ZigBee1086 无线数据采集卡

产品使用手册

北京阿尔泰科技发展有限公司

产品研发部修订

2015

V6.10.04

阿尔泰科技

www.art-control.com

目 录

1 模块结构及接线端子

1.1 接线端子

GND:电源负端

+Vs：电源正端

AI0+ ~ AI0-：1 路模拟量信号输入端子

DI0：数字量输入端

DO0：数字量输出端

1.2 LED 指示灯

ZIGBEE1086 模块有 2 个指示灯，分别为

① 红灯(Power)：电源指示灯，供电后常亮

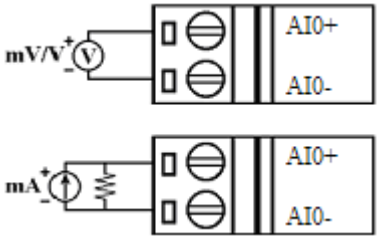
② 绿灯(Link)：通讯指示灯,ZIGBEE1086 模块在加入协调器(如 ZIGBEE1080)组建的网络后，此灯闪烁

2 模块主要性能指标

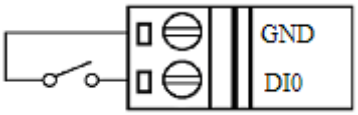
- 输入类型：0~5V，0~10V，±5V，±10V，0~20mA，4~20mA，±20mA
  客户需指定量程，其他非标准量程可定制
- 通道输入：1 路差分 (非隔离)
- 采样频率：10Hz
- 分辨率：16 bit
- 精度：见 TABEL1
- 数字量输入：干接点(非隔离)
- 集电极开路输出：外部供电电压：+5V，输出电流最大为 40mA
- 内置看门狗
- 电源：未处理 +10V ～ +30VDC
- 功耗：0.48W @24VDC （模块未使用 XBEE 网络传送数据）

3 接线图说明

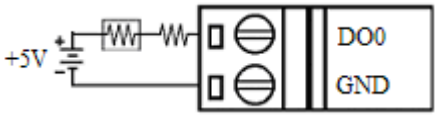
3.1 模拟量输入



3.2 干接点信号输入



3.3 集电极开路输出



4 模块相关表格

■ TABLE1　　模拟量输入类型配置代码及误差表

Input Type	Input Range	Typical Accuracy (%)	Maxinum Error (%)	Code (Hex)
±5V	±5V	±0.03	±0.05	08
±10V	±10V	±0.03	±0.05	09
±20mA	±20mA	±0.05	±0.08	0A
0~20mA	0~20mA	±0.05	±0.08	0B
4~20mA	4~20mA	±0.05	±0.08	0C
0~5V	0~5V	±0.03	±0.05	0D
0~10V	0~10V	±0.03	±0.05	0E

5 模块使用说明

本模块作为路由端或者终端节点与本公司的 Zigbee 数据服务器(如 Zigbee1080、Zigbee1080E 等)进行点对点或者点对多点组网连接时,模块内置 Zigbee 通讯模块的一些参数必须与服务器具有相同的设置，否则会造成模块不能自动组网连接，表现为 Zigbee1086 绿灯不闪烁同时远端的服务器(如 Zigbee1080、Zigbee1080E)网络连接指示灯不闪烁。

注意：出厂配置项参数只有使用本公司提供的 Zigbee 模块适配器及 X-CTU 软件才能查看。

内置 Zigbee 出厂配置项说明：

参数	范围	配置说明	备注
PAN ID	0x1~0xFFFFFFFFFFFFFFFF	同个网络 ID **必须相同**	出厂默认设置为 1
Scan Channels	0x1~0x7FFF 、0x1~0x3FFF	同个网络扫描通道**必须相同**	出厂默认设置为 3FFF
Baud Rate	1200~115200	此处必须设置为 38400，否则读取模块数据失败	

6MODBUS 协议地址分配

6.1 协议说明

ZIGBEE1086 遵循 ZIGBEE 网络数据传输方式，包括网络 API 方式传输协议和 MODBUS 协议部分。其中，API 方式为阿尔泰 ZIGBEE 系列产品通用通讯方式，MODBUS 部分为标准 MODBUS 协议，方便用户与组态软件进行连接。

