBuffer[11])：节点的 16 位网络地址高字节
13. Byte22(m_pBuffer[12])：节点的中继等级
14. Byte23(m_pBuffer[13])：节点的设备类型（0X00=从站，0X01=主站，0X02=中继）
15. Byte24(m_pBuffer[14])：第 1 组射频的配置信息
16. Byte25(m_pBuffer[15])：第 2 组射频的配置信息
17. Byte26(m_pBuffer[16])：父节点的网络地址低字节
18. Byte27(m_pBuffer[17])：父节点的网络地址高字节
19. Byte28(m_pBuffer[18])：工作信道
20. Byte29(m_pBuffer[19])：网络接入时槽
21. Byte30(m_pBuffer[20])：发射链路衰减
22. Byte31(m_pBuffer[21])：接收链路衰减
23. Byte32(m_pBuffer[22])：接收放大器配置
24. Byte33(m_pBuffer[23])：发射放大器配置
25. Byte34(m_pBuffer[24])：发射信号强度
26. Byte35(m_pBuffer[25])：发射信号质量
27. Byte36(m_pBuffer[26])：接收信号强度
28. Byte37(m_pBuffer[27])：接收信号质量

结构体 WiMinet_MeshItem 定义如下：

```
// ------------------------------------------------------------------------
// DESCRIPTION:
// ------------------------------------------------------------------------
typedef struct _WiMinet_Address_
{
   // ---------------------------------------------------------------------
   // DESCRIPTION: 节点的模组设置
   // ---------------------------------------------------------------------
   unsigned char                          m_iX2Chip;

   // ---------------------------------------------------------------------
   // DESCRIPTION: 节点的频段设置
   // ---------------------------------------------------------------------
   unsigned char                          m_iX2Band;

   // ---------------------------------------------------------------------
   // DESCRIPTION: 节点的MAC地址
   // ---------------------------------------------------------------------
   unsigned char                     m_pX64MAC[0X08];

   // ---------------------------------------------------------------------
```

```
    // DESCRIPTION: 节点的16位地址
    // ---------------------------------------------------------------------------
    unsigned short                              m_iTxAddr;

    // ---------------------------------------------------------------------------
    // DESCRIPTION: 节点的TreeID
    // ---------------------------------------------------------------------------
    unsigned char                               m_iTxTree;

    // ---------------------------------------------------------------------------
    // DESCRIPTION: 节点的设备类型
    // ---------------------------------------------------------------------------
    unsigned char                              m_iTxMode;

} WiMinet_Address;


// ---------------------------------------------------------------------------
// DESCRIPTION:
// ---------------------------------------------------------------------------
typedef struct _RFChip_RunTime_
{
    // ---------------------------------------------------------------------------
    // DESCRIPTION:
    // ---------------------------------------------------------------------------
    unsigned char                           m_iActive:0X01;

    // ---------------------------------------------------------------------------
    // DESCRIPTION:
    // ---------------------------------------------------------------------------
    unsigned char                         m_iWiMode:0X02;

    // ---------------------------------------------------------------------------
    // DESCRIPTION:
    // ---------------------------------------------------------------------------
    unsigned char                         m_iRFBand:0X01;

    // ---------------------------------------------------------------------------
    // DESCRIPTION:
    // ---------------------------------------------------------------------------
    unsigned char                         m_iUnused:0X04;

} RFChip_RunTime;

// ---------------------------------------------------------------------------
// DESCRIPTION:
// ---------------------------------------------------------------------------
typedef struct _WiMinet_RunTime_
{
    // ---------------------------------------------------------------------------
    // DESCRIPTION:
    // ---------------------------------------------------------------------------
    RFChip_RunTime                          m_iChipX1;

    // ---------------------------------------------------------------------------
    // DESCRIPTION:
    // ---------------------------------------------------------------------------
    RFChip_RunTime                          m_iChipX2;

    // ---------------------------------------------------------------------------
    // DESCRIPTION: 父节点地址
    // ---------------------------------------------------------------------------
    unsigned short                              m_iParent;

    // ---------------------------------------------------------------------------
    // DESCRIPTION: 通讯的信道
    // ---------------------------------------------------------------------------
    unsigned char                              m_iChanX1;
```

```
    // ------------------------------------------------------------------------
    // DESCRIPTION: 工作时隙
    // ------------------------------------------------------------------------
    unsigned char                                m_iX8Node;

    // ------------------------------------------------------------------------
    // DESCRIPTION: 发送链路衰减
    // ------------------------------------------------------------------------
    unsigned char                                 m_iTxLoss;

    // ------------------------------------------------------------------------
    // DESCRIPTION: 接收链路衰减
    // ------------------------------------------------------------------------
    unsigned char                                m_iRxLoss;

    // ------------------------------------------------------------------------
    // DESCRIPTION: 当前的 LNA 配置状态
    // ------------------------------------------------------------------------
    unsigned char                                m_iLNASet;

    // ------------------------------------------------------------------------
    // DESCRIPTION: 当前的 功放的配置状态
    // ------------------------------------------------------------------------
    unsigned char                                m_iTPASet;

    // ------------------------------------------------------------------------
    // DESCRIPTION: 本节点发送给父节点的信号强度 (RSSI)
    // ------------------------------------------------------------------------
    unsigned char                                m_iTxRSSI;

    // ------------------------------------------------------------------------
    // DESCRIPTION: 本节点发送给父节点的信号质量 (SLQI)
    // ------------------------------------------------------------------------
    unsigned char                                m_iTxSLQI;

    // ------------------------------------------------------------------------
    // DESCRIPTION: 本节点接收到的父节点的信号强度 (RSSI)
    // ------------------------------------------------------------------------
    unsigned char                                 m_iRxRSSI;

    // ------------------------------------------------------------------------
    // DESCRIPTION: 本节点接收到的父节点的信号质量 (SLQI)
    // ------------------------------------------------------------------------
    unsigned char                                m_iRxSLQI;

} WiMinet_RunTime;


// ----------------------------------------------------------------------------
// DESCRIPTION:
// ----------------------------------------------------------------------------
typedef struct _WiMinet_MeshItem
{
    // ------------------------------------------------------------------------
    // DESCRIPTION: 节点的信息
    // ------------------------------------------------------------------------
    WiMinet_Address                           m_Address;

    // ------------------------------------------------------------------------
    // DESCRIPTION: 节点的运行状态参数
    // ------------------------------------------------------------------------
    WiMinet_RunTime                         m_RunTime;

} WiMinet_MeshItem;
```

