

EtherCAT 总线型运动控制系统的特点与应用（上篇）

深圳市雷赛控制技术有限公司

左力

2019 年 7 月

目录

一. EtherCAT 总线	1
二. EtherCAT 总线型运动控制卡 DMC E3032.....	2
三. 松下 A6 系列 EtherCAT 伺服电机的应用	3
四. 雷赛 EtherCAT 总线型步进电机驱动器	18
五. 雷赛 EtherCAT 总线型闭环步进电机驱动器	21
六. EtherCAT 总线扩展模块	28
一) IO 扩展模块.....	28
二) 定位扩展模块.....	30
三) AD/DA 扩展模块.....	34
四) 编码器扩展模块.....	37
七. 多轴插补运动及总线模块的综合应用.....	42
八. 小结.....	49
九. 参考文献.....	51
十. 附录例程代码.....	52
例程 1: 伺服电机测试程序.....	52
例程 2: 步进电机及驱动器测试程序	57
例程 3: 闭环步进电机及驱动器测试程序	60
例程 4: 定位模块测试程序.....	64
例程 5: 编码器模块测试程序.....	67
例程 6: 曲线插补及各模块的应用	69

一. EtherCAT 总线

EtherCAT 总线是由德国倍福（BECKHOFF）自动化公司于 2003 年提出的实时工业以太网技术，具有高速和高效的优点。

其介质访问控制 MAC（MediaAccessControl）方式使用主从模式，主站发送以太网帧给各从站，从站从数据帧中抽取数据或将数据插入数据帧。从而彻底避免报文冲突。主站使用标准的以太网接口卡，从站使用专用的 EtherCAT 从站控制器 ESC（EtherCATSlaveController）。EtherCAT 物理层使用标准的以太网物理层器件。EtherCAT 协议结构如图 1.1 所示。

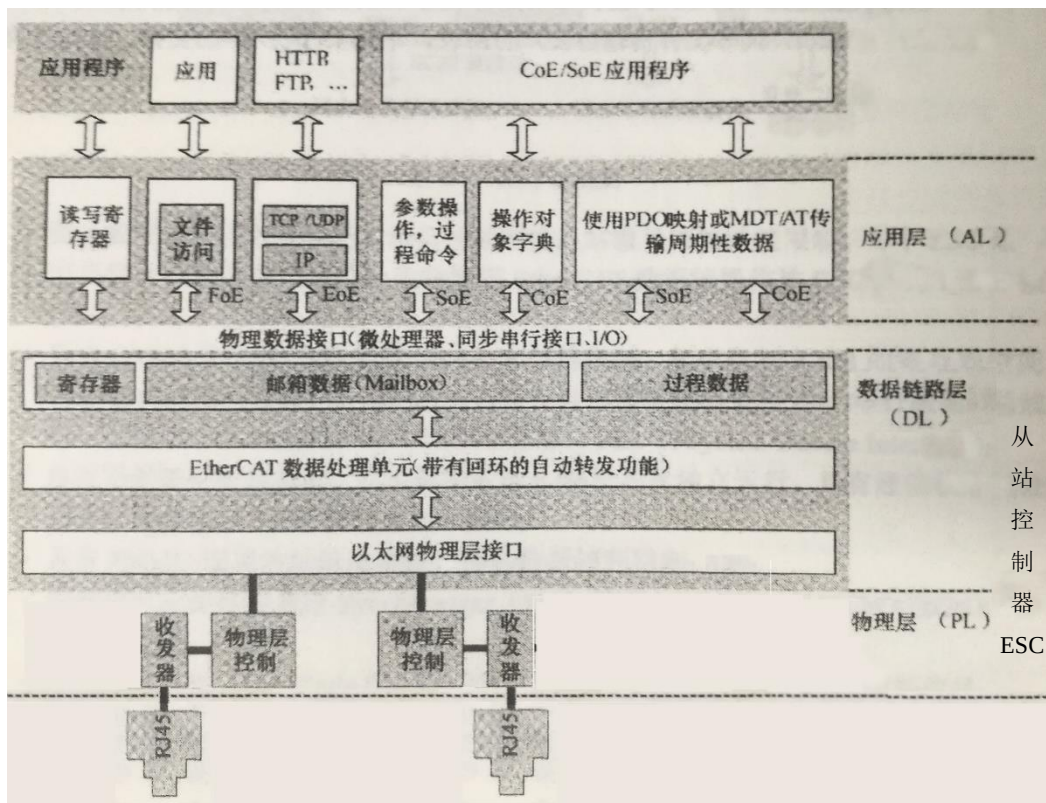


图 1.1. EtherCAT 协议结构

EtherCAT 包括以下几种应用协议：

- ◆ CoE(CANopen over EtherCAT)；该协议在 CANopen 协议的基础上做了扩充。
- ◆ SoE(Servo Drive over EtherCAT)；该协议用于伺服电机的控制。
- ◆ EoE(Ethernet over EtherCAT)
- ◆ FoE(File Access over EtherCAT)

EtherCAT 总线主要特点如下：

- ◆ 速度快，数据刷新周期可达 100 微秒。
- ◆ 同步性好，各从站节点设备的时钟同步精度可达 1 微秒。
- ◆ 通讯数据量大，波特率为 100Mbps。（比 CAN 总线的速度快 100 倍。）
- ◆ 从站节点数多，最大可有 65535 个。（比 CAN 总线的节点多 600 倍。）
- ◆ 通讯距离长，节点间最大距离可达 100 米。（比 CAN 总线的远 2.5 倍。）

- ◆ 完全符合以太网标准，可以与其他以太网设备及协议并存于同一总线，以太网交换机等标准结构主件也可用于 EtherCAT 总线。

二. EtherCAT 总线型运动控制卡 DMC E3032

2018 年雷赛公司推出的 DMC-E3032 运动控制卡最多可以控制 32 个 EtherCAT 总线型电机及扩展功能模块。可实现多轴点位运动、16 轴直线插补运动、3 轴空间圆弧插补、手轮控制、编码器位置检测、IO 控制、位置比较、位置锁存等功能。

DMC-E3032 运动控制卡的运动控制函数库功能丰富、易学易用，用户开发应用软件十分方便；并提供 C#、VB.net、Delphi、Labview 编程语言的例程。随卡免费提供的 Motion 调试软件，不但可以设置 EtherCAT 总线参数、总线节点的参数，而且可用于控制卡及运动控制系统的硬件测试。

DMC-E3032 运动控制卡的外形如图 2.1 所示，其主要参数如表 2.1 所示。

DMC-E3032 运动控制卡的卡号设置、IO 接口、编码器接口、手轮接口及驱动软件的安装方法都和普通脉冲运动控制卡相同。详见产品手册。



图 2.1. DMC-E3032 运动控制卡外观

表 2.1. DMC-E3032 运动控制卡的主要参数

技术指标	卡类型	DMC-E3032
支持在PC机中同时工作的卡数		8
单卡可扩展电机轴数		32
总线通讯速率		最大100Mbps
支持的总线周期		250us（16轴）、500us（32轴）、1ms（32轴）、2ms（32轴）
支持的插补坐标系个数		6
辅助编码器		2 个、32位有符号、4 MHz（4倍频后为16MHz）
通用数字输入口数量		8（可扩展）
通用数字输出口数量		8（可扩展）

三. 松下 A6 系列 EtherCAT 伺服电机的应用

一) A6 系列伺服电机的特点

松下公司 2016 年推出的 A6 系列产品的性能比 A5 系列有明显提高。如：电机尺寸更小、重量更轻；电机转速更高、响应更快，定位更精确，抑振效果更好。具体表现为：

- 最高转速由 5000rpm 提高到 6000rpm。
- 响应频率提高 139%，达 3.2kHz。
- 编码器分辨率提高至 23 位 ($2^{23} = 8388608$) (A5 系列：相对式 20 位、绝对式 17 位)
- A6 系列产品还增加了 EtherCAT 总线和 RTEX 总线型产品。

二) A6 系列 EtherCAT 总线型伺服电机的特点

A6 系列伺服电机 EtherCAT 总线的规格参数见表 3.1。

表 3.1. A6 系列伺服电机 EtherCAT 总线的规格参数表

项 目	规 格
Physical Layer	100BASE-TX (IEEE802.3)
波特率	100[Mbps] (Full duplex)
拓扑	线型(※线型连接以外请另外通过其他途径咨询)
连接电缆	屏蔽双绞线 CAT5e
电缆长度	节点间：最大 100[m]
连接从站(轴)数	最大 65535
通信端口	2ports (RJ45 connector)
Station Alias (ID)	设定范围 0~65535 <设定①> 低位 8bit: 旋转开关 2 位 (前面板) 上位 8bit: Object 3740h 或 <设定②> SII 保存值
Explicit Device ID	对应
设备 Profile	CoE (CANopen over EtherCAT)

A6 系列脉冲型驱动器的最大指令脉冲频率为 8MHz；电机每转 1 圈的最大指令脉冲数 (Pr0.08) 只能设为 8388608。

若电机最高转速为 6000 rpm，电机每转 1 圈的最大指令脉冲数设为 8388608。则脉冲输入频率为 800 MHz ($8388608 \text{ p/r} \times 100 \text{ r/s}$)。显然，这个与最大指令脉冲频率 8 MHz 有矛盾。

如果要想实现电机的最大分辨率，则电机要牺牲最大转速。这时电机最大转速仅为 60 rpm。

如果要想实现最大转速，就要牺牲电机的分辨率。电机的分辨率下降 100 倍，只能设为 83886，即电子齿轮比设为 100。

而 A6 系列总线型驱动器的电机每转 1 圈的指令脉冲数可设为 1~4294967295。

若设电机每转 1 圈的指令脉冲数为 8388608，和编码器分辨率相同，即电子齿轮比为 1，电机最高转速为 6000rpm 时，相当于接收的指令脉冲频率为 839 MHz。

也就是说：只有总线型驱动器才能将 A6 系列的 23 位编码器的定位精度发挥到极致。其定

位精度比脉冲型电机的高 100 倍！

三）伺服电机及驱动器的硬件连接

以 100W 伺服电机 MSMF012L1U2M 及 EtherCAT 总线驱动器 MADLN05BE 为例。电机、驱动器及电源的接线图如图 3.1 所示。

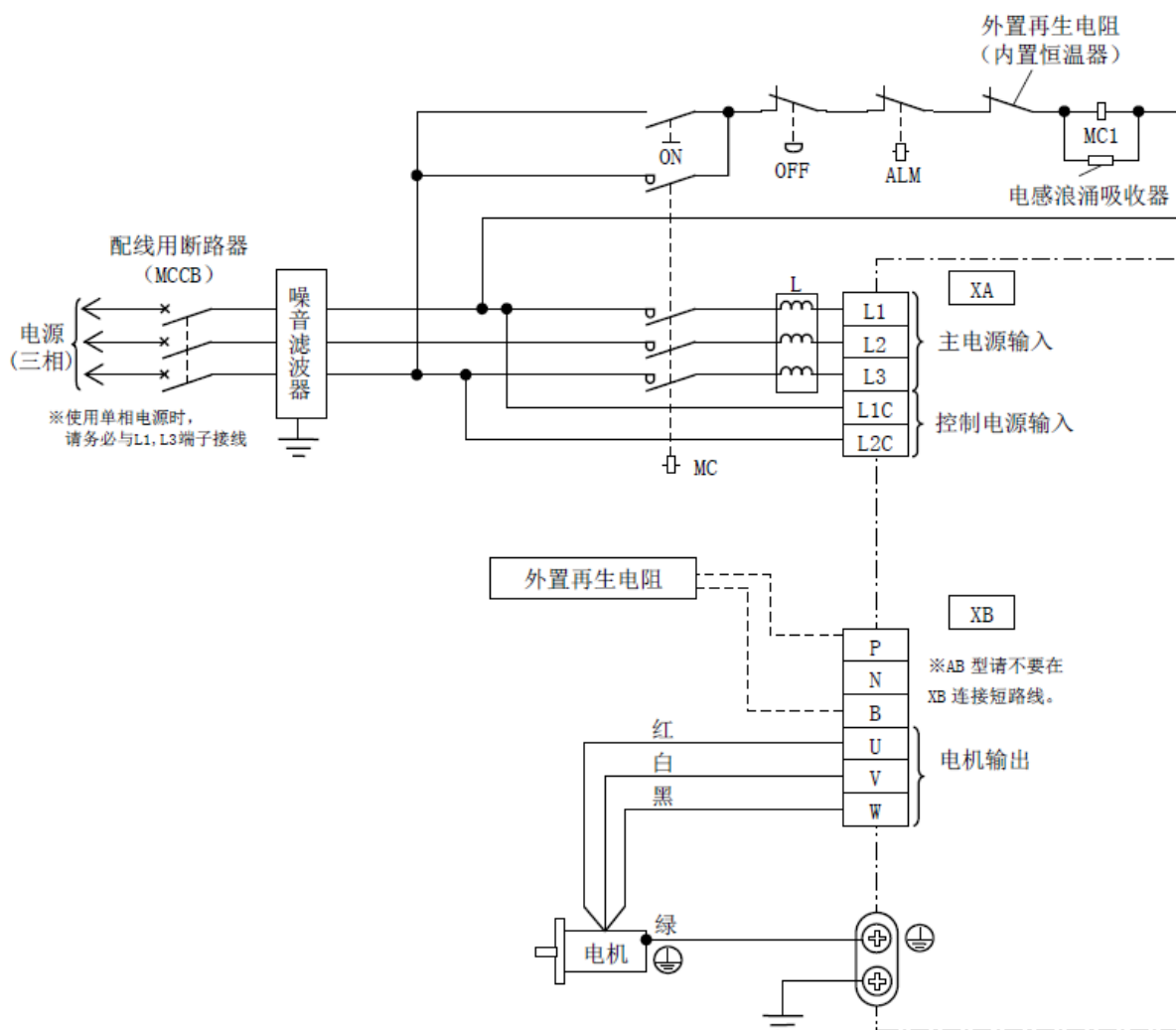


图 3.1. 电机、驱动器及电源接线图

驱动器上的 X4 为并行 I/O 连接器，是输入、输出信号接口。

连接器 X4 上有 3 个输出信号，如图 3.2 所示。出厂默认值分别是 S01 为刹车信号 BRK-OFF、S02 为通用输出信号 EX-OUT1、S03 为报警信号 ALM。这 3 个输入口也可以通过通过软件定义为其其他功能信号。

连接器 X4 上有 8 个输入信号。出厂默认值为：SI2 为正限位信号、SI3 为负限位信号、SI4 为原点信号。SI1、SI7、SI8 为通用输入信号，SI5、SI6 为外部锁存输入信号。

如果用信号的上升沿作为触发信号时，需要使用具有锁存补偿功能的输入口 SI5、SI6 或 SI7。为了提高信号响应速度，可将原点信号 Home 接入 SI5、正限位信号 POT 接入 SI6、负限位信号 NOT 接入 SI7，如图 3.2 所示。其他输入口可通过软件定义为通用输入信号或其他功能信号。

连接器 X4 的插头引脚编号如图 3.3 所示。

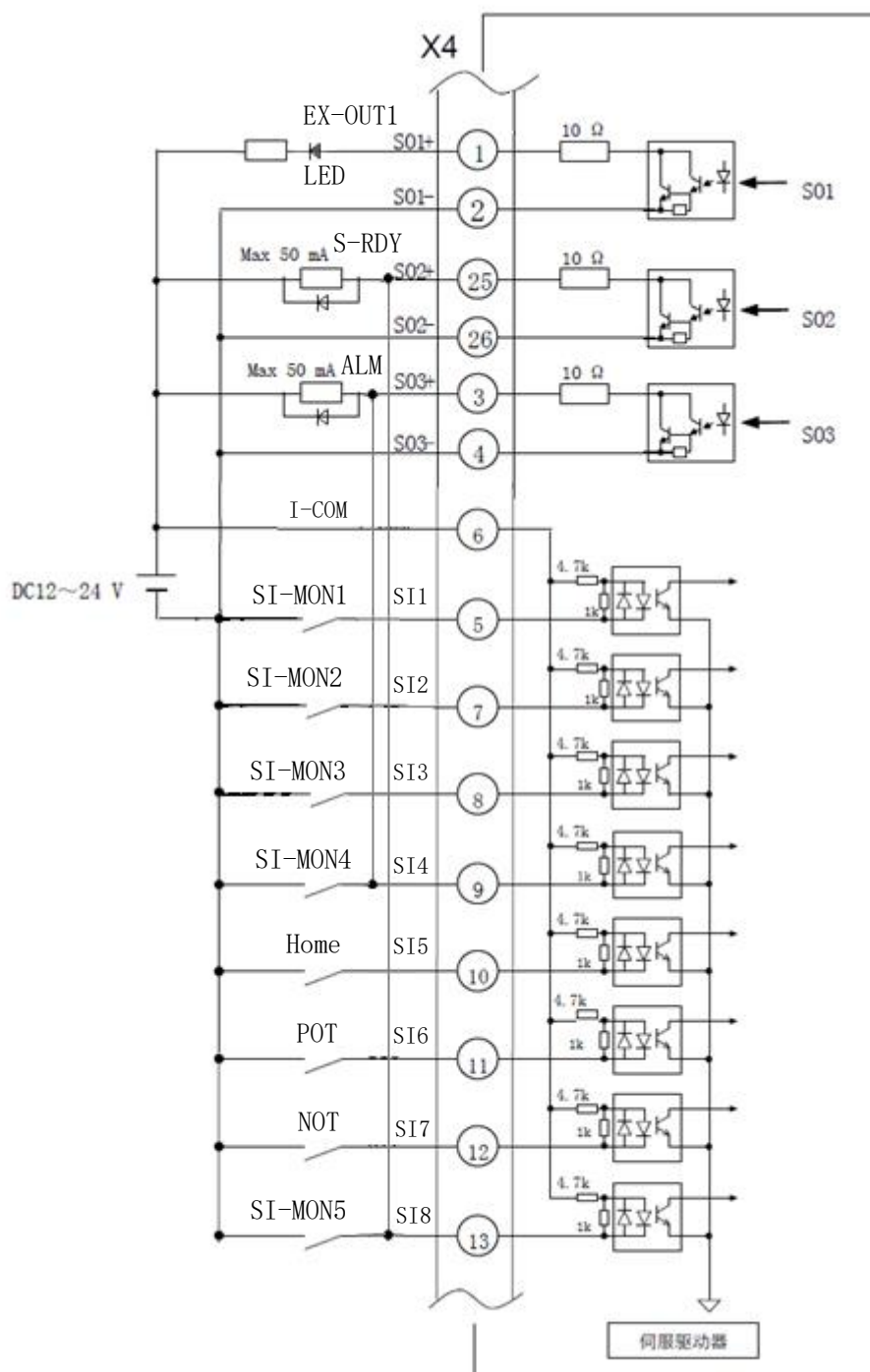


图 3.2. 数字输入输出信号接线图

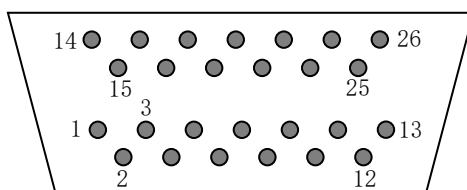


图 3.3. IO 插头引脚编号 (焊线端)

四) 伺服驱动器的参数设置

1. 安装软件

在松下伺服电机网页下载最新的电机调试软件 PANATERM Ver.6。(网址：<https://device.panasonic.cn/ac/c/motor/fa-motor/ac-servo/index.jsp>) 然后，安装软件 PANATERM。

2. 设置电子齿轮比

A6 系列 EtherCAT 总线型伺服驱动器的电子齿轮比的设置方式和脉冲型驱动器的不同，是通过修改 EtherCAT 总线的数据对象的参数完成设置。与电子齿轮比相关的数据对象定义及地址如下：

编码器增量：	608Fh-01h
电机转动圈数：	608Fh-02h
电机轴转动圈数：	6091h-01h
驱动轴转动圈数：	6091h-02h
进给量：	6092h-01h
驱动轴转动圈数：	6092h-02h

电子齿轮比的计算公式如下：

$$\text{电子齿轮比} = \frac{(\text{编码器增量}/\text{电机转动圈数}) \times (\text{电机轴转动圈数}/\text{驱动轴转动圈数})}{\text{进给量}/\text{驱动轴转动圈数}}$$

打开 PANATERM 软件，连接上驱动器后，在“其他”项中选“对象编辑器”，如图 3.4 所示，设置与电子齿轮比相关的数据对象为：

$$(608Fh-01h) = 2^{23}$$

$$(608Fh-02h) = (6091h-01h) = (6091h-02h) = (6092h-02h) = 1$$

$$(6092h-01h) = 2^{23}$$

即：电子齿轮比为 1。编码器的分辨率为 2^{23} /圈，电机转一圈需要 2^{23} 个脉冲。

注意：修改完参数后，需要点击“EEP”图标，将参数写入驱动器。

3. 设置编码器输出脉冲数

打开 PANATERM 软件中的“参数”项，将驱动器参数 Pr0.11（电机每旋转 1 圈输出脉冲数）设置为 2097152，即 2^{21} 。如图 3.5 所示。

因为在运动控制卡上编码器输入信号是 4 倍频，即得到的电机每旋转 1 圈输出脉冲数为 2^{23} 。

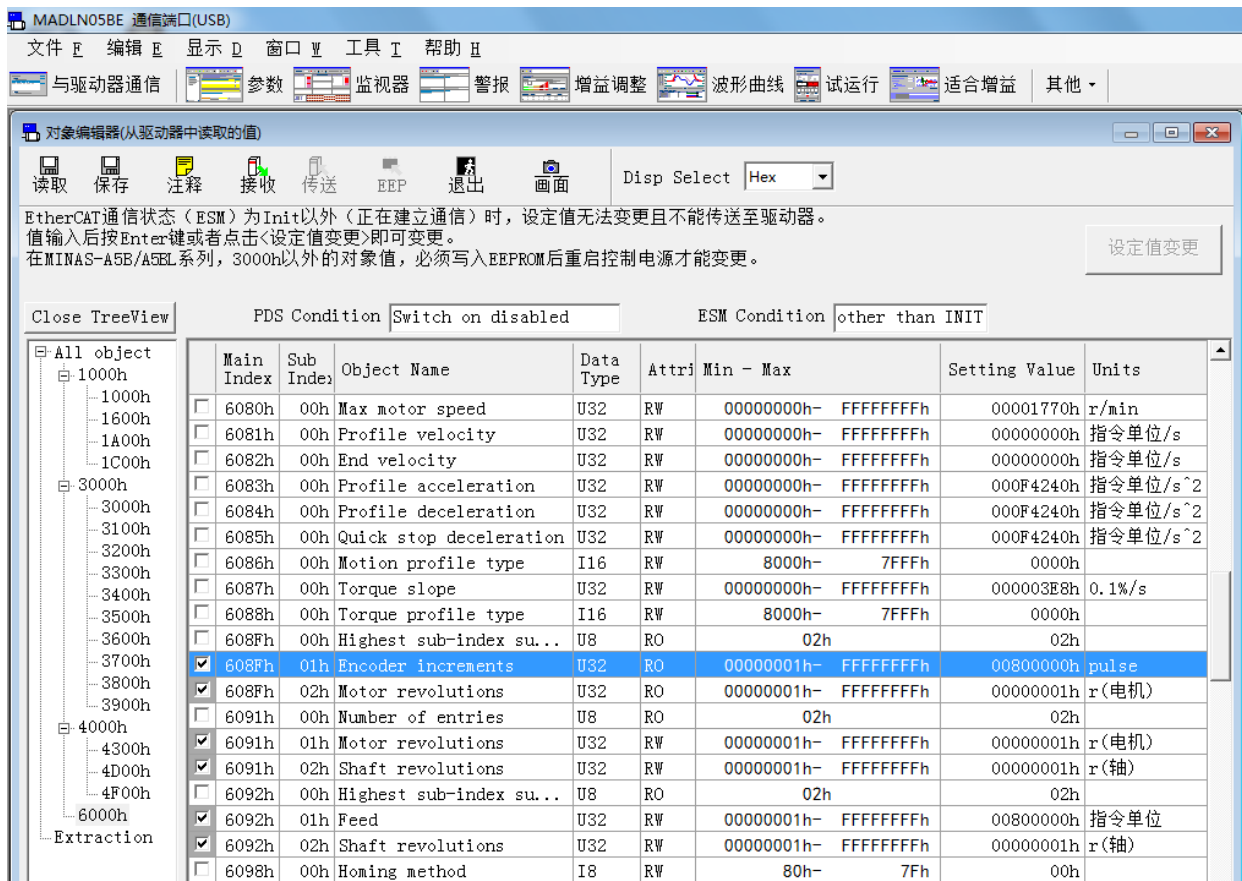


图 3.4. 设置伺服电机的数据对象



图 3.5. 设置伺服电机的参数

4. 设置定位完成范围 Pr4.31

该参数之间影响电机的定位精度。出厂设置为 10（指令脉冲数），对应的 Pr0.08 为 10000（电机每旋转 1 圈的指令脉冲数）。

因为我们已经将电机转一圈的脉冲数设为 2^{23} 个（8388608，比出厂值高了 838 倍）。故将 Pr4.31 设置为 50。定位精度比出厂设置值高了 167 倍。

另外，将参数 Pr4.32（定位完成输出设定）设为 1。即：无位置指令时，且位置偏差小于参数 Pr4.31 时，定位完成信号 INP 为 ON。

5. 自动调整电机的增益

每个电机的负载都不一样，为了使伺服电机工作在“快、准、稳”的最佳状态，必须调整电机的增益参数。使用 PANATERM 软件中的“适合增益”功能，可以自动调整电机的增益且效果良好。

表 3.2. 数字输入信号的定义表

信号名称	记号	设定值	
		常开	常闭
无效	-	00h	设定不可
正方向驱动禁止输入	POT	01h	81h
负方向驱动禁止输入	NOT	02h	82h
外部报警清除输入	A-CLR	04h	设定不可
强制报警输入	E-STOP	14h	94h
动态制动器 切换输入	DB-SEL	16h	设定不可
外部锁存输入 1	EXT1	20h	A0h
外部锁存输入 2	EXT2	21h	A1h
原点近傍输入	HOME	22h	A2h
回退动作输入	RET	27h	A7h
通用监视器输入 1	SI-MON1	2Eh	AEh
通用监视器输入 2	SI-MON2	2Fh	AFh
通用监视器输入 3	SI-MON3	30h	B0h
通用监视器输入 4	SI-MON4	31h	B1h
通用监视器输入 5	SI-MON5	32h	B2h

6. 设置输入信号

参照表 3.2 的定义，将连接器 X4 的 8 个输入信号的设置如表 3.3 所示。每个输入口的设定值有 3 个字节，对应于位置、速度、转矩这 3 种控制模式下的输入信号。

打开 PANATERM 软件中的“参数”项，将 Pr4.00~Pr4.07 设置为表 3.3 中所示值。

表 3.3. 连接器 X4 的 8 路输入信号的设置及参数值

输入口 名称	针脚号	对应 参数	设置值	设定状态					
				位置控制		速度控制		转矩控制	
				信号名	逻辑	信号名	逻辑	信号名	逻辑
SI1	5	Pr4.00	2E2E2Eh	SI-MON1	常开	SI-MON1	常开	SI-MON1	常开
SI2	7	Pr4.01	2F2F2Fh	SI-MON2	常开	SI-MON2	常开	SI-MON2	常开
SI3	8	Pr4.02	303030h	SI-MON3	常开	SI-MON3	常开	SI-MON3	常开
SI4	9	Pr4.03	313131h	SI-MON4	常开	SI-MON4	常开	SI-MON4	常开
SI5	10	Pr4.04	222222h	HOME	常开	HOME	常开	HOME	常开
SI6	11	Pr4.05	010101h	POT	常开	POT	常开	POT	常开
SI7	12	Pr4.06	020202h	NOT	常开	NOT	常开	NOT	常开
SI8	13	Pr4.07	323232h	SI-MON5	常开	SI-MON5	常开	SI-MON5	常开

5 个通用输入信号 SI-MON1~SI-MON5 映射至 EtherCAT 总线上的轴输入信号为 In19~In23。

原点信号 Home、正限位 POT、负限位 NOT 信号对应于轴输入信号 In2、In1、In0。

注意：定位完成信号 INP 对应于轴输入信号的 In24。

7. 设置输出信号

参照表 3.4 的定义，将连接器 X4 的 3 个输出信号的设置如表 3.5 所示。定位完成信号在速度和转矩模式下无效。

用 PANATERM 软件将 Pr4.10~Pr4.12 设置为表 3.5 中所示值。

表 3.4. 数字输出信号的定义表

信号名	记号	设定值
	外部输出	
无效	-	00h
报警输出	ALM	01h
伺服准备输出	S-RDY	02h
外部制动器解除信号	BRK-OFF	03h
定位完成	INP	04h
速度到达输出	AT-SPEED	05h
转矩限制中信号输出	TLC	06h
零速检出信号	ZSP	07h
速度一致输出	V-COIN	08h
警告输出 1	WARN1	09h
警告输出 2	WARN2	0Ah
位置指令有无输出	P-CMD	0Bh
定位完成 2	INP2	0Ch
速度限制中输出	V-LIMIT	0Dh
报警属性输出	ALM-ATB	0Eh
速度指令有无输出	V-CMD	0Fh
通用输出1	EX-OUT1	10h
set brake输出 *1)	set brake	11h
伺服使能开启状态输出	SRV-ST	12h
位置比较输出	CMP-OUT	14h
劣化诊断速度输出	V-DIAG	15h

表 3.5. 连接器 X4 的 8 路输出信号的定义及参数设置

输入口 名称	针脚号	对应 参数	设置值	设置状态		
				位置控制	速度控制	转矩控制
S01	1、2	Pr4.10	101010h	EX-OUT1	EX-OUT1	EX-OUT1
S02	25、26	Pr4.11	020202h	S-RDY	S-RDY	S-RDY
S03	3、4	Pr4.12	010101h	ALM	ALM	ALM