6.2 MODBUS 协议部分

6.2.1 读开关量输出状态

功能码：01
数据地址：00001
说明：读取输出继电器的状态

数据说明：

地址	描述	说明
00001	第 01 路开关量输出状态	=1 高电平 =0 低电平
保留		

MODBUS 请求

功能码	1 BYTE	0x01
起始地址	2 BYTE	0x0000 TO 0xFFFF
读取数量	2 BYTE	1 TO 8

MODBUS 响应

功能码	1 BYTE	0x01
字节计数	1 BYTE	N
线圈状态	**n** BYTE	**n** =N or N+1

N =读取数量/8 如果余数不为 0 则 N=N+1

错误 响应

功能码	1 BYTE	0x01+ 0x80
错误代码	1 BYTE	0x1 or 0x2

举例

请求		响应	
模块地址	数据（hex）	模块地址	数据（hex）
功能码	01	功能码	01
起始地址高(字节)	00	字节计数	01
起始地址低(字节)	00	0（ch）状态	01
读取数量高(字节)	00		
读取数量低(字节)	01		

6.2.2 读开关量输入

功能码：02

数据地址：10001

说明：读取输入开关量的状态

数据说明：

地址	描述	说明
10001	第 01 路开关量输入状态	＝0 没有通电 ＝1 接通电源
保 留		

MODBUS 请求

功能码	1 BYTE	0x02
起始地址	2 BYTE	0x0000 TO 0xFFFF
读取数量	2 BYTE	1 TO 2000(0x7D0)

MODBUS 响应

功能码	1 BYTE	0x02
字节计数	1 BYTE	N
输入状态	**n** BYTE	**n** =N or N+1

N =读取数量/8 如果余数不为 0 则 N=N+1

错误 响应

功能码	1 BYTE	0x02+ 0x80
错误代码	1 BYTE	0x1 or 0x2

举例

请求		响应	
模块地址	数据（hex）	模块地址	数据（hex）
功能码	02	功能码	02
起始地址高(字节)	00	字节计数	01
起始地址低(字节)	00	0(ch)状态	01
读取数量高(字节)	00		
读取数量低(字节)	01		

6.2.3 读保持寄存器

功能码：03

数据起始地址：40129～40931

说明：读取保持寄存器的值

数据说明：读取的是十六位整数或无符合整数

地址	描述	说明
40129	模块类型寄存器	如：1086（HEX）
40130	模块类型后缀寄存器	如：2020
40131	模块 MODBUS 协议标识	‘＋’：2B20(HEX) - ASC II
40132	模块版本号	如：0600（HEX）
40133	模块地址	如：01
40134	模块波特率	如：05-38400bit/s，此波特率固定，不可配置
保 留		
40257	第 1 路模拟量输入量程	Bit15_Bit 8 必须输入为 0。 Bit7_Bit 0 采集量程。
保 留		
ZIGBEE 网络配置部分		
40901	PAN ID	0-0x3FFF, 0xFFFF
保留		
40905	通讯波特率	固定为 05（38400bits/S），此波特率为内部通讯使用，不作更改
保留		
40911	ZIGBEE 模块节点标识	ZIGBEE 模块节点标识有两个 ASC II 字符，bit15~bit8 为首字符，bit7~bit0 为尾字符，用于标志网络内节点名称
保留		
40931	保存 ZIGBEE 模块配置	1: 保存配置 0: 无效 将所有配置信息设置完毕后，需要发送此命令，使 ZIGBEE 模块保存配置
保留		

MODBUS 请求

功能码	1 BYTE	0x03
起始地址	2 BYTE	0x0000 TO 0xFFFF
读取数量	2 BYTE	1 TO 125(0x7D)

MODBUS 响应

功能码	1 BYTE	0x03
字节计数	1 BYTE	N*2
输入状态	N*2 BYTE	

错误 响应

功能码	1 BYTE	0x03+ 0x80
错误代码	1 BYTE	0x1 or 0x2

举例

请求		响应	
模块地址	数据（hex）	模块地址	数据（hex）
功能码	03	功能码	03
起始地址高(字节)	01	字节计数	02
起始地址低(字节)	00	保持寄存器高	00
读取数量高(字节)	00	保持寄存器低	0A
读取数量低(字节)	01		

6.2.4 读输入寄存器

功能码：04
数据起始地址：30257
说明：读取输入数据

数据说明：读取的是十六位整数或无符合整数

地址	描述	说明
30257	第 1 路模拟量输入低 16 位	

MODBUS 请求

功能码	1 BYTE	0x04
起始地址	2 BYTE	0x0000 TO 0xFFFF
读取数量	2 BYTE	1 TO 125(0x7D)

MODBUS 响应

功能码	1 BYTE	0x04
字节计数	1 BYTE	N*2
输入状态	N*2 BYTE	

错误 响应

功能码	1 BYTE	0x04+ 0x80
错误代码	1 BYTE	0x1 or 0x2

举例

请求		响应	
模块地址	数据（hex）	模块地址	数据（hex）
功能码	04	功能码	04
起始地址高(字节)	01	字节计数	02
起始地址低(字节)	00	输入寄存器高（9）	00
读取数量高(字节)	00	输入寄存器低（9）	0A
读取数量低(字节)	01		