7. 局域拓扑关系发现

7.1 读取本地局域拓扑的节点信息

指令	读取本地局域拓扑的节点信息		
字节	参数	请求指令	应答指令
1	m_iHeader	0XAA	0XAA
2	m_iOpCode	0X14	0X94
3	m_iValueA	Get.Index	Read.Index
4	m_iValueB	Get.Count	Read.Count
5	m_iValueC	0X00	Total.Count
6	m_iValueD	0X00	0X00
7	m_iAmount	0X00	0X15 * Read.Count
8	m_iCRCode.MSB	0X##	0X##
9	m_iCRCode.LSB	0X##	0X##
10~30	m_pBuffer[0~20]		参考 “WiMinet_AP_Profile”定义
31~51	m_pBuffer[21~N]		依此类推 ……
备注-1	Get.Index：需要读取的记录序号（从 0X00 开始第一条） Get.Index：需要读取的记录条数		
备注-2	Read.Index：实际读取的记录序号（从 0X00 开始第一条） Read.Index：实际读取的记录条数 Total.Count：本地记录的总共条数		
请求示例	AA 14 00 FF 00 00 00 A7 98		
应答示例	AA 94 00 01 01 00 15 6B 4F 00 93 A8 00 E4 80 17 14 08 13 87 F1 00 09 02 50 FF EC 06 7A FF		