参数修改完后，点击“设定值变更”按钮，再点击“EEP”按钮，将参数存入伺服驱动器中。

8. 电机运动模式

众所周知，交流伺服电机有位置模式、速度模式和力矩模式。在使用 EtherCAT 总线伺服电机时，用户并不需要关注怎么设置电机的运动模式。运动控制器给伺服电机发什么指令，电机就在相应的模式下工作。

（注：力矩控制指令正在开发中。）

9. 电机的节点号

A6 系列伺服电机可以设置节点号。但是，雷赛 DMCE3032 运动控制卡是根据节点在总线上的物理位置确定其节点号，不识别 A6 系列伺服电机设置的节点号。

五) 用 Motion 软件测试伺服电机的方法

1. 启动 Motion 软件。可直接运行光盘中“Motion”子目录中的 Leadshine.DMCMotion.exe 文件。其主页如图 3.6 所示。如果右下角的“总线状态”栏提示总线错误，可用鼠标右键点击“DMC-E3032”图片，进入“硬件复位”，点击“冷复位”。可使总线恢复正常。

在没有添加任意一个总线设备时，总线状态会显示错误，但可直接进入下一步。



图 3.6. Motion 软件主页

2. 自动添加节点设备。在主页面上点击“总线配置”按钮，进入到如图 3.7 所示界面。

总线上的从站设备都接好并上电，然后右键点击主站，选择“扫描设备”进行自动扫描。如图 3.7 所示。扫描完成后，主站节点下会显示总线上所有的从站，如图 3.8 所示。

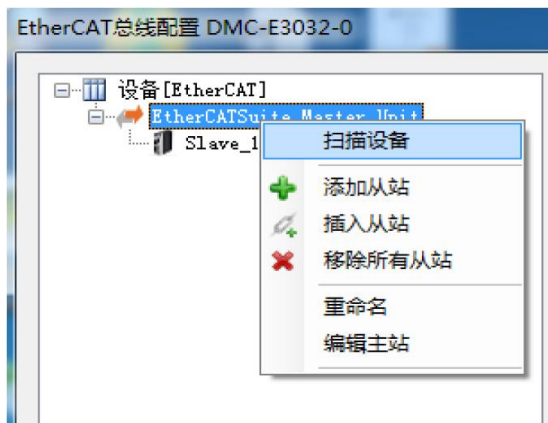


图 3.7. 自动扫描设备

然后，将总线周期时间设为 500 微秒。主站页面里的周期时间也改为 500 微秒。

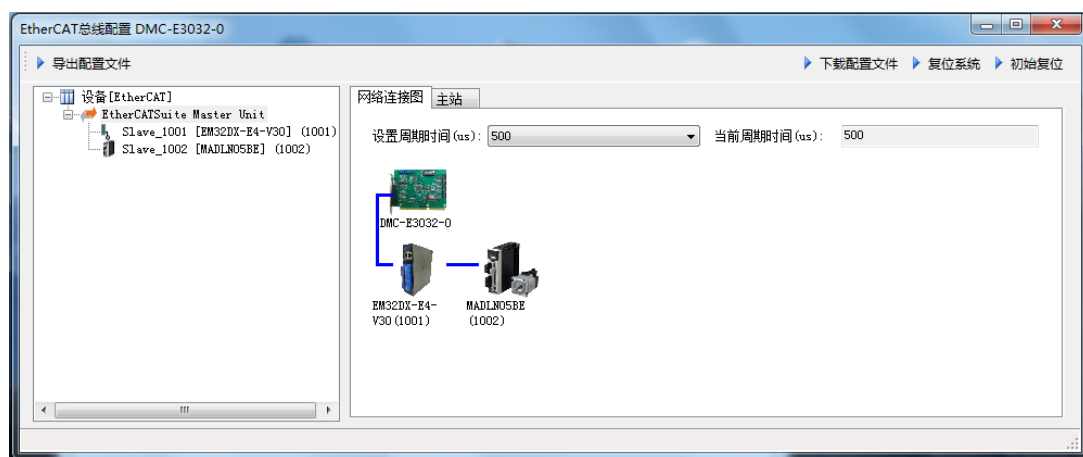


图 3.8. “总线配置” 页面

完成从站配置后，需要把配置文件下载到控制卡。点击“下载配置文件”完成配置文件下载；若软件未自动复位控制卡，需要手动点击“复位系统”来复位重启控制卡。

3. 测试输入信号。在主页面上点击“运动状态”按钮，进入到如图 3.9 所示界面。

在“专用信号状态”栏中，可以测试原点信号 ORG（对应于伺服电机上的 HOME 信号）、正限位信号 ELP（对应于伺服电机上的 POT 信号）和负限位信号 ELN（对应于伺服电机上的 NOT 信号）。

在如图 3.10 所示的“轴 IO 状态”栏中，可以测试输入信号 SI-MON1~5，它们对应于电机输入信号 In19~In23。

4. 输出信号的配置与测试。

在“总线配置”界面上，在左边的总线节点树上双击伺服电机驱动器“MADLN05BE”，看见如图 3.11 所示的界面。在“选择输出”栏中选中“Receive PDO mapping1”；然后，点击下方的“编辑”按钮。得到图 3.12 所示的 PDO 编辑界面。

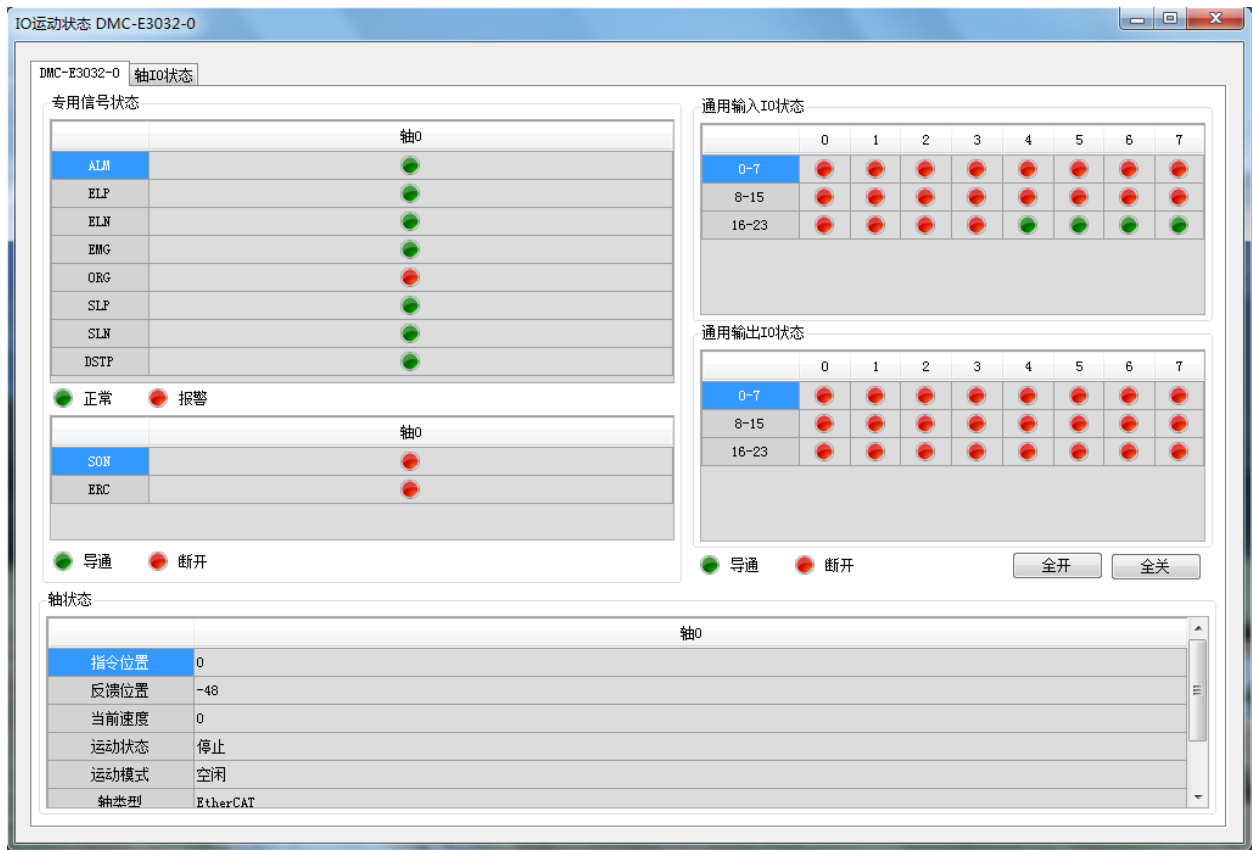


图 3.9. “专用信号状态” 页面

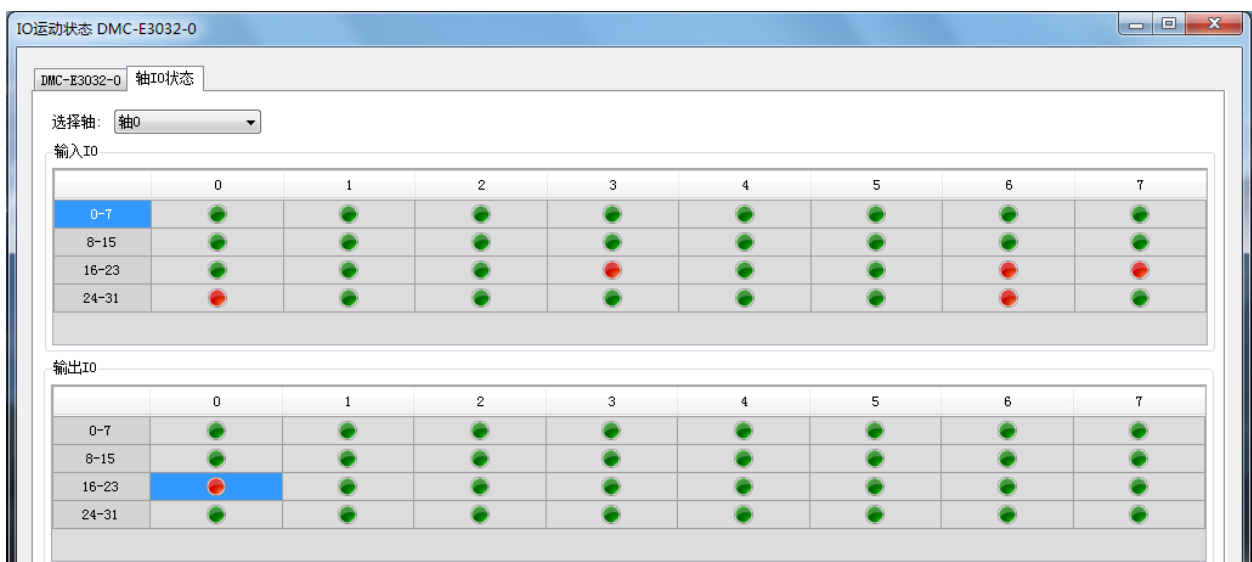


图 3.10. “轴 IO 状态” 页面

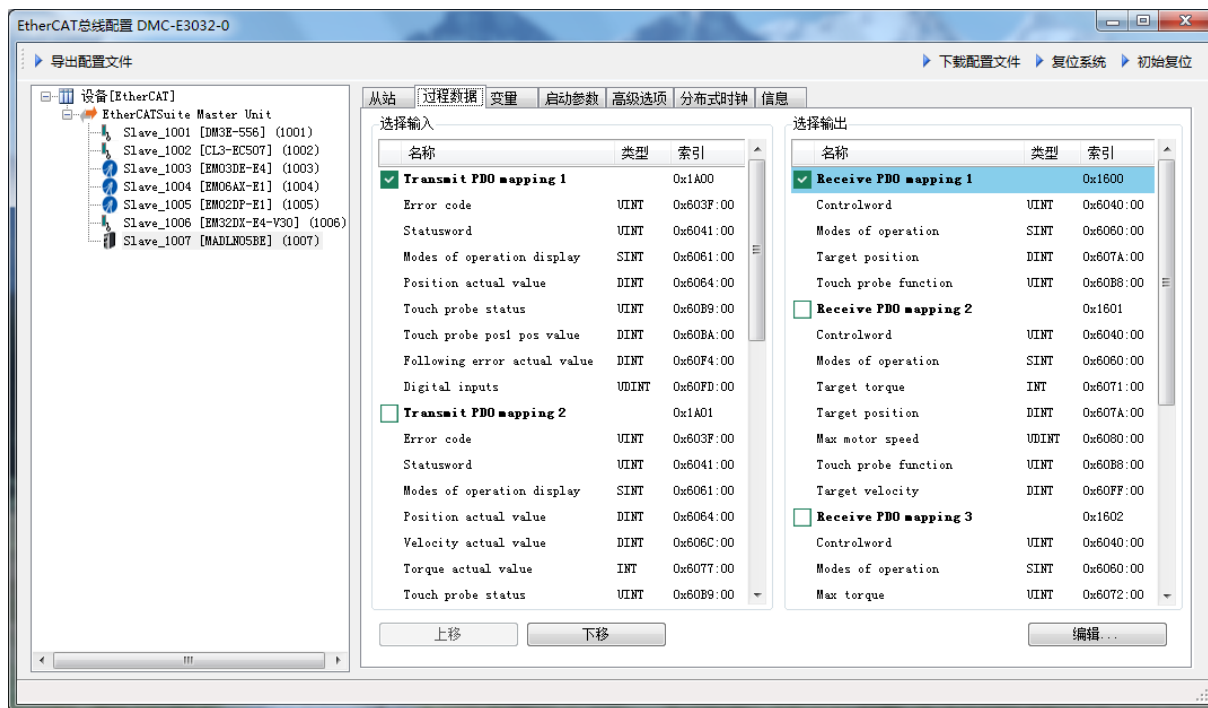


图 3.11. 从站的“过程数据”页面

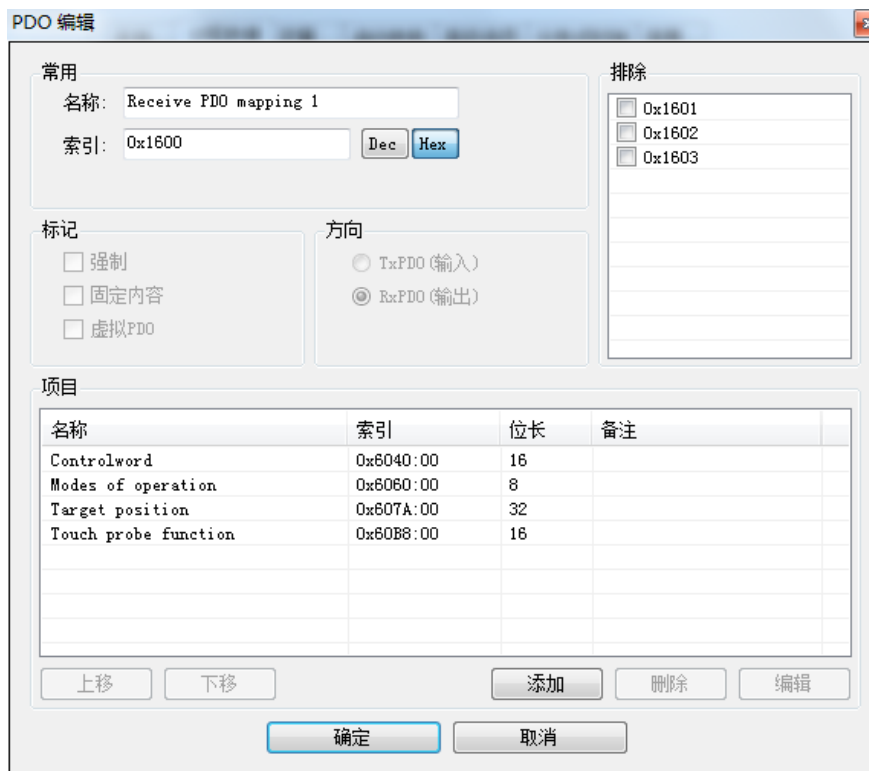


图 3.12. PDO 编辑界面

在“PDO 编辑”界面上，点击“添加”按钮，进入如图 3.13 所示“PDO 项编辑”界面。先后添加两个数据：PhysicalOutputs 和 Bit Mask，其地址和数据类型如图所示。点击“确定”，返回“过程数据”界面。这时，在“选择输出”栏中多出 2 个数据 PhysicalOutputs 和 Bit Mask。

下载配置文件后，在图 3.10 所示界面中可测试通用输出信号。EX-OUT1 对应于轴输出信号的 Out16。

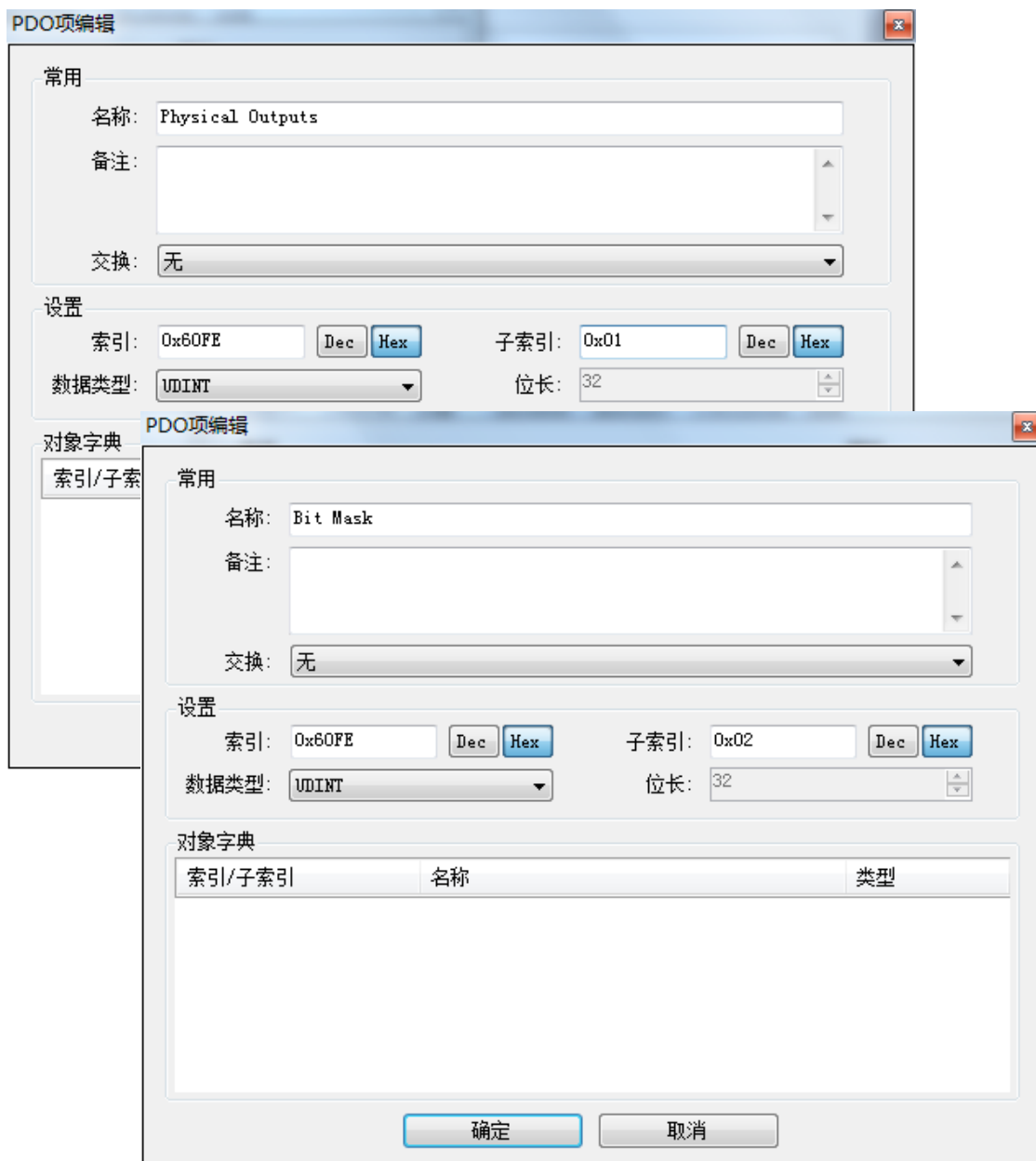


图 3.13. PDO 项编辑界面

5. 电机测试。在主页面上点击“功能测试”按钮，进入到如图 3.14 所示界面。在“单轴测试”栏中，进行电机测试。

注意：在点击“启动”按钮之前，必须点击“单轴使能开”按钮。



图 3.14. “单轴测试”页面

六) 用 C#编写程序控制松下伺服电机的方法

下面通过一个例程，演示 C#程序控制松下伺服电机的方法。

例程 1:

伺服电机的硬件连接及输入输出端的定义如图 3.1、3.2 所示，电机的参数设置如第三章第三节所示。

程序界面设计如图 3.15 所示，程序代码见附录中的例程 1。

该程序可实现的功能及编程要点如下：

1. 初始化运动控制卡

总线型运动控制卡的初始化过程与一般脉冲型运动控制卡的一样，代码详见附录例程 1 中的函数 Form1_Load(...)。

在伺服电机运动之前要将使能信号打开。其指令为：nmc_set_axis_enable(卡号, 轴号)；注意：不要忘记将 LTDMC.cs 拷贝至项目文件夹中，并在“解决方案”中添加 LTDMC.cs；并且要将 DMC2410.dll 拷贝至项目文件夹中的 bin 文件夹中的 Debug 文件夹内。

2. 关闭运动控制卡

关闭总线型运动控制卡的过程与一般脉冲型运动控制卡的一样，代码详见附录例程 1 中的函数 Form1_FormClosing(...)。

关闭伺服电机的使能信号指令为：nmc_set_axis_disable(卡号, 轴号)；

3. 点位运动

点位运动的指令与一般脉冲型运动控制卡的一样，代码详见附录例程 1 中的函数 button5_Click(...)。



图 3.15. 松下伺服电机测试程序界面

4. 连续正反转运动

伺服电机定位精度高。应该在运动指令发送结束后，再判断其到位信号 INP，以确定电机运动是否停止。代码详见附录例程 1 中的函数 Check_INP(…)

读取电机的输入口状态的指令为：nmc_get_axis_io_in(卡号, 轴号); 它一次将该电机所有的输入信号读入一个无符号 32 位变量中。判断到位信号 INP，还需要进行一次与运算，将第 24 位的到位信号 INP 解析出来，如以下代码所示。

```
inINP = LTDMC.nmc_get_axis_io_in(cardID0, 0);
inINP = inINP & 0x1000000;    // 读取第 24 位的到位信号 INP
```

5. Jog 运动

Jog 运动即为定速运动。按钮“Jog+”按下后，启动 vmove 指令，电机定速运动；按钮“Jog+”抬起后，启动减速停止指令，使定速运动停止。代码详见附录例程 1 中的函数 button1_MouseDown()、button1_MouseUp(…)

6. 定速运动及在线变速

按下“连续运动开始”按钮，电机以设定的速度连续运动。之后，每按一次“速度+10%”或“速度-10%”，电机都会进行在线变速。

在线变速的指令为：dmc_change_speed_unit(卡号, 轴号, 新速度, 变速调整时间);

7. 回原点运动

回原点运动和一般脉冲型运动控制卡的不一样。回零模式是按照 IEC61800-7CiA402 标准

协议定义的。常用的 4 种如下：

19 号回零方式：电机正方向旋转高速回原点，检测到原点信号后，低速后退，出零点检测范围后停止。

21 号回零方式：电机负方向旋转高速回原点，检测到原点信号后，低速后退，出零点检测范围后停止。

20 号回零方式：电机正方向旋转低速回原点，检测到原点信号即停止。

22 号回零方式：电机负方向旋转低速回原点，检测到原点信号即停止。

即在调用回原点模式设置函数 `nmc_set_home_profile`(卡号, 轴号, 回零模式, 回零低速, 回零高速, 回零加速时间, 回零减速时间, 回零偏移量) 时, 其中的“回零模式”要设为 19 或 21、20、22。

回原点运动的代码详见附录例程 1 中的函数 `button12_Click(...)`。

8. 输出口 Output1 的控制

电机输出口的控制指令和输入口的一样，也是一次将该电机所有的输出信号读出或写入。因此，在控制输出口 Output1 时，就要对输出口数据的第 16 位信号进行处理。置 1 用或运算、清 0 用与运算。代码详见附录例程 1 中的函数 `checkBox6_CheckedChanged(...)`。

9. 查询输入口状态

输入口状态由定时器定时查询其变化并显示其状态，定时周期 50 毫秒。代码附录例程 1 中的详见函数 `timer2_Tick(...)`。

10. 编码器位置显示与清零

编码器位置也是由定时器定时查询与显示，代码详见附录例程 1 中的函数 `timer2_Tick(...)`。

编码器位置清零的代码详见附录例程 1 中的函数 `button7_Click (...)`。

11. 总线状态显示与总线复位

总线状态也是由定时器定时查询与显示，代码详见函数 `timer2_Tick(...)`。

当总线状态异常时，可以使用指令 `dmc_cool_reset(0)` 对总线复位，同时关闭运动控制卡；复位过程需要 10 秒钟时间；总线复位完成后，需要对运动控制卡初始化，直接调用函数 `Form1_Load(sender, e)` 即可。代码详见附录例程 1 中的函数 `button8_Click(...)`。

四. 雷赛 EtherCAT 总线型步进电机驱动器

在 EtherCAT 总线型运动控制系统中，全部使用伺服电机显然是不合适的。

2016 年雷赛智能公司在国内首先推出了 EtherCAT 总线型步进电机驱动器 DM3E 系列产品。其外观与参数如图 4.1 和表 4.1 所示；其总线参数如表 4.2 所示。

其接口电路图如图 4.2 所示，其中 CN3 为 IO 接口。IO 接口的默认信号如图 4.2 所示，也可以更改为其他信号，详见该产品手册。



图 4.1. 雷赛 EtherCAT 步进电机驱动器外观

表 4.1. DM3E 系列驱动器参数

型号	相数	电流(A)	电压(V)	输入信号	输出信号	适配电机	外形尺寸
DM3E-522	2	0.3~2.2	20~50	4 路单端、1 路差分	2 路单端	20/28/35/42	118×90×34
DM3E-556	2	1.0~5.6	20~50	4 路单端、1 路差分	2 路单端	42/57/60/86	
DM3E-870	2	2.1~7.0	20~80	4 路单端、1 路差分	2 路单端	57/60/86	

表 4.2. DM3E 系列驱动器总线参数

项目	描述
物理层	100BASE-TX
通信连接器	RJ45 × 2
网络拓扑结构	总线型
波特率	100 Mbps (全双工)
通信对象	SDO: 服务数据对象 PDO: 过程数据对象 EMCY: 紧急事件
通讯协议标准	CoE (CANopen over EtherCAT)
设备协议标准	IEC61800-7 CiA 402 Drive Profile
控制模式	CSP (Cyclic Synchronous Position) 循环同步模式 PP (Profile Position) 协议位置模式 PV (Profile Velocity) 协议速度模式 HM (Homing) 回原点模式
同步模式	DC Synchronization 及 FreeRun 模式
循环周期	250us、500us、750us、1ms、2ms、4ms

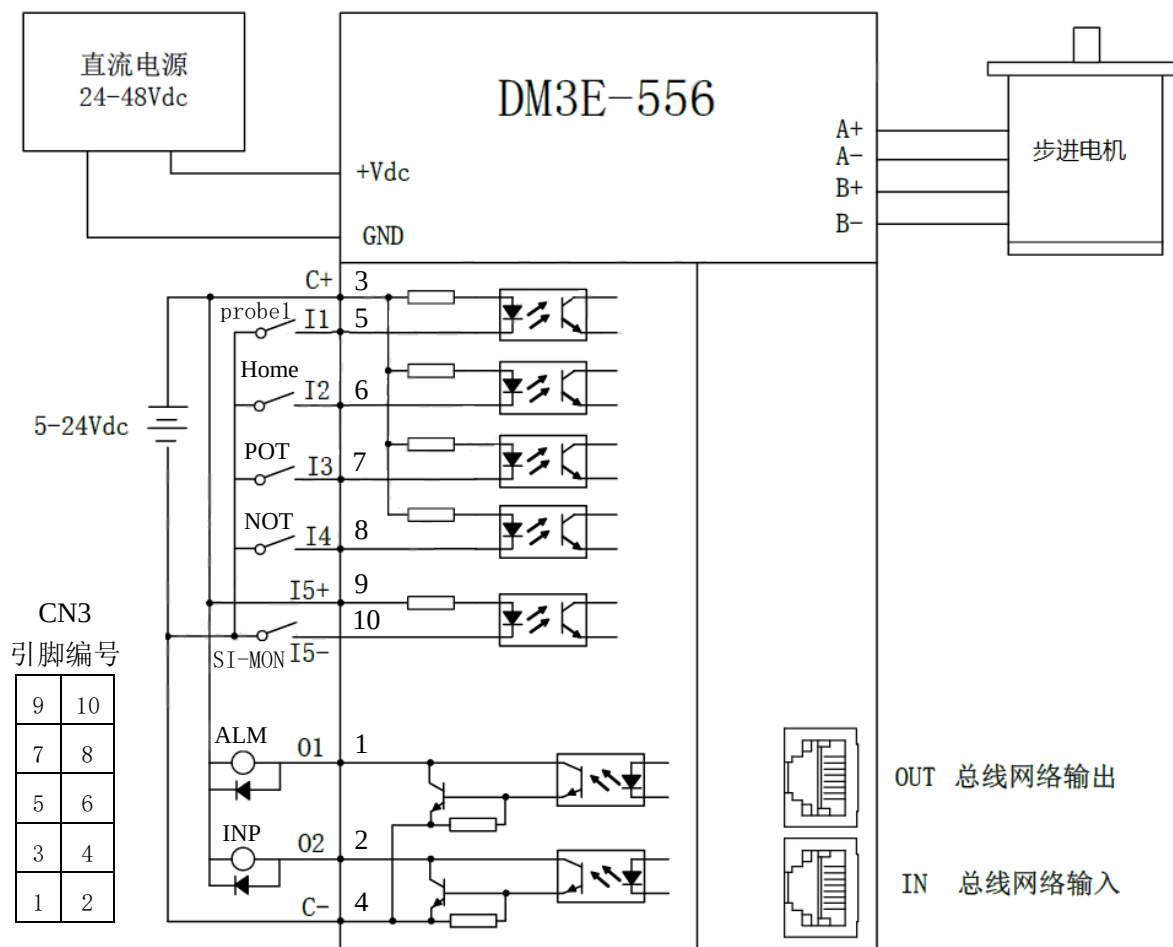


图 4.2. DM3E 系列驱动器接口电路

DM3E 系列驱动器接入总线的方法很简单，和松下交流伺服电机的相似。用雷赛 Motion 软件，在总线配置界面内，通过“扫描设备”、设置总线周期时间、下载配置文件，即自动配置完成；然后可进行电机的相关测试。（参见第三章第 5 节）

