MD20x 电调

FOC 控制器

UM01010101 1.1.00 Date:2023/11/15

类别	内容
关键词	用户手册
摘要	

修订历史

版本	日期	原因
V1.0.00	2023/05/12	创建文档
V1.1.00	2023/11/14	更新部分描述，更新文档 logo

目 录

1. 适用范围

本文档适用于所有 MD20x 系列产品，x 为模块软件代号（x=0、1、2）：

- 0 为伺服控制（力矩、速度、位置控制）；
- 1 为霍尔（力矩、速度控制、相序自学习）；
- 2 为无感（速度控制）。

2. 产品简介

2.1 产品概述

MD20x 电调驱动模块是由立功科技•求远电子针对无刷电机控制而设计的一款 FOC 驱动器，针对中低功率直流无刷电机（支持有感、无感）的高性能产品，适用于 15V~24V 的三相直流无刷电机，具有完善的保护机制和功能接口，满足各种应用场景。

2.2 主要特点

- 采用 FOC 矢量控制；
- 速度环、位置环、电流环，三环闭环控制；
- 支持顺风启动、逆风启动；
- 串口指令调速，基于 MODBUS RTU 协议
- 全面的错误报警机制，支持过压、欠压、过流等异常报警；
- 支持用户自适配电机参数，可适配各种三相无刷直流电机。

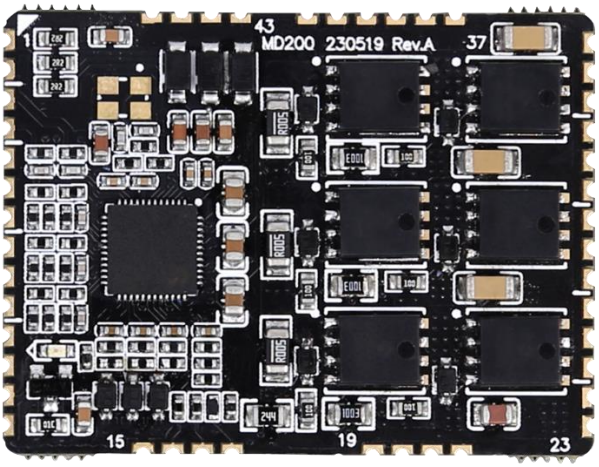

图 2.1 MD20x 电调模块

2.3 功能框图

MD20x 整体框图如图 2.2 所示，使用 15V~24V 直流电源输入，支持 RS485 串口通讯，可外接控制使能、刹车及方向控制信号，报警提示信号及报警信号输出，支持无感、伺服、霍尔电机的接入，伺服电机除速度控制外，还支持位置及力矩模式。