其中单个节点的信息的结构体 WiMinet_AP_Profile 定义如下：

```
// ---------------------------------------------------------------------------
// DESCRIPTION:
// ---------------------------------------------------------------------------
typedef struct _RFChip_MTRadio_
{
   // ---------------------------------------------------------------------------
   // DESCRIPTION:
```

```
    // ----------------------------------------------------------------------------
    unsigned char                          m_iAmount:0X04;

    // ----------------------------------------------------------------------------
    // DESCRIPTION:
    // ----------------------------------------------------------------------------
    unsigned char                          m_iRxBand:0X04;

} RFChip_MTRadio;

// ----------------------------------------------------------------------------
// DESCRIPTION:
// ----------------------------------------------------------------------------
typedef struct _WiMinet_AP_Profile_
{
    // ----------------------------------------------------------------------------
    // DESCRIPTION:
    // ----------------------------------------------------------------------------
    unsigned char                              m_iChanX1;

    // ----------------------------------------------------------------------------
    // DESCRIPTION:
    // ----------------------------------------------------------------------------
    unsigned short                             m_iTxAddr;

    // ----------------------------------------------------------------------------
    // DESCRIPTION:
    // ----------------------------------------------------------------------------
    unsigned char                               m_iTxTree;

    // ----------------------------------------------------------------------------
    // DESCRIPTION:
    // ----------------------------------------------------------------------------
    unsigned char                              m_iRxRSSI;

    // ----------------------------------------------------------------------------
    // DESCRIPTION:
    // ----------------------------------------------------------------------------
    unsigned char                              m_iRxSLQI;

    // ----------------------------------------------------------------------------
    // DESCRIPTION:
    // ----------------------------------------------------------------------------
    unsigned char                               m_iTxGain;

    // ----------------------------------------------------------------------------
    // DESCRIPTION:
    // ----------------------------------------------------------------------------
    unsigned char                              m_iRxGain;

    // ----------------------------------------------------------------------------
    // DESCRIPTION:
    // ----------------------------------------------------------------------------
    unsigned char                           m_iRFChip:0X01;

    // ----------------------------------------------------------------------------
    // DESCRIPTION:
    // ----------------------------------------------------------------------------
    unsigned char                          m_iRFBand:0X01;

    // ----------------------------------------------------------------------------
    // DESCRIPTION:
    // ----------------------------------------------------------------------------
    unsigned char                          m_iX2Chip:0X01;

    // ----------------------------------------------------------------------------
    // DESCRIPTION:
    // ----------------------------------------------------------------------------
```

```
unsigned char                        m_iX2Band:0X01;

// ------------------------------------------------------------------------
// DESCRIPTION:
// ------------------------------------------------------------------------
unsigned char                        m_iAuthen:0X01;

// ------------------------------------------------------------------------
// DESCRIPTION:
// ------------------------------------------------------------------------
unsigned char                         m_iOnPair:0X01;

// ------------------------------------------------------------------------
// DESCRIPTION:
// ------------------------------------------------------------------------
unsigned char                        m_iMember:0X01;

// ------------------------------------------------------------------------
// DESCRIPTION:
// ------------------------------------------------------------------------
unsigned char                        m_iUnUsed:0X01;

// ------------------------------------------------------------------------
// DESCRIPTION:
// ------------------------------------------------------------------------
unsigned short                            m_iX16PAN;

// ------------------------------------------------------------------------
// DESCRIPTION:
// ------------------------------------------------------------------------
unsigned char                        m_iLNASet:0X04;

// ------------------------------------------------------------------------
// DESCRIPTION:
// ------------------------------------------------------------------------
unsigned char                        m_iTPASet:0X04;

// ------------------------------------------------------------------------
// DESCRIPTION:
// ------------------------------------------------------------------------
RFChip_MTRadio                        m_MTRadio;

// ------------------------------------------------------------------------
// DESCRIPTION:
// ------------------------------------------------------------------------
unsigned char                              m_iStatus;

// ------------------------------------------------------------------------
// DESCRIPTION: Time slot=2 ^ m_iTSlice
// ------------------------------------------------------------------------
unsigned char                             m_iTSlice;

// ------------------------------------------------------------------------
// DESCRIPTION: BIT0
// ------------------------------------------------------------------------
unsigned char                         m_iBBCall:0X01;

// ------------------------------------------------------------------------
// DESCRIPTION: BIT1-BIT2
// ------------------------------------------------------------------------
unsigned char                        m_iTSMode:0X02;

// ------------------------------------------------------------------------
// DESCRIPTION: BIT3-BIT5
// ------------------------------------------------------------------------
unsigned char                       m_iHTMode:0X03;

// ------------------------------------------------------------------------
```

```
    // DESCRIPTION: BIT6
    // ---------------------------------------------------------------------------
    unsigned char                        m_iHTWork:0X01;

    // ---------------------------------------------------------------------------
    // DESCRIPTION: BIT7
    // ---------------------------------------------------------------------------
    unsigned char                         m_iHTBoot:0X01;

    // ---------------------------------------------------------------------------
    // DESCRIPTION:
    // ---------------------------------------------------------------------------
    signed short                                m_idBmVal;

    // ---------------------------------------------------------------------------
    // DESCRIPTION: The packet amount
    // ---------------------------------------------------------------------------
    unsigned char                               m_iAmount;

    // ---------------------------------------------------------------------------
    // DESCRIPTION: The sum value of the RSSI in dBm
    // ---------------------------------------------------------------------------
    signed short                                m_idBmSum;

} WiMinet_AP_Profile;
```