步进电机的峰值电流、每转脉冲数（相当于设置细分数）等参数可以通过 EtherCAT 总线修改 DM3E 系列驱动器的对象字典完成。常用参数如表 4.3 所示。

表 4.3. DM3E 系列驱动器常用参数

对象字典地址	名称	属性	类型	范围	默认值	单位	备注
2000H	峰值电流	R/W	UINT	100~7000	1000	mA	使能关闭后 修改参数
2001H	电机每转脉冲数	R/W	UINT	6400~51200	50000	Pulse	
2002H	全电流待机时间	R/W	UINT	10~3000	50	ms	
2003H	待机电流百分比	R/W	UINT	0~100	50	%	
2015H	电机运行方向	R/W	UINT	0~1	0		
2048H	母线电压	R	UINT	0~32767		V	
1010H-04H	保用户参数	R/W	UDINT	0~0xffffffff	0		保存命令： 0x65766173

注意：修改参数后，必须向对象字典（1010H-04H）写一个保存命令 0x65766173，参数才能写入 EEPROM，保留在驱动器中。保存数据的时间大约为 10 秒钟。

例程 2：

图 4.3 是一个 DM3E 系列驱动器及步进电机的测试程序界面程序，代码见附录中的例程 2。该程序可实现的功能及编程要点如下：



图 4.3. DM3E 系列驱动器及步进电机测试程序界面

1. 初始化运动控制卡、关闭运动控制卡

代码详见附录例程 2 中的函数 Form1_Load(...)、函数 Form1_FormClosing(...)。

2. 参数的读取与设置

读对象字典参数的指令为：

LTDMC.nmc_get_node_od(卡号, EtherCAT 端口号, 节点号, 对象字典索引, 子索引, 参数长度, ref 参数);

写对象字典参数的指令为：

LTDMC.nmc_set_node_od(卡号, EtherCAT 端口号, 节点号, 对象字典索引, 子索引, 参数长度, 参数);

代码详见附录例程 2 中的函数 button1_Click(...)、函数 button2_Click(...)。

注意：EtherCAT 端口号固定为 2。

3. 点位运动、Jog 运动和回原点运动

运动指令和例程 1 一样。代码详见附录例程 2 中的函数 button3_Click(...)、button5_MouseDown(...)、button5_MouseUp(...)、button4_Click(...)。

4. 读指令脉冲位置

其指令为：LTDMC.dmc_get_position_unit(卡号, 轴号, ref 指令位置);

代码详见附录例程 2 中的函数 timer1_Tick(...)。

五. 雷赛 EtherCAT 总线型闭环步进电机驱动器

雷赛公司于 2010 年在国内首创的闭环步进电机控制技术是在普通的步进电机上加装旋转编码器、并使用交流伺服电机控制算法，使步进电机的性能有了质的飞跃。雷赛闭环步进电机及驱动器的特点如下：

1. 步进电机采用 1000 线编码器做位置检测；也可定制 5000 线编码器。
2. 对位置偏差实时补偿，根本解决了普通步进电机的丢步问题。
3. 转速可达 2500rpm，有效力矩比普通步进电机提高 30%以上。
4. 振动小、噪声低，电机运行平稳。
5. 电机电流根据负载大小实时调节，所以电机发热量小。
6. 电机参数自动调整，智能水平高。

雷赛 EtherCAT 总线型闭环步进电机驱动器 CL3-EC 系列产品的参数如表 5.1 所示。

表 5.1. 雷赛 EtherCAT 总线型闭环步进电机驱动器参数

型号	电流(A)	电压	适配电机	外形尺寸
CL3-EC503	1~3	20~50Vdc	28/42	130×91×34
CL3-EC507	3~7	20~50Vdc	57/60	
CL3-EC808	4~8	20~80Vdc	86	151×113×40
CL3-EC808AC	4~8	20~80Vac	86	



图 5.1. 雷赛 CL3-EC 系列驱动器外形

CL3-EC 系列驱动器总线参数与雷赛 EtherCAT 步进电机驱动器一样，参见表 4.2。

CL3-EC 系列驱动器的接口电路图如图 5.2 所示。图中 CN4 接口为数字 I/O 接口，有 7 路输入信号、6 路输出信号；信号可自定义，出厂默认值如图所示。CN4 接口上还有一个专用的抱闸输出口，驱动电流高达 500mA，并集成了续流二极管，可以不用继电器直接驱动抱闸器。

CL3-EC 系列驱动器驱动器接入总线的方法和雷赛总线型步进电机的相似。下面以一个 CL3-EC507 驱动器为例，介绍其使用方法。

首先，用雷赛 Motion 软件，在总线配置界面内，通过“扫描设备”、设置总线周期时间，可自动找到 CL3 系列闭环步进电机驱动器，但没有自动完成轴映射，如图 5.3 所示。

在图 5.3 所示的界面中，点击“添加”按钮，手工进行轴映射。过程如图 5.4 所示。再点击图 5.5 中的“上移”按钮，将刚添加的“轴 3”上移为“轴 1”，如图 5.5 所示。

然后下载配置文件，完成配置工作。

至此，图 5.5 中的原点信号 Home，正负限位信号 POT、NOT，到位信号 INP、报警信号 ALM，通用输入信号 SI-MON1、SI-MON2 都可正常使用。

如果要使用图 5.5 中的通用输出口 EX-OUT1~EX-OUT4，还需要进行输出口过程数据的设置，方法如下。

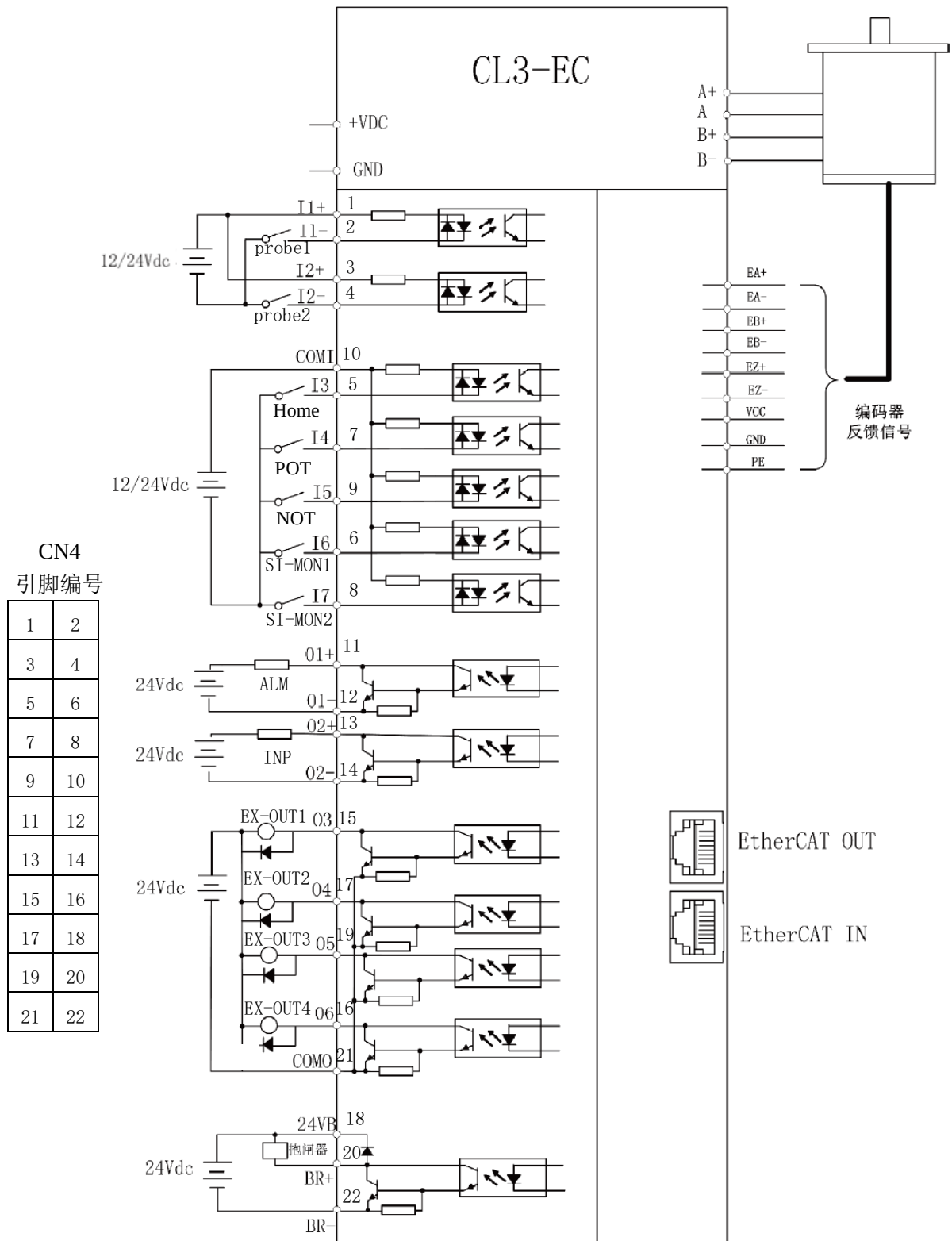


图 5.2. 雷赛 CL3-EC 系列闭环步进电机驱动器接口电路图

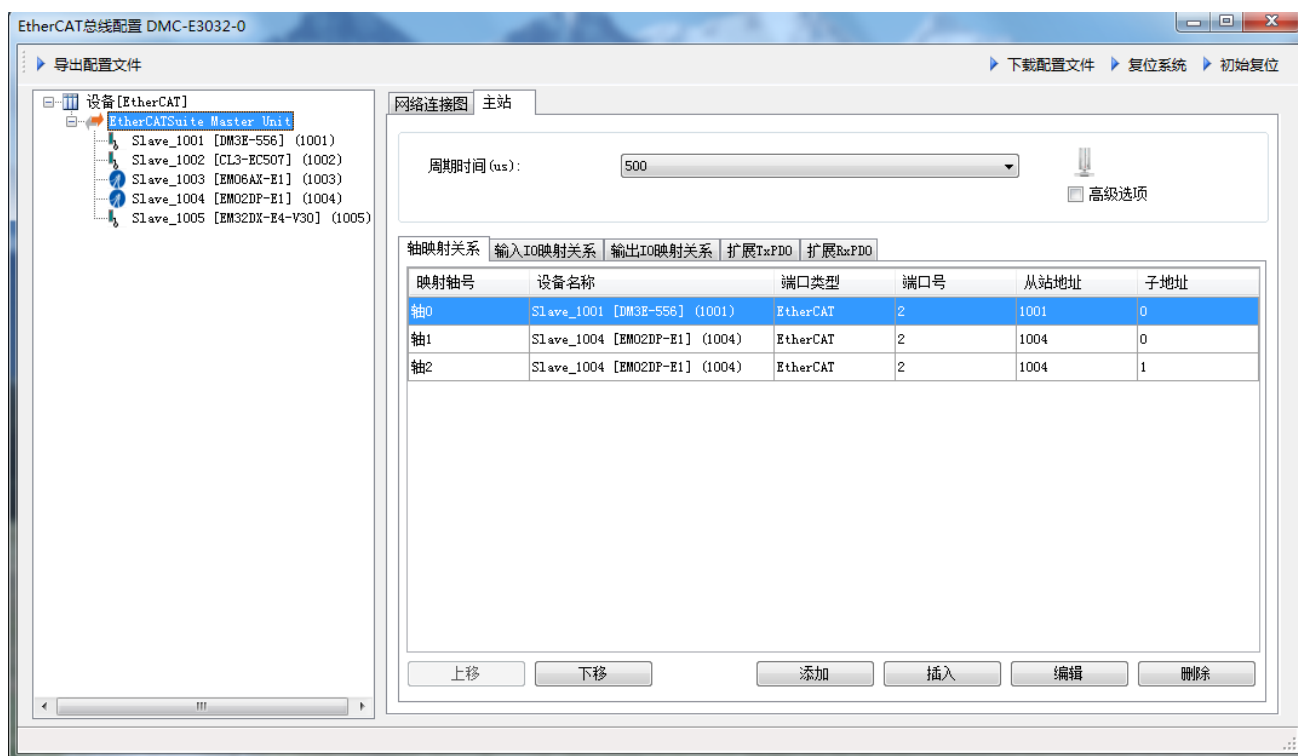


图 5.3. 扫描设备后，自动找到 CL3 系列闭环步进电机驱动器

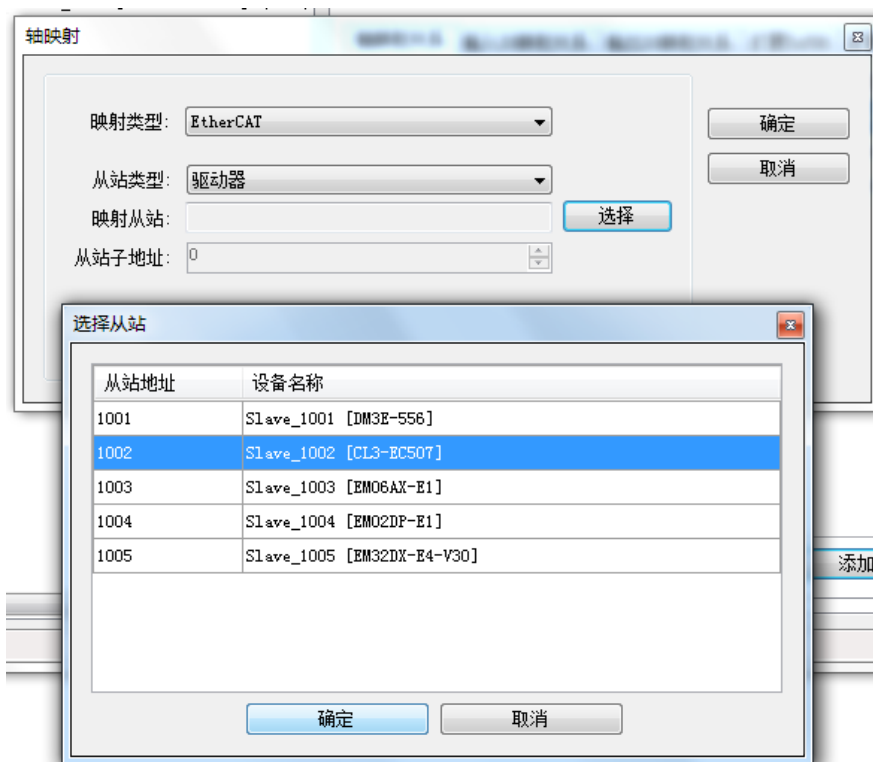


图 5.4. 手工进行轴映射

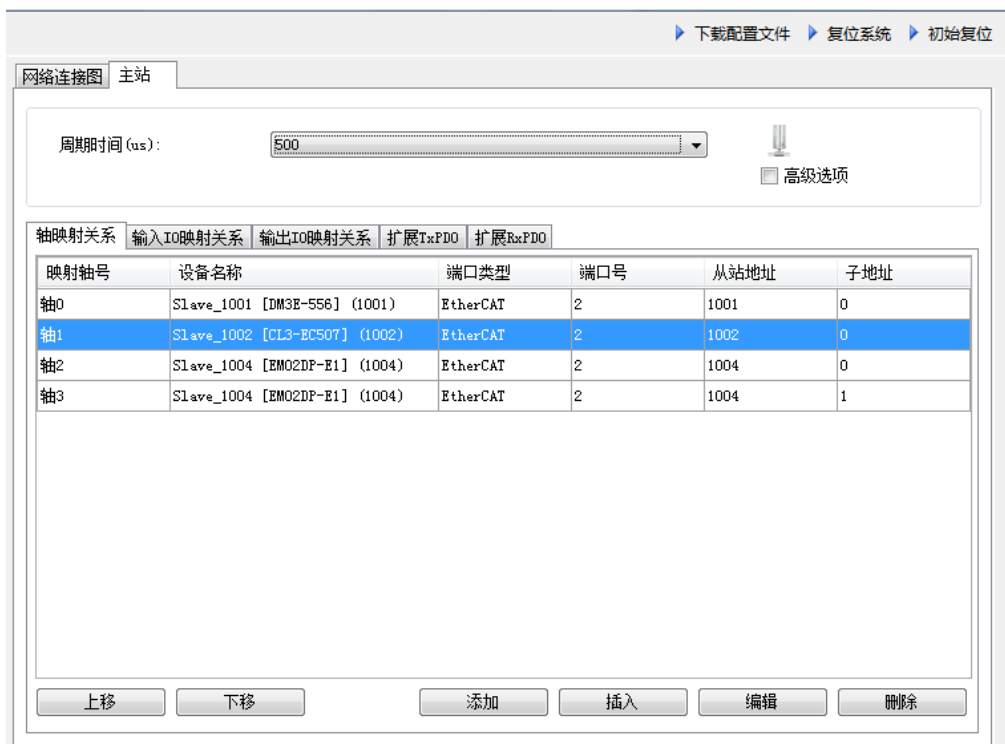


图 5.5. 完成轴映射

在 Motion 软件的“总线配置”界面左边的 EtherCAT 设备树中，双击“CL3-EC507”，出现图 5.6 所示的界面。

选中“ReceivePDO1”，点击下面的“编辑”按钮，进入到图 5.7 所示界面。点击“添加”按钮，进入如图 5.8 所示的界面。

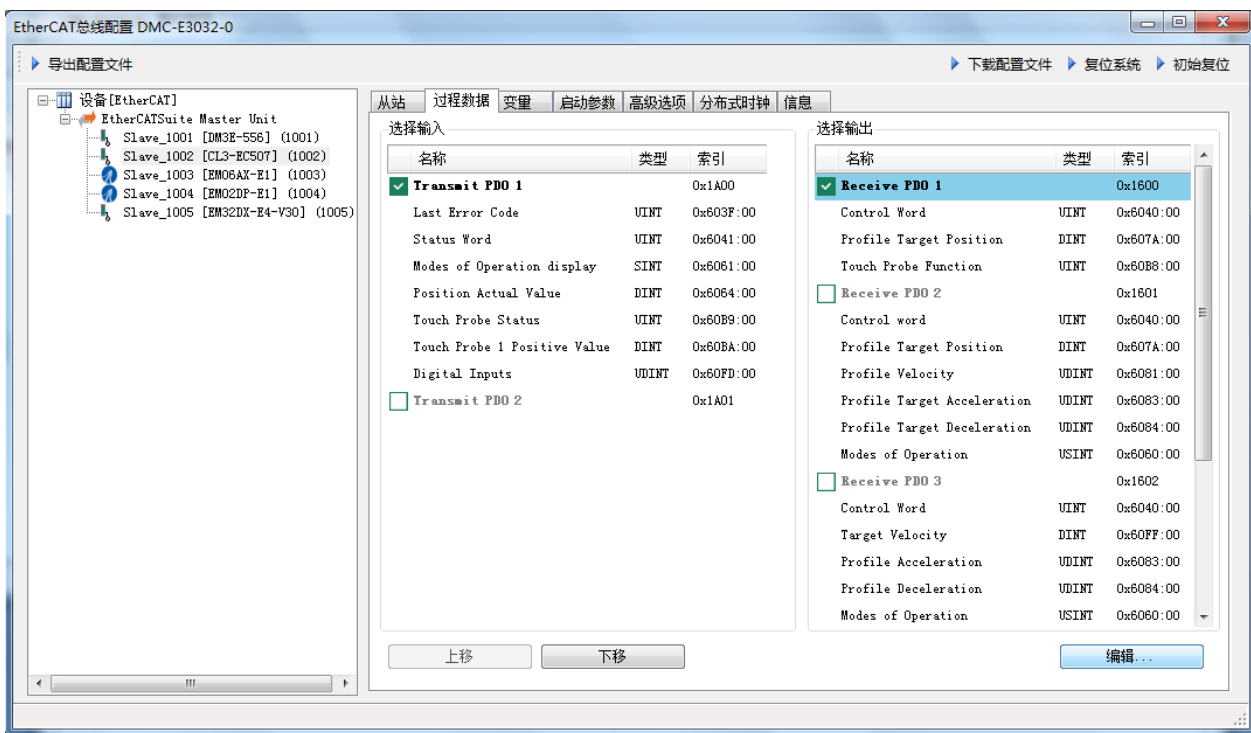


图 5.6. 打开从站的“过程数据”界面

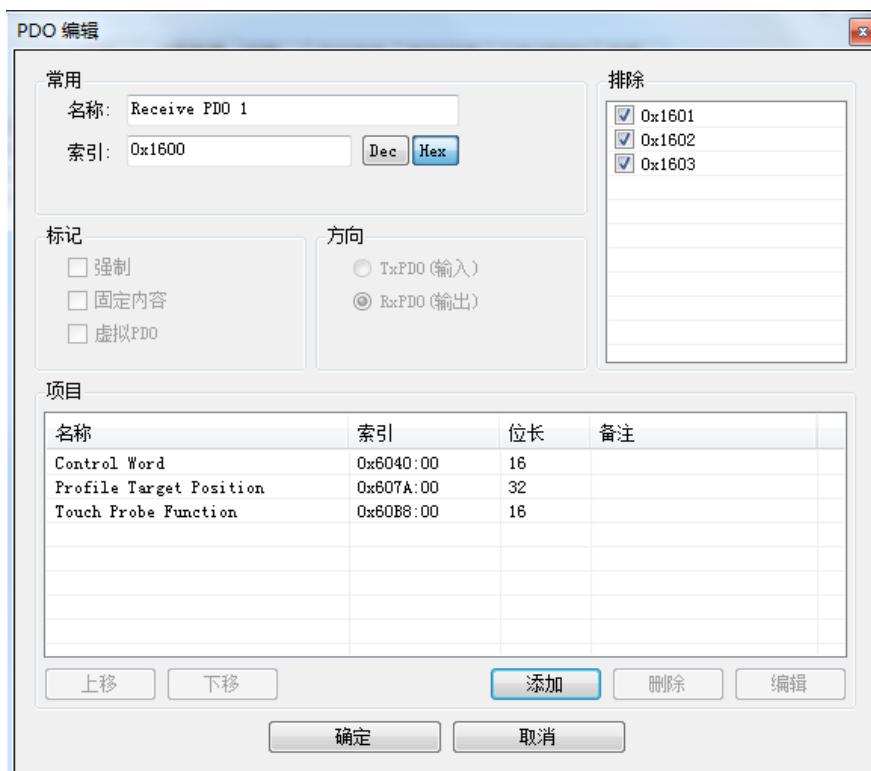


图 5.7. PDO 编辑界面

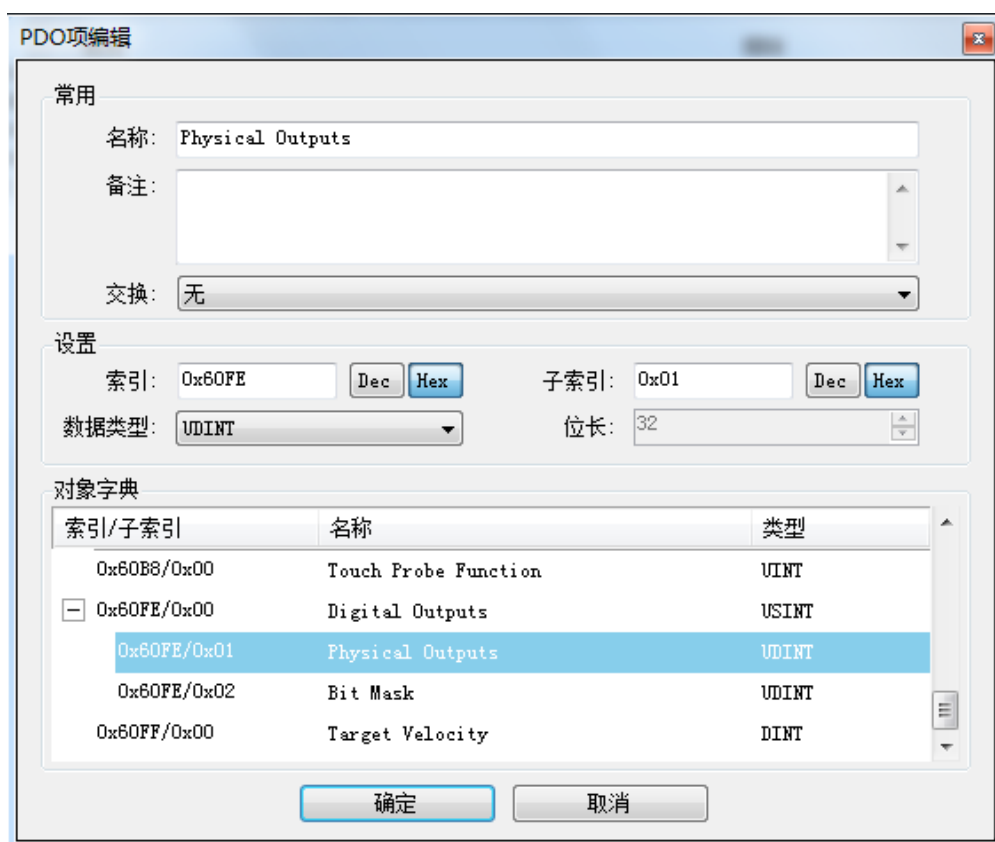


图 5.8. 添加 Physical Outputs 和 BitMask

在图 5.8 所示的界面中，选择“DigitalOutputs”，分别添加 Physical Outputs 和 BitMask 这两个数据；之后，可在输出数据中看到 Physical Outputs 和 BitMask，如图 5.9 所示。最后，还要下载配置文件。

CL3-EC 系列驱动器的通用 IO 信号与 EtherCAT 总线轴 IO 的映射关系如表 5.2 所示。控制通用 IO 信号的指令和松下交流伺服电机例程中的相同。

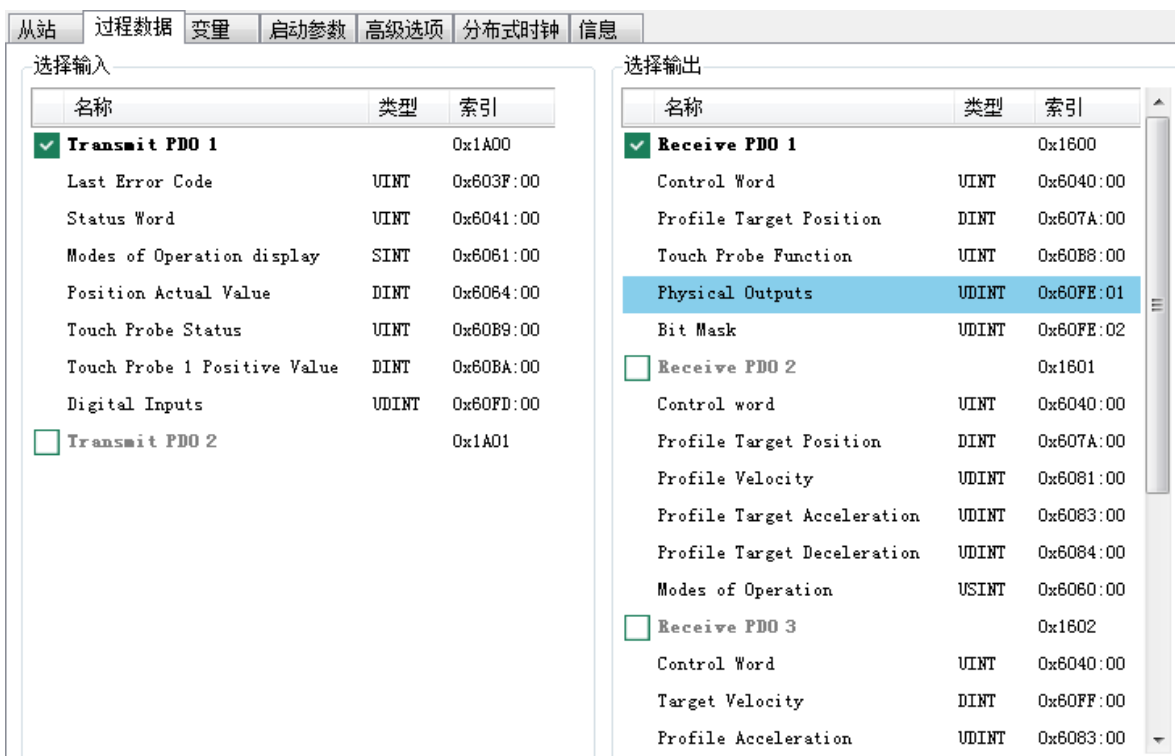


图 5.9. 完成输出口数据设置

表 5.2. CL3-EC 的通用 IO 信号与总线轴 IO 的映射关系

驱动器上的 IO 信号名称	轴 IO 信号的位号
SI-MON1	IN9
SI-MON2	IN10
EX-OUT1	OUT18
EX-OUT2	OUT19
EX-OUT3	OUT20
EX-OUT4	OUT21

闭环步进电机的峰值电流、每转脉冲数（相当于设置细分数）等参数可以通过 EtherCAT 总线修改 CL3-EC 系列驱动器的对象字典完成。常用参数如表 5.3 所示。

