锐特 RTE3128 总线运动控制卡

用户使用手册

本手册仅供参考，由于产品改进设计和功能等原因，恕不另行通知。

深圳锐特控制技术有限公司

2021-12-8

第 1 章 产品概述

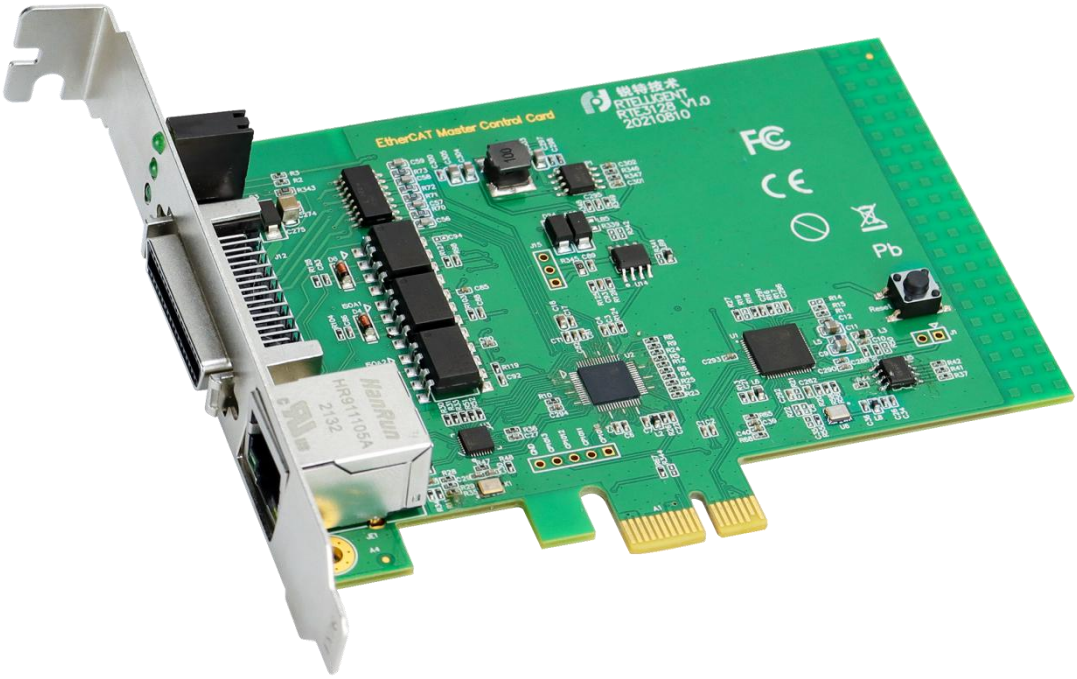


图 1 主卡图

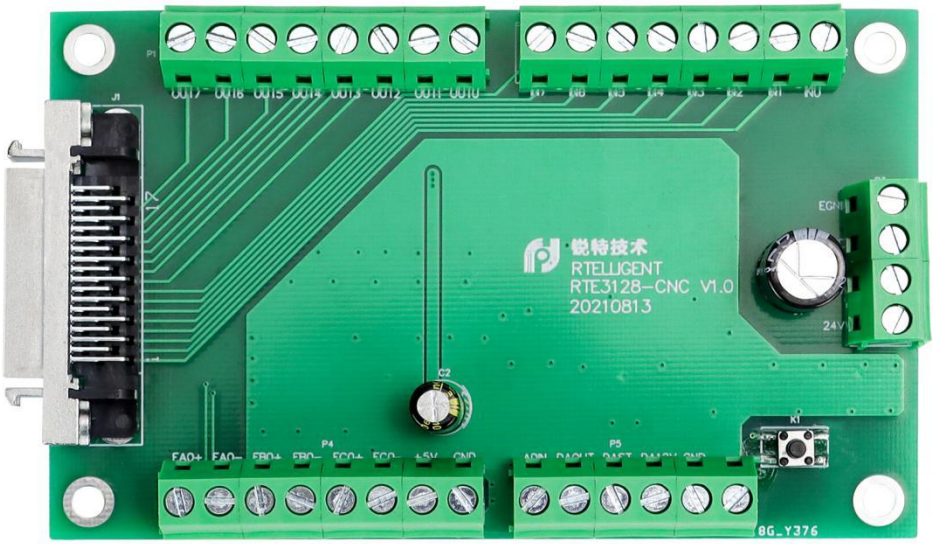


图 2 GPIO 接线板

1.1 产品特点

- 兼容 EtherCAT 总线步进电机驱动器、EtherCAT 伺服驱动器、 EtherCAT 总线 IO 模块
- 最多可支持 128 轴。
- 最快循环周期 125us
- 支持梯形、 S 形速度曲线控制。
- 支持在线变速、变位功能。
- 支持位置控制、速度控制。
- 支持 3 轴(XYZ)空间轴直线插补+N 轴辅助轴 (最多 42 组 3 轴空间直线插补)
- 支持 3 轴(XYZ) 空间轴圆弧插补+N 轴辅助轴 (最多 42 组 3 轴空间圆弧插补)
- 16 个通用输入/输出，I/O 引脚可以配置为 Input，推挽式 output，开漏式 output 或准双向模式。
- 1 组 12Bit 分辨率的 ADC 输入，ADC 时钟频率高达 72 MHz，采样率高达 5.14 MSPS 和 1 组 12BitDAC 输出
- 3 组 32bit 正交编码器输入接口，最高 24MHz 输入频率
- 输入电压 24V
- 1 组 DC5V/1A 输出

1.2 产品参数

网络标准	EtherCAT 总线
网络传输速率	100Mbps 全双工
网络接口	1 个自适应 RJ45 端口
指示灯	电源,系统状态,Link/act, IO 状态
最小通讯周期	125us
SPI 传输速率(Max)	96Mbps(Max)
邮箱通讯	CoE/FoE/EoE
PDO 资料长度	32Bytes~1408Bytes，默认为 112bytes
对象映射	Flexible
支持插补坐标系个数	42 组
编码器输入数	3 组
编码器输入频率(Max)	最大 24Mhz
通用数字 IO 输入数量	8 个，光电隔离， 上升时间：6us(TYP) - 18us(MAX) 下降时间：6us(TYP) - 18us(MAX)
通用数字 IO 输出数量	8 个，光电隔离，开漏输出，每组最大 500mA 开时间：1ms(TYP) - 1.3ms(MAX) 关时间：0.6ms(TYP) - 0.8ms(MAX)
ADC/DAC 数量	各 1 组
驱动程序	Win 7/Vista/XP 32/64,WIN10 32/64
运动控制 函数库	提供 Windows Visual Studio .Net framework DLL (支持 VC, VB, C#, VB.NET, LabVIEW, Delphi)
功耗	<1W
工作温度	-10-55°C
储存温度	-20-70°C
湿度	10%-90%，无凝结

PCB 尺寸	117*87.2mm
净重	71g

1.3 典型应用

RTE3128 总线运动控制卡能够广泛应用于各行各业自动化设备中。主要设备有：

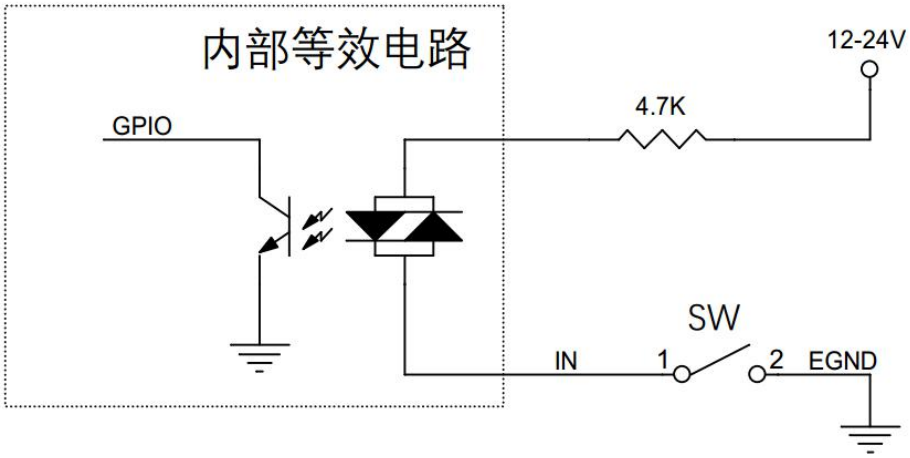
- 电子产品加工、装配设备，如：丝印机、贴片机、PCB 钻孔机等
- 激光加工设备，如：激光打标机、激光切割机等
- 机器视觉及自动检测设备，如：影像测量仪、电路板自动检测设备等
- 生物、医学自动采样、处理设备
- 专用工业机器人

在布线复杂和 EtherCat 从站较多的环境中具有显著优势。

图 3 应用案例

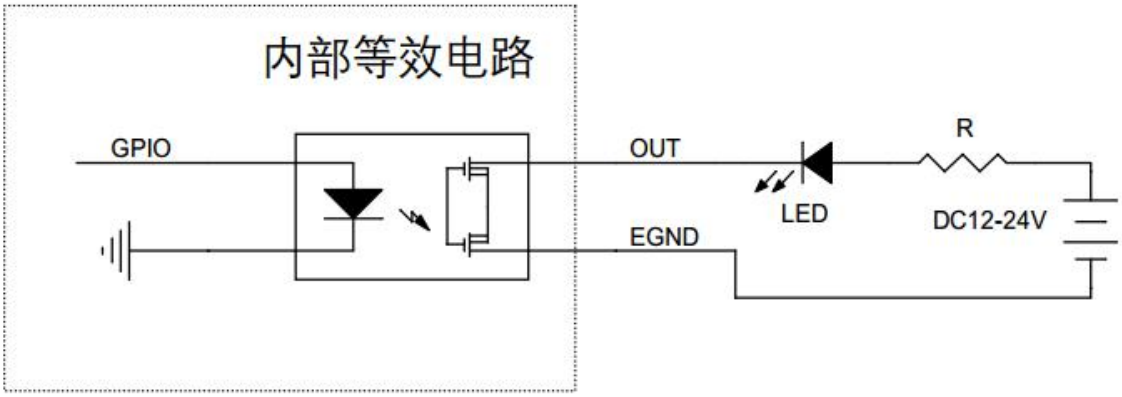
1.4 通用输入接口说明

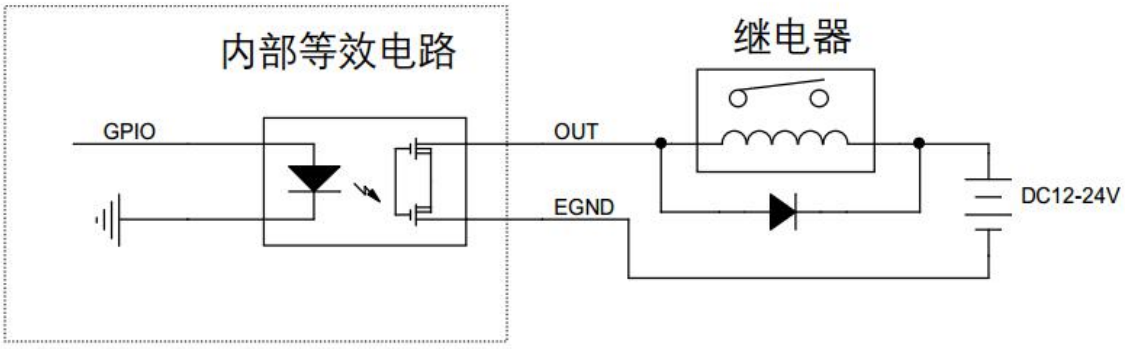
RTE3128 卡共提供 8 组通用数字 IO 输入，每组均带有光电隔离，接口类型为开关量输入，接口原理如下：



1.4 通用输出接口说明

RTE3128 卡共提供 8 组通用数字 IO 输出，各输出端口为光电隔离，开漏输出，下图分别是应用连接 LED 和继电器应用示意图，请注意输出端口必须串接负载，再连接到电源，不能直接连接到电源，否则会烧坏设备。





1.5 主板接口图

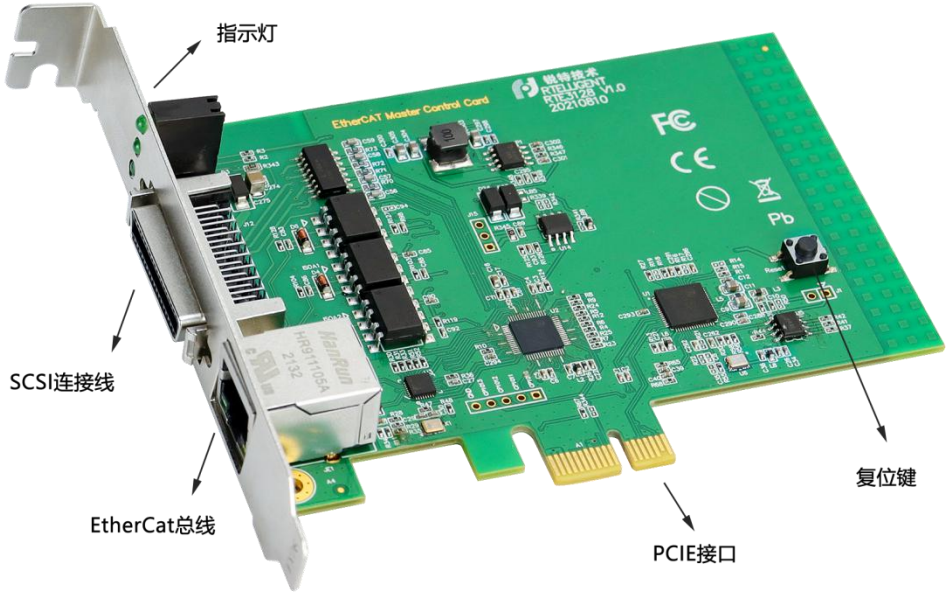


图 4 主板接口

1.6 输入输出 IO 板接口图

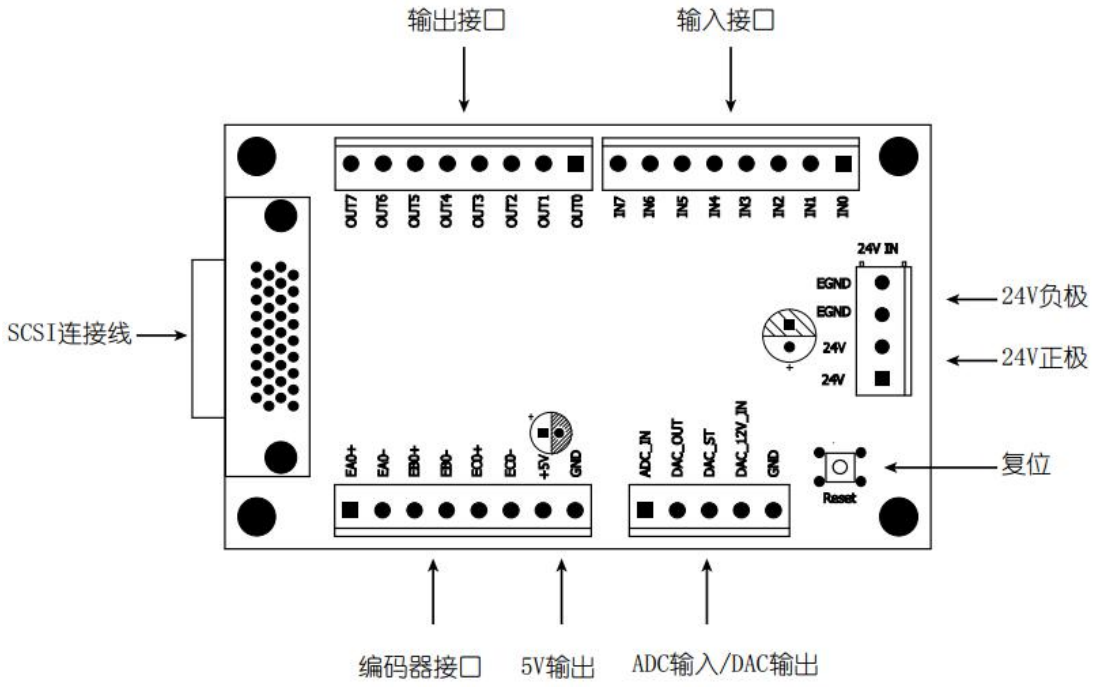


图 5 输入输出 IO 板接口

接线举例说明：

1. 24V 直流电源正负极，分别连接到右边的 4P 接线端子，上面 2 个为负极，下面 2 个为正极。
2. 输入 IO 连接到开关，开关的另一脚连接到 EGND。
3. 输出 IO 连接到继电器的负极端，继电器的正极端连接到电源的正极。