8. 本地网络接入控制

8.1 设置最大和最小注册路由等级

指令	设置最大和最小注册路由等级				
字节	参数	设置请求	读取请求	设置应答	读取应答
1	m_iHeader	0XAA	0XAA	0XAA	0XAA
2	m_iOpCode	0X4E	0X4F	0XCE	0XCF
3	m_iValueA	0X00	0X00	0X00	0X00
4	m_iValueB	0X00	0X00	0X00	0X00
5	m_iValueC	Router.Grade	0X00	Router.Grade	Router.Grade
6	m_iValueD	0X00	0X00	0X00	0X00
7	m_iAmount	0X02	0X00	0X02	0X02
8	m_iCRCode.MSB	0X##	0X##	0X##	0X##
9	m_iCRCode.LSB	0X##	0X##	0X##	0X##
10	m_pBuffer[0]	Router.Min		Router.Min	Router.Min
11	m_pBuffer[1]	Router.Max		Router.Max	Router.Max
参数-1	Router.Grade： 0X01：最小值 0X10：最大值 0X11：最大值和最小值				
参数-2	Router.Min：最小的路由等级，取值 0-63，0XFF 代表自动选择 Router.Max：最大的路由等级，取值 0-63，0XFF 代表自动选择				
设置示例	请求：AA 4E 00 00 10 00 02 0E 6E 00 3F 应答：AA CE 00 00 10 00 02 E6 69 00 3F				
读取示例	请求：AA 4F 00 00 00 00 00 27 C8 应答：AA CF 00 00 00 00 02 F2 69 00 00				

9. 本地 I/O 状态提示

9.1 设置本地 I/O 事件与状态提示

指令	设置本地 I/O 事件与状态提示				
字节	参数	设置请求	读取请求	设置应答	读取应答
1	m_iHeader	0XAA	0XAA	0XAA	0XAA
2	m_iOpCode	0X1F	0X30	0X9F	0XB0
3	m_iValueA	0X00	0X00	0X00	0X00
4	m_iValueB	0X00	0X00	0X00	0X00
5	m_iValueC	0X00	0X00	0X00	0X00
6	m_iValueD	0X00	0X00	0X00	0X00
7	m_iAmount	0X01	0X00	0X01	0X01
8	m_iCRCode.MSB	0X##	0X##	0X##	0X##
9	m_iCRCode.LSB	0X##	0X##	0X##	0X##
10	m_pBuffer[0]	IO.Status		IO.Status	IO.Status
参数-1	IO.Status：本地 I/O 状态提示的执行动作 （1）Bit0：蜂鸣器输出 0X01：使能 0X00：禁止 （2）Bit1：文本格式消息输出 0X01：使能 0X00：禁止 （3）Bit2-6：未定义 （4）Bit7：二进制格式消息输出 0X01：使能 0X00：禁止 注意：不同的状态可以通过位操作以提高效率，比如 0X83 = (0X80 | 0X01 | 0X02)				
设置示例	请求：AA 1F 00 00 00 00 01 A9 1A 01 应答：AA 9F 00 00 00 00 01 6F 7B 01				
读取示例	请求：AA 30 00 00 00 00 00 43 8F 应答：AA B0 00 00 00 00 01 5E A3 01				

10. 事件、通知与消息

指令	事件、通知与消息	
字节	参数	输出指令
1	m_iHeader	0XAA
2	m_iOpCode	0X2E
3	m_iValueA	Source.MSB
4	m_iValueB	Source.LSB
5	m_iValueC	EventID
6	m_iValueD	0X00
7	m_iAmount	Payload.Size
8	m_iCRCode.MSB	0X##
9	m_iCRCode.LSB	0X##
10	m_pBuffer[0]	Payload[0]
11	m_pBuffer[1]	Payload[1]
12	m_pBuffer[2]	依此类推 ……
13	m_pBuffer[3]	依此类推 ……
备注-1	无线模块主动输出，用户不需要做出应答	
备注-2	Source.MSB：原始发送节点的 16 位网络地址高字节 Source.LSB：原始发送节点的 16 位网络地址低字节	
备注-3	EventID：事件代码 0X80：注册对码关系 参数长度：8 字节 参数内容：从站节点的 8 字节的 MAC 地址 0X81：删除对码关系 参数长度：8 字节 参数内容：从站节点的 8 字节的 MAC 地址 0X82：主节点收到从节点的本地注册请求 参数长度：1 字节 参数内容：从站节点的设备属性 0X83：主节点收到从节点的本地注销请求	

	参数长度：1 字节 参数内容：从站节点的设备属性 0X84：从节点成功的注册到主节点 参数长度：1 字节 参数内容：从站节点的设备属性 0X85：从节点从本地网络中注销掉 参数长度：1 字节 参数内容：从站节点的设备属性 0X86：基站或者中继收到了干扰信号-1，有可能是带外干扰，有可能是带内干扰 参数长度：1 字节 参数内容：干扰信号的信号强度 0X87：基站或者中继收到了干扰信号-2，确认频带和信道完全重叠，属于带内干扰 参数长度：1 字节 参数内容：干扰信号的信道号码 0X88：基站或者中继收到了干扰信号-3，主站而不是中继设备收到了干扰 参数长度：1 字节 参数内容：干扰信号的信道号码 0X89：基站或者中继收到了干扰信号-4，主站开启了干扰躲避，需要执行跳频操作 参数长度：1 字节 参数内容：干扰信号的信道号码 0X8A：TCP 链路呼叫连接失败 参数长度：1 字节 参数内容：报文的编号 0X8B：TCP 链路建立成功，但是数据传输失败 参数长度：1 字节 参数内容：报文的 TaskID 号码 0X8C：TCP 传送完成报告，只有在打开了传送报告的时候才会有输出 参数长度：1 字节 参数内容：报文的 TaskID 号码 0X8D：主站搜索到合适的工作信道，锁定在该信道工作 参数长度：1 字节 参数内容：新的工作信道号码