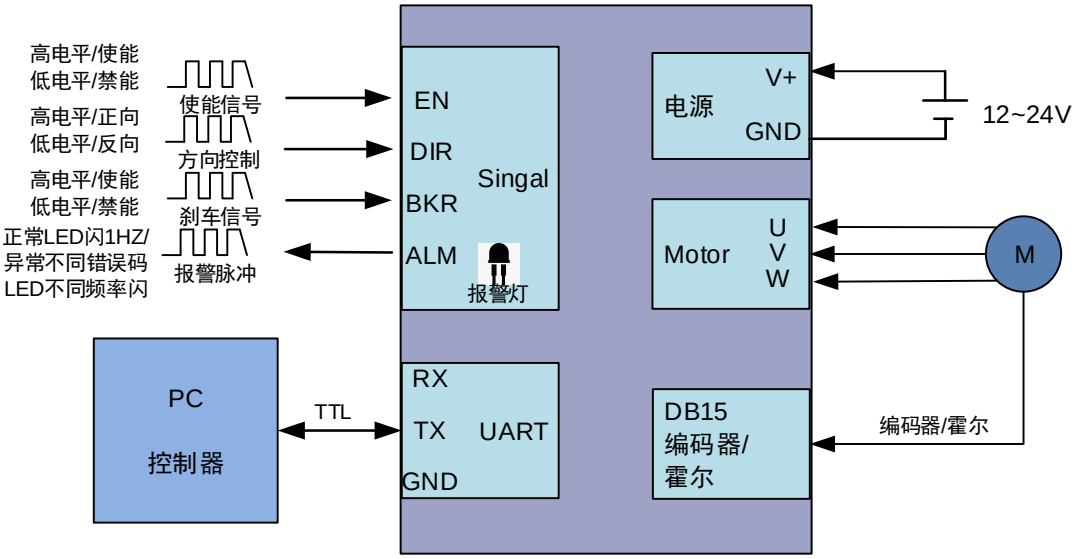

图 2.2　MD20x 控制框图

3. 输入输出信号

MD20x 电调支持完善的输入输出控制信号，可以非常方便的接入所需系统。输入控制信号包含电调使能控制信号、刹车信号、模式选择信号以及调速信号，输出信号主要是异常状态的报警信号，支持上报异常状态。

3.1 控制信号输入

MD20x 电调的控制信号使用差分输入，其他控制器可以通过这些控制引脚输入控制信号控制电机运行，支持的输入控制信号的说明，电调使能、刹车、转向功能，具体控制逻辑详见表 3.1。需注意方向引脚电平为 0 时，代表电机反转，此处反转意义为：若串口设置当前转速为 500RPM，则方向引脚电平置 0 时，电机会以-500RPM 运行；若串口设置当前转速为-500RPM，则方向引脚电平置 0 时，电机会以 500RPM 运行。

表 3.1 控制信号输入逻辑对照表

控制口	电平状态	控制功能	备注
使能	1（默认）	电调使能	电调能够驱动电机
	0	电调禁能	电调无法驱动电机
刹车	1（默认）	刹车禁能	电调能够驱动电机
	0	刹车使能	电调刹车，电机停止
方向	1（默认）	电机正转	电机正转
	0	电机反转	电机反转

3.2 报警信号输出

MD20x 电调具有报警保护功能，在检测到异常情况时，将会关闭电机并输出报警信号，其中除了使能和刹车控制之外，其余报警都需要清除错误才能解除报警。

表 3.2 报警信号对照表

报警代码	报警原因	报警脉冲频率	备注
0	正常工作	1Hz	正常状态
1	未使能	2 Hz	外部使能控制处于未使能状态
2	硬件过流	3 Hz	内部硬件过流警报（默认 30A）
3	刹车状态	4 Hz	外部刹车使能控制处于使能状态
4	软件过流	5 Hz	电流超过用户配置软件过流值
8	过压	6 Hz	电压超过用户配置最大电压
16	欠压	7 Hz	电压小于用户配置最小电压
32	缺相	8 Hz	电机相线接线缺失或者接触不良
128	超速	9 Hz	电机转速超过设定转速

256	堵转	10 Hz	由于负载造成电机转速过低或电流过大
512	启动失败	11 Hz	无感电机启动异常
1024	紧急停机	12 Hz	电机由异常情况而处于紧急停止状态
2048	校准错误	13 Hz	伺服电机上电后校准失败

4. 通讯协议

MD20x 电调的通信协议使用 MODBUS RTU 协议，电调模块作为 MODBUS 从机设备。模块串口通信参数为：波特率 115200，数据位 8bits，停止位 1bits，校验方式为无校验。

4.1 通讯方式

通信模式是主从方式，也即主机请求，从机应答的方式。电调模块作为从机，控制电调模块的设备为主机。

4.2 帧格式

主请求与从应答都使用

表 4.1 所示的帧格式。

表 4.1 帧格式

子节点地址	功能代码	数据	CRC	
1 字节	1 字节	0 到 252 字节	2 字节	
			CRC 低	CRC 高

4.2.1 子节点地址

合法的子节点地址为十进制 0－247。每个子设备被赋予 1－247 范围中的地址。主节点通过将子节点的地址放到报文的地址域对子节点寻址。当子节点返回应答时，它将自己的地址放到应答报文的地址域以让主节点知道哪个子节点在回答。