注意：编码器分辨率是编码器线数的 4 倍。如果，采用 5000 线的编码器，这编码器分辨率为 20000。和松下交流伺服电机不同，编码器分辨率不可随便改动。修改编码器分辨率后，必须重新启动驱动器才能生效。

例程 3:

图 5. 10 是 CL3-EC 系列驱动器及闭环步进电机测试程序的界面。其代码见附录中的例程 3。

表 5. 3. CL3-EC 系列驱动器的常用参数

对象字典地址	名称	属性	类型	范围	默认值	单位	备注
2000H	峰值电流	R/W	UINT	0~80	60	0.1A	电机使能关闭 后修改参数
2001H	电机每转脉冲数	R/W	UINT	200~51200	10000	Pulse	
2029H	编码器分辨率	R/W	UINT	200~512000	4000	Pulse	
2048H	母线电压	R	UINT	0~65535		0.1V	
1010H-04H	保存用户参数	R/W	UDINT	0~0xffffffff	0		保存命令： 0x65766173

图 5. 10. 闭环步进电机及驱动器的测试程序界面

六. EtherCAT 总线扩展模块

为了便于客户组建 EtherCAT 总线型运动控制系统，雷赛控制公司还研发了多种功能的 EtherCAT 总线扩展模块。如：IO 扩展模块用于 IO 信号的检测与控制；定位模块用于控制脉冲型电机；AD/DA 扩展模块用于模拟信号的采集与控制；编码器扩展模块用于编码器信号检测。

一) IO 扩展模块

EM32DX-E4 模块可控制 16 路输入和 16 路输出信号。IO 接口采用了光电隔离和滤波电路，抗干扰性能好。EM32DX-E4 模块的外形如图 6.1 所示，其接口定义如表 6.1 所示，其接口电路与雷赛公司的 IO 控制卡相同，其参数如表 6.2 所示。



图 6.1. EM32DX-E4 模块外形

表 6.1. IO 接口定义

序号	功能	序号	功能
1	IN0	2	OUT0
3	IN1	4	OUT1
5	IN2	6	OUT2
7	IN3	8	OUT3
9	IN4	10	OUT4
11	IN5	12	OUT5
13	IN6	14	OUT6
15	IN7	16	OUT7
17	GND	18	GND
19	IN8	20	OUT8
21	IN9	22	OUT9
23	IN10	24	OUT10
25	IN11	26	OUT11
27	IN12	28	OUT12
29	IN13	30	OUT13
31	IN14	32	OUT14
33	IN15	34	OUT15
35	GND	36	GND

表 6.2. IO 接口参数

输入特性		输出特性	
输入类型	低电平输入有效	输出类型	漏型输出，低电平有效
输入电压	21~27V DC	负载电压	21~27V DC
额定输入电压	24V DC	输出电流	300mA/通道
最大连续电压	30V DC	漏电流	最大 8uA/通道
浪涌	35V DC, 500ms	浪涌电流	2A, 100ms
导通电流	≥4.2mA(15V) 典型值 6.9 mA(24V)		
关断电流	≤1.2mA(5V)		
浪涌	35V DC, 500ms	浪涌电流	2A, 100ms
导通电流	≥4.2mA(15V) 典型值 6.9 mA(24V)		
关断电流	≤1.2mA(5V)		

EM32DX-E4 模块的使用方法如下。

1. 给 EM32DX-E4 模块接上 24V 直流电源，并接好网线。
2. 打开雷赛控制公司的调试软件 DMC Motion。在“总线配置”界面，鼠标点击 EtherCAT 主站，右键执行“扫描设备”。扫描后，EM32DX-E4 模块自动加入总线结构树中。参见图 3.7、3.8。
3. 点击“下载配置文件”，下载配置文件。
4. EM32DX-E4 模块的输入信号 In、输出信号 Out 的编号是接着 DMC-E3032 运动控制卡上的 IO 信号继续往后排列。即：输入信号为 In8~IN23，输出信号为 Out8~Out23。如图 6.2 所示。
5. 在 DMC Motion 主页面上选“运动状态”，可进入 IO 测试。如图 3.9 所示。
6. 控制 IO 信号的指令和普通运动控制卡的一样：

设置输出口的电平： `dmc_write_outbit(卡号,端口编号,输出值)`

读取输出端口的电平： `dmc_read_outbit(卡号,端口编号)`

读取输入端口的电平： `dmc_read_inbit(卡号,端口编号)`

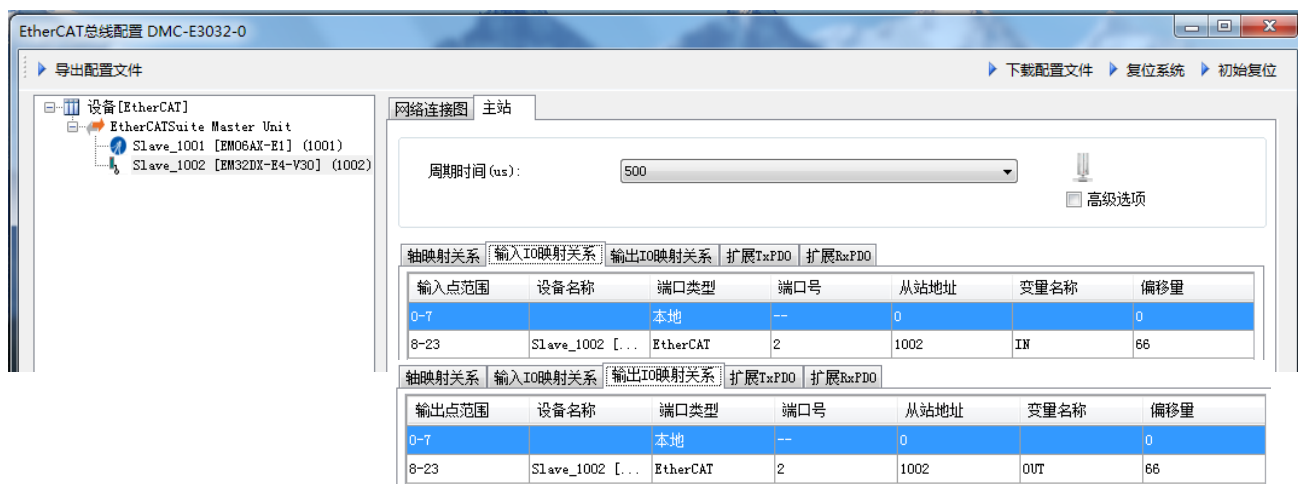


图 6.2. EM32DX-E4 模块 IO 信号的映射关系

IO 模块例程的主要代码如下。

```
private void timer1_Tick(object sender, EventArgs e) // 定时间隔50毫秒
{
    ushort i;
    for (i = 8; i < 24; i++) // 扫描一遍输入口
    {
        if (LTDMC.dmc_read_inbit(cardID, i) == 0) // 如果输入口i有信号
            LTDMC.dmc_write_outbit(cardID, i, 0); // 输出口i导通，指示灯亮
        else
            LTDMC.dmc_write_outbit(cardID, i, 1);
    }
}
```

该段函数是用定时器定时检测 IO 模块输入口，如某端口有输入信号则其对应的输出口导通。

二) 定位扩展模块

当 EtherCAT 总线控制系统中需要选用一些普通类型的电机时，可通过 EM02DP-E1 定位模块将其接入 EtherCAT 总线。

EM02DP-E1 定位模块外形如图 6.3 所示，它相当于一个 2 轴运动控制器，可控制 2 个脉冲型电机，具有点位运动、直线插补、定速运动等功能；除电机专用 I/O 口外，还有 8 路输入和 8 路输出信号。其参数详见表 6.3。

注意：不建议让定位模块控制的电机和其他节点上的总线电机进行插补运动。



图 6.3. EM02DP-E1 定位模块外观

表 6.3. EM02DP-E1 定位模块参数表

电机控制信号类型	脉冲+方向
脉冲控制信号频率范围	1~2M Hz
输入口导通电流	最小 4.2mA (15V)，典型值 6.9mA (24V)
输出口最大输出电流	300mA (21~27V)

EM02DP-E1 定位模块的 8 路通用输出信号可以通过拨码开关 SW2 设置初始电平。SW2 的定义如表 6.4 所示。

表 6.4. 拨码开关 SW2 设置输出口初始电平的定义

PIN1	PIN2	PIN3	PIN4	含义
ON		保留	保留	OUT0~OUT3 初始电平为低电平
OFF		保留	保留	OUT0~OUT3 初始电平为高电平
	ON	保留	保留	OUT4~OUT7 初始电平为低电平
	OFF	保留	保留	OUT4~OUT7 初始电平为高电平

EM02DP-E1 定位模块的脉冲输出电路、I/O 口电路和雷赛公司一般的运动控制器相同，详见用户手册。

EM02DP-E1 定位模块总线配置过程和 I/O 模块略有不同。过程如下：

1. 打开雷赛控制公司的调试软件 DMC Motion。在“总线配置”界面，鼠标点击 EtherCAT

- 主站，右键执行“扫描设备”。扫描后，EM02DP-E1 模块自动加入总线结构树中。
- EM02DP-E1 定位模块的 2 个轴自动完成映射，但其 IO 口要手动添加映射关系。从图 6.4 中可看出，其输入口的变量名称为 GeneralInputIO，其输出口的变量名称为 GeneralOutputIO。
- 点击“输入 IO 映射关系”选项卡，如图 6.5 所示。点击“添加”，然后“映射类型”选 EtherCAT、“从站类型”选数字 IO、“映射从站”选 EM02DP-E1、“映射变量”选 GeneralInputIO，然后点击“确定”。完成输入口映射，结果如图 6.6 所示，输入口编号为 IN24~31。

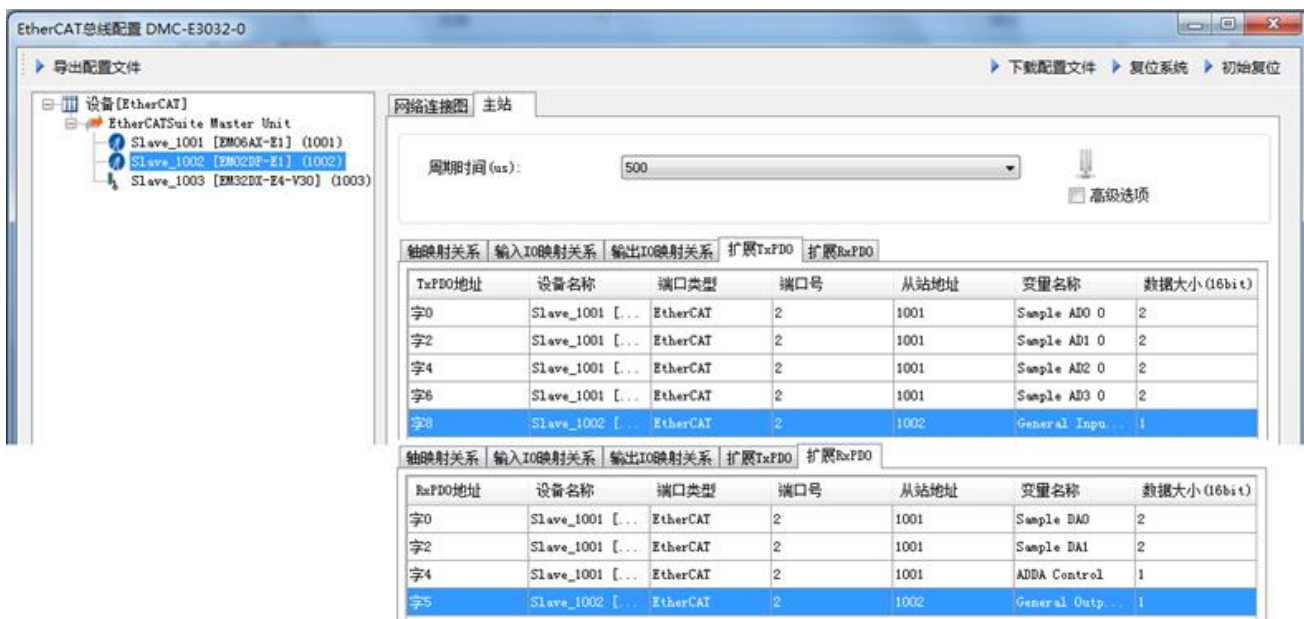


图 6.4. EM02DP-E1 定位模块 IO 口的变量名称

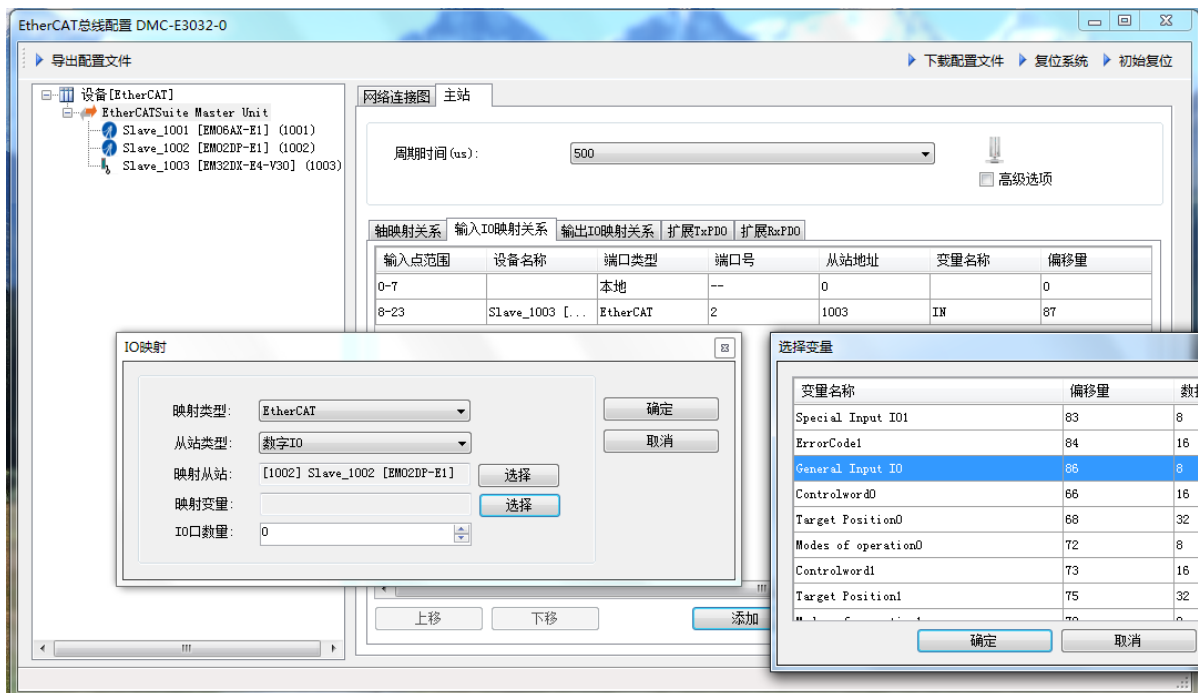


图 6.5. 添加 EM02DP-E1 定位模块的输入口映射

轴映射关系	输入IO映射关系	输出IO映射关系	扩展TxPDO	扩展RxPDO		
输入点范围	设备名称	端口类型	端口号	从站地址	变量名称	偏移量
0-7		本地	--	0		0
8-23	Slave_1003 [...]	EtherCAT	2	1003	IN	87
24-31	Slave_1002 [...]	EtherCAT	2	1002	General Inpu...	86

图 6.6. EM02DP-E1 定位模块的输入口映射关系

4. 点击“输出 IO 映射关系”选项卡，添加输出口映射关系，过程如图 6.7 所示；结果如图 6.8 所示，输出口编号为 OUT24~31。
5. 点击“下载配置文件”，下载配置文件。
6. 系统完成复位后，在 DMC Motion 主页面上选“运动状态”，在“专用信号状态”栏中可以测试 EM02DP-E1 定位模块的 IO 信号。如图 6.9 所示。
7. 在主页面上点击“功能测试”按钮，可以对定位模块的电机进行测试。注意：在电机运动之前要打开“使能”。

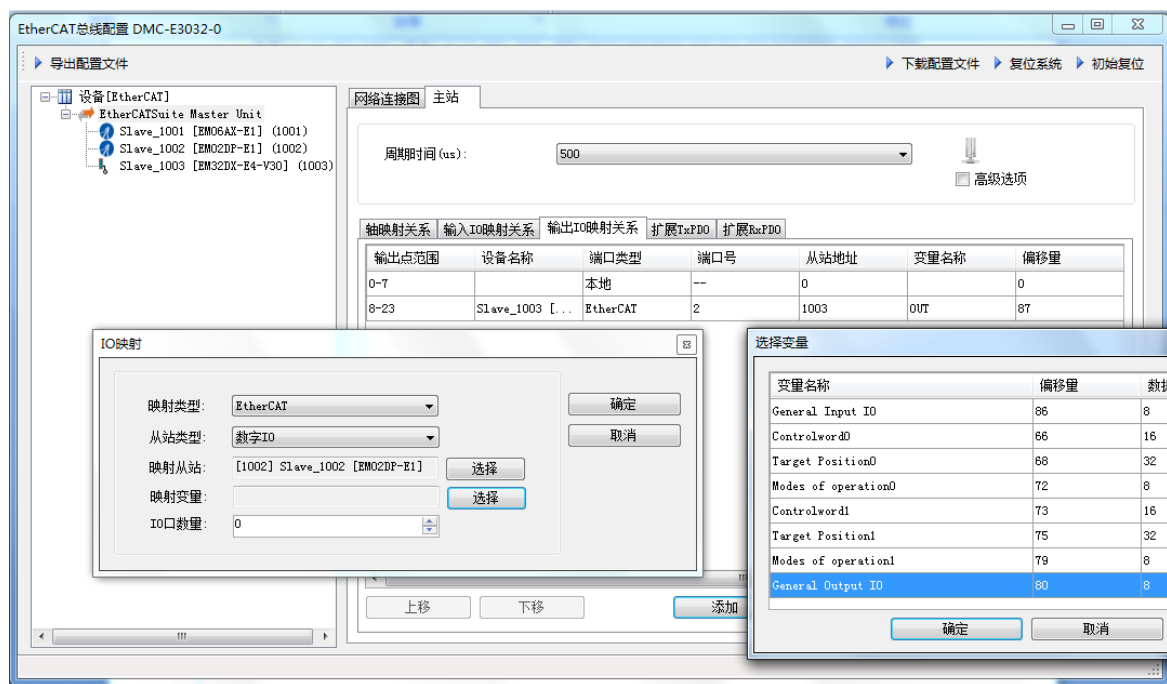


图 6.7. 添加 EM02DP-E1 定位模块的输出口映射

轴映射关系	输入IO映射关系	输出IO映射关系	扩展TxPDO	扩展RxPDO		
输出点范围	设备名称	端口类型	端口号	从站地址	变量名称	偏移量
0-7		本地	--	0		0
8-23	Slave_1003 [...]	EtherCAT	2	1003	OUT	87
24-31	Slave_1002 [...]	EtherCAT	2	1002	General Outp...	80

图 6.8. EM02DP-E1 定位模块的输出口映射关系



图 6.9. EM02DP-E1 定位模块 IO 口的测试

EM02DP-E1 定位模块控制电机运动的指令、控制 IO 的指令和运动控制卡的基本一样。只有回零模式和一般的运动控制卡不一样，且完成回零后自动将位置计数器清零。

EM02DP-E1 的轴回零模式有 6 种：

- | | |
|-----------------|-----------------|
| 模式 1：正向一次回零， | 模式 4：负向一次回零； |
| 模式 2：正向一次回零加反找， | 模式 5：负向一次回零加反找； |
| 模式 3：正向二次回零， | 模式 6：负向二次回零。 |

例程 4：

EM02DP-E1 定位模块的测试程序界面如图 6.10 所示，其程序代码见附录例程 4。



图 6.10. EM02DP-E1 定位模块测试程序界面

三) AD/DA 扩展模块

EM06AX-E1 模块有 4 路 AD 输入和 2 路 DA 输出，分辨率高达 16 位。其电源电路加有隔离元件，可以有效隔离外部电路的干扰。EM06AX-E1 模块的外形如图 6.11 所示，其接口参数如表 6.5 所示。



图 6.11. EM06AX-E1 模块外形

表 6.5. AD/DA 接口参数

模拟/数字(AD)部分	电压输入
电源电压	24Vdc (20Vdc~28Vdc)
模拟信号输入通道	4 路
输入范围	直流 -10V~10V
分辨率	16 位 (1 LSB = 0.3mV)
输入阻抗	100kΩ 以上
精度 (25℃)	±满量程的 0.01%
精度 (0~55℃)	±满量程的 0.02%
极限范围	直流 -12V~12V
响应时间	250us
数字/模拟(DA)部分	电压输出
模拟信号输出通道	2 路
输出范围	直流 -10V~10V
分辨率	16 位 (1 LSB = 0.3mV)
精度 (25℃)	±满量程的 0.01%
精度 (0~55℃)	±满量程的 0.02%
响应时间	250us
最大负载	≥5kΩ 以上

可通过拨码开关 S1 将输入、输出信号的范围设置为-10~10V 或 0~10V，详见表 6.6。

表 6.6. 拨码开关 S1 的设置

Pin1	Pin2	Pin3	Pin4	模式
OFF				AD 电压模式
ON				保留
	OFF			DA 电压模式
	ON			保留
		OFF		AD量程: -10~10V
		ON		AD 量程: 0~10V
			OFF	DA量程: -10~10V
			ON	DA 量程: 0~10V

模拟量输入接口、模拟量输出接口电路原理图如图 6.12、6.13 所示。

EM06AX-E1 模块总线配置过程与 I0 扩展模块相同。

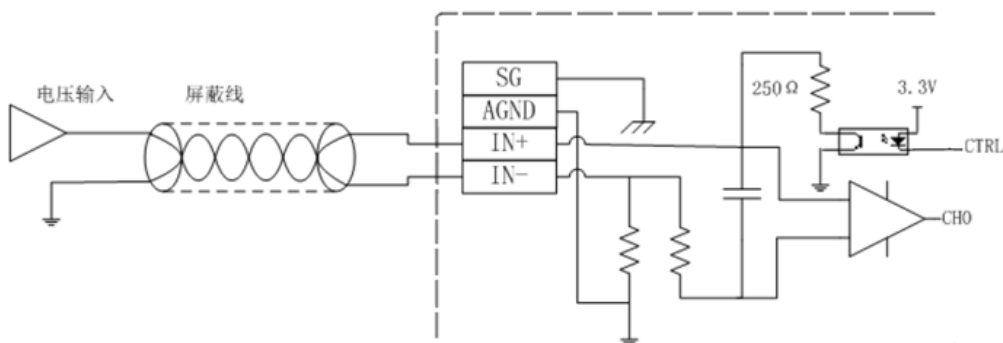


图 6.12. 模拟量输入接口电路原理图

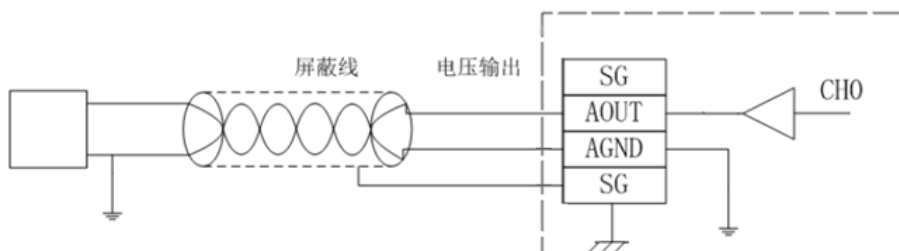


图 6.13. 模拟量输出接口电路原理图

AD/DA 的控制指令是通过总线读写模块的过程数据 PDO(ProcessDataObject)完成的, 读、写指令格式如下:

nmc_write_rxpdo_extra(卡号, 端口号, RxPDO 地址, 数据长度, 写入的数据);

nmc_read_txpdo_extra(卡号, 端口号, TxPDO 地址, 数据长度, ref 读出的数据变量);

在 MOTION 软件的“总线配置”页面上可以看到 AD/DA 模块的 PDO 数据的地址和端口号, 如图 6.14 所示。从图中可看出, 数模转换接口 DA0、DA1 的 RxPDO 地址分别为 0 和 2, AD/DA 的控制字的地址为 4; 模数转换接口 AD0、AD1、AD2 和 AD3 的 TxPDO 地址分别为 0、2、4 和 6。

EtherCAT总线配置 DMC-E3032-0

导出配置文件

设备 [EtherCAT]

- EtherCATSuite Master Unit
- Slave_1001 [EM06AX-E1] (1001)
- Slave_1002 [EM32DX-E4-V30] (1002)

网络连接图 主站

周期时间(us): 500

高级选项

RxPDO地址	设备名称	端口类型	端口号	从站地址	变量名称	数据大小(16bit)
字0	Slave_1001 [...]	EtherCAT	2	1001	Sample DA0	2
字2	Slave_1001 [...]	EtherCAT	2	1001	Sample DA1	2
字4	Slave_1001 [...]	EtherCAT	2	1001	ADDA Control	1

TxPDO地址	设备名称	端口类型	端口号	从站地址	变量名称	数据大小(16bit)
字0	Slave_1001 [...]	EtherCAT	2	1001	Sample AD0 0	2
字2	Slave_1001 [...]	EtherCAT	2	1001	Sample AD1 0	2
字4	Slave_1001 [...]	EtherCAT	2	1001	Sample AD2 0	2
字6	Slave_1001 [...]	EtherCAT	2	1001	Sample AD3 0	2

图 6.14. AD/DA 模块的过程数据 PDO 地址和端口

即:

DA 转换使能指令为: `nmc_write_rxpdo_extra(0, 2, 4, 2, 1);`

DA 转换指令为: `nmc_write_rxpdo_extra(0, 2, RxPDO 地址, 2, 写入的数据);`

AD 转换指令为: `nmc_read_txpdo_extra(0, 2, TxPDO 地址, 2, ref 读出的数据变量);`

其中:

卡号为 0; 端口号为 2; 数据长度=2, 是定义数据长度为 32 位。

在程序中与-10~10V 电压对应的数值为-100000~100000, 即: 电压的 10000 倍。详见程序代码。

AD/DA 模块的例程的主要代码如下, 其程序界面如图 6.15 所示。

```
ushort cardID = 0;           // 卡号
ushort portN = 2;           // 端口号
ushort dataL = 2;           // 数据长度
staticdouble Vout = 0;

privatevoid Form1_Load(object sender, EventArgs e)
{
    short num;
    num = LTDMC.dmc_board_init(); // 初始化运动控制卡
    if (num <= 0 || num > 8)
        MessageBox.Show("初始卡失败!", "出错");
    LTDMC.nmc_write_rxpdo_extra(cardID, portN, 4, dataL, 1); // AD接口使能打开
}

privatevoid button1_Click(object sender, EventArgs e) // D/A转换
{
    double DAvalue1 = double.Parse(textBox1.Text);
    LTDMC.nmc_write_rxpdo_extra(cardID, 2, 2, 2, (int)(DAvalue1 * 10000)); // 1号通道输出
}

privatevoid timer1_Tick(object sender, EventArgs e) // DA 及 AD 转换
{
    int Vin = 0;
    Vout = Vout + 0.1;
    if (Vout > 10.0) // 输出电压在-10~10V之间变化
        Vout = -10.0;
    LTDMC.nmc_write_rxpdo_extra(cardID, portN, 0, dataL, (int)(Vout * 10000));
    // D/A转换, DA的0号通道输出电压信号
    Thread.Sleep(5); // 延时5毫秒
    LTDMC.nmc_read_txpdo_extra(cardID, portN, 2, dataL, ref Vin); // A/D转换, 读AD的1号通道
    textBox2.Text = (Vin / 10000.0).ToString();
}
```

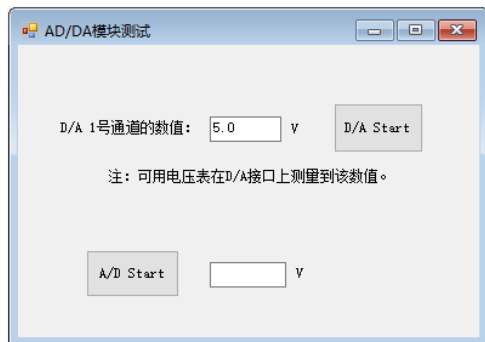


图 6.15. AD/DA 模块的过程数据 PDO 地址和端口

注: D/A 的 0 号通道输出电压已用导线接入 A/D 的 1 号通道。
即: 图 6.5 中的 AOUT 和 AGND 分别用导线与图 6.4 中的 IN+、IN- 相连。

四) 编码器扩展模块

EM03De-E4 模块具有 3 路 5V 差分编码器信号输入接口、4 路高速输入接口、3 路高速输出接口，可实现编码器计数、高速位置锁存和高速位置比较等功能。输入输出接口采用了光电隔离和滤波电路，抗干扰性能好。

如果不使用位置锁存和位置比较功能，输入和输出可作通用 I/O 口。

EM03De-E4 模块的外形如图 6.16 所示，其接口定义如表 6.7 所示，其接口电路与雷赛公司的运动控制相同，其参数如表 6.8 所示。



图 6.16. EM03De-E4 模块外观

表 6.7. EM03De-E4 模块接口定义

序号	功能	序号	功能
1	EA0+	2	EA1+
3	EA0-	4	EA1-
5	EB0+	6	EB1+
7	EB0-	8	EB1-
9	EZ0+	10	EZ1+
11	EZ0-	12	EZ1-
13	D5V	14	D5V
15	DGND	16	DGND
17	保留	18	保留
19	EA2+	20	IN0
21	EA2-	22	IN1
23	EB2+	24	IN2
25	EB2-	26	IN3
27	EZ2+	28	EGND
29	EZ2-	30	OUT0
31	D5V	32	OUT1
33	DGND	34	OUT2
35	保留	36	EGND