6.2.5 设置单个保持寄存器

功能码：06

MODBUS 请求

功能码	1 BYTE	0x06
设置地址	2 BYTE	0x0000 TO 0xFFFF
设置内容	2 BYTE	0x0000 to 0xFFFF

MODBUS 响应

功能码	1 BYTE	0x06
设置地址	2 BYTE	0x0000 TO 0xFFFF
设置内容	2 BYTE	0x0000 to 0xFFFF

错误 响应

功能码	1 BYTE	0x06+ 0x80
错误代码	1 BYTE	0x1 or 0x2

举例

请求		响应	
模块地址	数据（hex）	模块地址	数据（hex）
功能码	06	功能码	06
设置地址高(字节)	01	设置地址高(字节)	01
设置地址低(字节)	00	设置地址低(字节)	00
设置内容高(字节)	00	设置内容高(字节)	00
设置内容低(字节)	09	设置内容低(字节)	09

6.2.6 设置多个保持寄存器

功能码：10

MODBUS 请求

功能码	1 BYTE	0x10
设置起始地址	2 BYTE	0x0000 TO 0xFFFF
设置长度	2 BYTE	0x0000 TO 0x7B0
字节计数	1 BYTE	N*2
设置内容	N*2 BYTE	

MODBUS 响应

功能码	1 BYTE	0x10
设置起始地址	2 BYTE	0x0000 TO 0xFFFF
设置长度	2 BYTE	0x0000 TO 0x7B0

错误 响应

功能码	1 BYTE	0x10+ 0x80
错误代码	1 BYTE	0x1 or 0x2

举例

请求		响应	
模块地址	数据（hex）	模块地址	数据（hex）
功能码	10	功能码	10
设置地址高(字节)	01	设置地址高(字节)	01
设置地址低(字节)	00	设置地址低(字节)	00
设置数量高(字节)	00	设置数量高(字节)	00
设置数量低(字节)	01	设置数量低(字节)	01
字节计数	02		
设置内容高(字节)	00		
设置内容低(字节)	0D		

6.3 使用 MODBUS 协议模块注意事项

6.3.1 RTU 帧

使用 RTU 模式，每发一串完整的数据信息，称为一个 RTU 帧。每帧发送至少要以 3.5 个字符时间的间隔开始（如下表中的 T1-T4），在最后一个有效数据传输完成后，以一个 3.5 个字符时间的间隔作为该帧的结束。

消息帧格式表：

起始间隔	设备地址	功能代码	数据	CRC 校验	结束间隔
T1-T4	1 字节	1 字节	N 字节	2 字节	T1-T4

6.3.2 CRC 校验

使用 RTU 模式，消息帧包括了 CRC 校验值。整个消息帧中，除 CRC 校验的 2 个字节外的所有数据均参与 CRC 运算。

CRC 校验值长度为 2 个字节，消息帧内数据计算完成后，将 CRC 运算结果放到本消息的最后。接收设备收到消息帧后，重新计算 CRC 校验值，如和收到的校验值相同，则说明数据包传输正常；反之，则有误。