4.2.2 功能代码

指明主节点要执行的动作。读数据或者写数据。

4.2.3 数据

与功能码对应的数据。如：写数据时的写入内容，写入数量；读数据时的读取内容，读入数量。

4.2.4 CRC 校验

16 位循环冗余校验，对整个报文的内容进行 CRC 校验，最终生成一个 16 位的校验码。

4.3 数据模型

MODBUS 提供四种数据类型，分别是 1bit 的离散量输入与线圈，还有 16bit 的输入寄存器与保持寄存器。如表 4.2 所示。

表 4.2 数据模型

基本表格	对象类型	访问类型	内容
离散量输入	单个比特	只读	I/O 系统提供这种类型数据
线圈	单个比特	读写	通过应用程序改变这种类型数据
输入寄存器	16 比特字	只读	I/O 系统提供这种类型数据
保持寄存器	16 比特字	读写	通过应用程序改变这种类型数据

MD20x 电调模块中只使用 16bit 位的数据类型，也就是输入保持寄存器与保持寄存器。

输入寄存器是只读的，输入寄存器保存电调的实时信息，如实际转速，实际位置，当前电压、电流等。这些参数都是只读的，用户通过读取输入寄存器得到电调信息。

保持寄存器是可读可写的，保持寄存器中保存电调的目标速度，目标位置，目标力矩等参数。用户通过写入保持寄存器来设置电调速度，位置等信息。

4.4 功能代码定义

协议中的功能代码表明主节点要执行的动作。MODBUS 中公共功能码如图 4.1 所示。电调模块只使用 16 比特访问这一类的功能。常用的有 03，04，06, 16 这几个功能码。也就是读取输入寄存器，读取保持寄存器，写入保持寄存器。使用这几个功能码就能实现读取电调信息，以及设置电调信息。

				功能码			
				码	子码	（十六进制）	页
数据访问	比特访问	物理离散量输入	读输入离散量	02		02	11
		内部比特或物理线圈	读线圈	01		01	10
			写单个线圈	05		05	16
			写多个线圈	15		0F	37
	16 比特访问	输入存储器	读输入寄存器	04		04	14
		内部存储器或物理输出存储器	读多个寄存器	03		03	13
			写单个寄存器	06		06	17
			写多个寄存器	16		10	39
			读/写多个寄存器	23		17	47
			屏蔽写寄存器	22		16	46
	文件记录访问		读文件记录	20	6	14	42
			写文件记录	21	6	15	44
封装接口			读设备识别码	43	14	2B	

图 4.1 MODBUS 公用功能代码

4.5 CRC 校验

CRC 校验是基于循环冗余校验 (CRC - Cyclical Redundancy Checking) 算法的错误检验域。CRC 域检验整个报文的内容，校验码包含由两个 8 位字节组成的一个 16 位值。

CRC 的计算，开始对一个 16 位寄存器预装全 1。然后将报文中的连续的 8 位字节对其进行后续的计算。只有字符中的 8 个数据位参与生成 CRC 的运算，起始位，停止位和校验位不参与 CRC 计算。

CRC 的生成过程中，每个 8 位字符与寄存器中的值异或。然后结果向最低有效位(LSB)方向移动(Shift) 1 位，而最高有效位(MSB)位置充零。 然后提取并检查 LSB：如果 LSB 为 1， 则寄存器中的值与一个固定的预置值异或；如果 LSB 为 0， 则不进行异或操作。这个过程将重复直到执行完 8 次移位。完成最后一次（第 8 次）移位及相关操作后，下一个 8 位字节与寄存器的当前值异或，然后又同上面描述过的一样重复 8 次。当所有报文中子节都运算之后得到的寄存器中的最终值，就是 CRC。当 CRC 附加在报文之后时，首先附加低字节，然后是高字节。

具体实现可参考附录 A。

4.6 功能码使用详解

4.6.1 04(0x04) 读输入寄存器

功能码 0x04 用于读取输入寄存器的值。通过主机发送要读取的输入寄存器起始地址、读取数量，从机就会返回对应寄存器的值。通信的具体数据格式如表 4.3、表 4.4 所示。