表 6.8. EM03De-E4 模块主要参数表

编码器接口输入特性	
编码器组数	3 组(EA+EB+EZ)
输入类型	差分输入、单端输入
脉冲频率范围	脉冲方向: 0~4M Hz; AB 相: 0~2M Hz
高速输入接口特性	
输入通道数	4 路
输入电压	21~27V DC
导通电流	≥4.2mA(15V) 典型值 6.9 mA(24V)
高速输出接口特性	
输出通道数	3 路
输出类型	漏型输出, 低电平有效
负载电压	21~27V DC
输出电流	300mA/通道

EM03De-E4 模块的使用方法如下。

1. 给 EM03De-E4 模块接上 24V 直流电源，并接好网线。
2. 打开雷赛控制公司的调试软件 DMC Motion。在“总线配置”界面，鼠标点击 EtherCAT 主站，右键执行“扫描设备”。扫描后，EM03De-E4 模块自动加入总线结构树中。
3. 根据需要，在主站的“扩展 TxPDO”界面中添加相应的变量。然后下载配置文件。
4. 在程序中通过对象字典设置编码器的相关参数。
5. 在程序中通过过程数据（PDO）读编码器的相关检测数值。

EM03De-E4 模块需要设置的主要参数如表 6.9、6.10、6.11 所示。主要过程数据变量如表 6.12 所示。

下面通过一个实验，详细介绍 EM03De-E4 模块的主要功能。

该实验的电路图如图 6.17 所示。模块的编码器 1 输入口接入闭环步进电机的编码器信号。模块的高速输出口 OUT1 接有一个 LED 指示灯，并将 OUT1 的信号接入高速输入口 IN1。

用调试软件 DMC Motion 在主站的“扩展 TxPDO”界面中添加 2 个变量 Encoder1_Val 和 Ltc1_Encoder1Val，如图 6.18 所示。然后下载配置文件。

表 6.9. 设置编码器参数的对象字典

索引号	子索引号	名称	数据类型	属性	描述
6000H	01H	Encoder0_SetMode	Unsigned 8	r/w	编码器 0 模式： 0: AB 相 4 倍频（默认值） 1: 脉冲/方向模式
	02H	Encoder0_SetABPhase	Unsigned 8	r/w	编码器 0 AB 相模式下的方向： 0: 负方向 1: 正方向（默认值）
	03H	Encoder0_SetVal	Signed 32	r/w	设置编码器 0 的当前位置
6001H	编码器 1 参数设置。其子索引 01、02、03 的定义和 6000H 的相同。				
6002H	编码器 2 参数设置。其子索引 01、02、03 的定义和 6000H 的相同。				

表 6.10. 设置锁存器参数的对象字典

索引号	子索引号	名称	数据类型	属性	描述
6200H	01H	Ltc0_Clear	Unsigned 8	r/w	1: 锁存器 0 状态清除
	02H	Ltc0_SetMode	Unsigned 8	r/w	锁存器 0 工作模式： 0: 单次锁存, 1: 连续锁存
	03H	Ltc0_SetFollow	Unsigned 8	r/w	锁存器 0 锁存信号触发逻辑： 0: IN0 电平上升沿 1: IN0 电平下降沿 2: IN0 任意沿
	04H	Ltc0_SetFilterTime	Unsigned 32	r/w	设置滤波时间，单位微秒 0~65535，默认值为 0
6201H	锁存器 1 参数设置。其子索引 01、02、03、04 的定义和 6200H 的相同。				
6202H	锁存器 2 参数设置。其子索引 01、02、03、04 的定义和 6200H 的相同。				
6203H	锁存器 3 参数设置。其子索引 01、02、03、04 的定义和 6200H 的相同。				

表 6.11. 设置比较器参数的对象字典

索引号	子索引号	名称	数据类型	属性	描述
6400H	01H	Cmp0_Clear	Unsigned 8	r/w	1: 比较器 0 清除缓冲区及状态
	02H	Cmp0_SetMode	Unsigned 8	r/w	比较器 0 的触发模式: 0: 关闭; 1: 等于; 2: 小于; 3: 大于; 4: fifo; 5: linear
	03H	Cmp0_Encoder_Sel	Unsigned 8	r/w	选择编码器, 0、1、2
	04H	Cmp0_Out_Logic	Unsigned 8	r/w	比较器输出信号逻辑: 0: 低电平 1: 高电平
	05H	Cmp0_Out_timer	Unsigned 32	r/w	设置比较器 0 输出信号持续时间, 单位微秒。仅第 4、5 种模式有效
	06H	Cmp0_AddData_Val	Unsigned 32	r/w	比较器 0 添加比较点的位置
	07H	Cmp0_Linear_Number	Unsigned 32	r/w	比较器 0 采用线性比较时, 比较点数量
	08H	Cmp0_Linear_Interval	Signed 32	r/w	比较器 0 采用线性比较时, 比较点的增量值
6401H	比较器 1 参数设置。其子索引 01~08 的定义和 6400H 的相同。				
6402H	比较器 2 参数设置。其子索引 01~08 的定义和 6400H 的相同。				

表 6.12. 主要的过程数据变量

名称	变量	数据类型	属性
编码器 0 的值	Encoder0_Val	Signed32	r
编码器 1 的值	Encoder1_Val		
编码器 2 的值	Encoder2_Val		
锁存器 0 锁存编码器 0 的值	Ltc0_Encoder0Val		
锁存器 0 锁存编码器 1 的值	Ltc0_Encoder1Val		
锁存器 0 锁存编码器 2 的值	Ltc0_Encoder2Val		
锁存器 1 锁存编码器 0 的值	Ltc1_Encoder0Val		
锁存器 1 锁存编码器 1 的值	Ltc1_Encoder1Val		
锁存器 1 锁存编码器 2 的值	Ltc1_Encoder2Val		
锁存器 2 锁存编码器 0 的值	Ltc2_Encoder0Val		
锁存器 2 锁存编码器 1 的值	Ltc2_Encoder1Val		
锁存器 2 锁存编码器 2 的值	Ltc2_Encoder2Val		
锁存器 3 锁存编码器 0 的值	Ltc3_Encoder0Val		
锁存器 3 锁存编码器 1 的值	Ltc3_Encoder1Val		
锁存器 3 锁存编码器 2 的值	Ltc3_Encoder2Val		

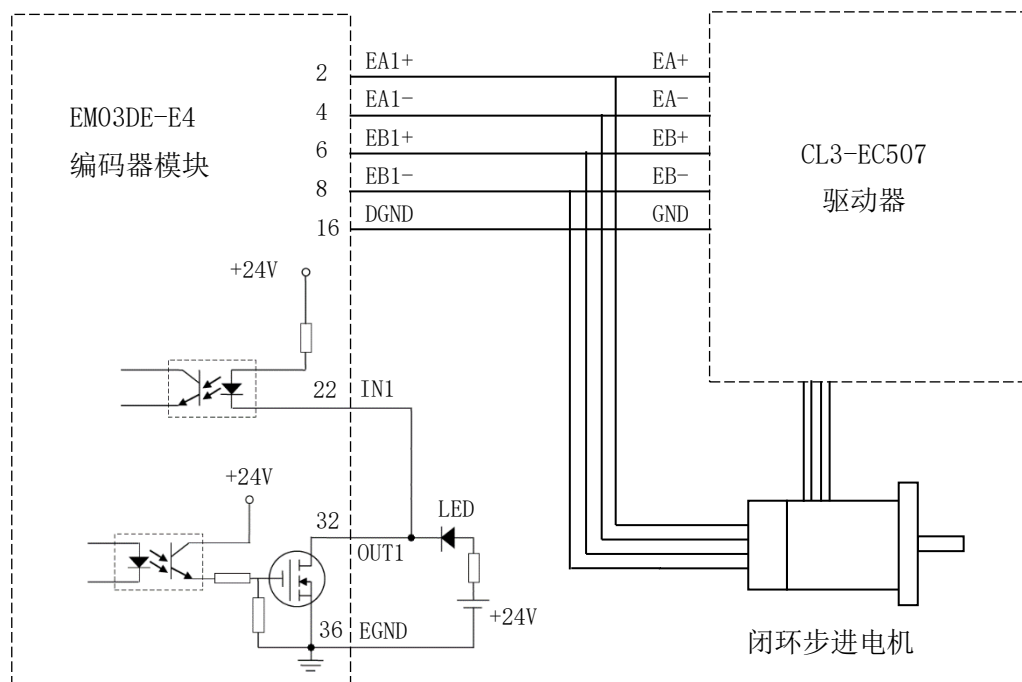


图 6.17. EM03De-E4 模块外观

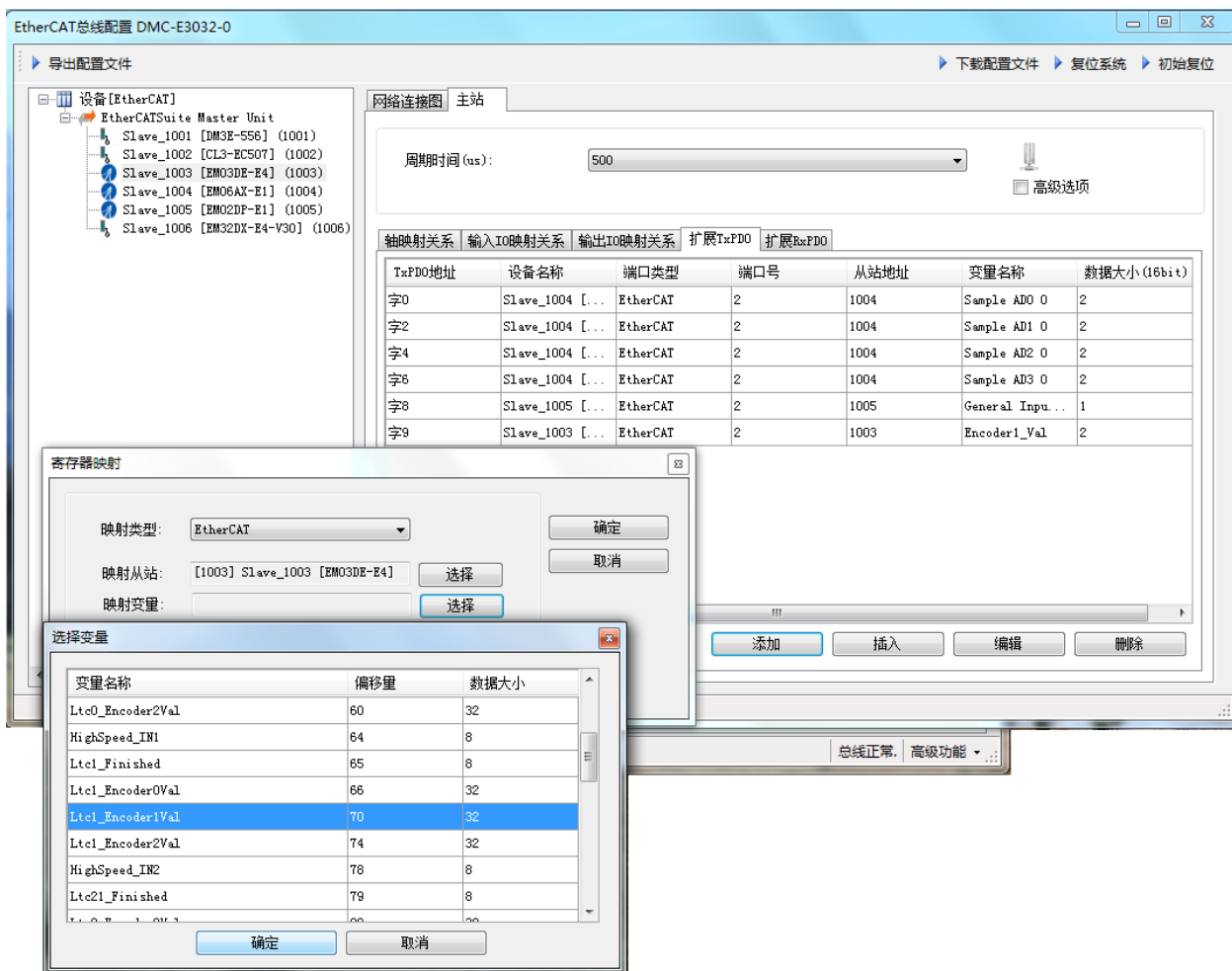


图 6.18. 添加 2 个 EM03De-E4 模块的过程数据变量

在“扩展 TxPDO”界面中可看到变量 Encoder1_Val 和 Ltc1_Encoder1Val 的地址分别为 9 和 11。

例程 5:

EM03De-E4 模块测试程序的界面如图 6.19 所示。

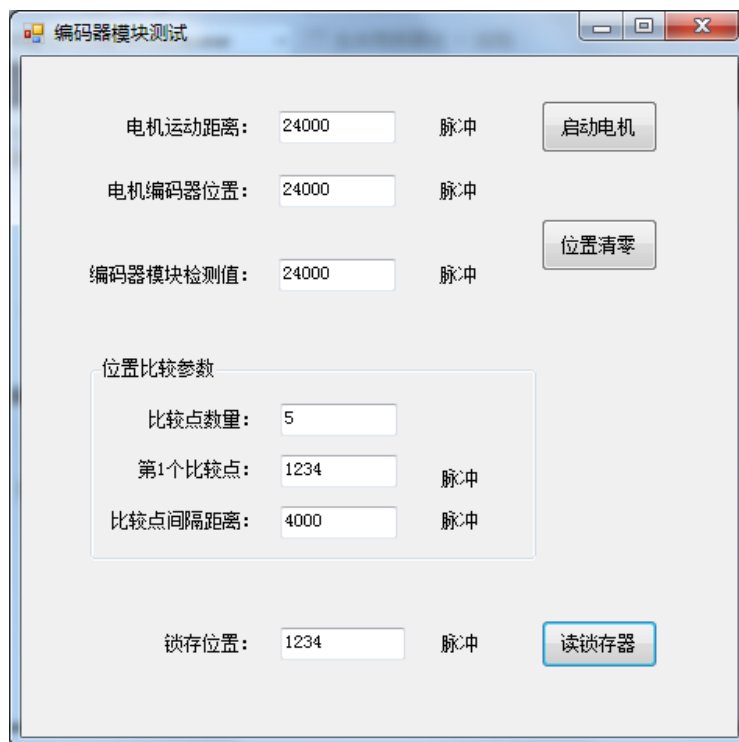


图 6.19. EM03De-E4 模块测试程序的界面及测试结果

该程序通过对象字典设置编码器 1 为 AB 相 4 倍频模式且方向为正；设置比较器 1 为线性比较模式、以低电平触发、选用编码器 1 的数据进行比较、触发信号持续 0.3 秒、第 1 个比较点位置、比较点数量、比较点间隔距离；设置锁存器 1 为单次触发模式、由信号的下降沿触发。并通过读过程数据变量 Encoder1_Val 和 Ltc1_Encoder1Val, 获取编码器 1 和锁存器 1 的数值。

从图 6.17 中可看出，位置锁存信号由位置比较信号提供。

如图 6.19 所示，该实验将位置比较点设为 5 个，第 1 个比较点为 1234，比较点间隔距离为 4000 个脉冲。

当程序启动后，首先按“位置清零”按钮，将电机和编码器模块的位置清零，然后按“启动电机”按钮，让电机运动。当电机运动了 1234 脉冲位置时，位置比较器触发输出口 OUT1，低电平持续 0.3 秒，即 LED 指示灯亮 0.3 秒；同时，该触发信号通过输入口 IN1 触发锁存器 1，将编码器 1 的当前位置 1234 锁存下来。

当电机继续运动 4000 个脉冲后，再次触发 OUT1，LED 亮 0.3 秒；电机每前进 4000 个脉冲，LED 就亮一次，直到电机运动到终点 24000 处，LED 共亮 5 次。

按“读锁存器”按钮，显示锁存位置为 1234 个脉冲。因为锁存器设置为单次触发模式，所以只记录比较器的第 1 个比较点位置。

电机反方向运动时，也可以做位置比较，但比较点间隔距离要设为负值。

该程序代码见附录中的例程 5。

七. 多轴插补运动及总线模块的综合应用

DMC-E3032 运动控制卡可以进行 2~16 轴的直线插补、2 轴圆弧插补、3 轴空间圆弧插补、3 轴绽放螺旋线插补、收敛螺旋线插补、圆柱螺旋线插补。

一) 曲线插补指令及其曲线方程

1. 基于圆心的 3 轴螺旋线

其的运动指令如下：

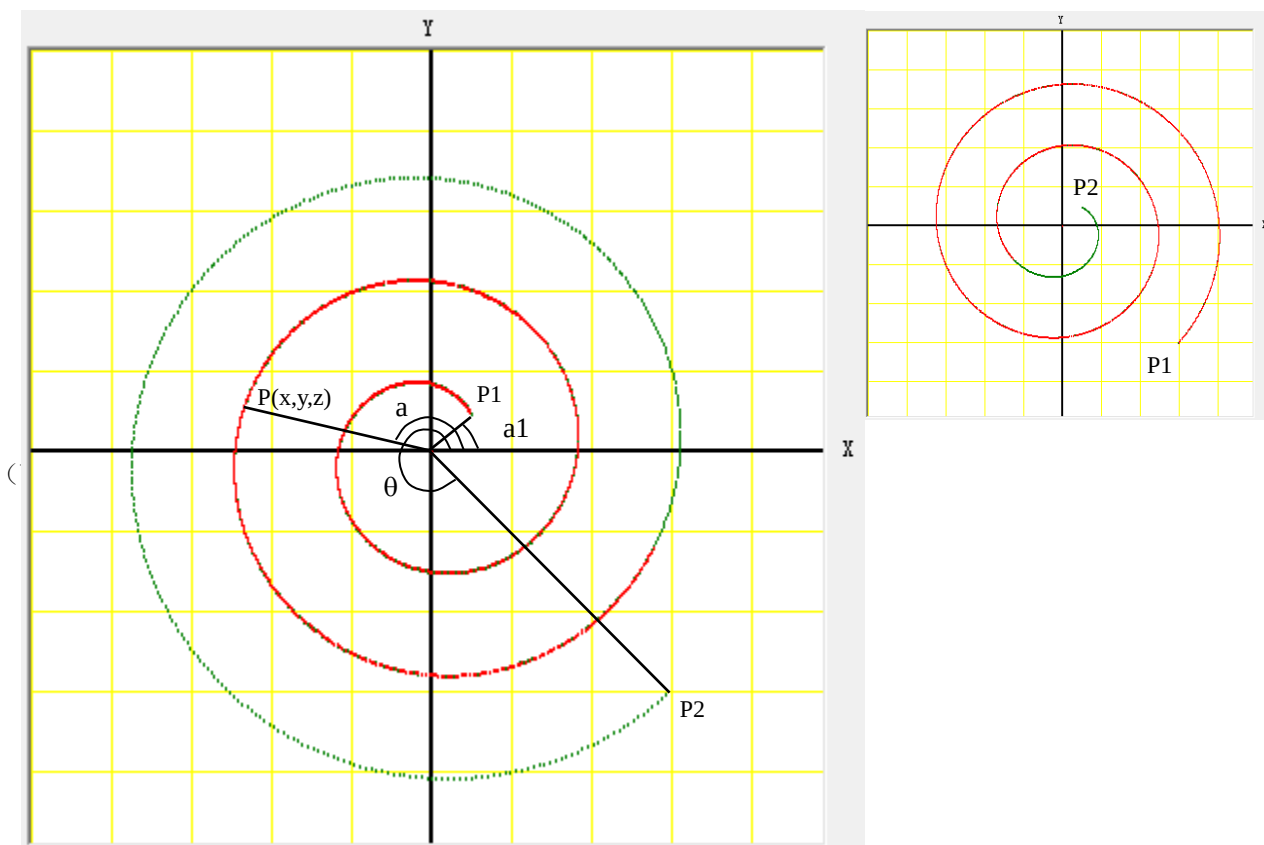
dmc_arc_move_center_unit (卡号, 坐标系号, 插补轴数, 轴号数组, 终点坐标数组,
圆心坐标数组, 圆弧方向, 圆弧圈数, 坐标类型)

如果只有 2 轴参与插补, 曲线为平面曲线; 如果是 3 轴参与插补, 则为空间螺旋线。

该指令允许参与插补的轴数为 16。从第 3 轴开始, 后续轴的运动规律一样, 线性跟随轴 1、轴 2 运动。

如图 7.1 (a) 所示, 已知螺旋线起点为 P1、终点为 P2, 圆心 O 为 (x0, y0)。

可计算出起点与 X 轴的夹角为 a1、终点与 X 轴的夹角为 a2。



(a) 绽放式

图 7.1. 螺旋线在平面上的投影及参数

螺旋线的参数方程如下, 其螺旋角 a 由 a1 转至 a2。

$$\begin{cases} x = x_0 + R \cos(a) \\ y = y_0 + R \sin(a) \\ z = z_1 + (z_2 - z_1) (a - a_1) / (a_2 - a_1) \end{cases} \quad (7.1)$$

式中：

$$R = R1 + (R2 - R1) (a - a1) / (a2 - a1)$$

$$R1 = (x1^2 + y1^2)^{0.5}$$

$$R2 = (x2^2 + y2^2)^{0.5}$$

$$a1 = \text{atan}((y1 - y0) / (x1 - x0)) \quad (-\pi < a1 < \pi)$$

$$\theta = \text{atan}((y2 - y0) / (x2 - x0)) \quad (-\pi < \theta < \pi)$$

$$a2 = \theta + 2 \times \pi \times n \quad (\text{当 } a1 \text{ 大于 } \theta \text{ 时, } a2 = \theta + 2 \times \pi \times (n+1))$$

$x0, y0$ 为圆心坐标；

$x1, y1, z1$ 为起点 $P1$ 的坐标； $x1 - x0$ 和 $y1 - y0$ 不可同时为 0。

$x2, y2, z2$ 为终点 $P2$ 的坐标。

当 $R1 < R2$ 时，曲线为绽放螺旋线，如图 7.1 (a) 所示；

当 $R1 = R2$ 时，曲线为圆柱螺旋线；

当 $R1 > R2$ 时，曲线为收敛螺旋线，如图 7.1 (b) 所示。

式 (7.1) 中的变量 a 如果从小到大由 $a1$ 变化至 $a2$ ，则轨迹为逆时针方向运动，如图 7.1 所示。如果变量 a 如果从大到小由 $a1$ 变化至 $-a2$ ，则轨迹为顺逆时针方向运动。

DMCE332 运动控制卡在进行螺旋线插补时，自动计算式 (7.1) 中的参数，无需人工干预。

2. 基于半径的 3 轴圆柱螺旋线

如图 7.2 所示，已知圆弧的起点、终点、半径，那么就有 4 条轨迹可选。顺时针方向圆弧轨迹称为顺圆；逆时针方向的圆弧轨迹称为逆圆。只有确定圆弧方向、圆弧角是小于 180 度还是大于 180 度，才能确定唯一的圆弧。

基于半径的 3 轴圆柱螺旋线的轨迹方程和式 (7.1) 相同，只是圆弧半径 R 为常数。

基于半径的 3 轴圆柱螺旋线的运动指令如下：

dmc_arc_move__radius_unit (卡号，坐标系号，插补轴数，轴号数组，终点坐标数组，圆弧半径，圆弧方向，圆弧圈数，坐标类型)

该指令允许参与插补的轴数为 16。从第 4 轴开始，后续轴的运动规律和 Z 轴相似，为线性跟随运动。

为提供圆弧角的信息，该指令参照 G 代码的习惯，当圆弧角小于 180 度时，半径为正；圆弧角大于 180 度时，半径为负。

图 7.3 为 2 轴插补轨迹，轨迹 C1 圆弧角小于 180 度，半径为 40000；轨迹 C2 圆弧角大于 180 度，半径为 -40000。C1、C2 起点、终点相同。

图 7.4 为圆柱螺旋线插补轨迹。起点坐标为 (0, 0, 0)，终点坐标为 (5000, 0, 4000)，半径为 2500，圆弧方向为顺圆，螺旋线为 2 圈。

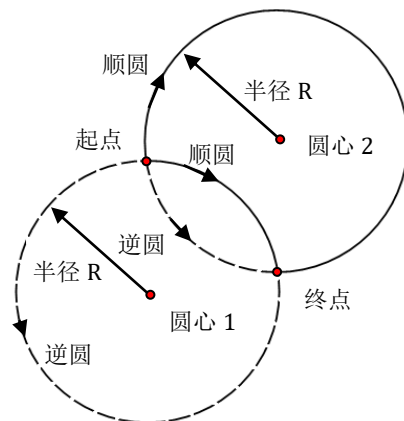


图 7.2. 基于半径的圆弧

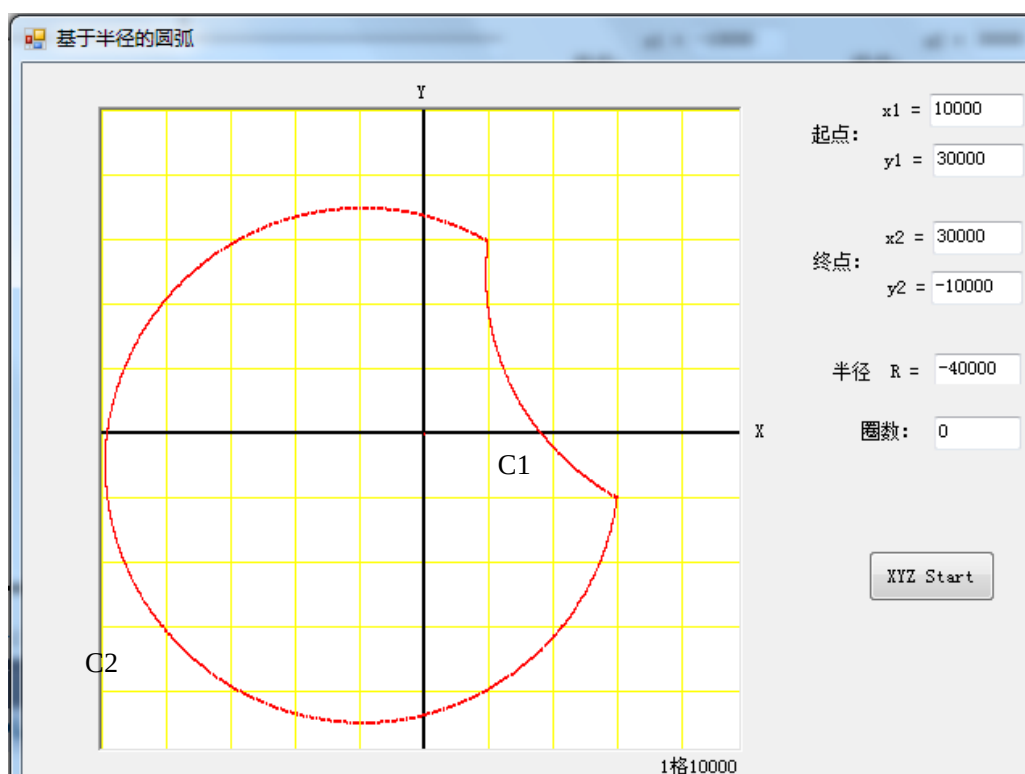


图 7.3. 基于半径的圆弧轨迹

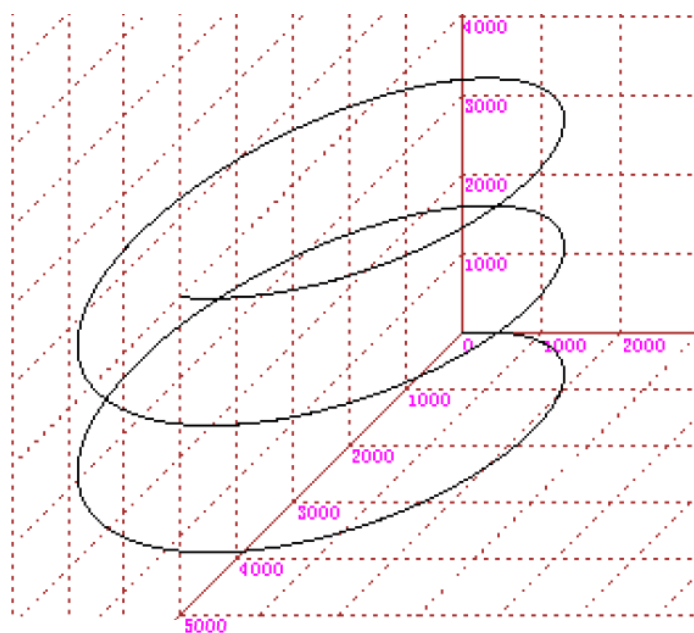


图 7.4. 基于半径的圆柱螺旋线轨迹

3. 基于中间点的空间圆弧插补

其运动指令如下：

dmc_arc_move__3points_unit (卡号, 坐标系号, 插补轴数, 轴号数组, 终点坐标数组,
中间点坐标数组, 圆弧方向, 圆弧圈数, 坐标类型)

所谓中间点为起点至终点间圆弧上的任意一个点。

该指令允许参与插补的轴数为16。

当圆弧圈数为负数时，表示为空间圆弧插补。其绝对值减1为空间圆弧的圈数。如：-1即0圈空间圆弧，-2即1圈空间圆弧；

当圆弧圈数为正数时，表示为圆柱螺旋线插补及圈数。

如图 7.5 所示，C1 为 XZ 平面上的一段圆弧（蓝色曲线，由折线连接圆弧上的 15 个点所得）。其起点坐标为（100000，20000，0），终点坐标为（20000，20000，0），圆心坐标为（60000，20000，30000），半径为 50000。取中间点为（60000，20000，80000）。