	0X8E：非法的网络地址 参数长度：0 字节 参数内容：无 0X8F：非法的 MAC 地址 参数长度：0 字节 参数内容：无 0X90：License 注册号码无效 参数长度：0 字节 参数内容：无 0X91：从站节点注册到根基站 参数长度：8 字节 参数内容：从站节点的 8 字节的 MAC 地址 0X92：本地节点的心跳活动信号 参数长度：0 字节 参数内容：无 0X93：节点因为 License 注册号码失效，开机 30 秒之后停止运行 参数长度：0 字节 参数内容：无 0X94：中继节点注册到根基站 参数长度：8 字节 参数内容：中继节点的 8 字节的 MAC 地址
备注-4	Payload.Size：事件的扩展报文长度，不同的事件扩展部分长度不一样
备注-5	Payload[0-N]：报文的静荷部分
示例报文	AA 2E 34 F1 82 00 01 FB BE 1D

11. 系统管理

11.1 系统软件复位

指令	系统软件复位		
字节	参数	请求指令	应答指令
1	m_iHeader	0XAA	0XAA
2	m_iOpCode	0X03	0X83
3	m_iValueA	0X00	0X00
4	m_iValueB	0X00	0X00
5	m_iValueC	Mode	Mode
6	m_iValueD	0X00	0X00
7	m_iAmount	0X00	0X00
8	m_iCRCode.MSB	0X##	0X##
9	m_iCRCode.LSB	0X##	0X##
备注-1	Mode：复位处理器模式 0XFF：冷复位处理器 0X00-0XFE：仅复位协议栈，处理器不复位		
备注-2	冷复位模式没有回应指令		
请求示例	AA 03 00 00 00 00 00 42 CC		
应答示例	AA 83 00 00 00 00 00 22 C4		