表 4.3 主机请求格式

功能码	1 个字节	0x04
输入寄存器起始地址	2 个字节	0x0000 至 0xFFFF
输入寄存器数量	2 个字节	0x0001 至 0x007D

表 4.4 从机响应格式

功能码	1 个字节	0x04
字节数	1 个字节	2×N
输入寄存器值	2×N 个字节 (N = 请求帧中的寄存器数量)	0x0000 至 0xFFFF

表 4.5 中提供了一个具体的示例。示例中请求读取节点地址 0x0A 的输入寄存器 0x09，结果为输入寄存器 0x09 对应数据为 0x000A。

表 4.5 输入寄存器读取示例

请求		响应	
节点地址	0x0A	节点地址	0x0A
功能码	0x04	功能码	0x04
起始地址 Hi	0x00	字节数	0x02
起始地址 Lo	0x09	输入寄存器 Hi	0x00

输入寄存器数量 Hi	0x00	输入寄存器 Lo	0x0A
输入寄存器数量 Lo	0x01	CRC 校验 Lo	0x9C
CRC 校验 Lo	0xE0	CRC 校验 Hi	0xF6
CRC 校验 Hi	0xB3		

4.6.2 03(0x03) 读保持寄存器

功能码 0x03 用于读取保持寄存器的值。通过主机发送要读取的保持寄存器起始地址、读取的数量，从机就会返回对应寄存器的值。通信的具体数据格式如表 4.6、表 4.7 所示。

表 4.6 主机请求格式

功能码	1 个字节	0x03
保持寄存器起始地址	2 个字节	0x0000 至 0xFFFF
保持寄存器数量	2 个字节	0x0001 至 0x007D

表 4.7 从机响应格式

功能码	1 个字节	0x03
字节数	1 个字节	2×N
输入寄存器值	2×N 个字节 (N = 请求帧中的寄存器数量)	0x0000 至 0xFFFF

表 4.8 提供了具体读取的示例，示例中请求读取节点地址 0x0A 的保持寄存器 0x03-0x05，读取出 0x03-0x05 保持寄存器的值为：0x0004，0x0005，0x0006。

表 4.8 保持寄存器读取示例

请求		响应	
节点地址	0x0A	节点地址	0x0A
功能码	0x03	功能码	0x03
起始地址 Hi	0x00	字节数	0x06
起始地址 Lo	0x03	寄存器值(0x03) Hi	0x00
保持寄存器数量 Hi	0x00	寄存器值(0x03) Lo	0x04
保持寄存器数量 Lo	0x03	寄存器值(0x04) Hi	0x00
CRC 校验 Lo	0xF4	寄存器值(0x04) Lo	0x05
CRC 校验 Hi	0xB0	寄存器值(0x05) Hi	0x00
		寄存器值(0x05) Lo	0x06
		CRC 校验 Lo	0x33

		CRC 校验 Hi	0x86

4.6.3 06(0x06) 写单个寄存器

功能码 0x06 用于写入单个保持寄存器的值。通过主机发送要写入的保持寄存器地址、写入的数值，从机就能完成写入，写入完成后发出应答。通信的具体数据格式如表 4.9、表 4.10 所示。

表 4.9 主机请求格式

功能码	1 个字节	0x06
保持寄存器地址	2 个字节	0x0000 至 0xFFFF
写入的寄存器值	2 个字节	0x0000 至 0xFFFF

表 4.10 从机响应格式

功能码	1 个字节	0x06
保持寄存器地址	2 个字节	0x0000 至 0xFFFF
写入的寄存器值	2 个字节	0x0000 至 0xFFFF

表 4.11 提供了具体的写入示例。示例中请求把节点地址 0x0A 的保持寄存器 0x06 写入为 0x1122。

表 4.11 写单个保持寄存器示例

请求		响应	
节点地址	0x0A	节点地址	0x0A
功能码	0x06	功能码	0x06
保持寄存器地址 Hi	0x00	保持寄存器地址 Hi	0x00
保持寄存器地址 Lo	0x06	保持寄存器地址 Lo	0x06
写入的寄存器值 Hi	0x11	写入的寄存器值 Hi	0x11
写入的寄存器值 Lo	0x22	写入的寄存器值 Lo	0x22
CRC 校验 Lo	0xF4	CRC 校验 Lo	0xF4
CRC 校验 Hi	0xF8	CRC 校验 Hi	0xF8