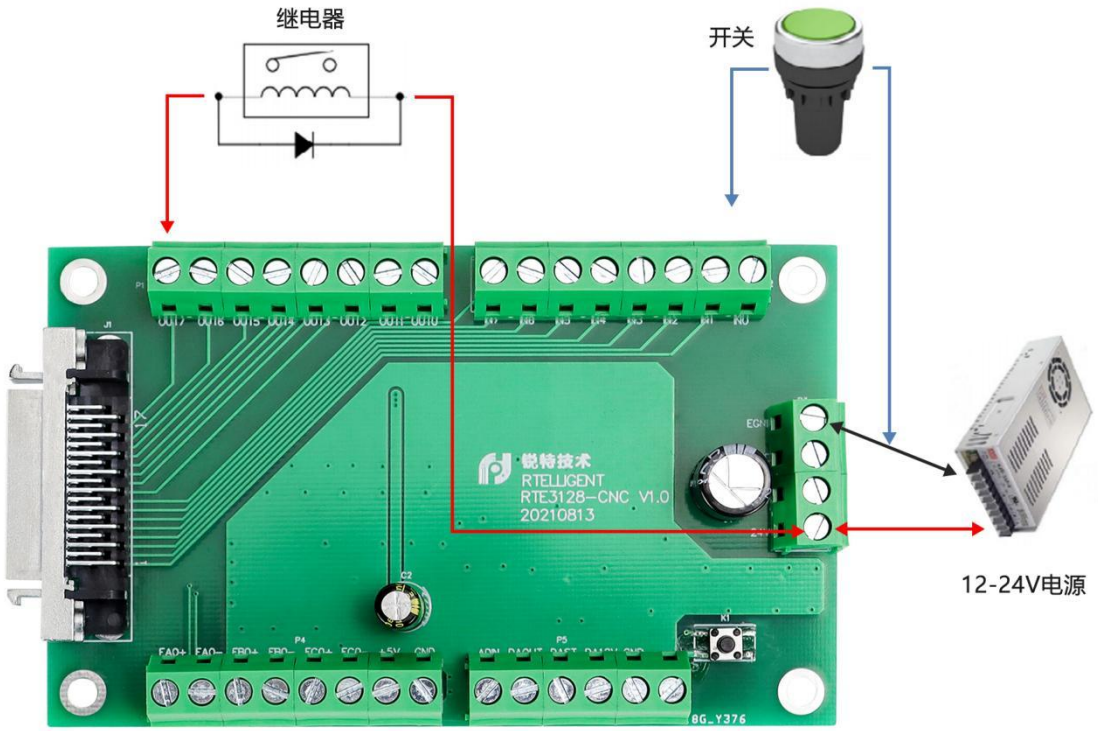
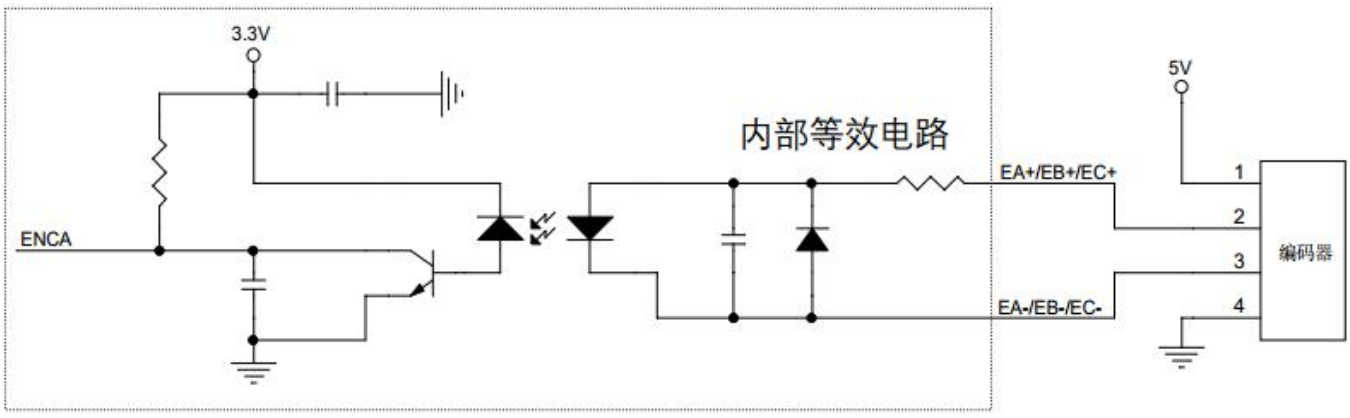


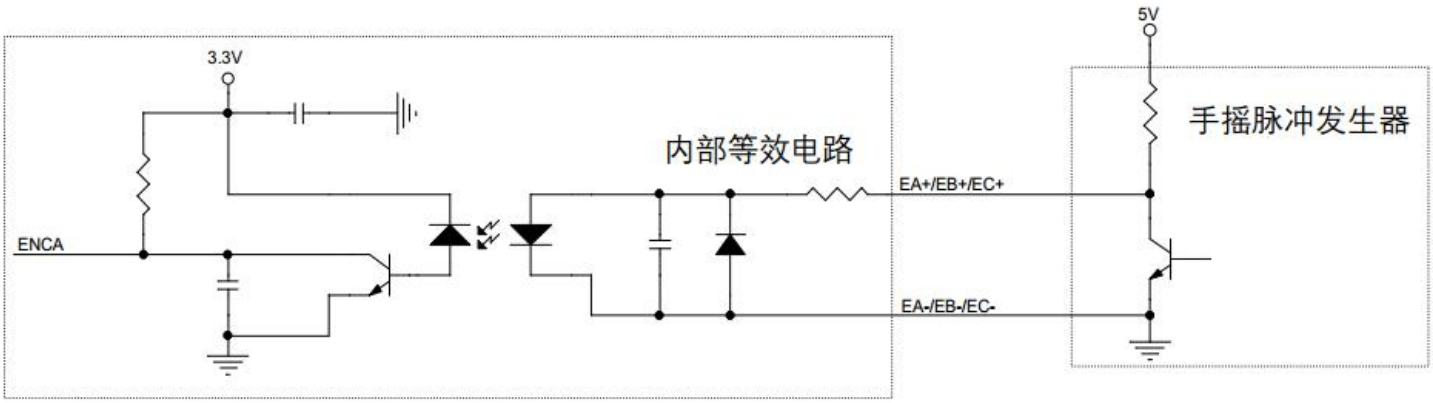
图 6 IO 接线板接线图

1.7 编码器信号输入接口说明

如使用差分输出的编码器，输入信号的正端接 EA+(或 EB+， EC+)端，负端接 EA-(或 EB-，EC-)端。



如使用手摇脉冲发生器，则编码器输出信号接 EA+（或 EB+， EC+）端， EA-（或 EB-， EC-）端连接到 GND。



备注：

1) 编码器等脉冲输入信号的 EA+、 EA-、 EB+、 EB-和 EC+、 EC-的差分信号电压差必须高于 3V，小于 5.5V，且输出电流不应小于 5mA。

2) 需要将编码器的 GND 和控制卡的 GND 连接。

3) RTE3128 配套的扩展板，EGND 为 24V 的外部输入电压的模拟电源负极。GND 为 5V 输出、编码器和 ADC/DCA 的数字电源负极。模拟负极和数字负极，不可直接连接！

第 2 章 Windows 环境下安装说明

2.1 硬件安装

RTE3128 运动控制卡硬件遵从 32bit PCIE 卡结构标准，其安装方法与其它 PCIE 卡，如：显卡、网卡等相似。具体步骤如下：

1）打开控制卡的包装;

2）操作员要带好防静电手套，并触摸一下地线，完全释放身上的静电；

3）关闭 PC 机以及一切与 PC 相连的设备；

4）打开 PC 机的机箱；

5）选择一个靠近处理器的 32bit PCIE 插槽，将控制卡垂直插入插槽中；

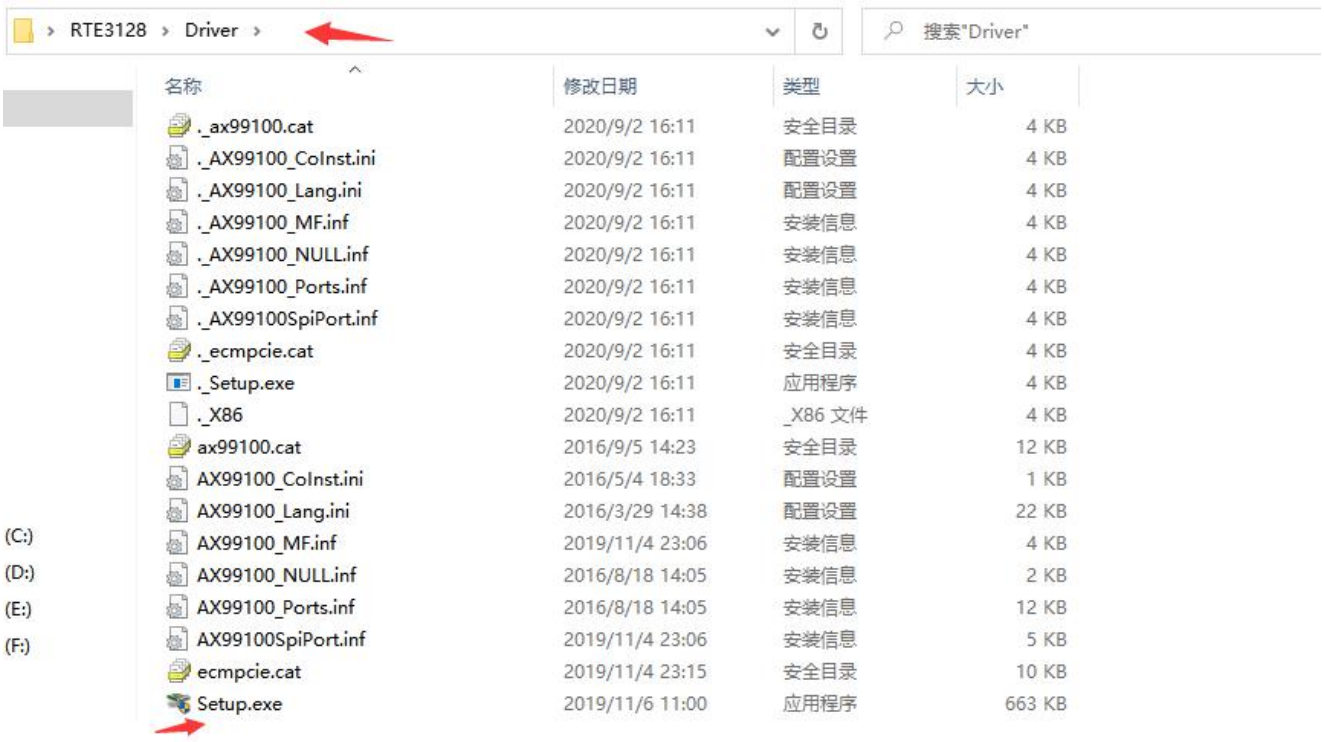
6）将控制卡用螺钉固定在 PC 机机箱上，确保紧固可靠；

7）将 GPIO 接口板用 SCSI 电缆线与控制卡对应的插座连接，并确保连接牢固可靠。

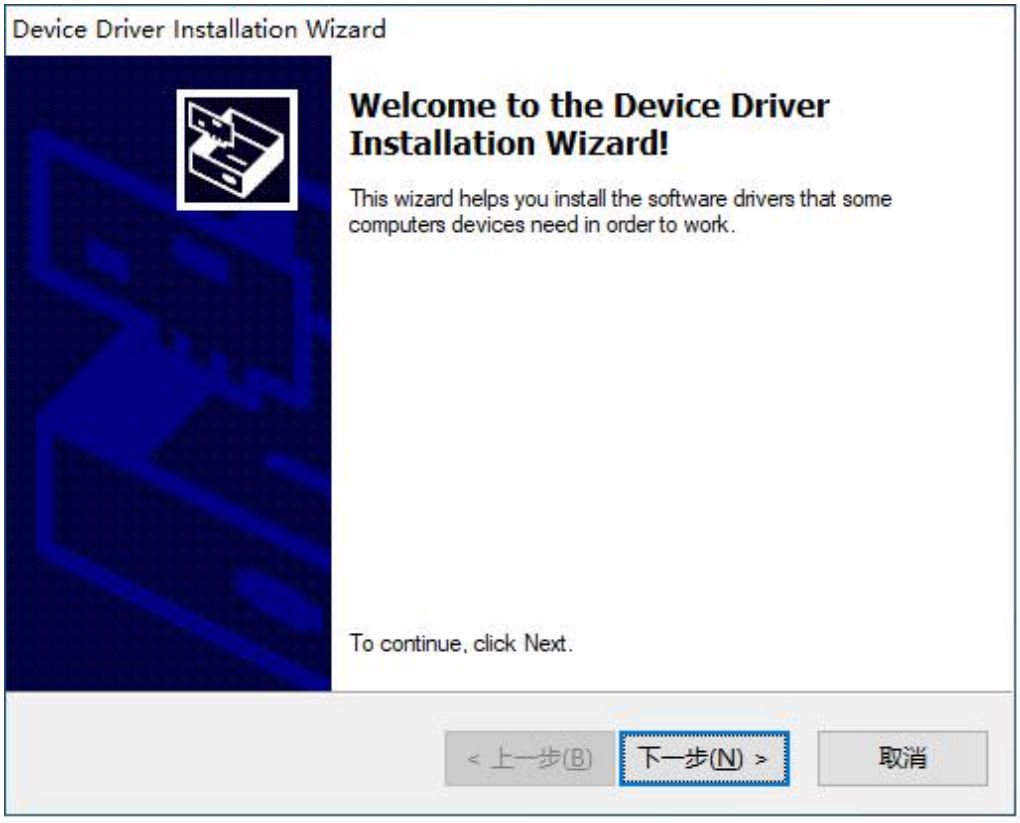
2.2 在 Windows 环境下，安装控制卡的驱动程序

1).将卡插入电脑的 PCI-Express 插槽内，开机，会找到如下设备：

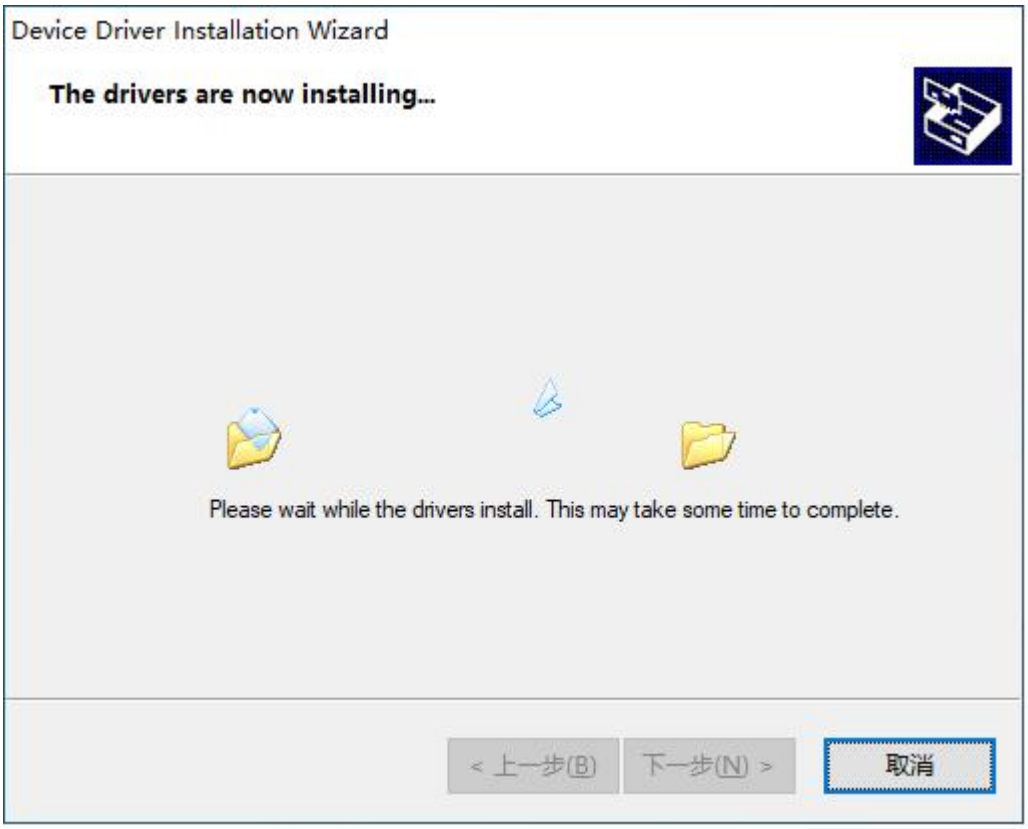
2).找到 Driver 文件夹下的 Setup 安装驱动


RTE3128 › Driver ›
搜索"Driver"
名称 修改日期 类型 大小
._ax99100.cat 2020/9/2 16:11 安全目录 4 KB
._AX99100_CoInst.ini 2020/9/2 16:11 配置设置 4 KB
._AX99100_Lang.ini 2020/9/2 16:11 配置设置 4 KB
._AX99100_MF.inf 2020/9/2 16:11 安装信息 4 KB
._AX99100_NULL.inf 2020/9/2 16:11 安装信息 4 KB
._AX99100_Ports.inf 2020/9/2 16:11 安装信息 4 KB
._AX99100SpiPort.inf 2020/9/2 16:11 安装信息 4 KB
._ecmpcie.cat 2020/9/2 16:11 安全目录 4 KB
._Setup.exe 2020/9/2 16:11 应用程序 4 KB
._X86 2020/9/2 16:11 _X86 文件 4 KB
ax99100.cat 2016/9/5 14:23 安全目录 12 KB
AX99100_CoInst.ini 2016/5/4 18:33 配置设置 1 KB
AX99100_Lang.ini 2016/3/29 14:38 配置设置 22 KB
AX99100_MF.inf 2019/11/4 23:06 安装信息 4 KB
AX99100_NULL.inf 2016/8/18 14:05 安装信息 2 KB
AX99100_Ports.inf 2016/8/18 14:05 安装信息 12 KB
AX99100SpiPort.inf 2019/11/4 23:06 安装信息 5 KB
ecmpcie.cat 2019/11/4 23:15 安全目录 10 KB
Setup.exe 2019/11/6 11:00 应用程序 663 KB
(C:)
(D:)
(E:)
(F:)