12. 附录-1：CRC16 算法

CRC-16 查表算法

```
static const unsigned char code HiCRCTable[] = {
0X00, 0XC1, 0X81, 0X40, 0X01, 0XC0, 0X80, 0X41, 0X01, 0XC0, 0X80, 0X41, 0X00, 0XC1, 0X81, 0X40,
0X01, 0XC0, 0X80, 0X41, 0X00, 0XC1, 0X81, 0X40, 0X00, 0XC1, 0X81, 0X40, 0X01, 0XC0, 0X80, 0X41,
0X01, 0XC0, 0X80, 0X41, 0X00, 0XC1, 0X81, 0X40, 0X00, 0XC1, 0X81, 0X40, 0X01, 0XC0, 0X80, 0X41,
0X00, 0XC1, 0X81, 0X40, 0X01, 0XC0, 0X80, 0X41, 0X01, 0XC0, 0X80, 0X41, 0X00, 0XC1, 0X81, 0X40,
0X01, 0XC0, 0X80, 0X41, 0X00, 0XC1, 0X81, 0X40, 0X00, 0XC1, 0X81, 0X40, 0X01, 0XC0, 0X80, 0X41,
0X00, 0XC1, 0X81, 0X40, 0X01, 0XC0, 0X80, 0X41, 0X01, 0XC0, 0X80, 0X41, 0X00, 0XC1, 0X81, 0X40,
0X00, 0XC1, 0X81, 0X40, 0X01, 0XC0, 0X80, 0X41, 0X01, 0XC0, 0X80, 0X41, 0X00, 0XC1, 0X81, 0X40,
```

```
0X01, 0XC0, 0X80, 0X41, 0X00, 0XC1, 0X81, 0X40, 0X00, 0XC1, 0X81, 0X40, 0X01, 0XC0, 0X80, 0X41,
0X01, 0XC0, 0X80, 0X41, 0X00, 0XC1, 0X81, 0X40, 0X00, 0XC1, 0X81, 0X40, 0X01, 0XC0, 0X80, 0X41,
0X00, 0XC1, 0X81, 0X40, 0X01, 0XC0, 0X80, 0X41, 0X01, 0XC0, 0X80, 0X41, 0X00, 0XC1, 0X81, 0X40,
0X00, 0XC1, 0X81, 0X40, 0X01, 0XC0, 0X80, 0X41, 0X01, 0XC0, 0X80, 0X41, 0X00, 0XC1, 0X81, 0X40,
0X01, 0XC0, 0X80, 0X41, 0X00, 0XC1, 0X81, 0X40, 0X00, 0XC1, 0X81, 0X40, 0X01, 0XC0, 0X80, 0X41,
0X00, 0XC1, 0X81, 0X40, 0X01, 0XC0, 0X80, 0X41, 0X01, 0XC0, 0X80, 0X41, 0X00, 0XC1, 0X81, 0X40,
0X01, 0XC0, 0X80, 0X41, 0X00, 0XC1, 0X81, 0X40, 0X00, 0XC1, 0X81, 0X40, 0X01, 0XC0, 0X80, 0X41,
0X01, 0XC0, 0X80, 0X41, 0X00, 0XC1, 0X81, 0X40, 0X00, 0XC1, 0X81, 0X40, 0X01, 0XC0, 0X80, 0X41,
0X00, 0XC1, 0X81, 0X40, 0X01, 0XC0, 0X80, 0X41, 0X01, 0XC0, 0X80, 0X41, 0X00, 0XC1, 0X81, 0X40 };

static const unsigned char code LoCRCTable[] = {
0X00, 0XC0, 0XC1, 0X01, 0XC3, 0X03, 0X02, 0XC2, 0XC6, 0X06, 0X07, 0XC7, 0X05, 0XC5, 0XC4, 0X04,
0XCC, 0X0C, 0X0D, 0XCD, 0X0F, 0XCF, 0XCE, 0X0E, 0X0A, 0XCA, 0XCB, 0X0B, 0XC9, 0X09, 0X08, 0XC8,
0XD8, 0X18, 0X19, 0XD9, 0X1B, 0XDB, 0XDA, 0X1A, 0X1E, 0XDE, 0XDF, 0X1F, 0XDD, 0X1D, 0X1C, 0XDC,
0X14, 0XD4, 0XD5, 0X15, 0XD7, 0X17, 0X16, 0XD6, 0XD2, 0X12, 0X13, 0XD3, 0X11, 0XD1, 0XD0, 0X10,
0XF0, 0X30, 0X31, 0XF1, 0X33, 0XF3, 0XF2, 0X32, 0X36, 0XF6, 0XF7, 0X37, 0XF5, 0X35, 0X34, 0XF4,
0X3C, 0XFC, 0XFD, 0X3D, 0XFF, 0X3F, 0X3E, 0XFE, 0XFA, 0X3A, 0X3B, 0XFB, 0X39, 0XF9, 0XF8, 0X38,
0X28, 0XE8, 0XE9, 0X29, 0XEB, 0X2B, 0X2A, 0XEA, 0XEE, 0X2E, 0X2F, 0XEF, 0X2D, 0XED, 0XEC, 0X2C,
0XE4, 0X24, 0X25, 0XE5, 0X27, 0XE7, 0XE6, 0X26, 0X22, 0XE2, 0XE3, 0X23, 0XE1, 0X21, 0X20, 0XE0,
0XA0, 0X60, 0X61, 0XA1, 0X63, 0XA3, 0XA2, 0X62, 0X66, 0XA6, 0XA7, 0X67, 0XA5, 0X65, 0X64, 0XA4,
0X6C, 0XAC, 0XAD, 0X6D, 0XAF, 0X6F, 0X6E, 0XAE, 0XAA, 0X6A, 0X6B, 0XAB, 0X69, 0XA9, 0XA8, 0X68,
0X78, 0XB8, 0XB9, 0X79, 0XBB, 0X7B, 0X7A, 0XBA, 0XBE, 0X7E, 0X7F, 0XBF, 0X7D, 0XBD, 0XBC, 0X7C,
0XB4, 0X74, 0X75, 0XB5, 0X77, 0XB7, 0XB6, 0X76, 0X72, 0XB2, 0XB3, 0X73, 0XB1, 0X71, 0X70, 0XB0,
0X50, 0X90, 0X91, 0X51, 0X93, 0X53, 0X52, 0X92, 0X96, 0X56, 0X57, 0X97, 0X55, 0X95, 0X94, 0X54,
0X9C, 0X5C, 0X5D, 0X9D, 0X5F, 0X9F, 0X9E, 0X5E, 0X5A, 0X9A, 0X9B, 0X5B, 0X99, 0X59, 0X58, 0X98,
0X88, 0X48, 0X49, 0X89, 0X4B, 0X8B, 0X8A, 0X4A, 0X4E, 0X8E, 0X8F, 0X4F, 0X8D, 0X4D, 0X4C, 0X8C,
0X44, 0X84, 0X85, 0X45, 0X87, 0X47, 0X46, 0X86, 0X82, 0X42, 0X43, 0X83, 0X41, 0X81, 0X80, 0X40 };

unsigned short QuickCRC16( unsigned char * pMsg, unsigned short iSize )
{
   unsigned char index;
   unsigned char iHiVal;
   unsigned char iLoVal;

   iHiVal = 0XFF;
   iLoVal = 0XFF;

   while ( iSize-- )
   {
      index  = iLoVal ^ *pMsg++;
      iLoVal = iHiVal ^ HiCRCTable[index];
      iHiVal = LoCRCTable[index];
   }
   return ( iHiVal << 8 | iLoVal );
}
```