4.6.4 16(0x10) 写多个寄存器

功能码 0x10 用于写入多个保持寄存器的值。通过主机发送要写入的保持寄存器起始地址、写入的寄存器数量、写入的数值，从机就能完成写入动作，写入完成后发出应答。具体

通信数据格式如表 4.12、表 4.13 所示。

表 4.12　主机请求示例

功能码	1 个字节	0x10
保持寄存器起始地址	2 个字节	0x0000 至 0xFFFF
写入的寄存器数量	2 个字节	0x0001 至 0x0078
写入的字节数	1 个字节	2×N
写入的寄存器值	2×N 个字节 (N=写入的寄存器数量)	0x0000 至 0xFFFF

表 4.13　从机响应格式

功能码	1 个字节	0x10
保持寄存器地址	2 个字节	0x0000 至 0xFFFF
写入的寄存器数量	2 个字节	0x0001 至 0x0078

表 4.14 提供了写入的具体示例。示例中请求把节点地址 0x0A 的保持寄存器 0x06 0x07 写入为 0x1122 0x3344。

表 4.14　写入多个保持寄存器示例

请求		响应	
节点地址	0x0A	节点地址	0x0A
功能码	0x10	功能码	0x10
寄存器起始地址 Hi	0x00	寄存器起始地址 Hi	0x00
寄存器起始地址 Lo	0x06	寄存器起始地址 Lo	0x06
写入寄存器数量 Hi	0x00	写入寄存器数量 Hi	0x00
写入寄存器数量 Lo	0x02	写入寄存器数量 Lo	0x02
写入的字节数	0x04	CRC 校验 Lo	0xA0
写入寄存器值 1Hi	0x11	CRC 校验 Hi	0xB2
写入寄存器值 1Lo	0x22		
写入寄存器值 2Hi	0x33		
写入寄存器值 2Lo	0x44		
CRC 校验 Lo	0xF4		
CRC 校验 Hi	0xF8		

4.7 电调模块 MODBUS 通讯定义

4.7.1 从机地址

默认为 0x0A，可通过修改通信指令修改从机地址。

4.7.2 寄存器定义

电调模块的输入寄存器(只读)定义如表 4.15 所示。

表 4.15 输入寄存器定义

输入寄存器地址	电调参数	单位
0x1000	当前转速(高 16 位)	RPM
0x1001	当前转速(低 16 位)	
0x1002	当前位置(高 16 位)	编码器线数
0x1003	当前位置(低 16 位)	
0x1004	当前力矩电流(高 16 位)	mA
0x1005	当前力矩电流(低 16 位)	
0x1006	运行错误码(高 16 位)	电机状态码
0x1007	运行错误码(低 16 位)	
0x1008	母线电压(高 16 位)	mV
0x1009	母线电压(低 16 位)	
0x100A	软件版本号	0xX000（X: 1:无感 FOC ; 2: 霍尔 FOC; 3: 伺服 FOC）
0x100B	硬件版本号	0xAABB(AA:硬件版本 BB: 硬件子版本)

电调模块的保持寄存器(可读可写)定义如表 4.16 所示。

表 4.16 保持寄存器定义

保持寄存器地址	电调参数	单位
0x0000	目标转速(高 16 位)	RPM
0x0001	目标转速(低 16 位)	
0x0002	目标位置(高 16 位)	编码器线数
0x0003	目标位置(低 16 位)	
0x0004	目标力矩电流(高 16 位)	mA
0x0005	目标力矩电流(低 16 位)	
0x0006	运行模式	2-速度模式 3-位置模式

0x0007	从机地址	1 - 247
0x0008	霍尔自学习标志	0-不进行霍尔相序学习，1-启动先进行霍尔相序自学习。自学习完成后电调将此值自动清零
0x000f	参数保存标志	写入 0x55AA 后参数将更新到 FLASH 中，写入成功后电调将此值清零