3).点下一步。


Device Driver Installation Wizard
Welcome to the Device Driver Installation Wizard!
This wizard helps you install the software drivers that some computers devices need in order to work.
To continue, click Next.
< 上一步(B)
下一步(N) >
取消

4).安装驱动



5).安装完成

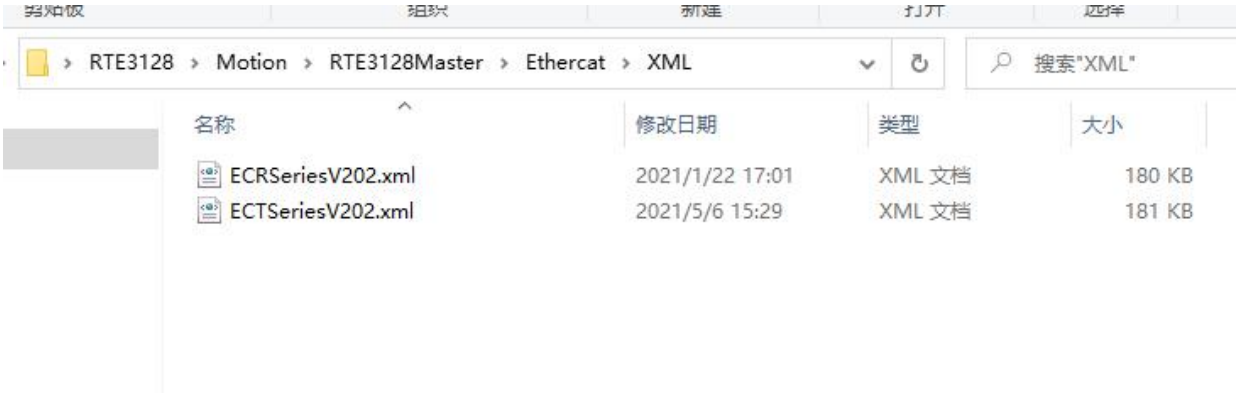
6).设备管理器，会新增 2 个串口和 3 个 PCIe Bridge Device ,表示安装成功。

第 3 章 RTE3128Master Demo 使用说明

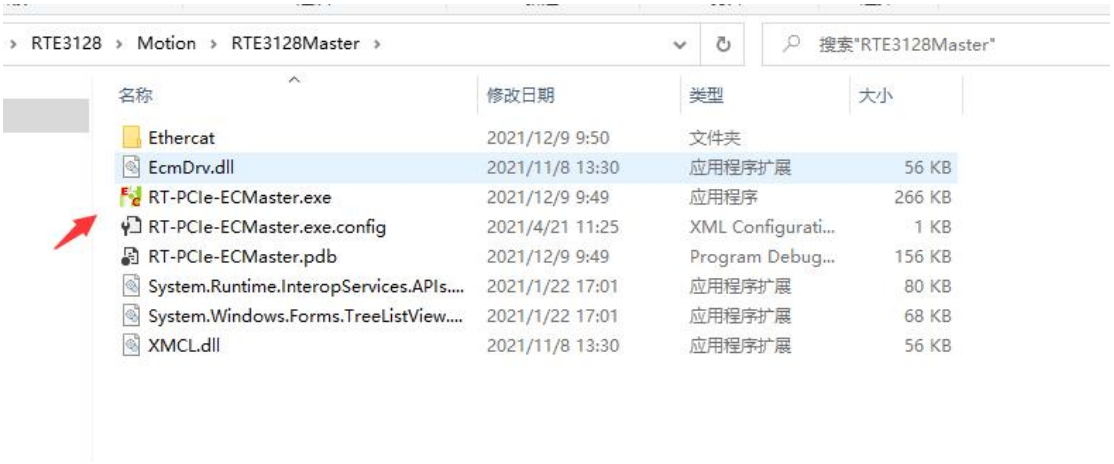
1）首先确认 PCIe 卡的正确安装，进入 Windows 后，打开设备管理器确认系统装置下有[ECM-PCIe EtherCAT Master Controller]。如下图：

2）请使用网线连接 RTE3128 的网口到 EtherCAT 从站设备的 IN 端，接好设备并通电。

3）找到 RTE3128Master 的 Ethercat 文件夹下的 XML 文件夹。并把您连接设备的 XML 描述文件放入此文件夹下。

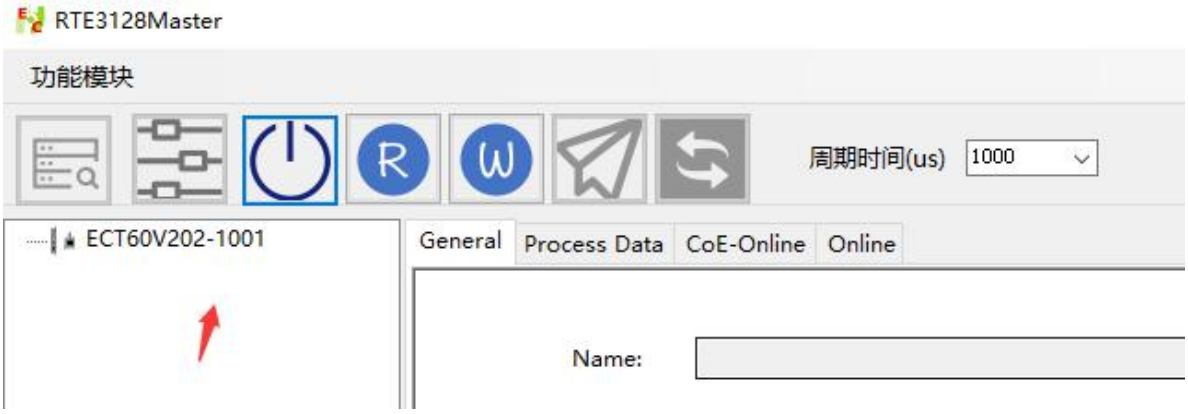


4）回到 RTE3128Master 文件夹下，点击 exe 程序文件启动调试软件。注意 EcmDrv.dll 和 XMCL.dll 两个文件必须在启动的同一文件夹下。

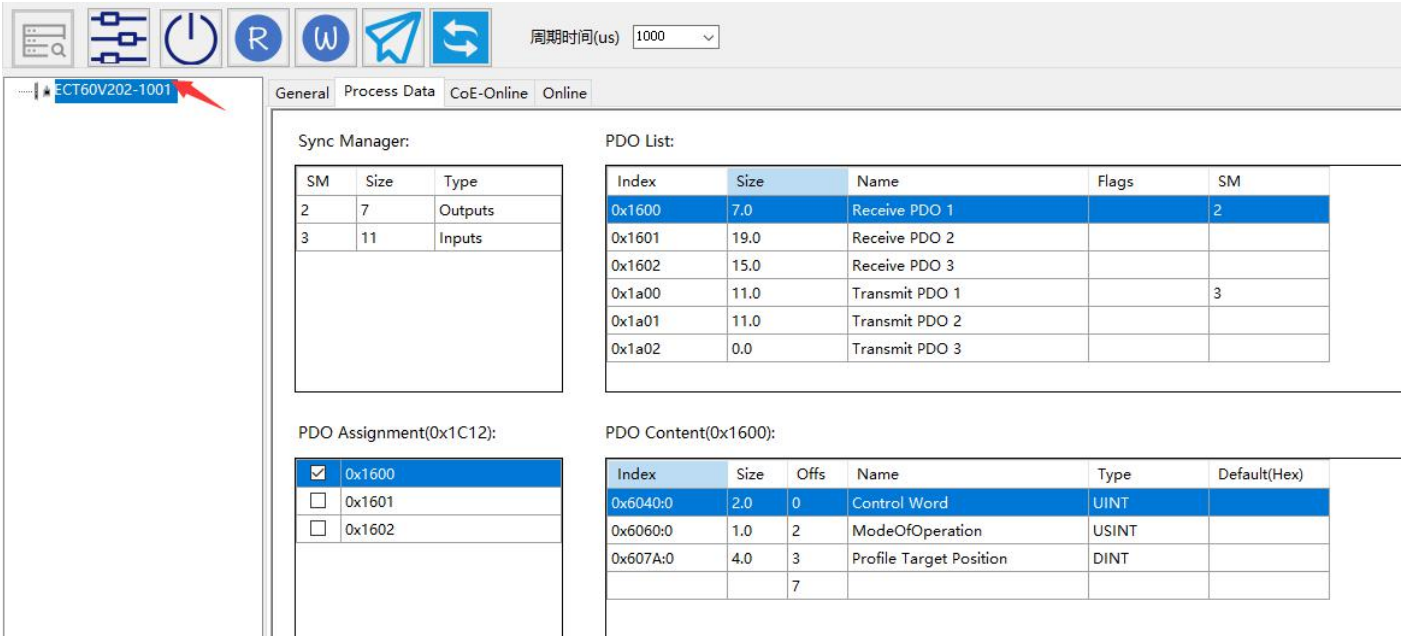


5）选择主站初始化时的循环周期。

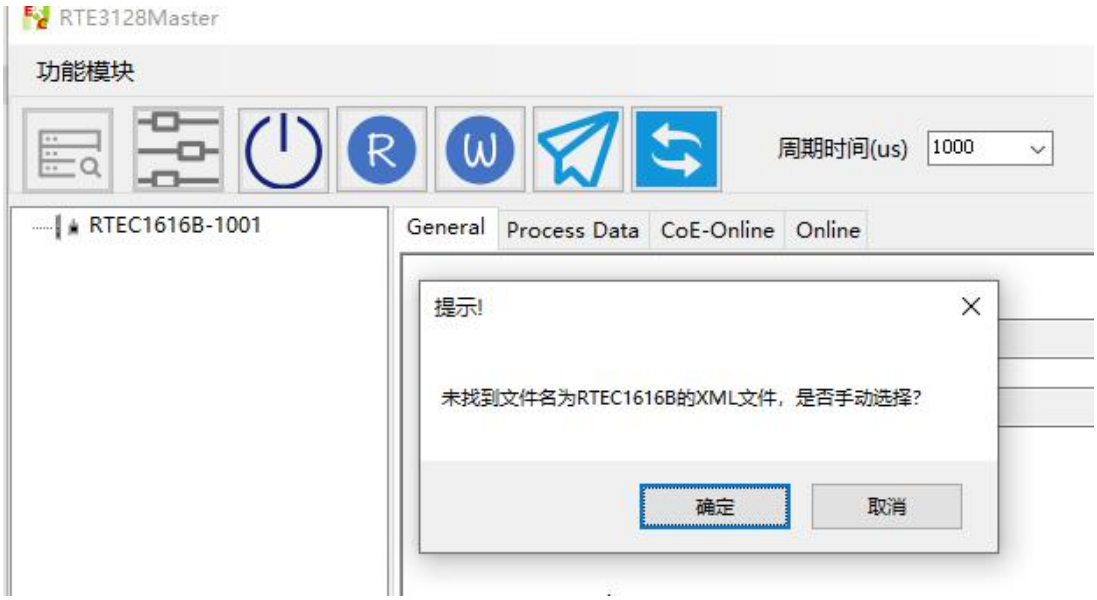
6）然后再点击设备搜索主站将会初始化并将搜索到的从站以列表的形式显示在左侧。



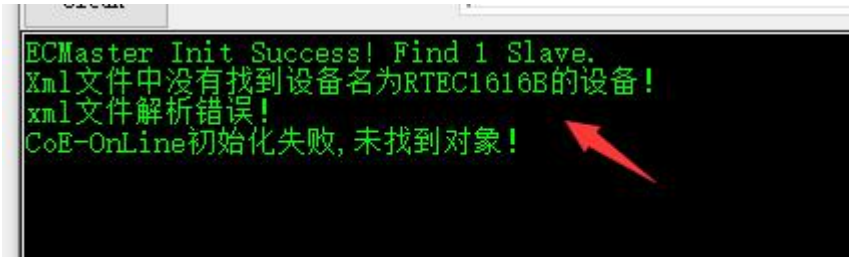
7）搜索到从站设备后从站进入 Pre-OP 状态下。点击左侧的设备将加载设备的 XML 描述文件并可以在右侧进行相应的过程数据配置，在配置完成后上方的激活配置并进入 DC 同步按钮变得可点击。

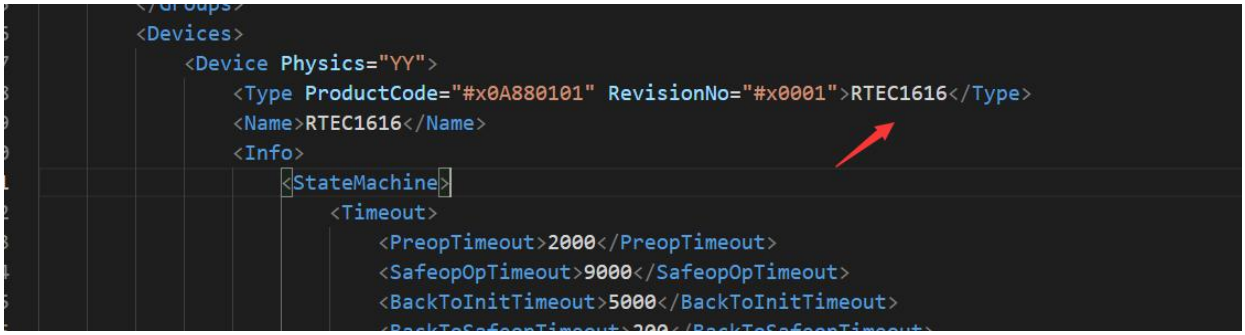


8）在未找到设备名匹配的 XML 描述文件时，可以手动进行选择。



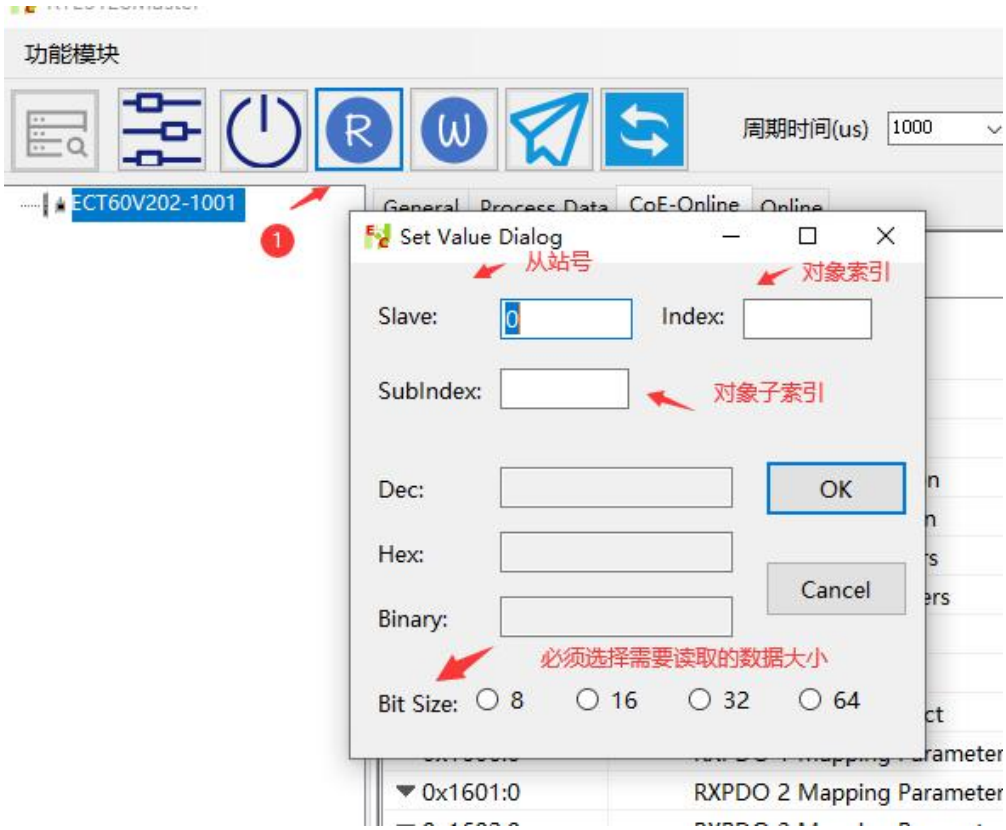
9）如果加载 XML 配置文件过程中出现如下设备名不匹配的情况则修改 XML 文件内的设备类型与提示的设备名相同即可。