CRC-16 公式算法

```
unsigned char ReverseBitOrder08( unsigned char iSrc ) small
{
   unsigned char index;
   unsigned char iDst;

   iDst = iSrc & 0X01;
   for( index = 0X00; index < 0X07; index++ )
   {
      iDst <<= 0X01;
      iSrc >>= 0X01;
      iDst |= ( iSrc & 0X01 );
   }
   return iDst;
}

unsigned short ReverseBitOrder16( unsigned short iSrc ) small
{
   unsigned char index;
   unsigned short iDst;

   iDst = iSrc & 0X01;
   for( index = 0X00; index < 0X0F; index++ )
   {
      iDst <<= 0X01;
      iSrc >>= 0X01;
      iDst |= ( iSrc & 0X01 );
   }
   return iDst;
}

unsigned short ArithCRC16( unsigned char * pMsg, unsigned short iSize ) small
{
   unsigned char  index;
   unsigned short iCRC;

   // The default value
   iCRC = 0XFFFF;
   while ( iSize-- )
   {
      iCRC ^= ( ( ( unsigned short ) ReverseBitOrder08( *pMsg ) ) << 0X08 );
      for ( index = 0X00; index < 0X08; index++ )
      {
         if ( iCRC & 0X8000 )
         {
            iCRC = ( iCRC << 1 ) ^ 0X8005;
         }
         else
```

```
            {
                iCRC <<= 1;
            }
        }
        pMsg++;
    }
    return ReverseBitOrder16( iCRC );
}
```