将 XZ 平面绕 Z 轴旋转 15 度，得新坐标系 X'Y'Z'及曲线 C2（绿色曲线）；

再将 X'Z'平面绕 X'轴旋转 15 度，得新坐标系 X''Y''Z''及曲线 C3（绿色曲线）。

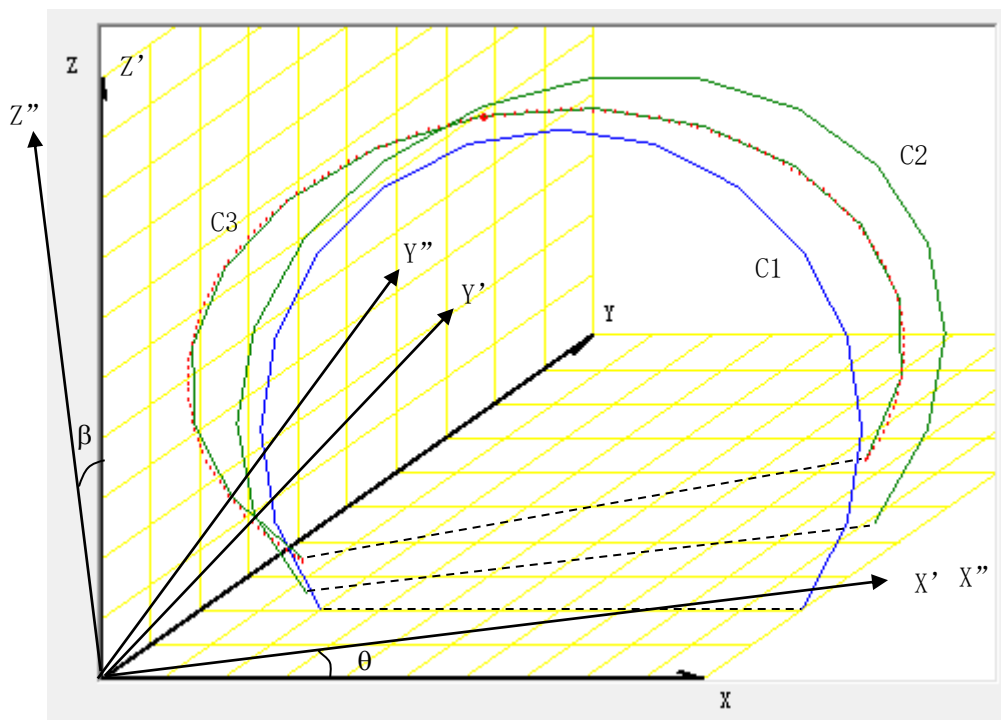


图 7.5. 基于中间点圆弧插补

在 XYZ 坐标系中的一个点（x，y，z）绕 Z 轴旋转 θ 角，得新的坐标（x'，y'，z'）。坐标旋转变换的公式如下：

$$\begin{cases} x' = \cos(\theta)x - \sin(\theta)y \\ y' = \sin(\theta)x + \cos(\theta)y \\ z' = z \end{cases} \quad (7.2)$$

在 X'Y'Z'坐标系中的一个点（x'，y'，z'）绕 Z'轴旋转 β 角，得新的坐标（x''，y''，z''）。坐标旋转变换的公式如下：

$$\begin{cases} x'' = x' \\ y'' = \cos(\beta)y' - \sin(\beta)z' \\ z'' = \sin(\beta)y' + \cos(\beta)z' \end{cases} \quad (7.3)$$

将圆弧 C1 的起点、终点和中间点的坐标分别带入公式（7.2）和（7.3），且 $\theta=\beta=15$ 度；得圆弧 C3 在 XYZ 坐标系中的起点坐标为（91416.202，43660.253，11698.729）；终点坐标为（14142.132，23660.251，6339.751）；中间点坐标为（52779.173，12954.732，86293.304），

如图 7.5 中的 C3 上的红点所示。

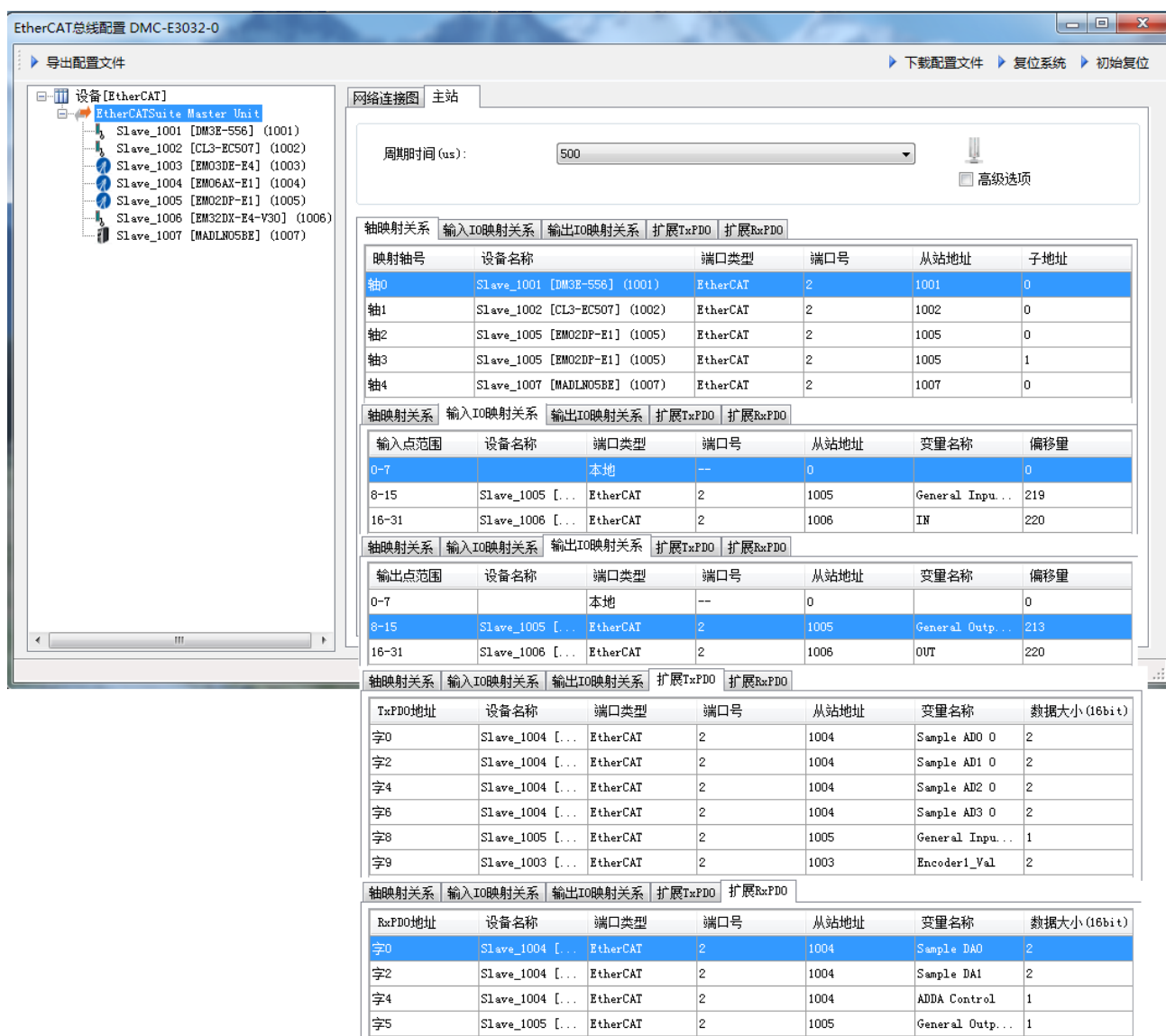
圆弧曲线 C3 既不和 XZ 平面平行、也不与 XY 或 YZ 平面平行，是一个典型的空间圆弧。将 C3 的起点、终点和中间点的坐标带入指令 `dmc_arc_move_3points_unit`，运行后，得图 7.5 中的红色轨迹。该轨迹与 C3 的理论轨迹完全重合。

二) 配置文件保存与下载

例程 6:

如图 7.6 所示，本例程硬件有 7 个节点，分别是#1001 步进电机驱动器 DM3E-556、#1002 闭环步进电机驱动器 CL3-EC507、#1003 编码器扩展模块 EM03DE-E4、#1004AD/DA 扩展模块 EM06AX-E1、#1005 定位扩展模块 EM02DP-E1、#1006IO 扩展模块 EM32DX-E4-V30、#1007 松下伺服电机驱动器 MADLN05BE。

各节点设置完成后，轴的映射关系、输入输出映射关系、PDO 映射关系如图 7.6 所示。



设备 [EtherCAT]

- EtherCATSuite Master Unit
 - Slave_1001 [DM3E-556] (1001)
 - Slave_1002 [CL3-EC507] (1002)
 - Slave_1003 [EM03DE-E4] (1003)
 - Slave_1004 [EM06AX-E1] (1004)
 - Slave_1005 [EM02DP-E1] (1005)
 - Slave_1006 [EM32DX-E4-V30] (1006)
 - Slave_1007 [MADLN05BE] (1007)

网络连接图 主站

周期时间 (us): 500

轴映射关系

映射轴号	设备名称	端口类型	端口号	从站地址	子地址
轴0	Slave_1001 [DM3E-556] (1001)	EtherCAT	2	1001	0
轴1	Slave_1002 [CL3-EC507] (1002)	EtherCAT	2	1002	0
轴2	Slave_1005 [EM02DP-E1] (1005)	EtherCAT	2	1005	0
轴3	Slave_1005 [EM02DP-E1] (1005)	EtherCAT	2	1005	1
轴4	Slave_1007 [MADLN05BE] (1007)	EtherCAT	2	1007	0

输入IO映射关系

输入点范围	设备名称	端口类型	端口号	从站地址	变量名称	偏移量
0-7		本地	--	0		0
8-15	Slave_1005 [...]	EtherCAT	2	1005	General Inpu...	219
16-31	Slave_1006 [...]	EtherCAT	2	1006	IN	220

输出IO映射关系

输出点范围	设备名称	端口类型	端口号	从站地址	变量名称	偏移量
0-7		本地	--	0		0
8-15	Slave_1005 [...]	EtherCAT	2	1005	General Outp...	213
16-31	Slave_1006 [...]	EtherCAT	2	1006	OUT	220

扩展TxPDO

TxPDO地址	设备名称	端口类型	端口号	从站地址	变量名称	数据大小 (16bit)
字0	Slave_1004 [...]	EtherCAT	2	1004	Sample AD0 0	2
字2	Slave_1004 [...]	EtherCAT	2	1004	Sample AD1 0	2
字4	Slave_1004 [...]	EtherCAT	2	1004	Sample AD2 0	2
字6	Slave_1004 [...]	EtherCAT	2	1004	Sample AD3 0	2
字8	Slave_1005 [...]	EtherCAT	2	1005	General Inpu...	1
字9	Slave_1003 [...]	EtherCAT	2	1003	Encoder1_Val	2

扩展RxPDO

RxPDO地址	设备名称	端口类型	端口号	从站地址	变量名称	数据大小 (16bit)
字0	Slave_1004 [...]	EtherCAT	2	1004	Sample DA0	2
字2	Slave_1004 [...]	EtherCAT	2	1004	Sample DA1	2
字4	Slave_1004 [...]	EtherCAT	2	1004	ADDA Control	1
字5	Slave_1005 [...]	EtherCAT	2	1005	General Outp...	1

图 7.6. 例程 6 硬件的节点及参数配置

注意：闭环步进电机驱动器需要手动添加轴；定位模块的 IO 需要手动添加映射关系；编码器模块的变量需要手动添加至 PDO 中；松下伺服电机驱动器、闭环步进电机驱动器的 PhysicalOutputs 和 Bit Mask 需要手动添加至从站的过程数据中。操作过程详见前述。

完成所有节点配置后，点击图 7.6 界面右上角的“下载配置文件”，即将配置文件下载至运动控制卡 DMC E3032 上的 flash 芯片中保存。如果硬件不变动，就不需要再用 Motion 软件做配置工作。

配置文件经 Motion 软件测试无误，点击图 7.6 左上角的“导出配置文件”，可将配置参数保存至硬盘指定位置，如图 7.7 所示。配置文件有 2 个，一个是 ini 文件一个是 eni 文件。

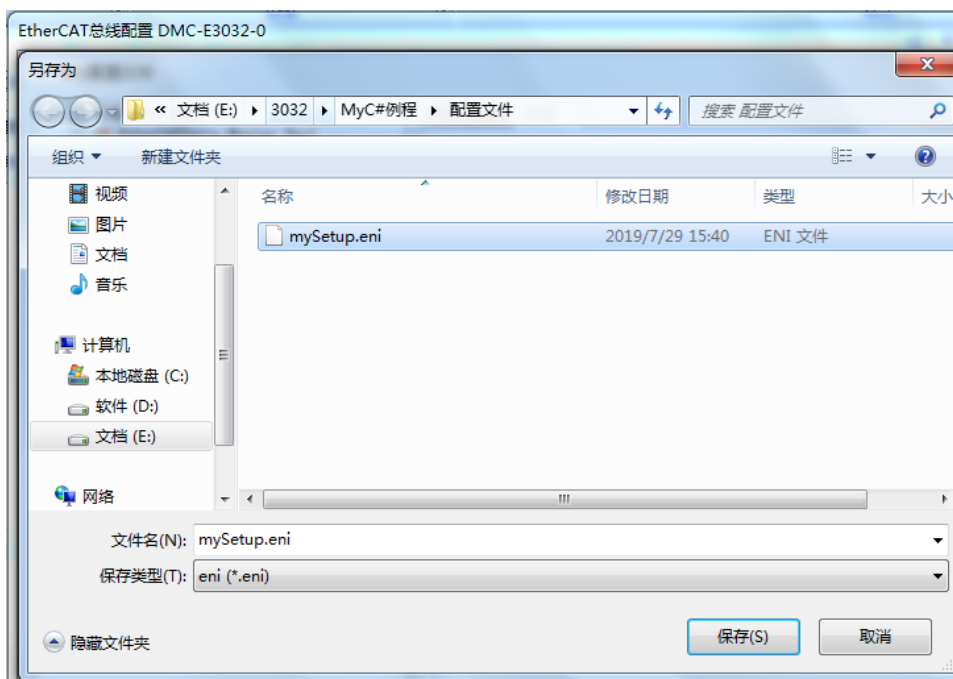


图 7.7. 保存配置文件至硬盘

在进行设备批量生产时，不需要用 Motion 软件对每台设备进行参数配置。在应用程序中，使用函数 `dmc_download_memfile(.....)` 可将配置文件下载至运动控制卡 DMC E3032 上。代码详见附录例程 6 中的函数 `downloadButton_Click(...)`。

三) 曲线插补及各模块的应用

在闭环步进电机驱动器输出 03 上接有一个 LED 灯（参见图 5.2）；在伺服电机驱动器 S01 上接有一个 LED 灯（参见图 3.2）。

AD/DA 扩展模块上的 D/A 的 0 号通道输出电压已用导线接入 A/D 的 1 号通道；编码器扩展模块 1 号接口与闭环步进电机编码器相连（参见图 6.17）。

例程 6 的程序界面如图 7.8 所示，程序代码见附录中的例程 6。

该程序可实现的主要功能及编程要点如下：

1. 平面圆弧插补

使用 `LTDMC.dmc_set_vector_profile_unit(...)` 指令设置插补运动的矢量速度，使用基于圆心的圆弧插补指令 `LTDMC.dmc_arc_move_center_unit(...)` 进行圆弧插补。24 段圆弧组成一

个完整的圆。如图 7.8 所示。

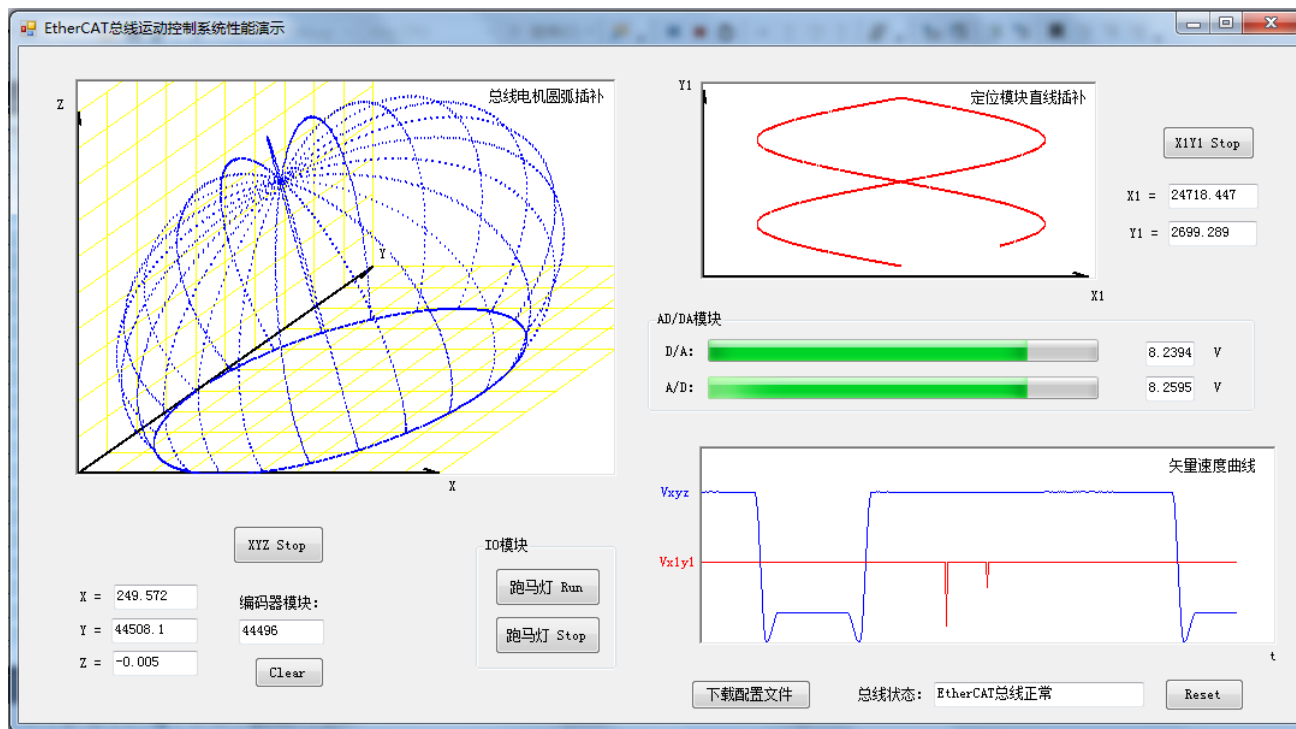


图 7.8. 例程 6 的界面及运行结果

在进行平面圆弧插补时，闭环步进电机驱动器输出口 O3 上的 LED 灯亮；平面圆弧插补结束后，该 LED 灯灭。

代码详见附录例程 6 中的函数 startButton_Click(...)。

2. 空间圆弧插补

使用基于中间点的圆弧插补指令 LDMC.dmc_arc_move_3points_unit(...) 进行空间圆弧插补，中间点的坐标为 (40000, 20000, 70000)。如图 7.8 所示，24 条空间圆弧都经过该点。

在等待空间圆弧插补结束时，要查询 3 个轴的电机状态，只有 3 个电机都停止运动，插补运动才结束。

在进行空间圆弧插补时，伺服电机驱动器输出口 S01 上的 LED 灯亮；空间圆弧插补结束后，该 LED 灯灭。

代码详见附录例程 6 中的函数 startButton_Click(...)。

3. 定位扩展模块的直线插补

采用多线程方式控制定位扩展模块的运动以提高程序的运行效率。

首先，将 2 条正弦曲线分为 60 段，将坐标保存在数组中。

然后，执行第一条线段的直线插补。该段插补速度由低速到高速，但停止速度和最大速度相同：

```
dmc_set_vector_profile_unit(cardID, 1, startVelY1, maxVelY1, TaccY1, TdecY1, maxVelY1);
```

之后各线段的插补速度：开始速度、最大速度、停止速度都与最大速度相同：

```
dmc_set_vector_profile_unit(cardID, 1, maxVelY1, maxVelY1, TaccY1, TdecY1, maxVelY1);
```

结束插补运动是使用带减速的急停指令。这样可以提高插补运动的连续性。代码详见附录

例程 6 中的函数 `start2Button_Click(...)`、`DrawXlYlcoordinate()` 和 `stop2Button_Click(...)`。

4. 插补速度的计算与显示

DMCE3032 运动控制卡由插补运动矢量速度的设置指令，但没有读取矢量速度的指令。因此要得到插补运动的矢量速度，需要进行计算。

将参与插补运动的各轴当前速度读出，然后矢量速度计算如下：

$$V = \text{Math.Sqrt}(\text{speedX} * \text{speedX} + \text{speedY} * \text{speedY} + \text{speedZ} * \text{speedZ});$$

代码详见附录例程 6 中的函数 `timer4_Tick(...)`。定时器 `timer4` 的定时周期为 15 毫秒。

5. 轴测图的实现

图 7.7 中的空间圆弧插补曲线显示在轴测图中，接近于人的视觉习惯，富有立体感。

将空间上的一个点 $P(x, y, z)$ 投影到图 7.7 中所示的轴测图上的公式如下：

$$\begin{cases} X = z + y \cos(a) \\ Y = z + y \sin(a) \end{cases}$$

式中： X, Y 是屏幕的坐标。 a 是轴测图中 x 轴与 y 轴的夹角。

代码详见附录例程 6 中的函数 `timer2_Tick(...)`。

6. IO 扩展模块输出跑马灯

使用 C 语言中的位运算，很容易实现跑马灯功能。其中用到了左移和非运算。

跑马灯程序也使用了多线程。代码详见附录例程 6 中的函数 `LEDrunButton_Click(...)`、`LED()` 和 `LEDstopButton_Click(...)`。

7. DA 和 AD 转换

DA 转换输出的数据与定位模块中的 X 轴坐标成正比；AD 转换读取的是 DA 口输出的电压信号。

代码详见附录例程 6 中的函数 `timer2_Tick(...)`。定时器 `timer2` 的定时周期为 30 毫秒。

8. 配置文件下载及总线复位

八. 小结

例程 6 是一个无限循环的程序。运行该程序几十小时，未发生一次总线异常现象。说明 DMC E3032 运动控制卡工作稳定可靠。

最令人担心的总线电机多轴插补功能表现良好，与脉冲式运动控制卡的性能相当。证明 EtherCAT 总线通讯速度确实快！

定位扩展模块用直线插补实现曲线轨迹运动的方法可行。但速度较大时，在线段衔接处有时会有明显的抖动，如图 7.7 矢量速度曲线所示。具有连续插补功能的总线型运动控制卡正在研发之中，期待其早日面市。

从表 8.1 的 EtherCAT 总线型运动控制卡、控制器与脉冲型运动控制卡的性能参数对比中可看出，采用 EtherCAT 总线控制系统方案，不仅是系统接线简单、方便，其运动控制性能比脉冲型运动控制卡也有很大的提升，电机数量、扩展模块的数量明显占优。只是各节点元件的总线参数配置相对麻烦一些。

表 8.1. EtherCAT 总线运动控制卡、控制器与脉冲型运动控制卡的对比

	3000 系列脉冲卡	E3032 总线卡	BAC332E 控制器
单个产品最大轴数	12 轴	32 轴	32 轴
电机信号线	4 条信号线	1 条网线	1 条网线
插补指令插补周期	250 微秒	250 微秒 (16 轴) 500 微秒 (32 轴)	250 微秒 (16 轴) 500 微秒 (32 轴)
直线插补轴数	12 轴	16 轴	16 轴
空间圆弧插补	无	有	有
空间螺旋线插补	无	有	有
PVT 函数	有	无	有
编码器接口	8 个	2 个	1 个
手轮接口	1 个	占用编码器接口	无
板上 IO 数量	16 IN, 16 OUT	8 IN, 8 OUT	12 IN, 12 OUT
可扩展模块	IO 模块、AD/DA 模块 (CAN 总线)	IO 模块、AD/DA 模块、 编码器模块 (EtherCAT)	IO 模块、AD/DA 模块、 编码器模块 (EtherCAT)
扩展模块最大数量	8	32	32
安装驱动软件	需要	需要	不需要 (脱机方式) 需要 (API 方式)
配置各节点总线参数	不需要	需要	需要

九. 参考文献

- [1] 郇极, 刘艳强. 工业以太网现场总线 EtherCAT 驱动程序设计及应用. 北京航空航天大学出版社, 2010 年 3 月
- [2] 深圳市雷赛控制技术有限公司. 雷赛 DMC-E3032 V2.0 运动控制卡用户使用手册. 2018 年 9 月
- [3] 松下电器产业株式会社汽车电子和机电系统公司. 技术资料 MINAS-A6B 系列基本功能规格篇. 2019 年 3 月
- [4] 松下电器产业株式会社汽车电子和机电系统公司. 技术资料 MINAS-A6B 系列 EtherCAT 通信规格篇. 2019 年 3 月
- [5] 深圳市雷赛智能控制股份有限公司. DM3E 系列步进驱动器用户手册 V2.02. 2018 年 8 月
- [6] 深圳市雷赛智能控制股份有限公司. CL3-EC 系列总线型闭环步进驱动器用户手册 V1.1. 2019 年 4 月
- [7] 深圳市雷赛控制技术有限公司. EtherCAT 总线 IO 扩展模块 EM32DX-E4 用户手册 V3.0. 2018 年 11 月
- [8] 深圳市雷赛控制技术有限公司. EM02DP-E1 定位模块手册 V2.0. 2017 年 7 月
- [9] 深圳市雷赛控制技术有限公司. EM06AX-E1 模拟量模块用户手册 V2.0. 2017 年 7 月
- [10] 深圳市雷赛控制技术有限公司. EM03DE-E4 编码器模块用户手册 V2.0. 2017 年 7 月

十. 附录例程代码

例程 1：伺服电机测试程序

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.Threading;
using csLTDMC; // 雷赛运动控制卡库函数

namespace OneACmotor
{
    public partial class Form1 : Form
    {
        ushort cardID = 0, axis = 0;           // 卡号, 轴号
        double equiv = 8388608.0 / 36000.0;     // 脉冲当量 8388608/36000
        double startVel = 1000;                 // 起始速度
        double maxVel = 136000;                 // 运行速度
        double Tacc = 0.1;                      // 加速时间
        double Tdec = 0.1;                      // 减速时间
        double stopVel = 1000;                  // 停止速度
        double S_para = 0.05;                   // S 段时间
        double dist = 36000;                     // 目标位置
        ushort posi_mode = 0;                    // 运动模式 0: 相对坐标模式
        double newSpeed = 0;
        int motorRun = 0;
        public Form1()
        {
            InitializeComponent();
        }

        private void Form1_Load(object sender, EventArgs e) // 控制卡初始化
        {
            short num;
            num = LTDMC.dmc_board_init(); // 获取卡数量
            if (num <= 0 || num > 8)
            {
                MessageBox.Show("初始卡失败!", "出错");
            }
            LTDMC.dmc_set_equiv(cardID, axis, equiv); // 设置脉冲当量
            LTDMC.dmc_set_s_profile(cardID, axis, 0, S_para); // 设置 S 段速度参数
            LTDMC.dmc_set_dec_stop_time(cardID, axis, Tdec); // 设置减速停止时间
            LTDMC.nmc_set_axis_enable(cardID, axis); // 轴使能
            timer2.Start();
        }

        private void Form1_FormClosing(object sender, FormClosingEventArgs e)
        {
        }
    }
}
```

```
LTDMC.nmc_set_axis_disable(cardID, axis);           // 关闭轴使能
LTDMC.dmc_board_close();
}

private void button5_Click(object sender, EventArgs e) // 点位运动到指定位置
{
    double degree;
    degree = double.Parse(textBox3.Text);
    degree = degree * 100.0;
    LTDMC.dmc_set_profile_unit(cardID, axis, 1000, 36000, 0.1, 0.1, 1000);
    LTDMC.dmc_pmove_unit(cardID, axis, degree, 0); // 定长运动
}

private void button2_Click(object sender, EventArgs e) // 电机连续正反转
{
    motorRun = 1;           // 电机连续正反转标志
    button2.Visible = false;
    while (motorRun == 1)
    {
        LTDMC.dmc_set_profile_unit(cardID, axis, startVel, 136000, Tacc, Tdec, stopVel);
        LTDMC.dmc_pmove_unit(cardID, axis, 2*dist, posi_mode); // 正转
        Check_INP(cardID, axis);           // 等待电机到位

        LTDMC.dmc_set_profile_unit(cardID, axis, startVel, 36000, Tacc, Tdec, stopVel);
        LTDMC.dmc_pmove_unit(cardID, axis, -1 * dist, posi_mode); // 反转
        Check_INP(cardID, axis);           // 等待电机到位;
    }
}

private void Check_INP(ushort cardID0, ushort axis0) // 等待电机到位
{
    int inINP;
    while (LTDMC.dmc_check_done(cardID0, axis0) == 0) // 等待运动指令结束
        Application.DoEvents();
    do           // 等待到位停止
    {
        Application.DoEvents();
        inINP = LTDMC.nmc_get_axis_io_in(cardID0, axis0);
        inINP = inINP & 0x1000000; // 按位与运算, 读取到位信号 INP
    }
    while (inINP == 0);
}

private void button3_Click(object sender, EventArgs e) // 电机连续正反转停止
{
    motorRun = 0;
    LTDMC.dmc_stop(cardID, axis, 0);           // 电机减速停止
    button2.Visible = true;
}

private void button1_MouseDown(object sender, MouseEventArgs e) // 定速运动
{
    LTDMC.dmc_set_profile_unit(cardID, axis, 1000, 36000, 0.1, 0.1, 1000);
    LTDMC.dmc_vmove(cardID, axis, 0);           // 正转
}
```