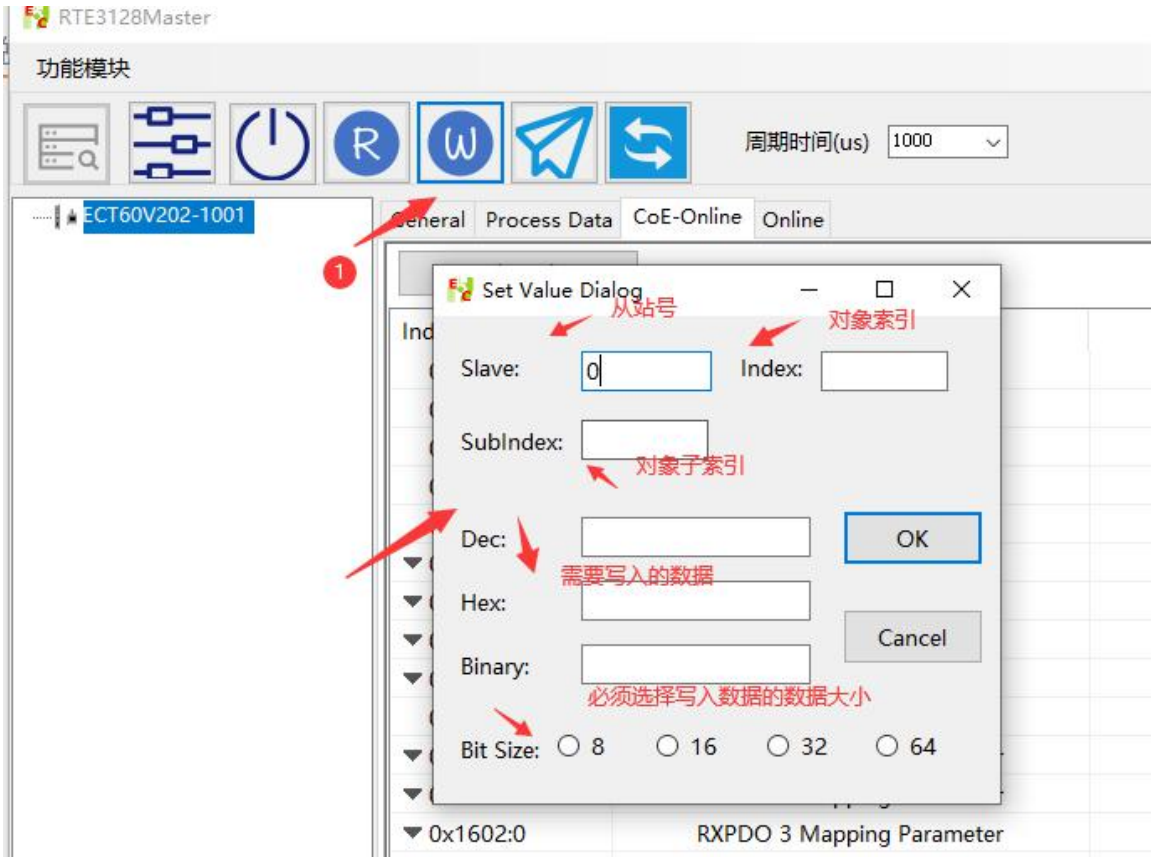


将上图中 XML 文件的 Type 结点的 RTEC1616 改为错误提示的设备名 RTEC1616B。两者对应匹配即可解决 XML 文件的加载错误。

10）在上方按钮中 R 按钮用来读取从站的 SDO 数据。

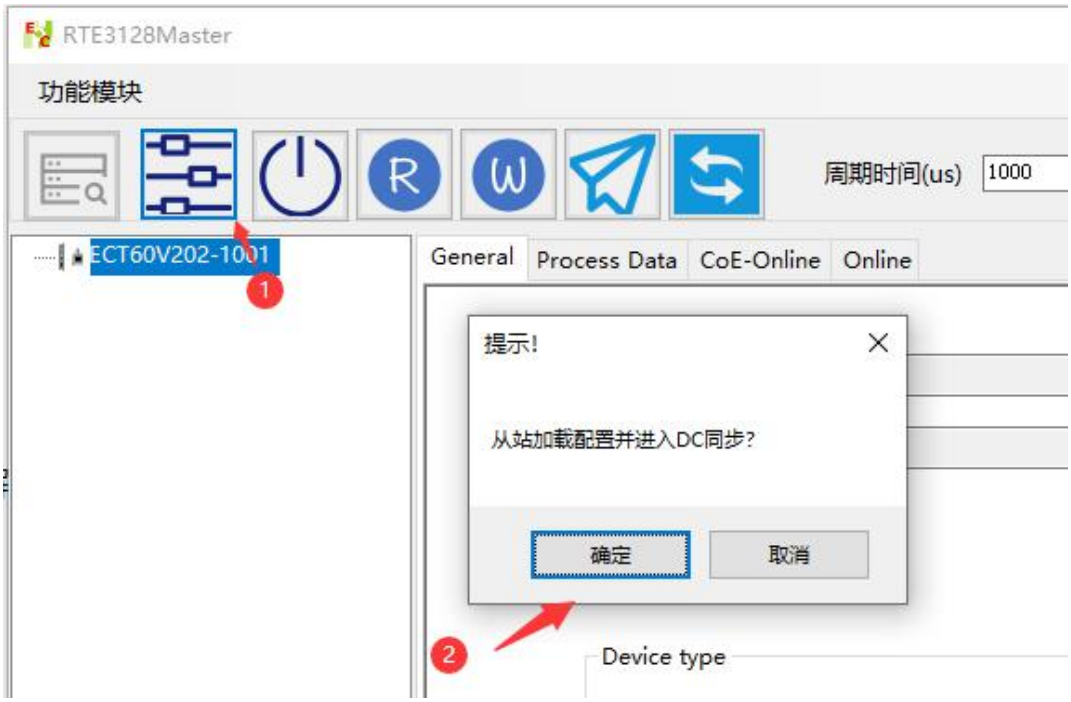


11）在上方按钮中 W 按钮用来写从站的 SDO 数据。

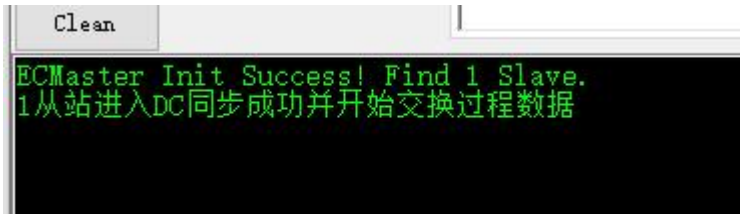


12）如果从站设备是属于锐特厂商的，则一键保存至设备和设备恢复出厂设置将变得可用。

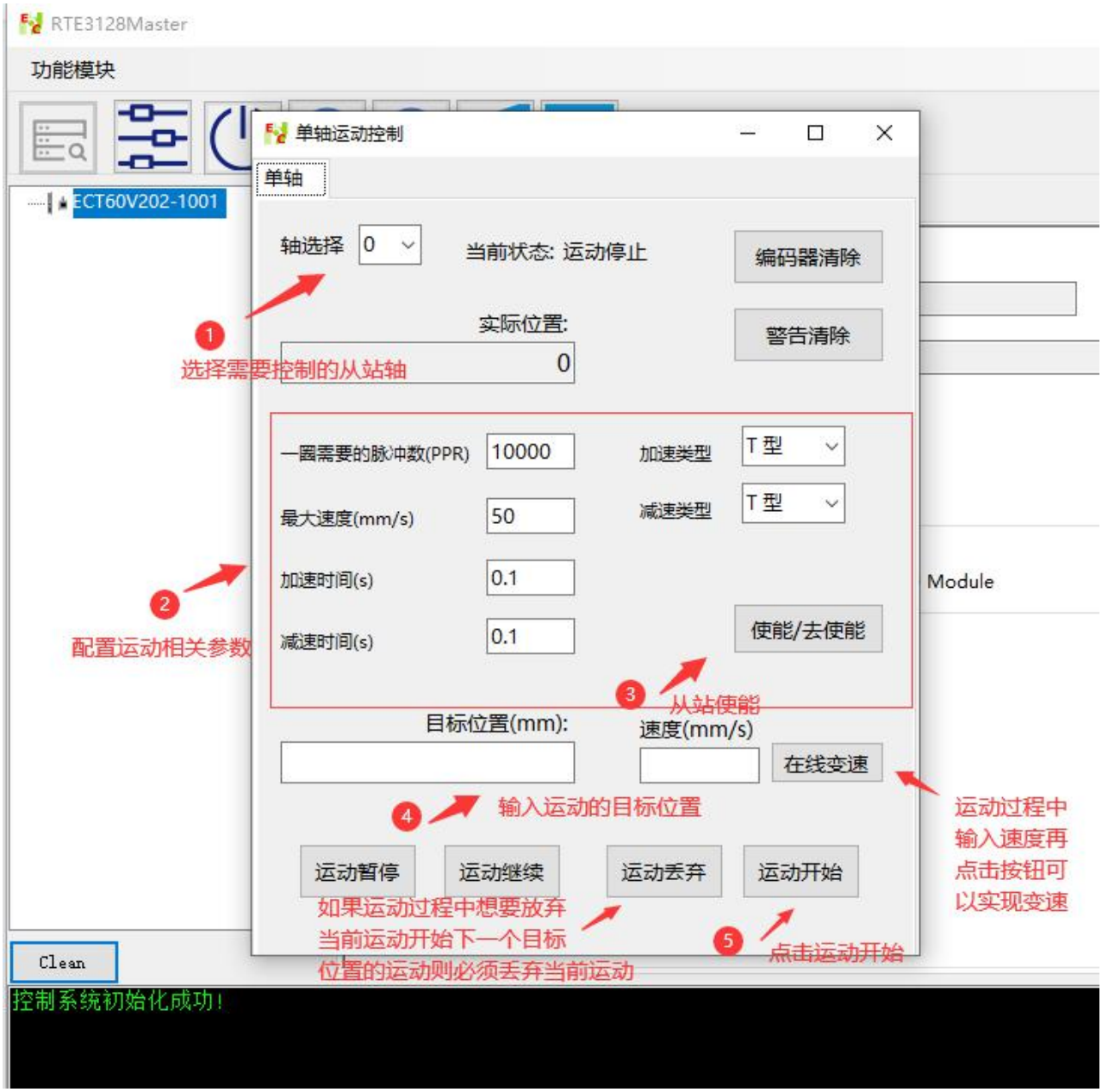
13）在配置完成后点击上方的激活配置并进入 DC 同步。



14）在成功后会出现提示消息，此时从站设备处于 OP 模式下。



15）单轴运动的简单启动。点击菜单栏下的“单轴运动控制”进入单轴运动控制界面。进入界面后信息提示栏出现“控制系统初始化成功!”字眼即可正确控制运动。

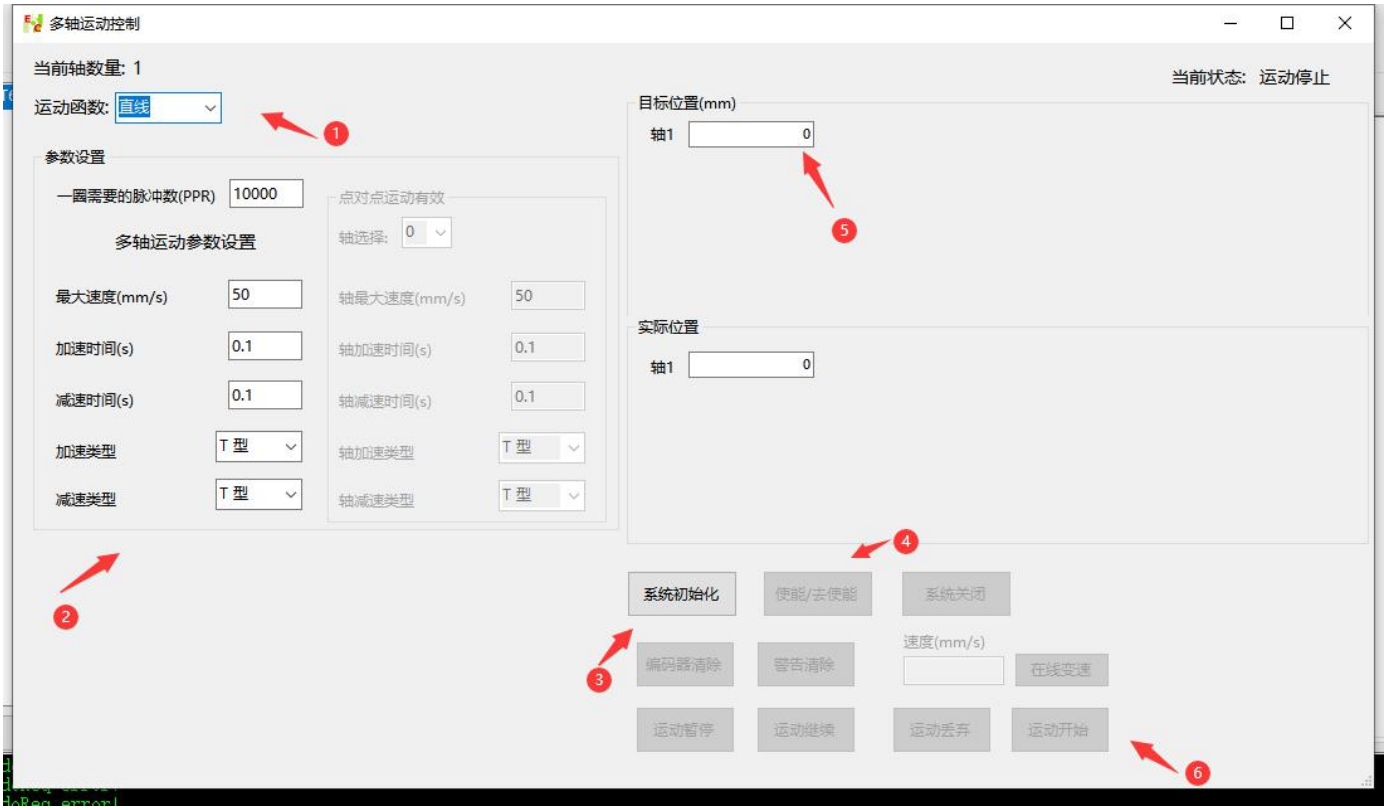


16）多轴运动的简单启动。点击菜单栏下的“多轴运动控制”进入多轴运动控制界面。

1. 选择运动函数
2. 配置运动相关参数
3. 点击系统初始化，如果初始化成功后则相关案件将会变得可点击。
4. 点击使能，驱动器将进入使能状态。

5. 输入运动的目标位置。

6. 点击运动开始就会开始选择的运动。



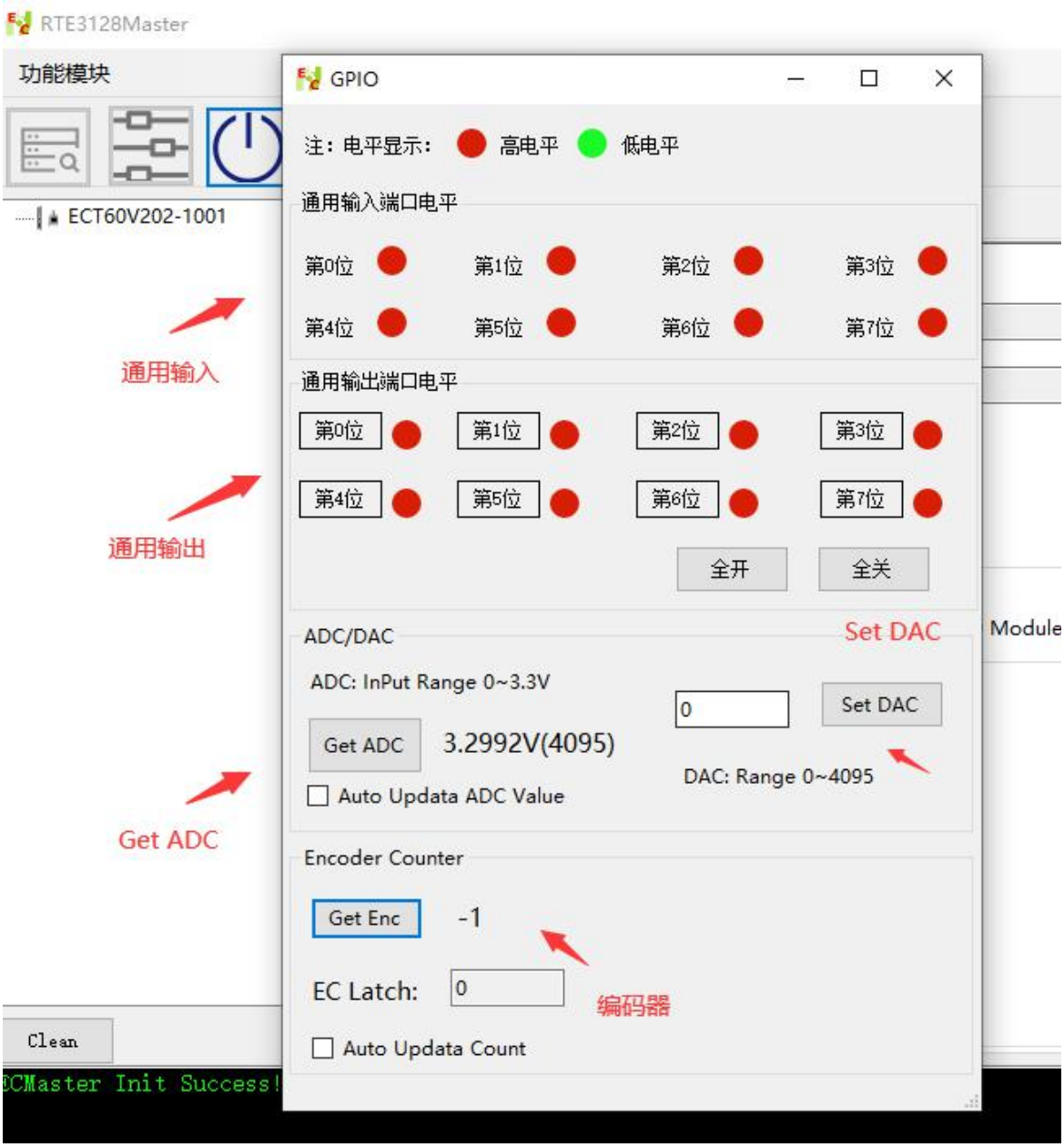
17）GPIO 控制界面的简单启动。

1）参考上面的硬件连接说明，使用配套的 SCSI 连接线连接小板，可以读取输入状态和设置输出状态；

2）ADC 的输入电压范围是 0-3.3V；

3）设置 DAC 输出后，使用万用表测试 DAC out 端和 GND，可以量测到电压值；

4）编码器，A 和 B 组检测到脉冲变化后，值会实时显示状态变化，C 组默认是 0，检测到变化后，会变成-1，不会实时显示状态变化。



第 4 章 RTE3128 功能介绍

锐特 RTE3128 运动控制卡是一款新型的 PCIE 总线运动控制卡。可以搭载多个标准 EtherCAT 从站设备，能够控制多个步进电机或数字式伺服电机；适合于多轴点位运动、插补运动、IO 控制等功能应用。

RTE3128 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。随卡免费提供的 Motion 调试软件，不但可以演示 RTE3128 系列卡的控制功能，而且可用于控制卡及运动控制系统的硬件测试。

4.1 运动控制功能

4.1.1 点对点运动(ptp)

点对点运动是指：运动控制器控制运动平台从当前位置开始以设定的速度运动到指定位置后准确地停止。

点对点运动只关注终点坐标，对运动轨迹的精度没有要求。点位运动的运动距离由用户输入的目标位置决定，运动速度由用户决定。

用户在 PC 机执行点位运动指令，即调用点位运动函数，并自动将运动参数通过 PCIE 总线接口传送至 RTE3128 运动控制卡，当从站处于 CSP 模式下，控制卡按照用户设定的运行速度和循环周期同步驱动器的当前位置，以达到到达目标位置准确停止的目的，将到达点对点运动的规划目的时，过程数据也停止对目标位置的规划。

4.1.1.1 梯型速度控制

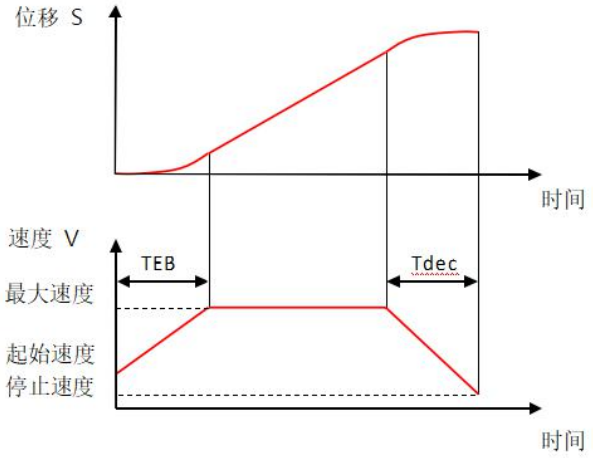


图2.3 梯形速度曲线及对应的位移曲线

TEB：总加速时间，Tdec：总减速时间

为了让平台在运动过程中能平稳加速、准确停止，一般采用梯形速度曲线控制运动过程， 如上图 所示。即：电机以起始速度开始运动，加速至最大速度后保持速度不变，结束前减速至停止速度，并停止。运动的距离由点对点运动指令决定。

4.1.1.2 S 型速度控制

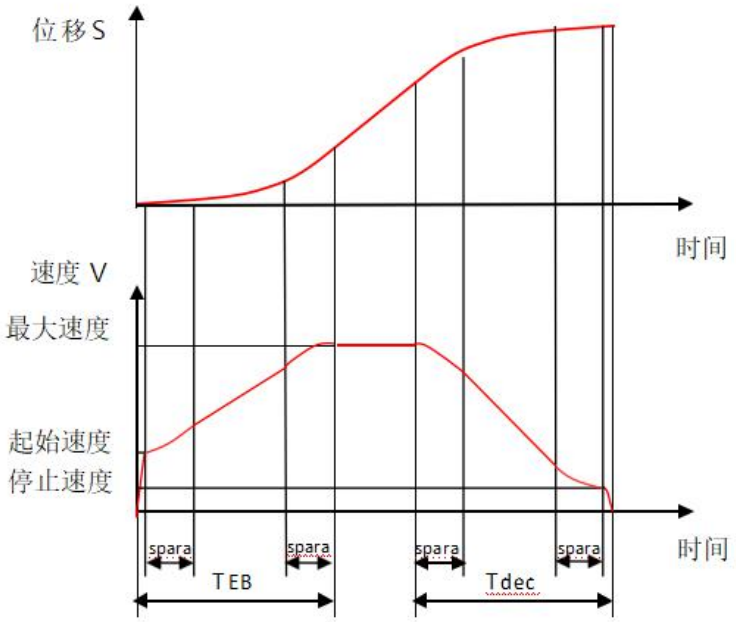


图2.4 S形速度曲线及对应的位移曲线

spara：S段时间，TEB：总加速时间，Tdec：总减速时间

为改善平台运动的平稳性，RTE3128 运动控制卡还提供了 S 形速度控制曲线。在 S 形速度控制过程中，指令位置频率从一个内部设定的速度快速加速到起始速度，然后作 S 形加速运动；运动结束前，指令位置频率作 S 形减速运动到停止速度，然后再快速减速到一个内部设定的速度，直到到达目标位置，如上图所示。

点对点运动相关函数说明

名称	功能
Ptp	启动点对点运动
XF_SetAxisAccDecType	设置轴加速和减速类型
XF_SetAxisAccType	设置轴加速类型
XF_SetAxisDecType	设置轴减速类型
XF_SetAxisVelocity	设置轴运行速度
XF_SetAxisBeginVelocity	设置轴启动速度
XF_SetAxisEndVelocity	设置轴终止速度
XF_SetAxisAccTime	设置轴加速时间
XF_SetAxisDecTime	设置轴减速时间