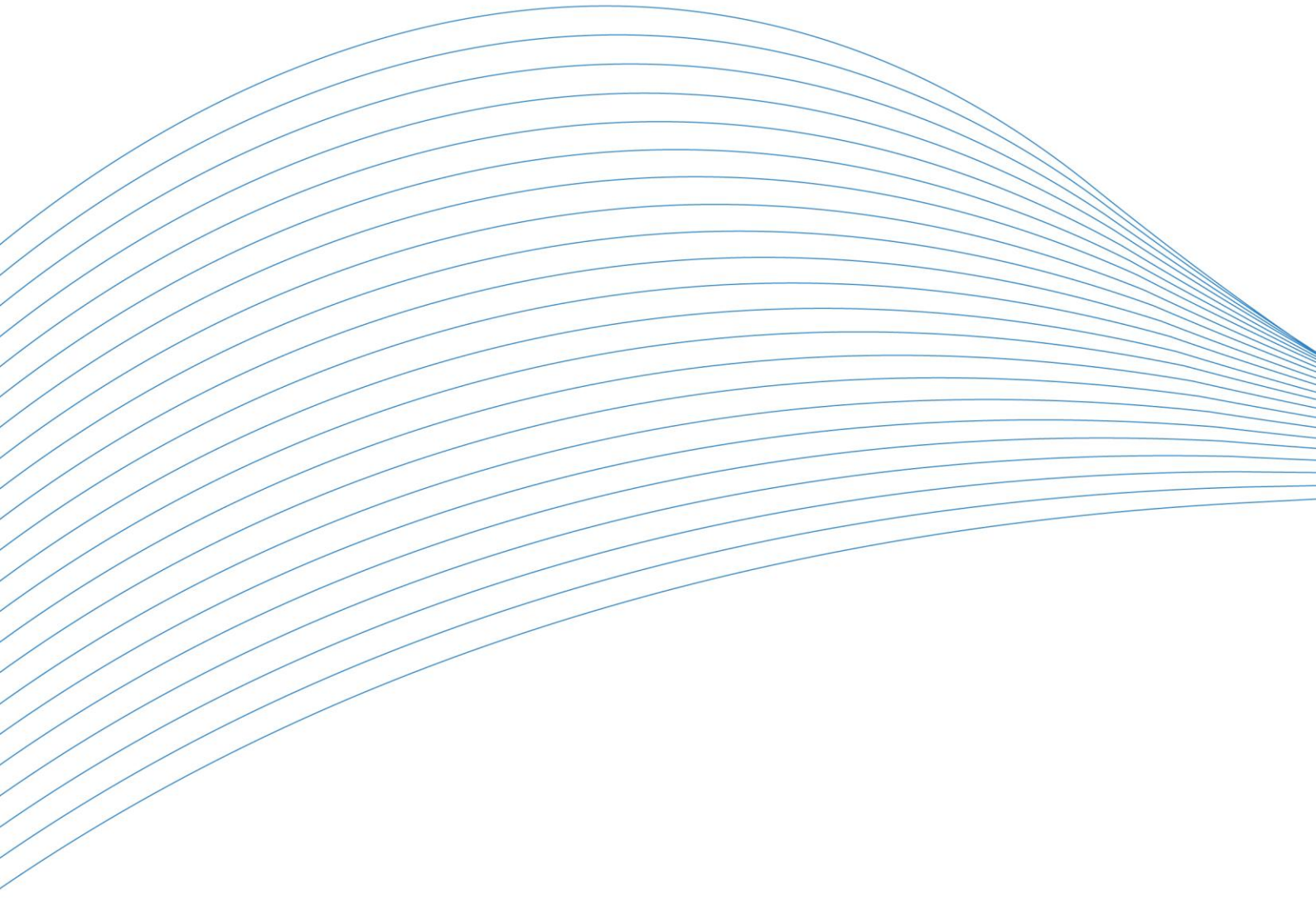
CRC 校验的 C 语言函数如下：

```
/**********************************************
-> 函数名称: CRCVerify ()
-> 函数功能: 校验接收和发送的命令(CRC)
-> 函数入口: 校验数据指针
             校验字节的数量
-> 函数出口: 无
**********************************************/
WORD CRCVerify (BYTE *pMsg,    WORD usDataLen)
{
    BYTE ucCRCHi = 0xFF ;     /* high byte of CRC initialized */
    BYTE ucCRCLo = 0xFF ;     /* low byte of CRC initialized */
    WORD uIndex    = 0;          /* will index into CRC lookup table */

    while (usDataLen--)        /* pass through message buffer */
    {
        uIndex     = ucCRCHi ^ *pMsg++ ; /* calculate the CRC */
        ucCRCHi = ucCRCLo ^ auchCRCHi[uIndex] ;
        ucCRCLo = auchCRCLo[uIndex] ;
    }
    return (ucCRCHi << 8 | ucCRCLo) ;
} /*返回的值即为 RTU 消息帧的校验值*/

/* Table of CRC values for high – order byte */
static BYTE auchCRCHi[] = {
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
} ;

/* Table of CRC values for low – order byte */
static char auchCRCLo[] = {
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
} ;
```

6.3.3 发送数据帧时的寄存器地址填写

在发送的数据帧中，需要指定要访问的寄存器地址。在本公司的 MODBUS 协议地址分配表中，寄存器的地址 = 数据帧中寄存器的地址 + 1。举例来说，需要访问输入寄存器的 30257 地址，其中“3”为输入寄存器地址的前缀，在某些组态软件中会用到，但是“3”不作为寄存器地址。“0257”是十进制数，表示寄存器地址，如果客户自己编写程序，则在发送的数据帧中，寄存器地址应填写“256”（即 0x0100）；如果在组态软件中，则直接填写“257”即可。



北京阿尔泰科技发展有限公司
服务热线：400-860-3335
邮编：100086
传真：010-62901157