```
}

private void button1_MouseUp(object sender, MouseEventArgs e) // 电机减速停止
{
    LTDMC.dmc_stop(cardID, axis, 0);
}

private void button6_MouseDown(object sender, MouseEventArgs e) // 定速运动
{
    LTDMC.dmc_set_profile_unit(cardID, axis, 1000, 36000, 0.1, 0.1, 1000);
    LTDMC.dmc_vmove(cardID, axis, 1); // 反转
}

private void button6_MouseUp(object sender, MouseEventArgs e) // 电机减速停止
{
    LTDMC.dmc_stop(cardID, axis, 0);
}

private void button10_Click(object sender, EventArgs e) // 定速运动
{
    double newSpeed;
    newSpeed = double.Parse(textBox4.Text);
    newSpeed = newSpeed * 36000.0;
    LTDMC.dmc_set_profile_unit(cardID, axis, 1000, newSpeed, 0.1, 0.1, 1000);
    LTDMC.dmc_vmove(cardID, axis, 1);
    button10.Visible = false;
}

private void button11_Click(object sender, EventArgs e) // 电机减速停止
{
    LTDMC.dmc_stop(cardID, axis, 0);
    button10.Visible = true;
}

private void button4_Click(object sender, EventArgs e) // 速度增加 10%
{
    LTDMC.dmc_read_current_speed_unit(cardID, axis, ref newSpeed); // 读取当前速度
    newSpeed = 1.1 * newSpeed;
    LTDMC.dmc_change_speed_unit(cardID, axis, newSpeed, 0.1); // 在线变速
}

private void button9_Click(object sender, EventArgs e) // 速度减小 10%
{
    LTDMC.dmc_read_current_speed_unit(cardID, axis, ref newSpeed);
    newSpeed = 0.9 * newSpeed;
    LTDMC.dmc_change_speed_unit(cardID, axis, newSpeed, 0.1);
}

private void button12_Click(object sender, EventArgs e) // 回原点运动
{

```

```
LTDMC.nmc_set_home_profile(cardID, axis, 21, 1000, 36000, 0.1, 0.1, 0);
LTDMC.dmc_home_move(cardID, axis);
Check_INP(cardID, axis); // 等待电机到位
LTDMC.dmc_set_position_unit(cardID, axis, 0); // 指定轴的指令位置清零
LTDMC.dmc_set_encoder_unit(cardID, axis, 0); // 指定轴的编码器位置清零
}

private void checkBox6_CheckedChanged(object sender, EventArgs e) //Output1 开启或关闭
{
    uint output1;
    output1 = LTDMC.nmc_get_axis_io_out(cardID, 0); // 读取输出口
    if (checkBox6.Checked == true)
        output1 = output1 | 0x10000; // 按位或运算, 第 16 位置 1
    else
        output1 = output1 & 0xFFFFEFFF; // 按位与运算, 第 16 位清 0
    LTDMC.nmc_set_axis_io_out(cardID, axis, output1);
}

private void timer2_Tick(object sender, EventArgs e)
{
    ushort errcode = 0;
    double pos = 0;
uint axisIn0, input1, input2, input3, input4, input5;
    uint inORG, inPOT, inNOT;

    axisIn0 = LTDMC.nmc_get_axis_io_in(cardID, 0); // 读取输入口
    inNOT = axisIn0 & 0x1; // 解析各输入信号
    inPOT = axisIn0 & 0x2;
    inORG = axisIn0 & 0x4;
    input1 = axisIn0 & 0x80000;
    input2 = axisIn0 & 0x100000;
    input3 = axisIn0 & 0x200000;
    input4 = axisIn0 & 0x400000; // 由 ALM 信号触发
    input5 = axisIn0 & 0x800000; // 由 S-RDY 信号触发

    if (input1 == 0)
        checkBox1.Checked = false;
    else
        checkBox1.Checked = true;
    if (input2 == 0)
        checkBox2.Checked = false;
    else
        checkBox2.Checked = true;
    if (input3 == 0)
        checkBox3.Checked = false;
    else
        checkBox3.Checked = true;
    if (input4 == 0)
        checkBox4.Checked = false;
    else
        checkBox4.Checked = true;
    if (input5 == 0)
        checkBox5.Checked = false;
    else
        checkBox5.Checked = true;
    if (inORG == 0)
```

```
        checkBox7.Checked = false;
    else
        checkBox7.Checked = true;
    if (inPOT == 0)
        checkBox8.Checked = false;
    else
        checkBox8.Checked = true;
    if (inNOT == 0)
        checkBox9.Checked = false;
    else
        checkBox9.Checked = true;

    LTDMC.dmc_get_encoder_unit(cardID, axis, ref pos); // 读取指定轴的编码器位置
    textBox1.Text = pos.ToString();

    LTDMC.nmc_get_errcode(cardID, 2, ref errcode); // 读取总线状态
    if (errcode == 0)
    {
        textBox2.Text = "EtherCAT 总线正常";
        textBox2.BackColor = Color.White;
    }
    else
    {
        textBox2.Text = "EtherCAT 总线出错";
        textBox2.BackColor = Color.Red;
    }
}

private void button7_Click(object sender, EventArgs e) // 位置清零
{
    LTDMC.dmc_set_encoder_unit(cardID, axis, 0); // 指定轴的编码器位置清零
}

private void button8_Click(object sender, EventArgs e) // 总线复位
{
    textBox2.BackColor = Color.White;
    timer2.Stop();
    LTDMC.dmc_cool_reset(0); // 总线卡冷复位
    LTDMC.dmc_board_close();

    for (int i = 100; i > 0; i--) // 总线卡硬件复位耗时 10s 左右
    {
        textBox2.Text = "总线卡硬件复位中 .... " + i.ToString();
        Application.DoEvents();
        Thread.Sleep(100); // 延时 100 毫秒
    }

    Form1_Load(sender, e); // 控制卡初始化
    MessageBox.Show("总线卡硬件复位完成, 请确认总线状态。");
    timer2.Start();
}
}
```

例程 2：步进电机及驱动器测试程序

```
using System;
using System.Windows.Forms;
using System.Threading;
using csLTDMC;

namespace SteppingMotor
{
    public partial class Form1 : Form
    {
        ushort cardID = 0, axis = 0;           // 卡号, 轴号
        ushort PortNum = 2;                     // EtherCAT端口号
        ushort NodeNum = 1001;                  // 从站EtherCAT地址
        ushort Index, SubIndex = 0, ValLength = 32;
        double equiv = 100.0;                   // 脉冲当量 100
        double startVel = 100;                  // 起始速度
        double maxVel = 3600;                   // 运行速度
        double Tacc = 0.1;                      // 加速时间
        double Tdec = 0.1;                      // 减速时间
        double stopVel = 100;                   // 停止速度
        double S_para = 0.05;                   // S段时间
        ushort posi_mode = 0;                   // 运动模式0: 相对坐标模式

        public Form1()
        {
            InitializeComponent();
        }

        private void button1_Click(object sender, EventArgs e) // 读对象字典的参数
        {
            int value = 0;
            Index = 0x2000;
            LTDMC.nmc_get_node_od(cardID, PortNum, NodeNum, Index, SubIndex, ValLength, ref value);
            textBox1.Text = value.ToString();
            Index = 0x2001;
            LTDMC.nmc_get_node_od(cardID, PortNum, NodeNum, Index, SubIndex, ValLength, ref value);
            textBox2.Text = value.ToString();
            Index = 0x2051;
            LTDMC.nmc_get_node_od(cardID, PortNum, NodeNum, Index, SubIndex, ValLength, ref value);
            textBox3.Text = value.ToString();
            Index = 0x2048;
            LTDMC.nmc_get_node_od(cardID, PortNum, NodeNum, Index, SubIndex, ValLength, ref value);
            textBox4.Text = value.ToString();
        }

        private void button2_Click(object sender, EventArgs e) // 写对象字典的参数
        {
            int i, value = 0;
            LTDMC.nmc_set_axis_disable(cardID, axis);           // 轴关使能
            Index = 0x2000;
            value = int.Parse(textBox1.Text);
            LTDMC.nmc_set_node_od(cardID, PortNum, NodeNum, Index, SubIndex, ValLength, value);
        }
    }
}
```

```
        Index = 0x2001;
        value = int.Parse(textBox2.Text);
        LTDMC.nmc_set_node_od(cardID, PortNum, NodeNum, Index, SubIndex, ValLength, value);
        Index = 0x2051;
        value = int.Parse(textBox3.Text);
        LTDMC.nmc_set_node_od(cardID, PortNum, NodeNum, Index, SubIndex, ValLength, value);
        Index = 0x1010;          // 将参数写入EEPROM
        value = 0x65766173;
        LTDMC.nmc_set_node_od(cardID, PortNum, NodeNum, Index, 0x04, ValLength, value);
    for (i = 10; i > 0; i--)
    {
        button2.Text = "写入中..." + i.ToString();
    this.Refresh();
        Thread.Sleep(1000);          // 延时1秒
    }
    button2.Text = "完成! ";
    this.Refresh();
    LTDMC.nmc_set_axis_enable(cardID, axis);          // 轴使能
    Thread.Sleep(1000);          // 延时1秒
    button2.Text = "写数据";
    this.Refresh();
}

private void button3_Click(object sender, EventArgs e) // 点位运动
{
    double degree;
    degree = double.Parse(textBox5.Text);
    LTDMC.dmc_set_profile_unit(cardID, axis, startVel, maxVel, Tacc, Tdec, stopVel);
    LTDMC.dmc_pmove_unit(cardID, axis, degree, posi_mode);
}

private void button4_Click(object sender, EventArgs e) // 回原点
{
    LTDMC.nmc_set_home_profile(cardID, axis, 21, startVel, maxVel, Tacc, Tdec, stopVel);
    LTDMC.dmc_home_move(cardID, axis);
}

private void button5_MouseDown(object sender, MouseEventArgs e) // Jog运动, 正方向
{
    LTDMC.dmc_set_profile_unit(cardID, axis, startVel, maxVel, Tacc, Tdec, stopVel);
    LTDMC.dmc_set_s_profile(cardID, axis, 0, S_para);
    LTDMC.dmc_ymove(cardID, axis, 1);
}

private void button5_MouseUp(object sender, MouseEventArgs e) // Jog运动减速停止
{
    LTDMC.dmc_stop(cardID, axis, 0);
}

private void button6_MouseDown(object sender, MouseEventArgs e) // Jog运动, 负方向
{
    LTDMC.dmc_set_profile_unit(cardID, axis, startVel, maxVel, Tacc, Tdec, stopVel);
    LTDMC.dmc_set_s_profile(cardID, axis, 0, S_para);
}
```

```
        LTDMC.dmc_vmove(cardID, axis, 0);
    }

private void button6_MouseUp(object sender, MouseEventArgs e) // Jog运动减速停止
{
    LTDMC.dmc_stop(cardID, axis, 0);
}

private void Form1_Load(object sender, EventArgs e)
{
    short num;
    num = LTDMC.dmc_board_init(); // 获取卡数量
    if (num <= 0 || num > 8)
    {
        MessageBox.Show("初始卡失败!", "出错");
    }
    LTDMC.dmc_set_equiv(cardID, axis, equiv); // 设置脉冲当量
    LTDMC.dmc_set_s_profile(cardID, axis, 0, S_para); // 设置S段速度参数
    LTDMC.dmc_set_dec_stop_time(cardID, axis, Tdec); // 设置减速停止时间
    timer1.Start();
}

private void Form1_FormClosing(object sender, FormClosingEventArgs e)
{
    LTDMC.nmc_set_axis_disable(cardID, axis); // 关闭轴使能
    LTDMC.dmc_board_close();
}

private void timer1_Tick(object sender, EventArgs e)
{
    double position = 0;
    LTDMC.dmc_get_position_unit(cardID, axis, ref position); // 读指令位置
    textBox6.Text = position.ToString();
}
}
```

例程 3：闭环步进电机及驱动器测试程序

```
using System;
using System.Windows.Forms;
using System.Threading;
using csLTDMC;

namespace ClosedLoopSteppingMotor
{
    public partial class Form1 : Form
    {
        ushort cardID = 0, axis = 1;           // 卡号, 轴号
        ushort PortNum = 2;                     // EtherCAT端口号
        ushort NodeNum = 1002;                  // 闭环步进电机驱动器节点号
        ushort Index, SubIndex = 0, ValLength = 32;
        double equiv = 1.0;                     // 脉冲当量 1
        double startVel = 1000;                 // 起始速度
        double maxVel = 100000;                 // 运行速度
        double Tacc = 0.1;                      // 加速时间
        double Tdec = 0.1;                      // 减速时间
        double stopVel = 1000;                  // 停止速度
        double S_para = 0.05;                   // S段时间
        ushort posi_mode = 0;                   // 运动模式0: 相对坐标模式

        public Form1()
        {
            InitializeComponent();
        }

        private void button1_Click(object sender, EventArgs e) // 读对象字典的参数
        {
            int value = 0;
            Index = 0x2000;
            LTDMC.nmc_get_node_od(cardID, PortNum, NodeNum, Index, SubIndex, ValLength, ref value);
            value = value * 100;
            textBox1.Text = value.ToString();
            Index = 0x2001;
            LTDMC.nmc_get_node_od(cardID, PortNum, NodeNum, Index, SubIndex, ValLength, ref value);
            textBox2.Text = value.ToString();
            Index = 0x2048;
            LTDMC.nmc_get_node_od(cardID, PortNum, NodeNum, Index, SubIndex, ValLength, ref value);
            value = (int)(value / 10.0);
            textBox4.Text = value.ToString();
        }

        private void button2_Click(object sender, EventArgs e) // 写对象字典的参数
        {
            int i, value = 0;
            LTDMC.nmc_set_axis_disable(cardID, axis);           // 轴关使能
            Index = 0x2000;
            value = (int)(int.Parse(textBox1.Text) / 100.0);
            LTDMC.nmc_set_node_od(cardID, PortNum, NodeNum, Index, SubIndex, ValLength, value);
            Index = 0x2001;
```

```
        value = int.Parse(textBox2.Text);
        LTDMC.nmc_set_node_od(cardID, PortNum, NodeNum, Index, SubIndex, ValLength, value);
        Index = 0x1010; // 将参数写入EEPROM
        value = 0x65766173;
        LTDMC.nmc_set_node_od(cardID, PortNum, NodeNum, Index, 0x04, ValLength, value);
    for (i = 10; i > 0; i--)
    {
        button2.Text = "写入中..." + i.ToString();
    this.Refresh();
        Thread.Sleep(1000); // 延时1秒
    }
    button2.Text = "完成! ";
    this.Refresh();
    LTDMC.nmc_set_axis_enable(cardID, axis); // 轴使能
    Thread.Sleep(1000); // 延时1秒
    button2.Text = "写数据";
    this.Refresh();
}

private void button3_Click(object sender, EventArgs e)
{
    double distance;
    distance = double.Parse(textBox5.Text);
    LTDMC.dmc_set_profile_unit(cardID, axis, startVel, maxVel, Tacc, Tdec, stopVel);
    LTDMC.dmc_pmove_unit(cardID, axis, distance, posi_mode); // 定长运动
}

private void button4_Click(object sender, EventArgs e) // 回原点
{
    LTDMC.nmc_set_home_profile(cardID, axis, 21, startVel, maxVel, Tacc, Tdec, stopVel);
    LTDMC.dmc_home_move(cardID, axis);
}

private void button5_MouseDown(object sender, MouseEventArgs e) // Jog运动, 正方向
{
    LTDMC.dmc_set_profile_unit(cardID, axis, startVel, maxVel, Tacc, Tdec, stopVel);
    LTDMC.dmc_set_s_profile(cardID, axis, 0, S_para);
    LTDMC.dmc_ymove(cardID, axis, 1);
}

private void button5_MouseUp(object sender, MouseEventArgs e) // Jog运动减速停止
{
    LTDMC.dmc_stop(cardID, axis, 0);
}

private void button6_MouseDown(object sender, MouseEventArgs e) // Jog运动, 负方向
{
    LTDMC.dmc_set_profile_unit(cardID, axis, startVel, maxVel, Tacc, Tdec, stopVel);
    LTDMC.dmc_set_s_profile(cardID, axis, 0, S_para);
    LTDMC.dmc_ymove(cardID, axis, 0);
}

private void button6_MouseUp(object sender, MouseEventArgs e) // Jog运动减速停止
```

```
{
    LTDMC.dmc_stop(cardID, axis, 0);
}

private void button7_Click(object sender, EventArgs e) // Reset
{
    LTDMC.dmc_set_position_unit(cardID, axis, 0); // 指定轴的指令位置清零
    LTDMC.dmc_set_encoder_unit(cardID, axis, 0); // 指定轴的编码器位置清零
}

private void checkBox3_CheckedChanged(object sender, EventArgs e) // 设置输出口EX-OUT1
{
    uint output1;
    output1 = LTDMC.nmc_get_axis_io_out(cardID, axis); // 读取输出口
    if (checkBox3.Checked == true)
        output1 = output1 | 0x40000; // 按位或运算, 第18位置1
    else
        output1 = output1 & 0xFFFBFFFF; // 按位与运算, 第18位清0
    LTDMC.nmc_set_axis_io_out(cardID, axis, output1); // 写输出口
}

private void checkBox4_CheckedChanged(object sender, EventArgs e) // 设置输出口EX-OUT2
{
    uint output1;
    output1 = LTDMC.nmc_get_axis_io_out(cardID, axis); // 读取输出口
    if (checkBox4.Checked == true)
        output1 = output1 | 0x80000; // 按位或运算, 第19位置1
    else
        output1 = output1 & 0xFFFF7FFFF; // 按位与运算, 第19位清0
    LTDMC.nmc_set_axis_io_out(cardID, axis, output1); // 写输出口
}

private void checkBox5_CheckedChanged(object sender, EventArgs e) // 设置输出口EX-OUT3
{
    uint output1;
    output1 = LTDMC.nmc_get_axis_io_out(cardID, axis); // 读取输出口
    if (checkBox5.Checked == true)
        output1 = output1 | 0x100000; // 按位或运算, 第20位置1
    else
        output1 = output1 & 0xFFEFFFFF; // 按位与运算, 第20位清0
    LTDMC.nmc_set_axis_io_out(cardID, axis, output1); // 写输出口
}

private void checkBox6_CheckedChanged(object sender, EventArgs e) // 设置输出口EX-OUT4
{
    uint output1;
    output1 = LTDMC.nmc_get_axis_io_out(cardID, axis); // 读取输出口
    if (checkBox6.Checked == true)
        output1 = output1 | 0x200000; // 按位或运算, 第21位置1
    else
        output1 = output1 & 0xFFDFFFFF; // 按位与运算, 第21位清0
    LTDMC.nmc_set_axis_io_out(cardID, axis, output1); // 写输出口
}
```

```
private void Form1_Load(object sender, EventArgs e) // 初始化总线卡
{
    short num;
    num = LTDMC.dmc_board_init(); // 获取卡数量
    if (num <= 0 || num > 8)
    {
        MessageBox.Show("初始卡失败!", "出错");
    }
    LTDMC.dmc_set_equiv(cardID, axis, equiv); // 设置脉冲当量
    LTDMC.dmc_set_s_profile(cardID, axis, 0, S_para); // 设置S段速度参数
    LTDMC.dmc_set_dec_stop_time(cardID, axis, Tdec); // 设置减速停止时间
    LTDMC.dmc_set_position_unit(cardID, axis, 0); // 指定轴的指令位置清零
    LTDMC.dmc_set_encoder_unit(cardID, axis, 0); // 指定轴的编码器位置清零
    LTDMC.nmc_set_axis_enable(cardID, axis); // 轴使能
    timer1.Start();
}

private void Form1_FormClosing(object sender, FormClosingEventArgs e)
{
    LTDMC.nmc_set_axis_disable(cardID, axis); // 关闭轴使能
    LTDMC.dmc_board_close(); // 关闭总线卡
}

private void timer1_Tick(object sender, EventArgs e)
{
    double position = 0;
    uint axisIn0, input1, input2;

    LTDMC.dmc_get_encoder_unit(cardID, axis, ref position); // 读取指定轴的编码器位置
    textBox6.Text = (position / 10.0).ToString();
    LTDMC.dmc_get_position_unit(cardID, axis, ref position); // 读取轴的脉冲指令位置
    textBox7.Text = position.ToString();

    axisIn0 = LTDMC.nmc_get_axis_io_in(cardID, axis); // 读取输入口
    input1 = axisIn0 & 0x200; // 读取第9位输入信号
    input2 = axisIn0 & 0x400; // 读取第10位输入信号
    if (input1 == 0)
        checkBox1.Checked = false;
    else
        checkBox1.Checked = true;
    if (input2 == 0)
        checkBox2.Checked = false;
    else
        checkBox2.Checked = true;
}
}
```

例程 4：定位模块测试程序

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using csLTDMC;           // 雷赛运动控制卡库函数

namespace motorModule
{
    public partial class Form1 : Form
    {
        ushort cardID = 0;           // 卡号
        ushort axisX = 0, axisY = 1; // 轴号
        double equiv = 1;             // 脉冲当量 1
        double startVel = 100;        // 起始速度
        double maxVel = 10000;        // 运行速度
        double Tacc = 0.3;             // 加速时间
        double Tdec = 0.3;             // 减速时间
        double stopVel = 100;         // 停止速度
        double S_para = 0.1;          // S段时间
        double distance = 64000;      // 目标位置
        ushort portN = 2;              // 端口号
        ushort dataL = 1;              // 数据长度
        int outputValue = 0;

        public Form1()
        {
            InitializeComponent();
        }

        private void Form1_Load(object sender, EventArgs e) // 运动控制卡初始化
        {
            short num;
            num = LTDMC.dmc_board_init(); // 获取卡数量
            if (num <= 0 || num > 8)
            {
                MessageBox.Show("初始卡失败!", "出错");
            }
            LTDMC.dmc_set_equiv(cardID, axisX, equiv); // 设置脉冲当量
            LTDMC.dmc_set_s_profile(cardID, axisX, 0, S_para); // 设置S段速度参数
            LTDMC.dmc_set_dec_stop_time(cardID, axisX, Tdec); // 设置减速停止时间
            LTDMC.nmc_set_axis_enable(cardID, axisX); // 轴使能
            LTDMC.dmc_set_equiv(cardID, axisY, equiv); // 设置脉冲当量
            LTDMC.dmc_set_s_profile(cardID, axisY, 0, S_para); // 设置S段速度参数
            LTDMC.dmc_set_dec_stop_time(cardID, axisY, Tdec); // 设置减速停止时间
            LTDMC.nmc_set_axis_enable(cardID, axisY); // 轴使能
            LTDMC.dmc_set_position_unit(cardID, axisX, 0);
            LTDMC.dmc_set_position_unit(cardID, axisY, 0);
            timer1.Start();
            timer2.Start();
        }
    }
}
```

```
private void Form1_FormClosing(object sender, FormClosingEventArgs e)
{
    LTDMC.dmc_board_close();
}

private void button1_Click(object sender, EventArgs e) // 点位运动
{
    distance = double.Parse(textBox1.Text);
    LTDMC.dmc_set_profile_unit(cardID, axisY, startVel, maxVel, Tacc, Tdec, stopVel);
    LTDMC.dmc_set_s_profile(cardID, axisY, 0, S_para);
    LTDMC.dmc_pmove_unit(cardID, axisY, distance, 0); // 运动模式=0, 相对坐标
}

private void button2_Click(object sender, EventArgs e) // 直线插补
{
    ushort[] axes = { axisX, axisY };
    double Xdestination = double.Parse(textBox2.Text);
    double Ydestination = double.Parse(textBox3.Text);
    double[] destination = { Xdestination, Ydestination };
    LTDMC.dmc_set_vector_profile_unit(cardID, 0, 100, 10000, 0.1, 0.1, 100);
    LTDMC.dmc_set_vector_s_profile(cardID, 0, 0, 0.05);
    LTDMC.dmc_line_unit(cardID, 0, 2, axes, destination, 1); // 运动模式=0, 绝对坐标
}

private void button3_Click(object sender, EventArgs e)
{
    // X、Y轴回原点, 模式5: 负向一次回零加反找
    LTDMC.nmc_set_home_profile(cardID, axisX, 5, startVel, maxVel, Tacc, Tdec, stopVel);
    LTDMC.dmc_home_move(cardID, axisX);
    LTDMC.nmc_set_home_profile(cardID, axisY, 5, startVel, maxVel, Tacc, Tdec, stopVel);
    LTDMC.dmc_home_move(cardID, axisY);
}

private void button4_MouseDown(object sender, MouseEventArgs e) // Jog运动, 负方向
{
    LTDMC.dmc_set_profile_unit(cardID, axisX, startVel, maxVel, Tacc, Tdec, stopVel);
    LTDMC.dmc_set_s_profile(cardID, axisX, 0, S_para);
    LTDMC.dmc_ymove(cardID, axisX, 0);
}

private void button4_MouseUp(object sender, MouseEventArgs e) // Jog运动减速停止
{
    LTDMC.dmc_stop(cardID, axisX, 0);
}

private void button5_MouseDown(object sender, MouseEventArgs e) // Jog运动, 正方向
{
    LTDMC.dmc_set_profile_unit(cardID, axisX, startVel, maxVel, Tacc, Tdec, stopVel);
    LTDMC.dmc_set_s_profile(cardID, axisX, 0, S_para);
    LTDMC.dmc_ymove(cardID, axisX, 1);
}

private void button5_MouseUp(object sender, MouseEventArgs e) // Jog运动减速停止
{
    LTDMC.dmc_stop(cardID, axisX, 0);
}

private void checkBox4_CheckedChanged(object sender, EventArgs e) // 输出端口1
{
}
```

```
if (checkBox4.Checked == true)
    LTDMC.dmc_write_outbit(cardID, 25, 0); // 1号端口输出为0, LED亮
else
    LTDMC.dmc_write_outbit(cardID, 25, 1); // 1号端口输出为1, LED灭
}

privatevoid checkBox5_CheckedChanged(object sender, EventArgs e) // 输出端口2
{
    if (checkBox5.Checked == true)
        LTDMC.dmc_write_outbit(cardID, 26, 0); // 6号端口输出为0, LED亮
    else
        LTDMC.dmc_write_outbit(cardID, 26, 1); // 6号端口输出为1, LED灭
}

privatevoid checkBox6_CheckedChanged(object sender, EventArgs e) // 输出端口3
{
    if (checkBox6.Checked == true)
        LTDMC.dmc_write_outbit(cardID, 27, 0); // 3号端口输出为0, LED亮
    else
        LTDMC.dmc_write_outbit(cardID, 27, 1); // 3号端口输出为1, LED灭
}

privatevoid timer1_Tick(object sender, EventArgs e) // 电机位置检测
{
    double Xpos = 0, Ypos = 0;
    LTDMC.dmc_get_position_unit(cardID, axisX, ref Xpos);
    textBox4.Text = Xpos.ToString();
    LTDMC.dmc_get_position_unit(cardID, axisY, ref Ypos);
    textBox5.Text = Ypos.ToString();
}

privatevoid timer2_Tick(object sender, EventArgs e) // 输入口检测
{
    if (LTDMC.dmc_read_inbit(cardID, 25) == 0)
        checkBox1.Checked = true;
    else
        checkBox1.Checked = false;
    if (LTDMC.dmc_read_inbit(cardID, 26) == 0)
        checkBox2.Checked = true;
    else
        checkBox2.Checked = false;
    if (LTDMC.dmc_read_inbit(cardID, 27) == 0)
        checkBox3.Checked = true;
    else
        checkBox3.Checked = false;
}
}
```

例程 5：编码器模块测试程序

```
public partial class Form1 : Form
{
    ushort cardID = 0, axis = 1;           // 卡号, 轴号
    ushort PortNum = 2;                    // EtherCAT端口号
    ushort NodeNum3 = 1003;                // 编码器模块点号
    ushort Index;
    double equiv = 1.0;                    // 脉冲当量 1
    double startVel = 1000;                 // 起始速度
    double maxVel = 50000;                  // 运行速度
    double Tacc = 0.1;                      // 加速时间
    double Tdec = 0.1;                      // 减速时间
    double stopVel = 1000;                  // 停止速度
    double S_para = 0.05;                  // S段时间
    ushort posi_mode = 0;                  // 运动模式0: 相对坐标模式

    public Form1()
    {
        InitializeComponent();
    }

    private void Form1_Load(object sender, EventArgs e)
    {
        short num;
        num = LTDMC.dmc_board_init();        // 获取卡数量
        if (num <= 0 || num > 8)
        {
            MessageBox.Show("初始卡失败!", "出错");
        }
        LTDMC.dmc_set_equiv(cardID, axis, equiv);           // 设置脉冲当量
        LTDMC.dmc_set_s_profile(cardID, axis, 0, S_para);   // 设置S段速度参数
        LTDMC.dmc_set_dec_stop_time(cardID, axis, Tdec);    // 设置减速停止时间
        LTDMC.dmc_set_position_unit(cardID, axis, 0);       // 指定轴的指令位置清零
        LTDMC.dmc_set_encoder_unit(cardID, axis, 0);        // 指定轴的编码器位置清零
        LTDMC.nmc_set_axis_enable(cardID, axis);            // 轴使能
        Index = 0x6001;                                     // 设置编码器1
        LTDMC.nmc_set_node_od(cardID, PortNum, NodeNum3, Index, 0x01, 8, 0);
                                                                // 0: AB相4倍频模式
        LTDMC.nmc_set_node_od(cardID, PortNum, NodeNum3, Index, 0x02, 8, 1);
                                                                // 1: 正方向
        LTDMC.nmc_set_node_od(cardID, PortNum, NodeNum3, Index, 0x03, 32, 0);
                                                                // 设置Encoder1的值为0
        timer1.Start();
    }

    private void Form1_FormClosing(object sender, FormClosingEventArgs e)
    {
        LTDMC.nmc_set_axis_disable(cardID, axis);           // 关闭轴使能
        LTDMC.dmc_board_close();
    }

    private void button1_Click(object sender, EventArgs e) // 开始测试编码器模块
    {
        double distance;
        int position, number, interval;
        distance = 10.0 * double.Parse(textBox1.Text);
    }
}
```

```
number = int.Parse(textBox4.Text);
position = int.Parse(textBox5.Text);
interval = int.Parse(textBox6.Text);
Index = 0x6201; // 设置锁存器1
LTDMC.nmc_set_node_od(cardID, PortNum, NodeNum3, Index, 1, 8, 1);
// 1: 清除状态标记和锁存寄存器值
LTDMC.nmc_set_node_od(cardID, PortNum, NodeNum3, Index, 2, 8, 0);
// 0: 单次锁存
LTDMC.nmc_set_node_od(cardID, PortNum, NodeNum3, Index, 3, 8, 1);
// 下降沿触发
Index = 0x6401; // 设置比较器1
LTDMC.nmc_set_node_od(cardID, PortNum, NodeNum3, Index, 1, 8, 1);
// 1: 清除缓冲区及比较状态
LTDMC.nmc_set_node_od(cardID, PortNum, NodeNum3, Index, 2, 8, 5);
// 5: 线性比较位置, 触发
LTDMC.nmc_set_node_od(cardID, PortNum, NodeNum3, Index, 3, 8, 1);
// 选择编码器1比较
LTDMC.nmc_set_node_od(cardID, PortNum, NodeNum3, Index, 4, 8, 0);
// 以低电平触发
LTDMC.nmc_set_node_od(cardID, PortNum, NodeNum3, Index, 5, 32, 300000);
// 触发后持续时间, 单位微秒
LTDMC.nmc_set_node_od(cardID, PortNum, NodeNum3, Index, 6, 32, position);
// 设置第1个比较点
LTDMC.nmc_set_node_od(cardID, PortNum, NodeNum3, Index, 7, 32, number);
// 设置比较点数量
LTDMC.nmc_set_node_od(cardID, PortNum, NodeNum3, Index, 8, 32, interval);
// 设置比较点间隔
LTDMC.dmc_set_profile_unit(cardID, axis, startVel, maxVel, Tacc, Tdec, stopVel);
LTDMC.dmc_pmove_unit(cardID, axis, distance, posi_mode); // 定长运动
}

private void button2_Click(object sender, EventArgs e) // 位置清零
{
    LTDMC.dmc_set_position_unit(cardID, axis, 0); // 指定轴的指令位置清零
    LTDMC.dmc_set_encoder_unit(cardID, axis, 0); // 指定轴的编码器位置清零
    Index = 0x6001;
    LTDMC.nmc_set_node_od(cardID, PortNum, NodeNum3, Index, 0x03, 32, 0);
    // 设置编码器1的值为0
}

private void button3_Click(object sender, EventArgs e) // 读锁存器
{
    int value = 0;
    LTDMC.nmc_read_txpdo_extra(cardID, PortNum, 11, 2, ref value);
    // 读锁存器1锁存的编码器1的值
    textBox7.Text = value.ToString();
}

private void timer1_Tick(object sender, EventArgs e)
{
    double position = 0;
    int value = 0;
    LTDMC.dmc_get_encoder_unit(cardID, axis, ref position); // 读取指定轴的编码器位置
    textBox2.Text = (position / 10.0).ToString();
    LTDMC.nmc_read_txpdo_extra(cardID, PortNum, 9, 2, ref value); // 读编码器1的值
    textBox3.Text = value.ToString();
}
}
```