4.1.2 直线插补运动(Line_Aux)

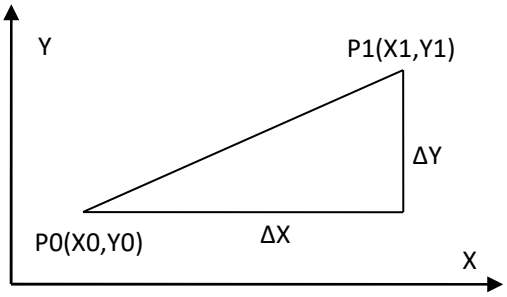
插补运动是为了实现轨迹控制，运动控制卡按照一定的控制策略控制多轴联动，使运动平台用微小直线段精确地逼近轨迹的理论曲线，保证运动平台从起点到终点上的所有轨迹点都控制在允许误差范围内。

RTE3128 支持 3 轴(XYZ)空间轴直线插补+N 轴辅助轴 (最多 42 组 3 轴空间直线插补)，空间轴和辅助轴同时启动同时结束，插补计算由运动控制函数库执行，用户只需将插补运动的速度、加速度、终点位置等参数写入相关函数即可。

1）两轴直线插补

2 轴直线插补从 P0 点运动至 P1 点，X、Y 轴同时启动，并同时到达终点； X、Y 轴的运动速度之比为 ΔX：ΔY，二轴合成的矢量速度为：

$$\frac{\Delta P}{\Delta t}=\sqrt{\left(\frac{\Delta X}{\Delta t}\right)^2+\left(\frac{\Delta Y}{\Delta t}\right)^2}$$

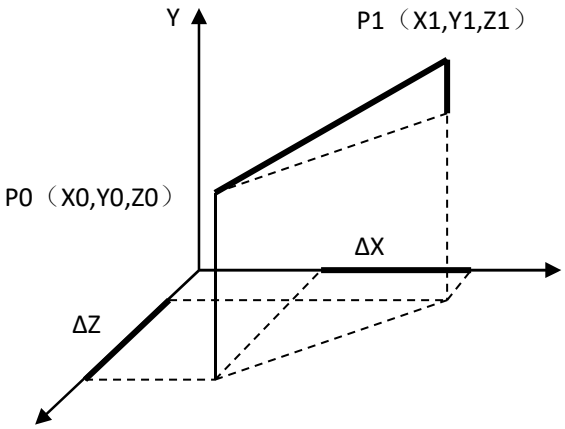


两轴直线插补

2）三轴直线插补

在 X、Y、Z 轴内直线插补，从 P0 点运动至 P1 点。插补过程中 3 轴的速度比为 ΔX：ΔY：ΔZ，三轴合成的矢量速度为：

$$\frac{\Delta P}{\Delta t}=\sqrt{\left(\frac{\Delta X}{\Delta t}\right)^2+\left(\frac{\Delta Y}{\Delta t}\right)^2+\left(\frac{\Delta Z}{\Delta t}\right)^2}$$



三轴直线插补

直线插补运动相关函数说明

名称	功能
Line_Aux	启动直线插补运动
XF_SetGroupAccDecType	设置插补群组加速和减速类型
XF_SetGroupAccType	设置插补群组加速类型
XF_SetGroupDecType	设置插补群组减速类型
XF_SetGroupMaxSpeed	设置插补群组运行速度
XF_SetGroupBeginSpeed	设置插补群组启动速度
XF_SetGroupEndSpeed	设置插补群组终止速度
XF_SetGroupAccTime	设置插补群组加速时间
XF_SetGroupDecTime	设置插补群组减速时间
XF_OverwriteSpeed	在线变速

4.1.3 圆弧插补运动(Arc_Aux)

RTE3128 控制卡支持 3 轴(XYZ) 空间轴圆弧插补+N 轴辅助轴 (最多 42 组 3 轴空间圆弧插补)，在进行圆弧插补运动时，确定一个圆弧需要当前位置、参考位置、目标位置。插补系统会自动计算出圆弧的轨迹然后启动插补。空间轴依照速度规划进行空间圆弧运动，圆弧由目前位置、经过参考位置，至目标位置结束。辅助轴运动与空间轴运动为同时开始，同时结束。

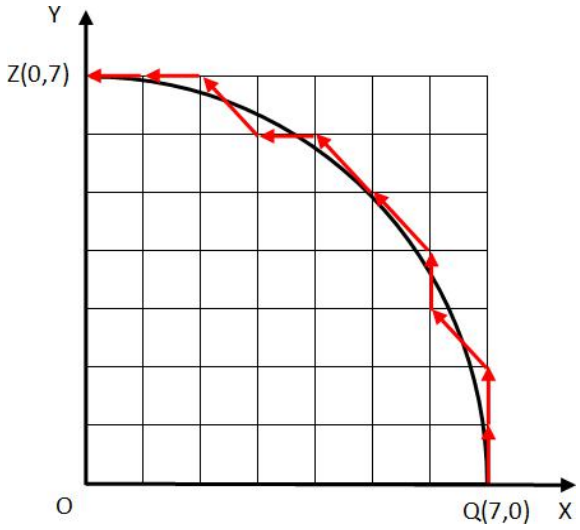


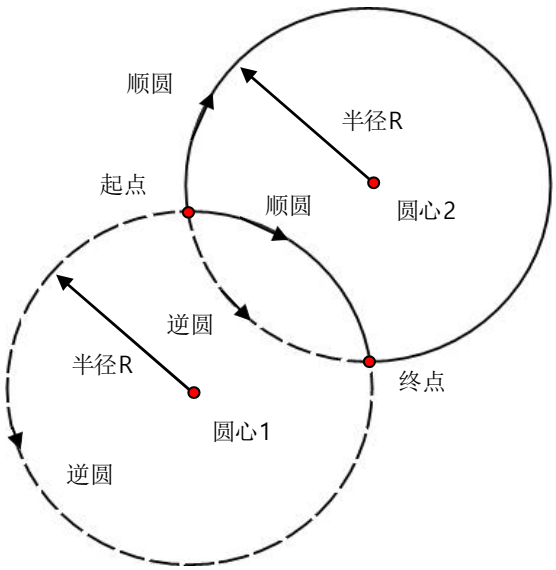
图 7 圆弧插补实例

圆弧插补运动相关函数说明

名称	功能
Arc_Aux	启动圆弧插补运动
XF_SetGroupAccDecType	设置插补群组加速和减速类型
XF_SetGroupAccType	设置插补群组加速类型
XF_SetGroupDecType	设置插补群组减速类型
XF_SetGroupMaxSpeed	设置插补群组运行速度
XF_SetGroupBeginSpeed	设置插补群组启动速度
XF_SetGroupEndSpeed	设置插补群组终止速度
XF_SetGroupAccTime	设置插补群组加速时间
XF_SetGroupDecTime	设置插补群组减速时间
XF_OverwriteSpeed	在线变速

4.1.4 圆周插补运动(Circle_Aux)

RTE3128 控制卡能够进行圆周插补。在进行圆周插补之前，必须确定圆周轨迹的参数。进行空间圆运动从当前位置出发，需要知道圆心、以及表示一个平面的法向量（比如 XYZ 中表示在 XY 平面上做圆周插补则法向量为（0，0，1））、还需要运动轴的目标位置，当空间轴和辅助轴同时运动时，空间轴的目标位置参数为无效参数，辅助轴的目标位置为有效参数。空间轴和辅助轴运动同时开始同时结束。



圆周插补轨迹

圆轴插补运动相关函数说明

名称	功能
Circle_Aux	启动圆弧插补运动
XF_SetGroupAccDecType	设置插补群组加速和减速类型
XF_SetGroupAccType	设置插补群组加速类型
XF_SetGroupDecType	设置插补群组减速类型
XF_SetGroupMaxSpeed	设置插补群组运行速度
XF_SetGroupBeginSpeed	设置插补群组启动速度
XF_SetGroupEndSpeed	设置插补群组终止速度
XF_SetGroupAccTime	设置插补群组加速时间
XF_SetGroupDecTime	设置插补群组减速时间
XF_OverwriteSpeed	在线变速