4.8 电调通讯示例

4.8.1 速度控制

1. 获取当前电机转速

   请求帧：0A 04 10 00 00 02 74 70

   应答帧：0A 04 04 00 00 03 E8 41 FA

   获取到的电机转速为 0x000003E8，即为 1000RPM。

2. 设置电机转速

   请求帧：0A 10 00 00 00 02 04 00 00 03 E8 D6 35

   应答帧：0A 10 00 00 00 02 40 B3

   设置电机的转速为 1000RPM。

4.8.2 位置控制

1. 获取当前电机位置

   请求帧：0A 04 10 02 00 02 D5 B0

   应答帧：0A 04 04 00 00 0B B8 46 06

   获取到的电机位置为 0x00000BB8，即为 3000 线。

2. 设置电机位置

   请求帧：0A 10 00 02 00 02 04 00 00 13 88 5A 04

   应答帧：0A 10 00 02 00 02 E1 73

   设置电机的转速为 5000 线。

4.8.3 控制模式选择

控制模式的切换仅支持 MD200、MD201。

1. 设置电调为力矩模式

   请求帧：0A 10 00 06 00 01 02 00 01 14 C6

   应答帧：0A 10 00 06 00 01 E0 B3

2. 设置电调为速度模式

   请求帧：0A 10 00 06 00 01 02 00 02 54 C7

   应答帧：0A 10 00 06 00 01 E0 B3

3. 设置电调为位置模式

   请求帧：0A 10 00 06 00 01 02 00 03 95 07

应答帧：0A 10 00 06 00 01 E0 B3

5. 电机适配流程

MD20x 电调出厂前默认支持的电机为 BL57 电机，用户所用电机不尽相同，此时用户需要根据自己的电机参数更新到电调，用户可使用 MD Tool 软件可实现参数更新。

使用 MD Tool 软件参数更新时，主要分为两个方式，分别是修改当前参数与导入模板数据。

5.1 修改电调参数

参数更新功能可以查看当前电调设置的电机参数，并在软件上修改参数。

在参数更新界面，点击“电调参数获取”按钮，可获取当前连接电调的参数。可直接在输入框中修改对应的参数，修改完后，点击“电调参数写入”按钮，即可完成写入。如图 5.1 所示。

图 5.1 参数修改

如果对参数的定义不清晰，可鼠标放在该参数标签上，查看该参数的说明。如图 5.2 所示。

图 5.2 参数提示

5.2 导入电机参数模板

MD Tool 内置电机模板库，用户可以选择电机模板库中的不同电机，将参数写入到电调上，来驱动对应的电机。

点击“选择参数模板”按钮，选择连接的电调版本，以及电机型号，点击“导入参数”按钮，接口导入该电机的参数。如图 5.3 所示。

参数导入后，点击“电调参数写入”按钮，即可将参数写入电调。复位电调后，参数生效。

图 5.3 参数模板导入

附录A CRC 校验

```
static const UCHAR aucCRCHi[] = {
    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 UCHAR aucCRCLo[] = {
    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
};

USHORT
usMBCRC16( UCHAR * pucFrame, USHORT usLen )
{
    UCHAR               ucCRCHi = 0xFF;
    UCHAR               ucCRCLo = 0xFF;
    int                 iIndex;

    while( usLen-- )
    {
        iIndex = ucCRCLo ^ *( pucFrame++ );
        ucCRCLo = ( UCHAR )( ucCRCHi ^ aucCRCHi[iIndex] );
        ucCRCHi = aucCRCLo[iIndex];
    }
    return ( USHORT )( ucCRCHi << 8 | ucCRCLo );
}
```

6. 免责声明

本着为用户提供更好服务的原则，广州立功科技股份有限公司（下称“立功科技”）在本手册中将尽可能地为用户呈现详实、准确的产品信息。但介于本手册的内容具有一定的时效性，立功科技不能完全保证该文档在任何时段的时效性与适用性。立功科技有权在没有通知的情况下对本手册上的内容进行更新，恕不另行通知。为了得到最新版本的信息，请尊敬的用户定时访问立功科技官方网站或者与立功科技工作人员联系。感谢您的包容与支持！

专业 . 专注成就梦想

Dreams come true with professionalism and dedication.