例程 6：曲线插补及各模块的应用

```
using System;
using System.Drawing;
using System.Threading;
using System.Windows.Forms;
using System.IO;
using csLTDMC;

namespace Interpolation
{
    public partial class Form1 : Form
    {
        const double PI = 3.1415926f;
        ushort cardID = 0;           // 卡号
        ushort PortNum = 2;          // EtherCAT 端口号
        ushort NodeNum3 = 1003;      // 编码器模块点号
        ushort Index;
        ushort dataL = 2;            // AD/DA 数据长度
        ushort axisX = 0, axisY = 1, axisX1 = 2, axisY1 = 3, axisZ = 4;      // 轴号

        double equivX = 10.0;        // 脉冲当量 10，一圈的距离 3600
        double startVelX = 1000;     // 起始速度
        double maxVelX = 10000;      // 运行速度
        double TaccX = 0.1;          // 加速时间
        double TdecX = 0.1;          // 减速时间
        double stopVelX = 1000;      // 停止速度
        double S_para = 0.05;        // S 段时间

        double equivY = 400.0 / 36.0; // 脉冲当量 9，一圈的距离 3600
        double TdecY = 0.1;          // 减速时间

        double equivZ = 8388608.0 / 3600.0; // 脉冲当量 8388608/3600，一圈的距离 3600
        double TdecZ = 0.1;          // 减速时间

        double equivX1 = 10;         // 脉冲当量 10，一圈的距离 640
        double equivY1 = 10;         // 脉冲当量 10
        double startVelY1 = 100;     // 起始速度
        double maxVelY1 = 2000;      // 运行速度
        double TaccY1 = 0.1;         // 加速时间
        double TdecX1 = 0.1;         // 减速时间
        double TdecY1 = 0.1;         // 减速时间

        int t = 0, t0 = 0;
        double V0 = 0, V10 = 0, V, V1;
        int flagS = 0, flagS1 = 0;
        double alpha = PI * 35 / 180.0;

        private Thread runX1Y1; // 定义线程名 runX1Y1
        private Thread runLED;  // 定义线程名 runLED

        public Form1()
        {
            InitializeComponent();
        }
    }
}
```

```
private void Form1_Load(object sender, EventArgs e)
{
    short num = LTDMC.dmc_board_init();    // 获取卡数量
    if (num <= 0 || num > 8)
    {
        MessageBox.Show("初始卡失败!", "出错");
    }
    LTDMC.dmc_set_equiv(cardID, axisX, equivX); // 设置脉冲当量
    LTDMC.dmc_set_equiv(cardID, axisY, equivY); // 设置脉冲当量
    LTDMC.dmc_set_equiv(cardID, axisZ, equivZ); // 设置脉冲当量
    LTDMC.dmc_set_equiv(cardID, axisX1, equivX1); // 设置脉冲当量
    LTDMC.dmc_set_equiv(cardID, axisY1, equivY1); // 设置脉冲当量

    LTDMC.dmc_set_s_profile(cardID, axisX, 0, S_para);
    LTDMC.dmc_set_s_profile(cardID, axisY, 0, S_para);
    LTDMC.dmc_set_s_profile(cardID, axisZ, 0, S_para);    // 设置 S 段速度参数
    LTDMC.dmc_set_s_profile(cardID, axisX1, 0, S_para);
    LTDMC.dmc_set_s_profile(cardID, axisY1, 0, S_para);

    LTDMC.dmc_set_dec_stop_time(cardID, axisX, TdecX);
    LTDMC.dmc_set_dec_stop_time(cardID, axisY, TdecY);
    LTDMC.dmc_set_dec_stop_time(cardID, axisZ, TdecZ);    // 设置减速停止时间
    LTDMC.dmc_set_dec_stop_time(cardID, axisX1, TdecX1);
    LTDMC.dmc_set_dec_stop_time(cardID, axisY1, TdecY1);
    LTDMC.nmc_set_axis_enable(cardID, axisX);
    LTDMC.nmc_set_axis_enable(cardID, axisY);
    LTDMC.nmc_set_axis_enable(cardID, axisZ);                // 轴使能
    LTDMC.nmc_set_axis_enable(cardID, axisX1);
    LTDMC.nmc_set_axis_enable(cardID, axisY1);

    Index = 0x6001;    // 设置编码器模块的编码器 1
    LTDMC.nmc_set_node_od(cardID, PortNum, NodeNum3, Index, 0x01, 8, 0);
    // 0: AB 相 4 倍频模式
    LTDMC.nmc_set_node_od(cardID, PortNum, NodeNum3, Index, 0x02, 8, 1);
    // 1: 正方向
    LTDMC.nmc_set_node_od(cardID, PortNum, NodeNum3, Index, 0x03, 32, 0);
    // 设置 Encoder1 的值为 0
    LTDMC.nmc_write_rxpdo_extra(cardID, PortNum, 4, dataL, 1); // AD/DA 接口使能打开
    clearButton_Click(sender, e);    // 设置起始位置为 0
    timer1.Start();
    timer2.Start();
    timer3.Start();
}

private void startButton_Click(object sender, EventArgs e)
{
    int i, X1, Y1, X2, Y2;
    double theta = PI / 12.0;
    ushort AxisNum = 3;    // 插补轴数
    ushort[] axesList = new ushort[] { axisX, axisY, axisZ };
    double[] targetPosition = new double[3];
    double[] middlePosition = new double[3];
    double[] centerPositon = new double[] { 40000, 40000 };
    uint outputY = 0, outputZ = 0;

    startButton.Visible = false;
    timer4.Start();
}
```

```
flagS = 1;
while (flagS == 1)
{
    DrawXYcoordinate();
    LTDMC.dmc_set_position_unit(cardID, axisX, 80000);
    LTDMC.dmc_set_position_unit(cardID, axisY, 40000);
    LTDMC.dmc_set_encoder_unit(cardID, axisY, 40000);
    LTDMC.dmc_set_position_unit(cardID, axisZ, 0);
    LTDMC.dmc_set_encoder_unit(cardID, axisZ, 0);          // 设置起始位置
    Index = 0x6001;
    LTDMC.nmc_set_node_od(cardID, PortNum, NodeNum3, Index, 0x03, 32,
        (int)(40000 * 40.0 / 36.0)); // 设置 Encoder1 的值
    middlePosition[0] = 40000;
    middlePosition[1] = 20000;
    middlePosition[2] = 70000;
    LTDMC.dmc_set_vector_s_profile(cardID, 0, 0, S_para); // 设置 S 段速度参数
    for (i = 0; i < 24; i++)
    {
        X1 = 40000 + (int)(40000 * Math.Cos(theta));
        Y1 = 40000 + (int)(40000 * Math.Sin(theta));
        X2 = 40000 + (int)(40000 * Math.Cos(theta + PI));
        Y2 = 40000 + (int)(40000 * Math.Sin(theta + PI));
        targetPosition[0] = X1;
        targetPosition[1] = Y1;
        targetPosition[2] = 0;
        LTDMC.dmc_set_vector_profile_unit(cardID, 0, startVelX, maxVelX,
            TaccX, TdecX, stopVelX);
        LTDMC.dmc_arc_move_center_unit(cardID, 0, AxisNum, axesList,
            targetPosition, centerPositon, 1, 0, 1); // 圆弧插补
        outputY = LTDMC.nmc_get_axis_io_out(cardID, axisY);
        outputY = outputY | 0x40000;          // 按位或运算, 第 18 位置 1
        LTDMC.nmc_set_axis_io_out(cardID, axisY, outputY); // Y 轴 out18LED 亮
        while (LTDMC.dmc_check_done(cardID, axisX) == 0 ||
            LTDMC.dmc_check_done(cardID, axisY) == 0 ||
            LTDMC.dmc_check_done(cardID, axisZ) == 0)
        {
            // 等待运动停止
            Application.DoEvents();
            Thread.Sleep(10);
            if (flagS == 0) break;
        }
        Thread.Sleep(50);          // 延时 50ms
        outputY = LTDMC.nmc_get_axis_io_out(cardID, axisY);
        outputY = outputY & 0xFFFBFFFF;      // 按位与运算, 第 18 位清 0
        LTDMC.nmc_set_axis_io_out(cardID, axisY, outputY); // Y 轴 out18LED 灭
        targetPosition[0] = X2;
        targetPosition[1] = Y2;
        targetPosition[2] = 0;
        LTDMC.dmc_set_vector_profile_unit(cardID, 0, startVelX, 5 * maxVelX,
            TaccX, TdecX, stopVelX);
        LTDMC.dmc_arc_move_3points_unit(cardID, 0, AxisNum, axesList,
            targetPosition, middlePosition, -1, 1);
        outputZ = LTDMC.nmc_get_axis_io_out(cardID, axisZ);
        outputZ = outputZ | 0x10000;          // 按位或运算, 第 16 位置 1
        LTDMC.nmc_set_axis_io_out(cardID, axisZ, outputZ); // Z 轴 out16LED 亮
        while (LTDMC.dmc_check_done(cardID, axisX) == 0 ||
            LTDMC.dmc_check_done(cardID, axisY) == 0 ||
            LTDMC.dmc_check_done(cardID, axisZ) == 0)
        {
            // 等待运动停止
        }
    }
}
```

```
        Application.DoEvents();
        Thread.Sleep(10);
        if (flagS == 0) break;
    }
    Thread.Sleep(50); // 延时 50ms
    outputZ = LTDMC.nmc_get_axis_io_out(cardID, axisZ);
    outputZ = outputZ & 0xFFFEFFFF; // 按位与运算, 第 16 位清 0
    LTDMC.nmc_set_axis_io_out(cardID, axisZ, outputZ); // Z 轴 out16LED 灭
    theta = theta + PI / 12.0 + PI;
    if (flagS == 0) break;
}
}
startButton.Visible = true;
}

private void stopButton_Click(object sender, EventArgs e)
{
    flagS = 0;
    LTDMC.dmc_stop(cardID, axisX, 0);
    LTDMC.dmc_stop(cardID, axisY, 0);
    LTDMC.dmc_stop(cardID, axisZ, 0);
    if (flagS1 == 0)
        timer4.Stop();
}

private void clearButton_Click(object sender, EventArgs e)
{
    LTDMC.dmc_set_position_unit(cardID, axisX, 0);
    LTDMC.dmc_set_position_unit(cardID, axisY, 0);
    LTDMC.dmc_set_encoder_unit(cardID, axisY, 0);
    LTDMC.dmc_set_position_unit(cardID, axisZ, 0);
    LTDMC.dmc_set_encoder_unit(cardID, axisZ, 0);
    Index = 0x6001;
    LTDMC.nmc_set_node_od(cardID, PortNum, NodeNum3, Index, 0x03, 32, 0);
    // 设置 Encoder1 的值为 0
}

private void start2Button_Click(object sender, EventArgs e)
{
    start2Button.Visible = false;
    DrawX1Y1coordinate();
    runX1Y1 = new Thread(axesX1Y1); // 创建线程
    runX1Y1.Priority = ThreadPriority.Normal; // 该线程优先级一般
    runX1Y1.Start(); // 启动线程
    flagS1 = 1;
    timer4.Start();
}

private void stop2Button_Click(object sender, EventArgs e)
{
    flagS1 = 0;
    LTDMC.dmc_stop(cardID, axisX1, 0);
    LTDMC.dmc_stop(cardID, axisY1, 0);
    runX1Y1.Abort(); // 终止线程
    start2Button.Visible = true;
    if (flagS == 0)
        timer4.Stop();
}
```

```
private void LEDrunButton_Click(object sender, EventArgs e)
{
    LEDrunButton.Visible = false;
    runLED = new Thread(LED);           // 创建线程
    runLED.Priority = ThreadPriority.Normal; // 该线程优先级一般
    runLED.Start();                     // 启动线程
}

private void LEDstopButton_Click(object sender, EventArgs e)
{
    runLED.Abort();                     // 终止线程
    LEDrunButton.Visible = true;
}

private void downloadButton_Click(object sender, EventArgs e)
{
    OpenFileDialog f1 = new OpenFileDialog();
    f1.Filter = "ini|*.ini";
    DialogResult res1 = f1.ShowDialog();
    if (res1 == DialogResult.OK)
    {
        FileStream fs = File.Open(f1.FileName, FileMode.Open, FileAccess.Read,
            FileShare.ReadWrite);
        StreamReader sr = new StreamReader(fs);
        string str = sr.ReadToEnd();
        sr.Close();
        fs.Close();
        byte[] buffer = Encoding.UTF8.GetBytes(str);
        byte[] fileincontrol = Encoding.UTF8.GetBytes("");
        ushort filetype = 201;
        short ret = -1;
        ret = LTDMC.nmc_set_cycletime(cardID, 2, 500); //设置总线周期
        ret = LTDMC.dmc_download_memfile(cardID, buffer, (uint)buffer.Length,
            fileincontrol, filetype); //下载配置文件
        if (ret == 0)
        {
            MessageBox.Show(" ini 文件下载成功! \n\n 请继续下载 eni 文件。");
        }
    }

    OpenFileDialog f2 = new OpenFileDialog();
    f2.Filter = "eni|*.eni";
    DialogResult res2 = f2.ShowDialog();
    if (res2 == DialogResult.OK)
    {
        FileStream fs = File.Open(f2.FileName, FileMode.Open, FileAccess.Read,
            FileShare.ReadWrite);
        StreamReader sr = new StreamReader(fs);
        string str = sr.ReadToEnd();
        sr.Close();
        fs.Close();
        byte[] buffer = Encoding.UTF8.GetBytes(str);
        byte[] fileincontrol = Encoding.UTF8.GetBytes("");
        ushort filetype = 200;
        short ret = -1;
        ret = LTDMC.dmc_download_memfile(cardID, buffer, (uint)buffer.Length,
            fileincontrol, filetype); //下载 ENI 文件
    }
}
```

```
        if (ret == 0)
        {
            MessageBox.Show("eni 文件下载成功!");
        }
    }
}

private void reset_Click(object sender, EventArgs e)
{
    textBox1.BackColor = Color.White;
    timer1.Stop();
    LTDMC.dmc_cool_reset(0); // 总线卡冷复位
    LTDMC.dmc_board_close();
    for (int i = 100; i > 0; i--) // 总线卡硬件复位耗时 10s 左右
    {
        textBox1.Text = "总线卡硬件复位中 .... " + i.ToString();
        Application.DoEvents();
        Thread.Sleep(100); // 延时 100 毫秒
    }
    Form1_Load(sender, e); // 控制卡初始化
    MessageBox.Show("总线卡硬件复位完成, 请确认总线状态。");
}

private void Form1_FormClosed(object sender, FormClosedEventArgs e)
{
    LTDMC.nmc_set_axis_disable(cardID, axisX);
    LTDMC.nmc_set_axis_disable(cardID, axisY);
    LTDMC.nmc_set_axis_disable(cardID, axisZ); // 关闭轴使能
    LTDMC.dmc_board_close();
}

private void timer1_Tick(object sender, EventArgs e)
{
    ushort errcode = 0;
    LTDMC.nmc_get_errcode(cardID, 2, ref errcode); // 读取总线状态
    if (errcode == 0)
    {
        textBox1.Text = "EtherCAT 总线正常";
        textBox1.BackColor = Color.White;
    }
    else
    {
        textBox1.Text = "EtherCAT 总线出错";
        textBox1.BackColor = Color.Red;
    }
}

private void timer2_Tick(object sender, EventArgs e)
{
    double positionX = 0, positionY = 0, positionZ = 0, positionX1 = 0, positionY1 = 0;
    int x, y, Vencoder = 0, Vin = 0;
    Pen BluePen = new Pen(Color.Blue, 1);
    Pen RedPen = new Pen(Color.Red, 1);
    LTDMC.dmc_get_position_unit(cardID, axisX, ref positionX); // 读轴的脉冲指令位置
    LTDMC.dmc_get_encoder_unit(cardID, axisY, ref positionY); // 读轴的编码器位置
    LTDMC.dmc_get_encoder_unit(cardID, axisZ, ref positionZ); // 读轴的编码器位置
    LTDMC.dmc_get_position_unit(cardID, axisX1, ref positionX1); // 读轴的脉冲指令位置
```

```
LTDMC.dmc_get_position_unit(cardID, axisY1, ref positionY1); //读轴的脉冲指令位置
LTDMC.nmc_read_txpdo_extra(cardID, PortNum, 9, 2, ref Vencoder); //读 Encoder1
XtextBox.Text = positionX.ToString();
YtextBox.Text = positionY.ToString();
ZtextBox.Text = positionZ.ToString();
X1textBox.Text = positionX1.ToString();
Y1textBox.Text = positionY1.ToString();
EncoderTextBox.Text = (Vencoder * 36.0 / 40.0).ToString();
LTDMC.nmc_write_rxpdo_extra(cardID, PortNum, 0, dataL,
(int)(positionX1 * 10 / 3.0)); // D/A 转换, DA 的 0 号通道输出电压信号
DAprogressBar.Value = (int)(positionX1 / 300);
DAtextBox.Text = (positionX1 * 10 / 30000.0).ToString();
LTDMC.nmc_read_txpdo_extra(cardID, PortNum, 2, dataL, ref Vin); //读 AD 的 1 号通道
ADprogressBar.Value = (int)(Vin / 1000);
ADtextBox.Text = ((double)(Vin / 10000.0)).ToString();
if (flagS == 1)
{
    Graphics g = pictureBox1.CreateGraphics(); // 创建 Graphics 对象
    g.PageUnit = GraphicsUnit.Pixel; // 以像素为单位
    g.PageScale = 1.0f; // 像素与坐标单位的比例,即 1 个单位为 1 个像素。
    g.TranslateTransform(0, 325); // y 轴下移 512, 方向仍朝下
    x = (int)((positionX + positionY * Math.Cos(alpha)) * 3 / 1000.0);
    y = (int)((positionZ + positionY * Math.Sin(alpha)) * 3 / 1000.0);
    g.DrawEllipse(BluePen, x, -1 * y, 1, 1); //画各个点
}
if (flagS1 == 1)
{
    Graphics g2 = pictureBox2.CreateGraphics(); // 创建 Graphics 对象
    g2.PageUnit = GraphicsUnit.Pixel; // 以像素为单位
    g2.PageScale = 1.0f; // 像素与坐标单位的比例,即 1 个单位为 1 个像素。
    g2.TranslateTransform(0, 160); // y 轴下移 512, 方向仍朝下
    x = (int)(positionX1 / 100.0);
    y = (int)(positionY1 / 100.0);
    g2.DrawEllipse(RedPen, x, -1 * y, 1, 1); //画各个点
}
}

private void timer3_Tick(object sender, EventArgs e)
{
    DrawXYcoordinate();
    DrawX1Y1coordinate();
    timer3.Stop();
}

private void timer4_Tick(object sender, EventArgs e)
{
    double speedX = 0, speedY = 0, speedZ = 0, speedX1 = 0, speedY1 = 0;
    LTDMC.dmc_read_current_speed_unit(cardID, axisX, ref speedX);
    LTDMC.dmc_read_current_speed_unit(cardID, axisY, ref speedY);
    LTDMC.dmc_read_current_speed_unit(cardID, axisZ, ref speedZ);
    LTDMC.dmc_read_current_speed_unit(cardID, axisX1, ref speedX1);
    LTDMC.dmc_read_current_speed_unit(cardID, axisY1, ref speedY1);
    V = Math.Sqrt(speedX * speedX + speedY * speedY + speedZ * speedZ);
    V1 = Math.Sqrt(speedX1 * speedX1 + speedY1 * speedY1);
    Graphics g3 = pictureBox3.CreateGraphics(); // 创建 Graphics 对象
    g3.PageUnit = GraphicsUnit.Pixel; // 以像素为单位
    g3.PageScale = 1.0f; // 像素与坐标单位的比例,即 1 个单位为 1 个像素。
    g3.TranslateTransform(0, 160); // y 轴下移 512, 方向仍朝下
```

```
Pen BluePen = new Pen(Color.Blue, 1);
Pen RedPen = new Pen(Color.Red, 1);
g3.DrawLine(BluePen, t0, -1 * (int)(V0 / 400), t, -1 * (int)(V / 400)); //画各个点
g3.DrawLine(RedPen, t0, -1 * (int)(V10 / 30), t, -1 * (int)(V1 / 30)); //画各个点
if (t >= 480)
{
    t = 0;
    g3.Clear(Color.White); // 画布清零
}
t0 = t;
V0 = V;
V10 = V1;
t = t + 1;
}

private void DrawXYcoordinate()
{
    int i, X1, Y1, X2, Y2;
    Graphics g = pictureBox1.CreateGraphics(); // 创建 Graphics 对象
    g.PageUnit = GraphicsUnit.Pixel; // 以像素为单位
    g.PageScale = 1.0f; // 像素与坐标单位的比例, 即 1 个单位为 1 个像素。
    g.TranslateTransform(0, 325); // y 轴下移 512, 方向仍朝下
    g.Clear(Color.White); // 画布清零
    Pen YellowPen = new Pen(Color.Yellow);
    Pen BlackPen = new Pen(Color.Black, 2);
    for (i = 0; i <= 300; i = i + 30)
    {
        X1 = (int)(i * Math.Cos(alpha));
        Y1 = (int)(i * Math.Sin(alpha));
        X2 = 300 + (int)(i * Math.Cos(alpha));
        Y2 = (int)(i * Math.Sin(alpha));
        g.DrawLine(YellowPen, X1, -1 * Y1, X2, -1 * Y2); //画 XY 平面水平线
        X1 = i;
        Y1 = 0;
        X2 = i + (int)(300 * Math.Cos(alpha));
        Y2 = (int)(300 * Math.Sin(alpha));
        g.DrawLine(YellowPen, X1, -1 * Y1, X2, -1 * Y2); //画 XY 平面垂直线
        X1 = 0;
        Y1 = i;
        X2 = (int)(300 * Math.Cos(alpha));
        Y2 = i + (int)(300 * Math.Sin(alpha));
        g.DrawLine(YellowPen, X1, -1 * Y1, X2, -1 * Y2); //画 YZ 平面水平线
        X1 = (int)(i * Math.Cos(alpha));
        Y1 = (int)(i * Math.Sin(alpha));
        X2 = (int)(i * Math.Cos(alpha));
        Y2 = 300 + (int)(i * Math.Sin(alpha));
        g.DrawLine(YellowPen, X1, -1 * Y1, X2, -1 * Y2); //画 YZ 平面垂直线
    }
    g.DrawLine(BlackPen, 0, 0, 300, 0); // 画 X 轴
    g.DrawLine(BlackPen, 287, -4, 300, 0); // 画 X 轴
    X2 = (int)(300 * Math.Cos(alpha));
    Y2 = (int)(300 * Math.Sin(alpha));
    g.DrawLine(BlackPen, 0, 0, X2, -1 * Y2); // 画 Y 轴
    g.DrawLine(BlackPen, X2, -1 * Y2, X2 - 10, -1 * (Y2 - 10)); //画 Y 轴
    g.DrawLine(BlackPen, 0, 0, 0, -300); // 画 Z 轴
    g.DrawLine(BlackPen, 0, -300, 2, -289); // 画 Z 轴
}
```

```
private void DrawX1Y1coordinate()
{
    Graphics g2 = pictureBox2.CreateGraphics(); // 创建 Graphics 对象
    g2.PageUnit = GraphicsUnit.Pixel;           // 以像素为单位
    g2.PageScale = 1.0f;                         // 像素与坐标单位的比例,即 1 个单位为 1 个像素。
    g2.TranslateTransform(0, 160);              // y 轴下移 165 , 方向仍朝下
    g2.Clear(Color.White);                       // 画布清零
    Pen BlackPen = new Pen(Color.Black, 2);
    g2.DrawLine(BlackPen, 0, 0, 320, 0);         //画 X 轴
    g2.DrawLine(BlackPen, 307, -4, 320, 0);      //画 X 轴
    g2.DrawLine(BlackPen, 0, 0, 0, -155);        //画 Y 轴
    g2.DrawLine(BlackPen, 0, -155, 2, -144);     //画 Y 轴
}

private void axesX1Y1()
{
    int i, pointN = 0;
    double x0 = 165, y0 = 10, L = 140, A = 120, m = 30, K = 100;
    double X1 = 0, Y1, X0 = x0, Y0 = y0;
    ushort AxisNum2 = 2;                        // 插补轴数
    ushort[] axesList2 = new ushort[] { axisX1, axisY1 };
    double[] targetPosition2 = new double[2];
    double[] Xdata = new double[100];
    double[] Ydata = new double[100];
    for (Y1 = y0; Y1 <= y0 + L; Y1 = Y1 + L / m)
    {
        X1 = (int)(x0 - A * Math.Sin(2 * PI * (Y1 - y0) / L));
        Xdata[pointN] = X1 * K;
        Ydata[pointN] = Y1 * K;
        pointN++;
    }
    for (Y1 = y0 + L - L / m; Y1 >= y0; Y1 = Y1 - L / m)
    {
        X1 = (int)(x0 + A * Math.Sin(2 * PI * (Y1 - y0) / L));
        Xdata[pointN] = X1 * K;
        Ydata[pointN] = Y1 * K;
        pointN++;
    }
    Xdata[pointN] = Xdata[0];
    Ydata[pointN] = Ydata[0];
    DrawX1Y1coordinate();
    LTDMC.dmc_set_position_unit(cardID, axisX1, x0 * K);
    LTDMC.dmc_set_position_unit(cardID, axisY1, 0 * K);
    targetPosition2[0] = x0 * K;
    targetPosition2[1] = y0 * K;
    LTDMC.dmc_set_vector_profile_unit(cardID, 1, startVelY1, maxVelY1,
    TaccY1, TdecY1, maxVelY1);
    LTDMC.dmc_set_vector_s_profile(cardID, 1, 0, 0); // 设置 S 段速度参数
    LTDMC.dmc_line_unit(cardID, 1, AxisNum2, axesList2, targetPosition2, 1); //定长运动
    while (LTDMC.dmc_check_done(cardID, axisX1) == 0 ||
        LTDMC.dmc_check_done(cardID, axisY1) == 0)
        Thread.Sleep(10); // 等待运动停止
    while (flagS1 == 1)
    {
        DrawX1Y1coordinate();
        LTDMC.dmc_set_vector_profile_unit(cardID, 1, maxVelY1, maxVelY1,
        TaccY1, TdecY1, maxVelY1);
        for (i = 1; i <= 60; i++)
    }
```

```
        {
            targetPosition2[0] = Xdata[i];
            targetPosition2[1] = Ydata[i];
            LTDMC.dmc_line_unit(cardID, 1, AxisNum2, axesList2, targetPosition2, 1);
            // 定长运动
            while (LTDMC.dmc_check_done(cardID, axisX1) == 0 ||
                LTDMC.dmc_check_done(cardID, axisY1) == 0)
            { } // 等待运动停止
        }
    }
}

private void LED()
{
    uint OutputVal = 0X100, NotOutputVal;
    while (true)
    {
        NotOutputVal = ~OutputVal; // 非运算
        LTDMC.dmc_write_outport(cardID, 0, NotOutputVal); // OutputVal=0~31
        Thread.Sleep(200); // 延时 200 毫秒
        OutputVal = OutputVal << 1; // 左移 1 位
        if (OutputVal == 0x0)
        {
            OutputVal = 0X100;
        }
    }
}
}
```