4.2 主站配置说明

EtherCAT 从站 设备必须支持 4 种状态，负责协调主站和从站应用程序在初始化和运行时的状态关系。

1）Init：初始化，简写为 I；

2）Pre-Operational：预运行，简写为 P；

3）Safe- Operational：安全运行，简写为 S；

4）Operational：运行，简写为 O。

EtherCAT 网络状态机如下图所示。

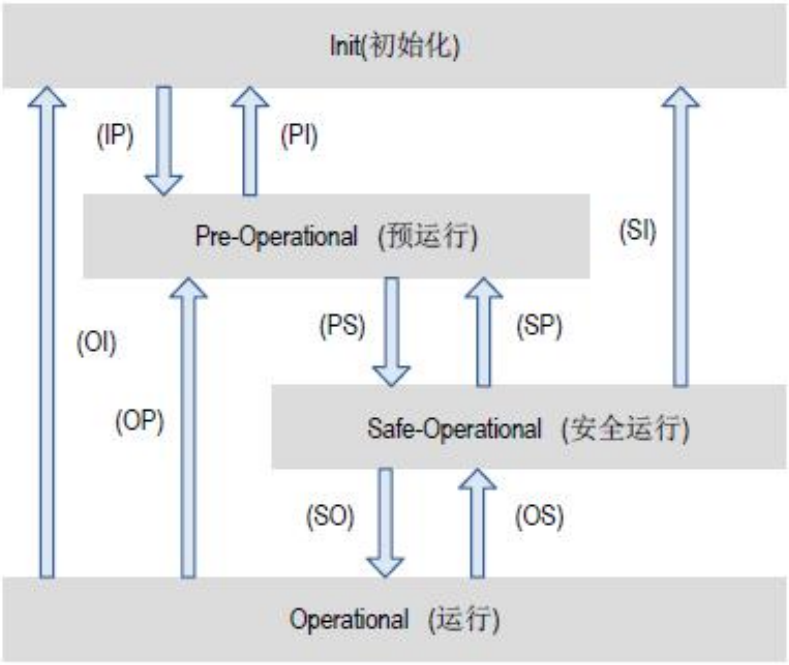


图 8 EtherCAT 状态机

从初始化状态向运行状态转化时，必须按照“初始化→预运行→安全运行→运行”的顺序转化，不可以越级。从运行状态返回时可以越级转化。状态的转化操作和初始化过程如下表：

状态	描述
Init	1) 没有邮箱通讯 2) 没有过程数据通讯
Init to Pre-Op	1) 主站配置链路层地址，启动邮箱通讯 2) 主站初始化 DC 同步时钟 3) 主站请求向 Pre-Op 状态转换 4) 主站设置 AL 控制寄存器 5) 从站检查邮箱是否初始化正确
Pre-Operation (Pre-OP)	1) 邮箱通讯被激活 2) 不能进行过程数据通讯
Pre-Op to Safe-Op	1) 主站为过程数据配置同步管理器(Sync Manager)

	通道和 FMMU 通道 2) 主站通过 SDO 对从站进行 PDO 数据映射及 Sync Manager PDO 参数设置 3) 主站请求向 Safe-Op 状态转换 4) 从站检查负责 PDO 数据的 Sync Manager 配置是否正确，如果主站发出启动同步请求，检查分布时钟的设置是否正确
Safe-Operation（Safe-Op）	1）从站应用程序将传送实际输入数据，不对输出进行操作，输出被设置为“安全状态”
Safe-Op to Op	1) 主站发送有效的输出数据 2) 主站请求向 Op 状态转换
Operational（Op）	可以进行过程数据通讯

通过 EcmDrv.dll 内的函数初始化主站也遵循 EtherCAT 状态转换的规则。RTE3128 主站配置流程大致如下图所示：

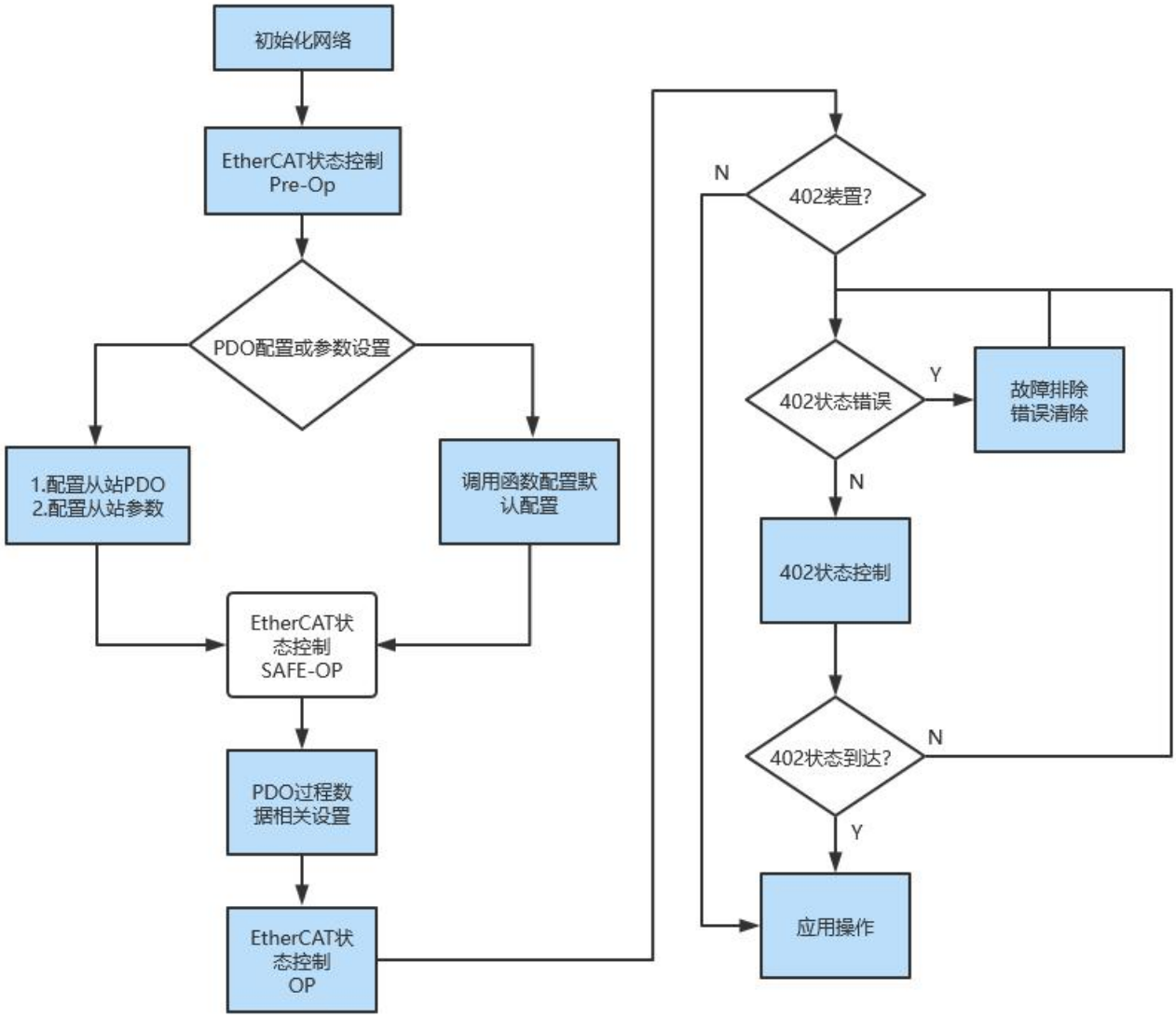


图 9 主站配置

4.2.1 主站初始化函数配置说明

1) 首先调用 int UserSys_Init(int u32SpiFreq) 函数初始化上位机和板卡的 SPI 通讯。参数 u332SpiFreq 为 SPI 通讯频率。

2) 调用 int ECM_InitLibrary(uint16_t *pu16SpiDataSize) 函数初始化函数库，设置 Spi 传输数据大小，参数 pu16SpiDataSize 为 0 时默认设置大小为 112。

3) 调用 int ECM_EcatInit(ushort DCActCode, uint CycTime) 函数初始化主站网络，参数 DCAtCode 为 DC 同步代码，0 为禁止 DC 同步，0x300 为同步 Sync0，0x700 同时激活 Sync0、Sync1；参数 CycTime 为循环周期时间，单位为 ns。

4) 可以使用 int ECM_EcatSlvCntGet() 函数获取连接到主站的从站个数。该函数返回获取到的从站个数。

5) 调用 int ECM_StateCheck(int u8Slave,int u8ExpectState,int TimeOutMS) 函数配置从站进入 Pre-Op 状态。参数 u8Slave 为从站编号，从站号码根据网口连接顺序从 0 递增；参数 u8ExpectState 为请求进入的状态，参数 TimeOutMs 为请求超时时间，单位为毫秒。

```
#define EC_STATE_NONE                   0x00
#define EC_STATE_INIT                   0x01
#define EC_STATE_PRE_OP                 0x02
#define EC_STATE_BOOT                   0x03
#define EC_STATE_SAFE_OP                0x04
#define EC_STATE_OPERATIONAL            0x08
#define EC_STATE_ACK                    0x10
#define EC_STATE_ERROR                  0x10
```

图 10 状态请求码

6) 调用 int ECM_EcatSdoSetPdoConfig(PDO_CONFIG_HEAD* pConfigData) 函数自定义配置过程数据 PDO，参数 PDO_CONFIG_HEAD 为存有 PDO 配置信息的结构体；在使用该函数成功配置 PDO 之后需要调用 int ECM_EcatReconfig() 函数重新分配内存空间。同时还可以通过调用 int ConfigDrive(int nList, int nSlvB, int nSlvE, int nJustAssign, int nPDOCnt, ushort RxMapIdx, ushort TxMapIdx) 函数来配置默认的 PDO 过程数据。

```
ECM_PACK_BEGIN
typedef struct ECM_PACK obj_entry_t
{
    uint8_t u8BitSize;
    uint8_t u8SubIdx;
    uint16_t u16Idx;
} OBJ_ENTRY_T;
ECM_PACK_END
ECM_PACK_BEGIN
typedef struct ECM_PACK pdo_config_t
{
    uint32_t    VID;
    uint32_t    PID;
    uint8_t     Slave;
    uint8_t     PDOCnt;
    uint16_t    SmaIdx;
    uint16_t    MapIdx[4];
    uint16_t    ObjsCnt[4];
    OBJ_ENTRY_T Table[32];
} PDO_CONFIG_HEAD;
ECM_PACK_END
```

图 11 PDO 结构体定义

```
ConfigDrive(1, 0, slaveCount - 1, 0, 1, 0x1600, 0x1A00); //PDO配置
```

图 12 PDO 默认配置参数格式

7) 调用 int ECM_SetTxFIFOCnt(byte u8TxCnt) 函数设置命令缓存区 Tx 的大小，参数 u8TxCnt 为需要设置的数据。

8) 调用 int ECM_SetRxFIFOCnt(byte u8RxCnt) 函数设置命令缓存区 Rx 的大小，参数 u8RxCnt 为需要设置的数据。

9) 调用 int ECM_InitFIFO() 函数初始化命令缓存区。

10) 调用 int SetDriverCmdOpMode(int nSlvBegin, int nSlvEnd, byte u8Mode, int nAxisPerSlv) 函数设置驱动设备的运行模式。在有多个从站设备连接时该该函数能够一次配置多个设备的运行模式，参数 nSlvBegin 为开始的从站号，参数 nSlvEnd 为结束的从站号，参数 u8Mode 为想要配置的模式，参数 nAxisPerSlv 为每个从站设备搭配几个轴。

11) 调用 int ECM_StateCheck(int u8Slave,int u8ExpectState,int TimeOutMS) 函数配置从站进入 SAFE-OP 状态。参数 u8Slave 为从站编号，从站号码根据网口连接顺序从 0 递增；参数 u8ExpectState 为请求进入的状态，参数 TimeOutMs 为请求超时时间，单位为毫秒。

12) 调用 int ECM_StateCheck(int u8Slave,int u8ExpectState,int TimeOutMS) 函数配置从站进入 OP 状态。参数 u8Slave 为从站编号，从站号码根据网口连接顺序从 0 递增；参数 u8ExpectState 为请求进入的状态，参数 TimeOutMs 为请求超时时间，单位为毫秒。

13) 调用 int ECM_EnableFIFO(int u8Enable) 函数使能命令缓存器工作，参数 u8Enable 为 1 时为启动，为 0 时为禁止。

14) 调用 int ECM_CheckDCStable() 函数检查设备是否成功进入时钟同步，当该函数返回 1 时表示成功进入 DC 同步，能够进行周期性的过程数据交换。

4.2.2 运动控制系统配置说明

RTE3128 的运动控制通过调用 XMCL.dl 内的函数实现。XMCL 运动控制函数库提供多轴多群组的运动控制功能。使用者可以通过配置群组的方式，规划机台实际马达轴与空间坐标关系，各群组为各自独立的运动坐标系统，使用者可以依照实际需求配置。

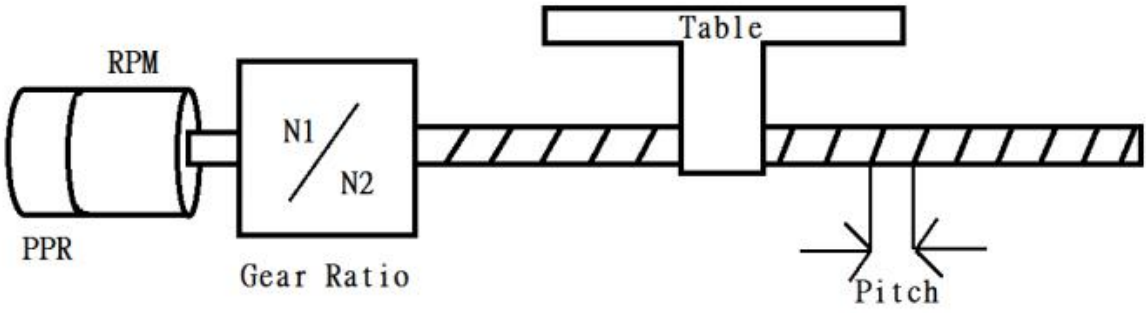
XMLC 运动控制函数库的主要功能如下：

- 系统设定：插值时间、设备的运动参数
- 速度规划：加减速时间、加减速类型、运行速度
- 空间运动：直线、圆弧、圆

- 点对点运动
- 运动状态控制：暂停、继续、丢弃
- 其他功能：轨迹平滑、在线变速、IO 控制等

XMCL 函数中部分参数设置与功能说明：

- 运动轴(Axis)：XMCL 采用直角坐标系统，运动命令的位置参数前三轴为空间轴(X 轴、Y 轴、Z 轴)，连接的其余设备为辅助轴(A0、A1、A2...)。
- 通道(CH)：在运动函数库内将连接的实际装置（如：驱动器、IO...）等定义为通道（CH）。
- 群组(Group)：运动函数库的运动命令以群组为单位，各群组可以视为独立的坐标系统。使用运动命令前需要先建立群组，配置运动轴的数量及对应的通道位置。建立群组后，可以使用运动命令指定群组进行多轴空间运动。
- 插值时间：插值时间以秒为单位，可设置范围为 0.005(0.5ms)至 0.1(100ms)。插值时间为 XMCL 内部计算最小时间单位，插值时间越小规划出来的运动轨迹越密，但是耗费的系统资源也就越多。请使用者依据实际状况及需求设置适合的插值时间。
- 一转需要的脉冲数(PPR)：马达转一圈，需要多少脉冲（位移量），用户可以根据驱动器设备的参数进行设定。
- 导螺杆螺距(Pitch)：导螺杆每转一圈，平台前进多少距离（mm），需要根据实际马达直径进行设置，
- 此轮比(GearRatio)：导螺杆转一圈，马达需要转几圈。
- 每分钟转速(RPM)：马达额定转速，根据驱动器参数配置。



- 加速时间(AccTime)：加速到运行速度需要的时间，依照马达及设备实际情况及能力设置适当值。
- 减速时间(DecTime)：减速到终止速度需要的时间，依照马达及设备实际情况及能力设置适当值。
- 运行速度(FeedSpeed)：马达正常运行时的速度，依照马达及设备实际情况及能力设置适当值。
- 加减速类型(Type)：提供 T 型及 S 型加减速类型，用户可根据需求进行设置。
- 空间运动：空间轴依照预先给定的速度规划进行直线、圆弧、圆周运动，辅助轴与空间轴同时开始同时结束。
- 运动状态控制：针对群组的运动状态进行控制，可以暂停目前运动命令、继续目前被暂停的运动命令、弃置目前运行的运动命令。

- 轨迹平滑：将两端运动命令进行平滑处理。优点是使速度变化平缓，减少抖动，缩短运动时间。缺点为两个运动命令的轨迹交接处失真。
- 在线变速：改变运动中命令的运行速度，不影响运动轨迹。
- IO 控制：可以通过 IO 控制从站设备。

函数配置过程如下：

1）在设备处于 OP 模式并成功 DC 同步后，调用 int XF_InitSystem(double dfCyctime, byte nChNum, MACH_PARAM[] pMachTbl) 函数初始化运动控制系统。其中参数 dfCyctime 为插值时间，插值时间以秒为单位，可设置范围为 0.005(0.5ms)至 0.1(100ms)；参数 ChNum 为通道数量；参数 pMachTbl 为设备的相关参数信息，其中结构体 MACH_PARAM 的定义如下：

```
typedef struct _MACH_PARAM_ {
    double  dfTa; //加速时间
    double  dfTd; //减速时间
    double  dfGearRatio; //齿轮比，默认 1
    double  dfPitch; //电机转一圈平台前进的距离，默认 10mm
    int nPPR; //电机转一圈需要的脉冲数，根据驱动器参数设置
    int nRPM; //电机的额定转速，根据驱动器参数配置
} MACH_PARAM;
```