13. 附录-2：CRC32 算法

```
// ------------------------------------------------------------------------------
// DESCRIPTION: CRC-32校验的字节表
// ------------------------------------------------------------------------------
static const unsigned long CRC32Table[256] = {
0X00000000, 0X77073096, 0XEE0E612C, 0X990951BA, 0X076DC419, 0X706AF48F, 0XE963A535, 0X9E6495A3,
0X0EDB8832, 0X79DCB8A4, 0XE0D5E91E, 0X97D2D988, 0X09B64C2B, 0X7EB17CBD, 0XE7B82D07, 0X90BF1D91,
0X1DB71064, 0X6AB020F2, 0XF3B97148, 0X84BE41DE, 0X1ADAD47D, 0X6DDDE4EB, 0XF4D4B551, 0X83D385C7,
0X136C9856, 0X646BA8C0, 0XFD62F97A, 0X8A65C9EC, 0X14015C4F, 0X63066CD9, 0XFA0F3D63, 0X8D080DF5,
0X3B6E20C8, 0X4C69105E, 0XD56041E4, 0XA2677172, 0X3C03E4D1, 0X4B04D447, 0XD20D85FD, 0XA50AB56B,
0X35B5A8FA, 0X42B2986C, 0XDBBBC9D6, 0XACBCF940, 0X32D86CE3, 0X45DF5C75, 0XDCD60DCF, 0XABD13D59,
0X26D930AC, 0X51DE003A, 0XC8D75180, 0XBFD06116, 0X21B4F4B5, 0X56B3C423, 0XCFBA9599, 0XB8BDA50F,
0X2802B89E, 0X5F058808, 0XC60CD9B2, 0XB10BE924, 0X2F6F7C87, 0X58684C11, 0XC1611DAB, 0XB6662D3D,
0X76DC4190, 0X01DB7106, 0X98D220BC, 0XEFD5102A, 0X71B18589, 0X06B6B51F, 0X9FBFE4A5, 0XE8B8D433,
0X7807C9A2, 0X0F00F934, 0X9609A88E, 0XE10E9818, 0X7F6A0DBB, 0X086D3D2D, 0X91646C97, 0XE6635C01,
0X6B6B51F4, 0X1C6C6162, 0X856530D8, 0XF262004E, 0X6C0695ED, 0X1B01A57B, 0X8208F4C1, 0XF50FC457,
0X65B0D9C6, 0X12B7E950, 0X8BBEB8EA, 0XFCB9887C, 0X62DD1DDF, 0X15DA2D49, 0X8CD37CF3, 0XFBD44C65,
0X4DB26158, 0X3AB551CE, 0XA3BC0074, 0XD4BB30E2, 0X4ADFA541, 0X3DD895D7, 0XA4D1C46D, 0XD3D6F4FB,
0X4369E96A, 0X346ED9FC, 0XAD678846, 0XDA60B8D0, 0X44042D73, 0X33031DE5, 0XAA0A4C5F, 0XDD0D7CC9,
0X5005713C, 0X270241AA, 0XBE0B1010, 0XC90C2086, 0X5768B525, 0X206F85B3, 0XB966D409, 0XCE61E49F,
0X5EDEF90E, 0X29D9C998, 0XB0D09822, 0XC7D7A8B4, 0X59B33D17, 0X2EB40D81, 0XB7BD5C3B, 0XC0BA6CAD,
0XEDB88320, 0X9ABFB3B6, 0X03B6E20C, 0X74B1D29A, 0XEAD54739, 0X9DD277AF, 0X04DB2615, 0X73DC1683,
0XE3630B12, 0X94643B84, 0X0D6D6A3E, 0X7A6A5AA8, 0XE40ECF0B, 0X9309FF9D, 0X0A00AE27, 0X7D079EB1,
0XF00F9344, 0X8708A3D2, 0X1E01F268, 0X6906C2FE, 0XF762575D, 0X806567CB, 0X196C3671, 0X6E6B06E7,
0XFED41B76, 0X89D32BE0, 0X10DA7A5A, 0X67DD4ACC, 0XF9B9DF6F, 0X8EBEEFF9, 0X17B7BE43, 0X60B08ED5,
0XD6D6A3E8, 0XA1D1937E, 0X38D8C2C4, 0X4FDFF252, 0XD1BB67F1, 0XA6BC5767, 0X3FB506DD, 0X48B2364B,
0XD80D2BDA, 0XAF0A1B4C, 0X36034AF6, 0X41047A60, 0XDF60EFC3, 0XA867DF55, 0X316E8EEF, 0X4669BE79,
0XCB61B38C, 0XBC66831A, 0X256FD2A0, 0X5268E236, 0XCC0C7795, 0XBB0B4703, 0X220216B9, 0X5505262F,
0XC5BA3BBE, 0XB2BD0B28, 0X2BB45A92, 0X5CB36A04, 0XC2D7FFA7, 0XB5D0CF31, 0X2CD99E8B, 0X5BDEAE1D,
0X9B64C2B0, 0XEC63F226, 0X756AA39C, 0X026D930A, 0X9C0906A9, 0XEB0E363F, 0X72076785, 0X05005713,
0X95BF4A82, 0XE2B87A14, 0X7BB12BAE, 0X0CB61B38, 0X92D28E9B, 0XE5D5BE0D, 0X7CDCEFB7, 0X0BDBDF21,
0X86D3D2D4, 0XF1D4E242, 0X68DDB3F8, 0X1FDA836E, 0X81BE16CD, 0XF6B9265B, 0X6FB077E1, 0X18B74777,
0X88085AE6, 0XFF0F6A70, 0X66063BCA, 0X11010B5C, 0X8F659EFF, 0XF862AE69, 0X616BFFD3, 0X166CCF45,
0XA00AE278, 0XD70DD2EE, 0X4E048354, 0X3903B3C2, 0XA7672661, 0XD06016F7, 0X4969474D, 0X3E6E77DB,
0XAED16A4A, 0XD9D65ADC, 0X40DF0B66, 0X37D83BF0, 0XA9BCAE53, 0XDEBB9EC5, 0X47B2CF7F, 0X30B5FFE9,
```

```
0XBDBDF21C, 0XCABAC28A, 0X53B39330, 0X24B4A3A6, 0XBAD03605, 0XCDD70693, 0X54DE5729, 0X23D967BF,
0XB3667A2E, 0XC4614AB8, 0X5D681B02, 0X2A6F2B94, 0XB40BBE37, 0XC30C8EA1, 0X5A05DF1B, 0X2D02EF8D };



// ****************************************************************************
// Design Notes:
// f(x) = X^32 + X^26 + X^23 + X^22 + X^16 + X^12 + X^11 + X^10 + X^8 + X^7 +
// X^5 + X^4 + X^2 + X + 1
// ----------------------------------------------------------------------------
unsigned long QuickCRC32( unsigned char * pMsg, unsigned short iSize )
{
   register unsigned long iCRC32;
   register unsigned long iValue;
   register unsigned long iIndex;

   // Initial CRC value
   iCRC32 = 0XFFFFFFFFF;
   while( iSize-- )
   {
      iValue = *pMsg;
      iIndex = ( iCRC32 ^ iValue ) & 0XFF;
      iCRC32 = ( ( iCRC32 >> 0X08 ) & 0X00FFFFFFF ) ^ CRC32Table[iIndex];
      pMsg++;
   }
   return ( iCRC32 ^ 0XFFFFFFFFF );
}
```