2）调用 int XF_CreateGroup(int nAxisNum, int[] pcAxisToCh) 函数创建群组号。参数 nAxisNum 为当前创建的群组的轴数量，参数 pcAxisToCh 为类型为 int 表示运动轴与从站编号的对应关系的数组。该函数返回成功创建的群组号。

3）调用 XF_SetGroupMaxSpeed、XF_SetGroupAccTime、XF_SetGroupDecTime、XF_SetGroupAccType、XF_SetGroupDecType 等函数能够设置该群组的相关运动参数。

4）调用 int XF_CoE_Servo(int nCh, int nEnable) 函数使能从站设备。参数 nCh 为从站号，nEnable 为 1 时表示使能，为 0 时表示禁止。

5）调用相关运动函数即可成功启动运动。

主站初始化函数库

int UserSys_Init(int u32SpiFreq)	
用户系统初始化	
参数	说明
u32spiFreq	SPI 通讯频率
返回值	负值为错误发生

void UserSys_Close(void)	
用户系统关闭	
参数	说明
无	
返回值	负值为错误发生

int UserDelay(int MicroSec)	
用户延时函数	
参数	说明
MicroSec	延迟时间，单位微妙(us)
返回值	负值为错误发生

int UserSpiDataExchange(int *pTxBuf,int *pRxBuf,int u32PackSize);	
用户串行接口数据交换	
参数	说明
pTxBuf	发送缓冲区指针
pRxBuf	接受缓冲区指针
U32PackSize	数据交换的数据包大小

返回值	负值为错误发生

int ECM_Drv402SM_AdvConfig(int TblIdx,int SlvIdx, int ContrlWordOffset, int StateWordOffset);	
指定从站控制字及状态字偏移量	
参数	说明
TblIdx	从站轴个数
SlvIdx	从站编号
ContrlWordOffset	控制字偏移
StateWordOffset	状态字偏移
返回值	负值为错误发生

int ECM_Drv402SM_Enable(uint8_t TblIdx, uint8_t SlvIdx);	
Canopen402 标准配置寄存器启用	
参数	说明
TblIdx	从站轴个数
SlvIdx	从站编号
返回值	负值为错误发生

int ECM_Drv402SM_StateCheck(int SlvIdx, int ExpectState, int TimeOutMS);	
驱动器 Canopen402 标准状态检查并切换。	
参数	说明
Slvidx	从站编号

ExpectState	检查状态(SERVO_ON_STATE 0x27,SERVO_OFF_STATE 0x21)
TimeOutMS	超时时间(毫秒)
返回值	负值为错误发生

int ECM_Drv402SM_StateSet(int SlvIdx, int State);	
切换指定从站 402 标准状态	
参数	说明
SlvIdx	从站编号
State	需要设置的状态
返回值	负值为错误发生

int ECM_Drv402SM_StateGet(int SlvIdx, int *pState);	
获取指定从站当前 402 标准状态	
参数	说明
SlvIdx	从站编号
pState	用来接收状态的 int 指针
返回值	负值为错误发生

int ECM_EcatInit(uint16_t DCActCode, uint32_t CycTime);	
初始化 EtherCAT 主站及从站，使用 EtherCAT 功能前必须先初始化 EtherCAT 网络。	
参数	说明
DCActCode	DC 同步代码：0 为禁止 DC 同步，0x300 为同步 Sync0，

	0x700 同时激活 Sync0、Sync1
CycTime	控制模块周期循环时间
返回值	负值为错误发生

int ECM_EcatPdoConfigGet(PDO_CONFIG_HEAD *pBuf);	
从站设备 PDO 过程数据配置获取	
参数	说明

pBuf	PDO_CONFIG_HEAD 结构指针
	`typedef struct ECM_PACK pdo_config_t`
	`{`
	`  uint32_t  VID;`
	`  uint32_t  PID;`
	`  uint8_t  Slave;`
	`  uint8_t  PDOCnt;`
	`  uint16_t  SmaIdx;`
	`  uint16_t  MapIdx[4];`
	`  uint16_t  ObjsCnt[4];`
	`  OBJ_ENTRY_T Table[32];`
	`} PDO_CONFIG_HEAD;`

	Slave：从站编号
	PDOCnt：需要获取的 PDO 数量
	SmaIdx：寄存器索引
	MapIdx：映射对象索引
	ObjsCnt：映射对象数量
	Table：映射对象数据
返回值	负值为错误发生

int ECM_EcatPdoConfigReq(int Slave, int SmaIdx);

PDO 配置请求	
参数	说明
Slave	从站编号
Smaidx	请求配置的寄存器对象索引
返回值	负值为错误发生

int ECM_EcatPdoConfigSet(int Slave, PDO_CONFIG_HEAD *pConfigData);	
PDO 配置设置	
参数	说明
Slave	从站编号

pConfigData	PDO_CONFIG_HEAD 结构指针
	`typedef struct ECM_PACK pdo_config_t`
	`{`
	`uint32_t VID;`
	`uint32_t PID;`
	`uint8_t Slave;`
	`uint8_t PDOCnt;`
	`uint16_t SmaIdx;`
	`uint16_t MapIdx[4];`
	`uint16_t ObjsCnt[4];`
	`OBJ_ENTRY_T Table[32];`
	`} PDO_CONFIG_HEAD;`

	Slave：从站编号
	PDOCnt：需要获取的 PDO 数量
	SmaIdx：寄存器索引
	MapIdx：映射索引
	ObjsCnt：映射对象数量
	Table：映射对象数据
返回值	负值为错误发生

int ECM_EcatPdoDataExchange(int u8OP, uint8_t *pRxData, uint8_t *pTxData, uint16_t *pu16DataSize);	
没有经过命令缓存区的 PDO 数据交换	
参数	说明
u8OP	操作模式，write = 5；read = 6；writeRead = 7。
pRxData	指向 RxPdo
pTxData	指向 TxPdo
pu16DataSize	指向存储要交换数据的大小的内存的指针。
返回值	负值为错误发生

int ECM_EcatPdoFifoDataExchange(uint8_t u8FifoThreshold, uint8_t *pRxData, uint8_t *pTxData, int u16DataSize, uint8_t *pu8RxPdoFifoCnt, uint8_t *CrcErrCnt, uint8_t *WkcErrCnt);	
通过命令缓存区周期交换 PDO 数据	
参数	说明
u8FifoThreshold	数据缓存器的最大值
pRxData	指向 RxPdo
pTxData	指向 TxPdo
pu16DataSize	要交换数据的大小
Pu8RxPdoFifoCnt	pdo 接收数据的数据缓存器个数指针
CrcErrCnt	返回校验错误计数
WkcErrCnt	返回子报文错误处理情况计数
返回值	负值为错误发生

int ECM_EcatPdoFifoIsFull(uint8_t u8FifoThreshold);	
用来判断命令缓存区是否达到参数的个数	
参数	说明
u8FifoThreshold	数据缓存器的最大值
返回值	负值为错误发生，已达到返回 1，未达到返回 0

int ECM_EcatReconfig();	
当重新规划 PDO 后，需使用此命令使芯片重新配置内存空间	
参数	说明
无	
返回值	负值为错误发生 0：超时 1：得到有效返回 -1：得到有效返回但是命令响应错误 -2：等待异步结束错误 -3：参数错误

int ECM_EcatSdoGet(uint8_t *pBuf);	
Ethercat 从站 sdo 数据获取	
参数	说明
pBuf	用来接收获取的数据的指针
返回值	负值为错误发生

int ECM_EcatSdoReq(uint8_t OP, uint8_t Slave, uint16_t Index, uint8_t SubIndex, uint16_t size, int Timeout, uint8_t *Data);	
Ethercat 从站 sdo 请求	
参数	说明
OP	读写模式 = 0 读模式 = 1
Slave	从站编号
Index	sdo 请求对象的索引
SubIndex	sdo 请求对象的子索引
Size	sdo 请求对象的数据大小
Timeout	请求的超时时间，微秒(us)
Data	存放写入的数据或者读取到的数据的指针
返回值	负值为错误发生

int8_t ECM_EcatSlvCntGet();	
获取连接从站的数量	
参数	说明
无	
返回值	获取到的数量 负值为错误发生

int ECM_EcatStateCheck(uint8_t u8Slave, uint8_t u8State);
Ethercat 从站状态检查并设置

参数	说明
u8Slave	从站编号
u8State	需要进入的状态 EC_STATE_NONE 0x00 EC_STATE_INIT 0x01 EC_STATE_PRE_OP 0x02 EC_STATE_BOOT 0x03 EC_STATE_SAFE_OP 0x04 EC_STATE_OPERATIONAL 0x08 EC_STATE_ACK 0x10 EC_STATE_ERROR 0x10
返回值	负值为错误发生

int ECM_EcatStateGet(uint8_t u8Slave, uint8_t *pu8State);	
Ethercat 从站状态获取	
参数	说明
u8Slave	从站编号
pu8State	存放获取数据的指针
返回值	负值为错误发生

int ECM_EcatStateUpdate();	
Ethercat 从站状态更新	
参数	说明
无	
返回值	负值为错误发生

int ECM_EcatStateSet(uint8_t u8Slave, uint8_t u8State);	
Ethercat 从站状态设置	
参数	说明
u8Slave	从站编号
u8State	需要设置的状态 EC_STATE_NONE 0x00 EC_STATE_INIT 0x01 EC_STATE_PRE_OP 0x02 EC_STATE_BOOT 0x03 EC_STATE_SAFE_OP 0x04 EC_STATE_OPERATIONAL 0x08 EC_STATE_ACK 0x10 EC_STATE_ERROR 0x10
返回值	负值为错误发生

int ECM_FifoRxPdoSizeGet();	
获取 RxPdo 资料长度，FIFO 内一笔资料为一个 PDO	
参数	说明
无	
返回值	成功为返回获取到的大小 负值为错误发生

int ECM_FifoTxPdoSizeGet();
获取 TxPdo 资料长度，FIFO 内一笔资料为一个 PDO

参数	说明
无	
返回值	成功为返回获取到的大小 负值为错误发生

int ECM_GetFirmwareVersion(uint8_t *pVersion, uint8_t *pExtVer);	
获取硬件版本	
参数	说明
pVersion	用来存放获取到数据的指针
pExtVer	用来存放获取到版本信息的指针
返回值	负值为错误发生

int ECM_InfoUpdate(uint8_t *pEcmStatus, uint8_t *pRxPDOFifoCnt, uint8_t *CrcErrCnt, uint8_t *WkcErrCnt);	
从站的信息更新	
参数	说明
pEcmStatus	状态指针
pRxPDOFifoCnt	存放 PDO 接收命令缓存区个数的指针
CrcErrCnt	存放校验错误数量的指针
WkcErrCnt	存放从站子报文错误处理情况数量的指针
返回值	负值为错误发生

int ECM_ShowPDOConfig(int Slave, int SmaIdx);	
打印从站的 PDO 配置	
参数	说明
Slave	从站编号
SmaIdx	需要打印的寄存器映射对象索引
返回值	负值为错误发生

int ECM_SlaveInfoGet(uint8_t slave, uint8_t info, uint8_t *pBuf);	
获取从站驱动器信息	
参数	说明
Slave	从站编号
Info	需要获取的信息
	ECM_SLV_INFO_vid = 0
	ECM_SLV_INFO_pid = 1
	ECM_SLV_INFO_rev = 2
	ECM_SLV_INFO_name = 3
	ECM_SLV_INFO_configadr = 4
	ECM_SLV_INFO_aliasadr = 5
	ECM_SLV_INFO_state = 6
	ECM_SLV_INFO_ALstatuscode = 7
	ECM_SLV_INFO_Obytes = 8
	ECM_SLV_INFO_Ibytes = 9
	ECM_SLV_INFO_hasdc = 10
	ECM_SLV_INFO_DCcycle = 11

	ECM_SLV_INFO_DCshift = 12
	ECM_SLV_INFO_DCactive = 13
pBuf	获取到信息的存放地址
返回值	负值为错误发生

int ECM_StateCheck(uint8_t u8Slave, uint8_t u8ExpectState, int TimeOutMS);	
从站状态检查。函数内部实现使用 ECM_EcatStateSet、ECM_EcatStateCheck 和 ECM_EcatStateGet 实现	
参数	说明
u8Slave	从站编号
u8ExpectState	需要设置或检查的状态
TimeOutMS	超时时间，单位毫秒(ms)
返回值	负值为错误发生

int ECM_WaitAsyncDone(int nMS);	
等待异步完成	
参数	说明
nMS	等待时间，单位毫秒(ms)
返回值	负值为错误发生

unsigned char *ECM_GetTxBuf();	
获取发送缓存区指针	
参数	说明

无	
返回值	成功则返回地址

unsigned char *ECM_GetRxBuf();	
获取接收缓存区指针	
参数	说明
无	
返回值	成功则返回地址

unsigned char *ECM_GetTxPdoBuf();	
获取指向 TxPdo 的指针	
参数	说明
无	
返回值	成功则返回地址

unsigned char *ECM_GetRxPdoBuf();	
获取指向 RxPdo 的指针	
参数	说明
无	
返回值	成功则返回地址

int ECM_IsAsyncBusy(void);	
检测异步是否繁忙	
参数	说明
无	
返回值	繁忙返回 1，否则返回 0

int ECM_CheckDCStable(void);	
检测是否 DC 同步成功	
参数	说明
无	
返回值	成功返回 1，失败返回其他

int ECM_InitLibrary(usigned short *pu16SpiDataSize);	
设置 SPi 数据大小并返回 IC 固件版本	
参数	说明
pu16SpiDataSize	SPI 数据大小，如果此参数为 0 表示系统将使用默认设置为 112Bytes
返回值	返回 IC 固件版本

int ECM_EcatSdoWrite(uint8_t Slave，uint16_t Index，uint8_t SubIndex，uint16_t size，int32_t Timeout，uint8_t *Data);
SDO 数据写

参数	说明
Slave	从站号
Index	索引号
SubIndex	子索引号
Size	数据大小
TimeOut	超时时间
Data	需要写的数据
返回值	负值为错误发生

int ECM_EcatSdoRead(uint8_t Slave, uint16_t Index, uint8_t SubIndex, uint16_t size, int32_t Timeout, uint8_t *Data);	
SDO 数据读	
参数	说明
Slave	从站号
Index	索引号
SubIndex	子索引号
Size	数据大小
TimeOut	超时时间
Data	读上来的数据存放在这
返回值	负值为错误发生

int ECM_GetPDOConfig(int32_t Slave, int32_t SmaIdx, PDO_CONFIG_HEAD *pPdoConfigBuf);	
获取从站的 PDO 配置	
参数	说明

Slave	从站号
SmaIdx	PDO 索引
pPdoConfigBuf	获取到的从站 PDO 配置 typedef struct ECM_PACK pdo_config_t { uint32_t VID; uint32_t PID; uint8_t Slave; uint8_t PDOCnt; uint16_t SmaIdx; uint16_t MapIdx[4]; uint16_t ObjsCnt[4]; OBJ_ENTRY_T Table[32]; } PDO_CONFIG_HEAD;
返回值	负值为错误发生

int ECM_ShowPDOConfig(int32_t Slave,int32_t Smaidx);	
在控制台打印输出从站 PDO 配置	
参数	说明
Slave	从站号
SmaIdx	从站包含的轴数
返回值	负值为错误发生

int ECM_SetTxFIFOCnt(uint8_t u8TxCnt);	
设置命令缓存区 Tx 命令个数	
参数	说明
u8TxCnt	需要设置的个数
返回值	负值为错误发生

int ECM_SetRxFIFOCnt(uint8_t u8RxCnt);	
设置命令缓存区 Rx 命令个数	
参数	说明
u8RxCnt	需要设置的个数
返回值	负值为错误发生

int ECM_GetTxFIFOCnt(uint8_t *pu8Cnt);	
获取命令缓存区 Tx 命令个数	
参数	说明
pu8Cnt	获取到的数存放在这
返回值	负值为错误发生

int ECM_GetRxFIFOCnt(uint8_t *pu8Cnt);	
获取命令缓存区 Rx 命令个数	
参数	说明
pu8Cnt	获取到的数存放在这
返回值	负值为错误发生

int ECM_ClearFIFO(uint8_t u8TxRx);
清理 FIFO

参数	说明
u8TxRx	清理的数量
返回值	负值为错误发生

int ECM_EnableFIFO(uint8_t u8Enable);	
启动 RxPDO FIFO 输出，ECM-XF 默认启动 RxPDO FIFO 输出，可以通过此命令关闭 RxPDO FIFO 周期输出，可用于预储存命令至 FIFO。	
参数	说明
u8Enable	1 为启动，0 为禁止
返回值	负值为错误发生

int ECM_InitFIFO(void);	
初始化 FIFO,配置空间	
参数	说明
无	
返回值	负值为错误发生

int ECM_SetSpiDataSize(uint16_t u16SpiDataSize);	
设置 SPI 数据大小	
参数	说明
u16SpiDataSize	需要设置的大小
返回值	负值为错误发生

int ECM_GetSpiDataSize(uint16_t *pu16SpiDataSize);	
获取 SPI 数据大小	
参数	说明
pu16SpiDataSize	获取到的数据大小存放在这
返回值	负值为错误发生

int ECM_EcatWkcErrorMaxSet(uint8_t u8Max);	
设置工作计数器错误最大值	
参数	说明
u8Max	需要设置的参数
返回值	负值为错误发生

int ECM_EcatWkcErrorMaxGet(uint8_t *pu8Max);	
获取工作计数器错误最大值	
参数	说明
pu8Max	获取到的数存放在这
返回值	负值为错误发生

int ECM_EcatSdoSetPdoConfig(PDO_CONFIG_HEAD *pConfigData);	
通过 Sdo 配置 PDO	
参数	说明

pConfigData	需要配置的 PDO 参数
	`typedef struct ECM_PACK pdo_config_t`
	`{`
	`uint32_t VID;`
	`uint32_t PID;`
	`uint8_t Slave;`
	`uint8_t PDOCnt;`
	`uint16_t SmaIdx;`
	`uint16_t MapIdx[4];`
	`uint16_t ObjsCnt[4];`
	`OBJ_ENTRY_T Table[32];`
	`} PDO_CONFIG_HEAD;`

返回值	负值为错误发生

运动控制库函数

int XF_InitSystem (double dfCyctime, int nChNum, MACH_PARAM *pMachTbl);	
初始化运动控制系统，设置插值时间和运动参数	
参数	说明
dfCyctime	插值时间，单位秒(s)，有效范围 0.0005-0.1(0.5ms-100ms)
nChNum	实际连接通道数量

pMachTbl	类型为 MACH_PARAM 结构，数量为 nChNum 的一维结构数组。
	`struct MACH_PARAM`
	`{`
	`public double dfTa; //加速时间`
	`public double dfTd; //减速时间`
	`public double dfGerarRatio; //齿轮比，默认1`
	`public double dfPitch; //电机转一圈，平台前进多少距离，默认10`
	`public int nPPR; //电机转一圈需要的脉冲数`
	`public int nRPM; //电机的额定转速`
	`};`

返回值	初始化成功返回 1，负值为错误发生 -3 nChNum>64 或 nChNum<=0 -2 已经初始化过

	-1 无法建立周期性过程通信

int XF_CloseSystem ();	
关闭运动控制系统	
参数	说明
无	
返回值	负值为错误发生

int XF_ClearAlarm (int Slave);	
清除驱动器的警告	
参数	说明
Slave	需要清除的从站编号
返回值	负值为错误发生

int XF_ClearENC(int Slave);	
清除编码器计数	
参数	说明
Slave	需要清除的从站编号
返回值	负值为错误发生

Int XF_CloseAllGroup();	
关闭所有已创建的群组	
参数	说明
无	
返回值	负值为错误发生

Int XF_CloseGroup(int group);	
关闭指定群组号的群组	
参数	说明
group	群组号
返回值	负值为错误发生

int XF_CoE_Servo (int slave, int nEnable);	
驱动器进入使能状态或驱动器去使能	
参数	说明
slave	需要操作的从站编号
nEnable	1 进入使能状态 0 离开使能状态
返回值	负值为错误发生 -1 为无从站连接

int XF_CreateGroup (int nAxisNum, int *pcAxisToCh);	
创建群组	
参数	说明
nAxisNum	群组的有效运动轴数量
PcAxisToCh	型别为 int，数量为 nAxisNum 的一维数组，用来设置运动轴与从站编号的对应关系。
返回值	返回群组号索引 负值为错误发生 -1 储存区空间不足 -2 已建立的群组已达最大值 -3 指定的从站编号其类型是 IO -4 建立互斥对象发生错误

int XF_DisableBlend (int nGroup);	
关闭轨迹平滑功能	
参数	说明
nGroup	群组号
返回值	负值为错误发生 -1 无此群组号索引 -2 该群组已关闭轨迹平滑功能 -3 群组号错误

int XF_EnableBlend (int nGroup);	
开启轨迹平滑功能	
参数	说明
nGroup	群组号
返回值	负值为错误发生 -1 无此群组号索引 -3 群组号错误

int XF_SetBlendTime (int nGroup,double dfTb);	
设置轨迹平滑时间	
参数	说明
nGroup	群组号
dfTb	轨迹平滑时间
返回值	负值为错误发生 -1 无此群组号索引 -3 群组号错误

int XF_GetActualPositionOffset(int Slave);	
获取从站实际位置偏移量	
参数	说明
Slave	需要获取的从站编号

返回值	返回偏移量 负值为错误发生

int XF_GetAcualPos(int Slave);	
获取从站的当前位置	
参数	说明
Slave	需要获取的从站编号
返回值	返回当前位置

int XF_GetCtrlWordOffset(int Slave);	
获取从站控制字的偏移量	
参数	说明
Slave	需要获取的从站编号
返回值	返回偏移量 负值为错误发生

int XF_GetENC(int nAxis);	
获取编码器位置信息	
参数	说明
nAxis	需要获取的轴号
返回值	返回当前轴的编码器位置信息 负值为错误发生

int XF_GetMotionStatus (int nGroup);	
获取指定群组的运动状态	
参数	说明
nGroup	需要获取的轴号
返回值	0：运动停止 1：运动进行中 2：运动结束 3：运动暂停 4：运动继续 5：运动放弃 负值为错误发生 -1 无此群组号 -2 运动状态未知 -3 群组号索引错误

int XF_GetStatusWordOffset(int Slave);	
获取从站状态字的偏移量	
参数	说明
Slave	需要获取的从站编号
返回值	返回偏移量 负值为错误发生

int XF_GetTargetPositionOffset(int Slave);	
获取从站目标位置偏移量	
参数	说明
Slave	需要获取的从站编号
返回值	返回偏移量 负值为错误发生

int XF_MotionAbort(int nGroup, int nTd, int nDecType);	
丢弃指定群组的目前运动	
参数	说明
nGroup	运动的群组号
nTd	减速时间
nDecType	减速类型 0：T 型减速 1：S 型减速
dfVe	终止速度
返回值	负值为错误发生 -1 无此群组号 -3 群组号索引错误

int XF_MotionConti (int nGroup);

运动继续	
参数	说明
nGroup	运动的群组号
返回值	负值为错误发生 -1 无此群组号 -3 群组号索引错误

int XF_MotionPause (int nGroup，int nTd, int nDecType, double dfVe);	
运动暂停	
参数	说明
nGroup	运动的群组号
nTd	减速时间
nDecType	减速类型 0：T 型减速 1：S 型减速
dfVe	终止速度
返回值	负值为错误发生 -1 无此群组号 -3 群组号索引错误

int XF_OverwriteSpeed (int nGroup, double Vnew);
强制覆写当前运动群组的最大速度

参数	说明
nGroup	运动的群组号
Vnew	覆写的速度
返回值	0 为正常，负值为错误发生 -1 无此群组号 -3 群组号索引错误

int XF_SetActualPositionOffset(int slave, int offset);	
设置实际位置的偏移量	
参数	说明
Slave	从站编号
offset	偏移量参数
返回值	负值为错误发生

int XF_SetAxisAccDecType(int nGroup, int nAxis, int nAccDecType);	
设置轴的加速类型和减速类型，点对点运动有效	
参数	说明
nGroup	群组号
nAxis	轴号
nAccDecType	类型 0 为 T 型，1 为 S 型
返回值	负值为错误发生 -1 无此群组号索引 -3 群组号错误

int XF_SetAxisAccTime (int nGroup, int nAxis, double dfTa);	
设置轴加速时间，点对点运动有效	
参数	说明
nGroup	群组号
nAxis	轴号
dfTa	加速时间
返回值	负值为错误发生 -1 系统未初始化或无此群组索引号 -3 群组号错误

int XF_SetAxisAccType (int nGroup, int nAxis, int nAccType);	
设置轴的加速类型，点对点运动有效	
参数	说明
nGroup	群组号
nAxis	轴号
nAccType	类型 0 为 T 型，1 为 S 型
返回值	负值为错误发生 -1 系统未初始化或无此群组索引号 -3 群组号错误

int XF_SetAxisBeginVelocity(int nGroup, int nAxis, double dfV);
设置单轴开始时速度，点对点运动有效

参数	说明
nGroup	群组号
nAxis	轴号
dfV	速度
返回值	负值为错误发生 -1 系统未初始化或无此群组索引号 -3 群组号错误

int XF_SetAxisDecTime (int nGroup, int nAxis, double dfTd);	
设置单轴减速时间，点对点运动有效	
参数	说明
nGroup	群组号
nAxis	轴号
dfTd	减速时间
返回值	负值为错误发生 -1 系统未初始化或无此群组索引号 -3 群组号错误

int XF_SetAxisDecType (int nGroup, int nAxis, int nDecType);
设置单轴减速类型

参数	说明
nGroup	群组号
nAxis	轴号
nDecType	减速类型 0 为 T 型，1 为 S 型
返回值	负值为错误发生 -1 系统未初始化或无此群组索引号 -3 群组号错误

int XF_SetAxisEndVelocity(int nGroup, int nAxis, double dfV);	
设置单轴结束时速度，点对点运动有效	
参数	说明
nGroup	群组号
nAxis	轴号
dfV	速度
返回值	负值为错误发生 -1 系统未初始化或无此群组索引号 -3 群组号错误

int XF_SetAxisVelocity (int nGroup, int nAxis, double dfV);	
设置运动轴的最大速度，点对点运动有效	
参数	说明
nGroup	群组号
nAxis	轴号

dfV	速度
返回值	负值为错误发生 -1 系统未初始化或无此群组索引号 -3 群组号错误

int XF_SetCtrlWordOffset(int Slave,int offset);	
设置控制字偏移量	
参数	说明
Slave	从站编号
offset	偏移量
返回值	负值为错误发生

int XF_SetGroupAccDecType(int nGroup, int nAccDecType);	
设置群组加速类型和减速类型	
参数	说明
nGroup	群组号
nAccDecType	类型，0 为 T 型，1 为 S 型
返回值	负值为错误发生 -1 系统未初始化或无此群组索引号 -3 群组号错误

int XF_SetGroupAccTime (int nGroup, double dfTa);	
设置群组加速时间，单位 Sec	
参数	说明
nGroup	群组号
dfTa	加速时间
返回值	负值为错误发生 -1 系统未初始化或无此群组索引号 -3 群组号错误

int XF_SetGroupAccType (int nGroup, int nAccType);	
设置群组加速类型	
参数	说明
nGroup	群组号
nAccType	加速类型，0 为 T 型，1 为 S 型
返回值	负值为错误发生 -1 系统未初始化或无此群组索引号 -3 群组号错误

int XF_SetGroupBeginSpeed(int nGroup, double dfV);	
设置群组开始速度	
参数	说明
nGroup	群组号

dfV	速度
返回值	负值为错误发生 -1 系统未初始化或无此群组索引号 -3 群组号错误

int XF_SetGroupDecTime (int nGroup, double dfTd);	
设置群组减速时间	
参数	说明
nGroup	群组号
dfTd	减速时间
返回值	负值为错误发生 -1 系统未初始化或无此群组索引号 -3 群组号错误

int XF_SetGroupDecType (int nGroup, int nDecType);	
设置群组减速类型	
参数	说明
nGroup	群组号
nDecType	减速类型，0 为 T 型，1 为 S 型
返回值	负值为错误发生 -1 系统未初始化或无此群组索引号 -3 群组号错误

int XF_SetGroupEndSpeed(int nGroup, double dfV);	
设置群组结束速度	
参数	说明
nGroup	群组号
dfV	速度
返回值	负值为错误发生 -1 系统未初始化或无此群组索引号 -3 群组号错误

int XF_SetGroupMaxSpeed (int nGroup, double dfVmax);	
设置群组最大运行速度，单位 mm/Sec	
参数	说明
nGroup	群组号
dfVmax	最大速度
返回值	负值为错误发生 -1 系统未初始化或无此群组索引号 -3 群组号错误

int XF_SetRxPDOSize(int Size);	
设置 pdo 接收区大小	
参数	说明
Size	大小

返回值	负值为错误发生

int XF_GetRxPDOSize(void);	
获取 pdo 接收区大小	
参数	说明
无	
返回值	获取到的大小

int XF_SetTxPDOSize(int Size);	
设置 pdo 发送区大小	
参数	说明
Size	大小
返回值	负值为错误发生

int XF_GetTxPDOSize(void);	
获取 pdo 发送区大小	
参数	说明
无	
返回值	获取到的大小

int XF_SetStatusWordOffset(int Slave,int offset);
设置状态字偏移量

参数	说明
Slave	从站编号
offset	偏移量
返回值	负值为错误发生

int XF_SetTargetPositionOffset(int Slave,int offset);	
设置目标位置偏移量	
参数	说明
Slave	从站编号
offset	偏移量
返回值	负值为错误发生

int Ptp(int nGroup, double *pdfTarPos, int nAxisNum);	
运动轴依照各次参数进行点对点运动	
参数	说明
nGroup	群组号
pdfTarPos	运动轴目标位置，型别为 double，数量为轴数的一维数组。
nAxisNum	运动轴数量
返回值	返回命令索引号 负值为错误发生 -1 无此群组索引号 -2nAxisNum 数量大于该群组轴数 -3 群组索引号错误 -4 无法建立此运动命令

	-11 空间不足

int Line_Aux(int nGroup, double *pdfTarPos, int nAxisNum);	
空间轴依照速度规划进行空间直线运动，辅助轴运动与空间轴运动为同时开始，同时结束。轴号(0-2)为空间轴，其他为辅助轴	
参数	说明
nGroup	群组号
pdfTarPos	运动轴目标位置，型别为 double，数量为 nAxisNum 的一维数组。
nAxisNum	运动轴数量
返回值	返回命令索引号 负值为错误发生 -1 无此群组索引号 -3 群组索引号错误 -4 无法建立此运动命令

int Arc_Aux (int nGroup, double *pdfRefPos, double *pdfTarPos, int nAxisNum);	
空间轴依照速度规划进行空间圆弧运动，圆弧由目前位置，经过参考位置，至目标位置结束。辅助轴运动与空间轴运动为同时开始，同时结束。	
参数	说明
nGroup	群组号
pdfRefPos	空间轴的圆弧参考位置，型别为 double，数量为 3 的一维数组

pdfTarPos	空间轴的圆弧目标位置，型别为 double，数量为参与运动轴数的一维数组
nAxisNum	运动轴数量，应大于等于 3
返回值	返回命令索引号 负值为错误发生 -1 无此群组索引号 -3 群组索引号错误 -4 无法建立此运动命令

int Circle_Aux (int nGroup, double *pdfCenter, double *pdfNVector, double dfTheta, double *pdfTarPos, int nAxisNum);	
空间轴依照速度规划进行空间圆运动，在法向量为 pdfNVector 的平面上，由目前位置再以 pdfCenter 为圆心，进行空间圆运动。辅助轴与空间轴运动同时开始，同时结束。	
参数	说明
nGroup	群组号
pdfCenter	空间圆的圆心位置，型别为 double，数量为 3 的一维数组。
pdfNVector	空间圆所在平面的法向量，型别为 double，数量为 3 的一维数组。
dfTheta	保留参数，可给任意值
pdfTarPos	运动轴的目标位置，前 3 轴目标位置为无效参数，可给任意值。型别为 double，数量为 nAxisNum 的一维数组。
nAxisNum	运动轴数量，大于等于 3
返回值	返回命令索引号 负值为错误发生 -1 无此群组索引号 -3 群组索引号错误

	-4 无法建立此运动命令

int XF_GetVersion();	
获取当前函数库版本	
参数	说明
无	
返回值	获取到的版本号

int XF_GetENC(int nCh);	
获取轴编码器位置	
参数	说明
nCh	轴号
返回值	获取到的位置

int XF_SetENC(int nCh, int32_t nValue);	
设置编码器值	
参数	说明
nCh	轴号
nValue	需要设置的值
返回值	负值为错误发生
int XF_SDO_Read(uint8_t Slave, uint16_t Index, uint8_t SubIndex, uint16_t size, int Timeout, uint8_t *Data);	
SDO 读	

参数	说明
Slave	从站编号
Index	需要读的索引号
SubIndex	需要读的子索引号
Size	数据大小
Timeout	超时时间
Data	读取到的数据存放在这
返回值	负值为错误发生

int XF_SDO_Write(uint8_t Slave, uint16_t Index, uint8_t SubIndex, uint16_t size, int Timeout, uint8_t *Data);	
SDO 写	
参数	说明
Slave	从站编号
Index	需要写的索引号
SubIndex	需要写的子索引号
Size	数据大小
Timeout	超时时间
Data	需要写入的数据存放在这
返回值	负值为错误发生